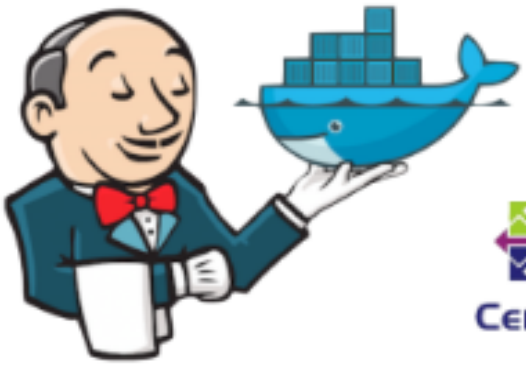
# Netkiller DevOps 手札

陈景峯 著



CentOS

# Netkiller DevOps 手札

目录


自述

1. 写给读者
2. 作者简介
3. 如何获得文档
4. 打赏（Donations）
5. 联系方式

I. 软件项目管理

1. 范围管理
    1. 宏观管理
    2. 你清楚你的工作职责吗?
        制度管理
        权力下放
        专业的人做专业的事
        总结
    3. 怎样防止踢皮球
        进入正题
        踢皮球几大害处
        场景一
        场景二
        踢皮球的风气是怎样形成的?
            技术人员的单一视角
            责任不明确
            缺乏沟通
            背黑锅
            员工问题
        怎样根治踢皮球
            调整组织架构
            禁止追查问题源头
            不懂技术的管理
            统一目标，价值观。

        防止问题扩大
    4. 内部外包与悬赏
        怎么样操作
        可能遇到的问题
        小结
    5. 团队膨胀的原因分析
        人才管理
            先从员工个人谈起
            技能管理
        再说说部门
            部门膨胀
        精细化管理带来的膨胀
2. 时间管理
    1. 优先级管理
    2. 时间管理的误区
        优化组织架构，精简机构
        命令决策一元化
        时间线
    3. 项目管理中工时计算的问题
        背景
        面临的问题
        工时去了哪里?
            洗手间，茶水间，吸烟
            看邮件，写邮件
            沟通
            查资料
            无关的会议
            不必要的拖延行为
            私人时间
        怎样改善面临的问题
        怎样计算项目工时?
    4. 项目延期
    5. 当日事当日毕
        工作时间如何规划?
        遇到工作被打断的情况
3. 沟通管理（Communication Management）

1. 表达方式
    如何提问
    拒绝反问和质问
    宽以律己，严以待人
    任务分配
    任务确认
2. 会议管理
    什么要开会?
    会议的时间成本
    集思广益纯属扯淡
    会议冲突
    避免议而不决
    会议记录
    会议地点
    与会人员
    怎样管理会议的时间呢?
3. 工作报告
    日报、周报，项目进度汇报
    为什么会出现频繁汇报?
    如何才能抛弃汇报制度?
4. 越级和跨部门沟通
5. 负面信息处理

4. 变更管理(Change Management)
    1. 什么是变更管理
    2. 需求变更
        为什么会变更
    3. 拥抱变更
5. 集成管理
    1. 配置管理
    2. 为什么持续集成难以普及
6. 质量管理
    1. 无缺点管理
    2. 自动化测试如何破局?
        认知问题
        生态的问题
        技术的问题

能力问题
氛围问题
最后
3. 为什么自动化测试难以推广
为什么自动化测试难以实施
是什么阻碍了自动化测试?
中国测试人员的人力成本
7. 风险管理
1. 项目管理绕不开问题
开发，测试与运维的关系
压力问题
重速度轻安全
技术实力
测试问题
运维问题
2. 程序猿说的「优化」是什么意思?
经验和能力不足
给前人擦屁股
先盖楼，后打地基
使用新技术和不熟悉的技术
最后总结
3. 制度、流程和规范的误区
故事一
故事二
故事三
故事四
故事五
总结
案例分析：怎样避免电梯伤人事件再发生
4. 因果图在运维工作中的应用
故障树分析(Fault Tree Analysis，FTA)
什么是因果图
为什么使用因果图
何时使用因果图
何处使用因果图
谁来负责制作因果图

- 怎样使用因果图
    - www.example.com, img.example.com
    - acc.example.com, api.example.com
    - cch.exampel.com, mq.exampe.com,
    - db.example.com
- 5. Incident Management(突发事件管理)
    - 突发事件处理流程
    - 事件处理方式
- 6. 监控的艺术
    - 背景
    - 概述
    - 怎样监控
        - 卫星监测
        - 逐级诊断
        - 模拟人工
        - 数据分析
        - 监控与开发
    - 总结

8. 成本管理
    1. 警惕IT黑洞
        - 什么是IT黑洞
        - IT黑洞产生的原因分析
            - 人的因素
            - 来自组织架构的问题

9. 人力资源管理
    1. 面试流程
    2. 技术面试
    3. 网络工程师面试题
        - Junior
        - Senior Network Engineer 高级网络工程师
    4. 运维工程师面试题
        - Junior
        - 高级运维工程师
    5. 软件工程师面试题
        - Junior Software Engineer
        - Senior Software Engineer

6. 架构师面试题
7. 数据库管理员
8. 测试工程师
10. 采购管理

II. 项目管理工具

11. Gitlab 项目管理

1. GitLab 安装与配置

1.1. Almalinux 9.0

Gitlab Runner

1.2. CentOS 8 Stream 安装 Gitlab

1.3. Docker 方式安装 Gitlab

Docker 运行

Docker compose 安装 Gitlab

Docker Compose 安装 gitlab-runner

1.4. Yum 安装 GitLab

GitLab Runner

1.5. 绑定SSL证书

1.6. Gitlab 管理

gitlab-rake 命令

gitlab-runner 命令

Gitlab 迁移，备份和恢复

备份

恢复

2. 初始化 Gitlab

2.1. 操作系统初始化

gitlab-runner

Docker

Java 环境 安装脚本

Node 环境

2.2. 创建用户

创建用户

2.3. 初始化组

2.4. 初始化标签

2.5. 初始化分支

2.6. 部署环境

3. 项目管理

- 3.1. 组织架构
  - 开发、测试和运维三个部门的关系
  - 权限角色
- 3.2. 项目计划
- 3.3. 工作流
- 3.4. 议题
  - Milestones 里程碑
  - 修正路线图（Roadmap）
  - 工作报告
  - 5W2H 任务分配法则
  - 任务/议题
    - 运维任务
    - 开发任务
    - 测试任务
    - 运营任务
- 3.5. 并行开发
  - 任务分解
  - 配套环境
  - 代码分支
    - 时间线分支
    - 分支权限
    - 功能分支
    - 合并流程
      - 合并分支
      - 除了单个文件
      - 合并分支解决冲突
    - Hotfix / BUG 分支
  - 前滚和后滚
    - 后滚操作
    - 前滚操作
    - 前滚后后滚常见操作
      - 导出最后一次修改过的文件
      - 导出指定版本区间修改过的文件
      - 回撤提交
      - 撤回单个文件提交
  - 提交代码怎样写注释信息

        Fixed Bug
        Implemented
        Add
    3.6. 升级与发布相关
        分支与版本的关系
        分支与标签的区别
        Release Notes
        License
    3.7. 代码审查

4. 通过GPG签名提交代码
    4.1. 创建证书
    4.2. 配置 Gitlab GPG
    4.3. 配置 Git
        全局配置
        本地配置
        提交代码
    4.4. FAQ
        error: gpg failed to sign the data
5. CI / CD
    5.1. 远程服务器配置
    5.2. 配置 CI / CD
        安装 GitLab Runner
        注册 gitlab-runner
        并发链接数设置
    5.3. Shell 执行器
        注册 Gitlab Runner 为 Shell 执行器
        生成 SSH 证书
        数据库环境
        Java 环境
            安装最新版 maven
            mvnd
        NodeJS
        Python 环境
        远程执行 sudo 提示密码
    5.4. tags 的使用方法

- 5.5. Docker 执行器
- 5.6. Kubernetes executor
  - 命名空间
  - 挂载卷
  - KUBERNETES_BEARER_TOKEN
  - 案例
- 5.7. Java 持续集成相关
  - 制作 Maven 镜像
  - JaCoCo
  - 并行开发解决版本冲突
- 5.8. 数据库结构监控
  - 什么是数据库结构版本控制
  - 为什么要做数据库结构本版控制
  - 何时做数据库结构本版控制
  - 在哪里做数据库结构本版控制
  - 谁来负责数据库结构本版控制
  - 怎样做数据库结构本版控制
    - 安装脚本
    - 启动脚本，停止脚本
    - 查看历史版本
  - CI/CD 配置
- 5.9. 持续部署 Nacos
  - nacos 持续部署工具
  - .gitlab-ci.yml 配置案例

6. Pipeline 流水线
   - 6.1. cache
     - Cache Key
     - 禁用 Cache
     - 定义多个缓存
   - 6.2. stages
     - 依赖关系
     - 禁用 stage
   - 6.3. variables
     - 列出所有环境变量
     - Git submodule
     - 通过条件，设置变量

- 6.4. script /before_script / after_script
  - 条件判断
  - 多行脚本
- 6.5. only and except
  - 匹配 feature / hotfix 分支
  - 监控文件变化
- 6.6. 构建物
  - 禁止 job 下载构建物
- 6.7. 允许失败
- 6.8. 定义何时开始job
- 6.9. services
- 6.10. tags
- 6.11. rules 规则
  - 条件判断
- 6.12. include 包含
- 6.13. 模版
- 6.14. release
- 6.15. 应用案例
  - Java
    - 使用 Docker 编译并收集构建物
    - Shell 执行器，远程部署物理机/虚拟机
    - Shell 执行器，远程部使用容器启动项目
    - Docker 执行器
  - Node
  - vue.js android
  - Python
  - docker
  - include 高级用法

7. 软件包与镜像库
   - 7.1. Maven 仓库
     - 将已存在的 JAR 文件部署到 Maven 仓库
   - 7.2. Python Pypi 仓库
     - 个人访问令牌
     - 手工上传包
     - 在持续集成中配置
   - 7.3. Node JS

        7.4. Docker registry
            配置 Docker registry
            手动构建镜像并上传至容器镜像库
            CI/CD 流水线配置
    8. 服务器端 hooks
        8.1. 创建全局 Server hooks
        8.2. 给单个仓库配置 Server hooks
            查看仓库目录
            创建 hooks 脚本
    9. 客户端 hooks
        9.1. 集成禅道
            Linux/MacOS
            Windows
            使用方法
    10. WebHook
    11. FAQ
        11.1. 查看日志
        11.2. debug runner
        11.3. gitolite 向 gitlab 迁移
        11.4. 修改主机名
        11.5. ERROR: Uploading artifacts as "archive" to coordinator... too large archive
        11.6. ERROR: Job failed (system failure): prepare environment: waiting for pod running: timed out waiting for pod to start. Check https://docs.gitlab.com/runner/shells/index.html#shell-profile-loading for more information
        11.7. 磁盘 100% 怎样清理
12. Jenkins
    1. 安装 Jenkins
        1.1. OSCM 一键安装
        1.2. Mac
        1.3. CentOS
        1.4. Ubuntu
        1.5. Docker
        1.6. Minikube

2. 配置 Jenkins
3. Jenkinsfile
    3.1. Jenkinsfile - Declarative Pipeline
        stages
        script
        junit
        withEnv
        parameters
        options
        triggers
        tools
        post
        when 条件判断
        抛出错误
        withCredentials
            token
        withMaven
        isUnix() 判断操作系统类型
        Jenkins pipeline 中使用 sshpass 实现 scp,
        ssh 远程运行
            后台运行
    3.2. Jenkinsfile - Scripted Pipeline
        git
        切换 JDK 版本
        groovy
        Groovy code
            Groovy 函数
        Ansi Color
        写文件操作
        modules 实现模块
        docker
        input
        if 条件判断
        Docker
        conditionalSteps
        nexus

- 3.3. 设置环境变量
  - 系统环境变量
- 3.4. agent
  - label
  - docker
    - 指定docker 镜像
    - args 参数
    - Docker outside of Docker (DooD)
    - 挂在宿主主机目录
    - 构建镜像
  - Dockerfile
- 3.5. Steps
  - parallel 平行执行
  - echo
  - catchError 捕获错误
  - 睡眠
  - 限制执行时间
  - 时间截
- 3.6. 版本控制
  - checkout
  - Git
- 3.7. 节点与过程
  - sh
  - Windows 批处理脚本
  - 分配工作空间
  - node
- 3.8. 工作区
  - 变更目录
  - 判断文件是否存在
  - 分配工作区
  - 清理工作区
  - 递归删除目录
  - 写文件
  - 读文件
4. Jenkins Job DSL / Plugin
5. Jenkins Plugin

    5.1. Blue Ocean
    5.2. Locale Plugin (国际化插件)
    5.3. github-plugin 插件
    5.4. Docker
        设置 Docker 主机和代理
        持久化
    5.5. JaCoCo
        Pipeline
    5.6. SSH Pipeline Steps
    5.7. Rancher
    5.8. Kubernetes 插件
        Kubernetes
        Kubernetes :: Pipeline :: Kubernetes Steps
        Kubernetes Continuous Deploy
        Kubernetes Cli
    5.9. HTTP Request Plugin
    5.10. Skip Certificate Check plugin
    5.11. Android Sign Plugin
  6. Jenkinsfile Pipeline Example
    6.1. Maven 子模块范例
    6.2. 使用指定镜像构建
    6.3. 命令行制作 Docker 镜像
    6.4. Yarn
    6.5. Android
13. SonarQube
  1. 安装
    1.1. Kubernetes 安装 SonarQube
    1.2. Docker
    1.3. netkiller-devops 安装
    1.4. SonarScanner
        Docker 安装
        本地安装
  2. 配置
    2.1. 登陆 SonarQube
    2.2. 本地 maven 执行 SonarQube
    2.3. 集成 Gitlab

        2.4. SonarScanner
            Node.js
    3. FAQ
        3.1. bootstrap check failure [1] of [1]: max virtual memory areas vm.max_map_count [65530] is too low, increase to at least [262144]
        3.2. failed: An API incompatibility was encountered while executing org.sonarsource.scanner.maven:sonar-maven-plugin:3.9.0.2155:sonar: java.lang.UnsupportedClassVersionError: org/sonar/batch/bootstrapper/EnvironmentInformation has been compiled by a more recent version of the Java Runtime (class file version 55.0), this version of the Java Runtime only recognizes class file versions up to 52.0
        3.3. [ERROR] An unknown compilation problem occurred
        3.4. can't have 2 modules with the following key
        3.5. Kubernetes 运行 sonar-scanner
14. Dagger
15. 持续集成工具
    1. Code Review
        1.1. Phabricator - an open source, software engineering platform
        1.2. Gerrit
        1.3. TeamCity
    2. Nexus Repository OSS
        2.1. 安装 Nexus
            Docker
        2.2. Nexus UI
        2.3. maven 设置
        2.4. Node.js
        2.5. Ruby
17. TRAC
    1. Ubuntu 安装

        1.1. source code
        1.2. easy_install
        1.3. Apache httpd
    2. CentOS 安装
        2.1. trac.ini
        2.2. standalone
        2.3. Using Authentication
        2.4. trac-admin
            Permissions
            Resync
    3. Project Environment
        3.1. Sqlite
        3.2. MySQL
        3.3. Plugin
            AccountManagerPlugin
            Subtickets
    4. trac.ini
        4.1. repository
        4.2. attachment 附件配置
    5. trac-admin
        5.1. adduser script
    6. Trac 项目管理
        6.1. Administration
            General
            Ticket System
            Version Control
        6.2. Wiki
        6.3. Timeline
        6.4. Roadmap
        6.5. Ticket
    7. FAQ
        7.1. TracError: Cannot load Python bindings for MySQL
    8. Apache Bloodhound
18. Redmine
    1. CentOS 安装

2. Redmine 运行
3. 插件
3.1. workflow
19. 项目管理工具
1. 禅道
2. TUTOS
III. 软件版本控制
20. Git - Fast Version Control System
1. Repositories 仓库管理
1.1. initial setup
1.2. 克隆代码
恢复文件
1.3. 切换分支
checkout master
checkout 分支
通过 checkout 找回丢失的文件
checkout 所有远程分支
使用 ours 与 theirs 解决冲突
使用远程分支强行覆盖本地分支
1.4. git-add - Add file contents to the index
1.5. Creating and Commiting
git-commit - Record changes to the repository
1.6. Status
git-status - Show the working tree status
1.7. Diff
--name-only 仅显示文件名
1.8. Push
1.9. Pull
1.10. fetch
1.11. Creating a Patch
1.12. reset
还原文件
2. 分支管理
2.1. 查看本地分支
2.2. 创建分支
2.3. 删除分支

    2.4. 切换分支
    2.5. 重命名分支
    2.6. git-show-branch - Show branches and their commits
3. git log
    3.1. hash-object
    3.2. 一行显示 --oneline
    3.3. 查看文件历史记录
    3.4. 格式化
4. reflog
5. 远程仓库
    5.1. 查看远程地址
        显示远程地址
    5.2. 添加远程仓库
    5.3. 修改 origin
    5.4. 删除 origin
    5.5. 仓库共享
        Setting up a git server
6. git show - Show various types of objects
    6.1. 查看指定版本的文件内容
7. 合并分支
    7.1. 合并分支
    7.2. rebase
    7.3. 合并分支解决冲突
    7.4. 终止合并
    7.5. 合并单个文件
    7.6. Git 合并特定 commits 到另一个分支
8. 比较文件
    8.1. 比较 SHA
    8.2. 分支比较
9. Submodule 子模块
    9.1. 添加模块
    9.2. checkout 子模块
    9.3. 删除子模块
10. Git Large File Storage
    10.1. 安装 LFS 支持

10.2. LFS lock
11. git config
11.1. git config
11.2. 查看配置
11.3. 编辑配置
11.4. 替换配置项
11.5. 配置默认分之
11.6. GPG签名
11.7. core.sshCommand
11.8. fatal: The remote end hung up unexpectedly
11.9. 忽略 SSL 检查
11.10. 配置忽略合并文件
11.11. .gitignore
11.12. .gitattributes
SVN Keywords
设置文件换行符
11.13. 配置模版目录
12. git-rev-parse - Pick out and massage parameters
12.1. 获得当前提交ID
13. git-daemon 服务器
13.1. git-daemon - A really simple server for git repositories
13.2. git-daemon-sysvinit
13.3. inet.conf / xinetd 方式启动
13.4. git-daemon-run
13.5. Testing
14. git-svn - Bidirectional operation between a single Subversion branch and git
15. Web Tools
15.1. viewgit
16. gitolite - SSH-based gatekeeper for git repositories
16.1. gitolite-admin
gitolite.conf
staff
repo
17. FAQ

17.1. 导出最后一次修改过的文件
17.2. 导出指定版本区间修改过的文件
17.3. 撤销当前修改，恢复到远程最后一次提交
17.4. 回撤提交
17.5. 撤回单个文件提交
17.6. 合并分支中的单个
17.7. 每个项目一个证书
17.8. fatal: Not possible to fast-forward, aborting.
17.9. receive.denyCurrentBranch
17.10. 更新所有项目以及分支
17.11. 找回丢失的分支
21. Subversion
1. Invoking the Server
1.1. Installing
Ubuntu
CentOS 5
classic Unix-like xinetd daemon
WebDav
项目目录结构
CentOS 6
1.2. standalone “daemon” process
starting subversion for debian/ubuntu
starting subversion daemon script for CentOS/Radhat
1.3. classic Unix-like inetd daemon
1.4. hooks
post-commit
1.5. WebDav
davfs2 - mount a WebDAV resource as a regular file system
2. repository 管理
2.1. create repository
2.2. user admin
2.3. authz
2.4. dump
3. 使用Subversion

        3.1. Initialized empty subversion repository for project
        3.2. ignore
        3.3. 关键字替换
        3.4. lock 加锁/ unlock 解锁
        3.5. import
        3.6. export 指定版本
        3.7. 修订版本关键字
        3.8. 恢复旧版本
    4. branch
        4.1. create
        4.2. remove
        4.3. switch
        4.4. merge
        4.5. relocate
    5. FAQ
        5.1. 递归添加文件
        5.2. 清除项目里的所有.svn目录
        5.3. color diff
        5.4. cvs2svn
        5.5. Macromedia Dreamweaver MX 2004 + WebDAV +Subversion
        5.6. 指定用户名与密码
22. cvs - Concurrent Versions System
    1. installation
        1.1. chroot
    2. cvs login l logout
    3. cvs import
    4. cvs checkout
    5. cvs update
    6. cvs add
    7. cvs status
    8. cvs commit
    9. cvs remove
    10. cvs log
    11. cvs annotate

12. cvs diff
13. rename file
14. revision
15. cvs export
16. cvs release
17. branch
    17.1. milestone
    17.2. patch branch
18. keywords

23. Miscellaneous
1. 代码托管
    1.1. sourceforge.net
        http://netkiller.users.sourceforge.net/ 页面
    1.2. Google Code
    1.3. GitHub
        首次操作
        clone 已经存在的仓库
2. GUI
    2.1. TortoiseSVN
    2.2. TortoiseGit
3. Browser interface for CVS and SVN version control repositories

IV. Configuration Management(配置管理)

24. Ansible - SSH-based configuration management, deployment, and task execution system
1. install
2. Getting Started
3. ansible - run a command somewhere else
    3.1. host-pattern
    3.2. -a MODULE_ARGS, --args=MODULE_ARGS module arguments
    3.3. -i INVENTORY, --inventory-file=INVENTORY specify inventory host file (default=/etc/ansible/hosts)
    3.4. -m MODULE_NAME, --module-name=MODULE_NAME module name to execute

(default=command)
3.5. -s, --sudo run operations with sudo (nopasswd)
3.6. -u REMOTE_USER, --user=REMOTE_USER connect as this user (default=root)
3.7. 使用实例
4. ansible-doc - Show Ansible module documentation
5. ansible-playbook - run an ansible playbook
5.1. 包含文件用法

25. Capistrano
26. Puppet
1. Installing Puppet CentOS 6.3
2. Puppet 签名
2.1. Agent 节点
2.2. Master 服务器
3. test
3.1. Master
3.2. Agent
4. 配置文件
4.1. /etc/sysconfig/puppet
4.2. /etc/puppet/fileserver.conf
5. manifests
5.1. node
5.2. group, user 用户组管理
group
user
5.3. file
ensure
source
owner, group, mode
5.4. package
5.5. service
5.6. exec
5.7. cron
6. modules
7. firewall 配置

8. debug
    8.1. master
    8.2. node
9. FAQ
    9.1. err: Could not request certificate: No route to host - connect(2)
    9.2. No help available unless you have RDoc::usage installed

27. SaltStack
1. 安装 Salt Stack
    1.1. 服务端安装
    1.2. 客户端安装
    1.3. 防火墙配置
    1.4. key 管理
    1.5. 测试
    1.6. Demo
2. salt-key - Salt key is used to manage Salt authentication keys
3. salt 命令
    3.1. cmd
        cmd.run
        cmd.script
    3.2. pkg.install
    3.3. network.interfaces
    3.4. salt example
4. /etc/salt/master
    4.1. File Server settings
    4.2. Pillar settings
    4.3. Node Groups
    4.4. File Server Backend
5. sls 脚本
    5.1. pkg
    5.2. service
6. FAQ
    6.1. Git fileserver backend is enabled in configuration but could not be loaded, is git-python

installed
28. Chef
    1. 安装 Chef
        1.1. CentOS
29. Cobbler
30. Cfengine
31. func
32. (R)?ex Deployment & Configuration Management
33. 基于Web的系统管理软件
    1. Webmin
        1.1. webalizer
    2. ajenti

表格清单

1. 表格标题

范例清单

11.1. Docker 部署 GitLab 查看登陆密码
11.2. Docker 部署 gitlab-runner 注册演示
11.3. Example - Release Notes
12.1. Shell Docker 示例
13.1. SonarQube pom.xml 配置
21.1. authz
26.1. puppetd
26.2. puppetca
27.1. salt command

# Netkiller DevOps 手札

## Software engineering platform, Integrated SCM & Project Management, Version Control System，Continuous Integration & Delivery

## 《Netkiller DevOps 手札》视频教程

**Mr. Neo Chan, 陈景峯(BG7NYT)**

中国广东省深圳市望海路半岛城邦三期
518067
+86 13113668890

<netkiller@msn.com>

$Date: 2013-04-10 15:03:49 +0800 (Wed, 10 Apr 2013) $

电子书最近一次更新于 2023-04-05 18:46:58

版权 © 2009-2023 Neo Chan

版权声明

转载请与作者联系，转载时请务必标明文章原始出处和作者信息及本声明。




http://www.netkiller.cn
http://netkiller.github.io
http://netkiller.sourceforge.net
微信公众号: netkiller
微信：13113668890 请注明“读者”
QQ：13721218 请注明“读者”
QQ群：128659835 请注明“读者”
知乎专栏 | 多维度架构






打开“微信 / 发现 / 搜一搜”

系列文档

Netkiller Linux 手札 Netkiller FreeBSD 手札 Netkiller Shell 手札 Netkiller Security 手札
Netkiller Web 手札 Netkiller Monitoring 手札 Netkiller Storage 手札 Netkiller Mail 手札
Netkiller Virtualization 手札 Netkiller Cryptography 手札

---

# 致读者

Netkiller 系列手札 已经被 Github 收录，并备份保存在北极地下250米深的代码库中，备份会保留1000年。

Preserving open source software for future generations



The world is powered by open source software. It is a hidden cornerstone of modern civilization, and the shared heritage of all humanity.

The GitHub Arctic Code Vault is a data repository preserved in the Arctic World Archive (AWA), a very-long-term archival facility 250 meters deep in the permafrost of an Arctic mountain.

We are collaborating with the Bodleian Library in Oxford, the Bibliotheca Alexandrina in Egypt, and Stanford Libraries in California to store copies of 17,000 of GitHub's most popular and most-depended-upon projects—open source's "greatest hits"—in their archives, in museum-quality cases, to preserve them for future generations.

https://archiveprogram.github.com/arctic-vault/

# 自述

Netkiller 手札系列电子书 http://www.netkiller.cn

## Netkiller DevOps 手札

陈景峯 著







知乎: netkiller | Bilibili: netkiller | QQ: 13721218 | 微信: 13113668890

《Netkiller 系列 手札》是一套免费系列电子书，netkiller 是 nickname 从1999 开使用至今，“手札” 是札记，手册的含义。

2003年之前我还是以文章形式在BBS上发表各类技术文章，后来发现文章不够系统，便尝试写长篇技术文章加上章节目录等等。随着内容增加，不断修订，开始发布第一版，第二版.....

IT知识变化非常快，而且具有时效性，这样发布非常混乱，经常有读者发现第一版例子已经过时，但他不知道我已经发布第二版。

我便有一种想法，始终维护一个文档，不断更新，使他保持较新的版本不过时。

第一部电子书是《PostgreSQL 实用实例参考》开始我使用 Microsoft Office Word 慢慢随着文档尺寸增加 Word 开始表现出力不从心。

我看到PostgreSQL 中文手册使用SGML编写文档，便开始学习 Docbook SGML。使用Docbook写的第一部电子书是《Netkiller Postfix Integrated Solution》这是Netkiller 系列手札的原型。

至于“手札”一词的来历，是因为我爱好摄影，经常去一个台湾摄影网站，名字就叫“摄影家手札”。

由于硬盘损坏数据丢失 《Netkiller Postfix Integrated Solution》 的 SGML文件已经不存在; Docbook SGML存在很多缺陷 UTF-8支持不好，转而使用Docbook XML.

目前技术书籍的价格一路飙升，动则￥80，￥100，少则￥50，￥60. 技术书籍有时效性，随着技术的革新或淘汰，大批书记成为废纸垃圾。并且这些书技术内容雷同，相互抄袭，质量越来越差，甚至里面给出的例子错误百出，只能购买影印版，或者翻译的版本。

在这种背景下我便萌生了自己写书的想法，资料主要来源是我的笔记与例子。我并不想出版，只为分享，所有我制作了基于CC License 发行的系列电子书。

本书注重例子，少理论（捞干货），只要你对着例子一步一步操作，就会成功，会让你有成就感并能坚持学下去，因为很多人遇到障碍就会放弃，其实我就是这种人，只要让他看到希望，就能坚持下去。

# 1. 写给读者

*为什么写这篇文章*

有很多想法,工作中也用不到所以未能实现，所以想写出来,和大家分享.有一点写一点,写得也不好,只要能看懂就行,就当学习笔记了.

开始零零碎碎写过一些文档，也向维基百科供过稿，但维基经常被ZF封锁，后来发现sf.net可以提供主机存放文档，便做了迁移。并开始了我的写作生涯。

这篇文档是作者20年来对工作的总结,是作者一点一滴的积累起来的，有些笔记已经丢失，所以并不完整。

因为工作太忙整理比较缓慢。目前的工作涉及面比较窄所以新文档比较少。

我现在花在技术上的时间越来越少，兴趣转向摄影，无线电。也想写写摄影方面的心得体会。

*写作动力:*

曾经在网上看到外国开源界对中国的评价，中国人对开源索取无度，但贡献却微乎其微.这句话一直记在我心中，发誓要为中国开源事业做我仅有的一点微薄贡献

另外写文档也是知识积累，还可以增加在圈内的影响力.

人跟动物的不同,就是人类可以把自己学习的经验教给下一代人.下一代在上一代的基础上再创新,不断积累才有今天.

所以我把自己的经验写出来,可以让经验传承

*没有内容的章节:*

目前我自己一人维护所有文档，写作时间有限，当我发现一个好主题就会加入到文档中，待我有时间再完善章节，所以你会发现很多章节是空无内容的.

文档目前几乎是流水帐试的写作，维护量很大，先将就着看吧.

我想到哪写到哪,你会发现文章没一个中心,今天这里写点,明天跳过本章写其它的.

文中例子绝对多,对喜欢复制然后粘贴朋友很有用,不用动手写,也省时间.

理论的东西,网上大把,我这里就不写了,需要可以去网上查.

我爱写错别字,还有一些是打错的,如果发现请指正.

文中大部分试验是在Debian/Ubuntu/Redhat AS上完成.

写给读者

至读者:

我不知道什么时候，我不再更新文档或者退出IT行业去从事其他工作，我必须给这些文档找一个归宿，让他能持续更新下去。

我想捐赠给某些基金会继续运转，或者建立一个团队维护它。

我用了20年时间坚持不停地写作，持续更新，才有今天你看到的《Netkiller 手扎》系列文档，在中国能坚持20年，同时没有任何收益的技术类文档，是非常不容易的。

有很多时候想放弃，看到外国读者的支持与国内社区的影响，我坚持了下来。

中国开源事业需要各位参与，不要成为局外人，不要让外国人说:中国对开源索取无度，贡献却微乎其微。

我们参与内核的开发还比较遥远，但是进个人能力，写一些文档还是可能的。

系列文档

下面是我多年积累下来的经验总结，整理成文档供大家参考:

Netkiller Architect 手札

Netkiller Developer 手札

Netkiller PHP 手札

Netkiller Python 手札

Netkiller Testing 手札

Netkiller Cryptography 手札
Netkiller Linux 手札
Netkiller FreeBSD 手札
Netkiller Shell 手札
Netkiller Security 手札
Netkiller Web 手札
Netkiller Monitoring 手札
Netkiller Storage 手札
Netkiller Mail 手札
Netkiller Docbook 手札
Netkiller Version 手札
Netkiller Database 手札
Netkiller PostgreSQL 手札
Netkiller MySQL 手札
Netkiller NoSQL 手札
Netkiller LDAP 手札
Netkiller Network 手札
Netkiller Cisco IOS 手札
Netkiller H3C 手札
Netkiller Multimedia 手札
Netkiller Management 手札
Netkiller Spring 手札
Netkiller Perl 手札
Netkiller Amateur Radio 手札

# 2. 作者简介

陈景峯 (ㄔㄣˊ ㄐㄧㄥˇ ㄈㄥ)

Nickname： netkiller | English name: Neo chen | Nippon name: ちん けいほう (音訳) | Korean name: 천징봉 | Thailand name: ภูมิภาพภูเขา | Vietnam: Trần Cảnh Phong

Callsign: BG7NYT | QTH: ZONE CQ24 ITU44 ShenZhen, China

程序猿，攻城狮，挨踢民工, Full Stack Developer, UNIX like Evangelist, 业余无线电爱好者（呼号：BG7NYT）,户外运动，山地骑行以及摄影爱好者。

《Netkiller 系列 手札》的作者

**成长阶段**

1981年1月19日(庚申年腊月十四)出生于黑龙江省青冈县建设乡双富大队第一小队

1989年9岁随父母迁居至黑龙江省伊春市，悲剧的天朝教育，不知道那门子归定，转学必须降一级，我本应该上一年级，但体制让我上学前班，那年多都10岁了

1995年小学毕业，体制规定借读要交3000两银子(我曾想过不升初中)，亲戚单位分楼告别平房，楼里没有地方放东西，把2麻袋书送给我，无意中发现一本电脑书BASIC语言，我竟然看懂了，对于电脑知识追求一发而不可收，后面顶零花钱，压岁钱主要用来买电脑书《MSDOS 6.22》《新编Unix实用大全》《跟我学Foxbase》。。。。。。

1996年第一次接触UNIX操作系统，BSD UNIX, Microsoft Xinux(盖茨亲自写的微软Unix，知道的人不多)

1997年自学Turbo C语言，苦于没有电脑，后来学校建了微机室才第一次使用QBASIC(DOS 6.22 自带命令)，那个年代只能通过软盘拷贝转播，Trubo C编译器始终没有搞到，

1997年第一次上Internet网速只有9600Bps, 当时全国兴起各种信息港域名格式是www.xxxx.info.net, 访问的第一个网站是NASA下载了很多火星探路者拍回的照片，还有“淞沪”sohu的前身

1998~2000年在哈尔滨学习计算机，充足的上机时间，但老师让我们练打字（明伦五笔/WT）打字不超过80个/每分钟还要强化训练，不过这个给我的键盘功夫打了好底。

1999年学校的电脑终于安装了光驱，在一张工具盘上终于找到了Turbo C, Borland C++与Quick Basic编译器，当时对VGA图形编程非常感兴趣，通过INT33中断控制鼠标，使用绘图函数模仿windows界面。还有操作 UCDOS 中文字库，绘制矢量与点阵字体。

2000年沉迷于Windows NT与Back Office各种技术，神马主域控制器，DHCP，WINS，IIS，域名服务器，Exchange邮件服务器，MS Proxy, NetMeeting...以及ASP+MS SQL开发；用56K猫下载了一张LINUX。ISO镜像，安装后我兴奋的24小时没有睡觉。

## 职业生涯

2001 年来深圳进城打工,成为一名外来务工者. 在一个4人公司做PHP开发，当时PHP的版本是2.0, 开始使用Linux Redhat 6.2.当时很多门户网站都是用FreeBSD,但很难搞到安装盘，在网易社区认识了一个网友,从广州给我寄了一张光盘，FreeBSD 3.2

2002 年我发现不能埋头苦干,还要学会"做人".后辗转广州工作了半年，考了一个Cisco CCNA认证。回到深圳重新开始，在车公庙找到一家工作做Java开发

2003 年这年最惨,公司拖欠工资16000元,打过两次官司2005才付清.

2004 年开始加入分布式计算团队,目前成绩，工作仍然是Java开发并且开始使用PostgreSQL数据库。

2004-10月开始玩户外和摄影

2005-6月成为中国无线电运动协会会员,呼号BG7NYT,进了一部Yaesu FT-60R手台。公司的需要转回PHP与MySQL，相隔几年发现PHP进步很大。在前台展现方面无人能敌，于是便前台使用PHP，后台采用Java开发。

2006 年单身生活了这么多年,终于找到归宿. 工作更多是研究PHP各种框架原理

2007 物价上涨,金融危机，休息了4个月（其实是找不到工作），关外很难上439.460中继，搞了一台Yaesu FT-7800.

2008 终于找到英文学习方法， 《Netkiller Developer 手札》，《Netkiller Document 手札》

2008-8-8 08:08:08 结婚,后全家迁居湖南省常德市

2009《Netkiller Database 手札》,2009-6-13学车，年底拿到C1驾照

2010 对电子打击乐产生兴趣，计划学习爵士鼓。由于我对Linux热爱，我轻松的接管了公司的运维部，然后开发运维两把抓。我印象最深刻的是公司一次上架10个机柜，我们用买服务器纸箱的钱改善伙食。我将40多台服务器安装BOINC做压力测试，获得了中国第二的名次。

2011 平凡的一年，户外运动停止，电台很少开，中继很少上，摄影主要是拍女儿与家人，年末买了一辆山地车

2012 对油笔画产生了兴趣，活动基本是骑行银湖山绿道，

2013 开始学习民谣吉他，同时对电吉他也极有兴趣；最终都放弃了。这一年深圳开始推数字中继2013-7-6日入手Motorola

MOTOTRBO XIR P8668，Netkiller 系列手札从Sourceforge向Github迁移；年底对MYSQL UDF，Engine与PHP扩展开发产生很浓的兴趣，拾起遗忘10+年的C，写了几个mysql扩展（图片处理，fifo管道与ZeroMQ），10月份入Toyota Rezi 2.5V并写了一篇《攻城狮的苦逼选车经历》

2014-9-8 在淘宝上买了一架电钢琴 Casio Privia PX-5S pro 开始陪女儿学习钢琴，由于这家钢琴是合成器电钢，里面有打击乐，我有对键盘鼓产生了兴趣。

2014-10-2号罗浮山两日游，对中国道教文化与音乐产生了兴趣，10月5号用了半天时间学会了简谱。10月8号入Canon 5D Mark III + Canon Speedlite 600EX-RT香港过关被查。

2014-12-20号对乐谱制作产生兴趣（https://github.com/SheetMusic/Piano），给女儿做了几首钢琴伴奏曲，MuseScore制谱然后生成MIDI与WAV文件。

2015-09-01 晚饭后拿起爵士鼓基础教程尝试在Casio Privia PX-5S pro演练，经过反复琢磨加上之前学钢琴的乐理知识，终于在02号晚上，打出了简单的基本节奏，迈出了第一步。

2016 对弓箭（复合弓）产生兴趣，无奈天朝法律法规不让玩。每周游泳轻松1500米无压力，年底入 xbox one s 和 Yaesu FT-2DR, 同时开始关注功放音响这块

2017 7月9号入 Yamaha RX-V581 功放一台，连接Xbox打游戏爽翻了，入Kindle电子书，计划学习蝶泳，果断放弃运维和开发知识体系转攻区块链。

2018 从溪山美地搬到半岛城邦，丢弃了多年攒下的家底。11 月开始玩 MMDVM，使用 Yaesu FT-7800 发射，连接MMDVM中继板，树莓派，覆盖深圳湾，散步骑车通联两不误。

2019 卖了常德的房子，住了5次院，哮喘反复发作，决定停止电子书更新，兴趣转到知乎，B站

2020 准备找工作

职业生涯路上继续打怪升级

# 3. 如何获得文档

下载 **Netkiller 手札 (epub,kindle,chm,pdf)**
EPUB
https://github.com/netkiller/netkiller.github.io/tree/master/download/epub
MOBI
https://github.com/netkiller/netkiller.github.io/tree/master/download/mobi
PDF
https://github.com/netkiller/netkiller.github.io/tree/master/download/pdf
CHM
https://github.com/netkiller/netkiller.github.io/tree/master/download/chm

**通过 GIT 镜像整个网站**

https://github.com/netkiller/netkiller.github.com.git

$ git clone https://github.com/netkiller/netkiller.github.com.git

**镜像下载**

整站下载

```
wget -m http://www.netkiller.cn/index.html
```

指定下载

```
wget -m wget -m http://www.netkiller.cn/linux/index.html
```

**Yum 下载文档**

获得光盘介质，RPM包，DEB包，如有特别需要，请联系我

YUM 在线安装电子书

http://netkiller.sourceforge.net/pub/repo/

```
# cat >> /etc/yum.repos.d/netkiller.repo <<EOF
[netkiller]
name=Netkiller Free Books
baseurl=http://netkiller.sourceforge.net/pub/repo/
enabled=1
gpgcheck=0
gpgkey=
EOF
```

查找包

```
# yum search netkiller

netkiller-centos.x86_64 : Netkiller centos Cookbook
netkiller-cryptography.x86_64 : Netkiller cryptography
Cookbook
netkiller-docbook.x86_64 : Netkiller docbook Cookbook
netkiller-linux.x86_64 : Netkiller linux Cookbook
netkiller-mysql.x86_64 : Netkiller mysql Cookbook
netkiller-php.x86_64 : Netkiller php Cookbook
netkiller-postgresql.x86_64 : Netkiller postgresql Cookbook
netkiller-python.x86_64 : Netkiller python Cookbook
netkiller-version.x86_64 : Netkiller version Cookbook
```

安装包

```
yum install netkiller-docbook
```

# 4. 打赏（Donations）

If you like this documents, please make a donation to support the authors' efforts. Thank you!

您可以通过微信，支付宝，贝宝给作者打赏。

**银行(Bank)**

招商银行(China Merchants Bank)

开户名：陈景峰

账号：9555500000007459

**微信（Wechat）**



**支付宝（Alipay）**



**PayPal Donations**

https://www.paypal.me/netkiller

# 5. 联系方式

主站 http://www.netkiller.cn/

备用 http://netkiller.github.io/

繁体网站 http://netkiller.sourceforge.net/


联系作者

Mobile: +86 13113668890

Email: netkiller@msn.com

QQ群: 128659835 请注明“读者”

QQ: 13721218

ICQ: 101888222


注：请不要问我安装问题！

博客 **Blogger**

知乎专栏 https://zhuanlan.zhihu.com/netkiller

LinkedIn: http://cn.linkedin.com/in/netkiller

OSChina: http://my.oschina.net/neochen/

Facebook: https://www.facebook.com/bg7nyt

Flickr: http://www.flickr.com/photos/bg7nyt/

Disqus: http://disqus.com/netkiller/

solidot: http://solidot.org/~netkiller/

SegmentFault: https://segmentfault.com/u/netkiller

Reddit: https://www.reddit.com/user/netkiller/

Digg: http://www.digg.com/netkiller

Twitter: http://twitter.com/bg7nyt

weibo: http://weibo.com/bg7nyt

**Xbox club**

我的 xbox 上的ID是 netkiller xbox，我创建了一个俱乐部 netkiller 欢迎加入。

**Radio**

CQ CQ CQ DE BG7NYT:

如果这篇文章对你有所帮助,请寄给我一张QSL卡片, qrz.cn or qrz.com or hamcall.net

Personal Amateur Radiostations of P.R.China

ZONE CQ24 ITU44 ShenZhen, China

Best Regards, VY 73! OP. BG7NYT

守听频率 DMR 438.460 -8 Color 12 Slot 2 Group 46001

守听频率 C4FM 439.360 -5 DN/VW

**MMDVM Hotspot:**

Callsign: BG7NYT QTH: Shenzhen, China

YSF: YSF80337 - CN China 1 - W24166/TG46001

DMR: BM_China_46001 - DMR Radio ID: 4600441

# 部分 I. 软件项目管理

这里讲述项目管理的基本知识与方法，软件项目管理与传统行业项目管理最大的区别可能是知识型人才的管理。所谓管理大可分为两类，一类是着重考察项目过程本身，一类是主要考察项目的参与者，前者着重于时间管理，后者倾向于绩效考核。

学习管理你千万不能陷入到管理学领域，很多管理者陷入一个误区，试图寻找一种管理工具(非软件，这里指的是管理方法)，通过工具解决项目管理问题。

管理软件开发团队，你只需要20%的管理学知识，更多的是对技术的掌握。

## 1. 敏捷开发

项目管理：项目管理从管理角度出发，通常根据软件工程方法实施，通常是告诉领导我们在做什么，但常常无法安照计划进行。 敏捷开发：从开发角度出发，告诉领导我们今天做完了什么！

我认为项目管理模式的软件开发团队，不理利于创新，会降低员工的积极性，员工没有参与感，将员工视为工时，一个部件，一个资源，任凭项目经理的调度，使用。员工的想法无法得到重视，仅仅是执行命令。 这种模式会浪费每个人20%的时间用来维护时间表。

我更喜欢敏捷开发团队，我更喜欢全栈开发人员，让开发人员参与的软件开发周期的每个环节中，人力资源利用率高，让开发工作成为有趣的事，从被动接收任务分配，到主动参与其中。

软件工程当下已经显得落后。尤其是快速变化的互联网行业。

# 第 1 章 范围管理

为了实现项目的目标，对项目的工作内容进行控制的管理过程。它包括范围的界定，范围的规划，范围的调整等。

我们将技术部划分为三个子部门，这是为了实现DEVOPS而制定。

1. 软件开发部

2. 软件测试部

3. 运维部

很多中国企业组织架构五花八门，这样会造成职责不清晰。例如运维挂在开发部下面，测试挂在开发部下面

## 1. 宏观管理

### 技术人员的大局观

我们从小的教育就是如何拆分问题、解决问题，这样做显然会使复杂的问题变得更容易些。但是这带来一个新问题，我们丧失了如何从宏观角度看问题，分析问题，解决问题，对更大的整体的内在领悟能力。这导致了我们对现有问题提出的解决方案，但无法预计实施该方案后产生的各种后果，为此我们付出了巨大代价。

而我们试考虑大局的时候，总要在脑子里重新排序，组合哪些拆分出来问题，给它们编组列单。 习惯性认为解决了所有微观领域的问题，那么宏观上问题就得到了解决。然而，这种做法是徒劳无益的，就好比试图通过重新拼起来的碎镜子来观察真实的影像。

所以在一段时间后，我们便干脆完全放弃了对整体的关注。

当今的社会，几乎所有的企业情况都是岗位职责清晰，分工明确，员工是企业机器上的一颗螺丝钉，我们在招聘下属的时候也仅仅

是用他的一技之长。项目一旦立项，我们就根据项目需求针对性性的招聘，短短半年团队就会膨胀数倍，但效率并不是成正比增长。另一个问题是这个庞大的团队合作起来并不尽人意。结果是 80% 协调的时间，20% 实际工作时间。

很多技术人员埋头在自己的专业领域，更擅长解决自己专业范畴的问题，这样是不对的，技术人员应该培养广泛的兴趣，不仅仅要成为技术全栈，还要涉略艺术等领域。

## 2. 你清楚你的工作职责吗?

对于基层员工来说，员工很清楚自己的工作职责。

你是否意识到很多管理层不清楚自己的工作职责呢？或者身为管理层的自己，也不知道自己的工作职责是什么?

常有老板跟员工抢工作，经理跟下属抢工作，这种情况在当下中国很常见，甚至可以说每个公司都有这种情况。

工作职责清楚，界限模糊，这也导致了中国企业人管而非制度管理的模式。如果某个管理层能力比较强，公司可能让他跨界负责多个部门。

为什么出现这种情况呢？我们逐条分析，大家看看是否解答这个疑问?

1. 刷存在感
2. 权力不下放，对其他人不信，不相信别人能干好，凡事都亲历亲为
3. 在中国企业管理采用的是人管，而非制度管理
4. 专业的人做专业的事

### 制度管理

中国的企业是人管，而非制度管理。我2001年的时候从事OA(办公自动化软件)软件的开发，时隔17年，再次关注OA，发现没有任何进步。 中国OA软件死在工作流上，没有一款OA能够适应企业的工作流程，几乎任何企业实施OA都需要定制开发，按照改企业的管理流程重新定义工作流，流程通常不是岗位流程，而是具体负责人， 当该企业有人事变动后整个流程将无法在使用，这就是中国特色的企业。

从 Lotus Notes 到 SAP 都遇到过这种问题，这些国际软件在中国的本地化，都是血泪史。

## 权力下放

权力下放对中国企业的老板来说是非常难的一件事，凡事都亲历亲为，尤其是涉及到钱的事情上：）

## 专业的人做专业的事

理论上专业的人做专业的事，但往往是非专业的人管专业的事，很多企业老板喜欢认用自己的熟人，亲戚，朋友等等，造成了非专业的人管理专业的事。 这些非专业人士非常想做点事，也必须做点事给上面看。他根本不知道团队在干什么事，于是让专业人士干不专业的事，例如写工作报告。

非专业管理层的在专业的团队中，无所事事，不帮倒忙已经是阿弥陀佛了。

一段子这样说的：利用人才，任用奴才，不用蠢才。

## 总结

我们需要自我审视，我们的工作职责究竟是什么? 做好自己的本质工作，相互彼此信任，相信老板，相信员工，相信同事。

## 3. 怎样防止踢皮球

踢皮球，让我想起儿时的一首童谣： 小皮球架脚踢，马莲开花二十一，二五六、二五七、二八二九三十一，三五六、三五七、三八三九、四十一、四五**五七 四八四九 五十一 五五六 五五七 五八五九六十一 六五六 六五七 六八六九七十一 七五六 七五七 七八七九八十一 八五六 八五七 八八八九九十一 九五六 九五七 九八九九一百一

这首童谣与本文没有任何关系。

### 进入正题

你是否思考过办公室内踢皮球的问题？风气是怎样形成的？问题根源在哪里？怎样解决？

踢皮球在中国企业是一个严重的问题，很多管理者都束手无策。本可以顺利解决的问题会被拖延，变得严重或无法解决；事情也会在推诿中陷入无休止的恶性循环之中。

### 踢皮球几大害处

1. 踢皮球让一个积极团队走向消极
2. 踢皮球让大家失去信任感
3. 踢皮球让员工失去团队合作
4. 踢皮球不能解决问题
5. 踢皮球削弱了团队的执行力、战斗力，导致整个团队都失去竞争力。

“踢皮球”大到会阻碍企业的发展，小到影响到员工的利益。

### 场景一

一个SQL导致性能低下，DBA看到了，但不知道怎么解决，他不知道业务的需求与逻辑。而开发人员说这也业务逻辑必须这样用。需求人员说：XXX都能实现，为什么我们做不了。

运维部说程序不稳定，导致服务器经常崩溃，开发部说程序没有问题， 开发部说测试部能力不行，测不出bug，导致将有问题的代码部署到生产环境，导致系统不稳定， 测试部说程序性能太低，酌开发部优化，开发部说数据库配置优化有问题，运维部的DBA说没有问题，DBA说问题出在SQL语句与数据库操作上。 开发部推给运维部，运维部只能增加IT成本，加服务器，最后硬件升级到无法再生，程序依然有问题，仍然没日没夜的救火。

面对这个问题你有怎么看？ 如果你身在其中怎么处理？上面的故事我认为这不算是踢皮球，仅仅是矛盾的开始阶段，如果不加以遏制，就会向踢皮球方向发展，后果十分严重，踢皮球风气一旦形成，再难改变。

问题出在哪里？这是因为开发人员不懂运维，运维不懂开发，测试不懂开发，也不懂运维。都只能从自己的角度与范畴，看问题，其实他们提出的问题都没有问题，都是合理的。

好了我已经告诉你问题出在哪里了， 你现在能想出解决方案吗?

## 场景二

产品部写了一个需求文档给开发部，开发部按照需求文档开发，测试部测试通过，产品上线。产品部说这不是我要产品，向上投诉。开发部说你文档就是这么写的，我就按照需求文档开发，你说的功能文档中没有提到，测试也通过了，测试部说我们只负责功能测试不负责产品正确与否，运维一边偷笑（还好不关我的事）。 产品部说这么简单的功能，需求文档仅仅描述我们需要什么，不可能系到每个细节。

面对这个问题你有怎么看？问题出在哪里？如果你身在其中怎么处理?

场景二中的例子没有场景一中的复杂。很多管理层会将其归为沟通存在问题，沟通的确有问题，但不是主要原因，我认为还有另一个原因，但很多公司将产品独立出技术，这就变成了外部矛盾，内部矛盾还好解决，开发/测试/运维还好都是技术范畴，我们可能通过DevOps解决开发/测试/运维面临的问题，但DevOps 没有涵盖产品这一环节，这是问题的根源。

好了我已经告诉你问题出在哪里了， 你现在能想出解决方案吗?

## 踢皮球的风气是怎样形成的?

### 技术人员的单一视角

技术人习惯从单一视角看问题，思考问题，例如

产品仅从产品视角看问题，开发仅从开发视角看问题，测出视角，运维视角.....

### 责任不明确

很多管理层认为踢皮球是责任不明确造成的，如果实行责任制并且赏罚分明就可以彻底解决踢皮球的问题。

我引用一段文字:

> 要从根本上医治“踢皮球”的顽疾，必须强化责任意识。 习惯于“踢皮球”的人最根本的是责任意识淡薄，畏难情绪严重。要强化其责任意识，首先要充分授权，面对不同的任务给予适当的人力物力财力支持； 其次要明确工作内容和范围，限定完成任务的时间，不留回旋的余地；第三要实时跟踪，加强督查，及时帮助解决遇到的困难； 第四要赏罚分明，不给踢皮球者藏身之地。强化了责任意识，团队的凝聚力、执行力、竞争力就会得到加强，问题就会迎刃而解。

我不这么认为，我认为恰恰是因为责任太明确，造成了踢皮球的风气。

责任制后员工，团队或部门，只要做好自己的工作，不出问题就能交工，但其他部门的问题呢？ 这个与我无关。慢慢团队合作与沟通越来越少，为踢皮球埋下伏笔。

如果出现耦合问题是难以分清责任，谁都不想承担责任，都认为是对方问题，这就是踢皮球的开始。

如果管理层没有处理好，让一方收了委屈，他一定会找机会把面子找回来。甚至不惜给对方挖坑。

缺乏沟通

很多管理层认为是沟通出了问题，沟通是十分重要，大家都意识到这个问题。企业也反复强调，加强沟通，然而没什么效果。 我一直观察发现，中国人，两个中国人，尤其是两个中国男人在一起喜欢争来争去，不知不觉在装逼，看别人都是傻逼，其实都是二逼。 两个中国在一起，很难接受对方的思想，不会倾听对方讲话。谁嘴快，嗓门高谁占上风，占上风者强加自己的思想给对方。 嘴拙嗓门低的人下次就不会在跟你沟通了，他的游戏规则是制定一个流程规范，下次找我办事别跟我“说”，也无需见面，按我的游戏规则走流程，发邮件或书面形式。 一个中国人是龙，两个中国人是虫。中国企业难做大，就是因为合伙人之争，两个人意见不一致，导致另一个合伙人撤出，比比皆是。 外国人呢，有句英文 “What can I do for you?” 我能为你做什么。外国人更多考虑对方的感受，倾听对方然后都做出让步，最终议题达成一致。

背黑锅

很多企业遇到问题，非打破砂锅问到底，找个倒霉的背黑锅。出现问题应该全力以赴解决问题，而不是找问题的原因，谁的责任。 适可而止，大家心里都明白就可以了，没有拙破这层窗户纸，都会心里感激你。员工或团队的愧疚感会让他们更敬业。

员工问题

问题在管理层，企业必须反省，不能将问题归罪于员工，什么样的企业文化，产生什么样的团队。

## 怎样根治踢皮球

一旦风气形成定局，笔者也没有好的方案能够快速见效，扭转局面成本极高。通过新鲜血液慢慢稀释一种方法。

我们应该用尽各种手段，不让团队朝这个方向发展。一但团队朝良性发展，走上正轨，就会压倒那些不好的风气。

裁判工作非常重要，做好裁判工作不能让员工或团队受委屈。

### 调整组织架构

很多问题源于组织架构不合理。

企业初期人少，组织架构没有那么复杂，很多人身兼数值，效率非常高。随着企业成长壮大，开始重新规划组织架构，分工明确，组织开始膨胀，慢慢产生内耗，指令难以下达与贯彻，这个时期的特点是，

上层有很多决策需要实施，下面的回答永远是需要加人。 不停的加人，人力永远不够，很快办公室就无法容纳过多的员工。当膨胀的一定程度时，企业已经无法承受，领导一个命令“裁员”。

裁员是有指标的，每个部门多少人，部门管理者根据自己的喜好裁量，很多优秀的人才在这个环节流失，N+1赔偿也是不小开支，对于企业来说长痛不如短痛。

组织膨胀，分工明确带来的问题就是岗位的工资不会很高，两个人干一个人的工作，同时带来沟通成本，你真的认为员工能100%饱和工作吗？，实际上员工平均每天真正意义的工作不到4个小时。所以我主张的是一个人干一个半甚至两个人的工作，支付1.5倍薪水。

这样对于企业节省办公资源，人力成本，沟通成本等等，对于员工来说1.5倍的行业薪水有足够的竞争力，员工的敬业度也会提升。

第一步精简组织架构，通常不要超过四层的组织架构，三层最佳，当然这要看企业的规模。

第二部是层次调整，组织架构不合理还表现在很多顶级平行部门，这些平行同级别部门沟通成本最高，沟通至少需要经过两个领导。内部沟通效率远比外部沟通快捷。例如上面的场景二的故事中，就是因为产品与技术是平级部门的关系造成的。

#### 禁止追查问题源头

出现问题应该全力以赴解决问题，而不是找问题的原因，谁的责任。

问题永远是团队的而不是个人的，点到为止，任何人试图找出真凶，必须阻止。黑锅团队背。

#### 不懂技术的管理

任用不懂技术的管理层，是很多技术人员吐槽的，不懂技术就不能做好裁判工作。

#### 统一目标，价值观。

每个部门/事业部都有自己的目标，应该说是符合部门利益的目标，这个目标汇总起来并不是企业的目标。目标不一致，协调起来就有问题。

#### 防止问题扩大

在团队当中总有这样一些人，他们可能跟高层走的比较近，或者表现自己，总喜欢将邮件抄送给高层。 一个小小的问题，分分钟内部就可以处理完成，被有些人CC给高层，本不是什么大问题，但CC给高层关注后，高层不明真相，风波来了。

我们要反思的远不止这些，限于篇幅先写到这里，后面可能会更新，请关注我的博客。

## 4. 内部外包与悬赏

目前大型集团企业都会划分很多部门或更大的组织架构事业部，带来一个问题，膨胀与内耗。

导致很多岗位工作量不饱和，另一些事业部某些岗位可能人手不足。于是我想出了内部外包与悬赏这个馊主意。

### 怎么样操作

内部外包一般分为两个层级:

1. 部门级的内部外包
2. 员工级的内部外包

部门级的内部外包，比较容易实施，即 A 部门在领导协调下，将一部分工作量交给 B 部门去做，然后由领导和双方确认即可。

员工级的内部外包，主要是A部门将员工外包给B部门一段时间，与借调不同，外包需要支付该员工薪资费用给B部门。

这里我们需要一个平台，用于发布工作包。平台可以部门对部门，部门对个人，个人对个人可能性不大。

可能整体打包，也可以采用单一任务方式，可以一次性合作，短期合作，长期合作。

平台的作用:

1. 事业部将本部工作外包给另一个事业部，整体打包，长时间合作。
2. 将本部工作外包给其他事业部员工，短期合作。
3. 本部无法解决得问题，在平台上悬赏解决，一次性合作。
4. 顾问咨询服务，动嘴不动手的工作，一次性合作。

这样可以最大化利用人力资源，调动员工的积极性，员工可以选择自己感兴趣的工作包。

内部外包对于企业来说仅仅是内部资金分配问题。

而悬赏对于快速解决紧急问题十分有效。我们也见过内部悬赏，都是危机时刻，例如前不久的携程时间，就曾内部悬赏。

对于每个事业部，不可能全能的，总有不擅长的领域，通过平台很好的解决了取长补短的问题，甚至可能A/B两个部门对调项目。

## 可能遇到的问题

那么内部外包与悬赏也并非完美

可能会出现一些问题:

1. 员工做外单，可能比在本部门赚的还多。从而影响本质工作。

2. 做外单的同时，突出接到本部的工作。

3. 工作难易度与费用定价问题，这个问题不是问题，遵循市场定价理论即可，定价低没有人接单，定价高会疯抢。

4. 本门员工一直在做另一个部门的工作，员工是否调岗更合适呢?

5. 有些员工可能认为将自己解决不了的问题发布到平台上悬赏，有损面子，或者担心公司怀疑自己的能力有问题。这里要做好员工的工作，悬赏是为了快速解决问题，自己摸索解决难题是需要时间成本的。

6. 本来是相互协作的问题却演变成了互相竞争问题

## 小结

1. 虽然叫做内部外包，并非只能通过资金结算，也可以通过其他形式体现，请其他事业部吃饭，旅游.....。

2. 注意事项，很多企业一旦财务和绩效考核介入，会牵扯出很多问题，导致这个方案无法实施。

## 5. 团队膨胀的原因分析

随着企业的成长与发展，扩张是必须的，有些扩张是合理。有时企业并没有扩张，但内部开始膨胀。

企业膨胀有很多原因，有外在原因，也有内在原因。大而空泛的原因这里不谈，我仅仅从各部门角度谈起。

加速膨胀的表象是，人手永远不够，增加人手后，但产出却不成正比。

### 人才管理

#### 先从员工个人谈起

企业需要员工创造最大化价值，那么员工掌握的技能越多，能胜任的岗位越多，对企业越有利。 专一领域技能越熟练，产生次品率越低，员工越优秀。

那么对于员工而言呢？对于大多数人来说，学习新技能，只是满足兴趣，对技术的热忱。

多掌握一种技能或者将工作做好，薪水不会有什么太大变化。最终很多员工放弃学习，随大流，随大流才是主流。

有些员工还会可以藏匿自己的技能，多一事不如少一事，很多时候员工同事负责两个项目，干好了没有奖励，干杂了还需承担责任。

#### 技能管理

管理层不知道员工技能情况，当企业需要人才的时候，首先想到的是招聘，而不是从内部获得。

### 再说说部门

#### 部门膨胀

公司现在需要一项新技术，内部学习是可能掌握该技术的。

首先管理层年纪偏大，已经过了学习期，能做到管理层位置上靠的是更多是办公室政治手段，好不容易爬到管理层，该享福了。

对于管理层来说，学习新技能是痛苦的，不如招聘一个岗位更省事。例如设置副手岗位，让副手专项负责该项目，或干脆独立出一个部门。

## 精细化管理带来的膨胀

很多企业迷信精细化管理，认为每个人一个岗位，做好自己岗位上的工作，整体就是最好。 于是制定各种岗位规范，工作流程，后果是什么呢？组织臃肿，流程繁琐，应变能力差。 当企业开始意识到精细化管理代理的问题后，就会进入下一步“裁员”。

# 第 2 章 时间管理

为了确保项目最终的按时完成的一系列管理过程。它包括具体活动的界定，如：活动排序、时间估计、进度安排及时间控制等项工作。

时间管理站项目管理40%的内容，很多人对项目管理的理解就是时间管理，其次就是质量管理。

时间是公司最稀缺资源，但多数时间并没有得到高效合理利用。若想让你的员工达到最高工作效率，必须珍视他们时间，制定规章合理分配时间，并投入精力尽可能为公司创造最大价值。

## 1. 优先级管理

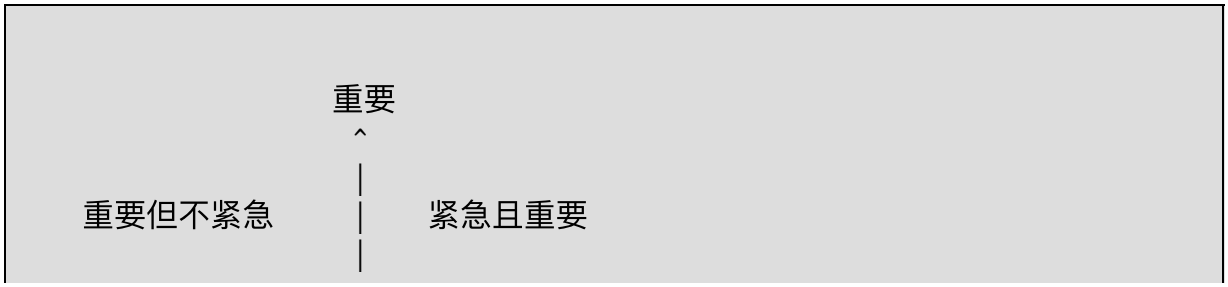
重要的事通常不紧急，紧急的事通常不重要

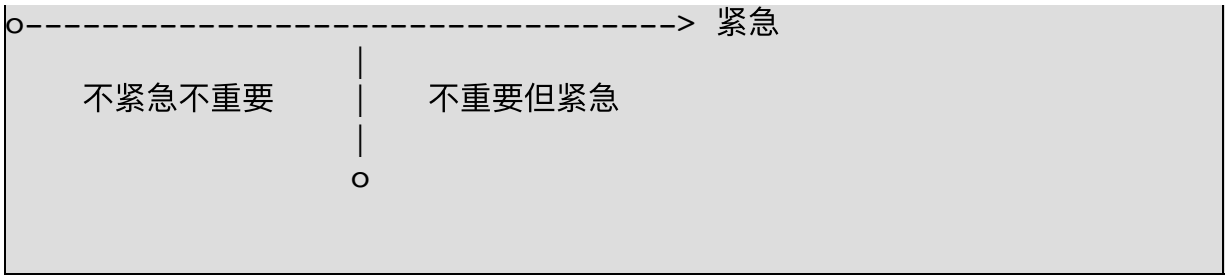
—德怀特·艾森豪威尔

什么是时间“四象限”法?

时间“四象限”法是美国的管理学家科维提出的一个时间管理的理论，把工作按照重要和紧急两个不同的程度进行了划分，基本上可以分为四个“象限”：既紧急又重要(如客户投诉、即将到期的任务、财务危机等)、重要但不紧急(如建立人际关系、人员培训、制订防范措施等)、紧急但不重要(如电话铃声、不速之客、部门会议等)、既不紧急也不重要(如上网、闲谈、邮件等)。

按处理顺序划分：先是既紧急又重要的，接着是重要但不紧急的，再到紧急但不重要的，最后才是既不紧急也不重要的。






对于大多人数，能排除「不重要也不紧急」的事情，但很难摆脱「不重要却紧急」的事务。

有有多少人慎重对待「重要但不那么紧急」的事情，并且把它排到日程表上?

陷入忙碌的人们通常会选择紧急处理的事情，却常常忽略重要的事情。

## 2. 时间管理的误区

时间管理强调怎样高效利用时间，例如，日程管理，任务排序，四象限分析，等等。

如果采用这样的管理方法，累死为止也无法完成工作。

我所理解的时间管理完全不同。

### 优化组织架构，精简机构

从企业管理者到一线员工之间的层级越多，信息流动和决策速度就越慢，时间成本就越高。

一家公司若增添一名管理者，平均会衍生出相当于1.5名全职员工所负担的新工作量，也就是说除了管理者本人自己负担的工作外，还要占用另一名员工50％的工作；每增添一名副总裁，则衍生出2.6名全职员工的工作量。雇用新的管理者或高级管理者，可能还需要接着雇用助理或办公室主任，带来一连串资源消耗，会进一步增加工作量和成本。随着工作越积越多，时间也变得更加短缺。

### 命令决策一元化

很多企业采用矩阵式结构经营，但决策权和问责制度模糊不清。因此，花在协调各职能和业务单元之间的时间急剧增多，成本随之增加。

另一种模式是分布式模式，它与矩阵模式的区别就在于，分布式的每个节点都能独立运作，而矩阵式看似每个节点独立，但离开中央却完全无法运行。

企业要么采用一元化决策，要么采用分布式决策。

### 时间线

被管理者痛苦的根源是什么？是无法对时间做主。

很多企业会对给员工培训时间管理，但是从未意识到时间线的概念。管理层多把时间管理理解成 Schedule(计划表)或者 ToDo List (待办事项)。

我理解的时间管理是 Time Master 我不知道怎么翻译更贴切，这个概念来自美国的一部科幻美剧，有个组织负责宇宙时间线的管理。

每个人都有自己的时间线，这个时间线一旦被打乱，就会影响他的节奏，如同多米诺骨牌效应，产生连锁反应。

每个人都能安排好自己的时间线，可能执行的时候会出现偏差，但大体上不会有什么问题。员工也是如此，每个人，每天做什么还是心里有数的，可是常常会被临时任务穿插进来，导致后面的多米诺骨牌效应，所以时间管理核心就是不要打乱每个人的时间线。

为什么管理层工作一天不觉得累，因为管理层更多是时间管理者，能够掌控时间，安排时间，调整时间。

作为员工只能是被安排时间，调整时间，打乱时间，最终导致多米诺骨牌效应，时间雪崩。

## 3. 项目管理中工时计算的问题

### 背景

1. 为什么项目总是不能按时结项?
2. 为什么工期一再延误?
3. 员工不够努力吗?
4. 时间去了哪里?

### 面临的问题

普遍问题是，我们至今对知识型工作者的做事效率，仍采用工业时代的评价模式。若工作者每小时的效率产出基本一致，那关注他们的工作时长便行之有理。 对于重复性劳动，这种评价模式可能确实管用，但对知识型工作者就不太适用。

### 工时去了哪里?

据统计一个典型的美国办公室工作者，每个工作日只能完成90分钟真正有意义的工作。

当天剩余的大部分时间，都被浪费在各种分心事务上，比如阅读新闻、网上冲浪、同事社交、吃零食、喝咖啡、翻看报纸、处理无关邮件、不必要的拖延行为、玩游戏、做白日梦等。 多说一句，美国办公室工作者在世界范围内还算最高效的人群。在其他很多国家，人们每天完成的实质工作甚至更少。

#### 洗手间，茶水间，吸烟

- 洗手间，先不分男女性别差异以及大小号，平均下来10~15分钟一次没有意见吧？那么一天至少两次吧?
- 茶水间，洗杯子，泡茶，清理茶叶残渣，遇到同时还需要寒暄几句。需要10分钟吧？ 一天至少2~4次吧?

- 吸烟，我不吸烟不太清楚，但是看到公司的烟友抽的爽，聊得欢。20分钟要吧?

看邮件，写邮件

我最讨厌邮件群发，或将不相关的邮件也CC给我。阅读这些不相关的邮件是非常浪费时间的，你不清楚那封邮件中会与你有关，只能逐一阅读。 写邮件更浪费时间，给领导写邮件浪费时间乘以二，既要要注意措辞，还要注意语气。 阅读邮件至少需要30分钟时间，写邮件需要1个小时

沟通

很多企业组织架构层级导致沟通不顺畅，增加沟通成本，工时内耗。企业应该为员工提供开放的沟通，而不是层层转达。

查资料

查资料并没有浪费工时，磨刀不误砍柴工。但资料查了，问题没有解决工时就被消耗了。

无关的会议

中国是开会就是你一句我一句出主意，大都无法落实。很多无关会议把你拉去旁听。一般会议时间都在2个小时左右。

不必要的拖延行为

员工拖延时间有很多原因，不一定都是员工的问题，多是企业的问题造成的，所以企业自身要找原因，不要归罪为员工问题。

私人时间

- 看手机短信，接听乱七八糟的电话（买保险的，贷款的，卖房的，头自理财的，诈骗的）
- 微信，朋友圈

- 看新闻，玩游戏

以上仅仅粗略列出几项影响工时的因素，但实际过程中远不止这些，会有更多因素会占用工时。

## 怎样改善面临的问题

死扣工时的时代已经过去了，我们应该更多关注员工产出成果。

欧美企业尽可能的为员工提供最好的工作，弹性工作时间甚至可以SOHO办公，不是让员工享受，而是让员工最大化产出。

目标管理，价值管理都胜过时间管理。

我认为项目管理应该改叫项目服务，项目服务能更描述项目人员的角色。

员工每天真正投入工作的时间越长，产出就越多，做有真正有意义的工作才是王道。

## 怎样计算项目工时?

项目管理中通常是采用8小时/每天，一周40小时来计算工时。

项目延期主要问题就是工时计算不合理，项目工时不能与8小时工作制挂钩。

8小时工作制，仅仅是规定员工在8小时之内要工作岗位上。

员工不是机器人，不可能8小时内，一刻不停的工作。

所以减去上洗手间，茶水间，吸烟，处理工作邮件，回复工作即时消息，开无用的会议等等时间，员工剩下多少真正有意义的工作时间?

所以我认为保守计算，项目工时应该按6小时计算，甚至一些特殊情况按4小时计算。

## 4. 项目延期

项目延期不一定是团队不够努力，导致拖延的原因有很多种，有些是管理者的责任，还有以下是团队成员的责任:

1. 工时计算本身有问题。
2. 追求完美。
3. 认为分配违背员工意愿。
4. 害怕犯错误，承担责任或受到批评。

## 5. 当日事当日毕

很多读者询问我是怎样学习的，他们无论如何也想象不出我是如何掌握如此多的跨界知识，且在每个领域都能达到一定深度。

休息时安心休息，工作时一心一意

我的学习原则是:

1. 只在公司学习
2. 不占用业余时间
3. 工作用到的技术才会学习

周围了解我的朋友都知道我爱好非常多。我从不把技术带回家，所以极少在家中学习技术，或者在家中完成未完成的工作。偶尔写作写的也是非技术文章。 在我看来每天在公司的8小时已经足够了，所以你需要充分的利用好这8个小时，至少不要浪费。很多同事在工作时间忙里偷闲看看新闻，刷刷朋友圈，而此时我在学习新的技术。

回家家里与技术完全切断，家中没有开通有线电视，已有十几年没有这看过电视，偶尔使用安卓盒子或App 看看CCTV9、CCTV10 主要看纪录片和动物世界居多，只看美剧，从不看国产电视剧。业余时间全部用在业余爱好上，无线电，自行车，汽车，户外运动，等等

**工作时间如何规划?**

我通常8:00 左右到公司，首先我会浏览一下开发软件资讯，包括业界比较热门的技术，看看那些软件升级了，关注一下 Release Note 文档，这样便可以了解了软件的版本间发生了那些变化。 几十年不间断地关注主流软件的 Release Note 才能对每个版本的变化了如指掌，这样在软件升级时才能从容不迫，我一向是喜欢追逐并尝试新版本的。

上午工作安排

```
8:50 ～ 9:00 处理邮件，回顾昨天的工作，以及安排好当日的工作，无论是自己的工作，还是团队的工作都做到 当日事,当日毕。

9:00  ~ 11:30 开始专心工作

11:30 ~ 12:00 稍作休息，放松一下，想一想之前的方案是周全，还有那些需要补充，处理一些杂事，例如回复电邮。 将一些经验，总结等等写入《Netkiller 手札》对应的章节。
```

下午工作安排

```
13:00 ~ 15:30 专心工作

15:30 ~ 16:00 稍作休息，放松一下，处理一些工作上的杂事。 将一些经验，总结等等写入《Netkiller 手札》对应的章节。

16:00 ~ 17:30 专心工作

17:30 ~ 18:00 提交 Issue/ Ticket，检查团队的当日工作。 将一些经验，总结等等写入《Netkiller 手札》对应的章节。
```

工作安排大致如此，实际情况时间点稍有出入，大致上:

```
专心工作（无打扰）—> 休息，思考，总结 —> 继续工作 —> 再思考，总结
```

《Netkiller 系列 手札》既是我的电子书，也是我的笔记，从2000年至今更新从未中断过。工作中需要用到什么知识，只需找到索引章节，拿来即用。手札内容涵盖了从开发，测试到运维所涉及的所有知识和工具。

**遇到工作被打断的情况**

处理原则是：

1. 告知：“我手头上有事，正在忙”
2. 协商：“XXX帮你处理，行吗？”
3. 计划：未来XXX时间处理
4. 答复：按照承诺的时间回电或答复

会议是十分浪费时间的，但又不得不开会，减少会议浪费工作时间非常重要，请阅读《Nekiller Management 手札》·沟通管理章节。

# 第 3 章 沟通管理（Communication Management）

为了确保项目的信息的合理收集和传输所需要实施的一系列措施，它包括沟通规划，信息传输和进度报告等。

我的要求就是单向精准，消息漏斗化。单向是有别于广播的。

很多企业喜欢使用广播试沟通，典型的例子是将电子邮件CC抄送给所有人，有关无关均抄送。这带来一个问题，员工每日面对一屏幕的电子邮件，找出与自己有关的邮件，既浪费时间也容易出错。所以电邮需要精准投递，不要发给无关的人。

消息漏斗化是指消息到达最终接收者，中间经过的环节不断过滤，消息量越来越少。举例，10个需求，5个评审通过，3个实现不了，1个开发实现不了，最终只有1个需求安排给开发者。

## 1. 表达方式

职场生存就不免不了与人打交道，高大上的说法叫“沟通”。不像生活中的沟通有天人的信任和默契，哪怕是不那么注意谈话方式，也不会产生误解。

职场上则不同，我们经常需要面对陌生人，需要更多技巧和规则。

### 如何提问

提问，询问，反问，质问

询问：是了解情况，是领导对下属的问话方式。评级之间了解情况的沟通方式。

提问：是引发思考，领导可以对下属提问，如果下属能向领导提问，表示该员工很有潜力，具备了上向管理的能力。

反问和质问：是无能领导问责下属的手段，平级之间采用这种沟通方式，会带来很多问题，例如背锅文化，踢皮球文化，一旦蔓延，愈演愈烈。

## 拒绝反问和质问

职场沟通应该尽量使用陈述句和祈使句，措辞需非常谨慎，以免产生歧义，甚至伤害对方的情绪而诱发矛盾。

当今的职场90、95后为主力，这一代独生子女内心比较脆弱，管理层稍有措辞不当，就会伤害到他们的情绪，后果是撂挑子就走。

无论是谁被反问的时候，都难免产生被冒犯的情绪。

XXXX 难道你们之前没有人对 XXX 进行测试验证吗?
我已经跟你们说我无数次，为什么还犯这种低级错误?
这个 XXX 问题，大概一个月前就提出来了，为什么还没处理好?

换一种说法

XXX看来是测试不足，您看我们采用 XXX 方看是否可行?
目前的问题是 XXX，我们应该讨论一下怎么彻底解决，不再发生同样的事情。
是什么影响了进度，让我们一起分析一下。

职场中，善用反问的同事是令人生厌的。换做官场，对上级或者平级采用反问的方式，仕途就此终结。

## 宽以律己，严以待人

中国人常常说“严以律己，宽以待人”，但你会发现现实中多数人是：“宽以律己，严以待人”.

中国的管理层都喜欢使用和珅这样的奴才，总是希望下属能揣摩出自己的意思。

对他人要求极高，例如：

这个你应该能想到

这么简单的问题你怎么就没想到？

这个产品做的真差劲

这是谁写的代码？

任何设计或编码都是团队辛苦努力的结果。面对这种情况，如果提示质疑者没有给出解决方案，这件事必须严肃处理，要对说话负责。

## 任务分配

一旦时间点确定，接下来就是分配任务倒指定开发人，任务的分配十分讲究，分配任务要精确描述，不能使用模糊语言，那样会造成误解。我的分配原则是5W2H方法：

- What：做什么事？
- Why：为什么做这件事？有什么意义？目的是什么？有必要吗？
- When：什么时候做，完成的时间是否适当？
- Where：在什么地方做，在什么范围内完成？
- Who：由谁负责做？由谁负责执行？谁更合适？熟练程度低的人能做吗？
- How：怎样做

- How much: 成本 （不是所有岗位都会涉及成本）

举例，运维任务

- What：为api服务器做负载均衡，多增加一个节点，负载均衡算法采用最小连接数。
- Why：目前api服务器只有一台，如果出现故障将影响到所有业务运行，顾该服务器存在单点故障，需要增加节点。
- When：本周内完成，周末上线。（此处可以写日期）
- Where：在A机柜，低2机位处，连接倒交换机第三个端口。
- Who：XXX负责网络配置，XXX负责上架，XXX 负责验收测试
- How：增加/etc/hosts设置如下
  - api.example.com  127.0.0.1
  - api1.example.com  192.168.2.5
  - api2.example.com  192.168.2.6

举例，开发任务

- What：增加图片验证码。
- Why：目前用户注册登陆以及发帖无验证吗，某些用户通过机器人软件批量开户/发广告帖，给管理带来很大困扰。
- When：2014-06-15 开始开发，2014-06-20 12:00 上线。
- Where：用户注册，登陆与发帖处增加该功能，。
- Who：张三负责验证码生成类的开发，李四负责用户注册，登陆UI修改，王五负责发帖UI的修改。
- How：具体怎么操作的细节，此处省略200字...

举例，测试任务

- What：测出XXX软件并发性能。
- Why：目前XXX软件在线任务达到200后，用户反映速度慢，经常掉线。
- When：故障时间点10：00AM，需要周二完成测试，周五完成优化，月底上线。（此处可以写日期）
- Where：在AAA分支检出代码，编译后部署到BBB环境。
- Who：XXX负责网络配置，XXX负责软件部署，XXX 负责测试
- How：具体怎么操作的细节，此处省略200字...

举例，促销任务

- What：XXX产品促销。
- Why：目前XXX产品在 XXX 市场占有率 XXX 用户反映 XXX。
- When：促销起始时间 XXX 结束时间 XXX
- Where：AAA 细分市场，BBB区域。
- Who：XXX负责 XX，XXX负责 XX，XXX 负责 XX
- How：具体怎么操作的细节，此处省略200字...
- How much: 成本XXX

**任务确认**

当接受分配的任务后，最好能够在沟通完毕后确认一下对方的意思，例如：

“XXX 总, 我没理解错误的话，您的意思是要求我：第一，.....; 第二，.....; 第三, .....; 是这样吗？ 没有问题的话我就按这个去执行了。”

## 2. 会议管理

### 什么要开会?

多数时候我们是被动参加会议！很少有人真正的去思考开会的目的和价值！

开会应该是为了明确工作任务、提升效率、减少沟通成本！

### 会议的时间成本

会议不仅占用管理层的时间，更占用员工的时间，所以要严控会议用时。

开会就要有解决方案，成熟的方案，否则不要开会，开了没有意义，浪费时间。

### 集思广益纯属扯淡

很多人相信三个臭皮匠，还是臭皮匠，顶一个诸葛亮。

在我看来三个臭皮匠，还是臭皮匠，永远不会成为诸葛亮，如果三个臭皮匠能成为诸葛亮，就不会出现三足鼎立的情况。

大部分的会议就浪费在听取广大群众的发言。

通常我们看到的会议就是针对XXXXX问题你们看看怎么做，你们大家商量一下，然后你一言，我一嘴，各个提建议，到头什么都没有解决，一份会议记录发给所有人，几乎没有人看。

提意见都很踊跃，具体到谁负责都开始低头，议而不决，会议内容无法落实。

### 会议冲突

会议出现交叉，追尾，就是如同赶饭局。

多任务处理和预约冲突使会议效率变低，公司不得不安排更多会议完成工作。

## 避免议而不决

会开了，问题仍然没有解决，这种议而不决非常普遍。

会议不能议而不决，必须有结论，会议的目的是针对方案细节依次敲定，然后进入到Ticket对应具体负责人。

## 会议记录

很少人看会议记录，没有必要写。

## 会议地点

会议地点可以随时随地，桌面讨论也是会议，不一定非要定会议室。

## 与会人员

与会议议题无关的的人员不应该参加会议。

## 怎样管理会议的时间呢?

取消一定数量的会议或者刻意压缩会议时间并不现实，因为促进合作和作出重大决定都需要开会研究。

我认为可以这样管理，首先规定一个部门或者管理层，一周或者一个月的会议时长。每组织一次会议，便扣除会议时长，并告知剩余时长。

同时制定下面六条规定，可以让许多公司大幅提高会议质量。

1. 会议审批，议题论证，组织会议人员资格，会议时间、与会人员都要严格审查

2. 议程目标明确。比如通知A事项，讨论B事项和决定C事项。会议过程也和会议目标一样简明扼要，与会者专注于完成具体目标。

3. 提前准备。所有周商业计划回顾都必须提前发出，让与会者能在会议前查阅。这一举措大大减少会上分享信息时间。

4. 按时开始。时长一小时会议如果晚5分钟开始，就会浪费8%的会议时间，但很多管理团队在任何其他职责领域都不会允许8%的浪费发生。

5. 敲定事项，会议的目的是针对方案细节依次敲定，什么是应具体负责人，什么时间完成。

6. 尽早结束，尤其是在会议无法得出明确结果的情况下。当公司内部会议效率下降，或与会者准备不充分时，乔布斯会马上“紧急叫停”。有人觉得这样做唐突无礼，但当会议不太可能得出理想结果时，该做法有效制止了时间和金钱浪费。

## 3. 工作报告

### 日报、周报，项目进度汇报

管理人员要求写日报、周报，项目进度汇报真有用吗?

答案是没有任何用处，写日报会影响员工工作，占用工作时间。

不要让员工为了写工作报告而写工作报告。

我从不要求团队写工作报告，因为项目管理软件和工具中Ticket/Issue一幕了然，甘特图已经清楚的标明每个人的工作任务，并且工作都一一确认后发出，对整个项目了如执掌，所以不需要工作报告。

工作报告并不能判断员工的工作量以及是否工作饱和，所以工作报告是不准确的，可以虚构，不实的，而员工为了写报告而写报告，造成时间成本浪费。

### 为什么会出现频繁汇报?

出现这种状况，是因为管理方式落后，管理层不作为，管理层战略不清晰，任务不明确，执行不到位，自己还没有想明白干什么？怎么干？就让下面开始干。

下面干了什么上面不清楚，所以才不断汇报，不断审批，不断修正混乱。

对于战略目标清晰，任务明确，下面只需执行，管理自上而下，一气呵成。上面对项目清清楚楚每个阶段做什么，下面也对自己的工作清清楚楚，规定的时间交付，这样沟通次数减少了，效率提高了，同时内耗减少。

我曾经在另一篇文章中写过《领导力，专业力，管理力》，当管理层缺乏“专业力” 的时候，只能依靠“管理力”项目管理知识和工具，

就会出现无法评估项目的进度，无法掌控任务的完成时间，无法判断员工的能力，最终无法把控项目的风险。

对项目的担心，焦虑，致使他必须时时刻刻听到下面的回报。

**如何才能抛弃汇报制度?**

专业力 > 管理力

如今管理层不在是从前那种管理班毕业，坐在办公室指点江山，喝喝茶，签签字。管理层必须具备“专业力”，专业的人才能干专业的事。

首先高层制定出目标清晰战略，接着讲战略目标分解成一个个里程碑，并分配到中层管理者，中层管理者将里程碑分解成具体的任务，并将任务指派到基层管理者，基层管理者负责组织，协调并带领一线员工按规定的时间完成分配的任务。

高层管理者紧盯路线图，中层管理者紧盯里程碑进度，基层管理者紧盯任务进度，一线员工将进度实时上报到项目管理系统。

分配任务不能 100% 满负荷工作，80% 即可，我们要预留出突发任务，以及随时产生的变更和延期等等。

## 4. 越级和跨部门沟通

谨慎处理越级和跨部门沟通，无论是对上还是对下的越级，都是职场中的忌讳。

对上越级沟通，会让自己的上司暗生不满。

对下的越级沟通，会损害直接下属的权威。

两者都会打乱其部门内部的权利结构和工作部署，影响整个组织架构管理。

## 5. 负面信息处理

任何公司内部都会时不时传出一些负面信息，例如，公司投资项目失败，高层政治斗争，销售业绩受挫，绯闻谣言。

怎样处理这些负面信息呢？答：欺上瞒下。

对下属，听而不说。

对平级，不听不说。

对上级，过滤后说。

# 第 4 章 变更管理(Change Management)

## 1. 什么是变更管理

项目变更管理是指项目组织为适应项目运行过程中与项目相关的各种因素的变化，保证项目目标的实现而对项目计划进行相应的部分变更或全部变更，并按变更后的要求组织项目实施的过程。

## 2. 需求变更

### 为什么会变更

为什么会出现变更呢？常规变更我们先不提（正常的变更例如部署的变化，软件的升级等等），软件开发中存在的变更更多是需求上的变化，为什么会出现需求变更呢？很多开发人员非常困惑。

我们再逐一分析:

1. 岗位的变化
2. 专业的问题
3. 缺乏详细设计文档

岗位的变化，早期软件开发是没有产品这一岗位，那时的需求分析是由系统分析员完成的，同时还要做一个详细设计文档，前者需要一定的技术背景，后者更资深，那时的变更反倒很少。

进入互联网时代，出现了产品这个岗位，这个岗位总体上参差不齐，年龄偏低，经验少，常常工作3~5年，产品这个岗位再大学里并没有这个专业，也就没有一个标准，所以这个行业的人来自五花八门的专业。任何一个岗位都需要时间来积累经验，一个经验不足的产品人员给出的需求往往存在很多问题，甚至不合理，另一方面企业更注重产品部门，导致产品比较强势，开发只能配合，常常是做到一般才发现需求不合理，接下来就是变更了.....

这导致了一个问题，非常有经验的开发人员不再指出产品的不合理之处，按照需求开发，出现问题走变更流程，产品害怕需求变更承担责任，即使需求是错误的也要求开发完成，坚持需求没有问题。有很多功能就不了了之

缺乏详细设计文档，互联网快速变化，导致一个问题没有时间做详细设计文档，软件的生命周期也短，大家都不愿意为了这么短周期

的开发去写设计文档，通常是按照需求直接开发，这也是需求变更频繁的一个原因。

## 3. 拥抱变更

软件开发界唯一永恒不变的主题就是------变化。

工作中我们常常遇到老板或者产品同事来了，告诉我们这个项目不用做了，你做的所有工作都白忙了。

更不能让我们无法接受的是能力不足的产品同事提供的需求，反复修改。

学会接受变更非常重要，你要从公司的角度思考，而非个人的情绪。

人们通常误解是，认定变化不会带来好处，其代价就是成本暴增。基于这一理由，我们必须要事先认真计划，以避免后期的大幅修改，但事与愿违，计划永远赶不上变化快。实际上即使变化发生，所增加的成本通常也可以借助良好的对策来加以抵消 严格控制我们必须完成的内容，并经常性地重新评估，从长期和短期角度问自己：“完成什么才是最重要的”？。

从员工的角度讲，企业支付薪水，是契约关系，让你做什么你就做什么，过多的抱怨也无法变化需求变更的事实。长此以往老板并不会责怪产品能力不行（有可能需求变更就是老板的意思），而是你的工作态度有问题，不配合其他同事的工作。

另一方面我们也需要建立需求提供方的问责制度。将需求变更与绩效考核挂钩。

# 第 5 章 集成管理

是指为确保项目各项工作能够有机地协调和配合所展开的综合性和全局性的项目管理工作和过程。它包括项目集成计划的制定，项目集成计划的实施，项目变动的总体控制等。

我习惯于将配置管理划为集成管理，我认为配置管理是软件集成的一个环节，你别较真，管理学本就没有规范而言，你的模式成功，你就可以著书立说，你就是权威，你就是标准。

## 1. 配置管理

是通过技术或行政手段对软件产品及其开发过程和生命周期进行控制、规范的一系列措施。配置管理的目标是记录软件产品的演化过程，确保软件开发者在软件生命周期中各个阶段都能得到精确的产品配置。

配置管理很多企业将其理解为应用软件的配置文件，这是错误的。所有影响软件正常安装，运行的配置项，都要纳入配置管理。

配置管理范围涵盖软硬件，包括:

1. 硬件：路由器，交换机，防火墙，负载均衡器，服务器.....
2. 系统软件：操作系统，应用服务器，数据库，缓存，消息队列.....
3. 应用软件配置文件：日志，接口，数据库连接池.....

任何项目应该有三套以上配置库，分别是开发，测试，生产

开发配置文件所涉及资源与权限仅限于开发环境，测试配置文件所涉及资源与权限也仅限于测试环境，生产环境也一样，应用程序部署到那个环境，就应该使用那套配置文件

## 2. 为什么持续集成难以普及

90% 的企业实施持续集成最终都失败告终，仅仅流于形式，对工作有个交代。

为什么每个部门都反应持续集成不好用？原因在于这些持续集成是个跨界应用，还有团队内各势力的理解不同，然后不一定配合。我之前的一篇文章谈过的企业多维度架构与多维度管理的问题(有兴趣可以在我的公众号netkiller-ebook中寻找《多维度架构》)。 开发者不懂测试与运维，测试不懂开发与运维，运维不懂开发与测试。开发，测试和运维成为三个孤立领域。实施持续集成需要跨界思维，跨界知识，否则就会出现：

- 开发说：你这个部署有问题，怎么在我本地运行好好的.
- 测试说：测试环境有问题？测试没有问题升级到生产就出问题？我现在还没有测试完，你的那边怎么升级了？
- 运维说：你这种开发不符合规范，无法实现部署。这种部署跟我们的不一样。

开发说：这不是我要的。测试说：这不是我要的。运维说：这不是我要的。

总之，对于不熟悉的领域心里没底，不知道他的内部结构，不知道出现问题怎么解决。持续集成只会给大家制造麻烦。

# 第 6 章 质量管理

## *SQA (Software Quality Assurance) / SCM (Software Configuration Management)*

是为了确保项目达到客户所规定的质量要求所实施的一系列管理过程。它包括质量规划，质量控制和质量保证等。

## 1. 无缺点管理

### zero defects management

由于周末经常外出自驾游，途中会经过东莞、惠州、观澜、大鹏等工业区，哪里的工厂给过一个很深的印象，每个工厂楼顶会有一个巨大的牌匾“已通过ISO 9001”。这让我开始思考以往的质量管理。

我认为质量管理方法可以分为两类:

1. 考察过程
2. 检验结果

传统劳动密集型产业可以采用考察过程（例如ISO9001），制定产生规范，产生预期结果。这种方法对于资本密集型产业或知识密集型产业并不适合。所以另一种检验结果的质量管理办法孕育而生。

简单的说，这种质量管理办法是:

1. 首先制定预期结果，
2. 项目完成后与期望结果对比
3. 输出验收报告
4. 根据验收报告做出处理

这种管理的方法存在很多弊端，工作中你会遇到下面这些问题：

考察结果的质量管理存在的弊端：

1. 无论如何你都不可能把所有预期结果都能考虑到

2. 所做的工作仅仅为了满足预期结果的验收

3. 对已知缺陷视而不见

4. 而对于验收人员，验收报告以外的缺陷，心照不宣

5. 无法预见缺陷，发现缺陷为时已晚,已经到了项目尾声。

举一个例子，国家检验奶粉有一个标准，一些不法企业在奶粉中添加三聚氰胺，可以通过检测，最终酿成惨剧。

无论是考察过程的质量管理还是检验结果的质量管理，这两种管理方式仅仅能做出合格的产品，无法做出精品。

丰田公司的一位高级管理人员说：“我们不应使用全面质量管理，因为这种管理充其量只能让缺点减至10%。如果我们生产400万辆汽车的话，便会有40万人购得一辆带毛病的车，这是生产与用户之间的最大危机，而推行无缺点管理则会消除这种现象。”现在，领先的日本公司逐渐由全面质量管理转向无缺点管理。

无缺点管理的范围已经超出了产品质量范畴

1. 计划缺陷

2. 设计缺陷

3. 产品缺陷

4. 研发缺陷

5. 开发缺陷

6. 工艺缺陷
7. 材料缺陷
8. 流程缺陷
9. 设备缺陷
10. 人的缺陷
11. 生产缺陷
12. 服务缺陷
13. 市场缺陷

## 2. 自动化测试如何破局?

最近面试软件自动化测试工程师，感想颇多。

面试者都来自大厂或大厂外包，华为，oppo，顺丰，沃尔玛，百丽，腾讯，字节…… 开始以为捡到宝了，即使没吃过猪肉也见过猪跑吧，起码参与过自动化测试，面试后大失所望。

面试了很多工作十年的测试工程师仍然在做功能测试，或是功能测试为主，自动化测试打酱油。

什么是打酱油？我们有自动化测试，我们做了。但是自动化对工作的贡献微乎其微，也就是说自动化测试并没有真正为企业带来价值，最后自动化测试脚本不在有人维护，被人遗忘。

问及为什呢自动化测试流于形式？ 实施自动化测试最终摆脱不了失败厄运，会不了了之。每个人都给出无数理由，在我看来是无数借口。

十年前我曾经写过关于自动化测试为什么难以普及的文章，时隔十年，都2021年了，自动化软件测试普及程度跟10几年前情况差不错。究竟问题出在哪里呢?

如果你是管理层，你会发现，自动化测试工程师人在招聘，事在做，钱在花，但是没有成绩。仍然人工测试为主，自动化辅助。

难道无法实现自动化为主，人工为辅吗？此前我在一直在外企工作，为什么外企能做到自动化为主的测试呢？ 我认为有一下几点：

1. 认知的问题
2. 生态问题
3. 技术问题
4. 能力问题
5. 氛围问题

### 认知问题

你问测试人员我们有没有做自动化，答案是:

认为自动化测试替代不了人工测试

这话没毛病，确实不能100%替代，但是自动化测试可以干80%的活。剩下20%人来干。

需求迭代快不适合做自动化，迭代快常常导致自动化脚本跑不通

我不这么认为，我们通过持续集成运行自动化测试脚本，一旦发现流水线测试失败就会立即修复自动化测试脚本。只要紧跟开发，开发动，我就动，联动开发，就可以解决这种问题。写测试脚本的工作量远没有开发的工作量和强度大。更多时候只是修改定位元素标签而已。

人工测试前首先要通过自动化测试，这样可以避免盲目测试。 也就是人工测试走了大部份流程后才发现往下走不通了，此时已经浪费了时间，为什么不让自动化程序去发现问题呢?

## 生态的问题

在国内包括大厂，软件自动化测试处于很低的水平，测试人员水平也相对低于其他团队，例如开发和运维。为什么 DevOps（运维自动化）在国内能风生水起? 因为DevOps 在为企业创造价值。DevOps 降低了IT成本，解决企业面临的众多IT痛点。但是你可能不知道，DevOps 思想刚刚进入国内的时候，很多运维人员是抵触的，他们不相信自动化，他们不敢在生产环境实施自动化，在相当长的几年中，DevOps 也是打酱油的状态，只在开发环境中部署。

管理层重视程度，管理层能力，管理层认知都决定最终结果。

管理层重视自动化测试，但是能力又无法推动。 招聘也存在问题，管理层的认知天花板决定他招聘进来的员工天花板。

即一流人才做面试官，只能招聘到二流人才，二流面试官，只能招聘来三流人才，以此类推。 最终一个乌合之众的测试团队被攒出

来。

## 技术的问题

互联网技术越来越复杂，HTML 4.0 的时候只有 form 表单提交，那时做自动化测试畅通无阻，非常顺利，后来有了ajax 和复杂UI，导致自动化测试难以进行。

很多测试工程师的开发水平仅限于测试，没有从事过前后端开发，遇到问题被卡住，解决不了，就放弃了自动化测试。

## 能力问题

测试团队能力不足是最大问题，多数测试人员的职业生涯规划是失败的，从功能测试走到自动化测试的人非常少。

成为自动化测试工程师，需要三个因素：

1. 自驱力
2. 外驱力
3. 环境因素

自驱力是自我学习的动力，外驱力是外部施压强制员工学习，以满足岗位需要，两种力都具备后还需要有环境，包括学习环境（氛围）应用环境（学以致用），实战机会等等。

## 氛围问题

很多公司的想法是招聘一两个自动化测试人员，更多配置是功能测试。

这个想法就是错误的，从一开始就注定了要以功能测试为主。尤其是当测试组的leader是功能测试者后，他会更坚信自动化测试替代不了人工测试，在自动化测试短期没有成绩的时候，他会本能否定自动化测试，最终将会从自动化专人工。

由于自动化测试人员少，就会有孤独感，遇到问题解决不了，没有人沟通，工作容易被卡住。

所以氛围很重要，要打造学习型团队。提供分工，教练，合作，咨询，培训， 提升团队整体素质。

## 最后

老生常谈，测试部门负责人的认知和格局天花板决定了测试团队的天花板，以及在自动化测试领域能走多远。

## 3. 为什么自动化测试难以推广

2005 第一次接触自动化测试，十年已经过去了，着眼身边的企业，真正实施自动化测试的企业非常少。 大部分企业，测试仍然处在，点鼠标阶段。测试人员通常是验收交付，而没有参与整个软件开发周期。

### 为什么自动化测试难以实施

为什么自动化测试难以实施，我想有几个问题，阻碍了自动测试普及。 其实懂得自动化测试工具的人还是很多的，自动化测试难以实施，并不是缺乏技术人才。Load Runner, QTP 等等很多测试人员都会使用，为什么他们放弃这些工具，改用手动测试呢?

1. 90%测试仍然处在功能测试
2. 很多测试人员没有开发背景
3. 测试角色，没有贯穿整个软件开发周期
4. 各种问题阻碍了自动化脚本
5. 在中国测试人员人力成本太低

随着技术发展，软件的多样性，已经不局限于基于CS结构的GUI, 基于BS浏览器WEB UI。例如目前的安卓系统，苹果IOS系统，微软的Windows Mobile 系统等等。 还有一些非人机交互界面，各种协议/接口，例如json,bson,xml-rpc,soap,mq(message queue)我认为这些都应该纳入自动化测试范畴。 这就需要测试人员具有一定的开发能力，且测试上述内容速要广泛的技术知识支撑。

我认为高级测试工程师，需要具备以下能力:

1. 嗅探器的使用
2. Debug工具
3. 负载均衡技术
4. TCP/IP，UDP
5. 了解各种协议族
6. 渗透/注入

7. 数据库与缓存
8. 搜索引擎
9. 文件系统与存储

就WEB测试而言，涉及的内容就太广泛了，从浏览器->WEB服务器->APP服务器->缓存->数据库，中间会经过各种代理，负载均衡，分布式文件系统等等。

配置这样一个测试环境都已经非常不容易，幸好我们可以采用自动化运维干这件事。

## 是什么阻碍了自动化测试?

1. 各种UI特效
2. 验证码，包括手机，图形，语音.....
3. 浏览器支持
4. 第三方插件(Flash,ActiveX...)
5. 技术封闭

互联网的快速发展 Load Runner, QTP 等等软件，我认为已经跟不上互联网的快速了，他们仍然按照传统周期发布软件更新。 而互联网需要的是快速变化，互联网应用程序开发者，需要体验更多的创新功能，软件软件发布周期至少一年一个版本。真的太慢了。

互联网不断加入的新技术成为了自动化测试障碍，传统软件无法支持这些新技术，甚至向微软这样的企业技术跟进都显得不给力。

Windows Automation 3.0 是非常高大上玩意，但是你在Microsoft官网能找到的资料，少之甚少，我不知道微软的目的何在。

只有 Load Runner, QTP 这些公司与微软合作，才能拿到Windows Automation API。

## 中国测试人员的人力成本

测试人员的薪水在开发团队中应该是处于中下等的。与高级程序员，软件架构师是有很大差距的。这也造成了自动化测试难以实施的原因。

我们需要从高级程序员，软件架构师转测试的高级测试人员。

我们需要黑客级的测试人员！！！

# 第 7 章 风险管理

涉及项目可能遇到各种不确定因素。它包括风险识别，风险量化，制订对策和风险控制等。

## 1. 项目管理绕不开问题

### 开发，测试与运维的关系

开发，测试，运维不是三个独立部门，他们相互紧密联系，但又相互制约:

开发只负责写程序，将运行无误的程序提交至版本库中

开发不能私自将程序交给运维部署，也不能将编译好的程序给运维测试。

测试部只能从版本库提取代码，然后编译，打包，运行，测试

不允许测试部将代码交给运维部部署

避免代码没有经过版本库流入生产环境，线下与线上代码不一致

运维部负责部署应用程序，配置管理，只接受测试部确认无误的版本，部署代码只能从版本库中提取

开发 -> 测试 -> 运维 贯穿始终。

### 压力问题

运维人员在互联网企业是最辛苦的，7*24小时待命，待遇一般。

很多程序写的非常差，BUG非常多，容易出现无响应，崩溃，开发人员一时也解决不了，只能硬上线。

最终这些问题都推给运维，通过运维手段去解决。这就让运维工作压力很大，状态紧绷，如履薄冰，如临深渊，随时随地处理故障。尤其是在双11这样的节日。

## 重速度轻安全

国内互联网企业成长速度是第一位，管理粗放，怎么快，怎么来。

期初公司也是重视安全的，例如账号权限分级管理，操作也有流程，审批。但是企业要速度，各部门都面临压力。从一周一个版本到一天升级一个版本，程序的质量也在下滑，常常升级失败，反复回撤，为了效率甚至线上直接开放给开发人员权限，在线上修改。很多开发人员拥有数据库权限。

另一个问题是人员流动，很多人离职很长一段时间，权限都没有收回，仍能远程进入系统。

国内企业没有花钱请安全顾问公司的习惯，自认为自己有能力解决安全问题。运维人员也不会向公司建议找第三方安全公司，他们担心公司怀疑他们的技术能力。

## 技术实力

你真的认为花高薪聘用的运维人员技术能力很强吗？哪些标着有BAT经验（大企业），渡过金的人运维人员能力很强吗？

很多运维人员仅仅是安装配置的水平，看了几本架构优化的书籍侃侃而谈。哪些在大企业干过的运维人员，也仅仅接触到平台的一角，他的权限根本无法窥视到全貌，无法使公司的运维水平上升到另一高度。

目前互联网企业运维人员普遍比较年轻，都是90后组成，这个年纪正是处于经验积累阶段和学习阶段，只有把系统当成了试验场才能学到东西，有很多事故是运维人员赶时髦，使用了一些新技术，不熟悉的技术（仅仅掌握皮毛），他们管这个叫“踩坑”。现在几乎没有企

业招聘80后做运维，因为他们不能加班，不愿加班，抗压性/服从性较差。

由于运维技术能力，眼界见识。不是他们没有做（备份，快照，防篡改技术，防删除技术），是他们是想不出来解决方案的。

运维人员也不会建议公司请第三方公司或技术顾问，一是要花钱，公司不同意。二是，他们怕公司怀疑他们的技术能力。就这样很多问题被尘封了，只有出现重大事故的时候你才意识到他们能力不行。

**测试问题**

首先回答你测试问题，测试人员的水平是整个团队中能力最差的，企业对技术人员重视重读顺序是，开发第一，运维第二，测试第三。人肉测试与自动化测试80%:20%的比例，掌握自动化测试高级测试人员薪水不低，很多企业不愿意或出于成本考虑，国内人力成本便宜大量采用人肉测试。无论是人肉还是自动化距离理想期望的测试结果还差的远。

自动化测试实施常常遇到很多问题（安全，技术，管理等等），例如

自动化测试攻城狮的技术水平有限（仅仅掌握了皮毛）需求的频繁变更让自动化测试脚本成为负担，最后不了了之复杂的技术栈，手机验证，图像验证，定位信息，摄像头操作，指纹，传感器，二进制位操作..... 一般测试攻城狮搞不了，卡在某个点过不去，最终放弃

导致自动化测试无法继续实施下去，一般除非是在职的高级软件攻城狮转测试攻城狮，他对需求清楚，功能清楚，协议清楚，数据清楚，业务逻辑走向清楚，否则一般的测试攻城狮根本搞不定这么复杂的测试脚本。企业也不舍得让这个开发主力转到测试部门去。

由于系统过于庞大，人肉测试所有功能是不现实的，所以常常是只测试新加入的功能或测试有依赖的功能。实际工作中，程序猿改动一行代码，牵一发动全身，尤其是那些复用的代码，加之程序猿人员

流动，当时负责该功能人早已经离职（平均三年离职），你不知道改动一处，会有什么影响，所以程序猿很抵触修改老代码和其他人写的代码，宁可自己重新写，这又造成了脏代码，代码不断膨胀，反证三年后他也可能离职，让新来的程序猿接盘。所以常常是新加入的功能测试OK，隐藏很深的一个功能出现的故障，测试无法覆盖到所有功能。

很多时候故障不是测试人员发现的，多半是用户发现的，用户投诉到客服，客服反馈给技术部。

运维问题

接下来回答你运维的问题，运维手段多着能，通过运维手段可以解决很多BUG问题。常用手段，例如万能重启，主备策略，负载均衡，快照，权限控制。我给你举几个例子。

如果代码容易崩溃，那就做一个自动重启的脚本，发现崩溃，立马重启。

如果代码并发有问题，例如做一个并发 10000的程序，结果程序猿水平有限1000并发就崩溃。就用负载均衡堆硬件，用10台服务器，先抗着

代码有漏洞，黑客入侵，修改页面植入木马。运维就通过权限限制文件只能读，不能修改。

代码有漏洞，导致黑客修改数据库数据。运维有多重解决方案，例如去掉修改权限，临时增加触发器，修改数据抛出错误中止操作

很多手段，这些手段可以为开发人员争取代码修复的时间

## 2. 程序猿说的「优化」是什么意思?

产品经理，程序猿口中的优化是什么意思？你是否想知道员工在干什么？是不是常常在会议上听到，产品经理和程序猿说在优化产品?

确实有20%的优化工作确实是在优化产品，产品经理在打磨产品，程序猿在优化性能。

但80%的优化工作其实是在修修补补，这些工作不创造任何价值，也是可以避免的。

### 经验和能力不足

成为一名优秀的攻城狮，至少需要 9-15年，攻城狮的经验是从失败中积累的，这个过程是漫长的，需要时间和参与项目的数量。如果一名攻城狮进入公司后只做一个项目，积累经验是有限的，攻城狮参与的项目越多积累经验就越多。

从招聘网站上的数据看，目前企业招聘追究性价比，会选择3-5年工作经验的员工，这些员工的经验积累不足，能力有限。很多企业为了弥补这些员工的经验和能力不足还会配置一个岗位叫架构师，一般也是5-8年工作经验。

成为一名优秀的架构师首先需要有足够的项目练手，目前市面上的很多所谓架构师都达不到架构师的水平，他们更多只是熟手程序猿。在我看来一个团队中仅仅有一两名架构师是不足以提高团队的整体素质的。

产品经理也是做的产品越多经验越丰富。

一个经验不足的产品经理在做项目的时候，设计出的产品存在各种缺陷，常常是提交给开发人员后，发现不对，需要立即修改。这也是产品跟开发人员的主要冲突和矛盾。老道的产品经理就会先不做修改，事后一段时间后再提交一次优化，把这个遗留问题解决掉。

开发人员也一样，经验不足，考虑不周，就会留下bug，甚至是刚刚提交代码，立马想到问题所在，必须在做一次修改。这也是开发人员跟测试人员的主要冲突和矛盾。测试人员刚刚测试完一轮，程序猿修改了代码，需要重新再测试一遍。老油条，他会评估BUG影响的范围，如果不影响正常使用或者达不到触发条件，就会先不修改代码，如果测试人员没有发现 这个BUG，就先上线，待日后优化掉。

对于有10年以上经验的老码农写程序根本不用什么架构师来优化。他写程序的时候已经考虑了所有可能的性能，数据库结构，缓存策略，性能，扩展性，高可用.....

**给前人擦屁股**

国内企业的人流量非常大，超过3年的老员工非常少，工作岗位细分，每个人都有自己负责的工作，即使是老员工也不愿意去维护其他同事写的代码，最终公司里没有一个人熟悉整个项目，总是那里出问题，临时去翻看代码。

团队需要不断补充新鲜血液，新员工至少三个月的时间才能独立工作，新员工对老代码不熟悉，问了一圈也得不到答案，于是他想还不如重写那些老代码。的确有可能代码被重写后性能提高了，更多的是修改旧代码产生的新BUG。

这个过程叫优化老代码

**先盖楼，后打地基**

很多互联网企业追求快，是先把项目做起来再说，随着业务的增长，发现系统已经无法满足，这时就期望高价招聘一个架构师解决系统存在的问题。

这种先盖楼，后打地基的做法，后期产生的优化是无穷无尽的，直到整个系统被新代码被替换掉，或是推倒重来。如果中途架构师离职，新来的架构师又是一套新操作。

**使用新技术和不熟悉的技术**

网上常常看到一些踩坑的文章，意思就是在实施某项目过程中遇到的问题。

使用新技术和不熟悉的技术，无法预判产生的影响和结果，才一次又一次的优化之后，对技术越来越熟悉，最终解决了问题。

在我看来之所以出现“踩坑”问题，是你让一个没有经验的人负责一个复杂的项目，搞出一堆问题，生产环境成了实验场。

不去追究责任，反倒鼓励员工的经验总结。对于员工来说倒是个非常好的练手机会，对于公司来说是各种损失。这种作风再当下比比皆是，抢险救灾领导被表扬，而不追究为什么发生事故。

## 最后总结

之所以现在网上各种优化类的文章，只能说明程序猿能力不足，经验不足，对技术掌握不全面，不深入。才会留那么多问题带日后解决，这些问题应该在设计之初就考虑，开发中就发现，不应该等到上线后再去验证。

软件的犯错成本太低了，你见过做硬件的天天优化吗?

## 3. 制度、流程和规范的误区

几乎所有的技术企业都会重视技术规范，为此制定各种规范，并要求员工严格执行。同时员工会想出各种对策，就这样形成了潜规则。

这些规范就好比“请保持室内卫生，不准乱丢垃圾，禁止随地吐痰，不要闯红灯” 一样没起到的实质作用。

管理层擅长制定乌托邦式的流程与规范，随便拿出一条都堪称完美，无懈可击，但没有考虑到执行结果，流程规范在执行过程中每个环节都会出现问题。任何一个环节出现问题就如同多米诺骨牌，造成连锁反应，最终无法控制。

我19年的职业生涯中在不同的公司任职过，几乎每到一家公司都会遇到各种规范，随着职业发展最后我也成为了规范的制定者，也曾经主持制定过开发规范，运维规范，测试规范等等。

我做过很多规范，文档无数，技术人员根本不会去看，通过开会向下传达，开会的人根本没有心思理会你的规范，规范执行阻力是很大的，效果也差。

终于有一天我意识问题的存在，开始反思，是否需要制定这些规范？制定流程规范的目的是什么？

有些强制的规范可以通过一些技术手段，避免出现。不会出现也就无需规范！

### 故事一

例如下面一个小故事，公司某部因为将开发数月的代码丢失了，导致测试无法进行，领导大发雷霆，某管理层制定了下面的规范，大意为。

1. 定期备份机制
2. 代码注释要求
3. 代码访问需要更高层的批准
4. 详细的部署文档
等等

我认为源码管理主要有两种手段，技术手段与管理手段。

我先谈谈管理手段：例如通常通过规章制度，责任追究等等手段，要求员工达到规范标准，但通常执行力都会打折，无法达到预期，人的不稳定性因素太多。往往发现员工没有按照规范操作为时已晚，将该员工辞退也无法挽回公司的损失。

就如公司规章制度写的清清楚楚，要求员工提交代码到版本库，但各种原因没有被执行，当代码丢失，从上至下追究责任，公司的损失无法挽回。

所以我主张技术手段：例如源码如果发布到线上，必须经过版本库，只能使用自动部署，不允许程序员私自将代码交给运维手工部署。另外发布代码的同事，可以不提供生产服务器登陆权限，他只能通过工具发布代码。

部署流程如下：

源码(程序员) 提交到development 分支UAT阶段 ----> 合并到 testing 分支Beta阶段（主管合并，程序员没有权限）------> master 分支(主管合并) -----> 自动部署系统(运维) ----> 生产服务器。

这样通过技术手段防止了代码因员工离职，硬盘损坏等等原因，导致代码丢失的可能。

代码发布者也无需对照部署文档，手动登陆服务器逐条按照部署说明书操作，防止了人员误操作，也提高了部署效率，节省了人力成本，通常在5分钟之内可以完成所有部署。

## 故事二

我再来举另外一个例子，就是开发中的编码规范，很多软件企业都有是不是?

例如要求程序员:
if
(){}
要写成
if ()
{
...
}
等等要求不一一列举，甚至组织代码评审解决编码规范问题。

我的建议为什么不在IDE上设置自动格式化，或者在svn/git提交的时候通过hook调用格式化程序。甚至可以做到，在提交的时候编译，编译不通过，就提交不上去。

## 故事三

管理层要求运维每天发送服务器状态报告，运维人员需要登录每个服务器或者从cacti等工具中获得服务器运行状态数据，然后制作一个报告文档，每天给各位发送一次。

运维需要一个专职人员做这个报告，这种报告几乎没有人看，就像“人民日报” 人民从来不看。

当运维事故该出现的时候还是会出现，老板一个一个骂，扣工资，扣奖金，运维觉得委屈，公司受到损失。平日里的这些工作并不能避免运维事故，也不能改善运维工作。

## 故事四

在举一个例子，运维工作要求备份数据，制定规范，A员工负责备份，B 员工负责检查A员工的备份。这个流程没有任何问题。

结果两年以后出事了，需要恢复数据，发现A没有备份，而B在一年前就再没有检查A的工作。

起初前一年还是按流程备份，后来A发现B不再严格检查工作，备份工作逐渐减少，最后停止了备份，一直相安无事，直到事发。

## 故事五

我曾经遇到过一个兢兢业业的管理者，他制定规范，要求值班的同事7*24小时，每间隔一定的时间做一次操作，验证系统正常运行，以便能够第一时间通知运维处理故障。

值班的同事偶尔偷懒，他就半夜起来监控他们工作。一个打工者能做到如此，真让人佩服。

但是故障的频率依然没有改观，运维仍是每天疲于奔命的救火。

但是我们有更好的方法，真的不必如此操劳且效率低下。

## 总结

上面的几个故事是一个无休止的死循环

```
出问题 -> 领导发火 -> 行政处分 -> 制定规范 -> 执行规范 -> 慢慢淡忘 -> 后续无人跟进 -> 石沉大海 -> 继续出问题。
```

流程与规范的制定需要需要满足几个条件：简单，易掌握，易执行，可重复执行

员工考虑的是尽快完成工作，规范不应成为完成工作的负担。

只有机器人才能100%执行流程，任何由人执行的流程规范都不可能做到100%执行，在军队中即使是严格训练过的士兵也常常犯错。

很多管理者将其归咎为“执行力”弱，我并不这么认为。有些犯错并不是执行力问题，也不是敬业度问题，可能需要从心理学角度解释。这是我在阅读几本心理学著作后发现的。

我觉得很多规范是形式主义。我一向主张实用主义。

通过技术手段可能避免很多没有意义规范，开发自动化，测试自动化，运维自动化，这是趋势也是我的努力的目标。

## 案例分析：怎样避免电梯伤人事件再发生

电梯与我们的生活是密不可分，生活在城市，很难想象如果没有电梯会怎么样。

### 事件起因、经过和结果

几年前有一起电梯（手扶梯）伤人事件，事情的起因是这样的，一家商场电梯维修了一半，为了不影响商场运业，维修人员几把盖板临时盖上，供顾客使用。

事件的经过是这样的，一位妈妈带着孩子来到商场购物，一脚踩在电梯盖板上，盖板突然反转，顾客直接掉进电梯，母亲顺势将孩子抛出，然后就被搅入传送机构，血肉模糊，最后惨死在电梯里。

最后我们看看处理结果，事件发生之后，新闻与评论无非是指责商场，指责电梯维保，指责生产厂家，探讨电梯行业等等舆论层面。

商场马上启动公关，进入司法理赔谈判等等。

我们看到舆论只会去职责，从未有人提出过解决方案。

怎样彻底解决电梯伤人，让这种事件不能再次发生呢?

社会舆论一边倒“管理层面”，也就是提高商场服务意识，工作人员敬业程度等等，但这些因素是不可控。

另一种是“技术层面”，通过技术手段解决电梯安全问题。在我看来此次事件是电梯设计缺陷所致。

**电梯伤人事件解决方案.** 现在我来说说技术层面“电梯伤人事件解决方案”。 首先在电梯盖板下面安装一个按压开关，当盖板安装好后，开关处于下压状态，电路导通，通过继电器，开电梯电源。 一旦盖板被打开，电梯立即停止供电。进一步延伸功能，当盖板被打开，停止电梯供电，同时触发警报。 怎么样？成本不足5元钱，安装两个零件彻底解决了安全问题， 在现有的电梯设备上改装，只需要几个小时就可以完成。 这种技术层面的解决方案，远比媒体/舆论期望的“管理层面解决方案”要好。 我们可以在很多方面作出技术改善，安装各种传感器，实现工作状态实时监控，故障报警，自动响应等等。

当下需要重点解决的是下面几点:

**实时监控.** 目前电梯都是每年巡检一次，使用过程中出现问题再找厂商上面维修。这样的检查是远远不够的，往往电梯出现问题，轻者将人困在其中，重者就会伤人。

例如我们可以采用很多传感器技术监控电梯:

1. 电梯门安全，目前很多电梯又具备防止夹人的功能，老式通过限位开关的那种逐渐淘汰，新的基本采用光传感器，我遇到过几次故障，例如反复无法关门。 我认为可能做很多改进，例如反复关门超过3~5次就发出报警，停止电梯运行。
2. 红外空间扫锚，当电梯停止运行（无人乘坐），扫描电梯室内热源，热源超过5分钟，发出警报。这种情况可能是小孩被困或者老人晕倒。

**远程监控.** 远程监控是指电梯传感器收集的数据，实时或定时发送到厂家，用作数据分析，归档备份

**故障报警.** 目前主要靠人来判断故障，然后决定是否需要通知厂商维修。 有了上面的监控数据，电梯出现故障厂家第一时间知道，并上门维修。 甚至厂家可以可通收集的监控数据，通过技术人员分析，决定是否远程遥闭电梯。

**自动响应.** 是电梯智能化的标志，电梯能根据预先设定的程序，对轻微故障，作出自动响应。如上面谈到的“电梯门安全”，“红外空间扫锚”。

**电梯行业应该引入4S概念.** 目前电梯市场十分混乱，我认为电梯行业应该引入4S概念。

- 是一种集销售（Sale）
- 零配件（Sparepart）
- 售后服务（Service）
- 信息反馈（Survey）

四位一体的销售企业，让电梯的销售，配件，售后保养，以及问题反馈流程化，标准化。

## 4. 因果图在运维工作中的应用

### 故障树分析(Fault Tree Analysis，FTA)



### 什么是因果图

鱼骨图，又名因果图，是一种发现问题“根本原因”的分析方法，我们将影响问题的因素与特性，按相互关联性整理而成的层次分明、条理清楚，并标出重要因素的图形就叫特性要因图、特性原因图。因其形状如鱼骨，所以又叫鱼骨图（以下称鱼骨图），它是一种透过现象看本质的分析方法。鱼骨图由日本管理大师石川馨先生所发明，故又名石川图。鱼骨图是一种发现问题“根本原因”的方法，它也可以称之为“Ishikawa”或者“因果图”。其特点是简捷实用，深入直观。它看上去有些像鱼骨，问题或缺陷（即后果）标在“鱼头”外。在鱼骨上长出鱼刺，上面按出现机会多寡列出产生问题的可能原因，有助于说明各个原因之间是如何相互影响的。

### 为什么使用因果图

在运维工作中，我们经常使用 过程中“故障树分析”，它主要用于出现故障时找到问题的源头。而因果图则是保证7*24运维有哪些影响因素。我认为将“故障树分析”与“因果图”互补使用更能解决运维中遇到的各种问题。

“因果图”能未雨绸缪，“故障树分析”可以亡羊补牢。

### 何时使用因果图

我认为任何环节都能使用因果图帮我们我们改善IT运维工作。

### 何处使用因果图

例如项目的部署先，部署中，部署后等等每个环节。部署前拿出因果图由为重要。

### 谁来负责制作因果图

问题总是受到一些因素的影响，我们通过头脑风暴法找出这些因素，并将它们与影响因素的特性值，整理，分类，层次化。

**注意**

我不喜欢开茶话会（中国式会议），参与人员应该每个人在会议前找出问题因素，会议中拿出问题的因素提交给会议主持者，会议目的是将每个人寻找出的影响问题的因素整理成为鱼骨图，而不是在会议上讨论找问题因素。

### 怎样使用因果图

下面我们提供一个鱼骨图分析案例



上图我们看到保障系统7*24小时运行有哪些因素印象，网站分为几个部分组成

网站

1. www.exampel.com 网站入口，主要是静态内容，或者已经将动态静态化。
2. img.exampel.com 图片服务器
3. acc.example.com, api.example.com 动态服务器
4. cch.exampel.com 缓存服务器， db.example.com 数据库服务器
5. mq.example.com 消息服务器

我通常给每个服务器指定一个主机名，有些事DNS解析的，有些事hosts文件设置例如 cch.example.com, db.example.com 不需要DNS解析。

现在我们分别解释每个节点与问题的影响因素，这里仅仅给出的一个简单的例子，也只能让你对因果图有个入门了解。

#### www.example.com, img.example.com

影响的因素主要是web服务器，IP地址，80端口，防火墙设置，DNS 解析等等

#### acc.example.com, api.example.com

除了web服务器，IP地址，80端口，防火墙设置，DNS 解析。他的影响因素包括

PHP版本，PHP扩展，PHP配置文件

#### cch.exampel.com, mq.exampe.com, db.example.com

影响的因素是防火墙，端口，数据库同步等等...

## 5. Incident Management(突发事件管理)

### 突发事件处理流程



### 事件处理方式

很多人顺着简单而直接的“事件 > 反应 > 结果” 连锁行为来反应。

遇有状况发生，第一时间不加思索地反映，造成的结果不但于事无补甚至造成二次故障。

这种不由自主或未经过思考的反应有时会导致灾难性的后果。

更好的选择是:

```
事件 -> 结果 -> 反应
```

说白了就是，遇事想清楚再动手不迟。

## 6. 监控的艺术

### 背景

每个企业都意识到监控工作的重要性，但80%企业的监控工作仍然处在监控的初级阶段。

什么是初级阶段呢?

1. 被动监控，故障发生运维人员永远不是第一个发现故障的人
2. 监控IP地址与TCP端口，很多时候HTTP 80端口正常接受请求，但WEB服务器不能正常工作。
3. 人肉监控（人肉运维），采用人海战术，桌面摆放很多显示器，甚至投影仪，要求监控者盯着各种仪表板界面，制定各种工作流程以及KPI考核监控人员。
4. 人肉测试，要求监控人员每间隔几分钟人工操作一次，以确认系统正常工作，例如（没15分钟登陆一次，下一笔顶单，做一次支付等等）。
5. 万能的重启，定其重启所有的服务器。

什么是中级阶段呢?

1. 报警：手机短信更靠谱，因为手机随身携带（邮件不算，邮件到达速度慢，各种因素不稳定）
2. 监控服务：探测服务的可用性，而不是仅仅监控端口，注意我是指私有协议的监控（HTTP，SMTP,FTP,MySQL 不算在内）
3. 故障分析：通过日志与调试工具分析软件BUG，指导开发人员改善软件质量，使其故障不会再次发生，达到不用restart重启方式解决故障
4. 半自动化测试

什么是高级阶段呢?

1. 我认为高级阶段是监控与灾备系统打通融合一体。

2. 除此之外监控与开发密切相关，在开发阶段需要为监控数据采集做铺垫，每开发一个新功能就要想到未来这个功能是否需要监控，怎样监控。
3. 数据前期采集与数据挖掘非常重要，监控不仅能做软件与硬件的性能分析，还能提供决策支持，这里又涉及了BI。
4. 除了监控，另一个息息相关的是自动故障转移，有兴趣可以看看我的其他文章 http://netkiller.github.io/journal/

监控从初级向中继再到高级，是转被动到主动，从人工到自动化。

监控不应该局限在硬件与服务，还应该延伸到业务领域。

## 概述

你在百度上搜索监控多半是一些开源或商业软件的安装配置指南。这些文章中会告诉你怎样监控CPU、内存、硬盘空间以及网络IP地址与端口号码。

开源软件无非是 Nagios, Cacti, Mrtg, Zibbix ..... 这些软件在我的电子出书《Netkiller Monitoring 手札》中都有详细说明安装与配置方法。

商业软件也有很多如 SolarWinds， Whit's Up，PRTG .....

所有的服务器，网络设备，监控你都做了，那么按照我上面的监控分级，你处于监控的那个阶段?

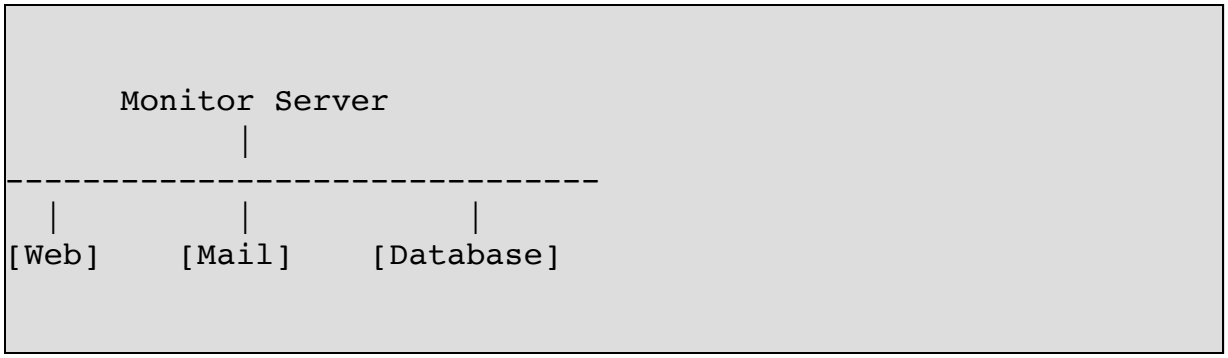
## 怎样监控

监控都有哪些手段跟方式呢?

### 卫星监测

中心卫星站为中心站点向外放射，通常是通过IP地址访问远程主机，实施监控，常用方法是SNMP,SSH,以及各种Agent(代理)，方式是

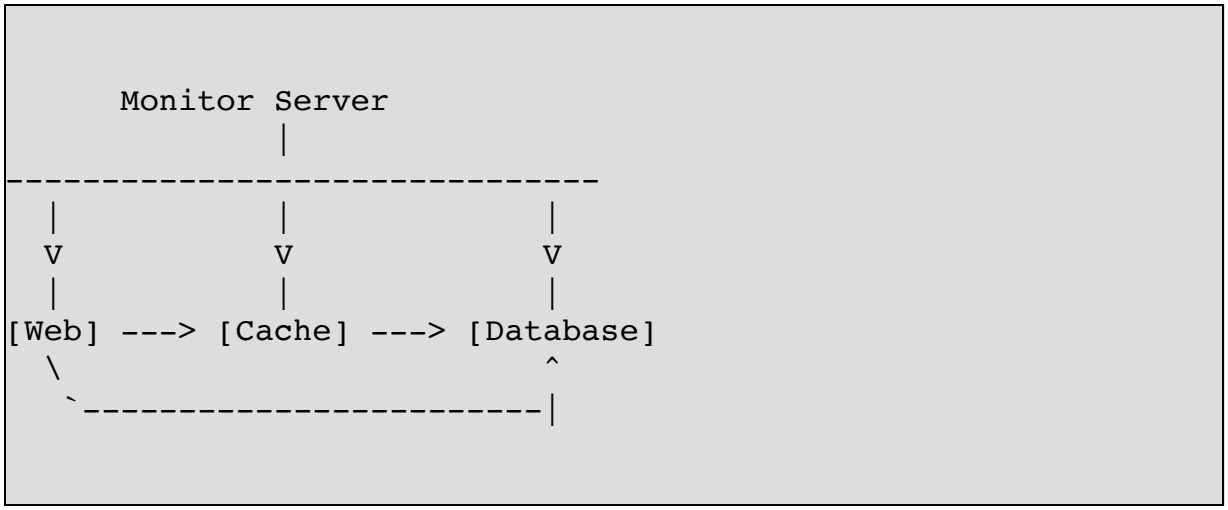
请求然后接收返回结果，通过结果判断主机状态。




以监控服务器为中心，星型散射连接其他监控节点，没有什么优点，缺点是Web跟Mail节点的通信没有监控

逐级诊断

一级一级的向下探测，寻找故障点，需要在各个节点埋探针。




首先监控服务器跟星型拓扑一样监控，再让Web节点去访问Cache节点然后返回监控结果，以此类推，让Cache节点访问Database, 让Web访问Database节点。

将所有业务逻辑都逐一模拟一次，任何一个环节出现问题，立即发出警告。

模拟人工

这里主要监控服务是否可用，可以检查软件的工作情况，涉及测试环节。

通过自动化测试工具辅助监控，例如模拟鼠标点击，键盘输入，可以监控图形界面程序与网页程序。

Windows 监控可以通过 Windows Automation API实现，通过程序控制，能够模拟人工操作软件，实现操作匹配返回结果实现自动化监控

Web页面监控的方案就太多了，比较经典的是Webdriver衍生出的各种工具Selenium - Web Browser Automation最为出名。我通过这个工具模拟用户操作，例如用户注册，登陆，发帖，下单等等，然后匹配返回结果实现自动化监控与报警

#### 数据分析

通过数据分析，将故障消灭在故障发生前。举一个例子，开发人员忘记设置redis 时间，虽然程序一直完好工作，但redis内存不断增长，总一天会出现故障。

我们通过采集redis状态信息，分析一段时间内数据变化发现了这个问题。

#### 监控与开发

谈到监控很多人认为这是运维的事情，实则不然，不懂运维的测试不是好开发。

开发过程中需要考虑到监控，例如Nginx的status模块， MySQL的show status命令, Redis的info命令，都是为监控预留的。那么你开发的程序是否考虑到了监控这块呢?

你可以通过日志形式或者管道，再或者Socket将程序的运行状态提供给监控采集程序。

## 总结

好的监控的能让你对系统了如指掌，做到心里有数。有数据才好说话。

# 第 8 章 成本管理

为了保证完成项目的实际成本、费用不超过预算成本、费用的管理过程。它包括资源的配置，成本、费用的预算以及费用的控制等项工作。

## 1. 警惕IT黑洞

### 什么是IT黑洞

IT黑洞是指企业在利用信息技术进行经营管理时，巨额的软硬件或软件投资并不能给企业带来预期的管理效率，企业在这方面的投资好像陷入一个“黑洞”的现象。

产生IT黑洞原因是，运维管理层没有能力解决生产中遇到的问题，害怕承担责任，从而将风险转嫁给第三方。试图说服企业，上了这些硬件就能保证生产安全稳定。

如此一来IT预算成倍增长，企业几乎将所有的技术都应用的生产环境中。

使用最新的防火墙，转发能力最强的路由器，吞吐最强的交换机，昂贵的负载均衡，高大上的SAN区域存储，去重复压缩备份，实时备份与恢复，主流的服务器甚至小型机。

我曾经写过一个段子调侃一下IT黑洞:

```
要是咱做首席架构师
一定要的选Java
选最好的五星级机房
万兆骨干以太网直接接入
至少百十来个机柜吧
什么防火墙，路由器，交换机，负载均衡呀
能给他接的全给他接上
楼上有健身房，楼内有游戏室
```

```
一进门儿，甭管有事儿没事儿都得跟人家说
may i help you sir?
一口地道的英国伦敦腔儿
倍儿有面子
什么 hibernate, struts, spring,ActiveMQ, kafka, spark 全都给我装上
别说代码多少行,光 xml 配置文件就100M多兆
一个 EMC 存储放那儿，干啥，存储Log4J的日志
一年服务费就得几万美金
这样一个系统，你猜得多少码农开发，得，光累死的就10好几个
这样的系统能用吗？当然不能
你还得找个外包公司，一打 Application Server管理员，18M认证的那种 ，
24*365
在招聘几十个运维的，二十四小时候着
系统光启动就得好几天
周围的公司不是Hadoop就是BigTable
你要是用PHP，MySQL都不好意思跟人家打招呼
就是一个字儿---“贵”
一天的网络流量得花个万八千的
你说这样的开发，码畜一天的多少钱
我觉得怎么着也得四千美金吧
四千美金？！那是成本
八千美金起
你别嫌贵，还不打折
你得研究老板的心理
愿意掏几千美金租五星级机房的企业
根本不在乎再多掏两千
什么叫土豪企业你知道吗?
土豪企业就是
买什么东西都买最贵的，不买最好的
所以，我们做项目的口号就是
不求最好，但求最贵.
```

设备该上的都上了，事故该来的仍然会来，使用第三方企业提供的解决方案未必靠谱，昂贵的硬件与软件投资并没有为创造出应有的价值。

起初IBM,EMC,VMware,Oracle,Microsoft,SAP 等等企业针对不同行业提出很多解决方案，例如ERP,CRM,零售业，航空业解决方案等等。传统行业经过半个世纪发展，提炼，已经有一个完整、完善、科学的管理方法，流程明确，如制造业有ISO标准，这些软件企业经过漫长

需求收集整理最终针对不同行业开发出完善的产品与行业解决方案。这些方案在传统企业是成功的，我们可以看到IBM,SAP,Oracle在各种传统行业取得了辉煌的成就。

然而，进入90年代，这种传统企业管理模式越来越跟不上时代的步伐，新经济时代的到来致使企业所处的时代背景和竞争环境发生了根本性的变化。 软件供应商的传统解决方案也不适合当下的企业，日益显露出其弊端。传统软件开发方式，一年过半年发布一个版本已经不能适应互联网时代，我们需要一周甚至每天一个版本。尤其在当前中国互联网大环境下，每天可能频繁更新数个版本到生产环境。

1. 技术创新持续进行、速度不断加快，企业竞争优势主要来自创新。随着科技的日新月异，人们的生活节奏越来越快以及对个性的追求，产品的生命周期不断缩短。 与工业经济时代不同，创新(Innovation)成为知识经济时代的首要目标。人们通过有计划的、连续不断的创新来赢得市场的竞争优势。 也就是说在知识经济时代创新不再具有一定的阶段性，产品变化不再具有相对稳定性的特点，从而企业通过将产品生产分解再分解，使生产的每一步骤规范化和简单化，并通过规模化大生产降低生产成本，获得市场竞争优势的历史成为过去。

2. 顾客需求瞬息万变，产品周期不断缩短。在知识经济时代，那种”生产什么就卖什么”的时代已经一去不复返了。如今的”买方市场”使顾客的选择范围大大拓宽，也使得他们对产品的期望值在不断提高，他们不再满足于合理的价格，而且还要追求产品的个性化，企业往往要根据顾客的需求”量体裁衣”。同时，市场竞争加剧，大量的替代产品使得任何一家企业都无法垄断市场，而贸易壁垒的取消还意味着顾客不仅仅可以从本国产品还能从外国产品中寻求其最佳利益，于是顾客不再有耐心为某一种产品而长时间地等待了。 企业如果不能即时对市场需求变化做出快速响应，不能在短时间内开发、生产并销售出其产品，企业就会被淘汰出局。

3. 竞争空间不断扩大，激烈程度不断加剧。随着全球经济一体化，企业竞争将不再受地域限制，任何企业都要承受来自国际化企业发展的竞争压力。另一方面，中小企业如雨后春笋，进行专

业化灵活多便的生产或服务，并以其低成本运营对规模化企业高成本运营直接产生竞争威胁。

## IT黑洞产生的原因分析

### 人的因素

1. 企业重管理，轻技术。趋向于管理层把人管好，而不是充分授权技术人员，重视技术研发。

2. 管理层没有能力解决生产中遇到的问题

3. 害怕承担责任，如果在生产环境使用MySQL出了问题怎么办，谁能承担责任？不如使用Oracle，出现问题厂家上门解决，将责任转嫁给厂商。

4. 求稳心态，不做事就不会出事，不要在我任上出事，评价下属工作的标准是办事的准确度如何，任何冒险与创新的行为都是不受欢迎的。 因此极大地抑制了成员自我决策的积极性与创造性。导致员工技能单一，适应性差，员工缺乏积极性、主动性、责任感差，致使工作和服务质量下降。

### 来自组织架构的问题

传统的企业组织理论告诉我们，当组织规模扩大到一定程度，必须通过增加管理层次来保证有效领导。在企业规模一定的情况下，管理幅度与管理层次成反比。当企业发展到一定规模后，这种管理体制的弊端就突显出来。

组织层次过多，各部门按专业职能划分，组织机构臃肿，助长官僚作风 这些都是出现IT黑洞的主要原因。各部门只关心本部门的工作，并以达到上级部门满意为准，缺乏合作与服务意识。 各部门往往会从自己局部的利益出发，精心构思自己的行为，使自己的目标凌驾于整个组织的目标之上。这种分散主义和利益分歧，或许能够实现局部利益的提高，但却弱化了整个组织的功效。

延伸阅读《Netkiller 系列 手札》

# 第 9 章 人力资源管理

## *Human Resources*

是为了保证所有项目关系人的能力和积极性都得到最有效地发挥和利用所做的一系列管理措施。它包括组织的规划、团队的建设、人员的选聘和项目团队建设等一系列工作。

更多请参考“人力资源管理” 相关章节。

## 1. 面试流程

简历筛选 -> 电话面试 -> 面试(face to face) -> 确定录用

注意：没有笔试这项，没有必要，可以放在面谈中，作为互动的一部分。

1. 不要盲目给应聘者打电话通知他过来面试，HR部门首先应对简历做初步筛选，然后把简历email给技术部门负责人。

2. 技术这边对简历进行审核，然后通知HR部门确认电话面试时间，并了解对方基本情况。

3. 配合HR部门做好电话面试，了解应聘者的技术水平，能力，是否可是胜任该职位。如果OK，通知面谈时间，地点，并发一封正式Email邀请函。

4. 面谈

5. 后面工作交给HR部门负责

## 2. 技术面试

**如何判断技术应聘者的段位?**

3:3:3 法则（本人发明的）什么是 3:3:3 法则？通常人们掌握一个领域的技能或知识需要最少 10000小时（10000小时定律[1]）也就是 3年，一般人的职业发展路径和规划也遵循这个原则。毕业进入社会，第一个三年是迷茫期，不清楚自己的发展方向，不清楚自己适合什么领域，经过10000小时过回到自己的方向和领域。 有了方向之后，确立自己的目标，第二个三年开始发力，集中提升自己。此时已经工作六年，完全掌握了工作所需的技能，接下来三年就是技术输出了，为企业创造价值。

关于面试题目

面试题目重点是对问题解决能力，和思路，方法，并没有正确答案。

解决问题的能力更重要。

---

[1] 美国两位畅销书作家，丹尼尔·科伊尔的 *《一万小时天才理论》* 与马尔科姆·格拉德韦尔的一本类似“成功学”的书 *《异类》* ，其核心都是“一万小时定律”，就是不管你做什么事情，只要坚持一万小时，基本上都可以成为该领域的专家。

# 3. 网络工程师面试题

## Junior

```
WAN/LAN：
- ADSL: PPPoE
- WiFi Router
- Cisco Switch 29xx, 35xx, 75xx; Router 28xx; VLAN
- IP Address A/B/C and netmask; Mac Address
- Route table
- Gateway / Proxy
- DHCP Wins DNS
- Firewall

Hardware:
- DDR I,II,III
- PCI-E
- USB / 1394
- IDE 版本，支持多少个设备，带宽
- STAT 版本，支持多少个设备，带宽
- Scsi 版本，支持多少个设备，带宽
- RAID Adapter 版载与实体卡的不同，引导过程，什么是hotswap，故障排除，
怎样创建RAID 0,1,5,10,50 它们之间不同
- BIOS,SCSI,RAID 固件升级

Operation System:
- Window
* 远程桌面链接数限制
* IIS 服务器
* Exchange 配置
- Linux

Network Application：
- Active Directory
-
HTTP/HTTPS, FTP, SMTP, POP, IMAP, NNTP
- SNMP 协议
- DNS 配置
- DHCP 配置
```

```
Security
- System Security
- Network Security

Phone System
- Call Center
- 座席

机房管理
- 规范
- 流程

模拟题:
- 校园网的组建
- IP冲突解决方案
```

**Senior Network Engineer 高级网络工程师**

**Route**

OSPF,RIP

NAT

**Firewall/Switch**

VLAN

链路负载均衡

一台 Linux 服务器上4个网卡，3个网口分别链接了四条线路，剩下的一个链接局域网。

怎样配置这个服务器

怎样使内网用户访问外放能够负载均衡3条线路

怎样指派某一个IP段或者某一组IP访问互联网通过指定的线路

防火墙/路由器怎样实现高可用

交换机的高可用

# 4. 运维工程师面试题

## Junior

情景模拟题

**Linux 基础**

```
Directory & File
- cp,mv,rm,cat
- vim, emacs

Monitor
- top, ps, vmstat, free

Permission
- User and Group Adminstrator
- Owner and Access permissions

Partition & File System
- EXT3,EXT4, XFS,ZFS,JFS,Btrfs
- label,format,repair, fdisk
- mount, umount

Network
- interface, ip address, netmask, gateway
- ifconfig, ip, route, netstat, ping, nslookup,
dig, tcpdump
- iptables, tc

Service
- inetd, xinetd
- init level rc.local, rc.d, rcX.d

Application
- Apache, PHP, MySQL, Resin, Tomcat, Jboss
```

```
- Samba, Vsftpd, proftpd, pureftpd
- openvpn, openssl, openssh

Shell
- awk, sed, grep, find,
- Bash,Tcsh,Ksh,Zsh
- Lists of Commands
  * ;, &, &&, ||
- Pipelines
  * | , |&
- Redirections Standard Input/Output and Standard
Error
  * >, >>, <, <<, 2>&1
- Special Parameters  * @ # ? - $ ! 0 -

Python or Perl
- argv,argc,
- threading, fork
- analyze access.log
```

## 服务器与硬件设备

raid原理

```
                                                              -
简述 RAID 0, 1, 5, 6, 10, 50?
                                                              -
RAID 0, 1, 5, 6 允许损害几块硬盘?
                                                              -
RAID 10, 50 允许损害几块硬盘?
                                                              -
什么Hotspare盘?
                                                              -
怎么更换损坏的硬盘?
```

重点是 RAID10允许允许损坏几块硬盘?

## 操作系统

### Linux 基本知识

- Linux 在什么情况下需要重新启动，什么情况不需要重新启动?
- 修改IP地址需要重启服务器吗?
- 修改DNS地址需要重启服务器吗?
- 安装了新软件或者更改了配置文件需要重启服务器吗?
- 怎样不重新启动系统，并且使配置文件生效?
- HUP 信号的作用是什么?

## 服务器

### FTP Server 基本知识

- 当使用系统帐号作为登录用户时，怎样将用户限制在指定的文件夹中，作为根目录?
- 虚拟用户的原理?
- 什么是chroot?

### WEB Server 基本知识

- 如何查看占用80端口的进程 lsof -i:80
- prefork 与 worker 原理和区别是什么?

```
                                                                -
虚拟主机原理?
                                                                -
HTTP状态码 2xx, 3xx, 4xx, 5xx ,404, 301
```

高级运维工程师

**Linux 优化**

Linux 默认同时能打开多少个文件?

Linux 默认同时能开启多少个TCP链接?

配置超过10000个链接数的服务器，Linux怎样优化?

**RAID 磁盘阵列**

简述 RAID?

RAID 0 5 6 10 50 都适用于那些场景?

数据库适用那种 RAID?

RAID 10 磁盘结构是怎样的，RAID 10 可以允许损坏那几块硬盘，请指出那就几块可以损坏，那几块不能损坏?

什么是逻辑卷，适合那些场景?

磁盘阵列

```

+------+------+------+
| HDD1 | HDD3 | HDD5 |
```

```
+------+------+------+
| HDD2 | HDD4 | HDD6 |
+------+------+------+
```

存储

谈谈 iSCSI 与 SAN

两个电脑挂载同一个 iSCSI 设备，是否可行?

**磁盘 IO**

服务器IO瓶颈都在那些地方?

**Fibre Channel vs FCoE**

请比较 Fibre 与 FCoE 有缺点，以及适合场景

网络

由于网卡损坏，或者水晶头接触不良导致断线，有没有解决方案?

当网卡1G已经不能满足通信要求，你怎么样应对，你的解决方案是什么?

**DNS**

简述如何将DNS服务从万网迁移至DnsPod，需要注意哪些事项?

什么是A记录，CNAME记录，TXT记录，MX记录，NS记录?

怎样查看域名的过期时间?

怎样配置SPF，DKIM?

**Linux 操作系统**

一个全新的 Linux 服务器，你会做哪些初始化操作?

RPM安装 与 编译安装有什么区别，是否编译安装性能会更好?

文件系统

简述 fdisk 与 gpt

文件系统怎样做快照，怎样快速恢复快照

文件系统损坏怎么修复

怎样查看磁盘的UUID

进程管理

怎么样查看某一个文件正在被那个程序访问?

**Web 服务器**

Apache/Nginx 默认链接数是多少?

怎样修改默认链接数?

怎样实现防盗链?

如果你的网站上面的数据内容，被别人抓取。怎样屏蔽爬虫？怎样快速找到抓取的IP地址？ 还应该做那些处理能避免再次发生?

怎样防止注入攻击?

注入都有哪些手段?

WEB 服务器安全配置都有哪些?

怎样保证 WEB 服务目录/文件的安全?

怎样防止文件被修改?

怎样第一时间发现文件被撰改？并提前拦截?

**Rewrite**

源地址： http://www.netkiller.cn/index.html?id=100

Rewrite后： http://www.netkiller.cn/article/100.html

请问如何实现

**Nginx location 基础知识?**

当前 document root 是 /www/example.com/www.example.com URL 为 http://www.example.com

现在需要实现 http://www.example.com/inc

将 inc 定为到 /www/example.com/inc.example.com

请问如何实现

应用服务器

谈谈Tomcat优化

Nginx 通过代理服务器访问 Tomcat， Java应用输出页面含有SSI标签，Nginx 怎样处理 java 输出的SSI标签

**Mail 服务器**

什么事别名

什么是虚拟域

怎样防垃圾邮件

怎么实现 SMTP 认证，SMTP加密算法有那几种

怎样配置 SSL SMTP/POP/IMAP

自建EDM（电子邮件推广）服务器需要哪些条件？注意事项？怎样避免被封锁。

邮件怎样转寄

怎样配置邮件列表服务器

怎样配置SPF，DKIM?

**攻击防守**

如果被挂马怎么除了?

被植入代码有哪些特点，怎样快速找到被植入的木马。

怎样监控恶意代码入住或修改

什么是UDP流量攻击，怎样防止UDP流量攻击?

**服务器监控**

服务器监控都有哪些手段?

网络设备都有哪些监控手段?

监控除了SNMP还有那些协议?

怎样监控硬件，例如硬盘损坏？服务器风扇停转?

**数据库**

出现锁表情况怎么处理?

## 5. 软件工程师面试题

### Junior Software Engineer

**HTML 与 Javascript**

怎样实现表单数据保存?

例如：开发一个用户注册程序，当用户进入你的网站注册用户，填写了一部分用户信息，之后因各种原因用户没有进行下去，或离开注册页面，或关闭浏览器。甚至关闭电脑 。

需求：无论何时用户再次打开你的网站，再次进入注册页面后，应该将之前用户填写的资料显示在表单内，用户只需要做没有完成的那部分工作即可。

编程语言

Ajax 如何跨域请求?

什么是SSO? Cookie 如何跨域请求?

HTTP协议面试题,请介绍下面HTTP头作用.

```
                                                        -
location
                                                        -
smaxage / max-age
                                                        -
ETAG
```

**SSI 服务器端包含应用**

## Senior Software Engineer

**设计一个分类功能?**

该功用于行政区域划分，商品分类，等等 例如中国->广东-深圳 .....

要求:

```
无限极分类，层次深度不限
快速检索，不能使用递归
只能使用一个数据库表实现
可以生成树形目录
```

这是 一个简单的 OOP 面试题，在做多年的面试经验中，发现很多人不知道怎样实现上面的问题

**开发框架**

```
简述 MVC 原理以及实现
怎样实现 URL 路由
怎样实现类，方法访问权限控制
请问下面代码怎么实现?

用你最熟悉的语言实现。
```

## 进程与多线程

什么是阻塞，什么是非阻塞?

什么是同步，什么是异步?

什么情况下使用线程锁?

进程与线程的区别?

进程间通信有那几种，线程通信有那几种，以及各自的优势?

## 消息队列

消息有哪些瓶颈?

## 序列化

什么是序列化?

常用序列化方式都有哪些?

## 用户注册的功能需求

场景模拟： 用户开户注册时常常填写了一部分资料，就离开了，有几种情况

用户放弃注册
网络连接失败
提交出错
其他链接吸引了用户点击

等等原因.....

需求:

要求记录用户填写资料，再次回来（数日/数月后）点击注册的时候，用户不用重新填写所有资料，只需完成未完成的部分即可。
重要资料例如手机，电邮，即时通讯号码等等需要记录到数据库，已被公司回访客户。
在用户成功注册后应该删除之前保留在数据库中的手机，电邮，即时通讯等等。
请问如何实现？资料怎样保存?

## 事务处理相关

简述什么是事务处理?

在不能使用数据库的事务处理以及锁（表锁/行级锁）时，怎么保持数据一致性？怎么解决数据库并发操作?

怎样解决避免多个用户读读取同一条数据记录?
怎样避免多个用户更新同一条数据

## 面向对象试题?

编写一个求和程序

```
s = new Sum();
s.add(10).add(5).add(6)
```

```

```

s.add(10).add(5).add(6).....add(3) 可以无限的写下去

```
obj = new Object()
obj.a()
obj.b()
obj.c()
...
...
obj.z()

obj.a().b().c() ... z()
```

与上面类似

这是 一个简单的 OOP 面试题，在做多年的面试经验中，发现很多人不知道怎样实现上面的问题

**编写一个文件copy的程序?**

要求复制一个文件，或者一个目录，目录下面可能包含文件和目录，目录深度未知。你所写的程序要考虑程序的，通用型，健壮性，稳定性，性能等等

这里没有准确的答案，这个问题主要考虑应聘者，对于这样一个简单的程序，他能考虑的深度与广度.

1. 复制文件或目录是否保留原有的权限与日期等信息
2. 目的目录或文件如果存在怎么处理，是覆盖还是增量复制，分别怎么实现

3. 怎样保证复制后，两边100%正确，没有丢失文件或者文件不一致

4. 复制过程中如果原文件被改动怎么办

5. 如果权限不足怎么处理

6. 对于大文件怎么处理，对于GB/TB/PB级别怎么处理

**写一个读TXT文件显示其内容程序你会考虑哪些细节?**

这个问题与上一个问题类似,如果应聘者立即给出这样的答案，他根本不合格。

```
                                                    f =
open(path/filename)

while s = f.read() {

print s
                                                    }

f.close()
```

如果他考虑问题的能涉及下面列出的几个选项，他写出的程序你绝对可以放心。

```
                                                    考虑
过目录不存怎么处理吗?
                                                    考虑
过目录权限不够吗?
                                                    考虑
过文件不存怎么处理吗?
                                                    考虑
过文件权限不够吗?
```

```
                                                                    考虑
过目录深度吗?
                                                                    考虑
过目录,文件中存在特殊字符吗?
                                                                    考虑
过文档大小1M, 100M , 500M , 1G , 10G 远远超过你的内存空间,怎么处理吗?
                                                                    考虑
TXT文件换行符(LF,CR,CRLF)吗?
                                                                    考虑
编码问题吗?
```

**SNS 社交网络，怎样解决朋友关系?**

例如：你有一个朋友，他有他的朋友，你可能认识他的朋友，他朋友的朋友可能认识你。

1. 怎样查询出你朋友的朋友
2. 怎样显示你是怎么认识，你朋友的朋友,例如：我 - 小王 - 小李 - 小张
3. 显示朋友关系图

**设计一个电商的商品数据库?**

一个商品有很多属性，例如尺寸，颜色这些属性有固定的值，而另一些属性如重量，体积是需要填写具体数值的，并且还有对应的单位。

1. 商品分类，可以无限层次，可能瞬间查出某一个品类下的所有商品
2. 每个品类的商品都有不同的产品属性，且很多属性可能公用，例如：颜色，重量

3. 要求可以检索商品，可以通过属性，分类，价格等等搜索

要求：

商品分类（上一个问题中已经实现）
商品属性，有多个属性，且数目不确定，所以需要设计成可以无限添加
商品属性即可下拉选择，也可以填写具体数值
问题：

商品搜索怎么解决，包含商品名称，属性，属性值，描述的搜索
商品的库存怎么设计
分类搜索，怎样列出所有子分类以及子分类下的所有分类（无限深度）

## 谈谈对缓存的认识?

从用户打开浏览器到返回数据都会经过那些缓存，怎么控制这些缓存

CDN 缓存的原理？CDN 都可能缓存那些内容?

网站首页90%的内容是静态的，但是用户登录状态，消息状态是动态的怎么解决?

JSON 可能缓存吗?

浏览器缓存与CDN缓存的关系，怎样实现用户浏览器与CDN同时缓存?

这个问题主要是网站性能优化方面所用到的技术

## 6. 架构师面试题

软件架构

插件的实现原理?

插件有几部分组成
如何实现插件安装，卸载，启用，禁用?
安装，卸载，启用，禁用怎样实现不停机，不关闭服务的情况进行?

如何设计一个MVC框架

怎样实现 URL 路由
怎样实现控制器
怎样实现视图
怎样实现模型

如何设计一个SOA框架?

框架分为几个部分?
采用什么协议与框架通信?
如果考虑到性能使用二进制协议你怎样实现?
如何解决并发冲突?
如何支持事务?
你怎样与消息队列集成或者通信?

设计一个分布式计划任务系统?

背景：计划任务即周期或定时运行的程序，我们要解决单点故障问题与负载均衡的问题，在一个分布式系统中单节点是不允许的。

设计要求：能够实现高可用，负载均衡，横向扩展

怎样处理同时运行产生的冲突问题?
怎样排队运行?
任务如何持久化?
一个节点宕机，另一个节点怎么接管没有完成的任务?
如何横向扩展?
扩展，收缩，维护如果能做到不停机，不影响业务?

## 瓶颈分析

请分析一下，下面图中可能出现的瓶颈并提出优化方案

User --> Wan -> Firewall -> Switch -> SLB -> Web Server -> Application Server --> Cache -> Database

## 设计一个**10G**交换的网络

## 7. 数据库管理员

**数据库安全性**

通过WEB入住或者上传脚本使用WEB服务器上的数据库账号进行数据撰改

怎样将损失降到最低?
怎样记录修改痕迹?
你怎么样防止合法用户撰改表数据?
你怎么样防止合法用户撰改数据表中的一行数据?
你怎么样防止合法用户撰改数据表中的一列数据?

**手机码号，电子邮箱，银行资料等等安全问题**

你怎样改密这些数据?
加密后怎么样解决查询与性能问题。?
如何解决索引问题?

**数据库设计**

设计商品分类表

分类层次与深度做多无限极分类
怎样做索引
设计在线用户表

**索引**

临时表需要建索引吗?
在什么情况下将使用索引
在什么情况下将使用不索引
在什么情况查询结果能被缓存
在什么情况查询结果不能被缓存

## 8. 测试工程师

高级测试工程师

**压力测试**

压力测试注意事项有哪些?

**自动化测试**

怎样解决图片验证吗输入的问题

怎会解决手机验证码输入问题

**注入测试**

POST 注入测试

GET 注入测试

SQL 注入测试

TCP 注入测试

**数据库测试**

数据库结构测试

存储过程测试

函数测试

防撰改测试

**Web 服务器压力测试**

需求：开发人员已经完成开发工作，现在需要对他的代码做一次压力测试 问题：你将怎样做这次压力测试？请描述出工作中的重点。 你现在有一个空的测试服务器，没有安装操作系统，这是起点。

**Restfull 测试**

GET 请求测试

POST 提交测试

# 第 10 章 采购管理

为了从项目实施组织之外获得所需资源或服务所采取的一系列管理措施。它包括采购计划，采购与征购，资源的选择以及合同的管理等项目工作。

采购管理这块运维涉及比较多，机房选型，机柜，路由器，交换机，防火墙，负载均衡，服务器，存储.....

# 部分 II. 项目管理工具

## project management tool

# 第 11 章 Gitlab 项目管理

实施DEVOPS首先我们要有一个项目管理工具。

我建议使用 Gitlab，早年我倾向使用Trac，但Trac项目一直处于半死不活状态，目前来看Trac 对于 Ticket管理强于Gitlab，但Gitlab发展的很快，我们可以看到最近的一次升级中Issue 加入了 Due date 选项。Gitlab已经有风投介入，企业化运作，良性发展，未来会超越Redmine等项目管理软件，成为主流。所以我在工具篇采用Gitlab，BTW 我没有使用 Redmine，我认为 Redmine 的发展方向更接近传统项目管理思维。

软件项目管管理，我需要那些功能，Ticket/Issue管理、里程碑管理、内容管理Wiki、版本管理、合并分支、代码审查等等

关于Gitlib的安装配置请参考
http://www.netkiller.cn/project/project/gitlab/index.html

# 1. GitLab 安装与配置

https://github.com/gitlabhq

GitLab是一个利用 Ruby on Rails 开发的开源应用程序，实现一个自托管的Git项目仓库，可通过Web界面进行访问公开的或者私人项目。

它拥有与Github类似的功能，能够浏览源代码，管理缺陷和注释。可以管理团队对仓库的访问，它非常易于浏览提交过的版本并提供一个文件历史库。团队成员可以利用内置的简单聊天程序(Wall)进行交流。它还提供一个代码片段收集功能可以轻松实现代码复用，便于日后有需要的时候进行查找。

GitLab 5.0以前版本要求服务器端采用 Gitolite 搭建，5.0版本以后不再使用 Gitolite ，采用自己开发的 gitlab-shell 来实现。如果你觉得安装麻烦可以使用 GitLab Installers 一键安装程序。

## 1.1. Almalinux 9.0

目前 gitlab 官方包还不支持，需要手工安装

### Gitlab Runner

```
[root@netkiller gitlab]# curl -L --output
/usr/local/bin/gitlab-runner "https://gitlab-runner-
downloads.s3.amazonaws.com/latest/binaries/gitlab-runner-linux-
amd64"
[root@netkiller gitlab]# chmod +x /usr/local/bin/gitlab-runner
[root@netkiller gitlab]# useradd --comment 'GitLab Runner' --
create-home gitlab-runner --shell /bin/bash
[root@netkiller gitlab]# sudo chmod 666 /var/run/docker.sock
[root@netkiller gitlab]# usermod -aG docker gitlab-runner
[root@cloud gitlab]# gitlab-runner install --user=gitlab-runner
--working-directory=/home/gitlab-runner
Runtime platform                                    arch=amd64
os=linux pid=66582 revision=32fc1585 version=15.2.1

[root@cloud gitlab]# systemctl enable gitlab-runner
[root@cloud gitlab]# systemctl start gitlab-runner
[root@cloud gitlab]# systemctl status gitlab-runner
● gitlab-runner.service - GitLab Runner
     Loaded: loaded (/etc/systemd/system/gitlab-runner.service;
enabled; vendor preset: disabled)
     Active: active (running) since Wed 2022-08-10 19:42:39
CST; 6s ago
   Main PID: 66679 (gitlab-runner)
      Tasks: 12 (limit: 203532)
     Memory: 11.3M
        CPU: 49ms
     CGroup: /system.slice/gitlab-runner.service
             └─66679 /usr/local/bin/gitlab-runner run --
working-directory /home/gitlab-runner --config /etc/gitlab-
runner/config.toml --service gitlab-runner --user gitlab-runner
```

## 1.2. CentOS 8 Stream 安装 Gitlab

```
dnf install langpacks-en glibc-all-langpacks -y
localectl set-locale LANG=en_US.UTF-8

sudo systemctl status firewalld
sudo firewall-cmd --permanent --add-service=http
sudo firewall-cmd --permanent --add-service=https
sudo systemctl reload firewalld

sudo dnf install postfix
sudo systemctl enable postfix
sudo systemctl start postfix


curl -s
https://packages.gitlab.com/install/repositories/gitlab/gitlab-
ce/script.rpm.sh | bash

EXTERNAL_URL="http://gitlab.example.com"

export LC_ALL=en_US.UTF-8
export LANG=en_US.UTF-8
export LC_CTYPE=UTF-8

dnf install -y gitlab-ce

cp /etc/gitlab/gitlab.rb{,.original}

gitlab-ctl reconfigure
```

查看 root 密码

```
[root@localhost ~]# cat  /etc/gitlab/initial_root_password
# WARNING: This value is valid only in the following conditions
#          1. If provided manually (either via
`GITLAB_ROOT_PASSWORD` environment variable or via
`gitlab_rails['initial_root_password']` setting in `gitlab.rb`,
it was provided before database was seeded for the first time
(usually, the first reconfigure run).
```

```
#          2. Password hasn't been changed manually, either via
UI or via command line.
#
#          If the password shown here doesn't work, you must
reset the admin password following
https://docs.gitlab.com/ee/security/reset_user_password.html#re
set-your-root-password.

Password: dpzQFzltaq0BhAwDnugMf6MCFbvItXDvC+RcuN9R+wg=

# NOTE: This file will be automatically deleted in the first
reconfigure run after 24 hours.
```

GitLab Runner

```
curl -sL
"https://packages.gitlab.com/install/repositories/runner/gitlab
-runner/script.rpm.sh" | sudo bash
dnf install gitlab-runner
```

配置文件 /etc/gitlab-runner/config.toml

```
[root@localhost ~]# systemctl restart gitlab-runner
```

## 1.3. Docker 方式安装 Gitlab

Docker 安装有个小缺点，不能使用 22 端口，因为 22 端口已经被宿主主机占用。

**Docker 运行**

```
export GITLAB_HOME=/srv/gitlab
```

```
sudo docker run --detach \
  --hostname gitlab.example.com \
  --publish 443:443 --publish 80:80 --publish 22:22 \
  --name gitlab \
  --restart always \
  --volume $GITLAB_HOME/config:/etc/gitlab \
  --volume $GITLAB_HOME/logs:/var/log/gitlab \
  --volume $GITLAB_HOME/data:/var/opt/gitlab \
  gitlab/gitlab-ce:latest
```

配置对外url，域名或者ip，公网能访问即可

```
vim /mnt/gitlab/etc/gitlab.rb
添加一下配置:
external_url    'http://127.0.0.1'  (你的域名或者ip地址)
```

配置邮箱

```
vim /mnt/gitlab/etc/gitlab.rb
gitlab_rails['smtp_enable'] = true
gitlab_rails['smtp_address'] = "smtp.qq.com"
gitlab_rails['smtp_port'] = 465
gitlab_rails['smtp_user_name'] = "13721218@qq.com"    (替换成自己
的QQ邮箱)
gitlab_rails['smtp_password'] = "xxxxx"
gitlab_rails['smtp_domain'] = "smtp.qq.com"
gitlab_rails['smtp_authentication'] = "login"
gitlab_rails['smtp_enable_starttls_auto'] = true
```

```
gitlab_rails['smtp_tls'] = true
gitlab_rails['gitlab_email_from'] = '13721218@qq.com'  (替换成自己的QQ邮箱，且与smtp_user_name一致)
```

重新启动gitlab

```
docker restart gitlab-ce
sudo docker logs -f gitlab
```

修改 /etc/gitlab/gitlab.rb 配置文件

```
docker exec -it gitlab editor /etc/gitlab/gitlab.rb
gitlab  |   docker restart gitlab
```

**Docker compose 安装 Gitlab**

宿主主机开放 80/443 端口

```
systemctl status firewalld
firewall-cmd --permanent --add-service=http
firewall-cmd --permanent --add-service=https
systemctl reload firewalld
```

创建工作目录

```
[root@localhost ~]# mkdir -p /opt/gitlab/{config,data,logs}
```

```
[root@localhost ~]# cd /opt/gitlab/
```

安装 gitlab

```
[root@localhost gitlab]# vim docker-compose.yaml
version: '3.9'
services:
    gitlab:
      image: 'gitlab/gitlab-ce:latest'
      container_name: "gitlab"
      restart: unless-stopped
      privileged: true
      hostname: 'gitlab.example.com'
      environment:
        TZ: 'Asia/Shanghai'
        GITLAB_OMNIBUS_CONFIG: |
          external_url 'https://gitlab.example.com'
          gitlab_rails['time_zone'] = 'Asia/Shanghai'
          gitlab_rails['smtp_enable'] = true
          gitlab_rails['smtp_address'] = "smtp.netkiller.cn"
          gitlab_rails['smtp_port'] = 465
          gitlab_rails['smtp_user_name'] =
"netkiller@netkiller.cn"
          gitlab_rails['smtp_password'] = "******"
          gitlab_rails['smtp_domain'] = "netkiller.cn"
          gitlab_rails['smtp_authentication'] = "login"
          gitlab_rails['smtp_enable_starttls_auto'] = true
          gitlab_rails['smtp_tls'] = true
          gitlab_rails['gitlab_email_from'] =
'netkiller@netkiller.cn'
          gitlab_rails['gitlab_shell_ssh_port'] = 22
      ports:
        - '80:80'
        - '443:443'
        - '23:22'
      volumes:
        - /opt/gitlab/config:/etc/gitlab
        - /opt/gitlab/logs:/var/log/gitlab
        - /opt/gitlab/data:/var/opt/gitlab
```

```
```

启动

```
[root@gitlab gitlab]# docker compose up
```

**例 11.1. Docker 部署 GitLab 查看登陆密码**

```
Neo-iMac:docker neo$ docker ps
CONTAINER ID   IMAGE                          COMMAND
CREATED          STATUS                       PORTS
NAMES
a762c0c8c950   gitlab/gitlab-ce:latest    "/assets/wrapper"
14 minutes ago   Up 14 minutes (healthy)   0.0.0.0:80->80/tcp,
22/tcp, 0.0.0.0:443->443/tcp   gitlab
433a04f60108   sonarqube:community        "/opt/sonarqube/bin/…"
10 days ago      Up 15 minutes              0.0.0.0:9000-
>9000/tcp                                sonarqube
ea753b0905f7   postgres:latest            "docker-entrypoint.s…"
10 days ago      Up 15 minutes              5432/tcp
postgresql

Neo-iMac:docker neo$ docker exec -it gitlab cat
/etc/gitlab/initial_root_password
# WARNING: This value is valid only in the following conditions
#          1. If provided manually (either via
`GITLAB_ROOT_PASSWORD` environment variable or via
`gitlab_rails['initial_root_password']` setting in `gitlab.rb`,
it was provided before database was seeded for the first time
(usually, the first reconfigure run).
#          2. Password hasn't been changed manually, either via
UI or via command line.
#
#          If the password shown here doesn't work, you must
reset the admin password following
https://docs.gitlab.com/ee/security/reset_user_password.html#re
set-your-root-password.
```

```
Password: LnfGjN5ySHSyTev8VSqCNFna0m43i3oF6FTU8QThoSQ=

# NOTE: This file will be automatically deleted in the first
reconfigure run after 24 hours.
```

## Docker Compose 安装 gitlab-runner

恢复配置文件，首次配置请跳过这步

```
cat > /srv/gitlab-runner/config.toml << EOF
concurrent = 1
check_interval = 0

[session_server]
  session_timeout = 1800

[[runners]]
  name = "microservice"
  url = "https://gitlab.netkiller.cn/"
  token = "cSB87csLVQnP-JfiU3rX"
  executor = "shell"
  [runners.custom_build_dir]
  [runners.cache]
    [runners.cache.s3]
    [runners.cache.gcs]
    [runners.cache.azure]
EOF
```

安装 gitlab-runner

```
version: '3.9'
services:
  gitlab-runner:
```

```
    container_name: "gitlab-runner"
    image: gitlab/gitlab-runner:alpine
    restart: unless-stopped
    depends_on:
      - gitlab
    privileged: true
    volumes:
      - /srv/gitlab-runner:/etc/gitlab-runner
      - /var/run/docker.sock:/var/run/docker.sock
      - /bin/docker:/bin/docker
```

启动 Gitlab runner

```
[root@netkiller gitlab]# docker compose up -d
```

注册 gitlab-runner 到 Gitlab

```
docker exec -it gitlab-runner gitlab-runner register
docker exec -it gitlab-runner gitlab-runner registe--url
https://gitlab.netkiller.cn/ --registration-token
GR13489417SgFtcXMtuyPSqNn9TCoCo
```

**例 11.2. Docker 部署 gitlab-runner 注册演示**

```
[root@localhost gitlab]# docker-compose exec gitlab-runner
gitlab-runner register
Runtime platform                                    arch=amd64
os=linux pid=77 revision=8b63c432 version=14.3.1
Running in system-mode.

Enter the GitLab instance URL (for example,
```

```
https://gitlab.com/):
http://192.168.30.12/
Enter the registration token:
suDmuiYsdYoEvhX1ppBy
Enter a description for the runner:
[1d9ca588f551]: development
Enter tags for the runner (comma-separated):
shell
Registering runner... succeeded
runner=suDmuiYs
Enter an executor: shell, ssh, docker+machine, docker-
ssh+machine, custom, parallels, virtualbox, kubernetes, docker,
docker-ssh:
shell
Runner registered successfully. Feel free to start it, but if
it's running already the config should be automatically
reloaded!
[root@localhost gitlab]#
```

## 1.4. Yum 安装 GitLab

```
yum localinstall -y https://downloads-
packages.s3.amazonaws.com/centos-6.6/gitlab-ce-
7.10.0~omnibus.2-1.x86_64.rpm

gitlab-ctl reconfigure

cp /etc/gitlab/gitlab.rb{,.original}
```

停止 GitLab 服务

```
# gitlab-ctl stop
ok: down: logrotate: 1s, normally up
ok: down: nginx: 0s, normally up
ok: down: postgresql: 0s, normally up
ok: down: redis: 0s, normally up
ok: down: sidekiq: 1s, normally up
```

```
ok: down: unicorn: 0s, normally up
```

启动 GitLab 服务

```
# gitlab-ctl start
ok: run: logrotate: (pid 3908) 0s
ok: run: nginx: (pid 3911) 1s
ok: run: postgresql: (pid 3921) 0s
ok: run: redis: (pid 3929) 1s
ok: run: sidekiq: (pid 3933) 0s
ok: run: unicorn: (pid 3936) 1s
```

配置gitlab

```
# vim /etc/gitlab/gitlab.rb
external_url 'http://gitlab.example.com'
```

SMTP配置

```
gitlab_rails['gitlab_email_enabled'] = true
gitlab_rails['gitlab_email_from'] = 'openunix@163.com'
gitlab_rails['gitlab_email_display_name'] = 'Neo'
gitlab_rails['gitlab_email_reply_to'] = 'noreply@example.com'

gitlab_rails['smtp_enable'] = true
gitlab_rails['smtp_address'] = "smtp.163.com"
gitlab_rails['smtp_user_name'] = "openunix@163.com"
gitlab_rails['smtp_password'] = "password"
gitlab_rails['smtp_domain'] = "163.com"
gitlab_rails['smtp_authentication'] = "login"
```

任何配置文件变化都需要运行 # gitlab-ctl reconfigure

WEB管理员

```
# Username: root
# Password: 5iveL!fe
```

**GitLab Runner**

```
curl -L
https://packages.gitlab.com/install/repositories/runner/gitlab-
ci-multi-runner/script.rpm.sh | sudo bash
sudo yum install gitlab-ci-multi-runner
```

进入 CI 配置页面
http://git.netkiller.cn/netkiller.cn/www.netkiller.cn/settings/ci_cd

Specific Runners 你将看到 CI 的URL和他的Token

Specify the following URL during the Runner setup:
http://git.netkiller.cn/ci

Use the following registration token during setup:
wRoz1Y_6CXpNh2JbxN_s

现在回到 GitLab Runner

```
# gitlab-ci-multi-runner register
Running in system-mode.

Please enter the gitlab-ci coordinator URL (e.g.
https://gitlab.com/):
http://git.netkiller.cn/ci
Please enter the gitlab-ci token for this runner:
wRoz1Y_6CXpNh2JbxN_s
Please enter the gitlab-ci description for this runner:
[iZ62yln3rjjZ]: gitlab-ci-1
Please enter the gitlab-ci tags for this runner (comma
separated):
test
```

```
Whether to run untagged builds [true/false]:
[false]:
Registering runner... succeeded
runner=wRoz1Y_6
Please enter the executor: docker, docker-ssh, shell, ssh,
virtualbox, docker+machine, docker-ssh+machine, kubernetes,
parallels:
shell
Runner registered successfully. Feel free to start it, but if
it's running already the config should be automatically
reloaded!
```

回到 Gitlab 页你将看到 Pending 状态变成 Running 状态

升级 GitLab Runner

```
yum install gitlab-ci-multi-runner
```

## 1.5. 绑定SSL证书

编辑 /etc/gitlab/gitlab.rb 文件

```


external_url 'https://git.netkiller.cn'

nginx['enable'] = true
nginx['redirect_http_to_https'] = true
nginx['ssl_certificate'] =
"/etc/gitlab/ssl/git.netkiller.cn.crt"
nginx['ssl_certificate_key'] =
"/etc/gitlab/ssl/git.netkiller.cn.key"
nginx['listen_https'] = true
nginx['http2_enabled'] = true


```

## 1.6. Gitlab 管理

## gitlab-rake 命令

进入容器

```
[root@localhost ~]# docker exec -it gitlab bash
```

重置密码

```
root@gitlab:~# gitlab-rake "gitlab:password:reset"
Enter username: neo
Enter password:
Confirm password:
Password successfully updated for user with username neo.
```

## gitlab-runner 命令

```
gitlab-runner register  #注册，非交互模式添加 --non-interactive
gitlab-runner list      #列出配置文件中已注册的配置项
gitlab-runner verify    #检查注册

#注销所有配置项
gitlab-runner unregister --all-runners

#使用令牌注销
gitlab-runner unregister --url http://gitlab.example.com/ --
token XXXXXXXX

#使用名称注销
gitlab-runner unregister --name test-runner
```

## Gitlab 迁移，备份和恢复

备份

```
[root@gitlab ~]# gitlab-rake gitlab:backup:create
```

备份数据保存在 /var/opt/gitlab/backups 目录中

```
[root@gitlab ~]# ls -l /var/opt/gitlab/backups | grep 'Feb 17'
-rw-------  1 git git 18946846720 Feb 17 14:14
1645078053_2022_02_17_14.7.2_gitlab_backup.tar
-rw-------  1 git git   878822637 Feb 17 13:58 artifacts.tar.gz
-rw-r--r--  1 git git         190 Feb 17 14:07
backup_information.yml
-rw-------  1 git git        1500 Feb 17 13:57 builds.tar.gz
drwxr-xr-x  2 git git          29 Feb 17 13:57 db
-rw-------  1 git git      129802 Feb 17 13:58 lfs.tar.gz
-rw-------  1 git git  2404907359 Feb 17 14:07 packages.tar.gz
-rw-------  1 git git         156 Feb 17 13:58 pages.tar.gz
-rw-------  1 git git 16017027489 Feb 17 14:06 registry.tar.gz
drwx------  3 git git          21 Feb 17 13:57 repositories
-rw-------  1 git git         147 Feb 17 13:58
terraform_state.tar.gz
-rw-------  1 git git    11061155 Feb 17 13:57 uploads.tar.gz
```

如需修改备份文件目录，可以通过修改 /etc/gitlab/gitlab.rb 配置文件来修改默认备份目录

```
gitlab_rails['backup_path'] = "/var/opt/gitlab/backups"
```

修改完成之后使用 gitlab-ctl reconfigure 命令重载配置文件即可

设置备份过期时间，单位是秒

```
[root@gitlab ~]# vim /etc/gitlab/gitlab.rb
gitlab_rails['backup_keep_time'] = 604800
```

创建定时任务，自动备份 gitlab

```
[root@gitlab ~]# crontab -e
0 2 * * * /opt/gitlab/bin/gitlab-rake gitlab:backup:create
```

恢复

将备份文件复制到 /var/opt/gitlab/backups/ 目录中，例如 1645078053_2022_02_17_14.7.2_gitlab_backup.tar，然后执行恢复命令

```
[root@gitlab ~]# gitlab-rake gitlab:backup:restore
BACKUP=1645078053_2022_02_17_14.7.2
```

注意去掉 _gitlab_backup.tar 后缀，只取 1645078053_2022_02_17_14.7.2

# 2. 初始化 Gitlab

Gitlab 安装完成之后，我们需要对它做一个初始化操作。

## 2.1. 操作系统初始化

CentOS 8 / Rockey 8.5 初始化脚本

```
dnf -y upgrade
dnf -y install epel-release

dnf install -y bzip2 tree psmisc \
telnet wget rsync vim-enhanced \
net-tools bind-utils

timedatectl set-timezone Asia/Shanghai
dnf install -y langpacks-en glibc-langpack-en
localectl set-locale LANG=en_US.UTF-8

cat >> /etc/environment <<EOF
LC_ALL=en_US.UTF-8
LANG=en_US.UTF-8
LC_CTYPE=UTF-8
EOF

cat >> /etc/profile.d/history.sh <<EOF
# Administrator specific aliases and functions for system security
export HISTSIZE=10000
export HISTFILESIZE=10000
export HISTTIMEFORMAT="%Y-%m-%d %H:%M:%S "
export TIME_STYLE=long-iso
EOF
source /etc/profile.d/history.sh

cp /etc/selinux/config{,.original}
sed -i "s/SELINUX=enforcing/SELINUX=disabled/" /etc/selinux/config
setenforce Permissive

cat >> /etc/sysctl.conf <<EOF

# Netkiller
net.ipv4.ip_local_port_range = 1025 65500
net.ipv4.tcp_tw_reuse = 1
```

```
net.ipv4.tcp_keepalive_time = 1800
net.core.netdev_max_backlog=3000
net.ipv4.tcp_max_syn_backlog = 1024
net.ipv4.tcp_max_tw_buckets = 4096
net.core.somaxconn = 1024

# TCP BBR
net.core.default_qdisc=fq
net.ipv4.tcp_congestion_control=bbr
EOF

sysctl -p

cat > /etc/security/limits.d/20-nofile.conf <<EOF

* soft nofile 65535
* hard nofile 65535

EOF

groupadd -g 80 www
adduser -o --uid 80 --gid 80 -G wheel -c "Web Application" www
```

### gitlab-runner

```
curl -L
"https://packages.gitlab.com/install/repositories/runner/gitlab-
runner/script.rpm.sh" | sudo bash
dnf install -y gitlab-runner
cp /etc/gitlab-runner/config.toml{,.original}
systemctl enable gitlab-runner
```

### Docker

```
dnf config-manager --add-
repo=https://download.docker.com/linux/centos/docker-ce.repo
dnf install -y docker-ce
```

```
systemctl enable docker
systemctl start docker

GID=$(egrep -o 'docker:x:([0-9]+)' /etc/group | egrep -o '([0-9]+)')
adduser -u ${GID} -g ${GID} -G wheel -c "Container Administrator"
docker

usermod -aG docker www
usermod -aG docker gitlab-runner
usermod -aG www gitlab-runner

dnf remove -y python36
dnf install -y python39
pip3 install docker-compose netkiller-devops
```

Mirror

```
cat << EOF > /etc/docker/daemon.json

{
        "registry-mirrors": [
        "https://docker.mirrors.ustc.edu.cn",
                "https://registry.docker-cn.com",
                "https://registry.cn-hangzhou.aliyuncs.com",
        "http://hub-mirror.c.163.com"
        ]
}

EOF
```

**Java 环境 安装脚本**

```
dnf install -y java-1.8.0-openjdk java-1.8.0-openjdk-devel maven
```

最新版 3.8.4 安装脚本

```
cd /usr/local/src/
wget https://dlcdn.apache.org/maven/maven-3/3.8.4/binaries/apache-
maven-3.8.4-bin.tar.gz
tar zxf apache-maven-3.8.4-bin.tar.gz
mv apache-maven-3.8.4 /srv/
rm -f /srv/apache-maven
ln -s /srv/apache-maven-3.8.4 /srv/apache-maven

alternatives --remove mvn /usr/share/maven/bin/mvn
alternatives --install /usr/local/bin/mvn apache-maven-3.8.4
/srv/apache-maven-3.8.4/bin/mvn 0

cp /srv/apache-maven/conf/settings.xml{,.original}
vim /srv/apache-maven/conf/settings.xml <<end > /dev/null 2>&1
:158,158d
:164,164s/$/ -->/
:wq
end

mvn -v
```

**Node 环境**

默认安装

```
dnf install -y nodejs
npm config set registry https://registry.npm.taobao.org
npm install -g cnpm
```

官网最新版

```
dnf remove -y nodejs

cd /usr/local/src
```

```
wget https://nodejs.org/dist/v16.13.1/node-v16.13.1-linux-x64.tar.xz
tar xf node-v16.13.1-linux-x64.tar.xz
mv node-v16.13.1-linux-x64 /srv/node-v16.13.1
rm -f /srv/node
ln -s /srv/node-v16.13.1 /srv/node

alternatives --install /usr/local/bin/node node /srv/node/bin/node
100 \
--slave /usr/local/bin/npm npm /srv/node/bin/npm \
--slave /usr/local/bin/npx npx /srv/node/bin/npx \
--slave /usr/local/bin/corepack corepack /srv/node/bin/corepack

node -v
```

```


dnf remove -y nodejs

cd /usr/local/src
wget https://nodejs.org/dist/v17.2.0/node-v17.2.0-linux-x64.tar.xz
tar xf node-v17.2.0-linux-x64.tar.xz
mv node-v17.2.0-linux-x64 /srv/node-v17.2.0
rm -f /srv/node
ln -s /srv/node-v17.2.0 /srv/node

alternatives --install /usr/local/bin/node node /srv/node/bin/node
100 \
--slave /usr/local/bin/npm npm /srv/node/bin/npm \
--slave /usr/local/bin/npx npx /srv/node/bin/npx \
--slave /usr/local/bin/corepack corepack /srv/node/bin/corepack

node -v
```

## 2.2. 创建用户

初始化GitLab，进入Admin area，单击左侧菜单Users，在这里为gitlab添加用户

创建用户

过程 11.1. 企业内部使用的 Gitlab 初始化

1. 关闭在线用户注册
2. Step 3.
    a. Substep a.
    b. Substep b.

## 2.3. 初始化组

为什么要使用组?

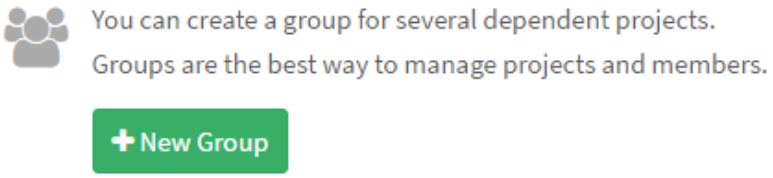
组可以共享标记、里程碑、议题、会员权限和Gitlab Runner 执行器，如果不实用组，就只能一个项目一个项目的去配置。

初始化GitLab组，我比较喜欢使用“域名”作为组名，例如example.com

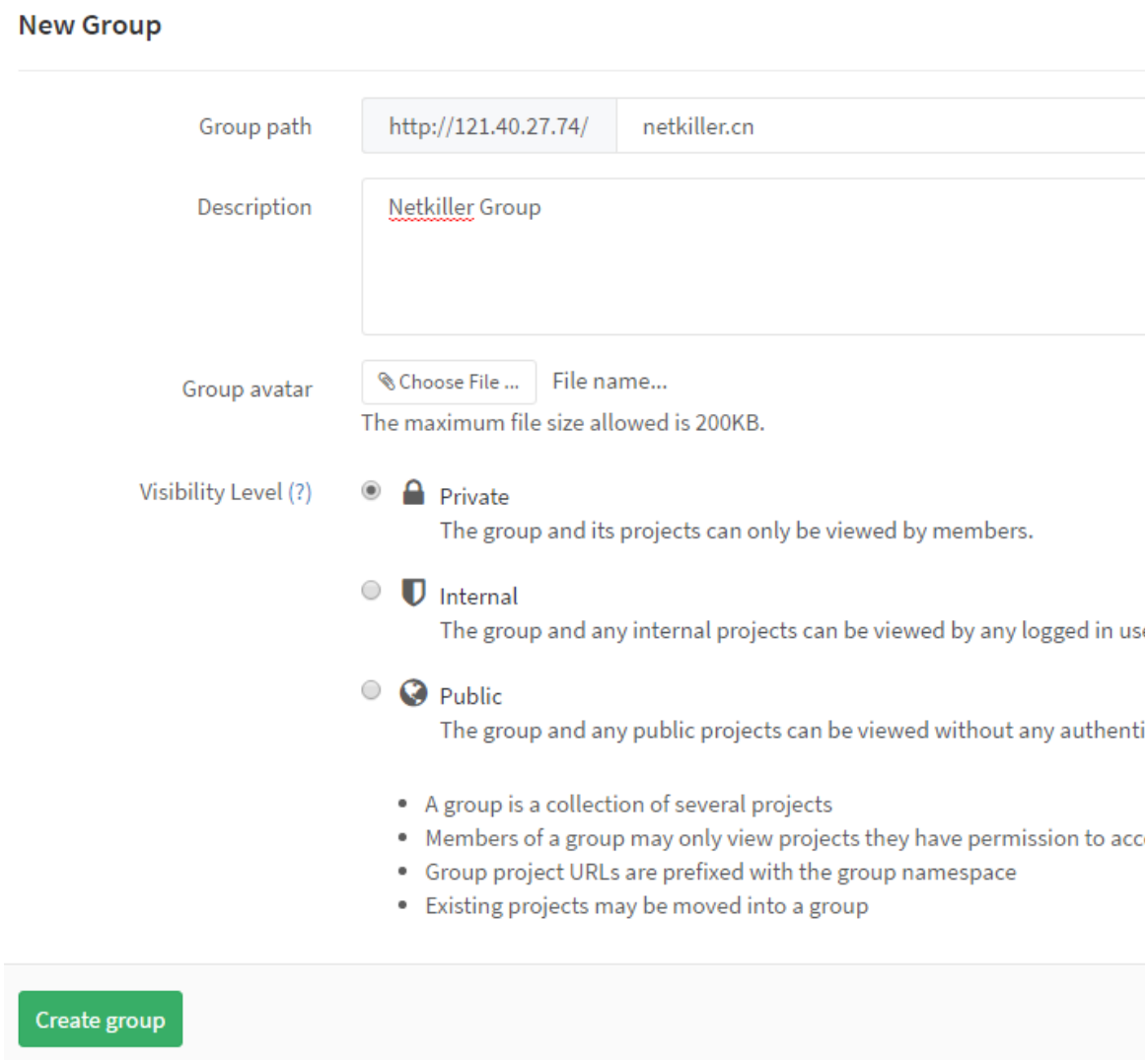
下面是创建组与项目的具体操作步骤

过程 11.2. Gitlab 初始化 - 创建组

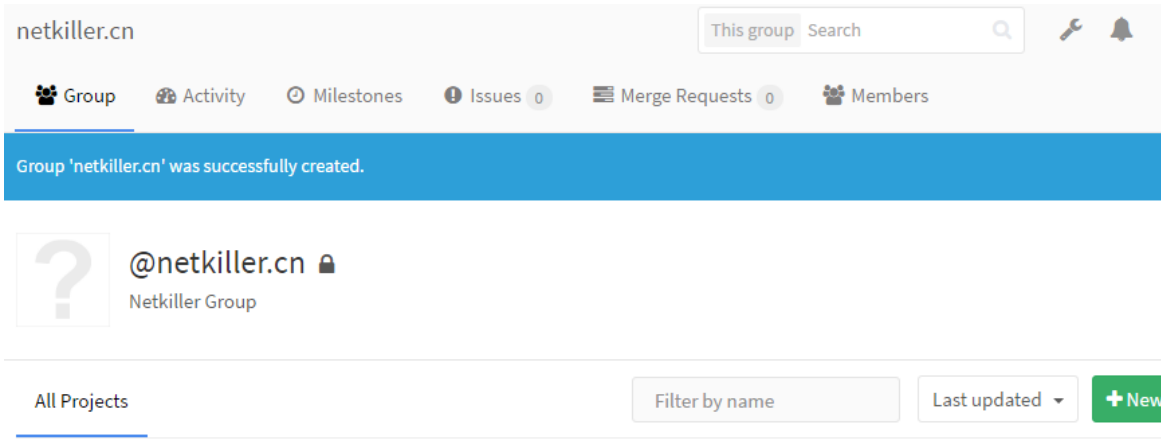
1. 点击 New Group 按钮新建一个组，我习惯每个域一个组，所以我使用 netkiller.cn 作为组名称




2. 输入 netkiller.cn 然后单击 Create group


New Group
Group path
http://121.40.27.74/
netkiller.cn
Description
Netkiller Group
Group avatar
Choose File ...
File name...
The maximum file size allowed is 200KB.
Visibility Level (?)
Private
The group and its projects can only be viewed by members.
Internal
The group and any internal projects can be viewed by any logged in us
Public
The group and any public projects can be viewed without any authenti
A group is a collection of several projects
Members of a group may only view projects they have permission to acc
Group project URLs are prefixed with the group namespace
Existing projects may be moved into a group
Create group

3. 组创建完毕


netkiller.cn
This group
Search
Group
Activity
Milestones
Issues 0
Merge Requests 0
Members
Group 'netkiller.cn' was successfully created.
@netkiller.cn
Netkiller Group
All Projects
Filter by name
Last updated
New

创建组后接下来创建项目

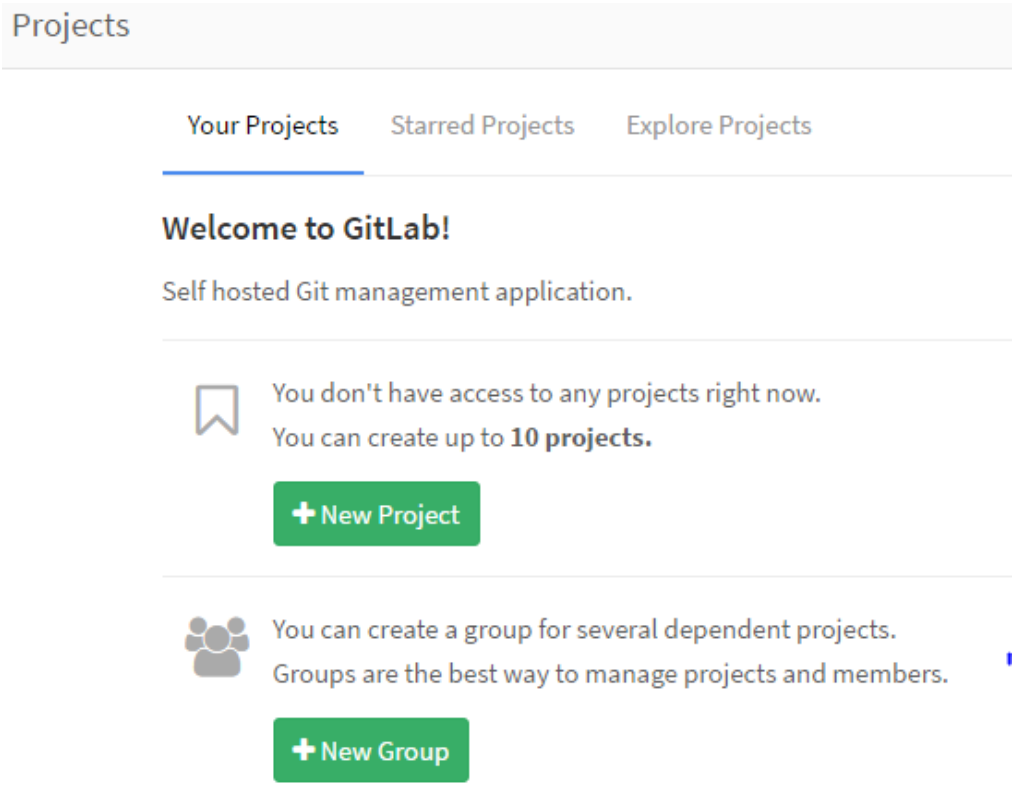
创建项目，我通常会在组下面创建项目，每个域名对应一个项目,例如www.example.com,images.example.com

版本库URL如下

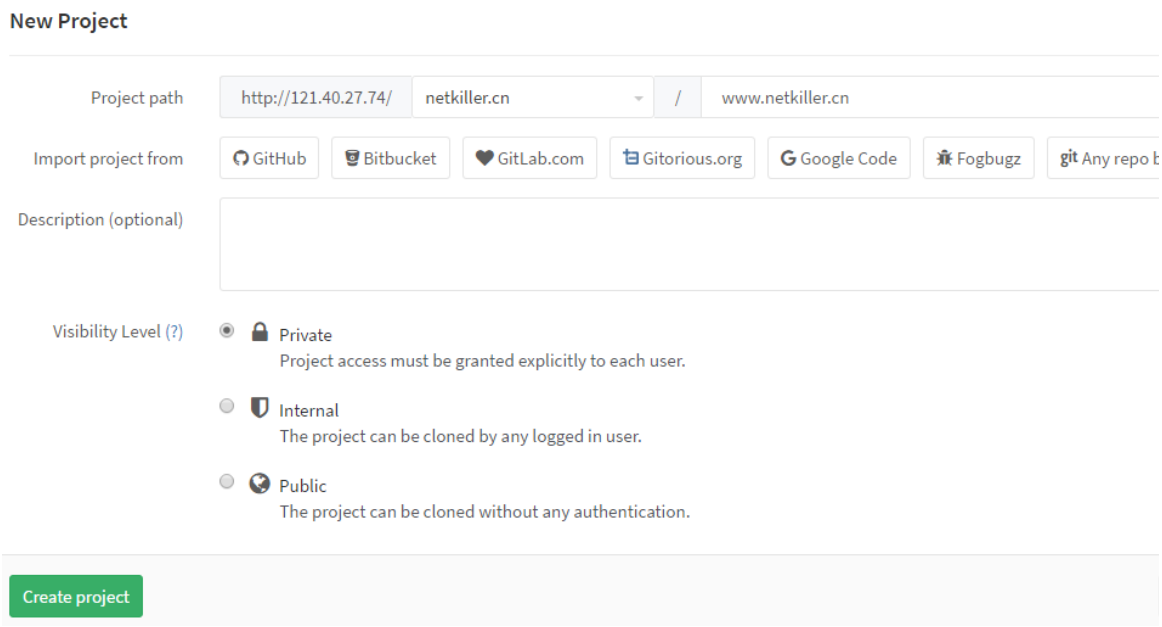
```
http: http://192.168.0.1/example.com/www.example.com.git
ssh: git@192.168.0.1:example.com/www.example.com.git
```

过程 11.3. Gitlab 初始化 - 创建项目

1. 单击 New Project 创建项目




2. 输入 www.netkiller.cn 并点击 Create project 按钮创建项目




3. 项目创建完毕




## 2.4. 初始化标签

参考 github 设置

bug Something isn't working
documentation Improvements or additions to documentation
duplicate This issue or pull request already exists

enhancement New feature or request
good first issue Good for newcomers
help wanted Extra attention is needed
invalid This doesn't seem right
question Further information is requested
wontfix This will not be worked on

通常定义四个状态，开发，测试，升级，完成

## 2.5. 初始化分支

起初我们应对并行开发在Subversion上创建分支，每个任务一个分支，每个Bug一个分支，完成任务或修复bug后合并到开发分支(development)内部测试，然后再进入测试分支(testing)提交给测试组，测试组完成测试，最后进入主干(trunk)。对于Subverion来说每一个分支都是一份拷贝，SVN版本库膨胀的非常快。

Git 解决了Svn 先天不足的分支管理功能，分支在GIT类似快照，同时GIT还提供了 pull request 功能。

我们怎样使用git 的分支功能呢？ 首先我们不再为每个任务创建一个分支，将任务分支放在用户自己的仓库下面，通过 pull request 合并，同时合并过程顺便code review。

master：是主干，只有开发部主管/经理级别拥有权限，只能合并来自testing的代码

testing: 测试分支，测试部拥有权限，此分支不能修改，只能从开发分支合并代码。

development：开发组的分支，Team拥有修改权限，可以合并，可以接受pull request, 可以提交代码

tag 是 Release 本版，开发部主管/经理拥有权限

分支的权限管理：

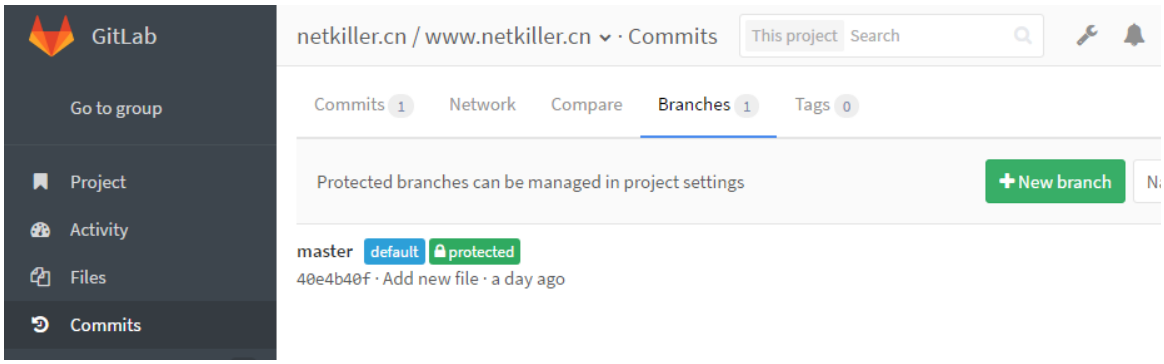
master: 保护

testing：保护

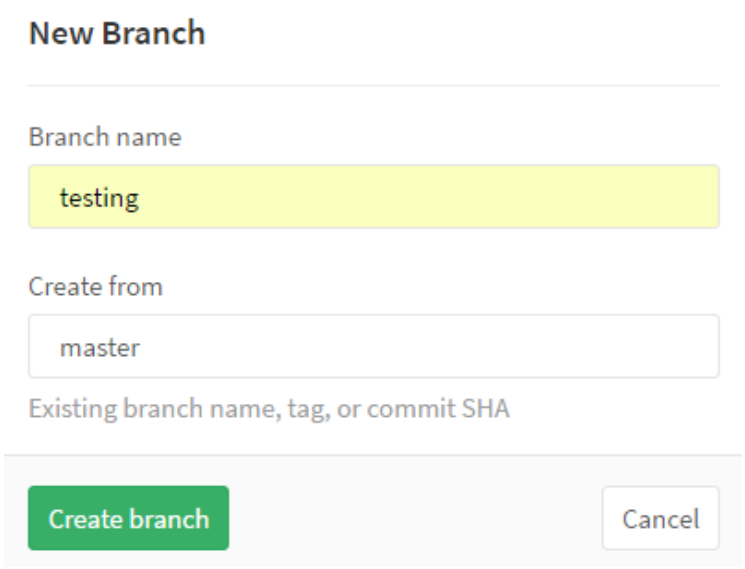
development：保护

过程 11.4. Gitlab 分支应用 - 创建分支

1. 首先，点击左侧 Commits 按钮，然后点击 Branches 按钮进入分支管理

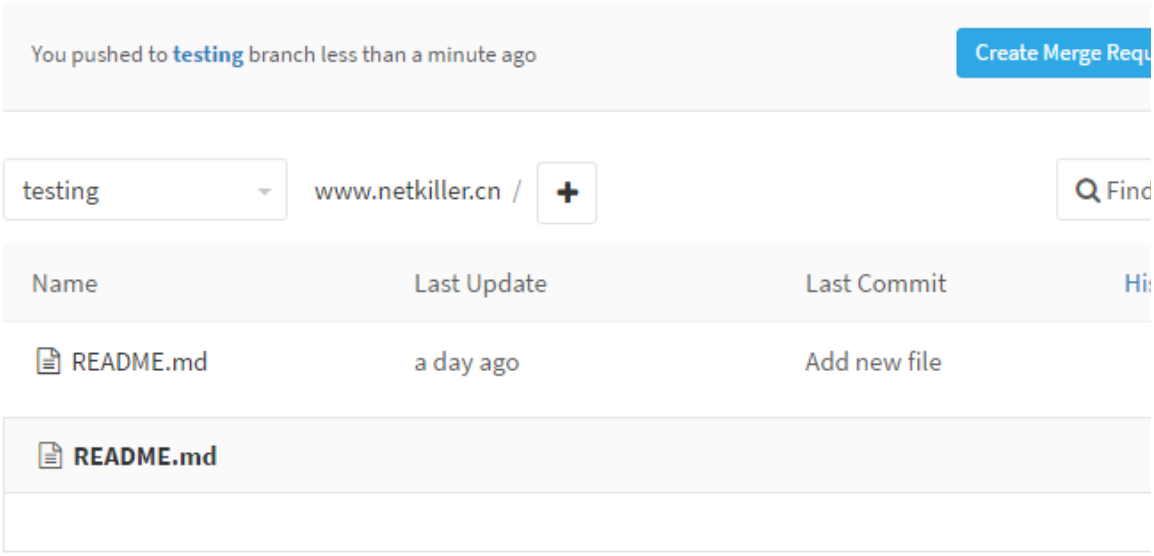



2. 点击 New branch 创建分支




在 Branch name 中输入分支名称，然后点击 Create branch 创建分支

3. 分支已经创建




重复上面步骤，完成development分支的创建。

保护分支：锁定分支，只允允许合并，不允许提交

过程 11.5. 保护分支

1. master

   testing

2. Step 2.

   a.

   b. Substep b.

## 2.6. 部署环境

# 3. 项目管理

## 3.1. 组织架构

开发部

产品部

- 产品经理，产品专员
- 平面设计，UI/UE

开发部

开发部

- 软件项目经理
- 开发组长（根据项目并行开发的产品线而定）
- 高级程序员，中级程序员，初级程序员

测试部

测试部

- 软件测试经理
- 测试组长（根据并行测试的项目数量而定)
- 高级测试工程师（自动化测试），中级测试工程师（功能测试），初级测试工程师（功能测试）

运维部

运维部

- 运维经理
- 运维组长（根据服务器数量而定）
- 高级运维工程师（运维工具研发），中级运维工程师（8小时，处理日常运维），初级运维工程师（7*24小计监控）

### 开发、测试和运维三个部门的关系

- 开发，测试，运维不是三个独立部门，他们相互紧密联系，但又相互制约
- 开发只负责写程序，将运行无误的程序提交至版本库中
- 开发不能私自将程序交给运维部署，也不能将编译好的程序给测试人员
- 测试部只能从版本库提取代码，然后编译，打包，运行，测试
- 不允许测试部将代码交给运维部部署
- 避免代码没有经过版本库流入生产环境，造成线下与线上代码不一致
- 运维部负责部署应用程序，配置管理，只接受测试部确认无误的版本，部署代码只能从版本库中获取

### 权限角色

文档角色：产品，设计

报告角色：测试

开发角色：开发

运维角色：运维

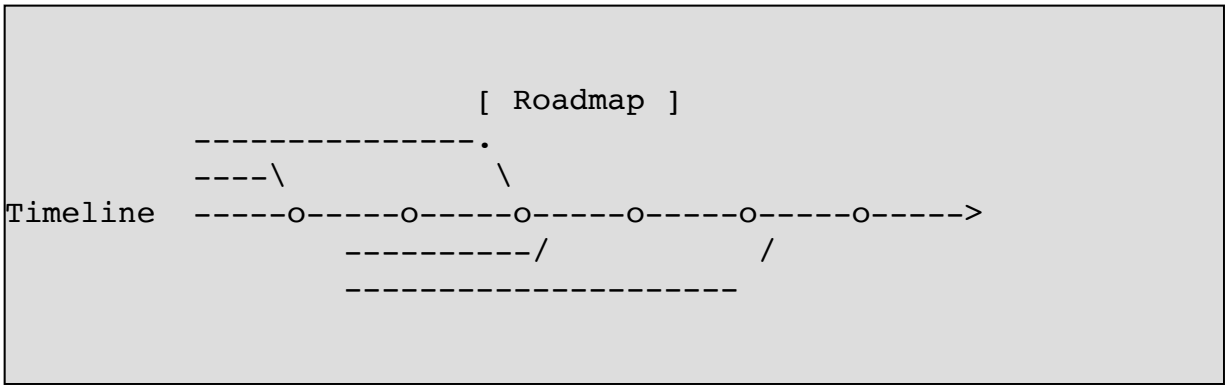
## 3.2. 项目计划

我常常把项目开发计划比做列车时刻表，每一个站对应一个项目节点即里程碑。

列车时刻表的概念来自早年我参与的一个英国项目，我们使用 TRAC 管理项目，这是一个古老的项目管理软件，他是很多现代项目管理软件的雏形，很多思想沿用至今，甚至无法超越它，由于他是 Python 开发，框架古老，后期无人维护更新跟不上时代节奏。 另一个项目模仿它90%的功能叫 Redmine，Redmine 红极一时，但是仍然没有统一江湖。直到 Github/Gitlab 出现，一站式解决了软件项目管理中遇到的各种刚需问题，TRAC，Redmine，Confluence，Bugzilla，Jira, Mantis, BugFree, BugZero…… 慢慢淡出人们视野。

在 TRAC 中，任务叫做 Ticker 翻译成中文就是“票”，项目是沿着 Roadmap 走，走过的路叫时间线 Timeline， 里程碑又形象的比做站点，每个 Milestone 里面是一组 Ticker。每次升级就如同买票上车，火车不等人，同理项目也按照自己的 Roadmap 运行，错过只能等下一班。

项目经理会在即时通信软件中，通知发车时间，需要升级同事就会将自己上手的Ticker代码合并主干，然后等待发车。




火车偶尔也会出现晚点，取消班次，临时停车或不停靠直接开往下一站的情况。项目也是如此：

晚点就是项目延期，取表班次就是停止本次里程碑的上线计划，临时停靠即热修复和紧急上线，不停靠就是跳过本次里程碑，下一个里程碑一次性解决。

我们常常会在即时通信软件中发布发车时间和询问发车时间。

项目计划应该是像列车时刻表一样，一旦你定好，就不能随意修改，必须按照设定的里程碑有条不紊的推进。

我们发现很多国内项目是被任务牵着做，即没有项目路线图，走到那里算那里，觉得差不多了就上线，相当随意。一旦出现交叉，冲突，就会手忙脚乱，回撤更是家常便饭。

## 3.3. 工作流

项目管理需要设计工作流

你会发现 Gitlab 并没有提供工作流的功能？为什么？你是否想过？不仅Gitlab 没有 ，微软的 Microsoft Project 也没有，为什么 Microsoft Office 不提供这种功能？

谈谈我的一段职业生涯，大约在2000年我来到深圳，第一份工就是OA（办公自动化）系统开发，当时有很多公司开发类似产品，也包括金山软件，用友等等。20年过去了，OA没有一个标准，也没有一个成功的产品，OA似乎成为企业数字化转型不上的工具之一，上了之后又发现这东西根本没法使用。

OA 没有成为主流的原因，死结就是工作流，每个公司都有自己的流程，无法统一标准，即使是管理学诞生的西方国家，也没有统一的流程，流程是随着市场和环境不断变化的，没有任何流程能始终延续。经历了德鲁克时代的企业到目前为止保留下来的流程也只有部分行政审批流程。

这就是微软不碰这块，IBM也做，Oracle 也不做的原因。虽然技术上已经又成熟的工作流引擎，图形化配置，使用过的人都表示巨难用，对于非技术的行政人员几乎都放弃了。

但凡设计流程，设计者都会表现自己，最终设计的出的流程无比复杂，看似流程堪称完美，执行起来不是内耗就是受阻。几乎都是做加法思维，能做减法思维的人少之又少。

工作流设计原则

1. 遵循减法原则而不是加法原则，参与人越少越好，节点越少越好。
2. 尽量线性，从一端流向另一段，中间尽量不出现分叉。
3. 尽可能不出现逻辑判断分叉，例如 A审批决定下一步是流向B还是C
4. 避免循环，依赖关系避免循环，即流程后退。

目的是让工作流可操作，易操作，能冗余。

下面这个流程有问题吗?



```
开发人员提测 -> 开发组长审批 -> 技术部门审批 -> 测试部门审批 -> 测试组长审批 -> 分配测试人员
```

稍具规模的企业不都是这样做的吗?

```
      开发人员提测            |          分配测试人员
-----------------------------------------------------
      开发组长审批            |          测试组长审批
-----------------------------------------------------
      技术部门审批            |          测试部门审批
```

矩阵转换一下，看的更清晰，工作流从一段流向另一段，经历两个部门，六个节点。理论上审批超过三层就要控制，超过三层就会影响进度。为什么会出现这种情况?

我做过分析，国内的管理层大可分为两类，一类是着重考察项目过程本身，一类是主要考察项目的参与者和结果，前者着重于时间管理，后者倾向于绩效考核。[2]

第一类管理者，很清楚项目的 Roadmap，所以根本无需做，技术部门审批到测试部门审批这个审批过程，这些工作都是在 Roadmap 会议定定好的，按时发车即可。

第二类管理者，通常是学管理出身，运用管理学工具管理项目，他无法参与项目过程，只能关注时间点和进度，不断催促，他需要知道现在做什么? 什么时候做完? 所以需要事事审批。

## 3.4. 议题

Issues 议题

**Milestones 里程碑**

敏捷开发中可以每周一个里程碑，或者每个月一个里程碑。

## 修正路线图（Roadmap）

每间隔一端时间需要调整一次 Roadmap 的设置，因为有些项目会延期，有些会提前完成，还有需求变更等因素都会影响 Roadmap。

## 工作报告

由于项目是由上至下层层分解下去的，制定了严格的Roadmap，每个参与者知道自己该做什么，如何做，什么时间完成，也就无需再向上汇报工作。

团队所要做的就是按照 Roadmap 的时间和节点走即可。

## 5W2H 任务分配法则

一旦时间点确定，接下来就是分配任务倒指定开发人，任务的分配十分讲究，分配任务要精确描述，不能使用模糊语言，那样会造成误解。我的分配原则是5W2H方法：

- What：做什么事？
- Why：为什么做这件事？有什么意义？目的是什么？有必要吗？
- When：什么时候做，完成的时间是否适当？
- Where：在什么地方做，在什么范围内完成？
- Who：由谁负责做？由谁负责执行？谁更合适？熟练程度低的人能做吗？
- How：怎样做
- How much: 成本 （不是所有岗位都会涉及成本）

## 任务/议题

议题

运维任务

举例，运维任务

- What：为api服务器做负载均衡，多增加一个节点，负载均衡算法采用最小连接数。
- Why：目前api服务器只有一台，如果出现故障将影响到所有业务运行，顾该服务器存在单点故障，需要增加节点。
- When：本周内完成，周末上线。（此处可以写日期）
- Where：在A机柜，低2机位处，连接倒交换机第三个端口。
- Who：XXX负责网络配置，XXX负责上架，XXX 负责验收测试
- How：增加/etc/hosts设置如下
  - api.example.com  127.0.0.1
  - api1.example.com  192.168.2.5
  - api2.example.com  192.168.2.6

开发任务

举例，开发任务

- What：增加图片验证码。
- Why：目前用户注册登陆以及发帖无验证吗，某些用户通过机器人软件批量开户/发广告帖，给管理带来很大困扰。
- When：2014-06-15 开始开发，2014-06-20 12:00 上线。
- Where：用户注册，登陆与发帖处增加该功能，。
- Who：张三负责验证码生成类的开发，李四负责用户注册，登陆UI修改，王五负责发帖UI的修改。
- How：具体怎么操作的细节，此处省略200字...

测试任务

举例，测试任务

- What：测出XXX软件并发性能。
- Why：目前XXX软件在线任务达到200后，用户反映速度慢，经常掉线。
- When：故障时间点10：00AM，需要周二完成测试，周五完成优化，月底上线。（此处可以写日期）
- Where：在AAA分支检出代码，编译后部署到BBB环境。
- Who：XXX负责网络配置，XXX负责软件部署，XXX 负责测试
- How：具体怎么操作的细节，此处省略200字...

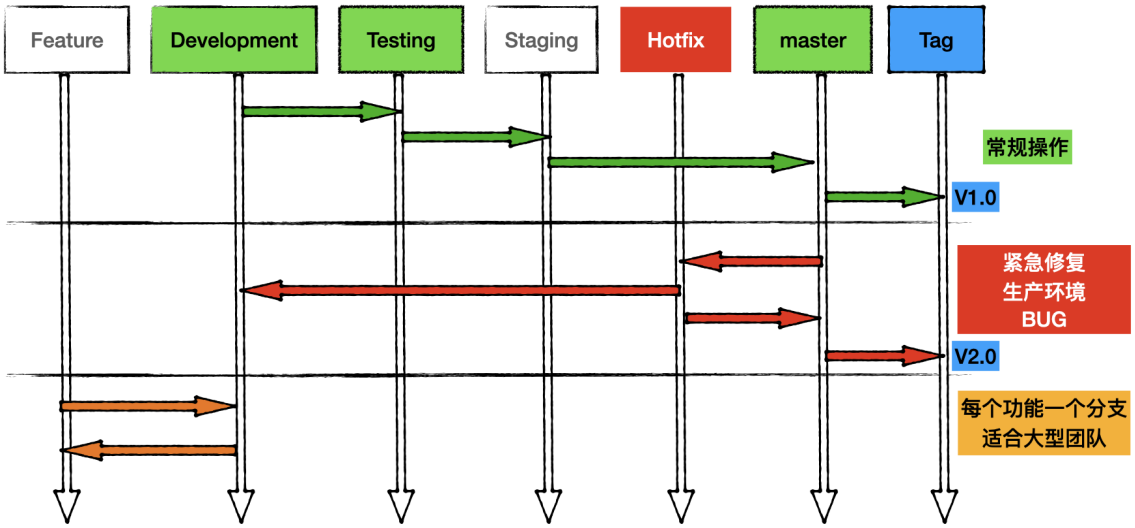
运营任务

举例，促销任务

- What：XXX产品促销。
- Why：目前XXX产品在 XXX 市场占有率 XXX 用户反映 XXX。
- When：促销起始时间 XXX 结束时间 XXX
- Where：AAA 细分市场，BBB区域。
- Who：XXX负责 XX，XXX负责 XX，XXX 负责 XX
- How：具体怎么操作的细节，此处省略200字...
- How much: 成本XXX

## 3.5. 并行开发

## 分支合并流程




多个功能并行开发最常遇到的问题就是冲突，例如A，B，C三个功能同时开发，共用一个分支（development）A开发完成，B功能开发1/3，C功能有BUG，此时升级A功能，B跟C也会被升级上去。

实现并行开发，需要满足两个条件。一是合理的任务分解，二是配套的环境，三是分支的应用。

### 任务分解

任务分解要尽可能解耦，出现交叉合并为一个任务。一个任务对应一个功能，功能与功能之间依赖关系必须理清，避免出现交叉依赖和循环依赖。

```
A -> B -> C
      \-> D -> E
          \--> F
```

### 配套环境

配套环境是指开发和测试环境，参考生产环境，以最小化实例，最小化节点，满足运行项目的环境，尽量减少环境差异，包括硬盘配置差异，网络差异，资源配置差异，以及应用软件安装配置等等差异。

准备配套环境

1. 开发环境(development)，也叫集成开发环境，为开发团队提供共享资源，因为每个程序员在自己的电脑上运行一整套的分布式系统不太现实，所以需要将公共部分抽离出来，集中提供服务，例如数据库，缓存，搜索引擎，配置中心，注册中心等等。
2. 测试环境(testing)，由于开发环境需要频繁合并新功能，部署，重启都会影响正常的测试，例如测试一般，开发环境上加入了新功能，此时会影响测试。所以我们需要一台独立，稳定的测试环境，这个环境由测试人员自己控制，什么时候部署，测试自己说了算。
3. 用户交付测试环境（staging）Stage/UAT 环境，Beta/Preview 演示环境，定期同步生产环境数据库。
4. 功能测试环境(feature/hotfix) 新功能，BUG修复等等。

上面三个环境，至少一台独立服务器，功能测试环境(feature/hotfix)需要若干台服务器。功能测试环境的服务器是共享的，即谁提测谁用，用过之后释放出来。

每个环境都有一整套，配套的服务，例如数据库，缓存，搜索引擎，消息队列等等……

## 代码分支

### 时间线分支

1. 开发分支 Development 面向开发人员
2. 测试分支 Testing 面向测试人员
3. 交付验收分支 Staging 交付验收分支，俗称 UAT 面向测试和客户
4. 生产分支 Production，面向用户

在小公司中通常会省去 UAT 这个环节，从 Testing直接上生产环境

### 分支权限

分支保护的目的，防止被误删除，禁止向该分支提交代码，代码只能通过合并方式进入该分支。

分支的权限管理：

1. master: 保护，不能修改代码，只能合并，只有管理员有权限push
2. staging：保护，不能修改代码，只能合并，只有管理员有权限push
3. testing：保护，不能修改代码，测试人员可以合并 merge
4. development：保护，开发人员可以修改代码，合并，push
5. tag 标签：保护，对应 Release 本版

#### 功能分支

功能分支（Feature），任务分解之后，每个功能对应一个分支，功能分支的代码来自 development 分支，我们会有很多功能分支，开发任务在功能分支上完成开发，开发完成后将任务标记为“测试”，测试部会安排测试环境，部署该分支上的代码，测试结果分为BUG和Pending（测试通过，挂起，等待发车）。

买票上车：在功能分支上，我们有很多开发完成功能，他们处于挂起状态，然后根据升级计划，有序的合并到开发分支，再到测试分支，最后升级到生产环境。

Feature 分支操作步骤：

1. 创建 Issue 议题
2. 从 Development 创建 Feature 分支
3. 获取 Feature 分支代码
4. 修改代码，提交代码，测试代码
5. 合并 Feature 分支到 Development 分支
6. 关闭 Issue 议题

#### 合并流程

代码合并流程



```
Development -> testing -> staging -> master(production)
```

合并分支

从 development 像 testing 分支合并

```
git checkout development
git pull
git checkout testing
git pull
git merge --no-ff "development"
git push
```

testing 分支向 master 分支合并

获取 testing 合并请求的分支

```
git fetch origin
git checkout -b "testing" "origin/testing"
```

如果此前已经执行过，使用下面命令切换分支即可，切换后 pull 代码，看看有什么新提交

```
git checkout "testing"
git pull
```

切换到 master 分支

```
git fetch origin
git checkout "master"
git branch --show-current
git merge --no-ff "testing"
```

将合并结果推送到远程

```
git push origin "master"
```

除了单个文件

从 development 到 testing

```
git checkout development
git pull
checkout testing
git checkout development public/doc/UserGuide.pdf
git status
git commit -a -m '手工合并'
git push
```

从 testing 到 staging

```
git checkout staging
git pull
git checkout testing public/doc/UserGuide.pdf
git commit -a -m '手工合并'
git push
```

从 stage 到 master

```
git checkout master
git pull
git checkout staging public/doc/UserGuide.pdf
git commit -a -m '手工合并'
git push
```

合并分支解决冲突

案例，例如我们从 testing 分支向 master 分支合并代码出现冲突，该如何解决呢?

首先，两个分支拉取最新代码

```
neo@MacBook-Pro-Neo ~/workspace/api.netkiller.cn % git checkout
testing
neo@MacBook-Pro-Neo ~/workspace/api.netkiller.cn % git pull
neo@MacBook-Pro-Neo ~/workspace/api.netkiller.cn % git checkout
master
neo@MacBook-Pro-Neo ~/workspace/api.netkiller.cn % git pull
```

然后合并分支，从 testing 分支向 master 合并

```
neo@MacBook-Pro-Neo ~/workspace/api.netkiller.cn % git merge --
no-ff testing
自动合并 neo-
incar/src/main/java/com/neo/incar/utils/PaperlessConfig.java
冲突（内容）：合并冲突于 neo-
incar/src/main/java/com/neo/incar/utils/PaperlessConfig.java
自动合并失败，修正冲突然后提交修正的结果。
```

出现冲突，编辑冲突文件

```
vim neo-
incar/src/main/java/com/neo/incar/utils/PaperlessConfig.java
```

保存后重看状态

```
neo@MacBook-Pro-Neo ~/workspace/api.netkiller.cn % git status
位于分支 master
您的分支与上游分支 'origin/master' 一致。

您有尚未合并的路径。
  (解决冲突并运行 "git commit")
  (使用 "git merge --abort" 终止合并)

要提交的变更:
        修改:      neo-admin/src/main/resources/application-
prod.yml
        修改:      neo-admin/src/main/resources/application-
test.yml
        修改:      neo-
common/src/main/java/com/neo/common/enums/IncarAttachTypeEnum.ja
va
        修改:      neo-
incar/src/main/java/com/neo/incar/service/impl/IncarAttachServic
eImpl.java

未合并的路径:
  (使用 "git add <文件>..." 标记解决方案)
        双方修改:    neo-
incar/src/main/java/com/neo/incar/utils/PaperlessConfig.java
```

将合并的文件添加到 git

```
```

```
neo@MacBook-Pro-Neo ~/workspace/api.netkiller.cn % git add neo-
incar/src/main/java/com/neo/incar/utils/PaperlessConfig.java
neo@MacBook-Pro-Neo ~/workspace/api.netkiller.cn % git status
位于分支 master
您的分支与上游分支 'origin/master' 一致。

所有冲突已解决但您仍处于合并中。
  （使用 "git commit" 结束合并）

要提交的变更：
        修改：      neo-admin/src/main/resources/application-
prod.yml
        修改：      neo-admin/src/main/resources/application-
test.yml
        修改：      neo-
common/src/main/java/com/neo/common/enums/IncarAttachTypeEnum.ja
va
        修改：      neo-
incar/src/main/java/com/neo/incar/service/impl/IncarAttachServic
eImpl.java
        修改：      neo-
incar/src/main/java/com/neo/incar/utils/PaperlessConfig.java
```

提交代码

```
neo@MacBook-Pro-Neo ~/workspace/api.netkiller.cn % git commit -a
-m '手工合并分支 testing -> master'
[master 3652bf8e] 手工合并分支 testing -> master
```

推送代码

```
neo@MacBook-Pro-Neo ~/workspace/api.netkiller.cn % git push
枚举对象中：1，完成.
对象计数中：100% (1/1)，完成.
写入对象中：100% (1/1)，240 字节 | 240.00 KiB/s，完成.
总共 1（差异 0），复用 0（差异 0），包复用 0
remote:
```

```
remote: To create a merge request for master, visit:
remote:
http://192.168.30.5/netkiller.cn/api.netkiller.cn/-/merge_reques
ts/new?merge_request%5Bsource_branch%5D=master
remote:
To http://192.168.30.5/netkiller.cn/api.netkiller.cn.git
   fcaefaf4..3652bf8e  master -> master
```

#### Hotfix / BUG 分支

Hotfix / BUG 分支与功能分支类似，都是用于存放 BUG，BUG分支会对应缺陷管理系统中的BUG ID，做到缺陷与代码可溯源。

hotfix 分支的使用场景，生产环境发现 bug 需要临时修复，testing 上面有正在进行的项目，不能从 testing -> master 合并，这时可以从 master -> hotfix 创建分支，修复和测试完成后合并到 master 分支，部署 production 环境。最后再将 hotfix 合并到 development 分支

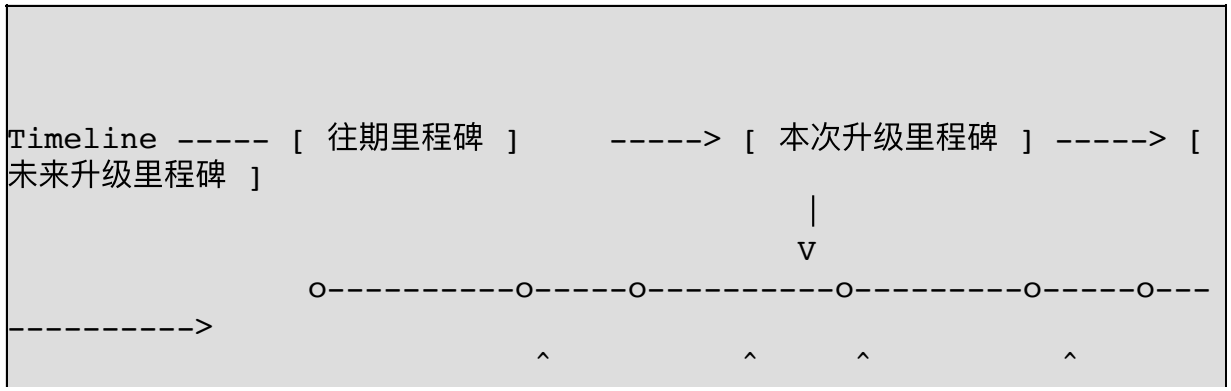
对于中小公司，团队人数少的情况，可以不用建立 BUG 分支，可以在功能分支上完成修复，再合并到 development 分支。

## 前滚和后滚

突发情况，临时决定撤掉某些功能，这是会用到前滚和后滚操作

#### 后滚操作

后滚操作举例



```
^      ^
                                A功能       B功能    C功能      D功能      E
功能     N功能
```

在本次升级的里程碑中，有五个功能搭便车，这个五个功能是按照顺序合并进来的，每次合并都可以找到对应的版本ID。

我们模拟一个场景，这五个功能是市场部的五个活动，现在由于各种原因，活动D这个功能需要撤掉，我们只需要找到 C功能的版本ID，将代码恢复到 C功能，然后重新合并一次 E功能

后滚到 C，然后增加 E功能

```
Timeline ----- [ 往期里程碑 ]     -----> [ 本次升级里程碑 ] -----> [
未来升级里程碑 ]
                                          |
                                          V
                  o----------o-----o----------o---------o
                             ^           ^     ^           ^
^
                                A功能       B功能    C功能      D作废
E功能
```

前滚操作

当撤掉的功能需要恢复时就是前滚操作

```
Timeline ----- [ 往期里程碑 ]     -----> [ 本次升级里程碑 ] -----> [
未来升级里程碑 ]
                                          |
                                          V
                  o----------o-----o----------o---------o
                             ^           ^     ^           ^
```

```
^
                                A功能       B功能    C功能      D功能      E
功能
```

前滚到任意提交版本

```


Timeline ----- [ 往期里程碑 ]     -----> [ 本次升级里程碑 ] -----> [
未来升级里程碑 ]
                                            |
                                            V
                  o----------o-----o----------o---------o-----o---
----------->
                                ^           ^     ^           ^
^      ^
                                A功能       B功能    C功能      D作废      E
功能     N功能
```

前滚后后滚常见操作

导出最后一次修改过的文件

有时我们希望把刚刚修改的文件复制出来，同时维持原有的目录结构，这样可能交给运维直接覆盖服务器上的代码。我们可以使用下面的命令完成这样的操作，而不用一个一个文件的复制。

```
git archive -o update.zip HEAD $(git diff --name-only HEAD^)
```

导出指定版本区间修改过的文件

首先使用git log查看日志，找到指定的 commit ID号。

```
$ git log
commit ee808bb4b3ed6b7c0e7b24eeec39d299b6054dd0
Author: 168 <lineagelx@126.com>
Date:   Thu Oct 22 13:12:11 2015 +0800

    统计代码

commit 3e68ddef50eec39acea1b0e20fe68ff2217cf62b
Author: netkiller <netkiller@msn.com>
Date:   Fri Oct 16 14:39:10 2015 +0800

    页面修改

commit b111c253321fb4b9c5858302a0707ba0adc3cd07
Author: netkiller <netkiller@msn.com>
Date:   Wed Oct 14 17:51:55 2015 +0800

    数据库连接

commit 4a21667a576b2f18a7db8bdcddbd3ba305554ccb
Author: netkiller <netkiller@msn.com>
Date:   Wed Oct 14 17:27:15 2015 +0800

    init repo
```

导入 b111c253321fb4b9c5858302a0707ba0adc3cd07 至 ee808bb4b3ed6b7c0e7b24eeec39d299b6054dd0 间修改过的文件。

```
$ git archive -o update2.zip HEAD $(git diff --name-only
b111c253321fb4b9c5858302a0707ba0adc3cd07)
```

回撤提交

首先 reset 到指定的版本，根据实际情况选择 --mixed 还是 --hard

```
git reset --mixed 096392721f105686fc3cdafcb4159439a66b0e5b --
or
git reset --hard 33ba6503b4fa8eed35182262770e4eab646396cd --
```

```
git push origin --force --all
or
git push --force --progress "origin" master:master
```

撤回单个文件提交

例如撤回 project/src/main/java/cn/netkiller/controller/DemoSceneController.java 到上一个版本

```
→  api.netkiller.cn git:(testing) git log
project/src/main/java/cn/netkiller/controller/DemoSceneControlle
r.java

commit b4609646ee60927fe4c1c563d07e78f63ab106ea (HEAD ->
testing, origin/testing)
Author: Neo Chen <netkiller@msn.com>
Date:   Wed Nov 17 18:49:27 2021 +0800

    手工合并，临时提交

commit bc96eb68ad73d5248c8135609191c51e258edf10
Author: Tom <tom@qq.com>
Date:   Thu Oct 21 16:29:20 2021 +0800

    获取激活场景

commit d564ea25bd556324f1f576357563a8ee77b3bdd9
Author: Tom <tom@qq.com>
Date:   Thu Oct 21 15:15:26 2021 +0800
```

```
    获取激活场景

commit d5a40165ad24a3a021fe58c6d78e0b7d97ab3cc5
Author: Tom <tom@qq.com>
Date:   Thu Oct 21 14:43:16 2021 +0800

    新增场景角色增加

commit aa98662cb9e781e328ee3d5cec23af29c81050d9
Author: Tom <tom@qq.com>
Date:   Thu Oct 21 09:55:29 2021 +0800

    新增场景角色增加

commit 140d22a8d4ea7fcc775d4372e8beb6d854831512
Author: Jerry <jerry@qq.com>
Date:   Sat Oct 16 15:27:30 2021 +0800

    场景接口修改

commit 2ddbb1ff933de663305db2396d99030c938c267a
Author: Tom <tom@qq.com>
Date:   Fri Oct 15 10:55:30 2021 +0800
```

只显示最后五条记录

```
→  api.netkiller.cn git:(testing) git log -5
project/src/main/java/cn/netkiller/controller/DemoSceneControlle
r.java
```

```
→  api.netkiller.cn git:(testing) git reset
bc96eb68ad73d5248c8135609191c51e258edf10
project/src/main/java/cn/netkiller/controller/DemoSceneControlle
r.java
Unstaged changes after reset:
M
```

```
project/src/main/java/cn/netkiller/controller/DemoSceneControlle
r.java
```

```
→  api.netkiller.cn git:(testing) ✗ git status
On branch testing
Your branch is up to date with 'origin/testing'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:
project/src/main/java/cn/netkiller/controller/DemoSceneControlle
r.java

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working
directory)
        modified:
project/src/main/java/cn/netkiller/controller/DemoSceneControlle
r.java

→  api.netkiller.cn git:(testing) ✗ git add
project/src/main/java/cn/netkiller/controller/DemoSceneControlle
r.java
→  api.netkiller.cn git:(testing) ✗ git commit -m '恢复到上一个版
本'
[testing 9959acd4] 恢复到上一个版本
 1 file changed, 6 insertions(+), 8 deletions(-)
```

## 提交代码怎样写注释信息

将任务ID写在代码提交注释信息当中，可以实现代码与任务的绑定，我们在项目平台上查看代码的时候，可以直接点击编号跳到对应的任务。这样便清晰的直到本次提交对应的任何和需求文档，便于代码溯源。

**Fixed Bug**

```
svn ci -m "- Fixed bug #53412 (your comment)"
```

**Implemented**

```
svn ci -m "- Implemented FR #53271, FR #52410 (Building multiple
XXXX binary)"
```

**Add**

```
svn ci -m "- Add Feature #534 (your message)"
```

## 3.6. 升级与发布相关

### 分支与版本的关系

各种版本来自与那个分支，它们的对应关系是什么?

分支与版本的关系:

1. Alpha 内部测试环境，面向测试人员，不稳定版本。来自 testing 分支
2. Beta / Preview / Unstable 新特性，预览版，面向用户体验。来自 staging 分支
3. Stable = Release 稳定版，发行版来自 master/main 以及 tag 标签

### 分支与标签的区别

分支与标签的区别是，分支中的代码可以修改，标签可以视为只读分支。

**Release Notes**

Release Notes 撰写说明

当一个项目升级时，需要写一个文档纪录这次变动

1. 内容包括
2. 新增了什么
3. 更改了什么
4. 修复了什么
5. 未解决得问题
6. 改善了什么
7. 忽略了什么

常用信息类型

```
New
Changed
Fixed
Unresolved
Improved
Ignore
```

**例 11.3. Example - Release Notes**

```
NEW - xxxxxxxxxxxx
CHANGED - xxxxxxxxxxxx
FIXED - xxxxxxxxxx
UNRESOLVED - xxxxxxxxx
IMPROVED - xxxxxxxxx
```

你也同样可以参考很多开源组织编写的Release Notes，例如apache, mysql, php 等等

**License**

使用开源软件需要知道各种 License 区别，以免出现法律纠纷。

GPL 你可以免费使用，但修改后必须开源。

GPLv3 你可以免费使用，但修改后必须开源，不允许加入闭源商业代码。

BSD 你可以免费使用，修改后可不开源，基本上你可以我所欲为。

Linux 中有许多BSD代码，但BSD却不能移植Linux 代码到BSD中，这是因为GPL License。

http://www.apache.org/licenses/

## 3.7. 代码审查

---

[2] *《Netkiller Management 手札》*

# 4. 通过GPG签名提交代码

## 4.1. 创建证书

```
Neo-iMac:workspace neo$ gpg --quick-generate-key netkiller@msn.com
About to create a key for:
    "netkiller@msn.com"

Continue? (Y/n) y
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
gpg: key 2F05850CF88E8B3A marked as ultimately trusted
gpg: directory '/Users/neo/.gnupg/openpgp-revocs.d' created
gpg: revocation certificate stored as '/Users/neo/.gnupg/openpgp-
revocs.d/085C991D914F0EBD60FFE33B2F05850CF88E8B3A.rev'
public and secret key created and signed.

pub   ed25519 2021-11-04 [SC] [expires: 2023-11-04]
      085C991D914F0EBD60FFE33B2F05850CF88E8B3A
uid                      netkiller@msn.com
sub   cv25519 2021-11-04 [E]
```

查看证书

```
Neo-iMac:workspace neo$ gpg -k
/Users/neo/.gnupg/pubring.kbx
------------------------------
pub   rsa2048 2021-10-08 [SC] [expires: 2023-10-08]
      70CECE32E5D67D12B95ED1E7F01C0CAEAAA458E6
uid           [ unknown] Neo Chen <netkiller@msn.com>
sub   rsa2048 2021-10-08 [E] [expires: 2023-10-08]
```

如果你已有证书，使用下面命令导出公钥和私钥证书

```
Neo-iMac:workspace neo$ gpg --import public.key
gpg: /Users/neo/.gnupg/trustdb.gpg: trustdb created
gpg: key F01C0CAEAAA458E6: public key "Neo Chen <netkiller@msn.com>" imported
gpg: Total number processed: 1
gpg:               imported: 1
```

测试签名

```
Neo-iMac:workspace neo$ echo "test" | gpg --clearsign
```

```
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA256

test
-----BEGIN PGP SIGNATURE-----

iQEzBAEBCAAdFiEEcM7OMuXWfRK5XtHn8BwMrqqkWOYFAmGDsLMACgkQ8BwMrqqk
WOYhcAf8C6XfBwEaVA1HVUdcqMVdq404hnRzeGOTu8XifTF+MMT0nA/GPbHQY76i
17pskWtjrj6y1aZ39/GiEnuXUqgfqvrWAWJymAMLi/v0xFJIJseCWoZ952zi5w6/
uWsM5GIMz0uBuu7/DfN8+XXaeyyvzhYvIMsKsbNEnDOLXORsUFWBNsyhZWaQa699
EbPLMBMP2xIdXr1/D+T6qfIf7iCgRPaPKizcZcymaCE1wFBOGQjgAzgFgQ8HCkCV
K1vtIMCBL9BJbCV5YolwB0Yrvaoi4EnforaM8L+7GBvBuEOsa3YNmUgcD6oLyWZX
LwSk4dGHC1Efk2Cy+e+XYGO3GQIBMw==
=7wHY
-----END PGP SIGNATURE-----
```

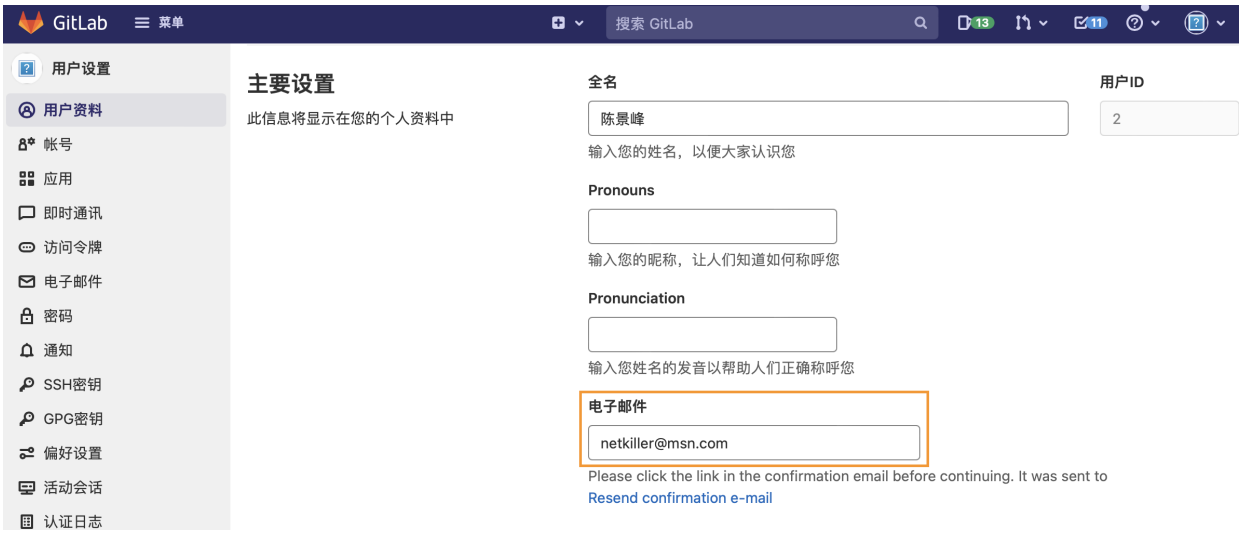
## 4.2. 配置 Gitlab GPG

导出公钥证书

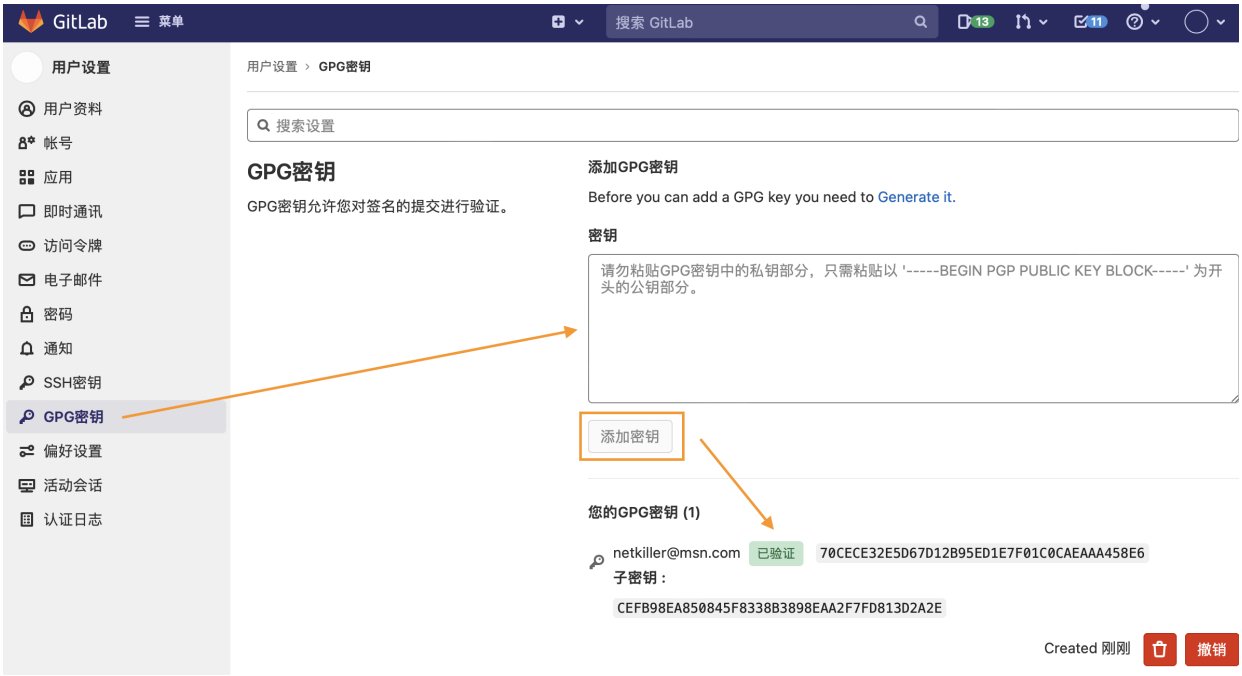
```
Neo-iMac:workspace neo$ gpg --armor --export netkiller@msn.com
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQENBGFgEfYBCACXIT6K61G3uwwFPxwKaKirZyhSnhh22CwTPEGkeviyXCCfpr2X
d8bjibOCwO8bigXFjaKuTikHmpppy7B/CKJ4OlsLXnoMnnSmynntudJ+jcGmC3/0
QE1nvDzqbe8L5KJ3TMgAuDUSp3QWXqIAXxQfEABLl49wJ11envwTXJVPG/ks2U3m
b/QAFZqd3AxUpEzASIKbtiB5JE/rxnhyZH7fHkt3vU2N3qAcUQ67cJN+thkMEsOo
wnp9eGvDv1qBieQKK5DzxC+a04p4cWv5z0rV4IEE3bRR2wKW45HI9Lmgz8zZyFcO
gTV1HshRYnDBVgzcnyombQfzbd76g5tBQC2vABEBAAG0HE5lbyBDaGVuIDxuZXRr
aWxsZXJAbXNuLmNvbT6JAVQEEwEIAD4WIQRwzs4y5dZ9Erle0efwHAyuqqRY5gUC
YWAR9gIbAwUJA8JnAAULCQgHAgYVCgkICwIEFgIDAQIeAQIXgAAKCRDwHAyuqqRY
5v8UB/9GuKFO86BprJUfPBOE4sqUPH44kLupVMuvM+XaBkuOQIT5q37MPoUpb3Uj
g7tV3Nc+6/VLTCDTERKEfV7PRke2UjjjdYf4EYA2PMVVtHEnWngKhVcMkD2iEvR2
ViCQQ6sCve5lefMQcPyLVMX1ynMOQCNiVcOZjfv+vW2H4BynZC6kG472a3TjoTKz
TlbrsiK/n7CSMLsevQh9UrG2n24rKfxQiWCo9tVxyWjcYLEO6yRzOxC+KnEBVr30
O86qn8A/soKY3PEWWUWCcve9g7Km3OVMQf3kJo+xy3hDafDhuBTvNUH3Bz9lwXa3
Sune2h5J77AbgUCHZSw4MZEWdknxuQENBGFgEfYBCACVjr3QGs1b2cei5sHyBO59
hC8VgehGs42jiItaNQSLpBO8g8Z2UbwcB9y3QWrbBITXfj1Jmy+XJInbc3FYYoZE
9bVHb+KjIR4JLqWrieGCWaKzl78ByRRKfQWO0di5OVMQBWg3yzd2dRJnvpa8+W60
ksHoyL0wcXLDbCxYxTNmpHacbvEJYe4zxYJxMyD3V8BEF/r6HtA8ZrhPHrI23AF6
iqSK7PIKAFBLIbU9jinncy/Vbv1DgXZrh72cxhl0n7hTgX8tI2gFRpz+p10iKX2B
zab3F4Ac1YNBy/F9tqIeCPBGK4CmFTtZkzpokevrIfzLThWuqRGIRtnwqlvMKHxz
ABEBAAGJATwEGAEIACYWIQRwzs4y5dZ9Erle0efwHAyuqqRY5gUCYWAR9gIbDAUJ
A8JnAAAKCRDwHAyuqqRY5hpyB/4hh3qMpSOtjOFS5nWGrYNb/o//YRKDwORjJUdI
t0A1RvQkIZEQ9MYR67xpQ8OO2JrsznB7yF0D/Wrmleuu9lY9IVgdaNdNYRRzAdam
MuU5hYe6cUkNudjekhWb2J77EIaL70g9tboEHlQEdVe/FesLg1iZVlPZaaN6UjN6
81AcVw3nloBgIHQUWWsdsSW5sTfymnMhtUfJVlPfeEagLIioTvTzUqy0LjjeIOhR
B1EXkjs/4g/20c/X9JH8z+QwnZ0lmHy9HzUl+g3zLQ7Vu2xaTwHgBWl5sGdkDkJX
RiSdzxKOlGfxNN0e5r7fUYvlCkqOvAFvdpZANcVYkWurjWt2
=W+8i
-----END PGP PUBLIC KEY BLOCK-----
```

确保邮箱与GPG密钥邮箱相同，否则会提示“未验证”




将公钥复制到输入框，然后点击“添加密钥”按钮




## 4.3. 配置 Git

查看密钥用户ID

```
Neo-iMac:workspace neo$ gpg --list-secret-keys --keyid-format=long
/Users/neo/.gnupg/pubring.kbx
------------------------------
sec   rsa2048/F01C0CAEAAA458E6 2021-10-08 [SC] [expires: 2023-10-08]
      70CECE32E5D67D12B95ED1E7F01C0CAEAAA458E6
uid                 [ultimate] Neo Chen <netkiller@msn.com>
ssb   rsa2048/EAA2F7FD813D2A2E 2021-10-08 [E] [expires: 2023-10-08]
```

```

```

注意：可以使用 F01C0CAEAAA458E6 也可以使用电子邮箱

### 全局配置

全局配置适用与所有仓库

```
Neo-iMac:workspace neo$ git config --global user.signingkey netkiller@msn.com
Neo-iMac:workspace neo$ git config --global commit.gpgsign true

Neo-iMac:workspace neo$ echo 'export GPG_TTY=$(tty)' >> /.bash_profile
Neo-iMac:workspace neo$ export GPG_TTY=$(tty)
Neo-iMac:workspace neo$ git commit -S -m "your commit message"
```

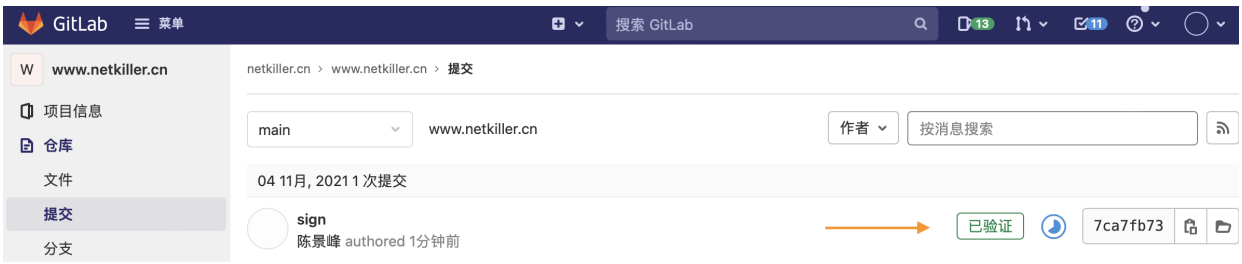
### 本地配置

本地仓库配置，可以单独配置每个仓库的证书。

```
Neo-iMac:workspace neo$ git config --local user.email netkiller@msn.com
Neo-iMac:workspace neo$ git config --local user.signingkey netkiller@msn.com
Neo-iMac:workspace neo$ git config --local commit.gpgsign true
Neo-iMac:workspace neo$ echo 'export GPG_TTY=$(tty)' >> /.bash_profile
Neo-iMac:workspace neo$ git config --list --local | grep user
user.email=netkiller@msn.com
user.signingkey=netkiller@msn.com
```

### 提交代码

提交代码后可以看到“已验证”图标




## 4.4. FAQ

### error: gpg failed to sign the data

```
Neo-iMac:www.netkiller.cn neo$ git commit -a -m 'sign'
error: gpg failed to sign the data
fatal: failed to write commit object
```

```

```

解决方案

```
Neo-iMac:workspace neo$ export GPG_TTY=$(tty)
```

# 5. CI / CD

https://gitlab.com/gitlab-examples

```
Gitlab(仓库) -> Gitlab Runner（持续集成/部署） -> Remote host（远程部署主机）
```

## 5.1. 远程服务器配置

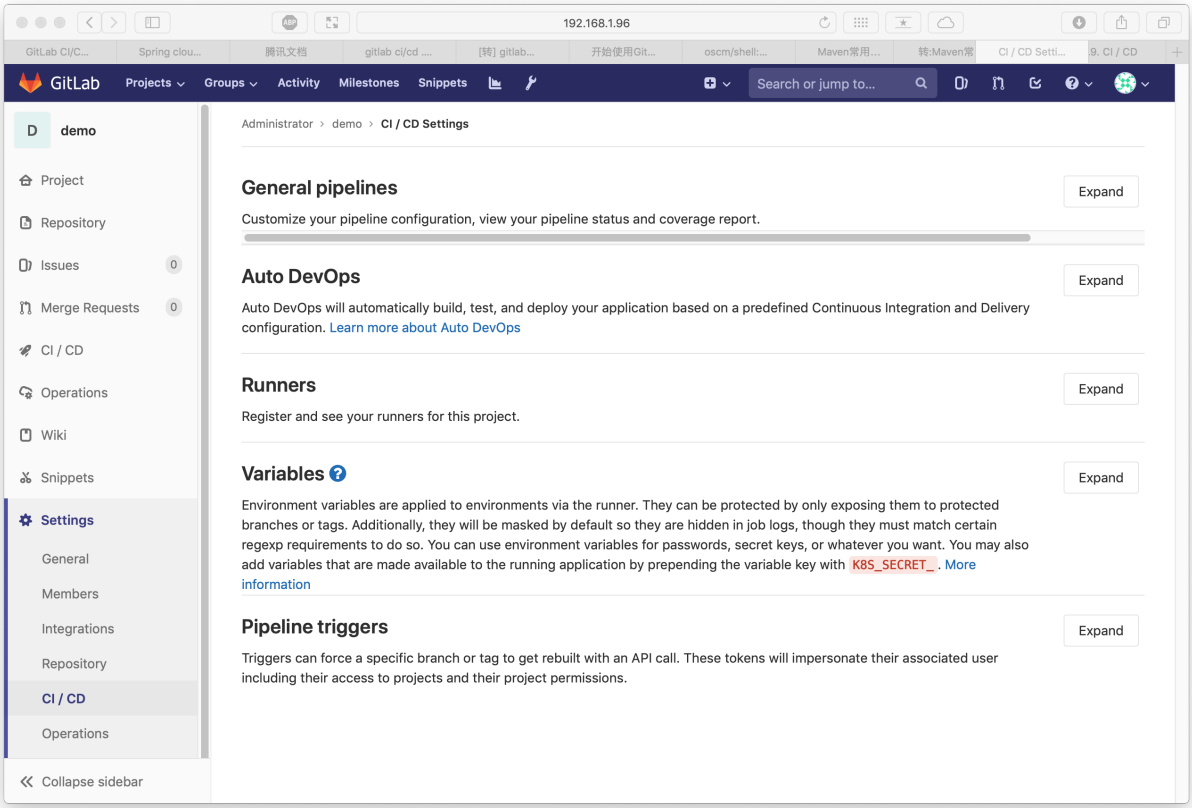
为远程服务器创建 www 用户，我们将使用该用户远程部署，远程启动程序。

```
[root@netkiller ~]# groupadd -g 80 www
[root@netkiller ~]# adduser -o --uid 80 --gid 80 -G wheel -c "Web Application" www
[root@netkiller ~]# id www
uid=80(www) gid=80(www) groups=80(www),10(wheel)
[root@netkiller ~]# PASSWORD=$(cat /dev/urandom | tr -dc [:alnum:] | head -c 32)
[root@netkiller ~]# echo www:${PASSWORD} | chpasswd
[root@netkiller ~]# echo "www password: ${PASSWORD}"
www password: 0Uz1heY9v9KJyRKbvTi0VlAzfEoFW9GH
```

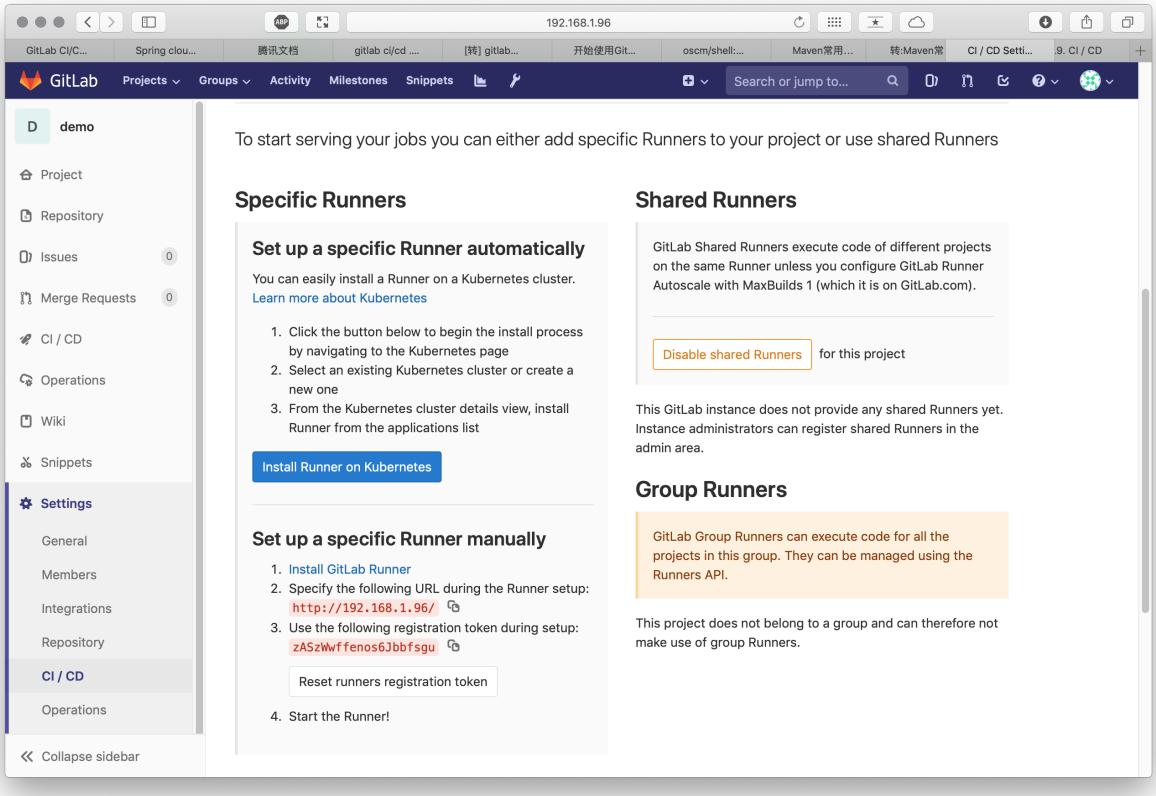
```
mkdir -p /opt/netkiller.cn/www.netkiller.cn
chown www:www -R /opt/netkiller.cn
```

## 5.2. 配置 CI / CD

进入项目设置界面，点击 Settings，再点击 CI / CD




点击 Expand 按钮 展开 Runners




这时可以看到 Set up a specific Runner manually, 后面会用到 http://192.168.1.96/ 和 zASzWwffenos6Jbbfsgu

## 安装 GitLab Runner

#### Install GitLab Runner

```
curl -L "https://packages.gitlab.com/install/repositories/runner/gitlab-runner/script.rpm.sh" |
sudo bash
dnf install gitlab-runner

cp /etc/gitlab-runner/config.toml{,.original}

systemctl enable gitlab-runner
```

## 注册 gitlab-runner

使用 SSH 登录 Gitlab runner 服务器，运行 gitlab-runner register

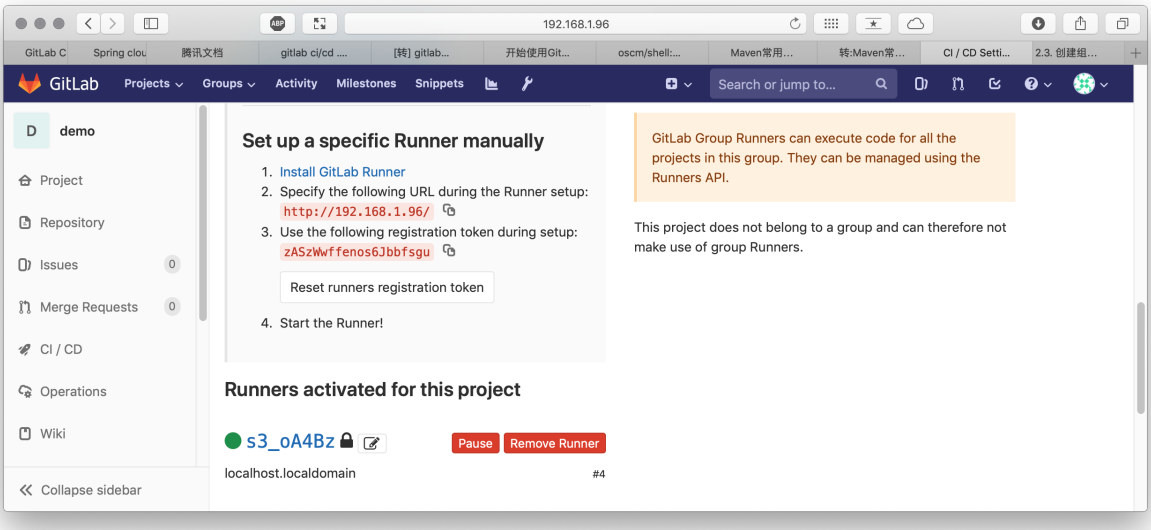
```
[root@localhost ~]# gitlab-runner register
Runtime platform                                    arch=amd64 os=linux pid=92925
revision=ac2a293c version=11.11.2
```

```
Running in system-mode.

Please enter the gitlab-ci coordinator URL (e.g. https://gitlab.com/):
http://192.168.1.96/
Please enter the gitlab-ci token for this runner:
zASzWwffenos6Jbbfsgu
Please enter the gitlab-ci description for this runner:
[localhost.localdomain]:
Please enter the gitlab-ci tags for this runner (comma separated):

Registering runner... succeeded                     runner=zASzWwff
Please enter the executor: docker, docker-ssh, shell, ssh, docker-ssh+machine, parallels,
virtualbox, docker+machine, kubernetes:
shell
Runner registered successfully. Feel free to start it, but if it's running already the config
should be automatically reloaded!
```

返回 gitlab 查看注册状态




并发链接数设置

编辑 /etc/gitlab-runner/config.toml 配置文件，修改 concurrent 数量

```
[root@localhost ~]# grep con /etc/gitlab-runner/config.toml
concurrent = 10
```

## 5.3. Shell 执行器

**Registering Runners**

注册 **Gitlab Runner** 为 **Shell** 执行器

```
[root@gitlab ~]# gitlab-runner register
Runtime platform                                    arch=amd64 os=linux pid=1020084
revision=c1edb478 version=14.0.1
Running in system-mode.

Enter the GitLab instance URL (for example, https://gitlab.com/):
http://git.netkiller.cn/
Enter the registration token:
DyKdKyaJaq5KN-irgNGz
Enter a description for the runner:
[gitlab]:
Enter tags for the runner (comma-separated):

Registering runner... succeeded                     runner=DyKdKyaJ
Enter an executor: parallels, virtualbox, docker+machine, custom, docker, docker-ssh, shell,
ssh, docker-ssh+machine, kubernetes:
shell
Runner registered successfully. Feel free to start it, but if it's running already the config
should be automatically reloaded!
```

/etc/gitlab-runner/config.toml 配置文件

```
[root@gitlab ~]# cat /etc/gitlab-runner/config.toml
concurrent = 1
check_interval = 0

[session_server]
  session_timeout = 1800

[[runners]]
  name = "gitlab"
  url = "http://git.netkiller.cn/"
  token = "kVkzjDM74xZUN-aKbdPp"
  executor = "shell"
  [runners.custom_build_dir]
  [runners.cache]
    [runners.cache.s3]
    [runners.cache.gcs]
    [runners.cache.azure]
```

### 生成 SSH 证书

持续集成和部署运行在 gitlab-runner 用户下，切换到 gitlab-runner 用户

```
[root@gitlab ~]# su - gitlab-runner
Last login: Mon Jul 19 19:01:37 CST 2021
```

生成 SSH 证书

```
[gitlab-runner@gitlab ~]$ ssh-keygen
Generating public/private rsa key pair.
```

```
Enter file in which to save the key (/home/gitlab-runner/.ssh/id_rsa):
Created directory '/home/gitlab-runner/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/gitlab-runner/.ssh/id_rsa.
Your public key has been saved in /home/gitlab-runner/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:l90LYBeSF9l9JHXJUHeO+IyvscCziz4C8vFNpJoKEjo gitlab-runner@gitlab
The key's randomart image is:
+---[RSA 3072]----+
|          ..o===B|
|          ..oo.**|
|          o.o . o|
|        .. = =   |
|.      oS o + +  |
|... o . .o   o . |
|E  o * o  + . o  |
|.o  + o o. + +   |
|  ..   oo.o.o    |
+----[SHA256]-----+
[gitlab-runner@gitlab ~]$
```

正常情况下，当我们链接一个 SSH 主机，会让我们输入 yes 确认继续链接。

```
[gitlab-runner@gitlab ~]$ ssh www@192.168.40.10
The authenticity of host '192.168.40.10 (192.168.40.10)' can't be established.
ECDSA key fingerprint is SHA256:xmFF266MPdXhnlAljS+QWhQsw6jOw1sOwQXRr/PHi2w.
Are you sure you want to continue connecting (yes/no/[fingerprint])?
```

配置 SSH

```
[gitlab-runner@gitlab ~]$ cat > ~/.ssh/config <<'EOF'
Host *
        ServerAliveInterval=30
        StrictHostKeyChecking no
        UserKnownHostsFile=/dev/null
EOF

chmod 600 -R ~/.ssh/config
```

授权远程执行 Shell

```
[gitlab-runner@gitlab ~]$ ssh-copy-id www@www.netkiller.cn
```

数据库环境

在构建过程中，我们需要备份数据库/同步数据库，下面安装了一些所需的工具

```
```

```
[root@localhost ~]# dnf install -y mysql
```

设置数据库备份账号和密码，这里偷懒使用了 root 账号，生产环境请创建专用的备份账号。

```
[root@localhost ~]# su - gitlab-runner
Last login: Wed Sep  1 19:17:48 CST 2021
[gitlab-runner@localhost ~]$ vim ~/.my.cnf
[gitlab-runner@localhost ~]$ cat ~/.my.cnf
[mysql]
user=root
password=test

[mysqldump]
user=root
password=test
```

测试数据库是否畅通

```
[gitlab-runner@localhost ~]$ mysql -h mysql.netkiller.cn
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 37602
Server version: 8.0.21 Source distribution

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

**Java 环境**

JRE：java-11-openjdk

JDK：java-11-openjdk-devel

```
[root@gitlab ~]# dnf install -y java-11-openjdk java-11-openjdk-devel
[root@gitlab ~]# dnf install -y maven
```

修改 Maven 镜像路

```
[root@gitlab ~]# vim /etc/maven/settings.xml
  <mirrors>
    <mirror>
      <id>aliyun</id>
```

```
      <name>aliyun maven</name>
      <url>http://maven.aliyun.com/nexus/content/groups/public/</url>
      <mirrorOf>central</mirrorOf>
    </mirror>
  </mirrors>
```

#### 安装最新版 maven

如果需要安装最新版本 maven 使用下面脚本。

```
#!/bin/bash

cd /usr/local/src/
wget https://mirrors.bfsu.edu.cn/apache/maven/maven-3/3.8.2/binaries/apache-maven-3.8.2-
bin.tar.gz
tar zxf apache-maven-3.8.2-bin.tar.gz
mv apache-maven-3.8.2 /srv/
rm -f /srv/apache-maven
ln -s /srv/apache-maven-3.8.2 /srv/apache-maven

alternatives --install /usr/local/bin/mvn apache-maven-3.8.2 /srv/apache-maven-3.8.2/bin/mvn 0
```

```
[root@localhost src]# mvn -v
Apache Maven 3.8.2 (ea98e05a04480131370aa0c110b8c54cf726c06f)
Maven home: /srv/apache-maven-3.8.2
Java version: 17-ea, vendor: Red Hat, Inc., runtime: /usr/lib/jvm/java-17-openjdk-17.0.0.0.26-
0.2.ea.el8.x86_64
Default locale: en_US, platform encoding: ANSI_X3.4-1968
OS name: "linux", version: "4.18.0-338.el8.x86_64", arch: "amd64", family: "unix"
```

apache-maven-3.8.2 配置

```
[root@localhost ~]# vim /srv/apache-maven/conf/settings.xml
  <mirrors>
    <!-- mirror
     | Specifies a repository mirror site to use instead of a given repository. The repository
that
     | this mirror serves has an ID that matches the mirrorOf element of this mirror. IDs are
used
     | for inheritance and direct lookup purposes, and must be unique across the set of
mirrors.
     |
    <mirror>
      <id>mirrorId</id>
      <mirrorOf>repositoryId</mirrorOf>
      <name>Human Readable Name for this Mirror.</name>
      <url>http://my.repository.com/repo/path</url>
    </mirror>
     -->
    <mirror>
      <id>maven-default-http-blocker</id>
      <mirrorOf>external:http:*</mirrorOf>
      <name>Pseudo repository to mirror external repositories initially using HTTP.</name>
      <url>http://0.0.0.0/</url>
```

```
      <blocked>true</blocked>
    </mirror>
  </mirrors>
```

apache-maven-3.8.2 默认会阻止其他镜像，需要会去掉 maven-default-http-blocker 配置

切换到 gitlab-runner 用户，随便运行一下 mvn 命令，这样就会产生 ~/.m2 文件夹

```
[root@gitlab ~]# su - gitlab-runner
[gitlab-runner@gitlab ~]$ mvn -v
```

**mvnd**

mvnd 是一个实验产品，用于替代 maven 编译速度比较快

```
cd /usr/local/src
wget https://github.com/apache/maven-mvnd/releases/download/0.7.1/mvnd-0.7.1-linux-amd64.zip
unzip mvnd-0.7.1-linux-amd64.zip
mv mvnd-0.7.1-linux-amd64 /srv/mvnd-0.7.1
ln -s /srv/mvnd-0.7.1 /srv/mvnd

alternatives --remove mvnd /usr/local/bin/mvnd
alternatives --install /usr/local/bin/mvnd mvnd-0.7.1 /srv/mvnd-0.7.1/bin/mvnd 0
```

修改配置文件 mvnd.properties 制定 JAVA_HOME

```
[root@localhost cloud.netkiller.cn]# grep java.home /srv/mvnd/conf/mvnd.properties
java.home=/usr/lib/jvm/java
```

### NodeJS

```
[root@netkiller ~]# dnf install -y nodejs
```

安装 cnpm

```
[root@netkiller ~]# npm config set registry https://registry.npm.taobao.org
[root@netkiller ~]# npm config get registry
https://registry.npm.taobao.org/
[root@netkiller ~]# npm install -g cnpm
```

yarn

```
[root@netkiller ~]# curl -sL https://dl.yarnpkg.com/rpm/yarn.repo -o /etc/yum.repos.d/yarn.repo
[root@netkiller ~]# dnf install -y yarn
```

```
yarn config set registry https://registry.npm.taobao.org
```

pm2 进程管理

```
[root@netkiller ~]# npm install -g pm2
```

设置 pm2 启动开启

```
[root@netkiller ~]# pm2 startup
[root@netkiller ~]# pm2 save --force
[root@netkiller ~]# systemctl enable pm2-root
[root@netkiller ~]# systemctl start pm2-root
[root@netkiller ~]# systemctl status pm2-root
```

**Python 环境**

```
[root@localhost ~]# dnf install -y python39
```

**远程执行 sudo 提示密码**

```
[gitlab-runner@gitlab api.netkiller.cn]$ ssh www@192.168.40.10 "sudo ls"
Warning: Permanently added '192.168.40.10' (ECDSA) to the list of known hosts.
sudo: a terminal is required to read the password; either use the -S option to read from
standard input or configure an askpass helper
```

解决方案一

```
ssh -t www@www.netkiller.cn "echo <yourpassword> |sudo -S <yourcommand>"
```

解决方案二

```
cat > /etc/sudoers.d/www <<-EOF
www    ALL=(ALL)    NOPASSWD: ALL
EOF
```

### 5.4. tags 的使用方法

tags 是给 Gitlab Runner 打个标签，我的用法是多次注册，例如 shell 执行器的标签是 shell, Docker 执行器的标签是 docker，这样便可以在.gitlab-ci.yml文件中来选择使用那个执行器来触发操作。

下面是 Shell 执行器

```
[root@localhost ~]# gitlab-runner register
Runtime platform                                    arch=amd64 os=linux pid=268363
revision=58ba2b95 version=14.2.0
Running in system-mode.

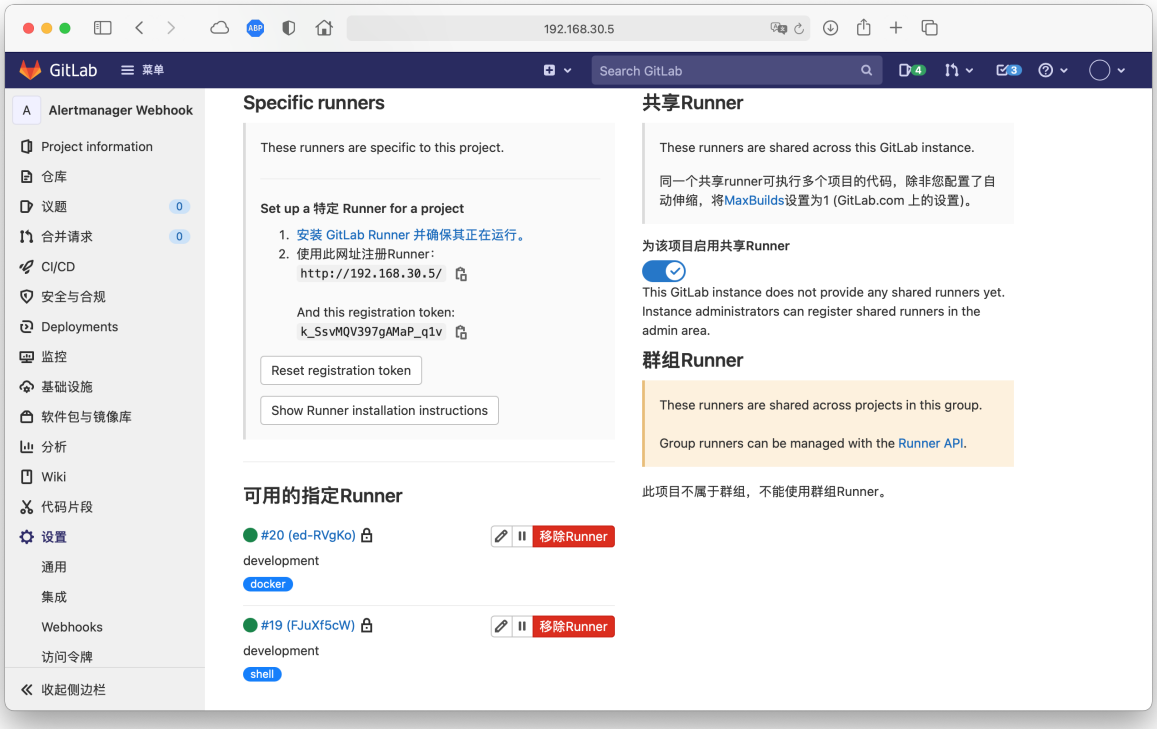
Enter the GitLab instance URL (for example, https://gitlab.com/):
http://git.netkiller.cn/
Enter the registration token:
k_SsvMQV397gAMaP_q1v
Enter a description for the runner:
[localhost.localdomain]: development
Enter tags for the runner (comma-separated):
shell
Registering runner... succeeded                     runner=k_SsvMQV
Enter an executor: docker, docker-ssh, virtualbox, docker-ssh+machine, kubernetes, custom,
parallels, shell, ssh, docker+machine:
shell
Runner registered successfully. Feel free to start it, but if it's running already the config
should be automatically reloaded!
```

下面是 Docker 执行器

```
[root@localhost ~]# gitlab-runner register
Runtime platform                                    arch=amd64 os=linux pid=268397
revision=58ba2b95 version=14.2.0
Running in system-mode.

Enter the GitLab instance URL (for example, https://gitlab.com/):
http://git.netkiller.cn/
Enter the registration token:
k_SsvMQV397gAMaP_q1v
Enter a description for the runner:
[localhost.localdomain]: development
Enter tags for the runner (comma-separated):
docker
Registering runner... succeeded                     runner=k_SsvMQV
Enter an executor: custom, docker-ssh, parallels, shell, ssh, docker-ssh+machine, docker,
virtualbox, docker+machine, kubernetes:
docker
Enter the default Docker image (for example, ruby:2.6):
maven:latest
Runner registered successfully. Feel free to start it, but if it's running already the config
should be automatically reloaded!
```

注册后的效果




```
[root@localhost ~]# cat /etc/gitlab-runner/config.toml
concurrent = 1
check_interval = 0

[session_server]
  session_timeout = 1800

[[runners]]
  name = "development"
  url = "http://git.netkiller.cn/"
  token = "EztTBypKRW5ibtC5rs2h"
  executor = "shell"
  [runners.custom_build_dir]
  [runners.cache]
    [runners.cache.s3]
    [runners.cache.gcs]
    [runners.cache.azure]

[[runners]]
  name = "development"
  url = "http://git.netkiller.cn/"
  token = "51948sQbQsXGV-RxFMty"
  executor = "docker"
  [runners.custom_build_dir]
  [runners.cache]
    [runners.cache.s3]
    [runners.cache.gcs]
```

```
    [runners.cache.azure]
  [runners.docker]
    tls_verify = false
    image = "maven:latest"
    privileged = false
    disable_entrypoint_overwrite = false
    oom_kill_disable = false
    disable_cache = false
    volumes = ["/cache"]
    shm_size = 0
```

### 5.5. Docker 执行器

gitlab-runner 用户需要 访问 /var/run/docker.sock 所以需要将 gitlab-runner 用户加入到 docker 组中。

```
[root@gitlab ~]# ll /var/run/docker.sock
srw-rw---- 1 root docker 0 Nov 25 17:04 /var/run/docker.sock

[root@gitlab ~]# id gitlab-runner
uid=989(gitlab-runner) gid=984(gitlab-runner) groups=984(gitlab-runner)

[root@gitlab ~]# usermod -aG docker gitlab-runner

[root@gitlab ~]# id gitlab-runner
uid=989(gitlab-runner) gid=984(gitlab-runner) groups=984(gitlab-runner),991(docker)
```

注册 Docker 执行器

```
[root@localhost ~]# gitlab-runner register
Runtime platform                                    arch=amd64 os=linux pid=268397
revision=58ba2b95 version=14.2.0
Running in system-mode.

Enter the GitLab instance URL (for example, https://gitlab.com/):
http://git.netkiller.cn/
Enter the registration token:
k_SsvMQV397gAMaP_q1v
Enter a description for the runner:
[localhost.localdomain]: development
Enter tags for the runner (comma-separated):
docker
Registering runner... succeeded                     runner=k_SsvMQV
Enter an executor: custom, docker-ssh, parallels, shell, ssh, docker-ssh+machine, docker,
virtualbox, docker+machine, kubernetes:
docker
Enter the default Docker image (for example, ruby:2.6):
maven:latest
Runner registered successfully. Feel free to start it, but if it's running already the config
should be automatically reloaded!
```

配置缓存

```
[root@localhost ~]# cat /etc/gitlab-runner/config.toml
```

```
concurrent = 1
check_interval = 0

[session_server]
  session_timeout = 1800

[[runners]]
  name = "development"
  url = "http://192.168.30.5/"
  token = "EztTBypKRW5ibtC5rs2h"
  executor = "shell"
  [runners.custom_build_dir]
  [runners.cache]
    [runners.cache.s3]
    [runners.cache.gcs]
    [runners.cache.azure]

[[runners]]
  name = "development"
  url = "http://192.168.30.5/"
  token = "GP-ozvd6uw2nDxyRohZ-"
  executor = "docker"
  [runners.custom_build_dir]
  [runners.cache]
    [runners.cache.s3]
    [runners.cache.gcs]
    [runners.cache.azure]
  [runners.docker]
    tls_verify = false
    image = "maven:latest"
    privileged = false
    disable_entrypoint_overwrite = false
    oom_kill_disable = false
    disable_cache = false
    volumes = ["/cache","/root/.m2"]
    pull_policy = ["never"]
    shm_size = 0
```

volumes = ["/cache","/root/.m2"] 将 Maven 仓库缓存

.gitlab-ci.yaml 编排脚本

```
cache:
  untracked: true

stages:
  - build
  - test
  - deploy

build-job:
  image: maven:3.8.2-openjdk-17
  stage: build
  tags:
    - docker
  script:
    - mvn clean package -Dmaven.test.skip=true
    - ls target/*.jar
  artifacts:
    name: "$CI_PROJECT_NAME"
    paths:
      - target/*.jar
```

```
test-job:
  image: maven:3.8.2-openjdk-17
  stage: test
  variables:
    GIT_STRATEGY: none
  tags:
    - docker
  script:
    - mvn test

deploy-job:
  stage: deploy
  variables:
    GIT_STRATEGY: none
    HOST: 192.168.30.14
    DOCKER_HOST: unix:///var/run/docker.sock mvn clean install docker:build
  environment:
    name: development
    url: https://api.netkiller.cn
  only:
    - development
  tags:
    - shell
  before_script:
    - mvn docker:build -DpushImage
    # - mvn docker:push
    - rm -rf *.sql.gz
    - mysqldump -hmysql.netkiller.cn test | gzip > test.$(date -u +%Y-%m-%d.%H:%M:%S).sql.gz
  artifacts:
    name: "$CI_PROJECT_NAME"
    paths:
      - ./*.sql.gz
  script:
    - scp src/main/docker/docker-compose.yaml www@$HOST:/opt/netkiller.cn/api.netkiller.cn/
    - ssh www@$HOST "sudo docker-compose -f /opt/netkiller.cn/api.netkiller.cn/docker-
compose.yaml up"
    - ssh www@$HOST "sudo docker-compose -f /opt/netkiller.cn/api.netkiller.cn/docker-
compose.yaml restart"
```

### 5.6. Kubernetes executor

```
[root@agent-5 ~]# gitlab-runner register
Runtime platform                                    arch=amd64 os=linux pid=1259091
revision=43b2dc3d version=15.4.0
Running in system-mode.

Enter the GitLab instance URL (for example, https://gitlab.com/):
https://gitlab.netkiller.cn/
Enter the registration token:
GR1348941WLrZvRebkiCocQgdGFwC
Enter a description for the runner:
[agent-5]: Kubernetes executor
Enter tags for the runner (comma-separated):
kubernetes
Enter optional maintenance note for the runner:

Registering runner... succeeded                     runner=GR1348941WLrZvReb
Enter an executor: docker-ssh, shell, docker+machine, docker-ssh+machine, kubernetes, custom,
docker, parallels, ssh, virtualbox:
kubernetes
Runner registered successfully. Feel free to start it, but if it's running already the config
```

```
should be automatically reloaded!

Configuration (with the authentication token) was saved in "/etc/gitlab-runner/config.toml"
```

/etc/gitlab-runner/config.toml

```
[root@agent-5 ~]# cat /etc/gitlab-runner/config.toml
concurrent = 1
check_interval = 0

[session_server]
  session_timeout = 1800

[[runners]]
  name = "Kubernetes executor"
  url = "https://gitlab.netkiller.cn/"
  id = 3
  token = "5J6amB15rYWie_zGscFC"
  token_obtained_at = 2022-10-19T07:16:07Z
  token_expires_at = 0001-01-01T00:00:00Z
  executor = "kubernetes"
  [runners.custom_build_dir]
  [runners.cache]
    [runners.cache.s3]
    [runners.cache.gcs]
    [runners.cache.azure]
  [runners.kubernetes]
    host = ""
    bearer_token_overwrite_allowed = false
    image = ""
    namespace = ""
    namespace_overwrite_allowed = ""
    pod_labels_overwrite_allowed = ""
    service_account_overwrite_allowed = ""
    pod_annotations_overwrite_allowed = ""
    [runners.kubernetes.affinity]
    [runners.kubernetes.pod_security_context]
    [runners.kubernetes.init_permissions_container_security_context]
      [runners.kubernetes.init_permissions_container_security_context.capabilities]
    [runners.kubernetes.build_container_security_context]
      [runners.kubernetes.build_container_security_context.capabilities]
    [runners.kubernetes.helper_container_security_context]
      [runners.kubernetes.helper_container_security_context.capabilities]
    [runners.kubernetes.service_container_security_context]
      [runners.kubernetes.service_container_security_context.capabilities]
    [runners.kubernetes.volumes]
    [runners.kubernetes.dns_config]
    [runners.kubernetes.container_lifecycle]
```

将CA证书复制给 gitlab-runner

```
mkdir /etc/ssl/kubernetes
[root@master ~]# scp /var/lib/rancher/k3s/server/tls/server-ca.crt root@agent-
5:/etc/ssl/kubernetes/ca.crt
```

或者用下面命令查看 CA 证书，然后保存到 /etc/ssl/kubernetes/ca.crt 文件

```
[gitlab-runner@agent-5 ~]$ kubectl config view --raw --minify --flatten -o
jsonpath='{.clusters[].cluster.certificate-authority-data}' | base64 -d
-----BEGIN CERTIFICATE-----
MIIBeDCCAR2gAwIBAgIBADAKBggqhkjOPQQDAjAjMSEwHwYDVQQDDBhrM3Mtc2Vy
dmVyLWNhQDE2NjI2MTYwOTMwHhcNMjIwOTA4MDU0ODEzWhcNMzIwOTA1MDU0ODEz
WjAjMSEwHwYDVQQDDBhrM3Mtc2VydmVyLWNhQDE2NjI2MTYwOTMwWTATBgcqhkjO
PQIBBggqhkjOPQMBBwNCAARGT8u7K3jFNKGid7qMWSYUMuv+kYQzvk5RQHFyEXA6
zNnGd0PBpDvsKpZGjkIwJnla0v98nFzsK6hp9eEDIVw3o0IwQDAOBgNVHQ8BAf8E
BAMCAqQwDwYDVR0TAQH/BAUwAwEB/zAdBgNVHQ4EFgQU7IRDLJoNv7bF8wkkJ4yQ
FA6gTBowCgYIKoZIzj0EAwIDSQAwRgIhAKMiz13pxq+IIXZfZT5R+Lh+pDoX2H1u
AskoLxoAutCPAiEA4ubxiK1DqjatxGb1/ovMLd4pfcPeAvg1AIokwhFhueU=
-----END CERTIFICATE-----
```

```
[root@master ~]# kubectl create serviceaccount secrets
serviceaccount/secrets created

[root@master ~]# kubectl create token secrets

[root@master ~]# cat role.yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: secrets
  namespace: default
rules:
  - apiGroups: ["*"]
    resources: ["pods"]
    verbs: ["list", "get", "watch", "create", "delete"]
  - apiGroups: ["*"]
    resources: ["pods/exec"]
    verbs: ["create"]
  - apiGroups: ["*"]
    resources: ["pods/log"]
    verbs: ["get"]
  - apiGroups: ["*"]
    resources: ["pods/attach"]
    verbs: ["list", "get", "create", "delete", "update"]
  - apiGroups: ["*"]
    resources: ["secrets"]
    verbs: ["list", "get", "create", "delete", "update"]
  - apiGroups: ["*"]
    resources: ["configmaps"]
    verbs: ["list", "get", "create", "delete", "update"]

[root@master ~]# kubectl create rolebinding gitlab-runner-binding --role=secrets --
serviceaccount=default:secrets
```

```
[[runners]]
  name = "Kubernetes executor"
  url = "https://gitlab.netkiller.cn/"
  id = 3
  token = "5J6amB15rYWie_zGscFC"
  token_obtained_at = 2022-10-19T07:16:07Z
  token_expires_at = 0001-01-01T00:00:00Z
  executor = "kubernetes"
  [runners.custom_build_dir]
  [runners.cache]
```

```
    [runners.cache.s3]
    [runners.cache.gcs]
    [runners.cache.azure]
  [runners.kubernetes]
    host = "https://k8s.netkiller.cn:6443"
    ca_file = "/etc/ssl/kubernetes/ca.crt"
    tls_verify = true
    bearer_token_overwrite_allowed = true
    image = ""
    namespace = ""
    namespace_overwrite_allowed = ""
    pod_labels_overwrite_allowed = ""
    service_account_overwrite_allowed = ""
    pod_annotations_overwrite_allowed = ""
    [runners.kubernetes.affinity]
    [runners.kubernetes.pod_security_context]
    [runners.kubernetes.init_permissions_container_security_context]
      [runners.kubernetes.init_permissions_container_security_context.capabilities]
    [runners.kubernetes.build_container_security_context]
      [runners.kubernetes.build_container_security_context.capabilities]
    [runners.kubernetes.helper_container_security_context]
      [runners.kubernetes.helper_container_security_context.capabilities]
    [runners.kubernetes.service_container_security_context]
      [runners.kubernetes.service_container_security_context.capabilities]
    [runners.kubernetes.volumes]
    [runners.kubernetes.dns_config]
    [runners.kubernetes.container_lifecycle]
```

.gitlab-ci.yml

```
cache:
#   untracked: true
  paths:
    - target/

#variables:
#  KUBERNETES_SERVICE_ACCOUNT: secrets
#  KUBERNETES_BEARER_TOKEN:
eyJhbGciOiJSUzI1NiIsImtpZCI6IktCOHRvYlZOLXFPRmEyb1JWdlQxSzBvN0tvZF9HNFBGRnlraDR5UU1jakkifQ.eyJh
dWQiOlsiaHR0cHM6Ly9rdWJlcm5ldGVzLmRlZmF1bHQuc3ZjLmNsdXN0ZXIubG9jYWwiLCJrM3MiXSwiZXhwIjoxNjY2MTc
0Mzk1LCJpYXQiOjE2NjYxNzA3OTUsImlzcyI6Imh0dHBzOi8va3ViZXJuZXRlcy5kZWZhdWx0LnN2Yy5jbHVzdGVyLmxvY2
FsIiwia3ViZXJuZXRlcy5pbyI6eyJuYW1lc3BhY2UiOiJkZWZhdWx0Iiwic2VydmljZWFjY291bnQiOnsibmFtZSI6InNlY
3JldHMiLCJ1aWQiOiIxMTM5NjVlMy1iZGVkLTQ5NGEtOGMyNS0zYjU3OTFmMTIzZjEifX0sIm5iZiI6MTY2NjE3MDc5NSwi
c3ViIjoic3lzdGVtOnNlcnZpY2VhY2NvdW50OmRlZmF1bHQ6c2VjcmV0cyJ9.KvTpp7vplWIBWZFKYK-
zPFk0NJhoiMHHiE-0Bj3qmFvHaxF2D3A8YrsEfQxSIJ1p8J5IZSPYiAX_BqoNi4-fziFsJIIbza0bcPj-
RWLoBY2Nz0gyzw93r2to71WS90OSxryWRMMmqr8rCFO0ewtdOPzC8-
PQ1uSIblJ3gM2EbN7b9VfHJssrog7UMWcT0GuiNM27AzwxYmEvkioeTmQaCzLNUGpyFnu1wg0e7mHzHMxPwSwMiOUFEE7SK
KFpyZT8ZLc8ZgEfZKMXK2FwCTpoktBr_h7u-2zpNK4x9Dwl2aqkMBNZL-QVNpaXXnA0K20PAVjK5-x7IkiELFFq_CSezg

stages:
  - build
  # - deploy

build-job:
  stage: build
  image: maven:latest
  tags:
    # - docker
    - kubernetes
  # before_script:
  # after_script:
  script:
```

```
    - mvn -T 1C -Dmaven.test.skip=true package
  artifacts:
    name: "$CI_PROJECT_NAME"
    paths:
      - target/*.jar
```

命名空间

```
[root@master ~]# kubectl create namespace gitlab
namespace/gitlab-runner created

[root@master ~]# kubectl create serviceaccount gitlab-runner -n gitlab
serviceaccount/secrets created

[root@master ~]# kubectl create token gitlab-runner -n gitlab

[root@master ~]# cat role.yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: gitlab-runner
  namespace: gitlab
rules:
  - apiGroups: ["*"]
    resources: ["pods"]
    verbs: ["list", "get", "watch", "create", "delete"]
  - apiGroups: ["*"]
    resources: ["pods/exec"]
    verbs: ["create"]
  - apiGroups: ["*"]
    resources: ["pods/log"]
    verbs: ["get"]
  - apiGroups: ["*"]
    resources: ["pods/attach"]
    verbs: ["list", "get", "create", "delete", "update"]
  - apiGroups: ["*"]
    resources: ["secrets"]
    verbs: ["list", "get", "create", "delete", "update"]
  - apiGroups: ["*"]
    resources: ["configmaps"]
    verbs: ["list", "get", "create", "delete", "update"]

[root@master ~]# kubectl apply -f role.yaml
role.rbac.authorization.k8s.io/gitlab created

[root@master ~]# kubectl create rolebinding gitlab-runner --namespace=gitlab --role=gitlab-
runner --serviceaccount=gitlab:gitlab-runner
```

```
kubectl create -n gitlab -f - <<EOF
apiVersion: v1
kind: Secret
metadata:
  name: gitlab-runner-token
  annotations:
    kubernetes.io/service-account.name: gitlab-runner
type: kubernetes.io/service-account-token
EOF
```

```
kubectl get secret gitlab-runner-token -o jsonpath='{.data.token}' -n gitlab | base64 -d
```

账号和Token创建完毕之后，使用下面命令检查

```
kubectl get namespace gitlab
kubectl get serviceaccounts gitlab-runner -n gitlab
kubectl get role gitlab-runner -n gitlab
kubectl get rolebinding gitlab-runner -n gitlab
kubectl get secrets gitlab-runner-token -n gitlab
kubectl get secret gitlab-runner-token -o jsonpath='{.data.token}' -n gitlab | base64 -d
```

挂载卷

创建 PVC

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    pv.kubernetes.io/bind-completed: "yes"
    pv.kubernetes.io/bound-by-controller: "yes"
    volume.beta.kubernetes.io/storage-provisioner: driver.longhorn.io
    volume.kubernetes.io/storage-provisioner: driver.longhorn.io
  creationTimestamp: "2022-10-19T12:46:33Z"
  finalizers:
  - kubernetes.io/pvc-protection
  managedFields:
  - apiVersion: v1
    fieldsType: FieldsV1
    fieldsV1:
      f:spec:
        f:accessModes: {}
        f:resources:
          f:requests:
            .: {}
            f:storage: {}
        f:storageClassName: {}
        f:volumeMode: {}
    manager: kube-explorer
    operation: Update
    time: "2022-10-19T12:46:33Z"
  - apiVersion: v1
    fieldsType: FieldsV1
    fieldsV1:
      f:metadata:
        f:annotations:
          .: {}
          f:pv.kubernetes.io/bind-completed: {}
          f:pv.kubernetes.io/bound-by-controller: {}
          f:volume.beta.kubernetes.io/storage-provisioner: {}
          f:volume.kubernetes.io/storage-provisioner: {}
      f:spec:
        f:volumeName: {}
    manager: k3s
    operation: Update
    time: "2022-10-19T12:46:35Z"
  - apiVersion: v1
```

```
    fieldsType: FieldsV1
    fieldsV1:
      f:status:
        f:accessModes: {}
        f:capacity:
          .: {}
          f:storage: {}
        f:phase: {}
    manager: k3s
    operation: Update
    subresource: status
    time: "2022-10-19T12:46:35Z"
  name: maven
  namespace: gitlab
  resourceVersion: "4846862"
  uid: ab0dc609-ff07-438e-b537-7e6182cec008
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
  storageClassName: longhorn
  volumeMode: Filesystem
  volumeName: pvc-ab0dc609-ff07-438e-b537-7e6182cec008
status:
  accessModes:
  - ReadWriteOnce
  capacity:
    storage: 10Gi
  phase: Bound

apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    pv.kubernetes.io/bind-completed: "yes"
    pv.kubernetes.io/bound-by-controller: "yes"
    volume.beta.kubernetes.io/storage-provisioner: driver.longhorn.io
    volume.kubernetes.io/storage-provisioner: driver.longhorn.io
  creationTimestamp: "2022-10-19T12:48:39Z"
  finalizers:
  - kubernetes.io/pvc-protection
  managedFields:
  - apiVersion: v1
    fieldsType: FieldsV1
    fieldsV1:
      f:spec:
        f:accessModes: {}
        f:resources:
          f:requests:
            .: {}
            f:storage: {}
        f:storageClassName: {}
        f:volumeMode: {}
    manager: kube-explorer
    operation: Update
    time: "2022-10-19T12:48:39Z"
  - apiVersion: v1
    fieldsType: FieldsV1
    fieldsV1:
      f:metadata:
        f:annotations:
          .: {}
          f:pv.kubernetes.io/bind-completed: {}
          f:pv.kubernetes.io/bound-by-controller: {}
          f:volume.beta.kubernetes.io/storage-provisioner: {}
```

```
          f:volume.kubernetes.io/storage-provisioner: {}
      f:spec:
        f:volumeName: {}
    manager: k3s
    operation: Update
    time: "2022-10-19T12:48:42Z"
  - apiVersion: v1
    fieldsType: FieldsV1
    fieldsV1:
      f:status:
        f:accessModes: {}
        f:capacity:
          .: {}
          f:storage: {}
        f:phase: {}
    manager: k3s
    operation: Update
    subresource: status
    time: "2022-10-19T12:48:42Z"
  name: builds
  namespace: gitlab
  resourceVersion: "4847301"
  uid: 09f9a7b0-558a-4142-b9b3-e1318aff223a
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
  storageClassName: longhorn-storage
  volumeMode: Filesystem
  volumeName: pvc-09f9a7b0-558a-4142-b9b3-e1318aff223a
```

配置 Gitlab Runner

```
[[runners]]
  name = "Kubernetes executor"
  url = "https://gitlab.netkiller.cn/"
  id = 3
  token = "5J6amB15rYWie_zGscFC"
  token_obtained_at = 2022-10-19T07:16:07Z
  token_expires_at = 0001-01-01T00:00:00Z
  executor = "kubernetes"
  [runners.custom_build_dir]
  [runners.cache]
    [runners.cache.s3]
    [runners.cache.gcs]
    [runners.cache.azure]
  [runners.kubernetes]
    host = "https://172.18.200.5:6443"
    #cert_file = "/etc/ssl/kubernetes/api.crt"
    #key_file = "/etc/ssl/kubernetes/api.key"
    ca_file = "/etc/ssl/kubernetes/ca.crt"
    tls_verify = true
    bearer_token =
"eyJhbGciOiJSUzI1NiIsImtpZCI6IktCOHRvYlZOLXFPRmEyb1JWdlQxSzBvN0tvZF9HNFBGRnlraDR5UU1jakkifQ.eyJ
hdWQiOlsiaHR0cHM6Ly9rdWJlcm5ldGVzLmRlZmF1bHQuc3ZjLmNsdXN0ZXIubG9jYWwiLCJrM3MiXSwiZXhwIjoxNjY2MT
g1NDc5LCJpYXQiOjE2NjYxODE4NzksImlzcyI6Imh0dHBzOi8va3ViZXJuZXRlcy5kZWZhdWx0LnN2Yy5jbHVzdGVyLmxvY
2FsIiwia3ViZXJuZXRlcy5pbyI6eyJuYW1lc3BhY2UiOiJnaXRsYWIiLCJzZXJ2aWNlYWNjb3VudCI6eyJuYW1lIjoiZ2l0
bGFiLXJ1bm5lciIsInVpZCI6IjdjYjhmNjM3LTc5YjQtNDc5Yi1hZjAzLTRhOGQ2ZDliM2YzOSJ9fSwibmJmIjoxNjY2MTg
xODc5LCJzdWIiOiJzeXN0ZW06c2VydmljZWFjY291bnQ6ZGl0bGFiOmdpdGxhYi1ydW5uZXIifQ.ExhowiFt9k-
vFwaxohs_KGF_JmzPIMnUyoOzENN9g250taWu3jwtRasixIckYhO5JavpzZRU1XTdhRmbEQ9hehlY5q6kXtGCZ4AImh-
q8gj7lglBLdWvZLMcmaqu3o4pIjb7Cw0qP25y-
```

```
F79IeDtUPZeLU2b_ocTIM5GPd5g6MxilTzw84GJ_hVevWVtdpTrRg48b64S6crAWpkUgMSVwyqwyjsTkGuEcn_V32fzO9ro
7EjumuWxVmOp5TA0mOXl4L8W1C6ZYVT_qGKsMJJ_AS3uZkKqcxT6FmSS0Ta4vjgQsVN1l6OjQNmrIWFBjRzNCiXPhlk5JzR
0ZID2J20UQQ"
    bearer_token_overwrite_allowed = true
    privileged = true
    image = ""
    namespace = "gitlab"
    namespace_overwrite_allowed = ""
    pod_labels_overwrite_allowed = ""
    service_account = "gitlab-runner"
    service_account_overwrite_allowed = ""
    pod_annotations_overwrite_allowed = ""
    [runners.kubernetes.affinity]
    [runners.kubernetes.pod_security_context]
    [runners.kubernetes.init_permissions_container_security_context]
      [runners.kubernetes.init_permissions_container_security_context.capabilities]
    [runners.kubernetes.build_container_security_context]
      [runners.kubernetes.build_container_security_context.capabilities]
    [runners.kubernetes.helper_container_security_context]
      [runners.kubernetes.helper_container_security_context.capabilities]
    [runners.kubernetes.service_container_security_context]
      [runners.kubernetes.service_container_security_context.capabilities]
    [runners.kubernetes.volumes]
    [[runners.kubernetes.volumes.host_path]]
      name = "cache"
      mount_path = "/cache"
      host_path = "/tmp/cache"
    [[runners.kubernetes.volumes.pvc]]
      name = "maven"
      mount_path = "/root/.m2"
    [[runners.kubernetes.volumes.pvc]]
      name = "builds"
      mount_path = "/builds"
    [runners.kubernetes.dns_config]
    [runners.kubernetes.container_lifecycle]
```

**KUBERNETES_BEARER_TOKEN**

/etc/gitlab-runner/config.toml

```
bearer_token_overwrite_allowed = false
改为
bearer_token_overwrite_allowed = true
```

.gitlab-ci.yml

```
variables:
  KUBERNETES_SERVICE_ACCOUNT: gitlab
  KUBERNETES_BEARER_TOKEN:
eyJhbGciOiJSUzI1NiIsImtpZCI6IktCOHRvYlZOLXFPRmEyb1JWdlQxSzBvN0tvZF9HNFBGRnlraDR5UU1jakkifQ.eyJh
dWQiOlsiaHR0cHM6Ly9rdWJlcm5ldGVzLmRlZmF1bHQuc3ZjLmNsdXN0ZXIubG9jYWwiLCJrM3MiXSwiZXhwIjoxNjY2MTg
1MTE3LCJpYXQiOjE2NjYxODE1MTcsImlzcyI6Imh0dHBzOi8va3ViZXJuZXRlcy5kZWZhdWx0LnN2Yy5jbHVzdGVyLmxvY2
FsIiwia3ViZXJuZXRlcy5pbyI6eyJuYW1lc3BhY2UiOiJnaXRsYWItcnVubmVyIiwic2VydmljZWFjY291bnQiOnsibmFtZ
SI6ImdpdGxhYiIsInVpZCI6Ijc3MGE4ODk1LTcwNjAtNGUwYi04MTc0LTVkNmMzNDUzYTQxOSJ9fSwibmJmIjoxNjY2MTgx
NTE3LCJzdWIiOiJzeXN0ZW06c2VydmljZWFjY291bnQ6bGFiLXJ1bm5lcjpnaXRsYWIifQ.PO96hsJSG7-
h2EJhDoUc6xWCPzDp9AwoZ1-CrsM-GUN73ft7ge9prdarq0of-rzgAP-wn-07r6vIxdcT2G0ZHl-
GsElyxaAm5Poz1WX2bQLuLvcpoFVQyGFH1Mkkno5MDCfi2CPfNYUuEBl5vQ_jDooE5dUnildLmbEv7ooxLYeWAPnEd5HmOy
```

```
XKjC2FiBqKQ88oxkDbmq3Hwbxm-
XmTma7T3NqXxST_m7qE6tb2n0RsG3o1lJGEDftdf9bF1eEfAykaa077tzHNNZtQvK87LK69XZVUbOVun_G98-
rLL7afSridgP6mhia6CPeful7xvedJ8l4g8V-Ku8qixKb5Yg
```

案例

```
[root@agent-5 ~]# cat /etc/gitlab-runner/config.toml
concurrent = 5
check_interval = 0

[session_server]
  session_timeout = 1800

[[runners]]
  name = "Kubernetes local cluster"
  url = "https://gitlab.netkiller.cn/"
  id = 41
  token = "y1QnvNhSwMYVX2-z3x4E"
  token_obtained_at = 2022-10-20T00:19:08Z
  token_expires_at = 0001-01-01T00:00:00Z
  executor = "kubernetes"
  [runners.custom_build_dir]
  [runners.cache]
    [runners.cache.s3]
    [runners.cache.gcs]
    [runners.cache.azure]
  [runners.kubernetes]
    host = "https://172.18.200.5:6443"
    ca_file = "/etc/ssl/kubernetes/ca.crt"
    bearer_token_overwrite_allowed = true
    bearer_token =
"eyJhbGciOiJSUzI1NiIsImtpZCI6IktCOHRvYlZOLXFPRmEyb1JWdlQxSzBvN0tvZF9HNFBGRnlraDR5UU1jakkifQ.eyJ
hdWQiOlsiaHR0cHM6Ly9rdWJlcm5ldGVzLmRlZmF1bHQuc3ZjLmNsdXN0ZXIubG9jYWwiLCJrM3MiXSwiZXhwIjoxNjY2Mj
MwNjc5LCJpYXQiOjE2NjYyMjcwNzksImlzcyI6Imh0dHBzOi8va3ViZXJuZXRlcy5kZWZhdWx0LnN2Yy5jbHVzdGVyLmxvY
2FsIiwia3ViZXJuZXRlcy5pbyI6eyJuYW1lc3BhY2UiOiJnaXRsYWIiLCJzZXJ2aWNlYWNjb3VudCI6eyJuYW1lIjoiZ2l0
bGFiLXJ1bm5lciIsInVpZCI6IjdjYjhmNjM3LTc5YjQtNDc5Yi1hZjAzLTRhOGQ2ZDliM2YzOSJ9fSwibmJmIjoxNjY2MjI
3MDc5LCJzdWIiOiJzeXN0ZW06c2VydmljZWFjY291bnQ6ZGl0bGFiOmdpdGxhYi1ydW5uZXIifQ.e6RvBIUCUlZ4ciVKRQ-
i2kEBBTXaPq876L-
EzA5NGNWxXmfkuyx_IpGrrzmTMPV9prR3XLQfmyT1OoEgkO8riMLtJL1yPZNiaTo0zhAfuWvvXP3gOnJsFRdcx2PsvA1dxu
uhtXNp1Y2BAboU8jV10-OK6WU40i3CKb9OTQpN-BRAlwXylQ-
55ZIRT7J3ghpCLfM6BplwZOYfCP6XbHJfgAKD2wy5sl8Ni1XIpsLUOp20TlBaG22k2admcWiRavxyjp68EHMwq3izI5_4qU
9hFcYBOsUY-PBC27nGw9ICH0sIRuq4Xs0GQ8oXOzS5Dbja5uKKVq6-bWV3onV13AjcxQ"
    image = ""
    namespace = "gitlab"
    namespace_overwrite_allowed = ""
    pull_policy = ["if-not-present"]
    image_pull_secrets = ["registry"]
    pod_labels_overwrite_allowed = ""
    service_account = "gitlab-runner"
    service_account_overwrite_allowed = ""
    pod_annotations_overwrite_allowed = ""
    [runners.kubernetes.affinity]
    [runners.kubernetes.pod_security_context]
    [runners.kubernetes.init_permissions_container_security_context]
      [runners.kubernetes.init_permissions_container_security_context.capabilities]
    [runners.kubernetes.build_container_security_context]
      [runners.kubernetes.build_container_security_context.capabilities]
    [runners.kubernetes.helper_container_security_context]
      [runners.kubernetes.helper_container_security_context.capabilities]
    [runners.kubernetes.service_container_security_context]
      [runners.kubernetes.service_container_security_context.capabilities]
    [runners.kubernetes.volumes]
```

```
      [[runners.kubernetes.volumes.host_path]]
        name = "docker"
        mount_path = "/var/run/docker.sock"
        host_path = "/var/run/docker.sock"

      [[runners.kubernetes.volumes.pvc]]
        name = "maven"
        mount_path = "/root/.m2"

      [[runners.kubernetes.volumes.pvc]]
        name = "builds"
        mount_path = "/builds"

      [[runners.kubernetes.volumes.pvc]]
        name = "cache"
        mount_path = "/cache"
    [runners.kubernetes.dns_config]
    [runners.kubernetes.container_lifecycle]

[[runners]]
  name = "Docker"
  url = "https://gitlab.netkiller.cn/"
  id = 42
  token = "Y7Df44aY8YrRwASUXWE5"
  token_obtained_at = 2022-10-20T02:40:25Z
  token_expires_at = 0001-01-01T00:00:00Z
  executor = "docker"
  [runners.custom_build_dir]
  [runners.cache]
    [runners.cache.s3]
    [runners.cache.gcs]
    [runners.cache.azure]
  [runners.docker]
    tls_verify = false
    image = "docker:latest"
    privileged = false
    disable_entrypoint_overwrite = false
    oom_kill_disable = false
    disable_cache = false
    volumes = ["/cache"]
    shm_size = 0

[[runners]]
  name = "Shell"
  url = "https://gitlab.netkiller.cn/"
  id = 43
  token = "s2tuoKTrj1s1iv_mfYh5"
  token_obtained_at = 2022-10-20T03:18:08Z
  token_expires_at = 0001-01-01T00:00:00Z
  executor = "shell"
  [runners.custom_build_dir]
  [runners.cache]
    [runners.cache.s3]
    [runners.cache.gcs]
    [runners.cache.azure]
```

.gitlab-ci.yml

```
stages:
  - build
  - docker
  - deploy
```

```
variables:
  DOCKER_REGISTRY: registry.netkiller.cn
  IMAGE:
$DOCKER_REGISTRY/$CI_COMMIT_BRANCH/$CI_PROJECT_NAME:$CI_COMMIT_SHORT_SHA-$CI_PIPELINE_ID
  KUBERNETES_BEARER_TOKEN:
eyJhbGciOiJSUzI1NiIsImtpZCI6IktCOHRvYlZOLXFPRmEyb1JWdlQxSzBvN0tvZF9HNFBGRnlraDR5UU1jakkifQ.eyJp
c3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW50Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1lc3BhY2U
iOiJnaXRsYWIiLCJrdWJlcm5ldGVzLmlvL3NlcnZpY2VhY2NvdW50L3NlY3JldC5uYW1lIjoiZ2l0bGFiLXJ1bm5lci10b2
tlbiIsImt1YmVybmV0ZXMuaW8vc2VydmljZWFjY291bnQvc2VydmljZS1hY2NvdW50Lm5hbWUiOiJnaXRsYWItcnVubmVyI
iwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9zZXJ2aWNlLWFjY291bnQudWlkIjoiN2NiOGY2MzctNzliNC00Nzli
LWFmMDMtNGE4ZDZkOWIzZjM5Iiwic3ViIjoic3lzdGVtOnNlcnZpY2VhY2NvdW50OmdpdGxhYjpnaXRsYWItcnVubmVyIn0
.pU4-
8D4szeL8iud1SvesdN7nV7L3GLaNsa2UbsxkGQ4SDGN85zKTXJl6MtqDsuJB9HBUlOTMnyEa0gCbgHOJlR3fd2HcegitrRL
eybvUuotniiLpCPO7vAO-oS5Fej7oUFBXqZJYIx-
xMbFoyt3rnGs273c_yE8avI8EGdEPNhOWRgF_GZBYstvwiEjO2IUDWbutzCTtGloPvJ5Ur0s7drLJkCQvT2nod5tSSnY5R0
lpNyD2FodkFR28KU1EgFoHUnH_ERtUAS5qObIETWSwm5SmCnd2Ogjh70DDxmIHSU-
saFU0zSqPpZ1oX9hgO9YMkcJXPHOEnqIVEagZ5CSf2w

cache:
  key: ${CI_COMMIT_REF_SLUG}
  paths:
    - target/

build-job:
  stage: build
  image: registry.netkiller.cn/common/maven:latest
  script:
    - mvn clean package -Dautoconfig.skip=true -Dmaven.test.skip=true -
Dmaven.test.failure.ignore=true
  after_script:
    - md5sum target/*.jar
  only:
    - dev
    - test
  tags:
    - kubernetes
  artifacts:
    name: "$CI_PROJECT_NAME"
    paths:
      - target/*.jar

build-docker:
  stage: docker
  image: docker:latest
  before_script:
    - echo "$CI_REGISTRY_PASSWORD" | docker login $DOCKER_REGISTRY --username $CI_REGISTRY_USER
--password-stdin
  after_script:
    - docker images | grep $CI_PROJECT_NAME
  script:
    - docker build -t $IMAGE -f Dockerfile .
    - docker push $IMAGE
  only:
    - dev
    - test
  tags:
    - kubernetes

deploy-job:
  stage: deploy
  variables:
    GIT_STRATEGY: none
  before_script:
    - kubectl -n test get pod | grep $CI_PROJECT_NAME
  script:
    - kubectl set image deployment/${CI_PROJECT_NAME} ${CI_PROJECT_NAME}=${IMAGE} -n
```

```
${CI_COMMIT_BRANCH}
  after_script:
    - kubectl -n test get pod | grep $CI_PROJECT_NAME
  only:
    - dev
    - test
  tags:
    - shell
  environment:
    name: $CI_COMMIT_BRANCH
    url: $CI_COMMIT_BRANCH.netkiller.cn/api/monitor/health
```

## 5.7. Java 持续集成相关

### 制作 Maven 镜像

自建 Maven 仓库，需要配置 settings.xml，这时就需要制作一个 Maven 镜像，同事有 COPY 命令把 settings.xml 文件复制到 /usr/share/maven/conf/settings.xml 目录

```
[root@netkiller jdk11]# ls
Dockerfile  build.sh  settings.xml

[root@netkiller jdk11]# cat Dockerfile
FROM maven:3.8.6-openjdk-11

COPY settings.xml /root/.m2/settings.xml
COPY settings.xml /usr/share/maven/conf/settings.xml

[root@netkiller jdk11]# cat build.sh
docker build -t "registry.netkiller.cn/share/maven:3.8.6-openjdk-11" .
docker push registry.netkiller.cn/share/maven:3.8.6-openjdk-11
```

### JaCoCo

JaCoCo Java Code Coverage Library https://www.jacoco.org/jacoco/index.html

pom.xml 中必须有单元测试依赖

```
            <dependency>
                    <groupId>org.springframework.boot</groupId>
                    <artifactId>spring-boot-starter-test</artifactId>
                    <scope>test</scope>
            </dependency>
            <dependency>
                    <groupId>junit</groupId>
                    <artifactId>junit</artifactId>
                    <scope>test</scope>
            </dependency>
```

不能跳过单元测试

```
```

```
                        <plugin>
                                <artifactId>maven-surefire-plugin</artifactId>
                                <configuration>
                                        <skip>false</skip>
                                </configuration>
                        </plugin>
```

添加 JaCoCo 插件

```
                        <plugin>
                                <groupId>org.jacoco</groupId>
                                <artifactId>jacoco-maven-plugin</artifactId>
                                <executions>
                                        <execution>
                                                <goals>
                                                        <goal>prepare-agent</goal>
                                                </goals>
                                        </execution>
                                        <execution>
                                                <id>report</id>
                                                <phase>test</phase>
                                                <goals>
                                                        <goal>report</goal>
                                                </goals>
                                        </execution>
                                </executions>
                        </plugin>
```

最后运行 mvn test 调试一下，输入类似下面

```
[INFO] ------------------------< cn.netkiller:config >-------------------------
[INFO] Building config 0.0.1-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- jacoco-maven-plugin:0.8.7:prepare-agent (default) @ config ---
[INFO] argLine set to -
javaagent:/Users/neo/.m2/repository/org/jacoco/org.jacoco.agent/0.8.7/org.jacoco.agent-0.8.7-
runtime.jar=destfile=/Users/neo/workspace/microservice/config/target/jacoco.exec
[INFO]
[INFO] --- maven-resources-plugin:3.2.0:resources (default-resources) @ config ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Using 'UTF-8' encoding to copy filtered properties files.
[INFO] Copying 1 resource
[INFO] Copying 6 resources
[INFO]
[INFO] --- maven-compiler-plugin:3.8.1:compile (default-compile) @ config ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-resources-plugin:3.2.0:testResources (default-testResources) @ config ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Using 'UTF-8' encoding to copy filtered properties files.
[INFO] Copying 1 resource
[INFO]
[INFO] --- maven-compiler-plugin:3.8.1:testCompile (default-testCompile) @ config ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-surefire-plugin:2.22.2:test (default-test) @ config ---
[INFO]
```

```
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 0, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO]
[INFO] --- jacoco-maven-plugin:0.8.7:report (report) @ config ---
[INFO] Loading execution data file /Users/neo/workspace/microservice/config/target/jacoco.exec
[INFO] Analyzed bundle 'config' with 1 classes
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  3.335 s
[INFO] Finished at: 2021-10-22T15:52:36+08:00
[INFO] ------------------------------------------------------------------------
```

配置持续集成流水线 .gitlab-ci.yml 文件

```
cache:
  untracked: true

stages:
  - build
  - test
  - deploy

test-job:
  stage: test
  variables:
    GIT_STRATEGY: none
  only:
    - tags
    - development
    - testing
  script:
    - mvn test
  after_script:
    - lrsync 'zito-admin/target/site/*'
www@report.netkiller.cn:/opt/netkiller.cn/report.netkiller.cn
    - wechat -t 1 代码覆盖率报告 http://report.netkiller.cn/jacoco/index.html
```

## 并行开发解决版本冲突

Maven 项目 parent 文件

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>cn.netkiller</groupId>
    <artifactId>test</artifactId>
    <version>0.0.1${project.branch}${project.phase}</version>
    <packaging>pom</packaging>
```

```
    <name>test</name>
    <url>http://maven.apache.org</url>
    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <maven.compiler.source>18</maven.compiler.source>
        <maven.compiler.target>${maven.compiler.source}</maven.compiler.target>
        <selenium.version>4.8.1</selenium.version>
        <project.phase>-SNAPSHOT</project.phase>
        <project.branch></project.branch>
    </properties>
    <distributionManagement>
        <repository>
            <id>repository</id>
            <name>Release repository</name>
            <url>https://maven.netkiller.cn/repository/repository/</url>
        </repository>
        <snapshotRepository>
            <id>snapshots</id>
            <name>Snapshots repository</name>
            <url>https://maven.netkiller.cn/repository/snapshots/</url>
        </snapshotRepository>

    </distributionManagement>
    <modules>
        <module>common</module>
        <module>selenium</module>
    </modules>

</project>
```

模块

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <packaging>jar</packaging>
    <parent>
        <groupId>cn.netkiller</groupId>
        <artifactId>test</artifactId>
        <version>0.0.1-SNAPSHOT</version>
    </parent>

    <artifactId>common</artifactId>
    <!--    <version>${project.parent.version}${project.branch}</version>-->

    <properties>
        <maven.compiler.source>18</maven.compiler.source>
        <maven.compiler.target>18</maven.compiler.target>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    </properties>

</project>
```

构建快照版本 common-0.0.1-SNAPSHOT.jar

```
```

```
neo@macbook-pro-neo ~/w/test> mvn clean package deploy

[INFO] ------------------------------------------------------------------------
[INFO] Reactor Build Order:
[INFO]
[INFO] test                                                               [pom]
[INFO] common                                                             [jar]
[INFO] selenium                                                           [jar]
[INFO]
[INFO] -------------------------< cn.netkiller:test >--------------------------
[INFO] Building test 0.0.1-SNAPSHOT                                       [1/3]
[INFO] --------------------------------[ pom ]---------------------------------
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ test ---
[INFO]
[INFO] ------------------------< cn.netkiller:common >-------------------------
[INFO] Building common 0.0.1-SNAPSHOT                                     [2/3]
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ common ---
[INFO] Deleting /Users/neo/workspace/test/common/target
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ common ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 0 resource
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ common ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 1 source file to /Users/neo/workspace/test/common/target/classes
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ common ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /Users/neo/workspace/test/common/src/test/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ common ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ common ---
[INFO] No tests to run.
[INFO]
[INFO] --- maven-jar-plugin:2.4:jar (default-jar) @ common ---
[INFO] Building jar: /Users/neo/workspace/test/common/target/common-0.0.1-SNAPSHOT.jar
[INFO]
[INFO] ------------------------< cn.netkiller:selenium >-----------------------
[INFO] Building selenium 0.0.1-SNAPSHOT                                   [3/3]
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ selenium ---
[INFO] Deleting /Users/neo/workspace/test/selenium/target
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ selenium ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory
/Users/neo/workspace/test/selenium/src/main/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.8.1:compile (default-compile) @ selenium ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 8 source files to /Users/neo/workspace/test/selenium/target/classes
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ selenium ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory
/Users/neo/workspace/test/selenium/src/test/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.8.1:testCompile (default-testCompile) @ selenium ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 2 source files to /Users/neo/workspace/test/selenium/target/test-classes
```

```
[INFO]
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ selenium ---
[INFO]
[INFO] --- maven-jar-plugin:3.1.1:jar (default-jar) @ selenium ---
[INFO] Building jar: /Users/neo/workspace/test/selenium/target/selenium-0.0.1-SNAPSHOT.jar
[INFO] ------------------------------------------------------------------------
[INFO] Reactor Summary for test 0.0.1-SNAPSHOT:
[INFO]
[INFO] test ............................................... SUCCESS [  0.125 s]
[INFO] common ............................................. SUCCESS [  1.422 s]
[INFO] selenium ........................................... SUCCESS [  1.457 s]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  3.158 s
[INFO] Finished at: 2023-04-05T18:32:45+08:00
[INFO] ------------------------------------------------------------------------
```

构建开发环境快照版本 common-0.0.1-dev-SNAPSHOT.jar

```
neo@macbook-pro-neo ~/w/t/common> mvn clean package -Dproject.branch=-dev
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------< cn.netkiller:common >-------------------------
[INFO] Building common 0.0.1-dev-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ common ---
[INFO] Deleting /Users/neo/workspace/test/common/target
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ common ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 0 resource
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ common ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 1 source file to /Users/neo/workspace/test/common/target/classes
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ common ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /Users/neo/workspace/test/common/src/test/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ common ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ common ---
[INFO] No tests to run.
[INFO]
[INFO] --- maven-jar-plugin:2.4:jar (default-jar) @ common ---
[INFO] Building jar: /Users/neo/workspace/test/common/target/common-0.0.1-dev-SNAPSHOT.jar
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  1.384 s
[INFO] Finished at: 2023-04-05T18:34:50+08:00
[INFO] ------------------------------------------------------------------------
```

构建RELEASE版本 common-0.0.1.jar

```
neo@macbook-pro-neo ~/w/t/common> mvn clean package -Dproject.phase=

[INFO] ------------------------------------------------------------------------
[INFO] Reactor Summary for test 0.0.1:
[INFO]
[INFO] common ............................................. SUCCESS [  6.828 s]
[INFO] selenium ........................................... SUCCESS [  6.351 s]
[INFO] test ............................................... SUCCESS [  3.274 s]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  16.631 s
[INFO] Finished at: 2023-04-05T16:16:28+08:00
[INFO] ------------------------------------------------------------------------
```

构建带有 “-RELEASE” 后缀的版本 common-0.0.1-RELEASE.jar

```
neo@macbook-pro-neo ~/w/t/common> mvn clean package -Dproject.phase=-RELEASE
[INFO] Scanning for projects...
[INFO]
[INFO] -----------------------< cn.netkiller:common >------------------------
[INFO] Building common 0.0.1-RELEASE
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ common ---
[INFO] Deleting /Users/neo/workspace/test/common/target
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ common ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 0 resource
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ common ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 1 source file to /Users/neo/workspace/test/common/target/classes
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ common ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /Users/neo/workspace/test/common/src/test/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ common ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ common ---
[INFO] No tests to run.
[INFO]
[INFO] --- maven-jar-plugin:2.4:jar (default-jar) @ common ---
[INFO] Building jar: /Users/neo/workspace/test/common/target/common-0.0.1-RELEASE.jar
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  1.656 s
[INFO] Finished at: 2023-04-05T18:36:30+08:00
[INFO] ------------------------------------------------------------------------
```

## 5.8. 数据库结构监控

什么是数据库结构版本控制

首先说说什么是数据库结构，什么事版本控制。

数据库结构是指数据库表结构，数据库定义语言导出的DDL语句。主要由CREATE TABLE, DROP TABLE等等构成。

再来说说什么事版本控制，如果你从事开发工作应该会很容易理解，版本控制就是记录每一次变化，可以随时查看历史记录，并可回撤到指定版本。

### 为什么要做数据库结构本版控制

软件开发过程中需要常常对数据库结构作调整，这是无法避免的，甚至很多想起启动后，需求还不明确，开发人员只能按照所理解需求创建表。需求往往会发生变化，一旦变化，代码需要修改，表结构也避免不了。 我们常常刚改好数据库结构，需求部门有发来通知，不用修改了，维持原有设计。甚至是过了几周再次回撤。

所以我们要将数据库结构的变化进行版本控制，通常的做法是DBA人工管理，但我觉完全可以自动化的工作，没有必要浪费人力资源，且自动化不会犯错更稳定，仅仅需要人工定期查看工作状态即可。

### 何时做数据库结构本版控制

任何时候都可以部署下面的脚本，对现有系统无任何影响。

### 在哪里做数据库结构本版控制

可以在版本控制服务器上，建议GIT仓库push到远程。

### 谁来负责数据库结构本版控制

DBA与配置管理员都可以做，通常DBA不接触版本库这块，建议创建一个backup用户给配置管理员。

### 怎样做数据库结构本版控制

#### 安装脚本

首先下载脚本 https://github.com/oscm/shell/blob/master/backup/backup.mysql.struct.sh

```
wget https://raw.githubusercontent.com/oscm/shell/master/backup/backup.mysql.struct.sh
mv backup.mysql.struct.sh /usr/local/bin
chmod +x /usr/local/bin/backup.mysql.struct
```

创建备份用户

```
CREATE USER 'backup'@'localhost' IDENTIFIED BY 'chen';
GRANT SELECT, LOCK TABLES  ON *.* TO 'backup'@'localhost';
FLUSH PRIVILEGES;
SHOW GRANTS FOR 'backup'@'localhost';
```

配置脚本

```
BACKUP_HOST="localhost"                 数据库主机
BACKUP_USER="backup"                    备份用户
BACKUP_PASS="chen"                              备份密码
```

```
BACKUP_DBNAME="neo netkiller"    版本控制那些数据库，多个数据库使用空格分隔
BACKUP_DIR=~/backup                                 数据库结构放在那里
GIT=git@gitlab.netkiller.cn:netkiller.cn/db.netkiller.cn.git
```

初始化仓库

```
# /usr/local/bin/backup.mysql.struct init
Initialized empty Git repository in /www/database/struct/.git/
```

启动脚本，停止脚本

```
# /usr/local/bin/backup.mysql.struct
Usage: /usr/local/bin/backup.mysql.struct {init|start|stop|status|restart}
```

开始脚本

```
# /usr/local/bin/backup.mysql.struct start
```

查看状态

```
# /usr/local/bin/backup.mysql.struct status
 9644 pts/1    S      0:00 /bin/bash /usr/local/bin/backup.mysql.struct start
```

停止脚本

```
# /usr/local/bin/backup.mysql.struct status
```

查看历史版本

通过 git log 命令查看历史版本

```
# cd /www/database/struct/
# git status
# On branch master
nothing to commit (working directory clean)
# git log
commit d38fc624c21cad0e2f55f0228bff0c1be981827c
Author: root <root@slave.example.com>
Date:   Wed Dec 17 12:33:55 2014 +0800
    2014-12-17.04:33:55
```

这里仅仅将数据库结构版本控制，关于版本控制软件更多细节，延伸阅读 《Netkiller Version 手札》

**CI/CD 配置**

```
```

```
stages:
  - watch
  - backup

build-job:
  stage: watch
  script:
    - wechat -t 10 数据库结构变更通知
"http://gitlab.netkiller.cn/netkiller.cn/db.netkiller.cn/-/commit/${CI_COMMIT_SHA}"
    - wechat -t 10 "$(git diff HEAD^)"

deploy-job:
  stage: backup
  script:
    - sqldump development
```

## 5.9. 持续部署 Nacos

背景，微服务开发中，常常会用到注册中心和配置中心，目前国内比较流行使用 Nacos，Nacos 的配置是保存在数据库中的，不方便维护。代码的变更与配置的版本是没有强关联的，尤其是并行开发中，我们需要多套开发和测试环境时，配置管理的工作会带来挑战，如果配置中心管理不善，就会出现各种问题，例如相互覆盖，版本不一致等等。

虽然 Nacos 也有历史记录，需要为每个人创建一个帐号，才能实现在版本管理中看到谁在什么时间修改了配置。但是即使这样，配置的版本与代码是没有关联的，我们不清楚不同的代码版本需要那些必要的配置项。

我更趋向让配置与代码版本强关联，实现配置的版本与代码的版本一致，让配置管理与持续集成和部署融合，配置变更由持续部署流水线自行完成，而不是让管理员去 Nacos 后台手工处理。于是变产生了下面的工具。

**nacos 持续部署工具**

安装 netkiller-devops 工具

```
# pip 命令安装:
root@netkiller ~# pip install netkiller-devops

# 如果此前已经安装，可以使用下面命令更新:
root@netkiller ~# pip install netkiller-devops --upgrade

# Docker 方式安装:
root@netkiller ~# docker pull netkiller/netkiller-devops:latest
root@netkiller ~# docker run --rm -it --name=netkiller --entrypoint=sh netkiller-devops:latest
/srv # nacos
Usage: nacos [options] message

Options:
  -h, --help            show this help message and exit
  -s localhost:8848, --server-addr=localhost:8848
                        localhost:8848
  -u USERNAME, --username=USERNAME
  -p PASSWORD, --password=PASSWORD
  -n public, --namespace=public
  -d DATAID, --dataId=DATAID
  -g DEFAULT_GROUP, --group=DEFAULT_GROUP

  Config:
```

```
    --push
    --show
    --save
    -f FILE, --file=FILE
                        .yaml file
    -t yaml, --type=yaml
                        yaml|text|json|xml|Properties
    --delete

Homepage: https://www.netkiller.cn      Author: Neo <netkiller@msn.com>
Help: https://github.com/netkiller/devops/blob/master/doc/
```

帮助信息

```
root@netkiller ~# nacos
Usage: nacos [options]

Options:
  -h, --help            show this help message and exit
  -s http://localhost:8848, --server-addr=http://localhost:8848
                        Nacos 服务器地址
  -u USERNAME, --username=USERNAME
                        用户名
  -p PASSWORD, --password=PASSWORD
                        密码
  -n public, --namespace=public
                        命名空间
  -d DATAID, --dataId=DATAID
                        配置ID
  -g DEFAULT_GROUP, --group=DEFAULT_GROUP
                        分组

  配置管理:
    --push              发布配置
    --show              查看配置
    --save              保存配置
    -f FILE, --file=FILE
                        .yaml 文件
    -t yaml, --type=yaml
                        yaml|text|json|xml|Properties
    --delete            删除配置

Homepage: https://www.netkiller.cn      Author: Neo <netkiller@msn.com>
Help: https://github.com/netkiller/devops/blob/master/doc/
```

发布 Nacos 配置

```
准备配置文件

root@netkiller ~# cat test.yaml
server:
  servlet:
    context-path: /netkiller
spring:
  servlet:
    multipart:
      max-file-size: 10MB
      max-request-size: 10MB
```

```
发布 test.yaml 配置文件到 Nacos 配置中心

root@netkiller ~# nacos -s http://nacos.netkiller.cn -u nacos -p nacos -n test -d test --push -
f test.yaml
```

查看 Nacos 配置

```
root@netkiller ~# nacos -s http://nacos.netkiller.cn -u nacos -p nacos -n test -d test --show
server:
  servlet:
    context-path: /netkiller
spring:
  servlet:
    multipart:
      max-file-size: 10MB
      max-request-size: 10MB
```

保存 Nacos 配置

```
root@netkiller ~# nacos -s http://nacos.netkiller.cn -u nacos -p nacos -n test -d test --save -
f save.yaml
root@netkiller ~# cat save.yaml
server:
  servlet:
    context-path: /netkiller
spring:
  servlet:
    multipart:
      max-file-size: 10MB
      max-request-size: 10MB
```

删除 Nacos 配置

```
root@netkiller ~# nacos -s http://nacos.netkiller.cn -u nacos -p nacos -n test -d test —delete

返回 None 表示配置不存在
root@netkiller ~# nacos -s http://nacos.netkiller.cn -u nacos -p nacos -n test -d test --show
None
```

**.gitlab-ci.yml 配置案例**

方案一，使用 Shell 执行器，在 gitlab runner 节点上安装 netkiller-devops 包

```
stages:
  - deploy
```

```
deploy-job:
  stage: deploy
  variables:
    NACOS: http://nacos.netkiller.cn
  before_script:
    - nacos -s $NACOS -u nacos -p nacos -n $CI_COMMIT_BRANCH -d $CI_PROJECT_NAME --show
  script:
    - nacos -s $NACOS -u nacos -p nacos -n $CI_COMMIT_BRANCH -d $CI_PROJECT_NAME --push -f
nacos/$CI_COMMIT_BRANCH.yaml
  after_script:
    - nacos -s $NACOS -u nacos -p nacos -n $CI_COMMIT_BRANCH -d $CI_PROJECT_NAME --show
  only:
    - dev
    - test
    - master
  tags:
    - shell
```

方案二、使用 Kubernetes 或者 Docker 执行器

```
deploy-job-kubernetes:
  stage: deploy
  image: netkiller/netkiller-devops:latest
  variables:
    NACOS: http://nacos.netkiller.cn:8848
  before_script:
    - cat nacos/$CI_COMMIT_BRANCH.yaml
  script:
    - nacos -s $NACOS -u nacos -p nacos -n $CI_COMMIT_BRANCH -d $CI_PROJECT_NAME --push -f
nacos/$CI_COMMIT_BRANCH.yaml
  after_script:
    - nacos -s $NACOS -u nacos -p nacos -n $CI_COMMIT_BRANCH -d $CI_PROJECT_NAME --show
  only:
    - dev
    - test
    - master
  tags:
    - kubernetes
```

# 6. Pipeline 流水线

## 6.1. cache

Java 缓存设置

```
image: maven:3.5.0-jdk-8

variables:
  MAVEN_OPTS: "-Dmaven.repo.local=.m2/repository"

cache:
  paths:
    - .m2/repository/
    - target/

stages:
  - build
  - test
  - package

build:
  stage: build
  script: mvn compile

unittest:
  stage: test
  script: mvn test


package:
  stage: package
  script: mvn package
  artifacts:
    paths:
      - target/java-project-0.0.1-SNAPSHOT.jar
```

Node 缓存设置

```
cache:
  paths:
    - node_modules
    - dist

# variables:
  # GIT_STRATEGY: clone
  # GIT_STRATEGY: fetch
  # GIT_CHECKOUT: "false"

stages:
  - build
  - test
  - deploy

build-job:
  stage: build
```

```
  only:
    - master
    - testing
    - development
  script:
    - echo "Compiling the code..."
    # - cnpm cache verify
    - cnpm install
    - cnpm run build:stage
    # - cnpm run build:prod
    - echo "Compile complete."

test-job:
  stage: test
  variables:
    GIT_STRATEGY: none
  only:
    - master
    - testing
    - development
  script:
    - echo "Running unit tests..."
    - sed -i 's#192.168.20.180#192.168.30.4#g' dist/umi.*.js
    - ls dist/*
    # - rm -rf *.tar.gz
    # - tar zcvf www.netkiller.cn.$(date -u +%Y-%m-%d.%H%M%S).tar.gz dist
    # - ls *.tar.gz
    - echo "Test complete."
  artifacts:
    name: "$CI_PROJECT_NAME"
    paths:
      - dist/*
      # - ./*.tar.gz


deploy-test-job:
  stage: deploy
  variables:
    GIT_STRATEGY: none
  only:
    - testing
    - development
  script:
    - echo "Deploying application..."
    - rsync -auzv dist/* www@192.168.30.10:/opt/www.netkiller.cn/
    - echo "Application successfully deployed."

deploy-prod-job:
  stage: deploy
  only:
    - master
  script:
    - echo "Deploying application..."
    - rsync -auzv --delete dist/* www@192.168.30.10:/opt/www.netkiller.cn/
    - echo "Application successfully deployed."
```

**Cache Key**

缓存在所有流水线间是共享的，如果同时有两个JOB在跑，缓存就可能受到影响，这时可以使用 cache key 解决。

```
```

```
对每个分支的每个 job 使用不同的 cache :

cache:
  key: ${CI_COMMIT_REF_SLUG}

每个分支的每个 job 使用不同的 stage:
cache:
  key: "$CI_JOB_NAME-$CI_COMMIT_REF_SLUG"

分支之间需要共享 cache，但是 pipeline 中的 job 之间的 cache 是相互独立的:
cache:
  key: "$CI_JOB_STAGE-$CI_COMMIT_REF_SLUG"

缓存只在相同 CI_PIPELINE_ID 中共享
cache:
  key: ${CI_PIPELINE_ID}
```

**禁用 Cache**

当定义了全局 cahce 后，想在 job 中禁用 Cache

```
cache:
  paths:
    - node_modules
    - dist
job:
  cache: {}
```

定义多个缓存

```
test-job:
  stage: build
  cache:
    - key:
        files:
          - Gemfile.lock
      paths:
        - vendor/ruby
    - key:
        files:
          - yarn.lock
      paths:
        - .yarn-cache/
  script:
    - bundle install --path=vendor
    - yarn install --cache-folder .yarn-cache
    - echo Run tests...
```

## 6.2. stages

定义 stages

```
stages:
  - build
  - test
  - deploy
```

依赖关系

dependencies 可以设置 job 的依赖关系

```
image: mileschou/php-testing-base:7.0

stages:
  - build
  - test
  - deploy

build_job:
  stage: build
  script:
    - composer install
  cache:
    untracked: true
  artifacts:
    paths:
      - vendor/

test_job:
  stage: test
  script:
    - php vendor/bin/codecept run
  dependencies:
    - build_job

deploy_job:
  stage: deploy
  script:
    - echo Deploy OK
  only:
    - release
  when: manual
```

**禁用 stage**

出于某种原因，我们想禁用某些 stage。可以在 job 前加一个 “.” 禁用它。

```
.deploy:
  image: maven:3.6-jdk-11
  tags:
    - shell
  script:
    - 'mvn deploy -s ci_settings.xml'
  # only:
    # - main
```

### 6.3. variables

```
job1:
  variables:
    FOLDERS: src test docs
  script:
    - |
      for FOLDER in $FOLDERS
        do
          echo "The path is root/${FOLDER}"
        done
```

列出所有环境变量

使用 export 列出所有环境变量

```
build-job:
  image: maven:3.8.2-openjdk-17
  stage: build
  # variables:
    # accessKeyId: 123456
    # accessSecret: 654321
  tags:
    - docker
  before_script:
    - export
    - cat src/main/resources/application.properties
  script:
    - mvn clean package -Dmaven.test.skip=true
    - ls target/*.jar
  artifacts:
    name: "$CI_PROJECT_NAME"
    paths:
      - target/*.jar
```

```
$ export
21declare -x CI="true"
22declare -x CI_API_V4_URL="http://192.168.30.5/api/v4"
23declare -x CI_BUILDS_DIR="/builds"
24declare -x CI_BUILD_BEFORE_SHA="213825d0cfd133aadb2648b0c1236f834e98972b"
25declare -x CI_BUILD_ID="4705"
26declare -x CI_BUILD_NAME="build-job"
27declare -x CI_BUILD_REF="61fe2acb56474b4b2ffb289de2c7d93afe514354"
28declare -x CI_BUILD_REF_NAME="development"
29declare -x CI_BUILD_REF_SLUG="development"
30declare -x CI_BUILD_STAGE="build"
31declare -x CI_BUILD_TOKEN="[MASKED]"
32declare -x CI_COMMIT_AUTHOR="neo <neo@t.com>"
33declare -x CI_COMMIT_BEFORE_SHA="213825d0cfd133aadb2648b0c1236f834e98972b"
34declare -x CI_COMMIT_BRANCH="development"
35declare -x CI_COMMIT_DESCRIPTION=""
36declare -x CI_COMMIT_MESSAGE="更新.gitlab-ci.yml文件"
```

```
37declare -x CI_COMMIT_REF_NAME="development"
38declare -x CI_COMMIT_REF_PROTECTED="true"
39declare -x CI_COMMIT_REF_SLUG="development"
40declare -x CI_COMMIT_SHA="61fe2acb56474b4b2ffb289de2c7d93afe514354"
41declare -x CI_COMMIT_SHORT_SHA="61fe2acb"
42declare -x CI_COMMIT_TIMESTAMP="2021-09-18T07:00:58+00:00"
43declare -x CI_COMMIT_TITLE="更新.gitlab-ci.yml文件"
44declare -x CI_CONCURRENT_ID="0"
45declare -x CI_CONCURRENT_PROJECT_ID="0"
46declare -x CI_CONFIG_PATH=".gitlab-ci.yml"
47declare -x CI_DEFAULT_BRANCH="development"
48declare -x
CI_DEPENDENCY_PROXY_GROUP_IMAGE_PREFIX="192.168.30.5:80/neo/dependency_proxy/containers"
49declare -x CI_DEPENDENCY_PROXY_PASSWORD="[MASKED]"
50declare -x CI_DEPENDENCY_PROXY_SERVER="192.168.30.5:80"
51declare -x CI_DEPENDENCY_PROXY_USER="gitlab-ci-token"
52declare -x CI_DISPOSABLE_ENVIRONMENT="true"
53declare -x CI_JOB_ID="4705"
54declare -x CI_JOB_IMAGE="maven:3.8.2-openjdk-17"
55declare -x CI_JOB_JWT="[MASKED]"
56declare -x CI_JOB_NAME="build-job"
57declare -x CI_JOB_STAGE="build"
58declare -x CI_JOB_STARTED_AT="2021-09-18T07:01:07Z"
59declare -x CI_JOB_STATUS="running"
60declare -x CI_JOB_TOKEN="[MASKED]"
61declare -x CI_JOB_URL="http://192.168.30.5/neo/alertmanager-webhook/-/jobs/4705"
62declare -x CI_NODE_TOTAL="1"
63declare -x CI_PAGES_DOMAIN="example.com"
64declare -x CI_PAGES_URL="http://neo.example.com/alertmanager-webhook"
65declare -x CI_PIPELINE_CREATED_AT="2021-09-18T07:00:58Z"
66declare -x CI_PIPELINE_ID="1866"
67declare -x CI_PIPELINE_IID="100"
68declare -x CI_PIPELINE_SOURCE="push"
69declare -x CI_PIPELINE_URL="http://192.168.30.5/neo/alertmanager-webhook/-/pipelines/1866"
70declare -x CI_PROJECT_CLASSIFICATION_LABEL=""
71declare -x CI_PROJECT_DIR="/builds/neo/alertmanager-webhook"
72declare -x CI_PROJECT_ID="23"
73declare -x CI_PROJECT_NAME="alertmanager-webhook"
74declare -x CI_PROJECT_NAMESPACE="neo"
75declare -x CI_PROJECT_PATH="neo/alertmanager-webhook"
76declare -x CI_PROJECT_PATH_SLUG="neo-alertmanager-webhook"
77declare -x CI_PROJECT_REPOSITORY_LANGUAGES="java"
78declare -x CI_PROJECT_ROOT_NAMESPACE="neo"
79declare -x CI_PROJECT_TITLE="Alertmanager Webhook"
80declare -x CI_PROJECT_URL="http://192.168.30.5/neo/alertmanager-webhook"
81declare -x CI_PROJECT_VISIBILITY="public"
82declare -x CI_REGISTRY_PASSWORD="[MASKED]"
83declare -x CI_REGISTRY_USER="gitlab-ci-token"
84declare -x CI_REPOSITORY_URL="http://gitlab-ci-token:[MASKED]@192.168.30.5/neo/alertmanager-
webhook.git"
85declare -x CI_RUNNER_DESCRIPTION="development"
86declare -x CI_RUNNER_EXECUTABLE_ARCH="linux/amd64"
87declare -x CI_RUNNER_ID="23"
88declare -x CI_RUNNER_REVISION="58ba2b95"
89declare -x CI_RUNNER_SHORT_TOKEN="GP-ozvd6"
90declare -x CI_RUNNER_TAGS="docker"
91declare -x CI_RUNNER_VERSION="14.2.0"
92declare -x CI_SERVER="yes"
93declare -x CI_SERVER_HOST="192.168.30.5"
94declare -x CI_SERVER_NAME="GitLab"
95declare -x CI_SERVER_PORT="80"
96declare -x CI_SERVER_PROTOCOL="http"
97declare -x CI_SERVER_REVISION="2da7c857960"
98declare -x CI_SERVER_URL="http://192.168.30.5"
99declare -x CI_SERVER_VERSION="14.2.1"
100declare -x CI_SERVER_VERSION_MAJOR="14"
101declare -x CI_SERVER_VERSION_MINOR="2"
```

```
102declare -x CI_SERVER_VERSION_PATCH="1"
103declare -x FF_CMD_DISABLE_DELAYED_ERROR_LEVEL_EXPANSION="false"
104declare -x FF_DISABLE_UMASK_FOR_DOCKER_EXECUTOR="false"
105declare -x FF_ENABLE_BASH_EXIT_CODE_CHECK="false"
106declare -x FF_GITLAB_REGISTRY_HELPER_IMAGE="true"
107declare -x FF_NETWORK_PER_BUILD="false"
108declare -x FF_SCRIPT_SECTIONS="false"
109declare -x FF_SKIP_DOCKER_MACHINE_PROVISION_ON_CREATION_FAILURE="true"
110declare -x FF_SKIP_NOOP_BUILD_STAGES="true"
111declare -x FF_USE_DIRECT_DOWNLOAD="true"
112declare -x FF_USE_DYNAMIC_TRACE_FORCE_SEND_INTERVAL="false"
113declare -x FF_USE_FASTZIP="false"
114declare -x FF_USE_LEGACY_KUBERNETES_EXECUTION_STRATEGY="false"
115declare -x FF_USE_NEW_BASH_EVAL_STRATEGY="false"
116declare -x FF_USE_POWERSHELL_PATH_RESOLVER="false"
117declare -x FF_USE_WINDOWS_LEGACY_PROCESS_STRATEGY="true"
118declare -x GITLAB_CI="true"
119declare -x GITLAB_FEATURES=""
120declare -x GITLAB_USER_EMAIL="neo@t.com"
121declare -x GITLAB_USER_ID="2"
122declare -x GITLAB_USER_LOGIN="neo"
123declare -x GITLAB_USER_NAME="neo"
124declare -x HOME="/root"
125declare -x HOSTNAME="runner-gp-ozvd6-project-23-concurrent-0"
126declare -x JAVA_HOME="/usr/java/openjdk-17"
127declare -x JAVA_VERSION="17"
128declare -x LANG="C.UTF-8"
129declare -x MAVEN_HOME="/usr/share/maven"
130declare -x OLDPWD="/"
131declare -x PATH="/usr/java/openjdk-
17/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
132declare -x PWD="/builds/neo/alertmanager-webhook"
133declare -x SHLVL="1"
```

**Git submodule**

```
variables:
  GIT_SUBMODULE_STRATEGY: recursive
```

通过条件，设置变量

```
job:
variables:
  DEPLOY_VARIABLE: "default-deploy"
rules:
  - if: $CI_COMMIT_REF_NAME == $CI_DEFAULT_BRANCH
    variables:                              # Override DEPLOY_VARIABLE defined
      DEPLOY_VARIABLE: "deploy-production"  # at the job level.
  - if: $CI_COMMIT_REF_NAME =~ /feature/
    variables:
      IS_A_FEATURE: "true"                  # Define a new variable.
script:
  - echo "Run script with $DEPLOY_VARIABLE as an argument"
  - echo "Run another script if $IS_A_FEATURE exists"
```

例子

```
build-job:
  stage: build
  image: registry.ejiayou.com/share/maven:3.8.6-openjdk-8
  variables:
    VERSION: -SNAPSHOT
  rules:
    - if: $CI_COMMIT_BRANCH == master
      variables:
        VERSION: ".RELEASE"
    - if: $CI_COMMIT_BRANCH == grey
      variables:
        VERSION: ".RELEASE"
  before_script:
    - rm -rf /root/.m2/repository/com/other/*
    - sed -i "s/\"dev\"/\"${CI_COMMIT_BRANCH}\"/" ${MODULE}/src/main/resources/log4j2.xml
    - sed -i "s/2\.3\.7\.RELEASE/2\.7\.7/" ${MODULE}/pom.xml
  script:
    - mvn -U -T 1C clean package -Densd.version=$VERSION
  after_script:
    - md5sum ${MODULE}/target/*.jar
  only:
    - dev
    - test
    - grey
    - master
  tags:
    - kubernetes
  artifacts:
    name: "$CI_PROJECT_NAME"
    paths:
      - ${MODULE}/target/*.jar
```

### 6.4. script /before_script / after_script

before_script: 在 pipeline 运行前执行脚本

after_script: 在 pipeline 完成之后执行脚本

```
cache:
  paths:
    - node_modules
    - dist

before_script:
  - cnpm install

stages:
  - build
  - test
  - deploy

build-dev-job:
  stage: build
  only:
    - development
  script:
    - npm run build:dev
```

```
build-test-job:
  stage: build
  only:
    - testing
  script:
    - npm run build:stage

build-prod-job:
  stage: build
  only:
    - master
  script:
    - npm run build:prod

test-job:
  stage: test
  variables:
    GIT_STRATEGY: none
  script:
    - echo "Running unit tests..."
    - find dist/
    - echo "Test complete."

deploy-dev-job:
  stage: deploy
  variables:
    GIT_STRATEGY: none
  only:
    - development
  script:
    - echo "Deploying application..."
    - rsync -auzv --delete dist/* www@192.168.30.11:/opt/netkiller.cn/admin.netkiller.cn/
    - echo "Application successfully deployed."

deploy-test-job:
  stage: deploy
  variables:
    GIT_STRATEGY: none
  only:
    - testing
  script:
    - echo "Deploying application..."
    - rsync -auzv --delete dist/* www@192.168.30.10:/opt/netkiller.cn/admin.netkiller.cn/
    - echo "Application successfully deployed."

deploy-prod-job:
  stage: deploy
  variables:
    GIT_STRATEGY: none
  only:
    - master
  script:
    - echo "Deploying application..."
    - rsync -auzv --delete dist/* www@139.16.10.12:/opt/netkiller.cn/admin.netkiller.cn/
    - echo "Application successfully deployed."
```

条件判断

```
script:
    - (if [ "$flag" == "true" ]; then kubectl apply -f demo1 --record=true; else kubectl apply
-f demo2 --record=true; fi);
```

```
deploy-dev:
        image: maven
        environment: dev
        tags:
         - kubectl
        script:
         - if [ "$flag" == "true" ]; then MODULE="demo1"; else MODULE="demo2"; fi
         - kubectl apply -f ${MODULE} --record=true
```

多行脚本

```
release-job:
  stage: release
  tags:
    - shell
  only:
    - master
  script:
    - |
      echo -e "
      @sfzito:registry=http://${CI_SERVER_HOST}/api/v4/projects/${CI_PROJECT_ID}/packages/npm/

//${CI_SERVER_HOST}/api/v4/projects/${CI_PROJECT_ID}/packages/npm/:_authToken=${CI_JOB_TOKEN}
      " > .npmrc
    - cnpm publish
  when: manual
```

```
script: |
    if [ "$flag" == "true" ]; then
      kubectl apply -f demo1 --record=true
    else
      kubectl apply -f demo2 --record=true
    fi
```

```
deploy-dev:
image: testimage
environment: dev
tags:
  - kubectl
script:
  - >
    if [ "$flag" == "true" ]; then
      kubectl apply -f demo1 --record=true
    else
      kubectl apply -f demo2 --record=true
    fi
```

### 6.5. only and except

only 用于匹配分支

```
deploy_job:
  stage: deploy
  script:
    - echo Deploy OK
  only:
    - master
  when: manual
```

only 可是使用正则表达式，还可能与 except 一同使用，用于排除分支

```
job:
  # use regexp
  only:
    - /^issue-.*$/
  # use special keyword
  except:
    - branches
```

使用关键字

```
job:
  # use special keywords
  only:
    - tags
    - triggers
```

only和except允许使用特殊的关键字：

- branches 匹配所有 git 分支
- tags 匹配所有 git tag
- triggers

匹配 **feature / hotfix** 分支

```
  only:  # 只对 feature/.* 开头 和 以 feature-.* 开头分支有效
    - /^feature\/.*$/
    - /^feature-.*$/
    - /^hotfix\/.*$/
    - /^hotfix-.*$/
```

匹配 feature / hotfix 分支

```
deploy-feature-job:
  stage: deploy
  variables:
    GIT_STRATEGY: none
    HOST: 192.168.30.14
    # DOCKER_HOST: unix:///var/run/docker.sock mvn clean install docker:build
  environment:
    name: feature
    url: https://api.netkiller.cn
  only:
    - /^feature\/.*/
  tags:
    - shell
  before_script:
    - mvn docker:build -DpushImage
    - rm -rf *.sql.gz
    - mysqldump -hmysql.netkiller.cn test | gzip > test.$(date -u +%Y-%m-%d.%H:%M:%S).sql.gz
  artifacts:
    name: "$CI_PROJECT_NAME"
    paths:
      - ./*.sql.gz
  script:
    - scp src/main/docker/docker-compose.yaml www@$HOST:/opt/netkiller.cn/api.netkiller.cn/
    - ssh www@$HOST "sudo docker-compose -f /opt/netkiller.cn/api.netkiller.cn/docker-
compose.yaml up"
    - ssh www@$HOST "sudo docker-compose -f /opt/netkiller.cn/api.netkiller.cn/docker-
compose.yaml restart"
  when: manual

deploy-hotfix-job:
  stage: deploy
  variables:
    GIT_STRATEGY: none
    HOST: 192.168.30.14
  environment:
    name: hotfix
    url: https://api.netkiller.cn
  only:
    - /^hotfix\/.*/
  tags:
    - shell
  before_script:
    - mvn docker:build -DpushImage
    - rm -rf *.sql.gz
    - mysqldump -hmysql.netkiller.cn test | gzip > test.$(date -u +%Y-%m-%d.%H:%M:%S).sql.gz
  artifacts:
    name: "$CI_PROJECT_NAME"
    paths:
      - ./*.sql.gz
  script:
    - scp src/main/docker/docker-compose.yaml www@$HOST:/opt/netkiller.cn/api.netkiller.cn/
    - ssh www@$HOST "sudo docker-compose -f /opt/netkiller.cn/api.netkiller.cn/docker-
compose.yaml up"
    - ssh www@$HOST "sudo docker-compose -f /opt/netkiller.cn/api.netkiller.cn/docker-
compose.yaml restart"
  when: manual
```

监控文件变化

```
```

```
docker build:
  script: docker build -t netkiller:$CI_COMMIT_REF_SLUG .
  only:
    refs:
      - branches
    changes:
      - Dockerfile
      - dockerfiles/**/*
      - more_scripts/*.{rb,py,sh}
      - "**/*.json"
```

## 6.6. 构建物

保留 api.netkiller.cn/target/*.jar 文件

```
cache:
#   untracked: true
  paths:
    - api.netkiller.cn/target/

stages:
  - build
  - test
  - deploy
  - database

build-job:
  stage: build
  before_script:
    - wechat -t 1 api.netkiller.cn $CI_COMMIT_AUTHOR 在 $CI_COMMIT_BRANCH 分支提交了代码
$CI_COMMIT_MESSAGE 正在构建中
    - voice $(echo "$CI_COMMIT_AUTHOR" | cut -d ' ' -f1) 在 API 项目 $CI_COMMIT_BRANCH 分支提交了
代码，正在构建中
    - if [ "$CI_PIPELINE_SOURCE" == "schedule" ]; then mvn clean; fi
  after_script:
    - wechat -t 1 api.netkiller.cn $CI_COMMIT_AUTHOR 在 $CI_COMMIT_BRANCH 分支代码完成编译和打包
  script:
    - mvn -T 1C -Dmaven.test.skip=true package
    - md5sum */target/*.jar
  artifacts:
    name: "$CI_PROJECT_NAME"
    paths:
      - api.netkiller.cn/target/*.jar
```

所有git没有追踪的文件视为构建物

```
artifacts:
  untracked: true
```

禁止 **job** 下载构建物

```
job:
```

```
  stage: build
  script: make build
  dependencies: []
```

### 6.7. 允许失败

设置当一个job运行失败之后并不影响后续的CI构建过程

```
job1:
  stage: build
  script:
  - execute_script_that_will_fail

job2:
  stage: test
  script:
  - execute_script_that_will_succeed
  allow_failure: true

job3:
  stage: deploy
  script:
  - deploy_to_staging
```

### 6.8. 定义何时开始job

when: 可以是on_success，on_failure，always或者manual

when可以设置以下值：

- on_success: 只有前面stages的所有工作成功时才执行。 这是默认值。
- on_failure: 当前面stages中任意一个jobs失败后执行。
- always: 无论前面stages中jobs状态如何都执行。
- manual: 手动执行

```
```

### 6.9. services

```
services:
- mysql

variables:
  # Configure mysql service (https://hub.docker.com/_/mysql/)
  MYSQL_DATABASE: hello_world_test
  MYSQL_ROOT_PASSWORD: mysql

connect:
  image: mysql
  script:
  - echo "SELECT 'OK';" | mysql --user=root --password="$MYSQL_ROOT_PASSWORD" --host=mysql
```

```
"$MYSQL_DATABASE"
```

### 6.10. tags

在 gitlab-runner register 的时候会提示：Please enter the gitlab-ci tags for this runner (comma separated):

如果你输入了标签就需要在 Pipeline 中设置 tags 否则 Pipeline 将不运行。

```
  only:
    - master
  tags:
    - ansible
```

```
# This file is a template, and might need editing before it works on your project.
# This is a sample GitLab CI/CD configuration file that should run without any modifications.
# It demonstrates a basic 3 stage CI/CD pipeline. Instead of real tests or scripts,
# it uses echo commands to simulate the pipeline execution.
#
# A pipeline is composed of independent jobs that run scripts, grouped into stages.
# Stages run in sequential order, but jobs within stages run in parallel.
#
# For more information, see: https://docs.gitlab.com/ee/ci/yaml/README.html#stages

stages:          # List of stages for jobs, and their order of execution
  - build
  - test
  - deploy

build-job:       # This job runs in the build stage, which runs first.
  stage: build
  tags:
    - neo
  script:
    - echo "Compiling the code..."
    - echo "Compile complete."

unit-test-job:   # This job runs in the test stage.
  stage: test    # It only starts when the job in the build stage completes successfully.
  tags:
    - neo
  script:
    - echo "Running unit tests... This will take about 60 seconds."
    - sleep 60
    - echo "Code coverage is 90%"

lint-test-job:   # This job also runs in the test stage.
  stage: test    # It can run at the same time as unit-test-job (in parallel).
  script:
    - echo "Linting code... This will take about 10 seconds."
    - sleep 10
    - echo "No lint issues found."

deploy-job:      # This job runs in the deploy stage.
  stage: deploy  # It only runs when *both* jobs in the test stage complete successfully.
  script:
    - echo "Deploying application..."
    - echo "Application successfully deployed."
```

```
```

### 6.11. rules 规则

```
job-name:
  script:
    - echo "i am potato"
  rules:
    - if: '$CI_COMMIT_BRANCH != "potato"'
```

条件判断

```
workflow:
  rules:
    - if: '$CI_PIPELINE_SOURCE == "schedule"'
      when: never
    - if: '$CI_PIPELINE_SOURCE == "push"'
      when: never
    - when: always

job:
  script: "echo Hello, Rules!"
  rules:
    - if: '$CI_MERGE_REQUEST_TARGET_BRANCH_NAME == "master"'
      when: always
    - if: '$VAR =~ /pattern/'
      when: manual
    - when: on_success
```

### 6.12. include 包含

```
include:
  - local: '.gitlab-ci-development.yml'
    rules:
      - if: '$CI_COMMIT_BRANCH == "development"'
  - local: '.gitlab-ci-staging.yml'
    rules:
      - if: '$CI_COMMIT_BRANCH == "staging"'
```

```
include:
  - local: builds.yml
    rules:
      - exists:
          - file.md
```

```
```

```
include: 'configs/*.yml'

# This matches all `.yml` files in `configs` and any subfolder in it.
include: 'configs/**.yml'

# This matches all `.yml` files only in subfolders of `configs`.
include: 'configs/**/*.yml'
```

### 6.13. 模版

```
demo1-deploy-dev:
  extends: .deploy-dev
  only:
    variables: [ $flag == "true" ]
  variables:
    MODULE: demo1

demo2-deploy-dev:
  extends: .deploy-dev
  only:
    variables: [ $flag == "false" ]
  variables:
    MODULE: demo2

.deploy-dev:
image: testimage
environment: dev
tags:
  - kubectl
script:
  - kubectl apply -f ${MODULE} --record=true
```

```
cache:
  untracked: true

stages:
  - build
  # - test
  - deploy

build development:
  stage: build
  tags:
    - cloud
  only:
    - development
  except:
    - feature
  script:
    - mvn -T 1C -Dmaven.test.skip=true clean package
  # when: manual

# unit-test-job:
#   stage: test
#   script:
#     - echo "Running unit tests... This will take about 60 seconds."
#     - echo "Code coverage is 90%"
```

```
# lint-test-job:
#   stage: test
#   script:
#     - echo "Linting code... This will take about 10 seconds."
#     - echo "No lint issues found."

deploy development:
  stage: deploy
  tags:
    - cloud
  only:
    - development
  script:
    - \cp -f auth/target/*.jar /opt/netkiller.cn/cloud.netkiller.cn
    - \cp -f gateway/target/*.jar /opt/netkiller.cn/cloud.netkiller.cn
    - \cp -f modules/*/target/*.jar /opt/netkiller.cn/cloud.netkiller.cn
  after_script:
    - python3 /opt/netkiller.cn/ops.netkiller.cn/docker.py -e experiment up
    - python3 /opt/netkiller.cn/ops.netkiller.cn/docker.py -e experiment restart
  when: manual

gateway-dev:
  extends: .deploy-dev
  # only:
    # variables: [ $flag == "true" ]
  variables:
    MODULE: gateway
  environment:
    url: https://${MODULE}.netkiller.cn
  script:
    - \cp -f ${MODULE}/target/*.jar /opt/netkiller.cn/cloud.netkiller.cn

auth-dev:
  extends: .deploy-dev
  variables:
    MODULE: auth
  script:
    - \cp -f ${MODULE}/target/*.jar /opt/netkiller.cn/cloud.netkiller.cn

queue-dev:
  extends: .deploy-dev
  variables:
    MODULE: incar
  script:
    - \cp -f modules/queue/target/*.jar /opt/netkiller.cn/cloud.netkiller.cn

ms-dev:
  extends: .deploy-dev
  variables:
    MODULE: ms
  script:
    - \cp -f modules/ms/target/*.jar /opt/netkiller.cn/cloud.netkiller.cn

system-dev:
  extends: .deploy-dev
  variables:
    MODULE: system
  script:
    - \cp -f modules/system/target/*.jar /opt/netkiller.cn/cloud.netkiller.cn

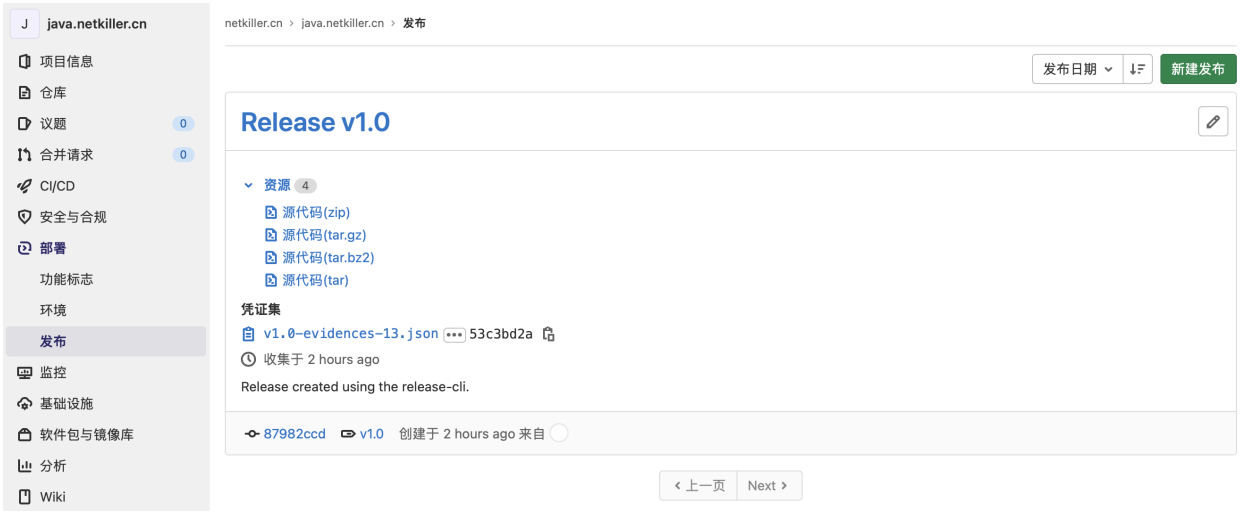
job-dev:
  extends: .deploy-dev
  variables:
    MODULE: job
  script:
    - \cp -f modules/job/target/*.jar /opt/netkiller.cn/cloud.netkiller.cn
```

```
.deploy-dev:
  stage: deploy
  tags:
  - cloud
  only:
  - development
  environment:
    name: development
  when: manual
  # before_script:
    # - mvn -T 1C -Dmaven.test.skip=true clean package
    # - python3 /opt/netkiller.cn/ops.netkiller.cn/docker.py -e experiment ps
  after_script:
    - python3 /opt/netkiller.cn/ops.netkiller.cn/docker.py -e experiment up ${MODULE}
    - python3 /opt/netkiller.cn/ops.netkiller.cn/docker.py -e experiment restart ${MODULE}
```

## 6.14. release

```
stages:
  - build
  - test
  - deploy
  - release

release_job:
  stage: release
  image: registry.gitlab.com/gitlab-org/release-cli:latest
  tags:
    - docker
  rules:
    - if: $CI_COMMIT_TAG                  # Run this job when a tag is created manually
  script:
    - echo 'Running the release job.'
  release:
    name: 'Release $CI_COMMIT_TAG'
    tag_name: '$CI_COMMIT_TAG'
    description: 'Release created using the release-cli.'
```



## 6.15. 应用案例

### Java

```
before_script:
 - echo "Execute scripts which are required to bootstrap the application. !"

after_script:
 - echo "Clean up activity can be done here !."

stages:
 - build
 - test
 - package
 - deploy

variables:
 MAVEN_CLI_OPTS: "--batch-mode"
 MAVEN_OPTS: "-Dmaven.repo.local=.m2/repository"

cache:
 paths:
  - .m2/repository/
  - target/

build:
 stage: build
 image: maven:latest
 script:
  - mvn $MAVEN_CLI_OPTS clean compile

test:
 stage: test
 image: maven:latest
 script:
  - mvn $MAVEN_CLI_OPTS test

package:
 stage: package
 image: maven:latest
 script:
  - mvn $MAVEN_CLI_OPTS package
 artifacts:
  paths: [target/test-0.0.1.war]


deploy_test:
 stage: deploy
 script:
  - echo "########   To be defined   ########"
 environment: staging

deploy_prod:
 stage: deploy
 script:
  - echo "########   To be defined   ########"
 only:
  - master
 environment: production
```

使用 **Docker** 编译并收集构建物

```
#image: java:8
#image: maven:latest
image: maven:3.5.0-jdk-8

stages:
  - build
  - test
  - package

build:
  stage: build
  script: mvn compile

unittest:
  stage: test
  script: mvn test


package:
  stage: package
  script: mvn package
  artifacts:
    paths:
      - target/java-project-0.0.1-SNAPSHOT.jar
```

**Shell 执行器，远程部署物理机/虚拟机**

```
cache:
  untracked: true

stages:
  - build
  - test
  - deploy

build-job:
  stage: build
  tags:
    - shell
  script:
    - mvn clean package -Dmaven.test.skip=true
    - ls target/*.jar
  artifacts:
    name: "$CI_PROJECT_NAME"
    paths:
      - target/*.jar

test-job:
  stage: test
  variables:
    GIT_STRATEGY: none
  only:
    - tags
    - testing
  script:
    - mvn test

deploy-job:
  stage: deploy
```

```
  variables:
    GIT_STRATEGY: none
    HOST: 192.168.30.14
  environment:
    name: development
    url: https://api.netkiller.cn
  only:
    - development
  tags:
    - shell
  before_script:
    - rm -rf *.sql.gz
    - mysqldump -hmysql.netkiller.cn test | gzip > test.$(date -u +%Y-%m-%d.%H:%M:%S).sql.gz
  # after_script:
  artifacts:
    name: "$CI_PROJECT_NAME"
    paths:
      - ./*.sql.gz
  script:
    - rsync -auzv target/*.jar www@$HOST:/opt/netkiller.cn/api.netkiller.cn/
    - ssh -f -C -q www@$HOST "pkill java; sleep 5; java -jar
/opt/netkiller.cn/api.netkiller.cn/alertmanager-0.0.1.jar >/dev/null 2>&1 &"
```

**Shell 执行器，远程部使用容器启动项目**

```
cache:
  untracked: true

stages:
  - build
  - test
  - deploy

build-job:
  stage: build
  tags:
    - shell
  script:
    - mvn clean package -Dmaven.test.skip=true
    - ls target/*.jar
  artifacts:
    name: "$CI_PROJECT_NAME"
    paths:
      - target/*.jar

test-job:
  stage: test
  variables:
    GIT_STRATEGY: none
  only:
    - tags
    - testing
  script:
    - mvn test

deploy-job:
  stage: deploy
  variables:
    GIT_STRATEGY: none
    HOST: 192.168.30.14
  environment:
    name: development
```

```
    url: https://api.netkiller.cn
  only:
    - development
  tags:
    - shell
  before_script:
    - rm -rf *.sql.gz
    - mysqldump -hmysql.netkiller.cn test | gzip > test.$(date -u +%Y-%m-%d.%H:%M:%S).sql.gz
  # after_script:
  artifacts:
    name: "$CI_PROJECT_NAME"
    paths:
      - ./*.sql.gz
  script:
    - rsync -auzv target/*.jar www@$HOST:/opt/netkiller.cn/api.netkiller.cn/
    - rsync -auzv src/main/docker/development/docker-compose.yaml
www@$HOST:/opt/netkiller.cn/api.netkiller.cn/
    - ssh www@$HOST "sudo docker-compose -f /opt/netkiller.cn/api.netkiller.cn/docker-
compose.yaml up -d api"
    - ssh www@$HOST "sudo docker-compose -f /opt/netkiller.cn/api.netkiller.cn/docker-
compose.yaml restart api"
```

**Docker 执行器**

```
cache:
  untracked: true

stages:
  - build
  - test
  - deploy

build-job:
  image: maven:3.8.2-openjdk-17
  stage: build
  tags:
    - docker
  script:
    - mvn clean package -Dmaven.test.skip=true
    - ls target/*.jar
  artifacts:
    name: "$CI_PROJECT_NAME"
    paths:
      - target/*.jar

test-job:
  image: maven:3.8.2-openjdk-17
  stage: test
  variables:
    GIT_STRATEGY: none
  tags:
    - docker
  script:
    - mvn test

deploy-job:
  stage: deploy
  variables:
    GIT_STRATEGY: none
    HOST: 192.168.30.14
  environment:
    name: development
```

```
    url: https://api.netkiller.cn
  only:
    - development
  tags:
    - shell
  before_script:
    # - DOCKER_HOST=unix:///var/run/docker.sock mvn clean install docker:build
    - mvn docker:build -DpushImage
    # - mvn docker:push
    - rm -rf *.sql.gz
    - mysqldump -hmysql.netkiller.cn test | gzip > test.$(date -u +%Y-%m-%d.%H:%M:%S).sql.gz
  artifacts:
    name: "$CI_PROJECT_NAME"
    paths:
      - ./*.sql.gz
  script:
    - scp src/main/docker/docker-compose.yaml www@$HOST:/opt/netkiller.cn/api.netkiller.cn/
    - ssh www@$HOST "sudo docker-compose -f /opt/netkiller.cn/api.netkiller.cn/docker-
compose.yaml up"
    - ssh www@$HOST "sudo docker-compose -f /opt/netkiller.cn/api.netkiller.cn/docker-
compose.yaml restart"
```

**Node**

```
cache:
  paths:
    - node_modules
    # - dist

stages:
  - build
  # - test
  - deploy

build-job:
  stage: build
  script:
    - npm install
#     - yarn install
#     - yarn run build

# unit-test-job:
#   stage: test
#   script:
#     - yarn run test

# lint-test-job:
#   stage: test
#   script:
#     - yarn run lint

deploy-job:
  stage: deploy
  script:
    - rsync -auzv --delete * www@192.168.30.10:/opt/netkiller.cn/www.netkiller.cn/
    - ssh www@192.168.0.10 "sudo pm2 --update-env restart
/opt/netkiller.cn/www.netkiller.cn/main.js"
```

**vue.js android**

```
build site:
  image: node:6
  stage: build
  script:
    - npm install --progress=false
    - npm run build
  artifacts:
    expire_in: 1 week
    paths:
      - dist

unit test:
  image: node:6
  stage: test
  script:
    - npm install --progress=false
    - npm run unit

deploy:
  image: alpine
  stage: deploy
  script:
    - apk add --no-cache rsync openssh
    - mkdir -p ~/.ssh
    - echo "$SSH_PRIVATE_KEY" >> ~/.ssh/id_dsa
    - chmod 600 ~/.ssh/id_dsa
    - echo -e "Host *\n\tStrictHostKeyChecking no\n\n" > ~/.ssh/config
    - rsync -rav --delete dist/ user@server.com:/your/project/path/
```

**Python**

```
cache:
  # key: $CI_COMMIT_REF_SLUG
  paths:
    - .pytest_cache
    - __pycache__

stages:
  - build
  - test
  - report

build-job:
  stage: build
  tags:
    - shell
  script:
    - pip3 install -r requirements.txt

unit-test-job:
  stage: test
  tags:
    - shell
  before_script:
    - wechat -t 2 开始接口自动化测试
  after_script:
    - wechat -t 2 接口自动化测试完成
```

```
  script:
    - cd api_test
    - pytest --no-header --tb=no --alluredir=/dev/shm/allure-results --clean-alluredir | wechat
-t 2 --stdin
    # - wechat -t 2 "$(cat output.log)"

# lint-test-job:
#   stage: test
#   tags:
#     - shell
#   script:
#     - pip3 install pylint
#     - pylint -j 4 api_test/*

report-job:
  stage: report
  tags:
    - shell
  after_script:
    - wechat -t 2 测试报告 http://test.netkiller.cn/test/index.html
  script:
    - allure generate /dev/shm/allure-results -o /dev/shm/allure-report --clean
    - lrsync '/dev/shm/allure-report/*'
www@test.netkiller.cn:/opt/netkiller.cn/test.netkiller.cn/test/
```

**docker**

```
cache:
  untracked: true

stages:
  - build
  - test
  - deploy

build-job:
  image: maven:3.8.2-openjdk-17
  stage: build
  tags:
    - docker
  script:
    - mvn clean package -Dmaven.test.skip=true
    - ls target/*.jar
  artifacts:
    name: "$CI_PROJECT_NAME"
    paths:
      - target/*.jar

test-job:
  image: maven:3.8.2-openjdk-17
  stage: test
  variables:
    GIT_STRATEGY: none
  tags:
    - docker
  script:
    - mvn test

deploy-job:
  stage: deploy
  variables:
    GIT_STRATEGY: none
```

```
    HOST: 192.168.30.14
    DOCKER_HOST: unix:///var/run/docker.sock mvn clean install docker:build
  environment:
    name: development
    url: https://api.netkiller.cn
  only:
    - development
  tags:
    - shell
  before_script:
    - mvn docker:build -DpushImage
    # - mvn docker:push
    - rm -rf *.sql.gz
    - mysqldump -hmysql.netkiller.cn test | gzip > test.$(date -u +%Y-%m-%d.%H:%M:%S).sql.gz
  artifacts:
    name: "$CI_PROJECT_NAME"
    paths:
      - ./*.sql.gz
  script:
    - scp src/main/docker/docker-compose.yaml www@$HOST:/opt/netkiller.cn/api.netkiller.cn/
    - ssh www@$HOST "sudo docker-compose -f /opt/netkiller.cn/api.netkiller.cn/docker-
compose.yaml up"
    - ssh www@$HOST "sudo docker-compose -f /opt/netkiller.cn/api.netkiller.cn/docker-
compose.yaml restart"
```

**include 高级用法**

.gitlab-ci.yml

```
cache:
  key: ${CI_COMMIT_REF_SLUG}
  # untracked: true
  paths:
    - dist

stages:
  - build
  # - test
  - deploy
  # - docker

build-job:
  stage: build
  tags:
    - feature
  only:
    - split
  except:
    - development
  script:
    # - mvn -T 1C -Dmaven.test.skip=true clean package
    - mvn -Dmaven.test.skip=true clean package

include:
  - '.gitlab-ci-docker.yml'
  - local: '.gitlab-ci-development.yml'
    rules:
      - if: '$CI_COMMIT_BRANCH == "development"'
  - local: '.gitlab-ci-staging.yml'
    rules:
      - if: '$CI_COMMIT_BRANCH == "staging"'
```

.gitlab-ci-development.yml

```
build development:
  stage: build
  # tags:
    # - cloud
  only:
    - development
  except:
    - feature
  before_script:
    - rm -rf dist && mkdir -p dist
  script:
    - mvn -T 1C -Dmaven.test.skip=true clean package
    # - mvnd clean package
  after_script:
    - find . \( ! -path "*/common/*" -a ! -path "*/lib/*" -a ! -path "./dist/*" -a ! -path
"./api/*" \) -type f -name "*.jar" -exec \cp -af {} dist \;
    - find dist/ -type f -name "*.jar" -exec md5sum {} \;
  # when: manual

all-in-one:
  stage: deploy
  # tags:
    # - cloud
  only:
    - development
  script:
    # - \cp -af auth/target/*.jar /opt/netkiller.cn/cloud.netkiller.cn
    # - \cp -af gateway/target/*.jar /opt/netkiller.cn/cloud.netkiller.cn
    # - \cp -af modules/*/target/*.jar /opt/netkiller.cn/cloud.netkiller.cn
    # - \cp -af visual/*/target/*.jar /opt/netkiller.cn/cloud.netkiller.cn
    - \cp -af dist/*.jar /opt/netkiller.cn/cloud.netkiller.cn
  after_script:
    - python3 /opt/netkiller.cn/ops.netkiller.cn/docker.py -e experiment up auth incar ms
system job activiti chain gateway
    - python3 /opt/netkiller.cn/ops.netkiller.cn/docker.py -e experiment restart up auth incar
ms system job activiti chain gateway
  when: manual


gateway:
  extends: .deploy-dev
  # only:
    # variables: [ $flag == "true" ]
  variables:
    MODULE: gateway
  environment:
    url: https://${MODULE}.netkiller.cn
  script:
    - \cp -af dist/${MODULE}.jar /opt/netkiller.cn/cloud.netkiller.cn

auth:
  extends: .deploy-dev
  variables:
    MODULE: auth
  script:
    - \cp -af dist/${MODULE}.jar /opt/netkiller.cn/cloud.netkiller.cn

incar:
  extends: .deploy-dev
  variables:
```

```
    MODULE: incar
  script:
    - \cp -af modules/incar/src/main/resources/CFCA /opt/netkiller.cn/cloud.netkiller.cn/
    - \cp -af dist/incar.jar /opt/netkiller.cn/cloud.netkiller.cn

ms:
  extends: .deploy-dev
  variables:
    MODULE: ms
  script:
    - \cp -af dist/ms.jar /opt/netkiller.cn/cloud.netkiller.cn

system:
  extends: .deploy-dev
  variables:
    MODULE: system
  script:
    - \cp -af dist/system.jar /opt/netkiller.cn/cloud.netkiller.cn

job:
  extends: .deploy-dev
  variables:
    MODULE: job
  script:
    - \cp -af dist/job.jar /opt/netkiller.cn/cloud.netkiller.cn

activiti:
  extends: .deploy-dev
  variables:
    MODULE: activiti
  script:
    - \cp -af dist/activiti.jar /opt/netkiller.cn/cloud.netkiller.cn

chain:
  extends: .deploy-dev
  variables:
    MODULE: chain
  script:
    - \cp -af dist/chain.jar /opt/netkiller.cn/cloud.netkiller.cn

msg:
  extends: .deploy-dev
  variables:
    MODULE: msg
  script:
    - \cp -af dist/msg.jar /opt/netkiller.cn/cloud.netkiller.cn

xxl-job-admin:
  extends: .deploy-dev
  variables:
    MODULE: xxl-job-admin
  script:
    - \cp -af dist/xxl-job-admin.jar /opt/netkiller.cn/cloud.netkiller.cn

ev:
  extends: .deploy-dev
  variables:
    MODULE: ev
  script:
    - \cp -af dist/ev.jar /opt/netkiller.cn/cloud.netkiller.cn

sfapi:
  extends: .deploy-dev
  variables:
    MODULE: sfapi
  script:
    - \cp -af dist/sfapi.jar /opt/netkiller.cn/cloud.netkiller.cn
```

```
.deploy-dev:
  stage: deploy
  # tags:
  # - cloud
  only:
  - development
  environment:
    name: development
  when: manual
  # before_script:
    # - mvn -T 1C -Dmaven.test.skip=true clean package
  after_script:
    - python3 /opt/netkiller.cn/ops.netkiller.cn/docker.py -e experiment up ${MODULE}
    - python3 /opt/netkiller.cn/ops.netkiller.cn/docker.py -e experiment restart ${MODULE}
    - python3 /opt/netkiller.cn/ops.netkiller.cn/docker.py -e experiment ps ${MODULE}
```

.gitlab-ci-staging.yml

```
staging build:
  stage: build
  only:
    - staging
  except:
    - feature
  before_script:
    - rm visual/report-platform/src/main/resources/bootstrap.yml
    - rm visual/report-platform/src/main/resources/bootstrap-stage.yml
    # - mv visual/report-platform/src/main/resources/bootstrap-stage.yml visual/report-
platform/src/main/resources/bootstrap.yml
    - ls visual/report-platform/src/main/resources/
  script:
    - mvn -T 1C -Dmaven.test.skip=true clean package

staging:
  stage: deploy
  only:
    - staging
  script:
    - \cp -f auth/target/*.jar /opt/netkiller.cn/cloud.netkiller.cn
    - \cp -f gateway/target/*.jar /opt/netkiller.cn/cloud.netkiller.cn
    - \cp -f modules/*/target/*.jar /opt/netkiller.cn/cloud.netkiller.cn
    - \cp -f visual/*/target/*.jar /opt/netkiller.cn/cloud.netkiller.cn
  after_script:
    - python3 /opt/netkiller.cn/ops.netkiller.cn/docker.py -e stage up
    - python3 /opt/netkiller.cn/ops.netkiller.cn/docker.py -e stage restart
  when: manual

report:
  extends: .deploy-template
  variables:
    MODULE: report
  environment:
    url: https://report.netkiller.cn
  script:
    - wechat -t 3 report.netkiller.cn $CI_COMMIT_AUTHOR Stage 环境「正在部署」
    - rsync -auzv visual/report-platform/target/*.jar
docker@$HOST:/opt/netkiller.cn/cloud.netkiller.cn/

gateway:
  extends: .deploy-template
  # only:
```

```
    # variables: [ $flag == "true" ]
  variables:
    MODULE: gateway
  environment:
    url: https://${MODULE}.netkiller.cn
  script:
    - \cp -f ${MODULE}/target/*.jar /opt/netkiller.cn/cloud.netkiller.cn

auth:
  extends: .deploy-template
  variables:
    MODULE: auth
  script:
    - \cp -f ${MODULE}/target/*.jar /opt/netkiller.cn/cloud.netkiller.cn

incar:
  extends: .deploy-template
  variables:
    MODULE: incar
  script:
    - \cp -f modules/incar/target/*.jar /opt/netkiller.cn/cloud.netkiller.cn

ms:
  extends: .deploy-template
  variables:
    MODULE: ms
  script:
    - \cp -f modules/ms/target/*.jar /opt/netkiller.cn/cloud.netkiller.cn

system:
  extends: .deploy-template
  variables:
    MODULE: system
  script:
    - \cp -f modules/system/target/*.jar /opt/netkiller.cn/cloud.netkiller.cn

job:
  extends: .deploy-template
  variables:
    MODULE: job
  script:
    - \cp -f modules/job/target/*.jar /opt/netkiller.cn/cloud.netkiller.cn


.deploy-template:
  stage: deploy
  variables:
    HOST: uat.netkiller.cn
  # tags:
  # - cloud
  only:
  - staging
  environment:
    name: staging
  when: manual
  before_script:
    - ssh docker@$HOST "sqldump stage -z"
    - wechat -t 3 ${MODULE}.netkiller.cn $CI_COMMIT_AUTHOR Stage 环境「备份数据库」
    # - ssh docker@$HOST "mdump stage -z"
    # - ssh docker@$HOST "cp /opt/netkiller.cn/api.netkiller.cn/admin.jar
/opt/backup/admin.$(date +%Y-%m-%d.%H:%M:%S).jar"
    # - rm -rf *.json.gz
    # - redis-dump -h 192.168.30.10 -d '0' | gzip > redis.db0.$(date +%Y-%m-
%d.%H:%M:%S).json.gz
    # - ssh docker@$HOST "sudo docker-compose -f /opt/netkiller.cn/ops.netkiller.cn/docker-
compose.yaml exec -it redis redis-cli -n 0 flushdb"
  after_script:
```

```
    # - md5sum admin/target/*.jar
    - ssh docker@$HOST "md5sum /opt/netkiller.cn/cloud.netkiller.cn/*.jar"
    - ssh docker@$HOST "python3 /opt/netkiller.cn/ops.netkiller.cn/docker.py -e stage up
${MODULE}"
    - ssh docker@$HOST "python3 /opt/netkiller.cn/ops.netkiller.cn/docker.py -e stage restart
${MODULE}"
    - ssh docker@$HOST "python3 /opt/netkiller.cn/ops.netkiller.cn/docker.py -e stage ps"
    - wechat -t 3 ${MODULE}.netkiller.cn $CI_COMMIT_AUTHOR Stage 环境「完成部署」
```

.gitlab-ci-feature.yml

```
build-feature-job:
  stage: build
  tags:
    - feature
  only:
    - feature
  except:
    - feature
  script:
    - mvn -T 1C -Dmaven.test.skip=true clean package
  # when: manual

# unit-test-job:
#   stage: test
#   script:
#     - echo "Running unit tests... This will take about 60 seconds."
#     - echo "Code coverage is 90%"

# lint-test-job:
#   stage: test
#   script:
#     - echo "Linting code... This will take about 10 seconds."
#     - echo "No lint issues found."

deploy-feature-job:
  stage: build
  tags:
    - cloud
  # variables:
    # HOST: 192.168.30.7
  environment:
    name: feature
    url: https://api.netkiller.cn
  only:
    # - /^feature\/.*/
    - feature
  before_script:
    - mvn -T 1C -Dmaven.test.skip=true clean package
  after_script:
    # - python3 /opt/netkiller.cn/ops.netkiller.cn/docker.py -e experiment up
    - python3 /opt/netkiller.cn/ops.netkiller.cn/docker.py -e experiment restart
    - wechat -t 1 cloud.netkiller.cn $CI_COMMIT_AUTHOR 在 $CI_COMMIT_BRANCH 部署完毕
    # - voice 环境部署完成
  script:
    - \cp -f auth/target/*.jar /opt/netkiller.cn/cloud.netkiller.cn
    - \cp -f gateway/target/*.jar /opt/netkiller.cn/cloud.netkiller.cn
    - \cp -f modules/*/target/*.jar /opt/netkiller.cn/cloud.netkiller.cn
    - \cp -f visual/*/target/*.jar /opt/netkiller.cn/cloud.netkiller.cn
    # - cp -r admin/src/main/resources/CFCA /opt/netkiller.cn/api.netkiller.cn/
  # when: manual
```

# 7. 软件包与镜像库

## 7.1. Maven 仓库

项目目录下面创建 ci_settings.xml 文件

```
<settings xmlns="http://maven.apache.org/SETTINGS/1.1.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.1.0
http://maven.apache.org/xsd/settings-1.1.0.xsd">
  <servers>
    <server>
      <id>gitlab-maven</id>
      <configuration>
        <httpHeaders>
          <property>
            <name>Job-Token</name>
            <value>${env.CI_JOB_TOKEN}</value>
          </property>
        </httpHeaders>
      </configuration>
    </server>
  </servers>
</settings>
```

修改 pom.xml 文件添加下面内容

```
<repositories>
  <repository>
    <id>gitlab-maven</id>
    <url>${env.CI_API_V4_URL}/projects/${env.CI_PROJECT_ID}/packages/maven</url>
  </repository>
</repositories>
<distributionManagement>
  <repository>
    <id>gitlab-maven</id>
    <url>${CI_API_V4_URL}/projects/${env.CI_PROJECT_ID}/packages/maven</url>
  </repository>
  <snapshotRepository>
    <id>gitlab-maven</id>
    <url>${CI_API_V4_URL}/projects/${env.CI_PROJECT_ID}/packages/maven</url>
  </snapshotRepository>
</distributionManagement>
```

修改 .gitlab-ci.yml 添加 Maven 部署命令

Docker 执行器

```
deploy:
  image: maven:3.6-jdk-11
  script:
    - 'mvn deploy -s ci_settings.xml'
  only:
```

```
    - main
```

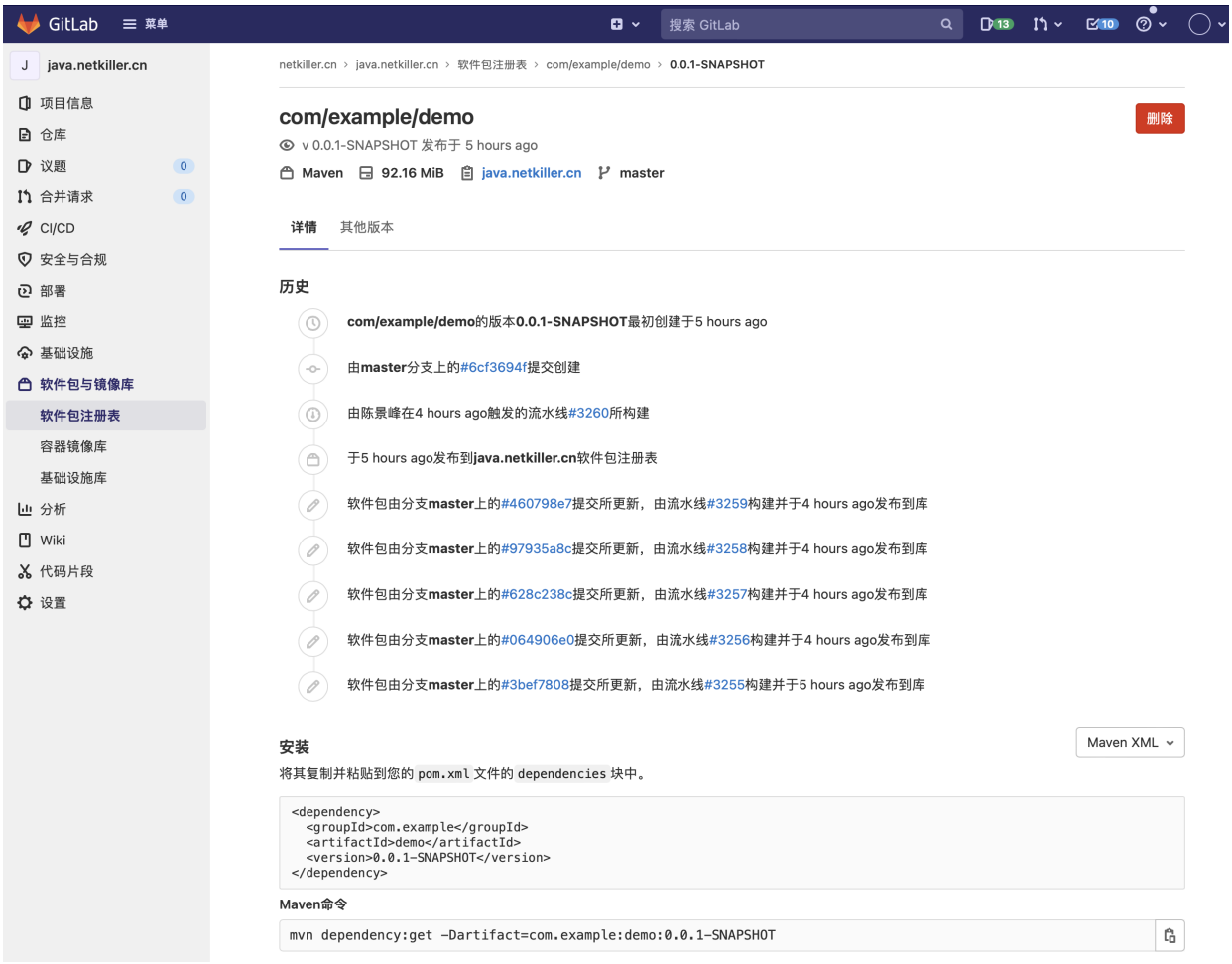
Shell 执行器

```
deploy:
  script:
    - 'mvn deploy -s ci_settings.xml'
  only:
    - main
```

Maven 部署的软件包




进入查看详情




将已存在的 JAR 文件部署到 Maven 仓库

```
package:
  stage: deploy
  variables:
    GIT_STRATEGY: none
  script:
    - mvn deploy:deploy-file -DrepositoryId=gitlab-maven -
Durl=http://registry.netkiller.cn/api/v4/projects/14/packages/maven -Dpackaging=jar -
Dfile=lib/cfca.jar -DgroupId=cn.netkiller -DartifactId=cfca -Dversion=1.0.0 -
Dmaven.test.skip=true -s .ci_settings.xml
    - mvn deploy:deploy-file -DrepositoryId=gitlab-maven -
Durl=http://registry.netkiller.cn/api/v4/projects/14/packages/maven -Dpackaging=jar -
Dfile=lib/ra-toolkit-3.6.28.2.jar -DgroupId=cn.netkiller -DartifactId=ra-toolkit -
Dversion=3.6.28.2 -Dmaven.test.skip=true -s .ci_settings.xml
    - mvn deploy -s .ci_settings.xml -Dmaven.test.skip=true
  when: manual
  allow_failure: true
  only:
    - testing
```

## 7.2. Python Pypi 仓库

# 个人访问令牌

## 创建访问令牌




**提示**

需要勾选 ☑ `api, read_api, read_registry, write_registry` 四个授权




将令牌复制出来保存好

## 手工上传包

创建或编辑 ~/.pypirc 文件

```
[distutils]
index-servers =
    gitlab

[gitlab]
repository = https://gitlab.example.com/api/v4/projects/<project_id>/packages/pypi
username = <your_personal_access_token_name>
password = <your_personal_access_token>
```

用户和密码，可以使用个人访问令牌、部署令牌和 Gitlab 用户密码

```
<project_id> 替换成你的项目URL 或者 项目 ID

例如我的项目地址是：http://registry.netkiller.cn/netkiller.cn/python.netkiller.cn/-/packages

repository =
https://gitlab.example.com/api/v4/projects/netkiller.cn%2Fpython.netkiller.cn/packages/pypi

将 “/” 替换成 “%2F”
```

查看项目 ID




下面是我配置，仅供参考

```
Neo-iMac:devops neo$ cat ~/.pypirc
[distutils]
index-servers =
    gitlab

[gitlab]
repository = http://registry.netkiller.cn/api/v4/projects/30/packages/pypi
username=pypi
password=QFatUEzEybBR6gxxF5K2
```

上传命令

```
Neo-iMac:devops neo$ python3 setup.py sdist bdist_wheel
Neo-iMac:devops neo$ twine upload --repository gitlab dist/*
```

上传演示

```
Neo-iMac:devops neo$ twine upload --repository gitlab dist/netkiller-devops-0.3.*
Uploading distributions to http://registry.netkiller.cn/api/v4/projects/30/packages/pypi
Uploading netkiller-devops-0.3.0.tar.gz
100%|████████████████████████████████████████████████████████████████████████| 37.3k/37.3k
[00:00<00:00, 426kB/s]
Uploading netkiller-devops-0.3.1.tar.gz
100%|████████████████████████████████████████████████████████████████████████| 37.3k/37.3k
[00:00<00:00, 462kB/s]
Uploading netkiller-devops-0.3.2.tar.gz
100%|████████████████████████████████████████████████████████████████████████| 37.3k/37.3k
[00:00<00:00, 436kB/s]
Uploading netkiller-devops-0.3.3.tar.gz
100%|████████████████████████████████████████████████████████████████████████| 37.5k/37.5k
[00:00<00:00, 486kB/s]
Uploading netkiller-devops-0.3.4.tar.gz
100%|████████████████████████████████████████████████████████████████████████| 37.4k/37.4k
[00:00<00:00, 475kB/s]
Uploading netkiller-devops-0.3.5.tar.gz
100%|████████████████████████████████████████████████████████████████████████| 37.5k/37.5k
[00:00<00:00, 490kB/s]
Neo-iMac:devops neo$
```

查看软件包




查看详细信息




## 在持续集成中配置

登陆 gitlab-runner 所在的服务器，如果只有 python 项目，建议使用 root 账号安装 twine 包

```
[root@localhost ~]# pip3 install twine
```

如果 gitlab-runner 是公共服务器，上面还会持续部署其他项目，为了项目更好隔离，可以使用 --user 参数，本地化安装

切换到 gitlab-runner，因为编译和打包，上传包都需要工作在 gitlab-runner 账号下面

```
[root@localhost ~]# su - gitlab-runner
```

安装 twine wheel 包

```
[gitlab-runner@localhost ~]$ pip3 install --user twine wheel
```

twine 将会被安装到 ~/.local/bin/twine 目录

```
[gitlab-runner@localhost ~]$ ls ~/.local/bin/twine
/home/gitlab-runner/.local/bin/twine
```

当然也可以将 twine wheel 放在 .gitlab-ci.yml 文件中，只是每次都安装一次，会影响构建性能。

```
cache:
  key: "$CI_COMMIT_REF_SLUG"
  paths:
    - dist/
stages:
  - build
  - test
  - deploy

build-job:
  stage: build
  tags:
    - shell
  before_script:
    - pip3 install --user netkiller-devops
    - pip3 install --user wheel twine
  script:
    - python3 setup.py sdist bdist_wheel
  # after_script:

deploy-job:
  stage: deploy
  tags:
    - shell
  before_script:
    - |
      cat > ~/.pypirc <<EOF
      [distutils]
      index-servers =
          gitlab

      [gitlab]
      repository = http://registry.netkiller.cn/api/v4/projects/30/packages/pypi
      username=pypi
      password=TUyGJW89wkdfjdh7QWAe
      EOF
    - cat  ~/.pypirc
  script:
    - ~/.local/bin/twine upload --repository gitlab dist/*
```

### 7.3. Node JS

创建项目访问令牌，这里不再赘述，前面已经讲过。

登陆到 gitlab-runner 服务器，安装 Node JS 环境

```
[root@localhost ~]# dnf install -y nodejs
[root@localhost ~]# npm install -g --registry=https://registry.npm.taobao.org cnpm
[root@localhost ~]# npm install -g --registry=https://registry.npm.taobao.org yarn2
```

打开 Node JS 项目，编辑 package.json 文件，修改项目名称 加入 scope 例如 "name": "demo" 改为 "name": "@netkiller/demo"，设置一个版本号 "version": "0.0.1"，然后将 "private": true 改为 "private": false

```
{
  "name": "@netkiller/demo",
  "version": "0.0.1",
  "private": false,
  "scripts": {
    "start": "node ./bin/www",
    "test": "mocha"
  },
  "dependencies": {
    "cookie-parser": "~1.4.3",
    "debug": "~2.6.9",
    "express": "~4.16.0",
    "http-errors": "~1.6.2",
    "morgan": "~1.9.0",
    "pug": "2.0.0-beta11"
  },
  "devDependencies": {
    "mocha": "^5.1.1",
    "supertest": "^3.0.0"
  }
}
```

配置 .gitlab-ci.yml 文件

```
cache:
  paths:
    - node_modules
    - dist

stages:
  - build
  - test
  - deploy

build-job:
  stage: build
  tags:
    - shell
```

```
  script:
    - cnpm install

deploy-job:
  stage: deploy
  tags:
    - shell
  script:
    - |
      echo -e "

@netkiller:registry=http://${CI_SERVER_HOST}/api/v4/projects/${CI_PROJECT_ID}/packages/npm/

//${CI_SERVER_HOST}/api/v4/projects/${CI_PROJECT_ID}/packages/npm/:_authToken=${CI_JOB_TOKEN}
      " > .npmrc
    - cnpm publish
```

## 7.4. Docker registry

Gitlab 默认不打开 docker registry 的功能，需要修改配置打开。

修改配置 /etc/gitlab/gitlab.rb 文件，将 registry_external_url 的值修改为 http://registry.netkiller.cn

**提示**

注意不能使用IP地址，如果使用IP地址必须配合端口号，且端口不能跟 Gitlab 冲突。

```
[root@gitlab ~]# grep 'registry_external_url' /etc/gitlab/gitlab.rb
# registry_external_url 'https://registry.example.com'
registry_external_url 'http://registry.netkiller.cn'
```

让配置生效

```
[root@gitlab ~]# gitlab-ctl reconfigure
```

检查配置文件

```
[root@gitlab ~]# cat /var/opt/gitlab/nginx/conf/gitlab-registry.conf
# This file is managed by gitlab-ctl. Manual changes will be
# erased! To change the contents below, edit /etc/gitlab/gitlab.rb
# and run `sudo gitlab-ctl reconfigure`.

## Lines starting with two hashes (##) are comments with information.
## Lines starting with one hash (#) are configuration parameters that can be uncommented.
##
###################################
##         configuration         ##
###################################


server {
  listen *:80;
  server_name registry.netkiller.cn;
```

```
  server_tokens off; ## Don't show the nginx version number, a security best practice

  client_max_body_size 0;
  chunked_transfer_encoding on;


  ## Real IP Module Config
  ## http://nginx.org/en/docs/http/ngx_http_realip_module.html

  ## HSTS Config
  ## https://www.nginx.com/blog/http-strict-transport-security-hsts-and-nginx/
  add_header Strict-Transport-Security "max-age=63072000";

  access_log  /var/log/gitlab/nginx/gitlab_registry_access.log gitlab_access;
  error_log   /var/log/gitlab/nginx/gitlab_registry_error.log error;

  location / {

    proxy_set_header Host $http_host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Proto http;

    proxy_read_timeout                  900;
    proxy_cache off;
    proxy_buffering off;
    proxy_request_buffering off;
    proxy_http_version 1.1;

    proxy_pass          http://localhost:5000;
  }


}
```




## 配置 Docker registry

在 Gitlab Runner 运行的机器上配置 Docker registry

配置 Docker 的 daemon.json 配置文件

```
{
  "experimental": false,
  "features": {
    "buildkit": true
  },
  "builder": {
    "gc": {
      "defaultKeepStorage": "20GB",
      "enabled": true
    }
  },
  "insecure-registries": [
    "registry.netkiller.cn"
  ]
}
```

重启 Docker 让 daemon.json

```
```

```
[root@gitlab ~]# systemctl reload docker
```

我使用的 Docker Desktop for Mac，在 GUI 中配置 daemon.json 然后重启 Docker Desktop

配置 /etc/hosts 文件

```
Neo-iMac:nginx neo$ grep 'registry' /etc/hosts
192.168.30.5    registry.netkiller.cn
```

Docker 登陆到 registry.netkiller.cn，登陆可以使用 gitlab 用户和密码，可以使用“个人访问令牌”和“部署令牌”，创建令牌需要给予 read_registry 和 write_registry 权限。

```
Neo-iMac:nginx neo$ docker login registry.netkiller.cn -u neo
Password:
Login Succeeded
```

登陆成功会显示 Login Succeeded 并且会在 ~/.docker/config.json 产生配置项

```
Neo-iMac:nginx neo$ cat ~/.docker/config.json
{
        "auths": {
                "https://index.docker.io/v1/": {},
                "registry.netkiller.cn": {}
        },
        "credsStore": "desktop"
}
```

手动构建镜像并上传至容器镜像库

构建镜像

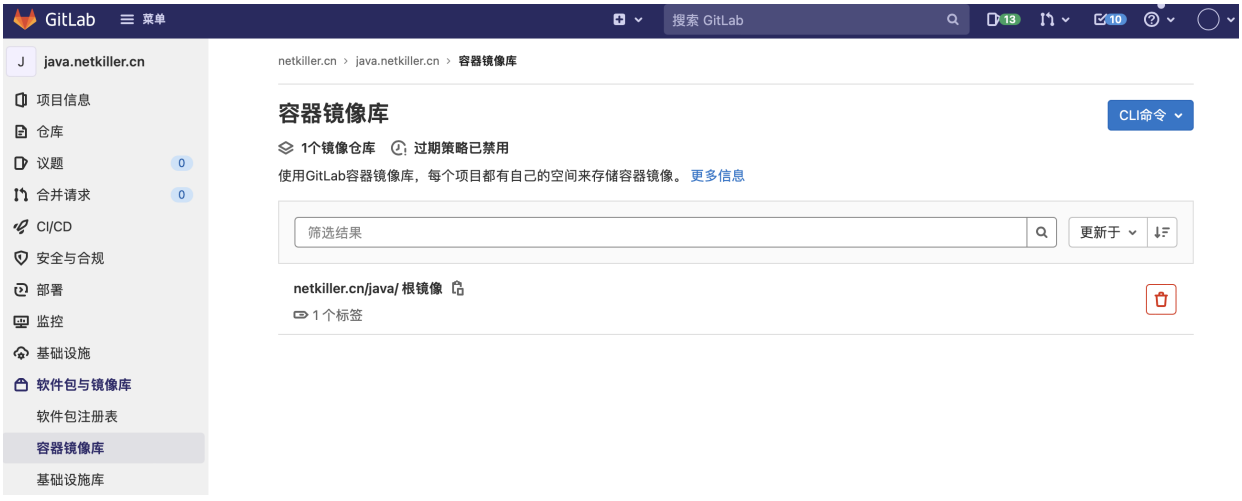
```
Neo-iMac:nginx neo$ docker build -t registry.netkiller.cn/netkiller.cn/java .
[+] Building 4.5s (9/9) FINISHED
 => [internal] load build definition from Dockerfile
0.3s
 => => transferring dockerfile: 37B
0.0s
 => [internal] load .dockerignore
0.4s
 => => transferring context: 2B
0.0s
 => [internal] load metadata for docker.io/library/nginx:latest
3.1s
 => [auth] library/nginx:pull token for registry-1.docker.io
0.0s
 => [1/4] FROM
docker.io/library/nginx:latest@sha256:dfef797ddddfc01645503cef9036369f03ae920cac82d344d58b637ee
861fda1
```

```
0.0s
 => CACHED [2/4] RUN apt update -y && apt install -y procps
0.0s
 => CACHED [3/4] RUN apt install -y iproute2 net-tools
0.0s
 => CACHED [4/4] WORKDIR /opt
0.0s
 => exporting to image
0.4s
 => => exporting layers
0.0s
 => => writing image sha256:549089448b9450a2515fd4653f35c4bb828079624edcbdbc2f0607ba3656598b
0.0s
 => => naming to registry.netkiller.cn/netkiller.cn/java
```

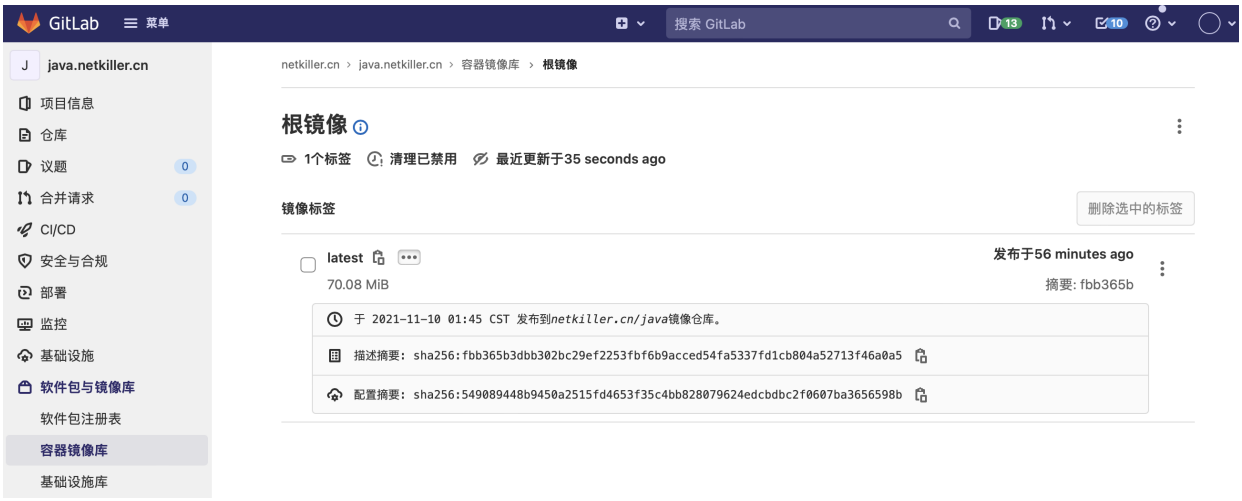
推送镜像

```
Neo-iMac:nginx neo$ docker push registry.netkiller.cn/netkiller.cn/java
Using default tag: latest
The push refers to repository [registry.netkiller.cn/netkiller.cn/java]
5f70bf18a086: Pushed
2d4c9573c0b6: Pushed
a8935bae4a3d: Pushed
280fbd619253: Pushed
921ee7f55927: Pushed
fc199aaed79a: Pushed
38aec0f8e5ed: Pushed
ea56d6ebf7e5: Pushed
e8b689711f21: Pushed
latest: digest: sha256:fbb365b3dbb302bc29ef2253fbf6b9acced54fa5337fd1cb804a52713f46a0a5 size:
2199
```

推送完成后，前往“容器镜像库”可以看到镜像




查看镜像




## CI/CD 流水线配置

Maven 项目

pom.xml 中添加

```
    <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
    <java.version>1.8</java.version>
    <docker.registry>registry.netkiller.cn</docker.registry>
    <docker.registry.name>netkiller.cn</docker.registry.name>
    <docker.image>api.netkiller.cn</docker.image>
    <docker.baseImage>openjdk:8-alpine</docker.baseImage>
</properties>
```

plugins 插件添加

```
                    <plugin>
                            <groupId>com.spotify</groupId>
                            <artifactId>docker-maven-plugin</artifactId>
                            <version>1.2.2</version>
                            <configuration>
                                    <!--
<imageName>${docker.registry}/${docker.registry.name}/${project.artifactId}</imageName> -->
                <imageName>${docker.registry}/${docker.registry.name}/${docker.image}
</imageName>
                                    <baseImage>${docker.baseImage}</baseImage>
                                    <maintainer>netkiller@msn.com</maintainer>
                                    <volumes>/tmp</volumes>
                                    <workdir>/srv</workdir>
                                    <env>
                                            <JAVA_OPTS>-server -Xms512m -Xmx4096m -
Djava.security.egd=file:/dev/./urandom</JAVA_OPTS>
                                    </env>
                                    <exposes>8080</exposes>
                                    <entryPoint>["sh", "-c", "java ${JAVA_OPTS} -jar
/srv/${project.build.finalName}.jar"]</entryPoint>
```

```
                                        <resources>
                                                <resource>
                                                        <targetPath>/srv</targetPath>
                                                        <directory>${project.build.directory}
</directory>

<include>${project.build.finalName}.jar</include>
                                                </resource>
                                        </resources>

<registryUrl>http://${docker.registry}/v2/</registryUrl>
                                        <imageTags>
                                                <imageTag>${project.version}</imageTag>
                                                <imageTag>latest</imageTag>
                                        </imageTags>
                                </configuration>
                        </plugin>
```

.gitlab-ci.yml

```
docker:
  stage: deploy
  before_script:
    - echo "glpat-amwpx6FWS1_mHTNUV7RU" | docker login -u "docker" --password-stdin
registry.netkiller.cn
  script:
    - mvn docker:build && mvn docker:push
  after_script:
    - wechat -t 1 api.netkiller.cn Docker 镜像制作完成
http://192.168.30.5/netkiller.cn/api.netkiller.cn/container_registry
  when: manual
  allow_failure: true
  only:
    - testing
```

下面在给一个 node js 项目的例子

准备 Dockerfile 文件

```
FROM nginx:alpine
LABEL author="neo"
VOLUME /etc/nginx
COPY dist/ /usr/share/nginx/html/
EXPOSE 80
WORKDIR /usr/share/nginx/html/
```

.gitlab-ci.yml

```
cache:
  key: ${CI_COMMIT_REF_SLUG}
  paths:
    - node_modules/
```

```
    - dist/
    - .sonar/

stages:
  - build
  - test
  - deploy
  - release

deploy feature:
  stage: build
  # variables:
  environment:
    name: feature
    url: https://www.netkiller.cn
  only:
    - /^feature\/.*/
    - feature
  tags:
    - cloud
  before_script:
    - cnpm install
    - rm -rf dist/*
    - cnpm run build:stage
  after_script:
    - wechat -t 1 www.netkiller.cn $CI_COMMIT_AUTHOR 在 $CI_COMMIT_BRANCH 环境部署完成
    - voice feature 环境部署完成
  script:
    - sed -i "s/{{description}}/$(date +"%Y-%m-%d %H:%M:%S")/" dist/index.html
    # - rsync -auv --delete dist/* /opt/netkiller.cn/car.netkiller.cn/
    - rm -rf /opt/netkiller.cn/car.netkiller.cn/*
    - \cp -af dist/* /opt/netkiller.cn/car.netkiller.cn/

feature docker:
  stage: deploy
  # variables:
  environment:
    name: feature
    url: https://www.netkiller.cn
  only:
    - /^feature\/.*/
    - feature
  tags:
    - cloud
  before_script:
    - echo "glpat-amwpx6FWS1_mHTNUV7RU" | docker login -u "docker" --password-stdin
registry.netkiller.cn
  after_script:
    - docker push registry.netkiller.cn/netkiller.cn/www.netkiller.cn -a
    - wechat -t 1 www.netkiller.cn $CI_COMMIT_AUTHOR 在 $CI_COMMIT_BRANCH Docker 镜像构建完成
    - voice feature 环境部署完成
  script:
    - docker build -t "registry.netkiller.cn/netkiller.cn/www.netkiller.cn:$(date +%Y-%m-
%d.%H%M)" .

release-job:
  stage: release
  tags:
    - shell
  only:
    - master
  script:
    - |
      echo -e "
      @sfzito:registry=http://${CI_SERVER_HOST}/api/v4/projects/${CI_PROJECT_ID}/packages/npm/

//${CI_SERVER_HOST}/api/v4/projects/${CI_PROJECT_ID}/packages/npm/:_authToken=${CI_JOB_TOKEN}
```

```
    " > .npmrc
  - cnpm publish
 when: manual
```

使用 ${CI_COMMIT_SHORT_SHA} 版本号作为镜像版本

```
feature docker:
  stage: deploy
  # variables:
  environment:
    name: feature
    url: https://admin.netkiller.cn
  only:
    - /^feature\/.*/
    - feature
  tags:
    - cloud
  before_script:
    - echo "glpat-amwpx6FWS1_mHTNUV7RU" | docker login -u "docker" --password-stdin
registry.netkiller.cn
  after_script:
    - docker push
"registry.netkiller.cn/netkiller.cn/admin.netkiller.cn:${CI_COMMIT_SHORT_SHA}"
    - wechat -t 1 admin.netkiller.cn $CI_COMMIT_AUTHOR 在 $CI_COMMIT_BRANCH Docker 镜像构建完成
    - voice feature 环境部署完成
  script:
    - docker build -t
"registry.netkiller.cn/netkiller.cn/admin.netkiller.cn:${CI_COMMIT_SHORT_SHA}" .
```

# 8. 服务器端 hooks

## Git server hooks

## 8.1. 创建全局 Server hooks

https://docs.gitlab.com/ee/administration/server_hooks.html

配置 custom_hooks_dir

```
vim /etc/gitlab/gitlab.rb

# 这个配置已经作废
gitlab_shell['custom_hooks_dir'] =
"/opt/gitlab/embedded/service/gitlab-shell/hooks"

# 在 gitaly 下面加入配置
gitaly['custom_hooks_dir'] =
"/var/opt/gitlab/gitaly/custom_hooks"
```

```
mkdir -p /var/opt/gitlab/gitaly/custom_hooks
vim /var/opt/gitlab/gitaly/custom_hooks/commit-msg
chmod +x /var/opt/gitlab/gitaly/custom_hooks/commit-msg
```

多个配置可以创建一个 commit-msg.d 目录，然后把多个脚本放入该目录

```
root@netkiller:/opt/gitlab# mkdir -p
/var/opt/gitlab/gitaly/custom_hooks/commit-msg.d
root@netkiller:/opt/gitlab# vim
```

```
/var/opt/gitlab/gitaly/custom_hooks/commit-msg.d/commit-msg
root@netkiller:/opt/gitlab# chmod +x
/var/opt/gitlab/gitaly/custom_hooks/commit-msg.d/commit-msg
root@netkiller:/opt/gitlab# gitlab-ctl reconfigure
```

## 8.2. 给单个仓库配置 Server hooks

查看仓库目录

https://gitlab.netkiller.cn/admin/projects/chenjingfeng/backup

Gitaly storage name: default

Gitaly relative path: @hashed/10/86/1086d35563c495c1cecbce12135cab3b945e01dd185ea2c1dc8ace5ad988977e.git

```
root@9b03d2708db7:/var/opt/gitlab# cat
/var/opt/gitlab/gitaly/config.toml | grep ^path
path = '/var/opt/gitlab/git-data/repositories'

root@9b03d2708db7:/var/opt/gitlab# cd /var/opt/gitlab/git-
data/repositories
root@9b03d2708db7:/var/opt/gitlab/git-data/repositories# cd
\@hashed/10/86/1086d35563c495c1cecbce12135cab3b945e01dd185ea2c1
dc8ace5ad988977e.git
```

创建 **hooks** 脚本

```
mkdir -p custom_hooks
vim custom_hooks/commit-msg
chmod +x custom_hooks/commit-msg
```

# 9. 客户端 hooks

## 9.1. 集成禅道

### Linux/MacOS

配置模版目录

```
test -d ~/workspace/template/hooks && exit
pip3 install requests
mkdir -p ~/workspace/template/hooks
curl -s
https://raw.githubusercontent.com/netkiller/devops/master/share
/git/hooks/commit-msg -o ~/workspace/template/hooks/commit-msg
git config --global init.templatedir ~/workspace/template/
```

已存在项目需要手工处理，运行下面脚本

```
pip3 install requests
curl -s
https://raw.githubusercontent.com/netkiller/devops/master/share
/git/hooks/commit-msg -o .git/hooks/commit-msg
chmod +x .git/hooks/commit-msg
```

### Windows

手工安装 Python 下载地址 https://www.python.org/ftp/python/3.11.1/python-3.11.1-amd64.exe，安装到 C:\Python 目录下

Window 11 也可以使用 Winget 安装

```
winget install python
```

安装完成之后安装依赖包

```
pip3 install requests
```

设置模板

```
mkdir c:\workspace\template\hooks
powershell curl -o c:\workspace\template\hooks\commit-msg
https://raw.githubusercontent.com/netkiller/devops/master/share
/git/hooks/commit-msg
git config --global init.templatedir c:\workspace\template
git config -l
```

已存在项目安装 Script

```
powershell curl -o .git/hooks/commit-msg
https://raw.githubusercontent.com/netkiller/devops/master/share
/git/hooks/commit-msg
```

## 使用方法

提交信息格式

代码提交时，提交信息这样写：

```
BUG 1234
```

如果本次提交代码修复了多个 BUG 这样写：

```
BUG 123 456 789
```

如果是需求，这样写：

```
TASK 123
```

还可以这样写：

```
BUG 123 456 789
TASK 1223 4556 1789
```

临时提交，不关联BUG和TASK这样写：

```
TMP 随便写点啥
```

# 10. WebHook

# 11. FAQ

## 11.1. 查看日志

```
gitlab-ctl tail
gitlab-ctl tail gitlab-rails
gitlab-ctl tail nginx/gitlab_error.log
```

## 11.2. debug runner

```
gitlab-runner -debug run
```

## 11.3. gitolite 向 gitlab 迁移

早期gitlab使用gitolite为用户提供SSH服务，新版gitlab有了更好的解决方案gitlab-shell。安装新版本是必会涉及gitolite 向 gitlab 迁移，下面是我总结的一些迁移经验。

第一步,将gitolite复制到gitlab仓库目录下

```
# cp -r /gitroot/gitolite/repositories/* /var/opt/gitlab/git-
data/repositories/
```

执行导入处理程序

```
# gitlab-rake gitlab:import:repos
```

上面程序会处理一下目录结构，例如

进入gitlab web界面，创建仓库与导入的仓库同名，这样就完成了导入工作。

**提示**

转换最好在git用户下面操作，否则你需要运行

```
# chown git:git -R /var/opt/gitlab/git-data/repositories
```

## 11.4. 修改主机名

默认Gitlab采用主机名，给我使用代理一定麻烦

```
git@hostname:example.com/www.example.com.git
http://hostname/example.com/www.example.com.git
```

我们希望使用IP地址替代主机名

```
git@172.16.0.1:example.com/www.example.com.git
http://172.16.0.1/example.com/www.example.com.git
```

编辑 /etc/gitlab/gitlab.rb 配置文件

```
external_url 'http://172.16.0.1'
```

重新启动Gitlab

```
# gitlab-ctl reconfigure
# gitlab-ctl restart
```

## 11.5. ERROR: Uploading artifacts as "archive" to coordinator... too large archive

持续集成提示错误

```
ERROR: Uploading artifacts as "archive" to coordinator... too
large archive  id=185 responseStatus=413 Request Entity Too
Large status=413 token=HKerPDE6
FATAL: too large
ERROR: Job failed: exit status 1
```

解决方案

Admin - Settings - CI/CD - Continuous Integration and Deployment

点击 Expand 展开配置项

Maximum artifacts size (MB): 修改构建无最大尺寸

## 11.6. ERROR: Job failed (system failure): prepare environment: waiting for pod running: timed out waiting for pod to start. Check https://docs.gitlab.com/runner/shells/index.html#shell-profile-loading for more information

Kubernetes 执行器提出该错误，分析原因是 pull 镜像超时。

解决方法，设置 poll_timeout = 3600

## 11.7. 磁盘 100% 怎样清理

删除过期的构建物

```
[root@netkiller ~]# rm -rf /opt/gitlab/data/gitlab-
rails/shared/artifacts/*/*/*/2021_*
```

清理日志

```
[root@netkiller ~]# find /opt/gitlab/logs -name "*.gz" -exec rm
-rf {} \;
```

# 第 12 章 Jenkins

# 1. 安装 Jenkins

## 1.1. OSCM 一键安装

```
yum install -y java-1.8.0-openjdk
curl -s
https://raw.githubusercontent.com/oscm/shell/master/project/jen
kins/jenkins.sh | bash
```

## 1.2. Mac

使用 pkg 方式安装，默认路径是 /Applications/Jenkins/jenkins.war

```
export
JAVA_HOME=/Library/Java/JavaVirtualMachines/jdk1.8.0_92.jdk/Con
tents/Home
java -jar jenkins.war --httpPort=8080
```

浏览器访问：http://localhost:8080

查看默认密码 /Users/neo/.jenkins/secrets/initialAdminPassword

```
neo@MacBook-Pro ~ % cat
/Users/neo/.jenkins/secrets/initialAdminPassword
6c7369afc6c1414586b6644657dd655a
```

下载 cloudbees 插件

```
neo@MacBook-Pro ~ % cd ~/.jenkins/plugins
neo@MacBook-Pro ~/.jenkins/plugins % wget
ftp://ftp.icm.edu.pl/packages/jenkins/plugins/cloudbees-
folder//6.7/cloudbees-folder.hpi
```

重启 Jenkens http://localhost:8080/restart

复制上面的密码，粘贴到浏览器中。

卸载 Jenkens

```
sudo rm -rf /var/root/.jenkins ~/.jenkins
sudo launchctl unload /Library/LaunchDaemons/org.jenkins-
ci.plist
sudo rm /Library/LaunchDaemons/org.jenkins-ci.plist
sudo rm -rf /Applications/Jenkins "/Library/Application
Support/Jenkins" /Library/Documentation/Jenkins

sudo rm -rf /Users/Shared/Jenkins
sudo dscl . -delete /Users/jenkins
sudo dscl . -delete /Groups/jenkins
sudo rm -f /etc/newsyslog.d/jenkins.conf
pkgutil --pkgs | grep 'org\.jenkins-ci\.' | xargs -n 1 sudo
pkgutil --forget
```

由于我的Mac 模式是 JDK 11，所以需要制定 JAVA_HOME 到 JDK 1.8，否则提示

```
Dec 27, 2018 9:20:33 AM Main main
SEVERE: Running with Java class version 55.0, but 52.0 is
```

```
required.Run with the --enable-future-java flag to enable such
behavior. See https://jenkins.io/redirect/java-support/
java.lang.UnsupportedClassVersionError: 55.0
        at Main.main(Main.java:139)

Jenkins requires Java 8, but you are running 11+28 from
/Library/Java/JavaVirtualMachines/jdk-11.jdk/Contents/Home
java.lang.UnsupportedClassVersionError: 55.0
        at Main.main(Main.java:139)
```

## 1.3. CentOS

```
wget -O /etc/yum.repos.d/jenkins.repo
https://pkg.jenkins.io/redhat-stable/jenkins.repo
rpm --import https://pkg.jenkins.io/redhat-
stable/jenkins.io.key

yum install -y jenkins
```

cat /etc/sysconfig/jenkins

```
## Path:        Development/Jenkins
## Description: Jenkins Automation Server
## Type:        string
## Default:     "/var/lib/jenkins"
## ServiceRestart: jenkins
#
# Directory where Jenkins store its configuration and working
# files (checkouts, build reports, artifacts, ...).
#
JENKINS_HOME="/var/lib/jenkins"

## Type:        string
## Default:     ""
## ServiceRestart: jenkins
```

```
#
# Java executable to run Jenkins
# When left empty, we'll try to find the suitable Java.
#
JENKINS_JAVA_CMD=""

## Type:        string
## Default:     "jenkins"
## ServiceRestart: jenkins
#
# Unix user account that runs the Jenkins daemon
# Be careful when you change this, as you need to update
# permissions of $JENKINS_HOME and /var/log/jenkins.
#
JENKINS_USER="jenkins"

## Type:        string
## Default: "false"
## ServiceRestart: jenkins
#
# Whether to skip potentially long-running chown at the
# $JENKINS_HOME location. Do not enable this, "true", unless
# you know what you're doing. See JENKINS-23273.
#
#JENKINS_INSTALL_SKIP_CHOWN="false"

## Type: string
## Default:     "-Djava.awt.headless=true"
## ServiceRestart: jenkins
#
# Options to pass to java when running Jenkins.
#
JENKINS_JAVA_OPTIONS="-Djava.awt.headless=true"

## Type:        integer(0:65535)
## Default:     8080
## ServiceRestart: jenkins
#
# Port Jenkins is listening on.
# Set to -1 to disable
#
JENKINS_PORT="8080"

## Type:        string
## Default:     ""
```

```
## ServiceRestart: jenkins
#
# IP address Jenkins listens on for HTTP requests.
# Default is all interfaces (0.0.0.0).
#
JENKINS_LISTEN_ADDRESS=""

## Type:        integer(0:65535)
## Default:     ""
## ServiceRestart: jenkins
#
# HTTPS port Jenkins is listening on.
# Default is disabled.
#
JENKINS_HTTPS_PORT=""

## Type:        string
## Default:     ""
## ServiceRestart: jenkins
#
# Path to the keystore in JKS format (as created by the JDK
'keytool').
# Default is disabled.
#
JENKINS_HTTPS_KEYSTORE=""

## Type:        string
## Default:     ""
## ServiceRestart: jenkins
#
# Password to access the keystore defined in
JENKINS_HTTPS_KEYSTORE.
# Default is disabled.
#
JENKINS_HTTPS_KEYSTORE_PASSWORD=""

## Type:        string
## Default:     ""
## ServiceRestart: jenkins
#
# IP address Jenkins listens on for HTTPS requests.
# Default is disabled.
#
JENKINS_HTTPS_LISTEN_ADDRESS=""
```

```
## Type:        integer(1:9)
## Default:     5
## ServiceRestart: jenkins
#
# Debug level for logs -- the higher the value, the more
verbose.
# 5 is INFO.
#
JENKINS_DEBUG_LEVEL="5"

## Type:        yesno
## Default:     no
## ServiceRestart: jenkins
#
# Whether to enable access logging or not.
#
JENKINS_ENABLE_ACCESS_LOG="no"

## Type:        integer
## Default:     100
## ServiceRestart: jenkins
#
# Maximum number of HTTP worker threads.
#
JENKINS_HANDLER_MAX="100"

## Type:        integer
## Default:     20
## ServiceRestart: jenkins
#
# Maximum number of idle HTTP worker threads.
#
JENKINS_HANDLER_IDLE="20"

## Type:        string
## Default:     ""
## ServiceRestart: jenkins
#
# Pass arbitrary arguments to Jenkins.
# Full option list: java -jar jenkins.war --help
#
JENKINS_ARGS=""
```

```

```

Nginx 配置

```
[root@netkiller ~]# cat /etc/nginx/conf.d/jk.netkiller.cn.conf
server {
    listen       80;
    server_name  jk.netkiller.cn;

    charset utf-8;

    location / {
        proxy_pass   http://127.0.0.1:8080;
    }

    #error_page  404              /404.html;

    # redirect server error pages to the static page /50x.html
    #
    error_page   500 502 503 504  /50x.html;
    location = /50x.html {
        root   /usr/share/nginx/html;
    }
}
```

查看管理员密码

```
cat /var/lib/jenkins/secrets/initialAdminPassword
```

## 1.4. Ubuntu

```
wget -q -O - https://pkg.jenkins.io/debian-
```

```
stable/jenkins.io.key | sudo apt-key add -
deb https://pkg.jenkins.io/debian-stable binary/

sudo apt-get update
sudo apt-get install jenkins
```

## 1.5. Docker

https://github.com/jenkinsci/docker/blob/master/README.md

8080端口是jenkins的端口，5000端口是master和slave通信端口

```
docker pull jenkins/jenkins:lts
docker run -p 8080:8080 -p 50000:50000 --name jenkins
jenkins/jenkins:lts
```

首次启动，不要使用 -d 参数，如果使用了 -d 参数可以通过docker logs -f jenkins查看控制台的密码

docker-compose 配置文件

```
version: '2'

services:
  jenkins:
    container_name: jenkins-lts
    ports: # 端口映射，9001为宿主机上的端口，相应的8080是容器运行起来时候jenkins服务的端口
      - 9001:8080
      - 50000:50000
    image: jenkins/jenkins:lts # 指定运行用哪一个镜像来运行容器
    volumes:
      - /home/jenkins/jenkins_home:/var/jenkins_home # 挂载指令，目的在于销毁容器时，并不影响jenkins数据
```

## 1.6. Minikube

创建 jenkins-namespace.yaml

```
apiVersion: v1
kind: Namespace
metadata:
  name: jenkins-project
```

创建命名空间

```
$ kubectl create -f jenkins-namespace.yaml
```

创建 jenkins-volume.yaml

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: jenkins-pv
  namespace: jenkins-project
spec:
  storageClassName: jenkins-pv
  accessModes:
    - ReadWriteOnce
  capacity:
    storage: 20Gi
  persistentVolumeReclaimPolicy: Retain
  hostPath:
    path: /opt/jenkins-volume/
```

创建卷

```
$ kubectl create -f jenkins-volume.yaml
persistentvolume “jenkins-pv” created
```

创建 jenkins-values.yaml 文件

```
# Default values for jenkins.
# This is a YAML-formatted file.
# Declare name/value pairs to be passed into your templates.
# name: value

## Overrides for generated resource names
# See templates/_helpers.tpl
# nameOverride:
# fullnameOverride:

Master:
  Name: jenkins-master
  Image: "jenkins/jenkins"
  ImageTag: "2.141"
  ImagePullPolicy: "Always"
  Component: "jenkins-master"
  UseSecurity: true
  AdminUser: admin
  # AdminPassword: <defaults to random>
  Cpu: "200m"
  Memory: "256Mi"
  ServicePort: 8080
  # For minikube, set this to NodePort, elsewhere use
LoadBalancer
  # <to set explicitly, choose port between 30000-32767>
  ServiceType: NodePort
  NodePort: 32000
  ServiceAnnotations: {}
```

```
  ContainerPort: 8080
  # Enable Kubernetes Liveness and Readiness Probes
  HealthProbes: true
  HealthProbesTimeout: 60
  SlaveListenerPort: 50000
  LoadBalancerSourceRanges:
    - 0.0.0.0/0
  # List of plugins to be install during Jenkins master start
  InstallPlugins:
    - kubernetes:1.12.4
    - workflow-aggregator:2.5
    - workflow-job:2.24
    - credentials-binding:1.16
    - git:3.9.1
    - greenballs:1.15
  # Used to approve a list of groovy functions in pipelines
used the script-security plugin. Can be viewed under
/scriptApproval
  ScriptApproval:
    - "method groovy.json.JsonSlurperClassic parseText
java.lang.String"
    - "new groovy.json.JsonSlurperClassic"
    - "staticMethod
org.codehaus.groovy.runtime.DefaultGroovyMethods leftShift
java.util.Map java.util.Map"
    - "staticMethod
org.codehaus.groovy.runtime.DefaultGroovyMethods split
java.lang.String"
  CustomConfigMap: false
  NodeSelector: {}
  Tolerations: {}

Agent:
  Enabled: true
  Image: jenkins/jnlp-slave
  ImageTag: 3.10-1
  Component: "jenkins-slave"
  Privileged: false
  Cpu: "200m"
  Memory: "256Mi"
  # You may want to change this to true while testing a new
image
  AlwaysPullImage: false
  # You can define the volumes that you want to mount for this
container
```

```
  # Allowed types are: ConfigMap, EmptyDir, HostPath, Nfs, Pod,
Secret
  volumes:
    - type: HostPath
      hostPath: /var/run/docker.sock
      mountPath: /var/run/docker.sock
  NodeSelector: {}

Persistence:
  Enabled: true
  ## A manually managed Persistent Volume and Claim
  ## Requires Persistence.Enabled: true
  ## If defined, PVC must be created manually before volume
will be bound
  # ExistingClaim:
  ## jenkins data Persistent Volume Storage Class
  StorageClass: jenkins-pv

  Annotations: {}
  AccessMode: ReadWriteOnce
  Size: 20Gi
  volumes:
  #  - name: nothing
  #    emptyDir: {}
  mounts:
  #  - mountPath: /var/nothing
  #    name: nothing
  #    readOnly: true

NetworkPolicy:
  # Enable creation of NetworkPolicy resources.
  Enabled: false
  # For Kubernetes v1.4, v1.5 and v1.6, use
'extensions/v1beta1'
  # For Kubernetes v1.7, use 'networking.k8s.io/v1'
  ApiVersion: networking.k8s.io/v1

## Install Default RBAC roles and bindings
rbac:
  install: true
  serviceAccountName: default
  # RBAC api version (currently either v1beta1 or v1alpha1)
  apiVersion: v1beta1
  # Cluster role reference
  roleRef: cluster-admin
```

使用 helm 安装 jenkins

```
$ cd ~/minikube-helm-jenkins
$ helm init
$ helm install --name jenkins -f helm/jenkins-values.yaml
stable/jenkins --namespace jenkins-project
```

查看 jenkins 密码

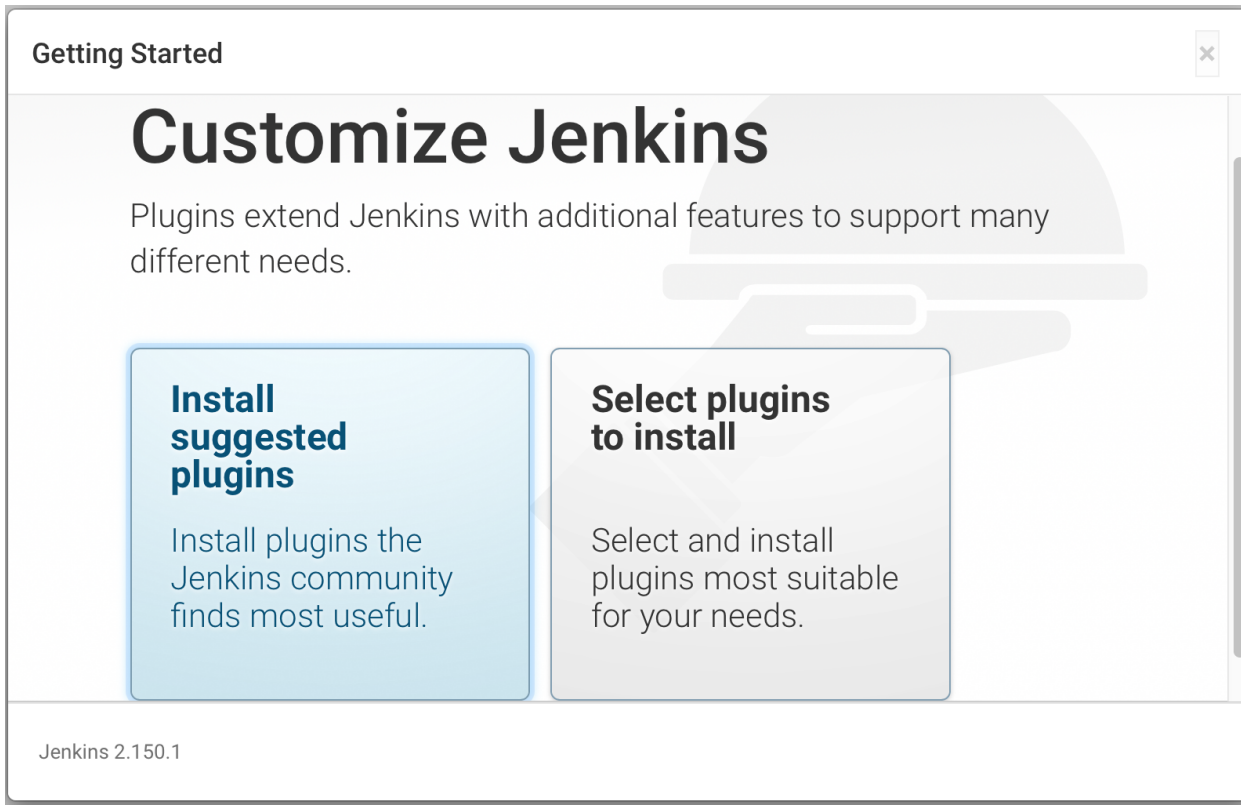
```
$ printf $(kubectl get secret --namespace jenkins-project
jenkins -o jsonpath="{.data.jenkins-admin-password}" | base64 -
-decode);echo
```

## 2. 配置 Jenkins

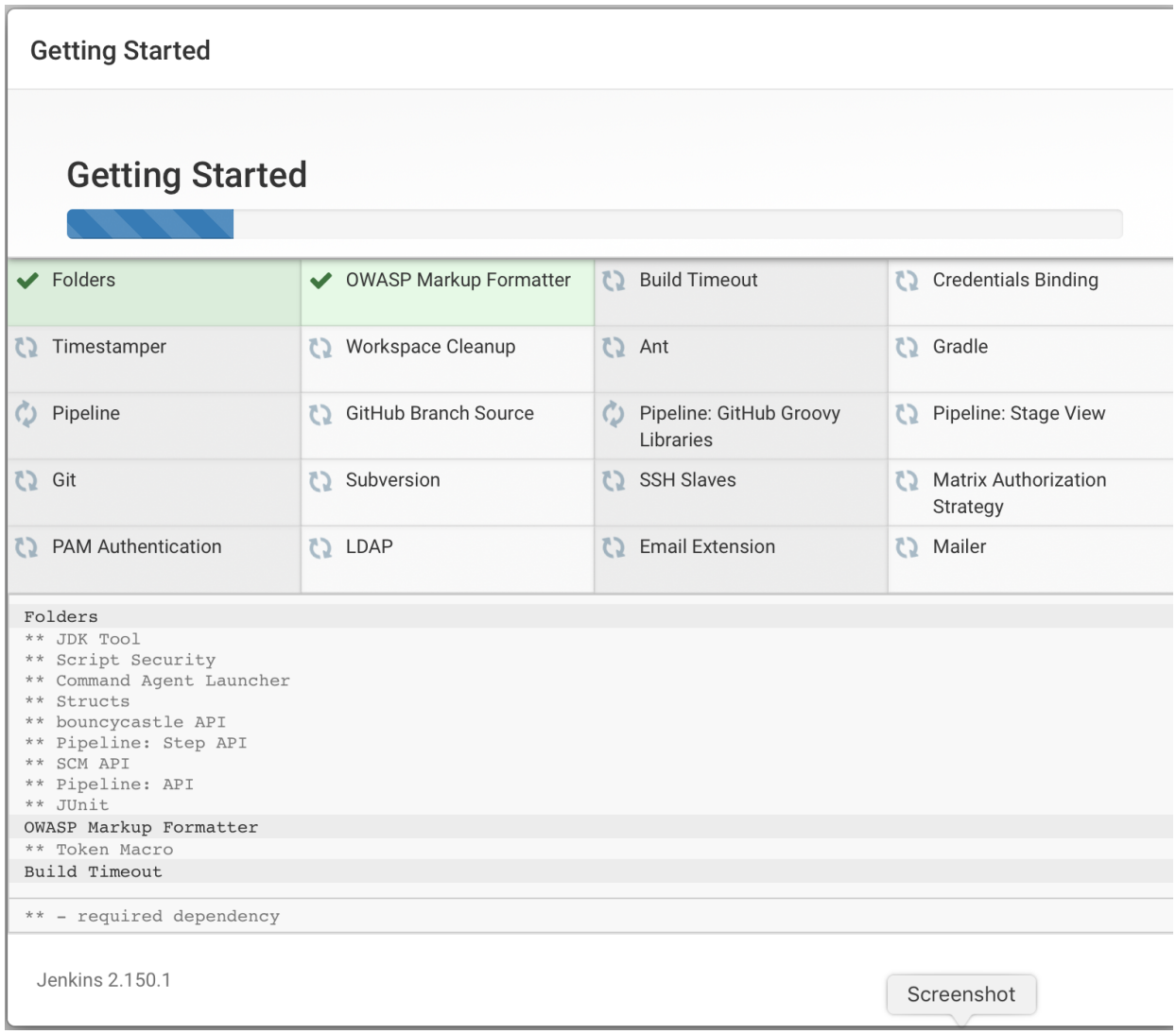
配置 Jenkins



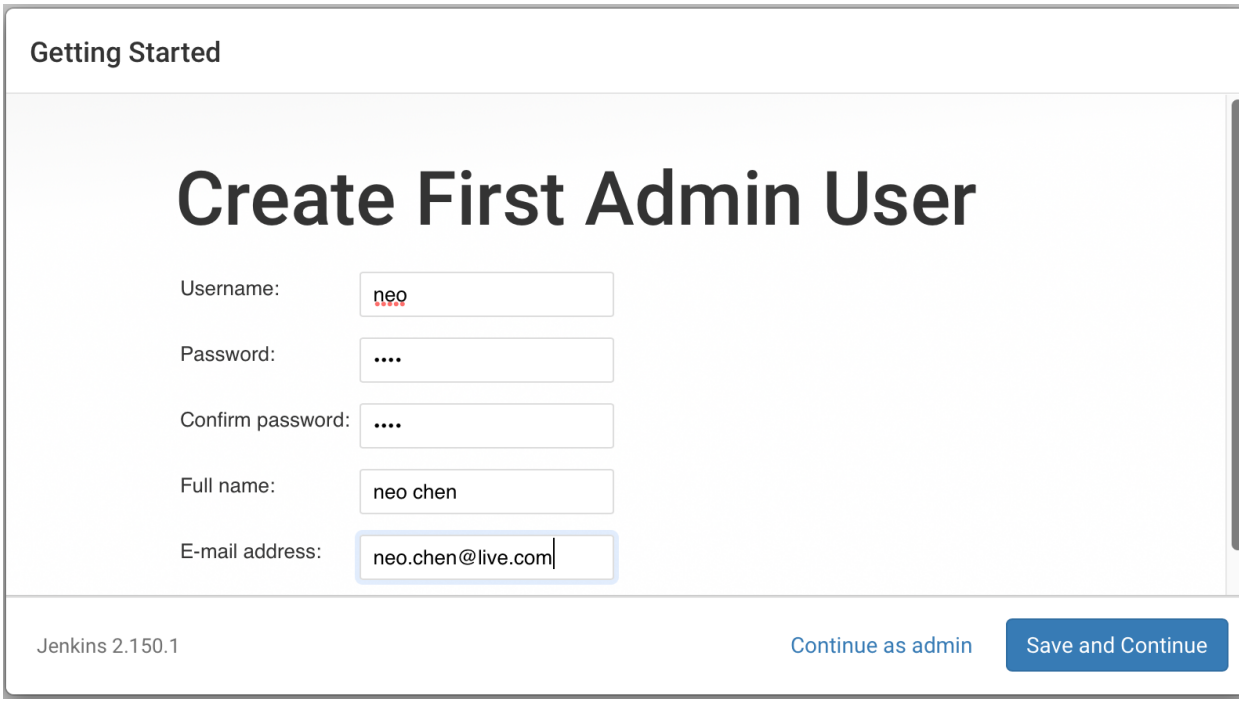

输入管理员密码


Getting Started
Customize Jenkins
Plugins extend Jenkins with additional features to support many different needs.
Install suggested plugins
Install plugins the Jenkins community finds most useful.
Select plugins to install
Select and install plugins most suitable for your needs.
Jenkins 2.150.1

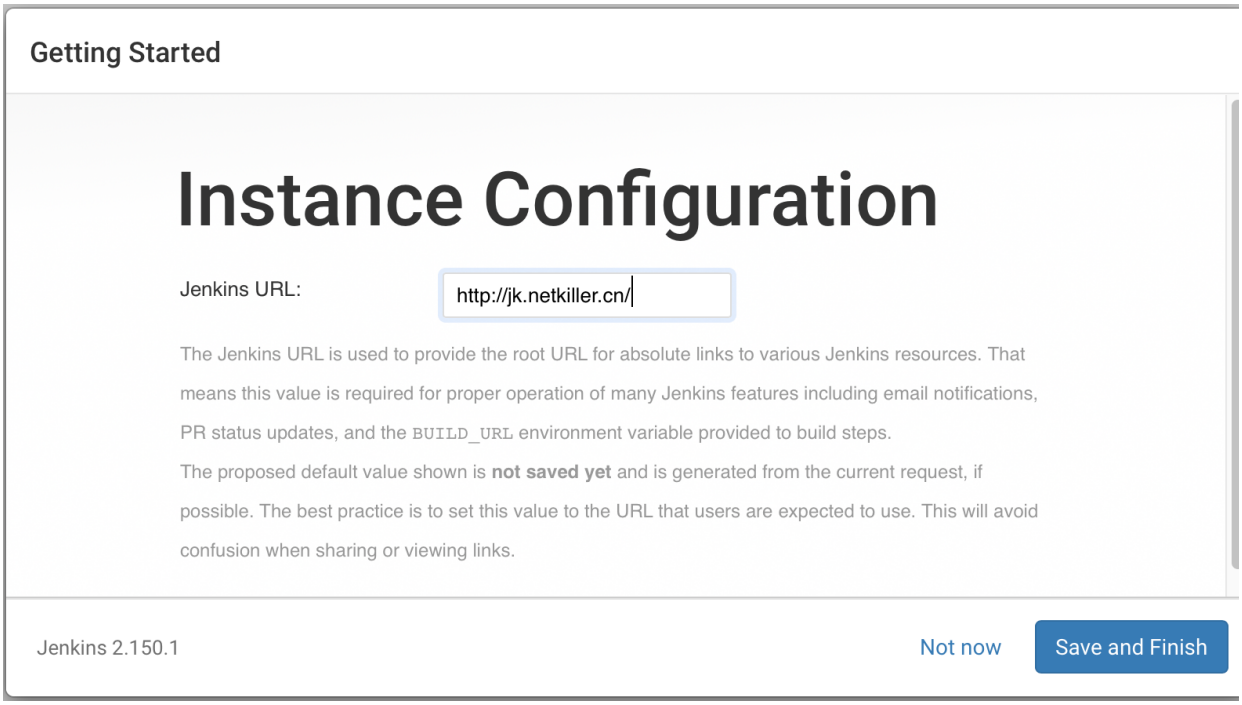
安装插件


Getting Started
Getting Started
Folders
OWASP Markup Formatter
Build Timeout
Credentials Binding
Timestamper
Workspace Cleanup
Ant
Gradle
Pipeline
GitHub Branch Source
Pipeline: GitHub Groovy Libraries
Pipeline: Stage View
Git
Subversion
SSH Slaves
Matrix Authorization Strategy
PAM Authentication
LDAP
Email Extension
Mailer
Folders
** JDK Tool
** Script Security
** Command Agent Launcher
** Structs
** bouncycastle API
** Pipeline: Step API
** SCM API
** Pipeline: API
** JUnit
OWASP Markup Formatter
** Token Macro
Build Timeout
** - required dependency
Jenkins 2.150.1
Screenshot

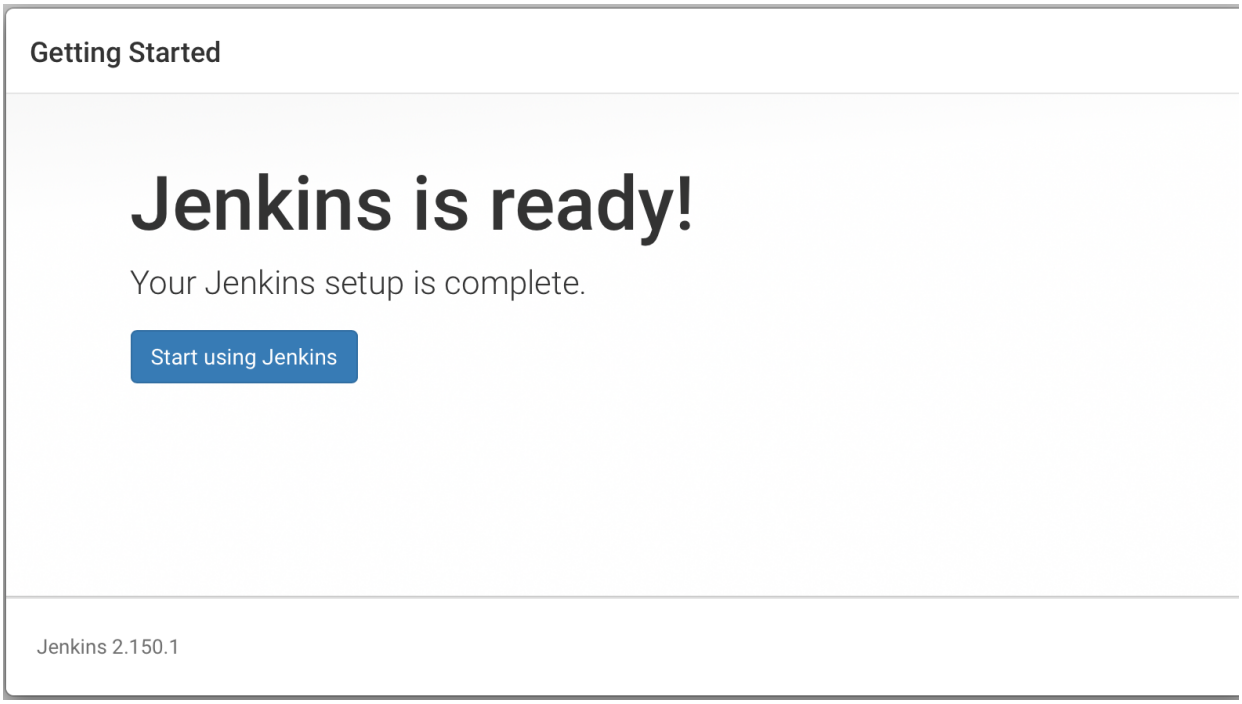
创建用户



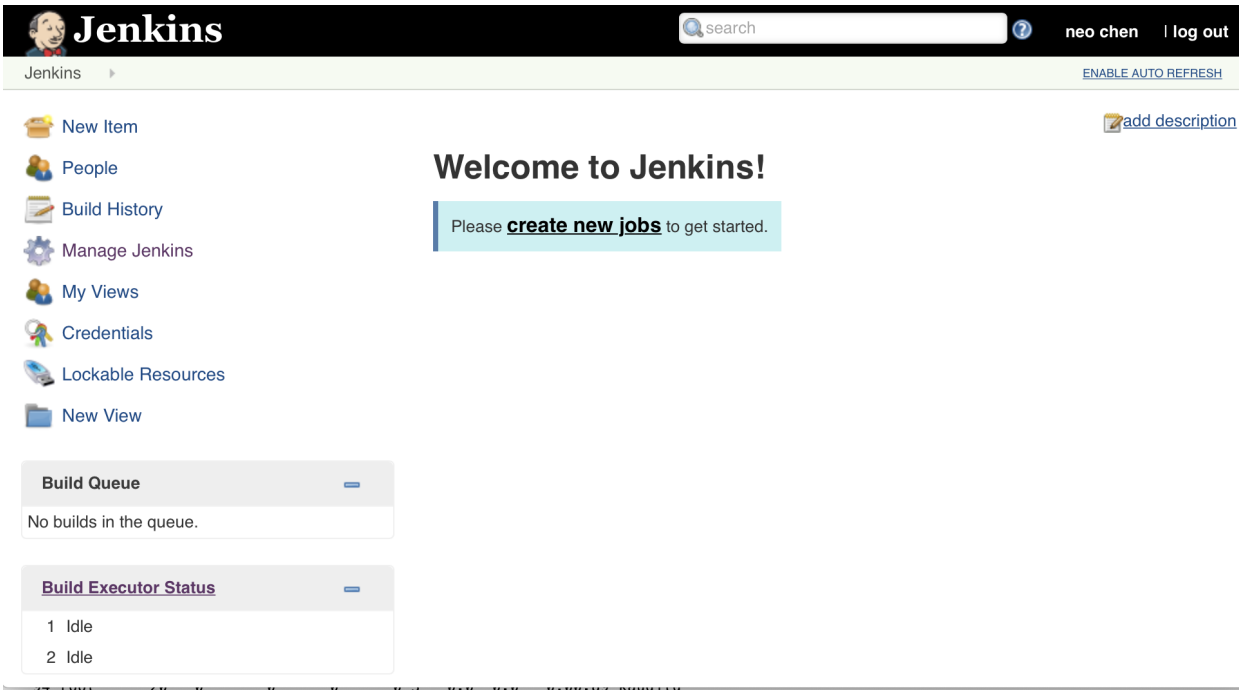

设置域名




开始使用 Jenkins


Getting Started
Jenkins is ready!
Your Jenkins setup is complete.
Start using Jenkins
Jenkins 2.150.1

Jenkins 界面


Jenkins
search
neo chen
log out
Jenkins
ENABLE AUTO REFRESH
New Item
People
Build History
Manage Jenkins
My Views
Credentials
Lockable Resources
New View
add description
Welcome to Jenkins!
Please create new jobs to get started.
Build Queue
No builds in the queue.
Build Executor Status
1 Idle
2 Idle

# 3. Jenkinsfile

## 3.1. Jenkinsfile - Declarative Pipeline

https://jenkins.io/doc/pipeline/examples/

### stages

```
pipeline {
    agent any

    stages {
        stage('Build') {
            steps {
                echo 'Building..'
            }
        }
        stage('Test') {
            steps {
                echo 'Testing..'
            }
        }
        stage('Deploy') {
            steps {
                echo 'Deploying....'
            }
        }
    }
}
```

### script

```
// Declarative //
```

```
pipeline {
    agent any
    stages {
        stage('Example') {
            steps {
                echo 'Hello World'
                script {
                    def browsers = ['chrome', 'firefox']
                    for (int i = 0; i < browsers.size(); ++i) {
                        echo "Testing the ${browsers[i]}
browser"
                    }
                }
                script {
                    // 一个优雅的退出pipeline的方法，这里可执行任意逻
辑
                    if( $VALUE1 == $VALUE2 ) {
                       currentBuild.result = 'SUCCESS'
                       return
                    }
                }
            }
        }
    }
}
```

## junit

junit4

```
                stage("测试") {
            steps {
                echo "单元测试中..."
                // 请在这里放置您项目代码的单元测试调用过程，例如：
                sh 'mvn test' // mvn 示例
                // sh './gradlew test'
                echo "单元测试完成."
                junit 'target/surefire-reports/*.xml' // 收集单
元测试报告的调用过程
```

```
            }
        }
```

junit5 测试报告路径与 junit4 的位置不同

```
                stage("测试") {
            steps {
                echo "单元测试中..."
                sh './gradlew test'
                echo "单元测试完成."
                junit 'build/test-results/test/*.xml'
            }
        }
```

**withEnv**

```
env.PROJECT_DIR='src/netkiller'
node {
    withEnv(["GOPATH=$WORKSPACE"]) {
        stage('Init gopath') {
            sh 'mkdir -p $GOPATH/{bin,pkg,src}'
        }

        stage('Build go proejct') {
            sh 'cd ${PROJECT_DIR}; go test && go build && go
install'
        }
    }
}
```

```
node {
```

```
     git 'https://github.com/netkiller/api.git'
     withEnv(["PATH+MAVEN=${tool 'm3'}/bin"]) {
          sh "mvn -B —Dmaven.test.failure.ignore=true clean
package"
     }
     stash excludes: 'target/', includes: '**', name: 'source'
}
```

**parameters**

参数指令，触发这个管道需要用户指定的参数，然后在step中通过params对象访问这些参数。

```
// Declarative //
pipeline {
    agent any
    parameters {
        string(name: 'PERSON', defaultValue: 'Mr Jenkins',
description: 'Who should I say hello to?')
    }
    stages {
        stage('Example') {
            steps {
                echo "Hello ${params.PERSON}"
            }
        }
    }
}
```

```
Jenkinsfile (Declarative Pipeline)
pipeline {
    agent any
    parameters {
        string(name: 'PERSON', defaultValue: 'Mr Jenkins',
description: 'Who should I say hello to?')
```

```
        text(name: 'BIOGRAPHY', defaultValue: '', description:
'Enter some information about the person')

        booleanParam(name: 'TOGGLE', defaultValue: true,
description: 'Toggle this value')

        choice(name: 'CHOICE', choices: ['One', 'Two',
'Three'], description: 'Pick something')

        password(name: 'PASSWORD', defaultValue: 'SECRET',
description: 'Enter a password')

        file(name: "FILE", description: "Choose a file to
upload")
    }
    stages {
        stage('Example') {
            steps {
                echo "Hello ${params.PERSON}"

                echo "Biography: ${params.BIOGRAPHY}"

                echo "Toggle: ${params.TOGGLE}"

                echo "Choice: ${params.CHOICE}"

                echo "Password: ${params.PASSWORD}"
            }
        }
    }
}
```

**options**

还能定义一些管道特定的选项，介绍几个常用的：

```
skipDefaultCheckout - 在agent指令中忽略源码checkout这一步骤。
timeout - 超时设置options { timeout(time: 1, unit: 'HOURS') }
```

```
retry - 直到成功的重试次数options { retry(3) }
timestamps - 控制台输出前面加时间戳options { timestamps() }
```

**triggers**

触发器指令定义了这个管道何时该执行，就可以定义两种cron和pollSCM

```
cron - linux的cron格式triggers { cron('H 4/* 0 0 1-5') }
pollSCM - jenkins的poll scm语法，比如triggers { pollSCM('H 4/* 0 0 1-5') }

// Declarative //
pipeline {
    agent any
    triggers {
        cron('H 4/* 0 0 1-5')
    }
    stages {
        stage('Example') {
            steps {
                echo 'Hello World'
            }
        }
    }
}
```

一般我们会将管道和GitHub、GitLab、BitBucket关联， 然后使用它们的webhooks来触发，就不需要这个指令了。

**tools**

定义自动安装并自动放入PATH里面的工具集合

```
// Declarative //
pipeline {
    agent any
    tools {
        maven 'apache-maven-3.5.0' ①
    }
    stages {
        stage('Example') {
            steps {
                sh 'mvn --version'
            }
        }
    }
}
```

注：① 工具名称必须预先在Jenkins中配置好了 → Global Tool Configuration.

**post**

post section 定义了管道执行结束后要进行的操作。支持在里面定义很多Conditions块： always, changed, failure, success 和 unstable。 这些条件块会根据不同的返回结果来执行不同的逻辑。

```
always：不管返回什么状态都会执行
changed：如果当前管道返回值和上一次已经完成的管道返回值不同时候执行
failure：当前管道返回状态值为"failed"时候执行，在Web UI界面上面是红色的标志
success：当前管道返回状态值为"success"时候执行，在Web UI界面上面是绿色的标志
unstable：当前管道返回状态值为"unstable"时候执行，通常因为测试失败，代码不合法引起的。在Web UI界面上面是黄色的标志

// Declarative //
pipeline {
    agent any
```

```
    stages {
        stage('Example') {
            steps {
                echo 'Hello World'
            }
        }
    }
    post {
        always {
            echo 'I will always say Hello again!'
        }
    }
}
```

失败发送邮件的例子

```
    post {
        failure {
            mail to: "${email}",
            subject: "Failed Pipeline:
${currentBuild.fullDisplayName}",
            body: "Something is wrong with ${env.BUILD_URL}"
        }
    }
```

**when 条件判断**

```
branch - 分支匹配才执行 when { branch 'master' }
environment - 环境变量匹配才执行 when { environment name:
'DEPLOY_TO', value: 'production' }
expression - groovy表达式为真才执行 expression { return
params.DEBUG_BUILD } }

// Declarative //
pipeline {
```

```
    agent any
    stages {
        stage('Example Build') {
            steps {
                echo 'Hello World'
            }
        }
        stage('Example Deploy') {
            when {
                branch 'production'
            }
            echo 'Deploying'
        }
    }
}
```

抛出错误

```
error '执行出错'
```

## withCredentials

#### withCredentials: Bind credentials to variables

#### token

```
node {
   withCredentials([string(credentialsId: 'token', variable:
'TOKEN')]) {
        sh('echo $TOKEN')
   }
}
```

**withMaven**

```
            withMaven(
        maven: 'M3') {
            sh "mvn test"
    }
```

**isUnix() 判断操作系统类型**

```
pipeline{

        agent any
        stages{
                stage("isUnix") {
                        steps{
                                script {
                                        if(isUnix() == true) {
                                                echo("this
jenkins job running on a linux-like system")
                                        }else {
                                                error("the
jenkins job running on a windows system")
                                        }
                                }
                        }
                }
        }
}
```

**Jenkins pipeline 中使用 sshpass 实现 scp, ssh 远程运行**

```
pipeline {
```

```
    agent {
        label "java-8"
    }
    stages  {

        stage("环境") {
            steps {
                parallel "Maven": {
                        script{
                        sh 'mvn -version'
                        }
                }, "Java": {
                    sh 'java -version'
                }, "sshpass": {
                        sh 'apt install -y sshpass'
                    sh 'sshpass -v'
                }
            }

        }

        stage("检出") {
            steps {
                checkout(
                  [$class: 'GitSCM', branches: [[name:
env.GIT_BUILD_REF]],
                  userRemoteConfigs: [[url: env.GIT_REPO_URL]]]
                )
            }
        }

        stage("构建") {
            steps {
                echo "构建中..."
                sh 'mvn package -Dmaven.test.skip=true'
                archiveArtifacts artifacts: '**/target/*.jar',
fingerprint: true
                echo "构建完成."
            }
        }

        stage("测试") {
            steps {
                parallel "单元测试": {
                    echo "单元测试中..."
```

```
                    sh 'mvn test'
                    echo "单元测试完成."
                    junit 'target/surefire-reports/*.xml'
                }, "接口测试": {
                    echo "接口测试中..."
                    // 请在这里放置您项目代码的单元测试调用过程，例如
mvn test
                    echo "接口测试完成."
                }, "测试敏感词":{
                    echo "Username: ${env.username}"
                        echo "Password: ${env.password}"
                }

            }

        }
        stage("运行"){
                steps {
                sh 'java -jar target/java-0.0.1-SNAPSHOT.jar'
            }
        }
        stage("部署"){
                steps {
              echo "上传"
              sh 'sshpass -p Passw0rd scp target/*.jar
root@dev.netkiller.cn:/root/'
              echo "运行"
              sh 'sshpass -p Passw0rd ssh root@dev.netkiller.cn
java -jar /root/java-0.0.1-SNAPSHOT.jar'

            }
        }
    }
}
```

后台运行

```
                stage("部署"){
            parallel{
                    stage("sshpass") {
```

```
                                steps{
                                    sh 'apt install -y sshpass'
                                sh 'sshpass -v'
                                }
                            }
                stage('stop') {
                                 steps {
                                    sh 'sshpass -p passw0rd ssh
-f dev.netkiller.cn pkill -f java-project-0.0.2-SNAPSHOT'
                                }
                            }
                            stage('start') {
                                 steps {
                                    sh 'sshpass -p passw0rd scp
target/*.jar dev.netkiller.cn:/root/'
                            sh 'sshpass -p passw0rd ssh -f
dev.netkiller.cn java -jar /root/java-project-0.0.2-
SNAPSHOT.jar'
                                 }
                            }
            }
        }
```

## 3.2. Jenkinsfile - Scripted Pipeline

```
// Jenkinsfile (Scripted Pipeline)
node {
    stage('Build') {
        echo 'Building....'
    }
    stage('Test') {
        echo 'Building....'
    }
    stage('Deploy') {
        echo 'Deploying....'
    }
}
```

**git**

```
node {

   stage('Checkout') {
      git 'https://github.com/bg7nyt/java.git'
   }

}
```

**切换 JDK 版本**

```
node('vagrant-slave') {
    env.JAVA_HOME="${tool 'jdk-8u45'}"
    env.PATH="${env.JAVA_HOME}/bin:${env.PATH}"
    sh 'java -version'
}
```

**groovy**

```
#!groovy
import groovy.json.JsonOutput
import groovy.json.JsonSlurper

/*
Please make sure to add the following environment variables:
HEROKU_PREVIEW=<your heroku preview app>
HEROKU_PREPRODUCTION=<your heroku pre-production app>
HEROKU_PRODUCTION=<your heroku production app>
Please also add the following credentials to the global domain
of your organization's folder:
Heroku API key as secret text with ID 'HEROKU_API_KEY'
```

```
GitHub Token value as secret text with ID 'GITHUB_TOKEN'
*/

node {

     server = Artifactory.server "artifactory"
     buildInfo = Artifactory.newBuildInfo()
     buildInfo.env.capture = true

    // we need to set a newer JVM for Sonar
    env.JAVA_HOME="${tool 'Java SE DK 8u131'}"
    env.PATH="${env.JAVA_HOME}/bin:${env.PATH}"

    // pull request or feature branch
    if  (env.BRANCH_NAME != 'master') {
        checkout()
        build()
        unitTest()
        // test whether this is a regular branch build or a
merged PR build
        if (!isPRMergeBuild()) {
            preview()
            sonarServer()
            allCodeQualityTests()
        } else {
            // Pull request
            sonarPreview()
        }
    } // master branch / production
    else {
        checkout()
        build()
        allTests()
        preview()
        sonarServer()
        allCodeQualityTests()
        preProduction()
        manualPromotion()
        production()
    }
}

def isPRMergeBuild() {
    return (env.BRANCH_NAME ==~ /^PR-\d+$/)
}
```

```
def sonarPreview() {
    stage('SonarQube Preview') {
        prNo = (env.BRANCH_NAME=~/^PR-(\d+)$/)[0][1]
        mvn "org.jacoco:jacoco-maven-plugin:prepare-agent
install -Dmaven.test.failure.ignore=true -Pcoverage-per-test"
        withCredentials([[$class: 'StringBinding',
credentialsId: 'GITHUB_TOKEN', variable: 'GITHUB_TOKEN']]) {
            githubToken=env.GITHUB_TOKEN
            repoSlug=getRepoSlug()
            withSonarQubeEnv('SonarQube Octodemoapps') {
                mvn "-Dsonar.analysis.mode=preview -
Dsonar.github.pullRequest=${prNo} -
Dsonar.github.oauth=${githubToken} -
Dsonar.github.repository=${repoSlug} -
Dsonar.github.endpoint=https://api.github.com/
org.sonarsource.scanner.maven:sonar-maven-plugin:3.2:sonar"
            }
        }
    }
}

def sonarServer() {
    stage('SonarQube Server') {
        mvn "org.jacoco:jacoco-maven-plugin:prepare-agent
install -Dmaven.test.failure.ignore=true -Pcoverage-per-test"
        withSonarQubeEnv('SonarQube Octodemoapps') {
            mvn "org.sonarsource.scanner.maven:sonar-maven-
plugin:3.2:sonar"
        }

        context="sonarqube/qualitygate"
        setBuildStatus ("${context}", 'Checking Sonarqube
quality gate', 'PENDING')
        timeout(time: 1, unit: 'MINUTES') { // Just in case
something goes wrong, pipeline will be killed after a timeout
            def qg = waitForQualityGate() // Reuse taskId
previously collected by withSonarQubeEnv
            if (qg.status != 'OK') {
                setBuildStatus ("${context}", "Sonarqube
quality gate fail: ${qg.status}", 'FAILURE')
                error "Pipeline aborted due to quality gate
failure: ${qg.status}"
            } else {
                setBuildStatus ("${context}", "Sonarqube
```

```
quality gate pass: ${qg.status}", 'SUCCESS')
            }
        }
    }
}



def checkout () {
    stage 'Checkout code'
    context="continuous-integration/jenkins/"
    context += isPRMergeBuild()?"pr-
merge/checkout":"branch/checkout"
    checkout scm
    setBuildStatus ("${context}", 'Checking out completed',
'SUCCESS')
}

def build () {
    stage 'Build'
    mvn 'clean install -DskipTests=true -
Dmaven.javadoc.skip=true -Dcheckstyle.skip=true -B -V'
}


def unitTest() {
    stage 'Unit tests'
    mvn 'test -B -Dmaven.javadoc.skip=true -
Dcheckstyle.skip=true'
    if (currentBuild.result == "UNSTABLE") {
        sh "exit 1"
    }
}

def allTests() {
    stage 'All tests'
    // don't skip anything
    mvn 'test -B'
    step([$class: 'JUnitResultArchiver', testResults:
'**/target/surefire-reports/TEST-*.xml'])
    if (currentBuild.result == "UNSTABLE") {
        // input "Unit tests are failing, proceed?"
        sh "exit 1"
    }
```

```
}

def allCodeQualityTests() {
    stage 'Code Quality'
    lintTest()
    coverageTest()
}

def lintTest() {
    context="continuous-integration/jenkins/linting"
    setBuildStatus ("${context}", 'Checking code conventions',
'PENDING')
    lintTestPass = true

    try {
        mvn 'verify -DskipTests=true'
    } catch (err) {
        setBuildStatus ("${context}", 'Some code conventions
are broken', 'FAILURE')
        lintTestPass = false
    } finally {
        if (lintTestPass) setBuildStatus ("${context}", 'Code
conventions OK', 'SUCCESS')
    }
}

def coverageTest() {
    context="continuous-integration/jenkins/coverage"
    setBuildStatus ("${context}", 'Checking code coverage
levels', 'PENDING')

    coverageTestStatus = true

    try {
        mvn 'cobertura:check'
    } catch (err) {
        setBuildStatus("${context}", 'Code coverage below 90%',
'FAILURE')
        throw err
    }

    setBuildStatus ("${context}", 'Code coverage above 90%',
'SUCCESS')

}
```

```
def preview() {
    stage name: 'Deploy to Preview env', concurrency: 1
    def herokuApp = "${env.HEROKU_PREVIEW}"
    def id = createDeployment(getBranch(), "preview",
"Deploying branch to test")
    echo "Deployment ID: ${id}"
    if (id != null) {
        setDeploymentStatus(id, "pending",
"https://${herokuApp}.herokuapp.com/", "Pending deployment to
test");
        herokuDeploy "${herokuApp}"
        setDeploymentStatus(id, "success",
"https://${herokuApp}.herokuapp.com/", "Successfully deployed
to test");
    }
    mvn 'deploy -DskipTests=true'
}

def preProduction() {
    stage name: 'Deploy to Pre-Production', concurrency: 1
    switchSnapshotBuildToRelease()
    herokuDeploy "${env.HEROKU_PREPRODUCTION}"
    buildAndPublishToArtifactory()
}

def manualPromotion() {
    // we need a first milestone step so that all jobs entering
this stage are tracked an can be aborted if needed
    milestone 1
    // time out manual approval after ten minutes
    timeout(time: 10, unit: 'MINUTES') {
        input message: "Does Pre-Production look good?"
    }
    // this will kill any job which is still in the input step
    milestone 2
}

def production() {
    stage name: 'Deploy to Production', concurrency: 1
    step([$class: 'ArtifactArchiver', artifacts:
'**/target/*.jar', fingerprint: true])
    herokuDeploy "${env.HEROKU_PRODUCTION}"
    def version =
getCurrentHerokuReleaseVersion("${env.HEROKU_PRODUCTION}")
```

```
    def createdAt =
getCurrentHerokuReleaseDate("${env.HEROKU_PRODUCTION}",
version)
    echo "Release version: ${version}"
    createRelease(version, createdAt)
    promoteInArtifactoryAndDistributeToBinTray()
}

def switchSnapshotBuildToRelease() {
    def descriptor = Artifactory.mavenDescriptor()
    descriptor.version = '1.0.0'
    descriptor.pomFile = 'pom.xml'
    descriptor.transform()
}

def buildAndPublishToArtifactory() {
        def rtMaven = Artifactory.newMavenBuild()
        rtMaven.tool = "Maven 3.x"
        rtMaven.deployer releaseRepo:'libs-release-local',
snapshotRepo:'libs-snapshot-local', server: server
        rtMaven.resolver releaseRepo:'libs-release',
snapshotRepo:'libs-snapshot', server: server
        rtMaven.run pom: 'pom.xml', goals: 'install',
buildInfo: buildInfo
        server.publishBuildInfo buildInfo
}

def promoteBuildInArtifactory() {
        def promotionConfig = [
            // Mandatory parameters
            'buildName'          : buildInfo.name,
            'buildNumber'        : buildInfo.number,
            'targetRepo'         : 'libs-prod-local',

            // Optional parameters
            'comment'            : 'deploying to production',
            'sourceRepo'         : 'libs-release-local',
            'status'             : 'Released',
            'includeDependencies': false,
            'copy'               : true,
            // 'failFast' is true by default.
            // Set it to false, if you don't want the promotion
to abort upon receiving the first error.
            'failFast'           : true
        ]
```

```
        // Promote build
        server.promote promotionConfig
}

def distributeBuildToBinTray() {
        def distributionConfig = [
            // Mandatory parameters
            'buildName'             : buildInfo.name,
            'buildNumber'           : buildInfo.number,
            'targetRepo'            : 'reading-time-dist',
            // Optional parameters
            //'publish'               : true, // Default: true.
If true, artifacts are published when deployed to Bintray.
            'overrideExistingFiles' : true, // Default: false.
If true, Artifactory overwrites builds already existing in the
target path in Bintray.
            //'gpgPassphrase'         : 'passphrase', // If
specified, Artifactory will GPG sign the build deployed to
Bintray and apply the specified passphrase.
            //'async'                 : false, // Default:
false. If true, the build will be distributed asynchronously.
Errors and warnings may be viewed in the Artifactory log.
            //"sourceRepos"           : ["yum-local"], // An
array of local repositories from which build artifacts should
be collected.
            //'dryRun'                : false, // Default:
false. If true, distribution is only simulated. No files are
actually moved.
        ]
        server.distribute distributionConfig
}

def promoteInArtifactoryAndDistributeToBinTray() {
    stage ("Promote in Artifactory and Distribute to BinTray")
{
        promoteBuildInArtifactory()
        distributeBuildToBinTray()
    }
}

def mvn(args) {
    withMaven(
        // Maven installation declared in the Jenkins "Global
Tool Configuration"
```

```
        maven: 'Maven 3.x',
        // Maven settings.xml file defined with the Jenkins
Config File Provider Plugin

        // settings.xml referencing the GitHub Artifactory
repositories
         mavenSettingsConfig: '0e94d6c3-b431-434f-a201-
7d7cda7180cb',
        // we do not need to set a special local maven repo,
take the one from the standard box
        //mavenLocalRepo: '.repository'
        ) {
        // Run the maven build
        sh "mvn $args -Dmaven.test.failure.ignore"
    }
}

def herokuDeploy (herokuApp) {
    withCredentials([[$class: 'StringBinding', credentialsId:
'HEROKU_API_KEY', variable: 'HEROKU_API_KEY']]) {
        mvn "heroku:deploy -DskipTests=true -
Dmaven.javadoc.skip=true -B -V -D heroku.appName=${herokuApp}"
    }
}

def getRepoSlug() {
    tokens = "${env.JOB_NAME}".tokenize('/')
    org = tokens[tokens.size()-3]
    repo = tokens[tokens.size()-2]
    return "${org}/${repo}"
}

def getBranch() {
    tokens = "${env.JOB_NAME}".tokenize('/')
    branch = tokens[tokens.size()-1]
    return "${branch}"
}

def createDeployment(ref, environment, description) {
    withCredentials([[$class: 'StringBinding', credentialsId:
'GITHUB_TOKEN', variable: 'GITHUB_TOKEN']]) {
        def payload = JsonOutput.toJson(["ref": "${ref}",
"description": "${description}", "environment":
"${environment}", "required_contexts": []])
        def apiUrl =
```

```
"https://api.github.com/repos/${getRepoSlug()}/deployments"
        def response = sh(returnStdout: true, script: "curl -s
-H \"Authorization: Token ${env.GITHUB_TOKEN}\" -H \"Accept:
application/json\" -H \"Content-type: application/json\" -X
POST -d '${payload}' ${apiUrl}").trim()
        def jsonSlurper = new JsonSlurper()
        def data = jsonSlurper.parseText("${response}")
        return data.id
    }
}

void createRelease(tagName, createdAt) {
    withCredentials([[$class: 'StringBinding', credentialsId:
'GITHUB_TOKEN', variable: 'GITHUB_TOKEN']]) {
        def body = "**Created at:** ${createdAt}\n**Deployment
job:** [${env.BUILD_NUMBER}]
(${env.BUILD_URL})\n**Environment:** [${env.HEROKU_PRODUCTION}]
(https://dashboard.heroku.com/apps/${env.HEROKU_PRODUCTION})"
        def payload = JsonOutput.toJson(["tag_name":
"v${tagName}", "name": "${env.HEROKU_PRODUCTION} -
v${tagName}", "body": "${body}"])
        def apiUrl =
"https://api.github.com/repos/${getRepoSlug()}/releases"
        def response = sh(returnStdout: true, script: "curl -s
-H \"Authorization: Token ${env.GITHUB_TOKEN}\" -H \"Accept:
application/json\" -H \"Content-type: application/json\" -X
POST -d '${payload}' ${apiUrl}").trim()
    }
}

void setDeploymentStatus(deploymentId, state, targetUrl,
description) {
    withCredentials([[$class: 'StringBinding', credentialsId:
'GITHUB_TOKEN', variable: 'GITHUB_TOKEN']]) {
        def payload = JsonOutput.toJson(["state": "${state}",
"target_url": "${targetUrl}", "description": "${description}"])
        def apiUrl =
"https://api.github.com/repos/${getRepoSlug()}/deployments/${de
ploymentId}/statuses"
        def response = sh(returnStdout: true, script: "curl -s
-H \"Authorization: Token ${env.GITHUB_TOKEN}\" -H \"Accept:
application/json\" -H \"Content-type: application/json\" -X
POST -d '${payload}' ${apiUrl}").trim()
    }
}
```

```
void setBuildStatus(context, message, state) {
  // partially hard coded URL because of
https://issues.jenkins-ci.org/browse/JENKINS-36961, adjust to
your own GitHub instance
  step([
      $class: "GitHubCommitStatusSetter",
      contextSource: [$class:
"ManuallyEnteredCommitContextSource", context: context],
      reposSource: [$class: "ManuallyEnteredRepositorySource",
url: "https://octodemo.com/${getRepoSlug()}"],
      errorHandlers: [[$class:
"ChangingBuildStatusErrorHandler", result: "UNSTABLE"]],
      statusResultSource: [ $class:
"ConditionalStatusResultSource", results: [[$class:
"AnyBuildResult", message: message, state: state]] ]
  ]);
}

def getCurrentHerokuReleaseVersion(app) {
    withCredentials([[$class: 'StringBinding', credentialsId:
'HEROKU_API_KEY', variable: 'HEROKU_API_KEY']]) {
        def apiUrl = "https://api.heroku.com/apps/${app}/dynos"
        def response = sh(returnStdout: true, script: "curl -s
-H \"Authorization: Bearer ${env.HEROKU_API_KEY}\" -H \"Accept:
application/vnd.heroku+json; version=3\" -X GET
${apiUrl}").trim()
        def jsonSlurper = new JsonSlurper()
        def data = jsonSlurper.parseText("${response}")
        return data[0].release.version
    }
}

def getCurrentHerokuReleaseDate(app, version) {
    withCredentials([[$class: 'StringBinding', credentialsId:
'HEROKU_API_KEY', variable: 'HEROKU_API_KEY']]) {
        def apiUrl =
"https://api.heroku.com/apps/${app}/releases/${version}"
        def response = sh(returnStdout: true, script: "curl -s
-H \"Authorization: Bearer ${env.HEROKU_API_KEY}\" -H \"Accept:
application/vnd.heroku+json; version=3\" -X GET
${apiUrl}").trim()
        def jsonSlurper = new JsonSlurper()
        def data = jsonSlurper.parseText("${response}")
        return data.created_at
```

```
    }
}
```

## Groovy code

#### Groovy 函数

```

node {
    stage("Test") {
        test()
    }
}

def test() {
    echo "Start"
    sleep(5)
    echo "Stop"
}
```

## Ansi Color

```

// This shows a simple build wrapper example, using the
AnsiColor plugin.
node {
    // This displays colors using the 'xterm' ansi color map.
    ansiColor('xterm') {
        // Just some echoes to show the ANSI color.
        stage "\u001B[31mI'm Red\u001B[0m Now not"
    }
}
```

写文件操作

```
// This shows a simple example of how to archive the build
output artifacts.
node {
    stage "Create build output"

    // Make the output directory.
    sh "mkdir -p output"

    // Write an useful file, which is needed to be archived.
    writeFile file: "output/usefulfile.txt", text: "This file
is useful, need to archive it."

    // Write an useless file, which is not needed to be
archived.
    writeFile file: "output/uselessfile.md", text: "This file
is useless, no need to archive it."

    stage "Archive build output"

    // Archive the build output artifacts.
    archiveArtifacts artifacts: 'output/*.txt', excludes:
'output/*.md'
}
```

**modules 实现模块**

```
def modules = [
  'Java',
  'PHP',
  'Python',
  'Ruby'
]

node() {
```

```
  stage("checkout") {
    echo "checkout"
  }

  modules.each { module ->
    stage("build:${module}") {
      echo "${module}"
    }
  }
}
```

**docker**

```
node('master') {

    stage('Build') {
        docker.image('maven:3.5.0').inside {
            sh 'mvn --version'
        }
    }

    stage('Deploy') {
        if (env.BRANCH_NAME == 'master') {
            echo 'I only execute on the master branch'
        } else {
            echo 'I execute elsewhere'
        }
    }
}
```

**input**

```
node {
    stage('Git') {
```

```
        def branch = input message: 'input branch name for this
job', ok: 'ok', parameters: [string(defaultValue: 'master',
description: 'branch name', name: 'branch')]
        echo branch
    }
}
```

```
node {
    stage('Git') {
        def result = input message: 'input branch name for this
job', ok: 'ok', parameters: [string(defaultValue: 'master',
description: 'branch name', name: 'branch'),
string(defaultValue: '', description: 'commit to switch', name:
'commit')]

        echo result.branch
        echo result.commit
    }
}

node {
    stage('Git') {
        def result = input message: 'input branch name for this
job', ok: 'ok', parameters: [string(defaultValue: 'master',
description: 'branch name', name: 'branch'),
string(defaultValue: '', description: 'commit to switch', name:
'commit')]

        sh "echo ${result.branch}"
        sh "echo ${result.commit}"
    }
}
```

**if 条件判断**

```
```

```
node {
    dir('/var/www') {
        stage('Git') {
            if(fileExists('project')) {
                dir('project') {
                    sh 'git fetch origin'
                    sh 'git checkout master'
                    sh 'git pull'
                }
            } else {
                sh 'git clone
git@git.netkiller.cn:neo/project.git project'
            }
        }
    }
}
```

## Docker

```
node {
        stage("Checkout") {
                checkout([$class: 'GitSCM', branches: [[name:
env.GIT_BUILD_REF]],userRemoteConfigs: [[url:
env.GIT_REPO_URL]]])
        }

        docker.image('ruby').inside {
                stage("Init") {
                        sh 'pwd && ls'
                        sh 'gem install rails'
                        // sh 'gem install ...'
                }
                stage("Test") {
                        sh 'ruby tc_simple_number.rb'
                }
                stage("Build") {
                        sh 'ruby --version'
                        archiveArtifacts artifacts: 'bin/*',
fingerprint: true
```

```
                }
                stage("Deploy") {
                        sh 'rsync -auzv --delete *
www@host.netkiller.cn:/path/to/dir'
                }
        }
}
```

### conditionalSteps

```
def projectName = 'myProject'

def jobClosure = {
  steps {
    conditionalSteps {
      condition {
        fileExists(projectName+'/target/test.jar',
BaseDir.WORKSPACE)
      }
      runner('Fail')
      steps {
        batchFile('echo Found some tests')
      }
    }
  }
}

freeStyleJob('AAA-Test', jobClosure)
```

### nexus

```
stage("Deploy") {
    nexusArtifactUploader artifacts: [
        [artifactId: 'java11', type: 'jar', file:
```

```
'target/java11.jar']
    ],
    groupId: 'org.springframework.samples',
    nexusUrl: 'netkiller.cn/repository/maven/',
    nexusVersion: 'nexus3',
    protocol: 'http',
    repository: 'maven',
    version: '2.0.0.BUILD'
}
```

## 3.3. 设置环境变量

environment定义键值对的环境变量

```
// Declarative //
pipeline {
    agent any
    environment {
        CC = 'clang'
    }
    stages {
        stage('Example') {
            environment {
                DEBUG_FLAGS = '-g'
            }
            steps {
                sh 'printenv'
            }
        }
    }
}
```

```
// Declarative //
pipeline {
    agent any
```

```
    environment {
        CC = 'clang'
    }
    stages {
        stage('Example') {
            environment {
                AN_ACCESS_KEY = credentials('my-prefined-
secret-text') ③
            }
            steps {
                sh 'printenv'
            }
        }
    }
}
```

系统环境变量

```
echo "Running ${env.BUILD_ID} on ${env.JENKINS_URL}"
```

```
${env.WORKSPACE}
println env.JOB_NAME
println env.BUILD_NUMBER
```

打印所有环境变量

```
node {
    echo sh(returnStdout: true, script: 'env')
    echo sh(script: 'env|sort', returnStdout: true)
    // ...
}
```

```
pipeline {
        agent any
    stages {
        stage('Environment Example') {
            steps {
              script{
                def envs = sh(returnStdout: true, script:
'env').split('\n')
                                    envs.each { name  ->
                                    println "Name: $name"
                                    }
              }
            }
        }
    }
}
```

## 3.4. agent

agent 指令指定整个管道或某个特定的stage的执行环境。它的参数可用使用:

```
agent:
any - 任意一个可用的agent
none - 如果放在pipeline顶层，那么每一个stage都需要定义自己的agent指令
label - 在jenkins环境中指定标签的agent上面执行，比如agent { label
'my-defined-label' }
node - agent { node { label 'labelName' } } 和 label一样，但是可用
定义更多可选项
docker - 指定在docker容器中运行
dockerfile - 使用源码根目录下面的Dockerfile构建容器来运行
```

## label

```
        agent {
        label "java-8"
    }
```

## docker

https://jenkins.io/doc/book/pipeline/docker/

添加 jenkins 用户到 docker 组

```
[root@localhost ~]# gpasswd -a jenkins docker
Adding user jenkins to group docker

[root@localhost ~]# cat /etc/group | grep ^docker
docker:x:993:jenkins
```

**指定docker 镜像**

```
pipeline {
    agent { docker { image 'maven:3.3.3' } }
    stages {
        stage('build') {
            steps {
                sh 'mvn --version'
            }
        }
    }
}
```

```
pipeline {
    agent { docker { image 'php' } }
    stages {
        stage('build') {
            steps {
                sh 'php --version'
            }
        }
    }
}

pipeline {
    agent {
        docker { image 'php:latest' }
    }
    stages {
        stage('Test') {
            steps {
                sh 'php --version'
            }
        }
    }
}
```

**args 参数**

挂在 /root/.m2 目录

```
pipeline {
    agent {
        docker {
            image 'maven:latest'
            args '-v $HOME/.m2:/root/.m2'
        }
    }
```

```
    stages {
        stage('Build') {
            steps {
                sh 'mvn -B'
            }
        }
    }
}
```

**Docker outside of Docker (DooD)**

```
        docker.image('maven').inside("-v
/var/run/docker.sock:/var/run/docker.sock -v
/usr/bin/docker:/usr/bin/docker") {
                sh 'docker images'
        }
```

挂在宿主主机目录

```
node {
    stage("Checkout") {
        checkout(
          [$class: 'GitSCM', branches: [[name:
env.GIT_BUILD_REF]], userRemoteConfigs: [[url:
env.GIT_REPO_URL]]]
        )
                sh 'pwd'
    }
        docker.image('maven:latest').inside("-v
/root/.m2:/root/.m2") {
        stage("Build") {
                sh 'java -version'
                sh 'mvn package -Dmaven.test.failure.ignore -
Dmaven.test.skip=true'
                archiveArtifacts artifacts: '**/target/*.jar',
```

```
fingerprint: true
        }

        stage("Test") {
                sh 'java -jar target/webflux-0.0.1-SNAPSHOT.jar
&'
                sleep 20
                sh 'mvn test -Dmaven.test.failure.ignore'
                junit 'target/surefire-reports/*.xml'
        }
    }
}
```

构建镜像

```
node {
    checkout scm

    docker.withRegistry('http://hub.netkiller.cn:5000') {

        def customImage = docker.build("project/api:1.0")

        /* Push the container to the custom Registry */
        customImage.push()
    }
}
```

容器内运行脚本

```
node {
    checkout scm

    def customImage = docker.build("my-image:${env.BUILD_ID}")

    customImage.inside {
```

```
        sh 'make test'
    }
}
```

```
                dir ('example') {
            /* 构建镜像 */
            def customImage = docker.build("example-
group/example:${params.VERSION}")

            /* hub.netkiller.cn是你的Docker Registry */
            docker.withRegistry('https://hub.netkiller.cn/',
'docker-registry') {
                /* Push the container to the custom Registry */
                // push 指定版本
                customImage.push('latest')
            }
        }
```

```
stage('DockerBuild') {
    sh """
    rm -f src/docker/*.jar
    cp target/*.jar src/docker/*.jar
    """

    dir ("src/docker/") {
        def image = docker.build("your/demo:1.0.0")
        image.push()
    }
}
```

## Dockerfile

创建 Dockerfile 文件

```
FROM node:7-alpine

RUN apk add -U subversion
```

创建 Jenkinsfile 文件

```
// Jenkinsfile (Declarative Pipeline)
pipeline {
    agent { dockerfile true }
    stages {
        stage('Test') {
            steps {
                sh 'node --version'
                sh 'svn --version'
            }
        }
    }
}
```

## 3.5. Steps

### parallel 平行执行

```
        stage('test') {
      parallel {
        stage('test') {
          steps {
            echo 'hello'
          }
        }
        stage('test1') {
          steps {
```

```
            sleep 1
          }
        }
        stage('test2') {
          steps {
            retry(count: 5) {
              echo 'hello'
            }

          }
        }
      }
```

**echo**

```
    stage('Deploy') {
        echo 'Deploying....'
    }
```

**catchError 捕获错误**

```
node {
    catchError {
        sh 'might fail'
    }
    step([$class: 'Mailer', recipients: 'admin@somewhere'])
}


        stage('teatA') {
      steps {
        catchError() {
          sh 'make'
        }
```

```
        mail(subject: 'Test', body: 'aaaa', to:
'netkiller@msn.com')
      }
    }
```

睡眠

```
node {
    sleep 10
    echo 'Hello World'
}
```

```
sleep(time:3,unit:"SECONDS")
```

限制执行时间

```
              stage('enforce') {
          steps {
            timeout(activity: true, time: 1) {
              echo 'test'
            }

          }
        }
```

时间截

```
        stage('timestamps') {
          steps {
            timestamps() {
              echo 'test'
            }
          }
    }
```

## 3.6. 版本控制

### checkout

https://github.com/jenkinsci/workflow-scm-step-plugin/blob/master/README.md

下面配置适用与 Webhook 方式

```
        stage('checkout') {
      steps {
        checkout(scm: [$class: 'GitSCM', branches: [[name:
env.GIT_BUILD_REF]],
                            userRemoteConfigs: [[url:
env.GIT_REPO_URL]]], changelog: true, poll: true)
      }
    }
```

从 URL 获取代码

```
node {
   checkout ([$class: 'GitSCM', branches: [[name: '*/master']],
doGenerateSubmoduleConfigurations: false, extensions: [],
```

```
submoduleCfg: [], userRemoteConfigs: [[credentialsId: '', url:
'https://github.com/bg7nyt/java.git']]])
}
```

**Git**

```
                stage('Git') {
                steps {
                git(url:
'https://git.dev.tencent.com/netkiller/java.git', branch:
'master', changelog: true, poll: true)
                }
        }
```

## 3.7. 节点与过程

**sh**

```
                stage("build") {
            steps {
                sh "mvn package -Dmaven.test.skip=true"
            }
        }
```

```
                steps {
                script{
                sh 'find /etc/'
                        }
                }
```

```
```

例 12.1. Shell Docker 示例

Shell Docker 使用 docker 命令完成构建过程

```
registryUrl='127.0.0.1:5000'         # 私有镜像仓库地址
imageName='netkiller/project'    # 镜像名称
imageTag=$BRANCH                     # 上面设置Branch分支，这里可以
当做环境变量使用

echo ' >>> [INFO] enter workspace ...'

cd $WORKSPACE/                               # 进入到jenkins的工作区，
jenkins会将gitlab仓库代码pull到这里，用于制作镜像文件

# 根据不同的Branch生成不同的swoft的配置文件，区分测试还是生成等
echo ' >>> [INFO] create startup.sh ...'
(
cat << EOF

# 启动 Shell 写在此处

EOF
) > ./entrypoint.sh

# 生成 Dockerfile
echo ' >>> [INFO] begin create Dockerfile ...'
rm -f ./Dockerfile
(
cat << EOF
FROM netkiller/base
LABEL maintainer=netkiller@msn.com

COPY . /var/www/project
WORKDIR /var/www/project
EXPOSE 80
ENV PHP_ENVIRONMENT $BRANCH
ENTRYPOINT [ "bash", "/var/www/project/entrypoint.sh" ]
EOF
) > ./Dockerfile
```

```
#删除无用镜像
echo ' >>> [INFO] begin cleaning image ...'
for image in `docker images | grep -w $imageName | grep -i -w
$imageTag | awk '{print $3}'`
do
    docker rmi $image -f
done

#制作镜像
echo ' >>> [INFO] begin building image ...'
docker build --tag $imageName:$imageTag --rm .

#给镜像打标签
img=`docker images | grep -w $imageName | grep -i -w $imageTag
| awk '{print $3}'`
docker tag $img $registryUrl/$imageName:$imageTag

#push到私有镜像仓库
echo ' >>> [INFO] begin publishing image ...'
docker push $registryUrl/$imageName:$imageTag

#删除刚刚制作的镜像，释放存储空间
echo ' >>> [INFO] cleaning temporary building ...'
docker rmi -f $imageName:$imageTag
docker rmi -f $registryUrl/$imageName:$imageTag
```

## Windows 批处理脚本

```
        stage('bat') {
        steps {
            bat(returnStatus: true, returnStdout: true, label:
'aa', encoding: 'utf-8', script: 'dir')
        }
    }
```

分配工作空间

```
      stage('alocate') {
      steps {
          ws(dir: 'src') {
              echo 'aaa'
          }

      }
  }
```

**node**

```
      stage('node') {
      steps {
          node(label: 'java-8') {
            sh 'mvn package'
          }

      }
      }
```

## 3.8. 工作区

变更目录

```
      stage('subtask') {
      steps {
          dir(path: '/src') {
              echo 'begin'
              sh '''mvn test'''
```

```
                echo 'end'
                }
                }
        }
```

判断文件是否存在

```
        stage('exists') {
          steps {
            fileExists '/sss'
          }
    }
```

```
def exists = fileExists 'file'

if (exists) {
    echo 'Yes'
} else {
    echo 'No'
}
```

```
if (fileExists('file')) {
    echo 'Yes'
} else {
    echo 'No'
}
```

```
```

```
pipeline{
        agent any
        stages{
      stage("Checkout") {
            steps {
                checkout(
                  [$class: 'GitSCM', branches: [[name:
env.GIT_BUILD_REF]],
                  userRemoteConfigs: [[url: env.GIT_REPO_URL]]]
                )
            }
        }

                stage("fileExists") {
                        steps{
                echo pwd()
                sh 'ls -1'
                                script {
                                        def exists = fileExists
'README.md'
                    if (exists) {
                        echo 'Yes'
                    } else {
                        echo 'No'
                    }
                                }
                        }
                }
        }
}
```

分配工作区

```
    stage('alocate') {
        steps {
            ws(dir: 'src') {
                echo 'aaa'
            }
```

```
        }
    }
```

清理工作区

```
        stage('test') {
        steps {
            cleanWs(cleanWhenAborted: true, cleanWhenFailure:
true, cleanWhenNotBuilt: true, cleanWhenSuccess: true,
cleanWhenUnstable: true, cleanupMatrixParent: true, deleteDirs:
true, disableDeferredWipeout: true, notFailBuild: true,
skipWhenFailed: true, externalDelete: '/aa')
        }
    }
```

递归删除目录

```
        stage('deldir') {
          steps {
            deleteDir()
          }
    }
```

写文件

```
              stage('write') {
          steps {
            writeFile(file: 'hello.txt', text: 'Helloworld')
```

```
    }
  }
```

读文件

```
  stage('read') {
    steps {
      readFile 'hello.txt'
    }
  }
```

# 4. Jenkins Job DSL / Plugin

```
def gitUrl = 'git://github.com/jenkinsci/job-dsl-plugin.git'

job('PROJ-unit-tests') {
    scm {
        git(gitUrl)
    }
    triggers {
        scm('*/15 * * * *')
    }
    steps {
        maven('-e clean test')
    }
}

job('PROJ-sonar') {
    scm {
        git(gitUrl)
    }
    triggers {
        cron('15 13 * * *')
    }
    steps {
        maven('sonar:sonar')
    }
}

job('PROJ-integration-tests') {
    scm {
        git(gitUrl)
    }
    triggers {
        cron('15 1,13 * * *')
    }
    steps {
        maven('-e clean integration-test')
    }
}
```

```
job('PROJ-release') {
    scm {
        git(gitUrl)
    }
    // no trigger
    authorization {
        // limit builds to just Jack and Jill
        permission('hudson.model.Item.Build', 'jill')
        permission('hudson.model.Item.Build', 'jack')
    }
    steps {
        maven('-B release:prepare release:perform')
        shell('cleanup.sh')
    }
}
```

```
job('PROJ-unit-tests') {
    scm {
        git('https://github.com/bg7nyt/java.git')
    }
    triggers {
        scm('*/15 * * * *')
    }
    steps {
        maven('-e clean test')
    }
}
```

# 5. Jenkins Plugin

## 5.1. Blue Ocean

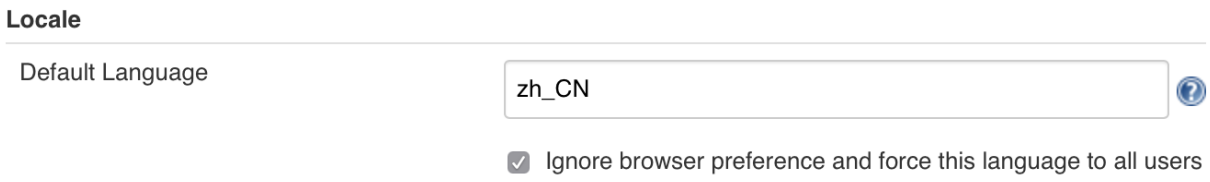
https://jenkins.io/doc/book/blueocean/getting-started/

http://jk.netkiller.cn/blue/

## 5.2. Locale Plugin (国际化插件)

安装Locale Plugin, 重启生效。

```
配置【Manage Jenkins】>【Configure System】>  【Locale】
```




Default Language 填写 zh_CN，勾选忽略浏览器设置强制设置语言

## 5.3. github-plugin 插件

https://github.com/jenkinsci/github-plugin

```
git clone https://github.com/jenkinsci/github-plugin.git
mkdir target/classes
```

修改 rest-assured 去掉 exclusions 配置项

```
        <dependency>
            <groupId>com.jayway.restassured</groupId>
            <artifactId>rest-assured</artifactId>
            <!--1.7.2 is the last version that use a compatible
groovy version-->
            <version>1.7.2</version>
            <scope>test</scope>
            <exclusions>
                <exclusion>
                    <groupId>org.apache.httpcomponents</groupId>
                    <artifactId>*</artifactId>
                </exclusion>
            </exclusions>
        </dependency>
```

编译插件

```
[root@netkiller github-plugin]# mvn hpi:hpi
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------
---------------
[INFO] Building GitHub plugin 1.29.4-SNAPSHOT
[INFO] ------------------------------------------------------------
---------------
[INFO]
[INFO] --- maven-hpi-plugin:1.120:hpi (default-cli) @ github ---
[INFO] Generating /srv/github-plugin/target/github/META-
INF/MANIFEST.MF
[INFO] Checking for attached .jar artifact ...
[INFO] Generating jar /srv/github-plugin/target/github.jar
[INFO] Building jar: /srv/github-plugin/target/github.jar
[INFO] Exploding webapp...
[INFO] Copy webapp webResources to /srv/github-
plugin/target/github
[INFO] Assembling webapp github in /srv/github-
plugin/target/github
[INFO] Generating hpi /srv/github-plugin/target/github.hpi
[INFO] Building jar: /srv/github-plugin/target/github.hpi
```

```
[INFO] ------------------------------------------------------------
---------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------
---------------
[INFO] Total time: 4.161s
[INFO] Finished at: Mon Jan 07 12:03:45 CST 2019
[INFO] Final Memory: 29M/290M
[INFO] ------------------------------------------------------------
---------------
```

进入 github --> Settings --> Developer settings --> Personal Access Token --> Generate new token

repo 和 admin:repo_hook

Settings -> Webhooks -> Add webhook

系统管理 --> 系统设置 --> GitHub --> Add GitHub Sever

## 5.4. Docker

**This plugin integrates Jenkins with Docker**

https://jenkins.io/doc/book/pipeline/docker/

```
vim /lib/systemd/system/docker.service

ExecStart=/usr/bin/dockerd -H fd://

改为

ExecStart=/usr/bin/dockerd -H fd:// -H
unix:///var/run/docker.sock -H tcp://0.0.0.0:2375
```

吧 jenkins 用户添加到 docker 组

```
gpasswd -a jenkins docker
```

重启 docker

```
systemctl daemon-reload
systemctl restart docker

如果是 Docker 方式运行 Jenkins 需要启动 jenkins
docker start jenkins
```

参考例子

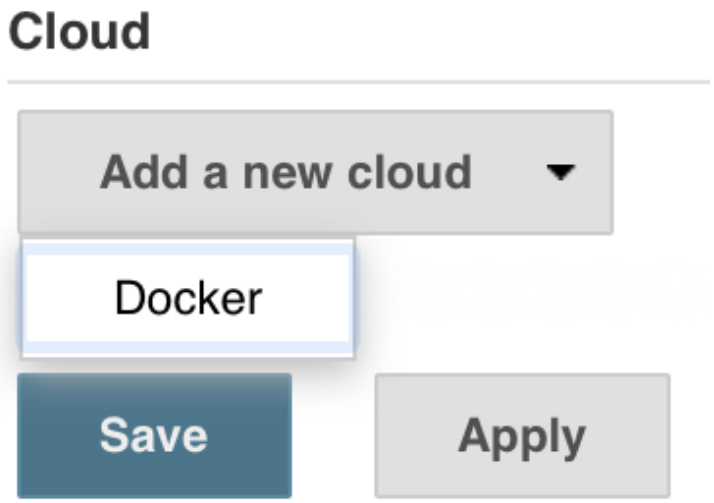
```
root@ubuntu:~# cat /lib/systemd/system/docker.service
[Unit]
Description=Docker Application Container Engine
Documentation=https://docs.docker.com
After=network-online.target docker.socket firewalld.service
Wants=network-online.target
Requires=docker.socket

[Service]
Type=notify
# the default is not to use systemd for cgroups because the
delegate issues still
# exists and systemd currently does not support the cgroup
feature set required
# for containers run by docker
ExecStart=/usr/bin/dockerd -H fd:// -H
unix:///var/run/docker.sock -H tcp://0.0.0.0:2375
ExecReload=/bin/kill -s HUP $MAINPID
LimitNOFILE=1048576
# Having non-zero Limit*s causes performance problems due to
```

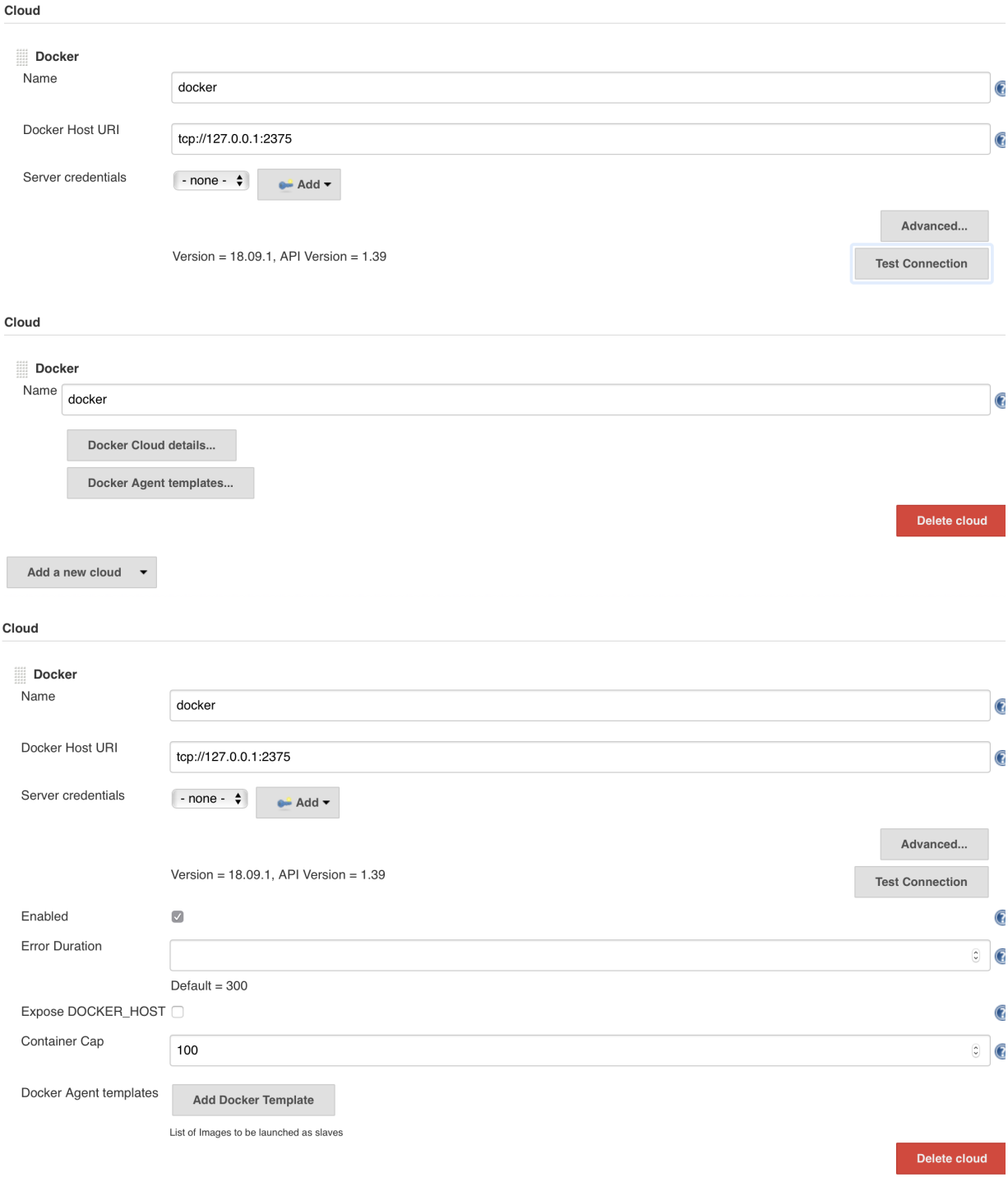
```
accounting overhead
# in the kernel. We recommend using cgroups to do container-
local accounting.
LimitNPROC=infinity
LimitCORE=infinity
# Uncomment TasksMax if your systemd version supports it.
# Only systemd 226 and above support this version.
TasksMax=infinity
TimeoutStartSec=0
# set delegate yes so that systemd does not reset the cgroups
of docker containers
Delegate=yes
# kill only the docker process, not all processes in the cgroup
KillMode=process
# restart the docker process if it exits prematurely
Restart=on-failure
StartLimitBurst=3
StartLimitInterval=60s

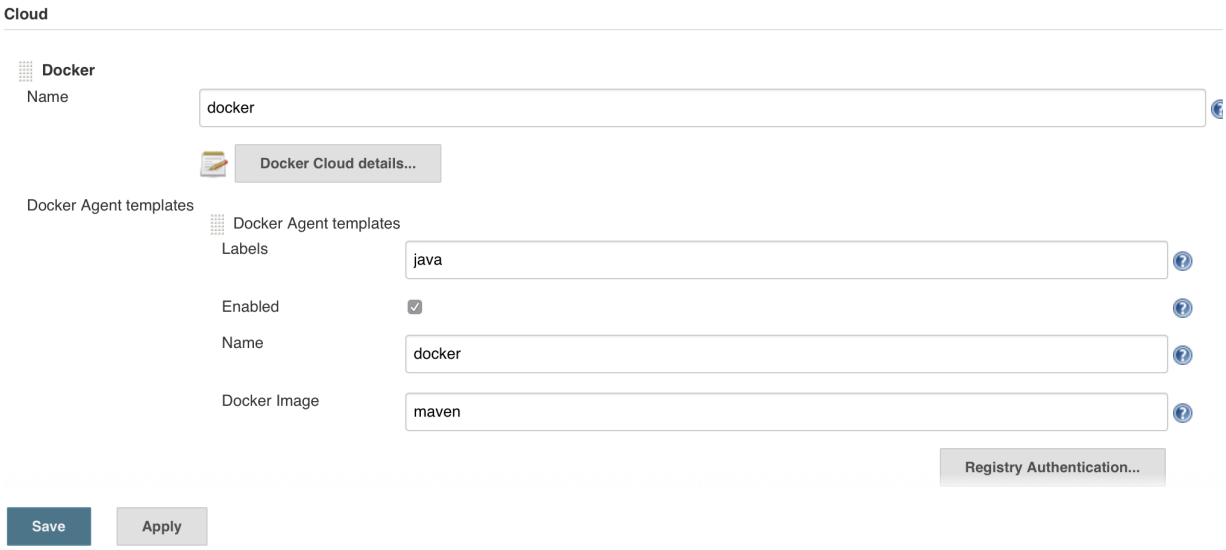
[Install]
WantedBy=multi-user.target
```

**设置 Docker 主机和代理**



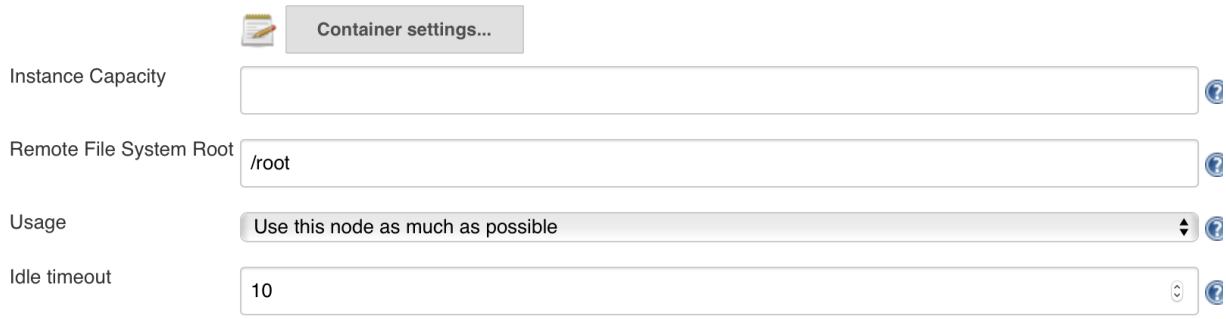

输入 Docker 主机的IP地址，类似 tcp://172.16.0.10:2375


Cloud
Docker
Name
docker
Docker Host URI
tcp://127.0.0.1:2375
Server credentials
- none -
Add
Advanced...
Version = 18.09.1, API Version = 1.39
Test Connection
Cloud
Docker
Name
docker
Docker Cloud details...
Docker Agent templates...
Delete cloud
Add a new cloud
Cloud
Docker
Name
docker
Docker Host URI
tcp://127.0.0.1:2375
Server credentials
- none -
Add
Advanced...
Version = 18.09.1, API Version = 1.39
Test Connection
Enabled
Error Duration
Default = 300
Expose DOCKER_HOST
Container Cap
100
Docker Agent templates
Add Docker Template
List of Images to be launched as slaves
Delete cloud

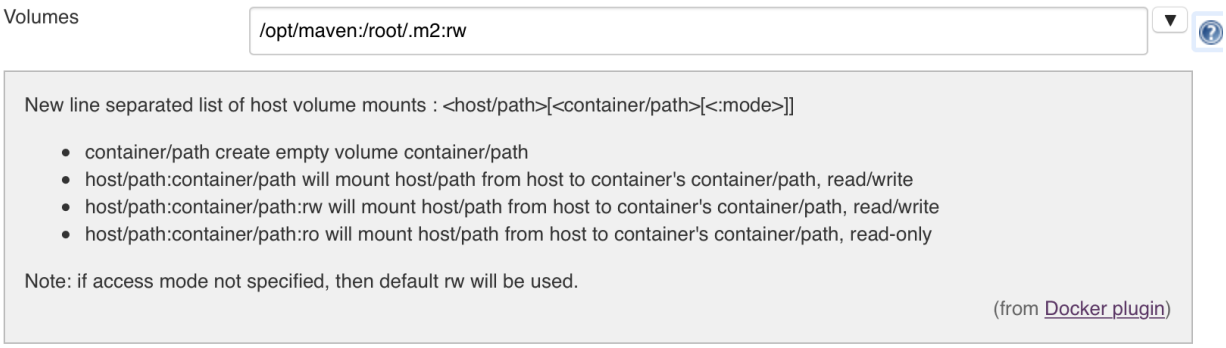



## 持久化

例如持续集成过程中，我们不希望每次都从 maven 镜像下载编译依赖的包，或者构建物我们需要永久保留等等，这时就需要做持久化




例如我们将宿主主机的 /opt/maven 挂载到 Docker 容器的 /root/.m2 目录。这样就实现了 maven 的持久化。只需写入 /opt/maven:/root/.m2 即可




当持续机构运行完毕 docker 容器被清理，但是 /opt/maven 并不会被清理，下次构建时，在将它挂载到 /root/.m2 即可。

## 5.5. JaCoCo

### This plugin integrates JaCoCo code coverage reports to Jenkins.

https://jenkins.io/doc/pipeline/steps/jacoco/

### Pipeline

```
stage('Build') {
     steps {
        sh 'mvn test'
        junit '*/build/test-results/*.xml'
        step( [ $class: 'JacocoPublisher' ] )
     }
}
```

配置jacoco

```
The jacoco pipeline step configuration uses this format:

step([$class: 'JacocoPublisher',
```

```
      execPattern: 'target/*.exec',
      classPattern: 'target/classes',
      sourcePattern: 'src/main/java',
      exclusionPattern: 'src/test*'
])
Or with a simpler syntax for declarative pipeline:

jacoco(
      execPattern: 'target/*.exec',
      classPattern: 'target/classes',
      sourcePattern: 'src/main/java',
      exclusionPattern: 'src/test*'
)
```

完整的例子

```
node {
    stage('Checkout') {
        git 'https://github.com/bg7nyt/junit4-jacoco.git'
    }
    stage('Build') {
        sh "mvn -Dmaven.test.failure.ignore clean package"
    }
    stage('Test') {
        sh "mvn test"
    }
    stage('Results') {
        junit '**/target/surefire-reports/TEST-*.xml'
        archive 'target/*.jar'
        step( [ $class: 'JacocoPublisher' ] )
    }
}
```

## 5.6. SSH Pipeline Steps

使用说明: https://github.com/jenkinsci/ssh-steps-plugin#pipeline-steps

```
!groovy
def getHost(){
    def remote = [:]
    remote.name = 'mysql'
    remote.host = '192.168.8.108'
    remote.user = 'root'
    remote.port = 22
    remote.password = 'qweasd'
    remote.allowAnyHosts = true
    return remote
}
pipeline {
    agent {label 'master'}
    environment{
        def server = ''
    }
    stages {
        stage('init-server'){
            steps {
                script {
                   server = getHost()
                }
            }
        }
        stage('use'){
            steps {
                script {
                  sshCommand remote: server, command: """
                  if test ! -d aaa/ccc ;then mkdir -p
aaa/ccc;fi;cd aaa/ccc;rm -rf ./*;echo 'aa' > aa.log
                  """
                }
            }
        }
    }
}
####################################
node {
  def remote = [:]
  remote.name = 'test'
  remote.host = 'test.domain.com'
  remote.user = 'root'
  remote.password = 'password'
```

```
  remote.allowAnyHosts = true
  stage('Remote SSH') {
    sshCommand remote: remote, command: "ls -lrt"
    sshCommand remote: remote, command: "for i in {1..5}; do
echo -n \"Loop \$i \"; date ; sleep 1; done"
  }
}
node {
  def remote = [:]
  remote.name = 'test'
  remote.host = 'test.domain.com'
  remote.user = 'root'
  remote.password = 'password'
  remote.allowAnyHosts = true
  stage('Remote SSH') {
    writeFile file: 'abc.sh', text: 'ls -lrt'
    sshScript remote: remote, script: "abc.sh"
  }
}
node {
  def remote = [:]
  remote.name = 'test'
  remote.host = 'test.domain.com'
  remote.user = 'root'
  remote.password = 'password'
  remote.allowAnyHosts = true
  stage('Remote SSH') {
    writeFile file: 'abc.sh', text: 'ls -lrt'
    sshPut remote: remote, from: 'abc.sh', into: '.'
  }
}
node {
  def remote = [:]
  remote.name = 'test'
  remote.host = 'test.domain.com'
  remote.user = 'root'
  remote.password = 'password'
  remote.allowAnyHosts = true
  stage('Remote SSH') {
    sshGet remote: remote, from: 'abc.sh', into: 'abc_get.sh',
override: true
  }
}
node {
  def remote = [:]
```

```
  remote.name = 'test'
  remote.host = 'test.domain.com'
  remote.user = 'root'
  remote.password = 'password'
  remote.allowAnyHosts = true
  stage('Remote SSH') {
    sshRemove remote: remote, path: "abc.sh"
  }
}
def remote = [:]
remote.name = "node-1"
remote.host = "10.000.000.153"
remote.allowAnyHosts = true
node {
    withCredentials([sshUserPrivateKey(credentialsId: 'sshUser',
keyFileVariable: 'identity', passphraseVariable: '',
usernameVariable: 'userName')]) {
        remote.user = userName
        remote.identityFile = identity
        stage("SSH Steps Rocks!") {
            writeFile file: 'abc.sh', text: 'ls'
            sshCommand remote: remote, command: 'for i in
{1..5}; do echo -n \"Loop \$i \"; date ; sleep 1; done'
            sshPut remote: remote, from: 'abc.sh', into: '.'
            sshGet remote: remote, from: 'abc.sh', into:
'bac.sh', override: true
            sshScript remote: remote, script: 'abc.sh'
            sshRemove remote: remote, path: 'abc.sh'
        }
    }
}
################################
```

## 5.7. Rancher

https://plugins.jenkins.io/rancher

https://jenkins.io/doc/pipeline/steps/rancher/

创建 Rancher API

在Jenkins的Credentials中添加一个类型为Username with password的认证，username和password分别对应于上一步生成的Access Key和Secret Key，如下图

然后在语法生成器中，找到rancher进行如下图的配置：

设置 environmentId ，找到你的集群，点击进入，看到URL https://rancher.netkiller.cn/v3/clusters/c-mx88f，“c-mx88f” 就是 environmentId

```
stage('Rancher') {
    rancher confirm: false, credentialId: 'b56bd9b2-3277-4072-
baae-08d73aa26549', endpoint: 'https://rancher.netkiller.cn/v2',
environmentId: 'test', environments: '', image: '*/demo:1.0.0',
ports: '', service: 'jenkins/demo', timeout: 50
}
```

## 5.8. Kubernetes 插件

### Kubernetes

https://plugins.jenkins.io/kubernetes

https://github.com/jenkinsci/kubernetes-plugin

```
def label = "mypod-${UUID.randomUUID().toString()}"
podTemplate(label: label) {
    node(label) {
        stage('Run shell') {
            sh 'echo hello world'
        }
    }
}
```

### Kubernetes :: Pipeline :: Kubernetes Steps

```

```

### Kubernetes Continuous Deploy

https://plugins.jenkins.io/kubernetes-cd

### Kubernetes Cli

:

## 5.9. HTTP Request Plugin

```
GET
https://<rancher_server>/v3/project/<project_id>/workloads/deplo
yment:<rancher_namespace>:<rancher_service> # 获取一个服务的详细信息
GET https://<rancher_server>/v3/project/<project_id>/pods/?
workloadId=deployment:<rancher_namespace>:<rancher_service> # 获
取服务的所有容器信息
DELETE
https://<rancher_server>/v3/project/<project_id>/pods/<rancher_n
amespace>:<container_name> # 根据容器名删除容器
PUT
https://<rancher_server>/v3/project/<project_id>/workloads/deplo
yment:<rancher_namespace>:<rancher_service> # 更新服务
```

```
// 查询服务信息
def response = httpRequest acceptType: 'APPLICATION_JSON',
authentication: "${RANCHER_API_KEY}", contentType:
'APPLICATION_JSON', httpMode: 'GET', responseHandle:
```

```
'LEAVE_OPEN', timeout: 10, url:
"${rancherUrl}/workloads/deployment:${rancherNamespace}:${ranche
rService}"
def serviceInfo = new
JsonSlurperClassic().parseText(response.content)
response.close()

def dockerImage = imageName+":"+imageTag
if (dockerImage.equals(serviceInfo.containers[0].image)) {
    // 如果镜像名未改变，直接删除原容器
    // 查询容器名称
    response = httpRequest acceptType: 'APPLICATION_JSON',
authentication: "${RANCHER_API_KEY}", contentType:
'APPLICATION_JSON', httpMode: 'GET', responseHandle:
'LEAVE_OPEN', timeout: 10, url: "${rancherUrl}/pods/?
workloadId=deployment:${rancherNamespace}:${rancherService}"
    def podsInfo = new
JsonSlurperClassic().parseText(response.content)
    def containerName = podsInfo.data[0].name
    response.close()
    // 删除容器
    httpRequest acceptType: 'APPLICATION_JSON', authentication:
"${RANCHER_API_KEY}", contentType: 'APPLICATION_JSON', httpMode:
'DELETE', responseHandle: 'NONE', timeout: 10, url:
"${rancherUrl}/pods/${rancherNamespace}:${containerName}"

} else {
    // 如果镜像名改变，使用新镜像名更新容器
    serviceInfo.containers[0].image = dockerImage
    // 更新
    def updateJson = new JsonOutput().toJson(serviceInfo)
    httpRequest acceptType: 'APPLICATION_JSON', authentication:
"${RANCHER_API_KEY}", contentType: 'APPLICATION_JSON', httpMode:
'PUT', requestBody: "${updateJson}", responseHandle: 'NONE',
timeout: 10, url:
"${rancherUrl}/workloads/deployment:${rancherNamespace}:${ranche
rService}"
}
```

## 5.10. Skip Certificate Check plugin

https://wiki.jenkins.io/display/JENKINS/Skip+Certificate+Check+plugin

```
[root@localhost ~]# tail -f /var/log/jenkins/jenkins.log

javax.net.ssl.SSLPeerUnverifiedException: peer not authenticated
```

## 5.11. Android Sign Plugin

Android Sign Plugin 依赖 Credentials Plugin，因为 Credentials Plugin 只支持 PKCS#12 格式的证书，所以先需要将生成好的 JKS 证书转换为 PKCS#12 格式：

```
keytool -importkeystore -srckeystore netkiller.jks -srcstoretype
JKS -deststoretype PKCS12 -destkeystore netkiller.p12
```

复制代码添加类型为 credential，选择上传证书文件，将 PKCS#12 证书上传到并配置好 ID，本项目中使用了 ANDROID_SIGN_KEY_STORE 作为 ID。

```
pipeline {
    ...

    stages {
        ...

        stage("Sign APK") {
            steps {
                echo 'Sign APK'
                signAndroidApks(
                    keyStoreId: "ANDROID_SIGN_KEY_STORE",
                    keyAlias: "tomczhen",
```

```
                    apksToSign: "**/*-prod-release-
unsigned.apk",
                    archiveSignedApks: false,
                    archiveUnsignedApks: false
                )
            }
        }

        ...
    }

    ...
}
```

# 6. Jenkinsfile Pipeline Example

## 6.1. Maven 子模块范例

Maven 子模块创建方法
https://www.netkiller.cn/java/build/maven.html#maven.module

目录结构

```
Project
    |
    |--- common (Shared)
    |      | ---pom.xml
    |--- project1 (depend common)
    |      |--- pom.xml
    |--- project2 (depend common)
    |      |--- pom.xml
    |---pom.xml
```

构建 父项目

```
pipeline {
    agent {
        label "default"
    }
    stages  {

        stage("检出") {
            steps {
                checkout(
                  [$class: 'GitSCM', branches: [[name:
env.GIT_BUILD_REF]],
                  userRemoteConfigs: [[url: env.GIT_REPO_URL]]]
```

```
                )
            }
        }

        stage("构建") {
            steps {
                echo "构建中..."
                sh 'mvn package -Dmaven.test.skip=true' // mvn
示例
                archiveArtifacts artifacts: '**/target/*.jar',
fingerprint: true // 收集构建产物
                echo "构建完成."
            }
        }

        stage("测试") {
            steps {
                echo "单元测试中..."
                // 请在这里放置您项目代码的单元测试调用过程，例如：
                sh 'mvn test' // mvn 示例
                echo "单元测试完成."
                junit '**/target/surefire-reports/*.xml' // 收集
单元测试报告的调用过程
            }
        }

        stage("部署") {
            steps {
                echo "部署中..."
                echo "部署完成"
            }
        }
    }
}
```

构建共享项目

```
pipeline {
    agent {
        label "default"
```

```
    }
    stages  {

        stage("检出") {
            steps {
                checkout(
                  [$class: 'GitSCM', branches: [[name:
env.GIT_BUILD_REF]],
                  userRemoteConfigs: [[url: env.GIT_REPO_URL]]]
                )
            }
        }

        stage("构建") {
            steps {
                echo "构建中..."
                dir(path: 'common') {
                        sh 'mvn package -Dmaven.test.skip=true'
// mvn 示例
                        archiveArtifacts artifacts:
'**/target/*.jar', fingerprint: true // 收集构建产物
                }
                echo "构建完成."
            }
        }

        stage("测试") {
            steps {
                echo "单元测试中..."
                sh 'mvn test' // mvn 示例
                echo "单元测试完成."
                junit 'target/surefire-reports/*.xml' // 收集单
元测试报告的调用过程
            }
        }

        stage("部署") {
            steps {
                echo "部署中..."
                dir(path: 'common') {
                        sh 'mvn install'
                }
                echo "部署完成"
            }
        }
```

```
    }
}
```

构建 project1 和 project2

```
pipeline {
    agent {
        label "default"
    }
    stages  {

        stage("检出") {
            steps {
                checkout(
                  [$class: 'GitSCM', branches: [[name:
env.GIT_BUILD_REF]],
                  userRemoteConfigs: [[url: env.GIT_REPO_URL]]]
                )
            }
        }
                stage("共享库") {
            steps {
                echo "构建中..."
                dir(path: 'common') {
                        sh 'mvn install -Dmaven.test.skip=true'
// mvn 示例
                        archiveArtifacts artifacts:
'**/target/*.jar', fingerprint: true // 收集构建产物
                }
                echo "构建完成."
            }
        }
        stage("构建") {
            steps {
                echo "构建中..."
                    dir(path: 'project1') {
                    sh 'mvn package -Dmaven.test.skip=true' //
mvn 示例
                            archiveArtifacts artifacts:
'**/target/*.jar', fingerprint: true // 收集构建产物
```

```
                }
                echo "构建完成."

            }
        }

        stage("测试") {
            steps {
                echo "单元测试中..."
                sh 'mvn test' // mvn 示例
                echo "单元测试完成."
                junit 'target/surefire-reports/*.xml' // 收集单
元测试报告的调用过程
            }
        }

        stage("部署") {
            steps {
                echo "部署中..."
                // 部署脚本
                echo "部署完成"
            }
        }
    }
}
```

## 6.2. 使用指定镜像构建

```
pipeline {
    agent any
    stages {
        stage("Checkout") {
            steps {
                sh 'ci-init'
                checkout(
                        [$class           : 'GitSCM', branches:
[[name: env.GIT_BUILD_REF]],
                         userRemoteConfigs: [[url:
env.GIT_REPO_URL]]]
                )
```

```
            }
        }


        stage("Compile") {

            // 构建的 docker 镜像
            agent {
                docker { image 'maven' }
            }

            steps {
                echo "构建中..."
                sh 'mvn -v'
                sh 'mvn compile'
            }
        }


        stage('Test') {

            agent {
                docker { image 'maven' }
            }

            steps {
                echo '单元测试...'
                sh 'mvn test'
                junit 'target/surefire-reports/*.xml'
            }

        }

        stage("Deploy") {
            steps {
                echo "部署中..."
                echo "部署完成"
            }
        }
    }
}
```

## 6.3. 命令行制作 Docker 镜像

```
pipeline {
    agent any
    stages {

        stage('Build') {

            steps {
                echo '编译中...'
                // 编译 docker 镜像
                sh "docker build $tag $contextPath"
            }

        }

        stage('Push Image') {

            steps {

                sh "echo ${REGISTRY_PASS} | docker login -u
${REGISTRY_USER} --password-stdin ${REGISTRY_URL}"
                sh "docker tag ${image} ${registry_image}"
                sh "docker push ${registry_image}"

            }

        }

    }
}
```

```
pipeline {

    agent any

    stages {
        stage("Checkout") {
```

```
          steps {
            checkout([
              $class: 'GitSCM',
              branches: [[name: env.GIT_COMMIT]],
              extensions: [[$class: 'PruneStaleBranch']],
              userRemoteConfigs: [[
                url: env.GIT_REPO_URL,
                refspec: "+refs/heads/*:refs/remotes/origin/*"
              ]]
            ])

            sh '''
                #!/bin/bash
                echo ${GIT_COMMIT}
                echo ${REF}
                echo ${GIT_LOCAL_BRANCH}
            '''
          }
        }

        stage('Build') {
            steps{
                echo "Building begin"
                script{
                    // 设置镜像名
                    env.BUILD_MODULE = "common"
                    env.DOCKER_IMAGE_TAG = env.BUILD_MODULE +
':' + env.GIT_COMMIT
                    env.DOCKER_REMOTE_IMAGE_TAG =
"${env.REGISTRY_URL}/${env.DOCKER_IMAGE_TAG}"

                    sh "docker login ${DOCKER_REGISTER_URL} -u
${DOCKER_REPOSITORY_USERNAME} -p ${DOCKER_REPOSITORY_PASSWORD}"

                    def statusCode = sh(script:"docker pull
${DOCKER_REMOTE_IMAGE_TAG}", returnStatus:true)

                                            // 判断该镜像在仓库是否存在
                    if (statusCode != 0) {

                        sh '''
                        #!/bin/bash

                        # build docker image
                        docker build . -f Dockerfile -t
```

```
${DOCKER_IMAGE_TAG}

                        # tag docker image
                        docker tag ${DOCKER_IMAGE_TAG}
${DOCKER_REMOTE_IMAGE_TAG}

                    }
                }
                echo "Build end"
            }
        }

        stage('Deploy') {
            steps{
                echo "Deploying begin"
                script{
                        # push to
                        docker push ${DOCKER_REMOTE_IMAGE_TAG}

                        # rm
                        docker rmi ${DOCKER_IMAGE_TAG}
                        docker rmi ${DOCKER_REMOTE_IMAGE_TAG}
                        '''
                }
                echo "Deploy end"
            }
        }
    }
}
```

## 6.4. Yarn

```
pipeline {
    agent {
        label "default"
    }
    stages  {

        stage("检出") {
            steps {
```

```
                checkout(
                  [$class: 'GitSCM', branches: [[name:
env.GIT_BUILD_REF]],
                  userRemoteConfigs: [[url: env.GIT_REPO_URL]]]
                )
            }
        }
                stage("环境") {
                steps {
                sh 'apt install -y apt-transport-https'
                sh "curl -sS
https://dl.yarnpkg.com/debian/pubkey.gpg | apt-key add -"
                sh 'echo "deb https://dl.yarnpkg.com/debian/
stable main" | tee /etc/apt/sources.list.d/yarn.list'
                sh 'cat /etc/apt/sources.list.d/yarn.list'
                sh 'apt update && apt install -y yarn'
                sh 'yarn --version'
            }
        }
        stage("构建") {
            steps {
                echo "构建中..."
                sh 'yarn add webpack'
                sh 'node -v'
            }
        }

        stage("测试") {
            steps {
                echo "单元测试中..."
            }
        }

        stage("部署") {
            steps {
                // sh './deploy.sh'
            }
        }
    }
}
```

## 6.5. Android

进入项目目录，找到 local.properties 文件，打开文件

```
## This file is automatically generated by Android Studio.
# Do not modify this file -- YOUR CHANGES WILL BE ERASED!
#
# This file should *NOT* be checked into Version Control
Systems,
# as it contains information specific to your local
configuration.
#
# Location of the SDK. This is only used by Gradle.
# For customization when using a Version Control System, please
read the
# header note.
sdk.dir=/Users/neo/Library/Android/sdk
```

sdk.dir 是 Android SDK 存放目录，进入该目录

```
neo@MacBook-Pro ~ % ll /Users/neo/Library/Android/sdk/
total 0
drwxr-xr-x   3 neo  staff    96B Oct 23 09:56 build-tools
drwxr-xr-x  18 neo  staff   576B Oct 23 09:55 emulator
drwxr-xr-x   6 neo  staff   192B Oct 23 10:21 extras
drwxr-xr-x   3 neo  staff    96B Oct 23 11:35 fonts
drwxr-xr-x   4 neo  staff   128B Oct 23 11:00 licenses
drwxr-xr-x   3 neo  staff    96B Oct 23 09:55 patcher
drwxr-xr-x  19 neo  staff   608B Oct 23 09:56 platform-tools
drwxr-xr-x   4 neo  staff   128B Oct 23 10:23 platforms
drwxr-xr-x  24 neo  staff   768B Oct 23 10:57 skins
drwxr-xr-x   4 neo  staff   128B Oct 23 10:23 sources
drwxr-xr-x   4 neo  staff   128B Oct 24 15:06 system-images
drwxr-xr-x  14 neo  staff   448B Oct 23 09:55 tools

neo@MacBook-Pro ~ % ll /Users/neo/Library/Android/sdk/licenses
total 16
-rw-r--r--  1 neo  staff    41B Oct 23 10:23 android-sdk-
license
-rw-r--r--  1 neo  staff    41B Oct 23 11:00 android-sdk-
```

```
preview-license

neo@MacBook-Pro ~ % cat
/Users/neo/Library/Android/sdk/licenses/android-sdk-license

d56f5187479451eabf01fb78af6dfcb131a6481e
```

/Users/neo/Library/Android/sdk/licenses/android-sdk-license 便是当前 Android SDK License 文件

如果你安装了多个版本的 SDK，例如 android-26， android-27，android-28 可以看到三行字串。

```
24333f8a63b6825ea9c5514f83c2829b004d1fee 这是 Android 8.0 -
android-26
d56f5187479451eabf01fb78af6dfcb131a6481e 这是 Android 9.0 -
android-28
```

```
pipeline {
        agent any
    stages  {

        stage("Checkout") {
            steps {
                checkout(
                  [$class: 'GitSCM', branches: [[name:
env.GIT_BUILD_REF]],
                  userRemoteConfigs: [[url: env.GIT_REPO_URL]]]
                )
            }
        }

        stage("Android SDK") {
            steps {
                script{
```

```
                        if (fileExists('sdk-tools-linux-
4333796.zip')) {
                                            echo 'Android SDK 已安
装'
                                            } else {
                                            echo '安装 Android SDK'

                                    sh '''
# rm -rf sdk-tools-linux-4333796.* tools platforms platform-
tools
wget https://dl.google.com/android/repository/sdk-tools-linux-
4333796.zip
unzip sdk-tools-linux-4333796.zip
                                            '''


                                    sh 'yes|tools/bin/sdkmanager --
licenses'
                                    //sh 'yes|tools/bin/sdkmanager
"platform-tools" "build-tools;26.0.3" "platforms;android-26"'
// andorid 8.0
                                    //sh 'yes|tools/bin/sdkmanager
"platform-tools" "platforms;android-27"' // andorid 8.1
                                    sh 'yes|tools/bin/sdkmanager
"platform-tools" "platforms;android-28"'        // andorid 9.0
                                    sh '(while sleep 3; do echo
"y"; done) | tools/android update sdk -u'

                                    sh 'tools/bin/sdkmanager --
list'
                            }
                }
                echo '安装 Android SDK License'
                writeFile(file: 'platforms/licenses/android-
sdk-license', text: '''
8933bad161af4178b1185d1a37fbf41ea5269c55
24333f8a63b6825ea9c5514f83c2829b004d1fee
d56f5187479451eabf01fb78af6dfcb131a6481e
                                            ''')
                                    sh 'ls -1 platforms'
            }
        }

        stage("Build") {
            steps {
                echo "构建中..."
```

```
                sh './gradlew'
                echo "构建完成."
            }
        }
        stage("Test") {
            steps {
                echo "单元测试中..."
                sh './gradlew test'
                echo "单元测试完成."
                //junit 'app/build/test-results/*/*.xml'
            }
        }
        stage("Package") {
            steps {
                sh './gradlew assemble'
                // 收集构建产物
                archiveArtifacts artifacts:
'app/build/outputs/apk/*/*.apk', fingerprint: true
            }
        }

        stage("Deploy") {
            steps {
                echo "部署中..."
                // sh './deploy.sh' // 自研部署脚本
                echo "部署完成"
            }
        }
    }
}
```

# 第 13 章 SonarQube

https://www.sonarqube.org

# 1. 安装

## 1.1. Kubernetes 安装 SonarQube

```
import sys, os

sys.path.insert(0, '/Users/neo/workspace/GitHub/devops')
from netkiller.kubernetes import *

namespace = 'default'

service = Service()
service.metadata().name('sonarqube')
service.metadata().namespace(namespace)
service.spec().selector({'app': 'sonarqube'})
service.spec().type('NodePort')
service.spec().ports([{
        'name': 'sonarqube',
        'protocol': 'TCP',
        'port': 80,
        'targetPort': 9000
}])

statefulSet = StatefulSet()
statefulSet.metadata().namespace(namespace)
statefulSet.metadata().name('sonarqube').labels({'app':
'sonarqube'})
statefulSet.spec().replicas(1)
statefulSet.spec().serviceName('sonarqube')
statefulSet.spec().selector({'matchLabels': {'app':
'sonarqube'}})
statefulSet.spec().template().metadata().labels({'app':
'sonarqube'})
```

```
statefulSet.spec().template().spec().containers(
).name('postgresql').image('postgres:latest').ports([{
        'containerPort': 5432
}]).env([
                {'name': 'TZ', 'value': 'Asia/Shanghai'},
                {'name': 'LANG', 'value': 'en_US.UTF-8'},
                {'name': 'POSTGRES_USER', 'value': 'sonar'},
                {'name': 'POSTGRES_PASSWORD', 'value': 'sonar'}
]).volumeMounts([
        {
                'name': 'postgresql',
                'mountPath': '/var/lib/postgresql'
        },
        {
                'name': 'postgresql',
                'mountPath': '/var/lib/postgresql/data',
                'subPath' : 'data'
        },
])

statefulSet.spec().template().spec().containers(
).name('sonarqube').image('sonarqube:community').ports([{
        'containerPort': 9000
}]).env([
                {'name': 'TZ', 'value': 'Asia/Shanghai'},
                {'name': 'LANG', 'value': 'en_US.UTF-8'},
                {'name': 'SONAR_JDBC_URL', 'value':
'jdbc:postgresql://localhost:5432/sonar'},
                {'name': 'SONAR_JDBC_USERNAME', 'value':
'sonar'},
                {'name': 'SONAR_JDBC_PASSWORD', 'value':
'sonar'}
]).resources().livenessProbe().readinessProbe().volumeMounts([
        {
                'name': 'sonarqube',
                'mountPath': '/opt/sonarqube/data',
                'subPath' : 'data'
        },
        {
                'name': 'sonarqube',
                'mountPath': '/opt/sonarqube/extensions',
                'subPath' : 'extensions'
        },
]).securityContext({'privileged': True})
```

```
statefulSet.spec().template().spec().volumes([
        {
        'name': 'sonarqube',
        'persistentVolumeClaim': {
                'claimName': 'sonarqube'
        }
},
        {
        'name': 'postgresql',
        'persistentVolumeClaim': {
                'claimName': 'postgresql'
        }
}
])
statefulSet.spec().volumeClaimTemplates([{
        'metadata':{'name': 'sonarqube'},
        'spec':{
                'accessModes': [ "ReadWriteOnce" ],
                'storageClassName': "local-path",
                'resources':{'requests':{'storage': '2Gi'}}
        }
},{
        'metadata':{'name': 'postgresql'},
        'spec':{
                'accessModes': [ "ReadWriteOnce" ],
                'storageClassName': "local-path",
                'resources':{'requests':{'storage': '2Gi'}}
        }
}
])


ingress = Ingress()
ingress.apiVersion('networking.k8s.io/v1')
ingress.metadata().name('sonarqube')
ingress.metadata().namespace(namespace)
ingress.spec().rules([
{
        'host': 'sonarqube.netkiller.cn',
        'http':{
                'paths': [{
                        'pathType':
Define.Ingress.pathType.Prefix,
                        'path': '/',
                        'backend':{
```

```
                                    'service':{
                                            'name':'sonarqube',
                                            'port':{'number': 80}
                                    }
                            }}]}
}
])

compose = Compose('development')
compose.add(service)
compose.add(statefulSet)
compose.add(ingress)

kubeconfig = '/Users/neo/workspace/kubernetes/office.yaml'
# kubeconfig = os.path.expanduser('~/workspace/ops/k3s.yaml')

kubernetes = Kubernetes(kubeconfig)
kubernetes.compose(compose)
kubernetes.main()
```

连接 sonarqube，注意在容器内部访问 sonarqube 的地址是 sonar.host.url=http://sonarqube.default.svc.cluster.local，如果是外部连接才需要走 ingress sonar.host.url=http://sonarqube.netkiller.cn，还要注意一点 kubernetes service 端口是80 不是 9000

```
sonarqube-check:
        stage: test
        image: registry.netkiller.cn/share/maven:3.8.6-openjdk-
11
        variables:
                # SONAR_USER_HOME: "${CI_PROJECT_DIR}/.sonar"
# Defines the location of the analysis task cache
                GIT_DEPTH: "0"  # Tells git to fetch all the
branches of the project, required by the analysis task
        cache:
                key: "${CI_JOB_NAME}"
                paths:
                - .sonar/cache
        # before_script:
```

```
                # - cat ${MODULE}/pom.xml
        script:
                - mvn -T 1C clean verify sonar:sonar -
Dsonar.projectKey=end-fscs -
Dsonar.host.url=http://sonarqube.default.svc.cluster.local -
Dsonar.login=sqp_d1edb4be69ecc1b3b0ef66f06c4e395822a16a58
        only:
                - office
                - dev
                - test
        tags:
                - kubernetes
        allow_failure: true
```

还有一点需要注意，必须使用 openjdk-11，SonarQube 不支持 Java 1.8

## 1.2. Docker

```
docker volume create --name sonarqube_data
docker volume create --name sonarqube_logs
docker volume create --name sonarqube_extensions

docker run -d --name sonarqube \
    -p 9000:9000 \
    -e
SONAR_JDBC_URL=jdbc:postgresql://db.netkiller.cn:5432/sonar \
    -e SONAR_JDBC_USERNAME=sonar \
    -e SONAR_JDBC_PASSWORD=sonar \
    -v sonarqube_data:/opt/sonarqube/data \
    -v sonarqube_extensions:/opt/sonarqube/extensions \
    -v sonarqube_logs:/opt/sonarqube/logs \
    sonarqube:community
```

Docker compose

```
version: "3"

services:
  sonarqube:
    container_name: sonarqube
    image: sonarqube:community
    restart: always
    depends_on:
      - db
    environment:
      SONAR_JDBC_URL: jdbc:postgresql://db:5432/sonar
      SONAR_JDBC_USERNAME: sonar
      SONAR_JDBC_PASSWORD: sonar
    volumes:
      - sonarqube_data:/opt/sonarqube/data
      - sonarqube_extensions:/opt/sonarqube/extensions
      - sonarqube_logs:/opt/sonarqube/logs
    ports:
      - "9000:9000"
  db:
    container_name: postgresql
    image: postgres:latest
    restart: always
    environment:
      POSTGRES_USER: sonar
      POSTGRES_PASSWORD: sonar
    volumes:
      - postgresql:/var/lib/postgresql
      - postgresql_data:/var/lib/postgresql/data

volumes:
  sonarqube_data:
  sonarqube_extensions:
  sonarqube_logs:
  postgresql:
  postgresql_data:
```

/etc/sysctl.conf 增加配置项，否则无法启动 sonarqube，提示 sonarqube | bootstrap check failure [1] of [1]: max virtual memory areas vm.max_map_count [65530] is too low, increase to at least [262144]

```
vm.max_map_count=655360
```

## 1.3. netkiller-devops 安装

```
pip install netkiller-devops
```

创建 sonarqube 文件

```
#!/usr/bin/env python3
from netkiller.docker import *

projectVolume = Volumes()
projectVolume.create('sonarqube_data')
projectVolume.create('sonarqube_extensions')
projectVolume.create('sonarqube_logs')
projectVolume.create('postgresql')
projectVolume.create('postgresql_data')
# projectVolume.create('')

sonarqube = Services('sonarqube')
sonarqube.container_name('sonarqube').image('sonarqube:commun
ity').restart('always').ports("9000:9000")
sonarqube.environment([

'SONAR_JDBC_URL=jdbc:postgresql://postgresql:5432/sonar',
        'SONAR_JDBC_USERNAME=sonar',
        'SONAR_JDBC_PASSWORD=sonar'
]).volumes([
        'sonarqube_data:/opt/sonarqube/data',
    'sonarqube_extensions:/opt/sonarqube/extensions',
    'sonarqube_logs:/opt/sonarqube/logs'
]).depends_on('postgresql')
```

```
postgresql = Services('postgresql')
postgresql.container_name('postgresql').image('postgres:lates
t').restart('always')
postgresql.environment([
        'POSTGRES_USER=sonar',
    'POSTGRES_PASSWORD=sonar'
]).volumes([
        'postgresql:/var/lib/postgresql',
    'postgresql_data:/var/lib/postgresql/data'
])

project = Composes('project')
project.version('3.9')
project.volumes(projectVolume)
project.services(sonarqube)
project.services(postgresql)

if __name__ == '__main__':
        try:
                docker = Docker()
                docker.environment(project)
                docker.main()
        except KeyboardInterrupt:
                print ("Crtl+C Pressed. Shutting down.")
```

## 1.4. SonarScanner

### Docker 安装

```
docker run \
    --rm \
    -e SONAR_HOST_URL="http://${SONARQUBE_URL}" \
    -e SONAR_LOGIN="myAuthenticationToken" \
    -v "${YOUR_REPO}:/usr/src" \
    sonarsource/sonar-scanner-cli
```

## 本地安装

SonarQube 必须使用 Java 11

```
[root@localhost ~]# dnf install java-11-openjdk java-11-
openjdk-devel
```

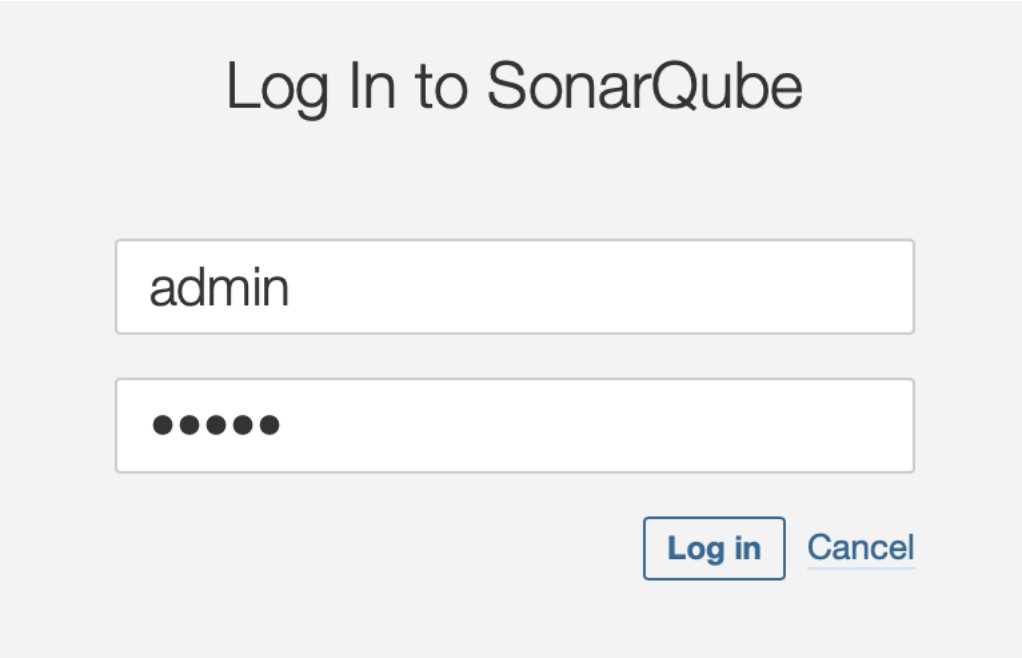
安装 SonarScanner

```
```

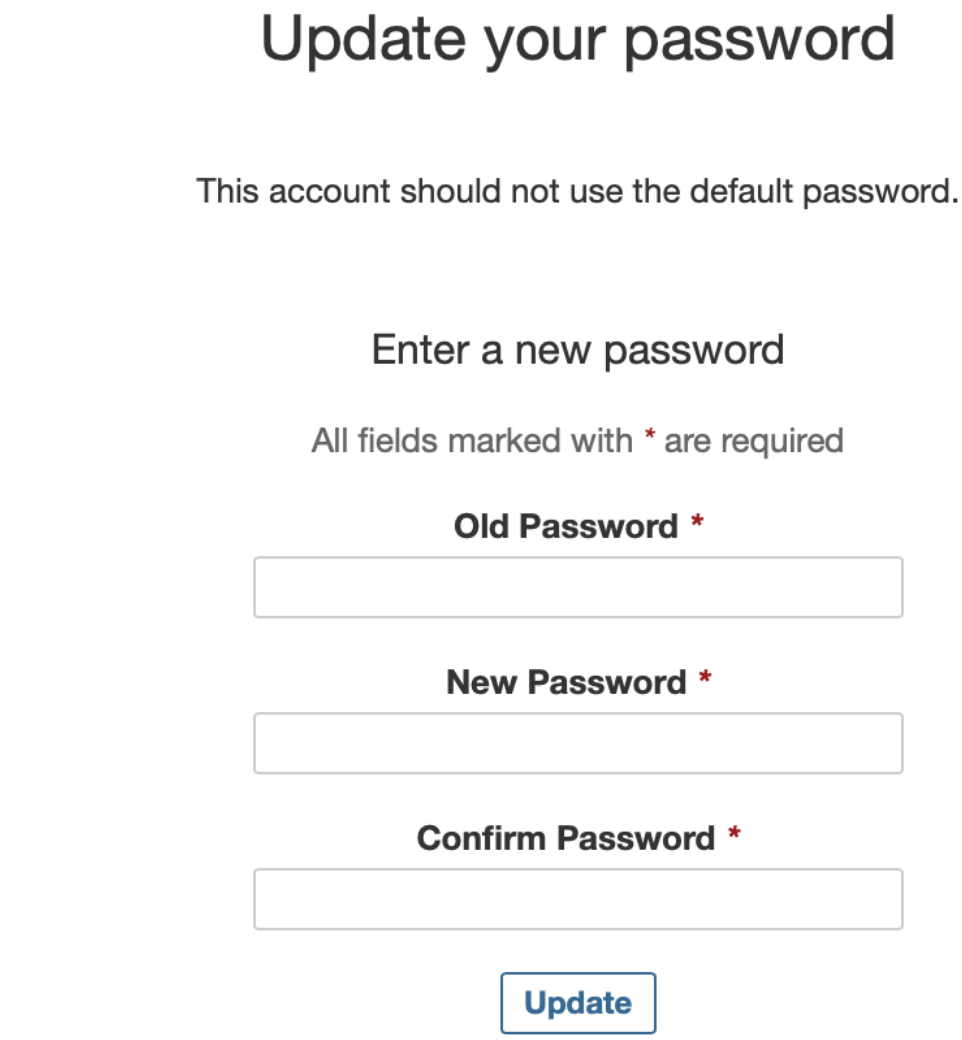
# 2. 配置

## 2.1. 登陆 SonarQube

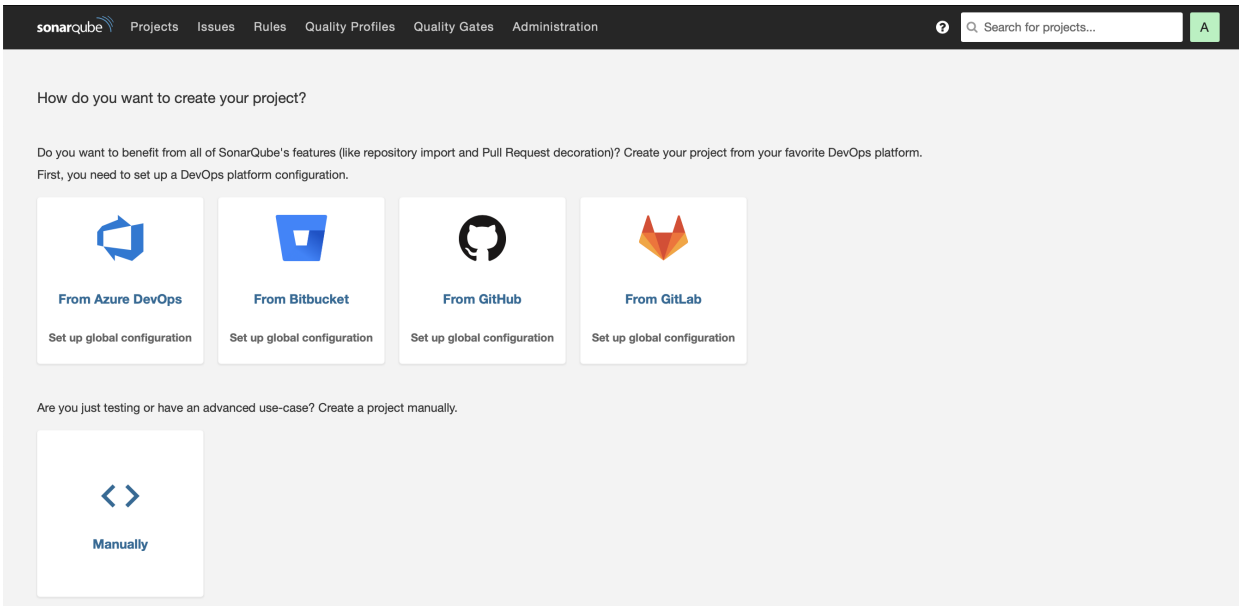
登陆 SonarQube，默认用户：admin， 密码：admin http://localhost:9000




首次登陆会提示修改密码

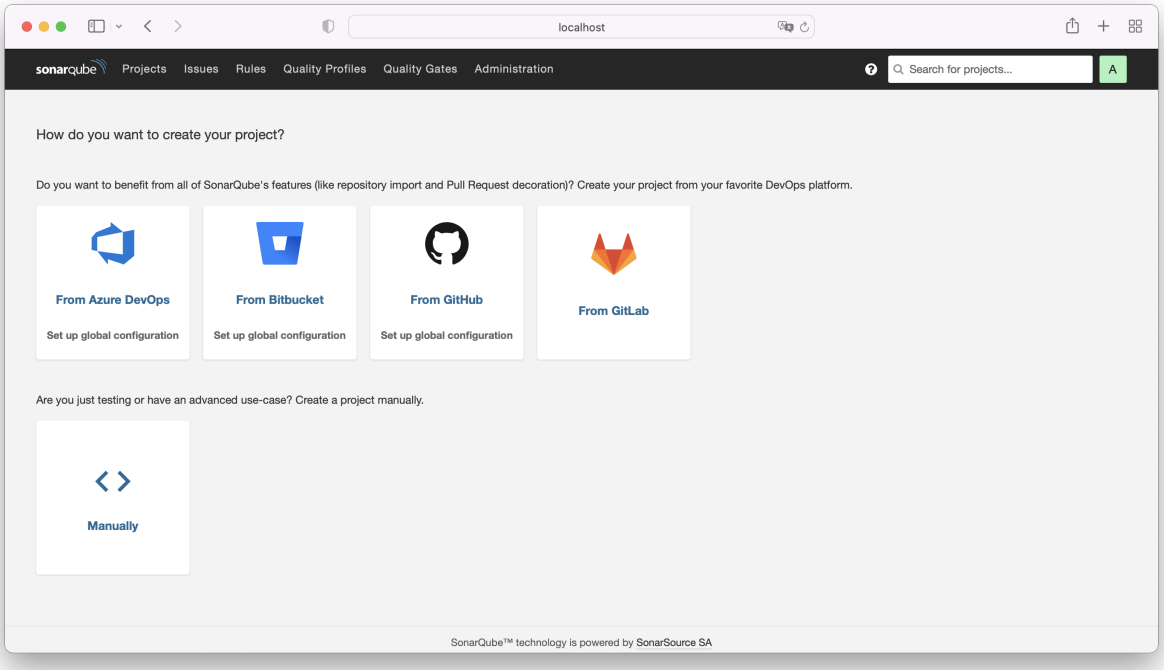

Update your password
This account should not use the default password.
Enter a new password
All fields marked with * are required
Old Password *
New Password *
Confirm Password *
Update

登陆成功

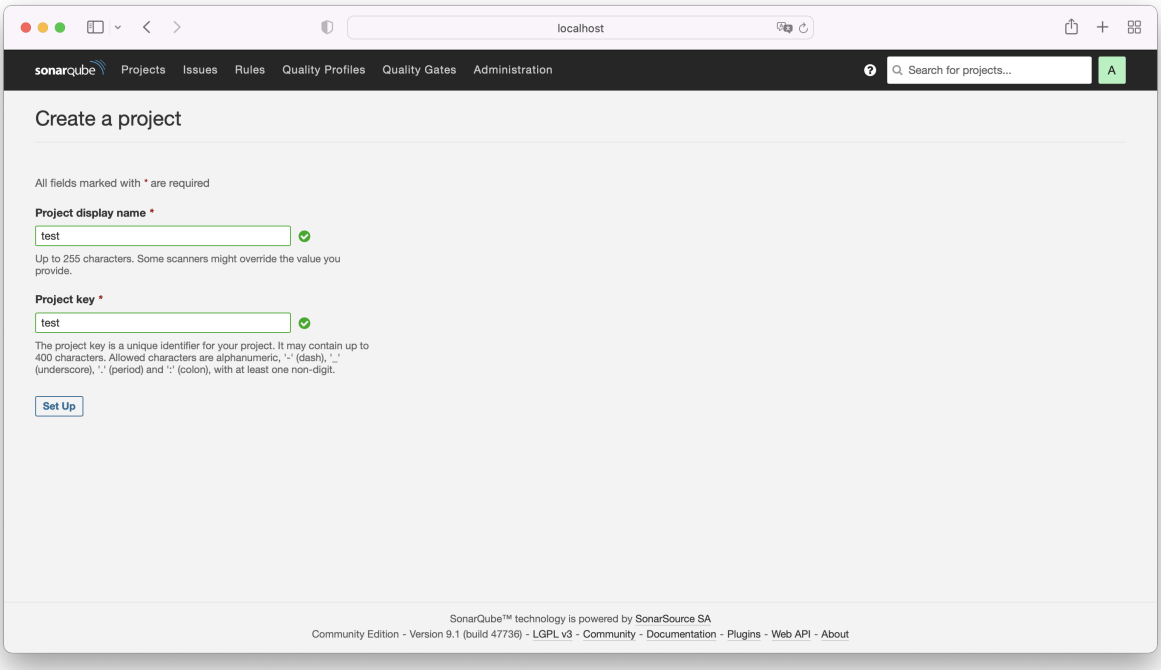



## 2.2. 本地 maven 执行 SonarQube

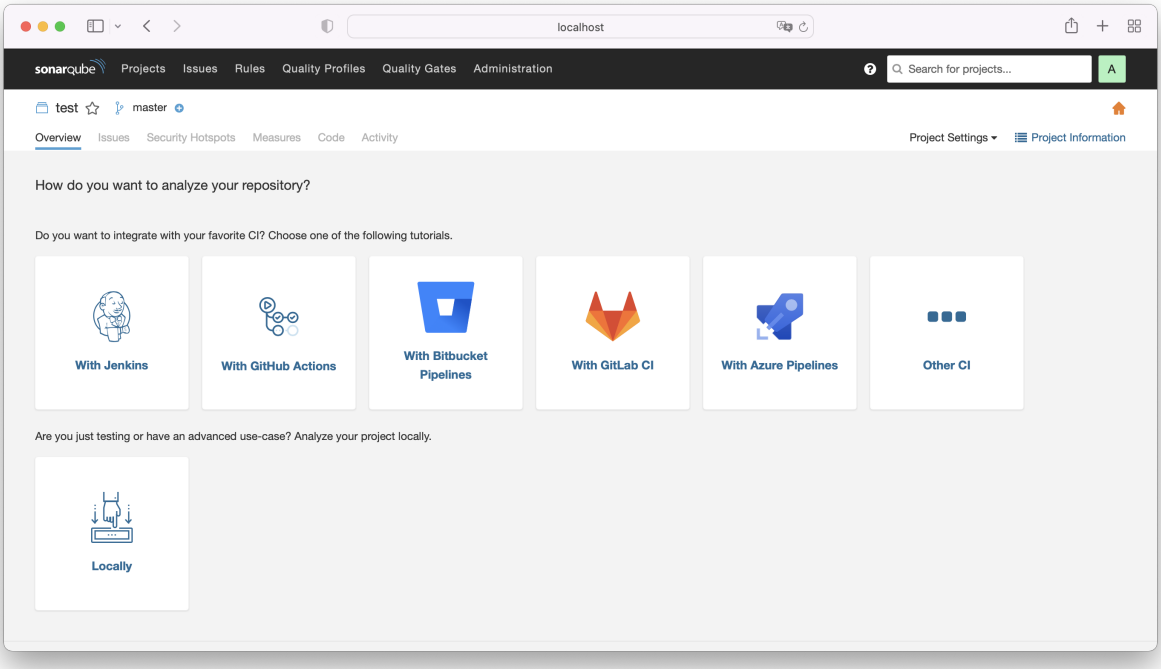
手工创建一个项目



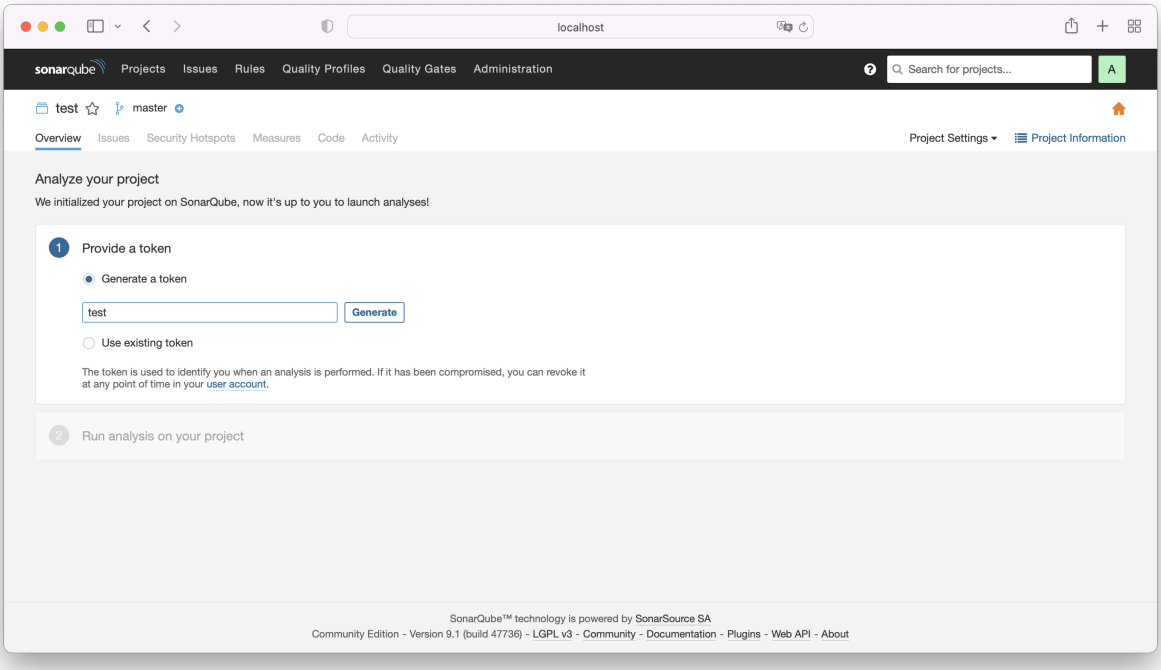

输入项目名称和密钥，然后点击“Set Up” 按钮



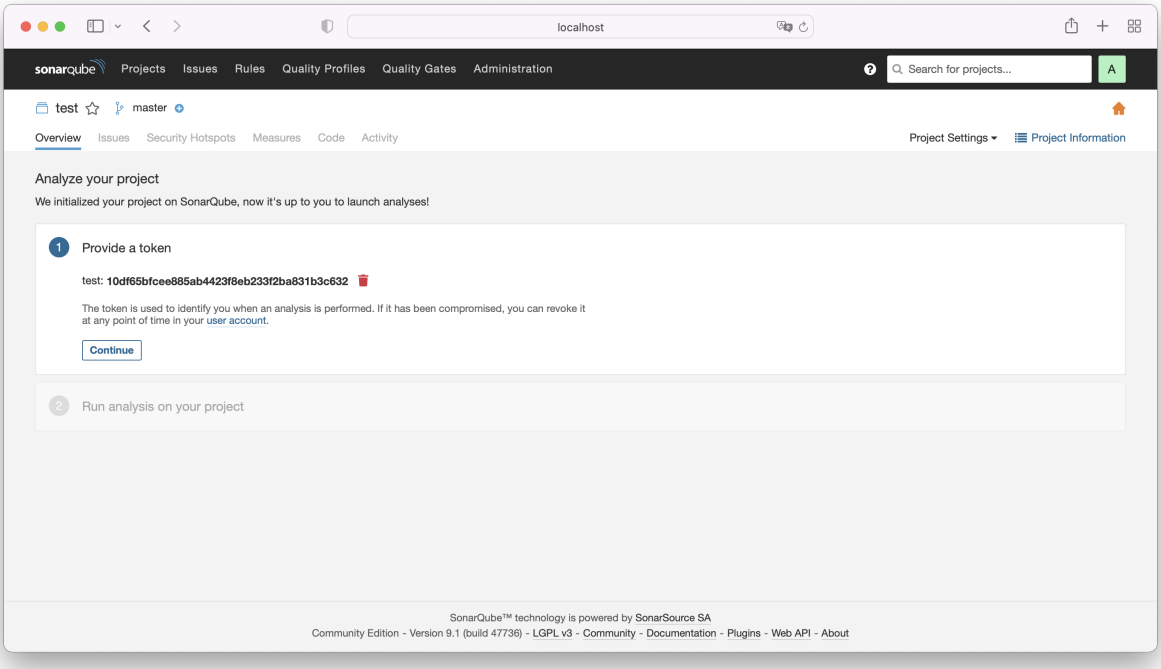

点击 "Locally" 分析本地项目



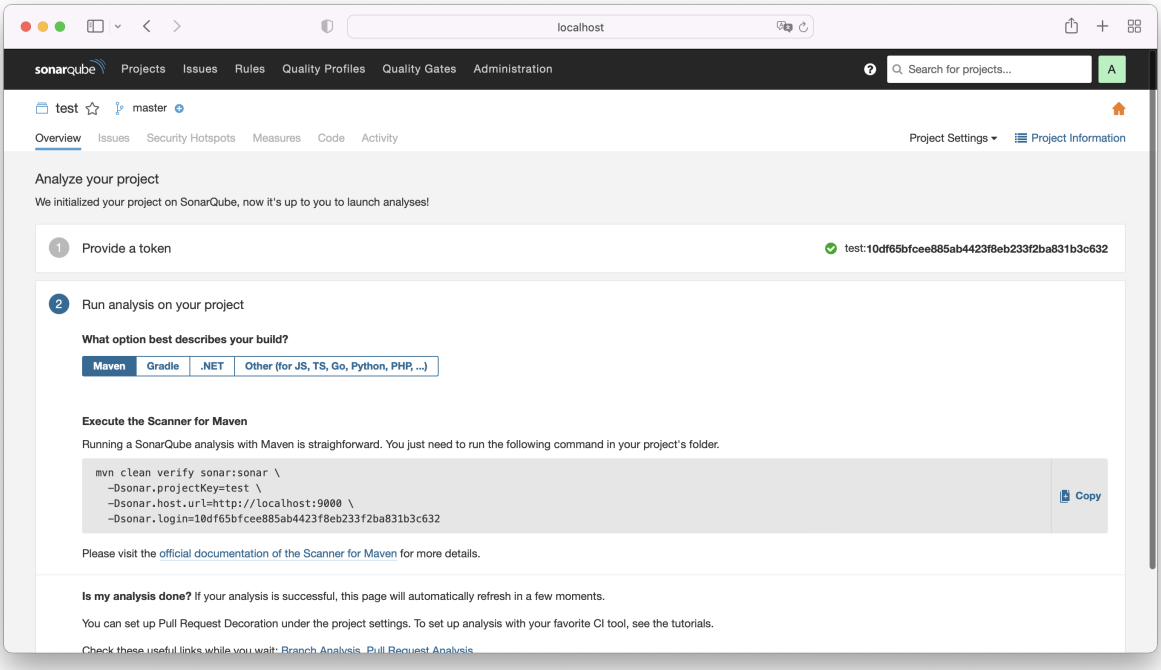

输入项目名称，点击“Generate”按钮生成 Token



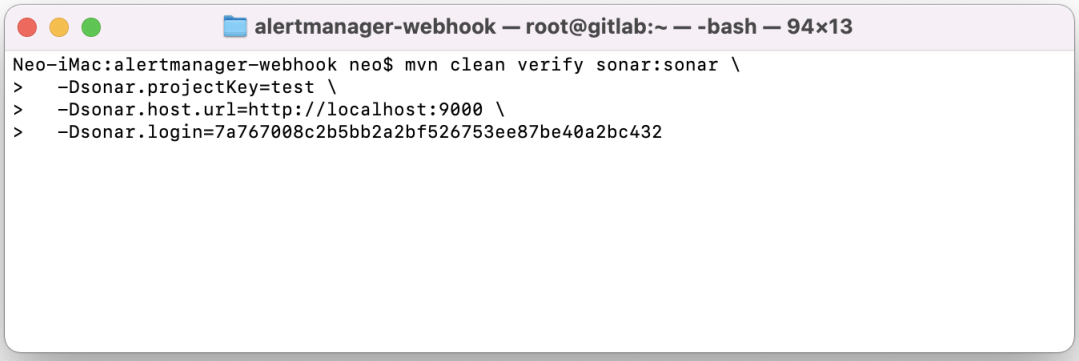

将 Token 保存好，然后点击 “Continue” 按钮继续




选择你的构建方式，我使用的是 Maven




复制 Maven 命令，然后在你的项目下面执行。



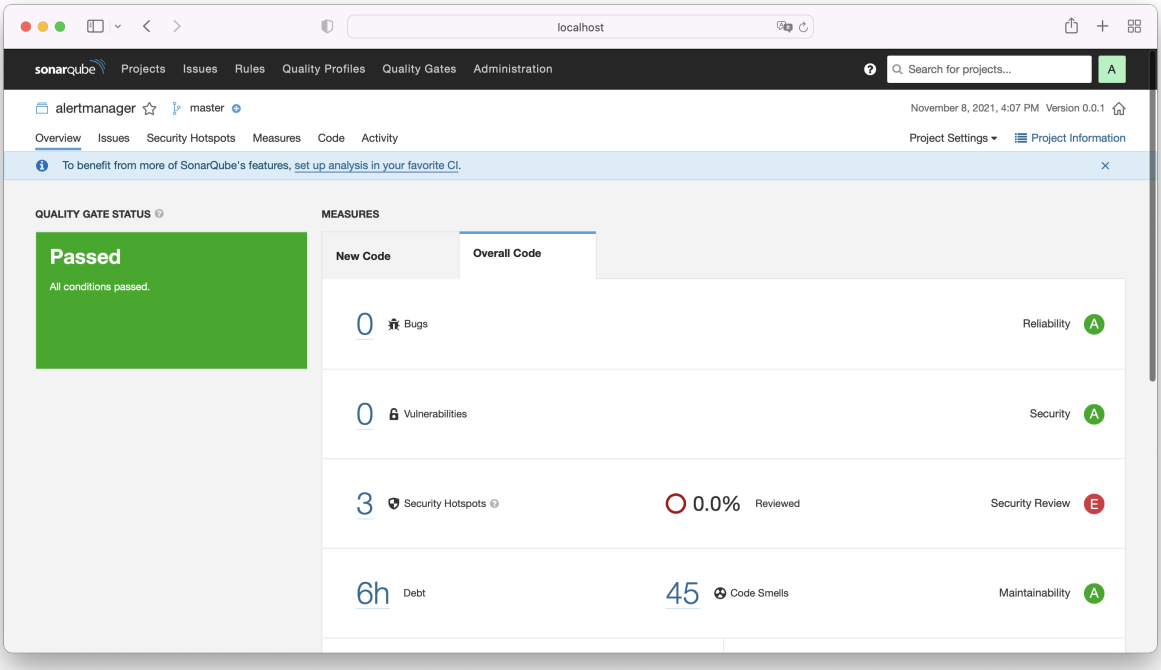

```
mvn clean verify sonar:sonar \
  -Dsonar.projectKey=test \
  -Dsonar.host.url=http://192.168.30.20:9000 \
  -Dsonar.login=e4294feaa6e9f830bdb109a310de6cd59f3a0443
```

执行会输出下面信息

```
[INFO] ----------------------< cn.netkiller:alertmanager >----------------------
[INFO] Building alertmanager 0.0.1
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- sonar-maven-plugin:3.9.0.2155:sonar (default-cli) @ alertmanager ---
[INFO] User cache: /Users/neo/.sonar/cache
[INFO] SonarQube version: 9.1.0
[INFO] Default locale: "en_CN", source code encoding: "UTF-8"
[INFO] Load global settings
[INFO] Load global settings (done) | time=199ms
[INFO] Server id: 243B8A4D-AXz9icqihL5ZxuJK9yra
[INFO] User cache: /Users/neo/.sonar/cache
[INFO] Load/download plugins
[INFO] Load plugins index
[INFO] Load plugins index (done) | time=81ms
[INFO] Load/download plugins (done) | time=316ms
[INFO] Process project properties
[INFO] Process project properties (done) | time=13ms
[INFO] Execute project builders
[INFO] Execute project builders (done) | time=1ms
[INFO] Project key: test
[INFO] Base dir: /Users/neo/workspace/alertmanager-webhook
[INFO] Working dir: /Users/neo/workspace/alertmanager-webhook/target/sonar
[INFO] Load project settings for component key: 'test'
[INFO] Load project settings for component key: 'test' (done) | time=58ms
[INFO] Load quality profiles
[INFO] Load quality profiles (done) | time=203ms
[INFO] Load active rules
[INFO] Load active rules (done) | time=5861ms
[INFO] Indexing files...
[INFO] Project configuration:
[INFO] 7 files indexed
[INFO] 0 files ignored because of scm ignore settings
[INFO] Quality profile for java: Sonar way
[INFO] Quality profile for xml: Sonar way
[INFO] ------------- Run sensors on module alertmanager
[INFO] Load metrics repository
[INFO] Load metrics repository (done) | time=67ms
[INFO] Sensor JavaSensor [java]
[INFO] Configured Java source version (sonar.java.source): 17
[INFO] JavaClasspath initialization
[INFO] JavaClasspath initialization (done) | time=10ms
[INFO] JavaTestClasspath initialization
[INFO] JavaTestClasspath initialization (done) | time=1ms
[INFO] Java "Main" source files AST scan
[INFO] 5 source files to be analyzed
[INFO] Load project repositories
[INFO] Load project repositories (done) | time=63ms
[INFO] 5/5 source files have been analyzed
[INFO] Java "Main" source files AST scan (done) | time=2271ms
[INFO] Java "Test" source files AST scan
[INFO] 1 source file to be analyzed
[INFO] 1/1 source file has been analyzed
[INFO] Java "Test" source files AST scan (done) | time=41ms
[INFO] No "Generated" source files to scan.
[INFO] Sensor JavaSensor [java] (done) | time=2833ms
[INFO] Sensor CSS Rules [cssfamily]
[INFO] No CSS, PHP, HTML or VueJS files are found in the project. CSS analysis is skipped.
[INFO] Sensor CSS Rules [cssfamily] (done) | time=1ms
[INFO] Sensor JaCoCo XML Report Importer [jacoco]
[INFO] 'sonar.coverage.jacoco.xmlReportPaths' is not defined. Using default locations:
target/site/jacoco/jacoco.xml,target/site/jacoco-
it/jacoco.xml,build/reports/jacoco/test/jacocoTestReport.xml
[INFO] No report imported, no coverage information will be imported by JaCoCo XML Report
Importer
```

```
[INFO] Sensor JaCoCo XML Report Importer [jacoco] (done) | time=2ms
[INFO] Sensor C# Project Type Information [csharp]
[INFO] Sensor C# Project Type Information [csharp] (done) | time=0ms
[INFO] Sensor C# Analysis Log [csharp]
[INFO] Sensor C# Analysis Log [csharp] (done) | time=55ms
[INFO] Sensor C# Properties [csharp]
[INFO] Sensor C# Properties [csharp] (done) | time=0ms
[INFO] Sensor SurefireSensor [java]
[INFO] parsing [/Users/neo/workspace/alertmanager-webhook/target/surefire-reports]
[INFO] Sensor SurefireSensor [java] (done) | time=2ms
[INFO] Sensor JavaXmlSensor [java]
[INFO] 1 source file to be analyzed
[INFO] 1/1 source file has been analyzed
[INFO] Sensor JavaXmlSensor [java] (done) | time=201ms
[INFO] Sensor HTML [web]
[INFO] Sensor HTML [web] (done) | time=2ms
[INFO] Sensor XML Sensor [xml]
[INFO] 1 source file to be analyzed
[INFO] 1/1 source file has been analyzed
[INFO] Sensor XML Sensor [xml] (done) | time=179ms
[INFO] Sensor VB.NET Project Type Information [vbnet]
[INFO] Sensor VB.NET Project Type Information [vbnet] (done) | time=14ms
[INFO] Sensor VB.NET Analysis Log [vbnet]
[INFO] Sensor VB.NET Analysis Log [vbnet] (done) | time=42ms
[INFO] Sensor VB.NET Properties [vbnet]
[INFO] Sensor VB.NET Properties [vbnet] (done) | time=0ms
[INFO] ------------- Run sensors on project
[INFO] Sensor Zero Coverage Sensor
[INFO] Sensor Zero Coverage Sensor (done) | time=23ms
[INFO] Sensor Java CPD Block Indexer
[INFO] Sensor Java CPD Block Indexer (done) | time=23ms
[INFO] SCM Publisher SCM provider for this project is: git
[INFO] SCM Publisher 7 source files to be analyzed
[INFO] SCM Publisher 7/7 source files have been analyzed (done) | time=169ms
[INFO] CPD Executor 1 file had no CPD blocks
[INFO] CPD Executor Calculating CPD for 4 files
[INFO] CPD Executor CPD calculation finished (done) | time=7ms
[INFO] Analysis report generated in 56ms, dir size=142.8 kB
[INFO] Analysis report compressed in 60ms, zip size=34.4 kB
[INFO] Analysis report uploaded in 121ms
[INFO] ------------- Check Quality Gate status
[INFO] Waiting for the analysis report to be processed (max 300s)
[INFO] QUALITY GATE STATUS: PASSED - View details on http://localhost:9000/dashboard?id=test
[INFO] Analysis total time: 23.392 s
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  01:15 min
[INFO] Finished at: 2021-11-08T15:21:59+08:00
[INFO] ------------------------------------------------------------------------
```

Maven 执行完成之后 SonarQube 会自动展示分析结果




这种方式需要手工执行 Maven，每次都需要指定三个参数，-Dsonar.projectKey=test -Dsonar.host.url=http://192.168.30.20:9000 -Dsonar.login=e4294feaa6e9f830bdb109a310de6cd59f3a0443，有没有更好的解决方案呢?

我们可以将这些参数写入到 setting.xml / pom.xml 文件，方法如下:

project/build/plugins 下面增加 sonar-maven-plugin

```
                    <plugin>
                            <groupId>org.sonarsource.scanner.maven</groupId>
                            <artifactId>sonar-maven-plugin</artifactId>
                            <version>3.9.0.2155</version>
                    </plugin>
```

project/profiles 下面增加 sonar，profile 有两种写法，一种是使用用户名和密码，另一种是使用token

```
                <profile>
                        <id>sonar</id>
                        <activation>
                                <activeByDefault>true</activeByDefault>
                        </activation>
                        <properties>
                                <!-- Optional URL to server. Default value is
http://localhost:9000 -->
                                <sonar.host.url>http://localhost:9000</sonar.host.url>
                                <sonar.login>admin</sonar.login>
                                <sonar.password>your_password</sonar.password>
                                <!-- <sonar.inclusions>**/*.java,**/*.xml</sonar.inclusions> --
```

```
>
                                <!-- <sonar.exclusions>**/cn/netkiller/test/*
</sonar.exclusions> -->
                        </properties>
                </profile>

                <profile>
                        <id>sonar</id>
                        <activation>
                                <activeByDefault>true</activeByDefault>
                        </activation>
                        <properties>
                                <!-- Optional URL to server. Default value is
http://localhost:9000 -->
                                <sonar.host.url>http://localhost:9000</sonar.host.url>

<sonar.login>510966107d69cd32448fcc4372d1383e8d21092b</sonar.login>
                                <sonar.password></sonar.password>
                        </properties>
                </profile>
```

配置完成之后使用 mvn verify sonar:sonar 测试

```
Neo-iMac:microservice neo$ mvn verify sonar:sonar -Dmaven.test.skip=true
```

下面是完整的例子

**例 13.1. SonarQube pom.xml 配置**

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>

        <groupId>cn.netkiller</groupId>
        <artifactId>microservice</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <packaging>pom</packaging>

        <name>microservice</name>
        <url>http://www.netkiller.cn</url>
        <description>Demo project for Spring Boot</description>

        <organization>
                <name>Netkiller Spring Cloud 手札</name>
                <url>http://www.netkiller.cn</url>
        </organization>

        <developers>
                <developer>
                        <name>Neo</name>
                        <email>netkiller@msn.com</email>
                        <organization>Netkiller Spring Cloud 手札</organization>
                        <organizationUrl>http://www.netkiller.cn</organizationUrl>
                        <roles>
                                <role>Author</role>
                        </roles>
```

```
        </developer>
    </developers>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
        <java.version>17</java.version>
        <maven.compiler.source>${java.version}</maven.compiler.source>
        <maven.compiler.target>${java.version}</maven.compiler.target>
        <maven.compiler.release>${java.version}</maven.compiler.release>
        <spring-boot.version>2.4.0.RELEASE</spring-boot.version>
        <spring-cloud.version>2020.0.4</spring-cloud.version>
        <!-- <docker.registry>127.0.0.1:5000</docker.registry> -->
        <docker.registry>registry.netkiller.cn:5000</docker.registry>
        <docker.registry.name>netkiller</docker.registry.name>
        <docker.image.prefix>netkiller</docker.image.prefix>
        <docker.image>mcr.microsoft.com/java/jre:15-zulu-alpine</docker.image>
    </properties>

    <repositories>
        <repository>
            <id>alimaven</id>
            <name>Maven Aliyun Mirror</name>
            <url>http://maven.aliyun.com/nexus/content/repositories/central/</url>
            <releases>
                <enabled>true</enabled>
            </releases>
            <snapshots>
                <enabled>false</enabled>
            </snapshots>
        </repository>
    </repositories>

    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.5.6</version>
        <relativePath />
    </parent>

    <dependencyManagement>
        <dependencies>
            <dependency>
                <groupId>org.springframework.cloud</groupId>
                <artifactId>spring-cloud-dependencies</artifactId>
                <version>${spring-cloud.version}</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
        </dependencies>
    </dependencyManagement>

    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-actuator</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <scope>test</scope>
        </dependency>
```

```
                <dependency>
                        <groupId>org.projectlombok</groupId>
                        <artifactId>lombok</artifactId>
                </dependency>
        </dependencies>

        <modules>
                <module>eureka</module>
                <module>gateway</module>
                <module>config</module>
                <module>webflux</module>
                <module>openfeign</module>
                <module>restful</module>
                <module>sleuth</module>
                <module>oauth2</module>
                <module>welcome</module>
                <module>test</module>
                <module>aliyun</module>
        </modules>

        <profiles>
                <profile>
                        <id>dev</id>
                        <properties>
                                <profiles.active>dev</profiles.active>
                        </properties>
                        <activation>
                                <activeByDefault>true</activeByDefault>
                        </activation>
                </profile>
                <profile>
                        <id>prod</id>
                        <properties>
                                <profiles.active>prod</profiles.active>
                        </properties>
                </profile>
                <profile>
                        <id>test</id>
                        <properties>
                                <profiles.active>test</profiles.active>
                        </properties>
                </profile>

                <profile>
                        <id>sonar</id>
                        <activation>
                                <activeByDefault>true</activeByDefault>
                        </activation>
                        <properties>
                                <!-- Optional URL to server. Default value is
http://localhost:9000 -->
                                <sonar.host.url>http://localhost:9000</sonar.host.url>
                                <sonar.login>admin</sonar.login>
                                <sonar.password>******</sonar.password>
                                <!-- <sonar.inclusions>**/*.java,**/*.xml</sonar.inclusions> --
>
                                <!-- <sonar.exclusions>**/cn/netkiller/test/*
</sonar.exclusions> -->
                        </properties>
                </profile>

        </profiles>

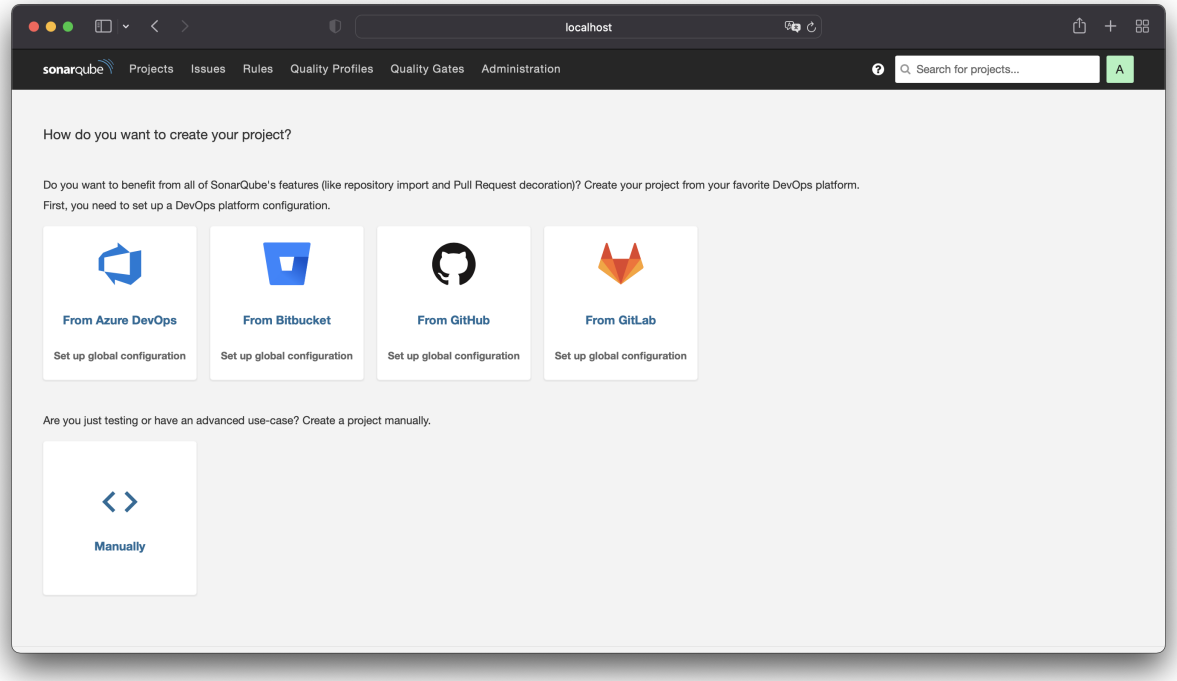
        <build>
                <plugins>
                        <plugin>
                                <artifactId>maven-surefire-plugin</artifactId>
```

```
                        <configuration>
                                <skip>true</skip>
                        </configuration>
                </plugin>
                <plugin>
                        <groupId>org.sonarsource.scanner.maven</groupId>
                        <artifactId>sonar-maven-plugin</artifactId>
                        <version>3.9.0.2155</version>
                </plugin>
                <plugin>
                        <groupId>org.jacoco</groupId>
                        <artifactId>jacoco-maven-plugin</artifactId>
                        <version>0.8.7</version>
                        <executions>
                                <execution>
                                        <goals>
                                                <goal>prepare-agent</goal>
                                        </goals>
                                </execution>
                                <execution>
                                        <id>report</id>
                                        <phase>test</phase>
                                        <goals>
                                                <goal>report</goal>
                                        </goals>
                                </execution>
                        </executions>
                </plugin>
        </plugins>
    </build>

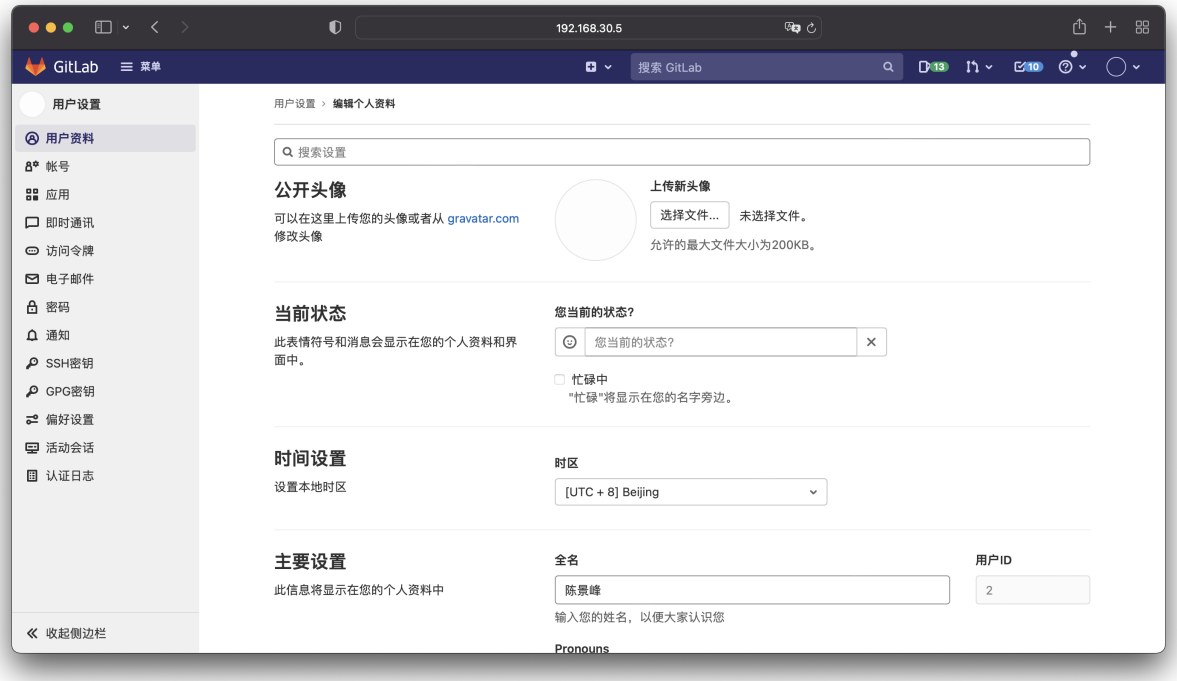
</project>
```

### 2.3. 集成 Gitlab

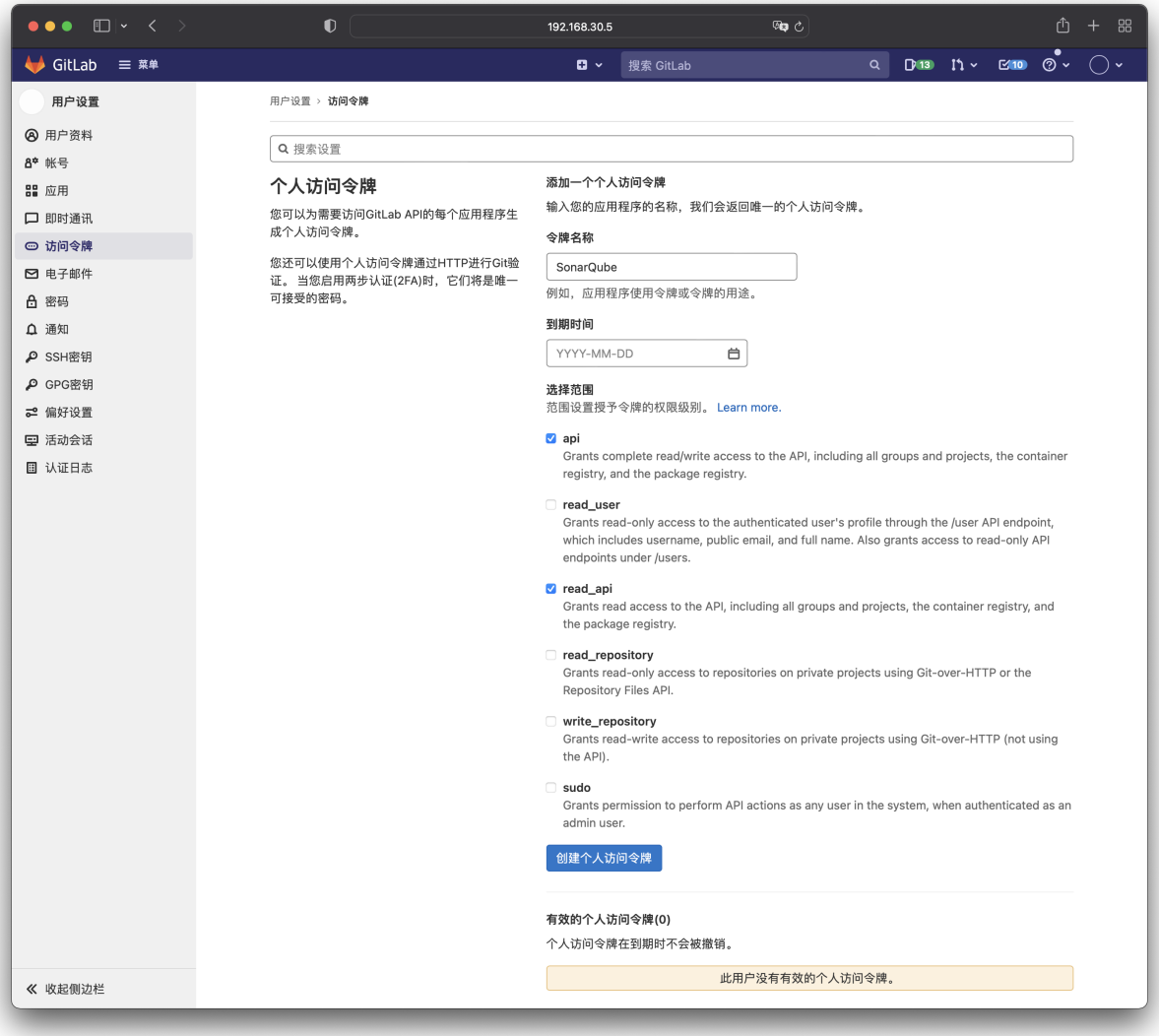
创建项目



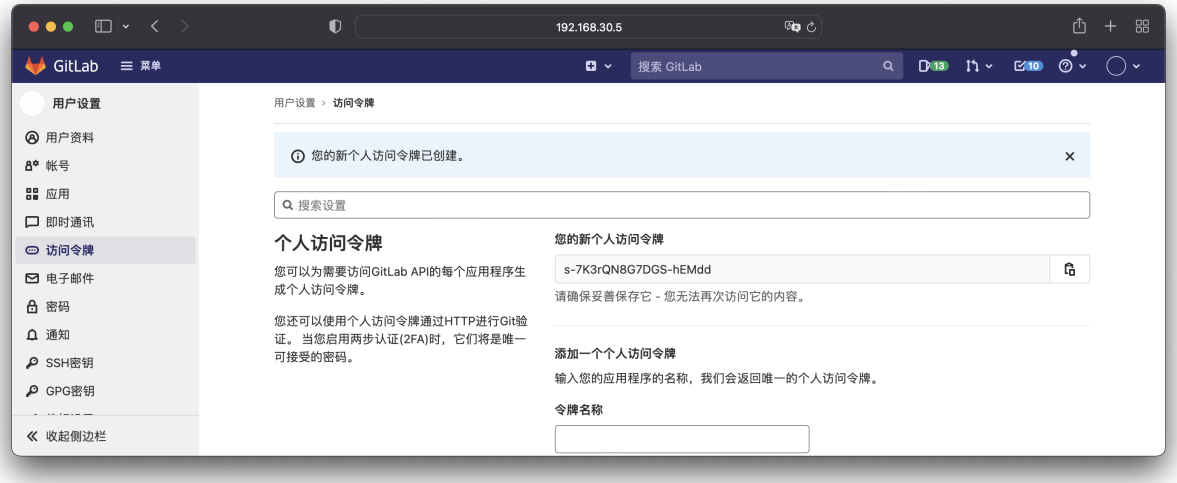

选择 “From GitLab”，现在切换到 Gitlab，进入用户设置



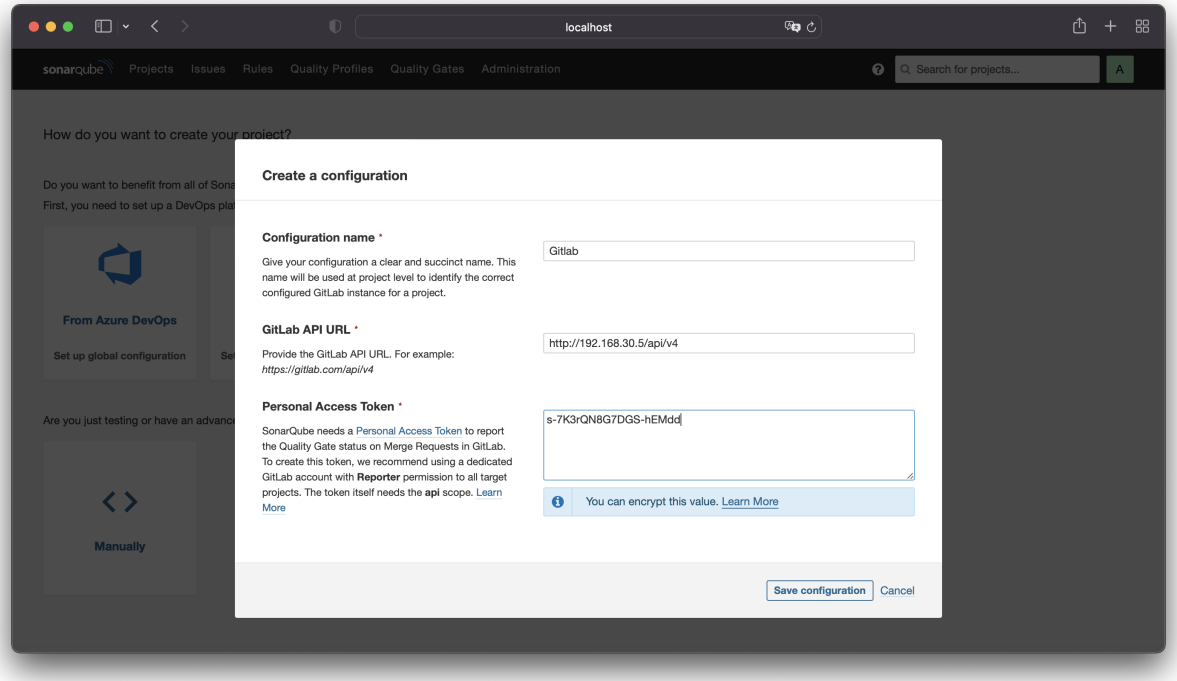

选择访问令牌



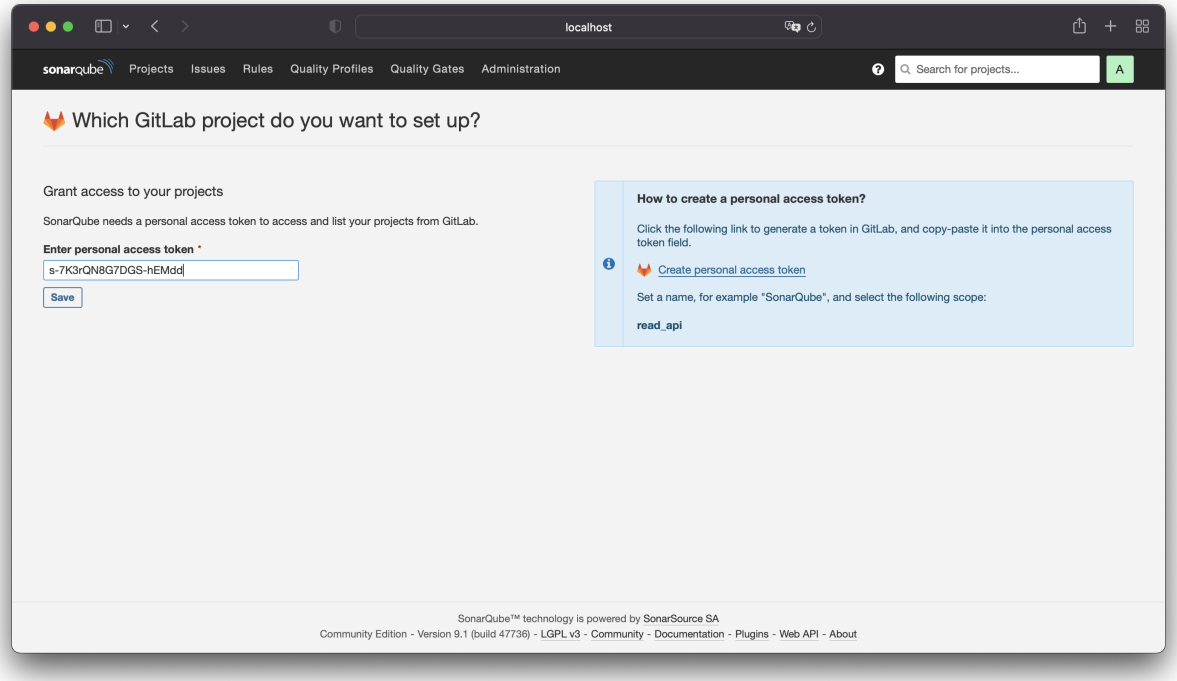

输入令牌名称，勾选 api 和 read_api，最后点击“创建个人访问令牌”按钮



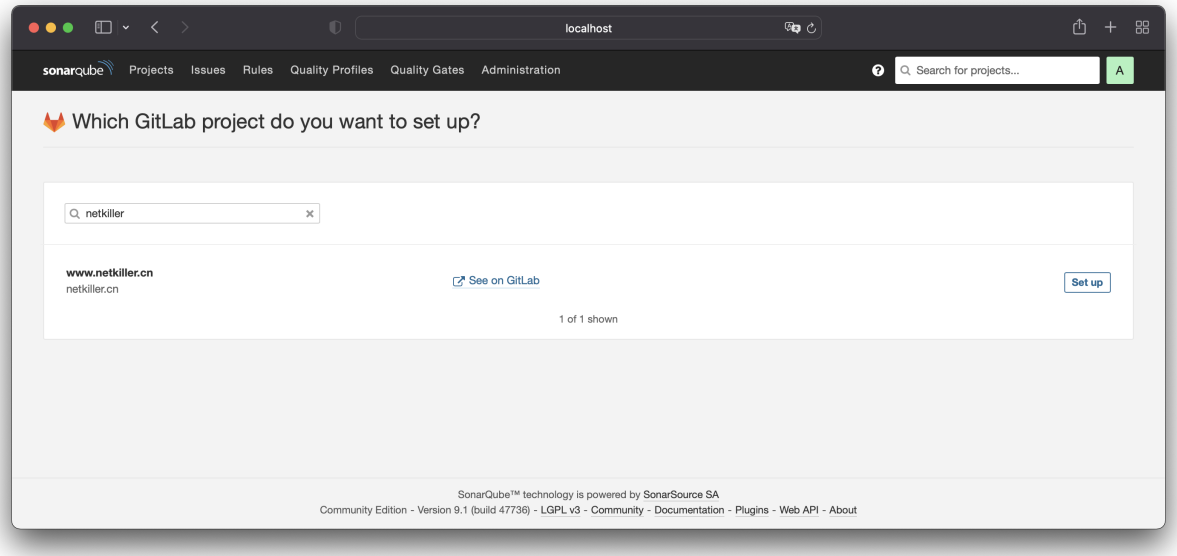

复制“您的新个人访问令牌”



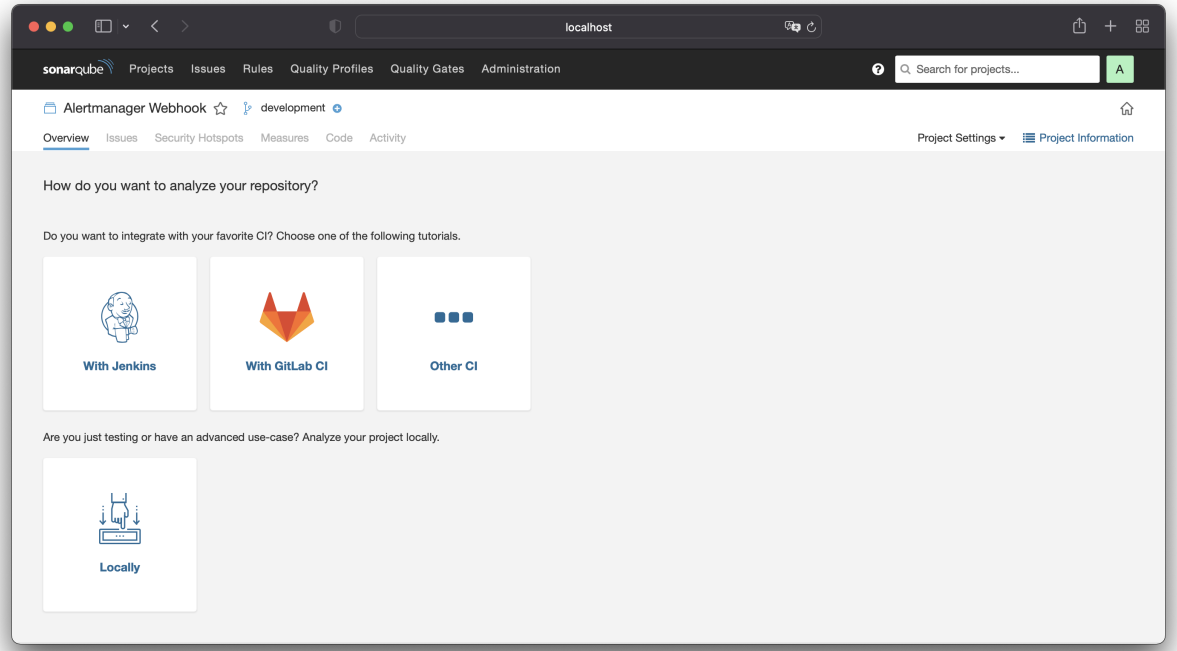

回到 SonarQube，输入配置名称 Configuration name，GitLab API URL和Personal Access Token (Gitlab中创建的个人访问令牌)



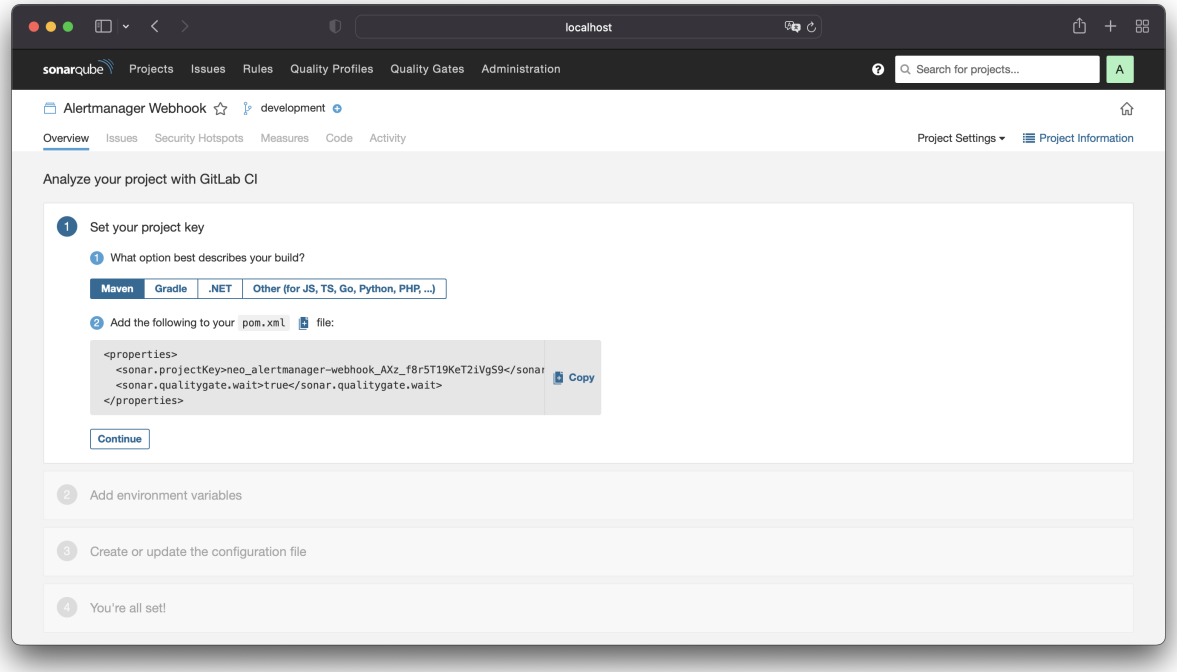

再次输入个人访问令牌




如果令牌正确，将会看到 Gitlab 那边的项目列表，如果项目很多，可以在查询框内输入关键字查找，选择你需要扫描的项目，点击“Set up” 按钮




选择 With GitLab CI




选择 Maven，复制配置项，添加到 Maven 的 pom.xml 中，配置类似下面

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>

        <groupId>com.example</groupId>
        <artifactId>demo</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <packaging>jar</packaging>

        <name>demo</name>
        <description>Demo project for Spring Boot</description>

        <parent>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-parent</artifactId>
                <version>2.0.1.RELEASE</version>
                <relativePath/> <!-- lookup parent from repository -->
        </parent>

        <properties>
                <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
                <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
                <java.version>1.8</java.version>


                <sonar.projectKey>api.netkiller.cn_AXz_oa0aOCAK34bOh_gg</sonar.projectKey>
                <sonar.qualitygate.wait>true</sonar.qualitygate.wait>

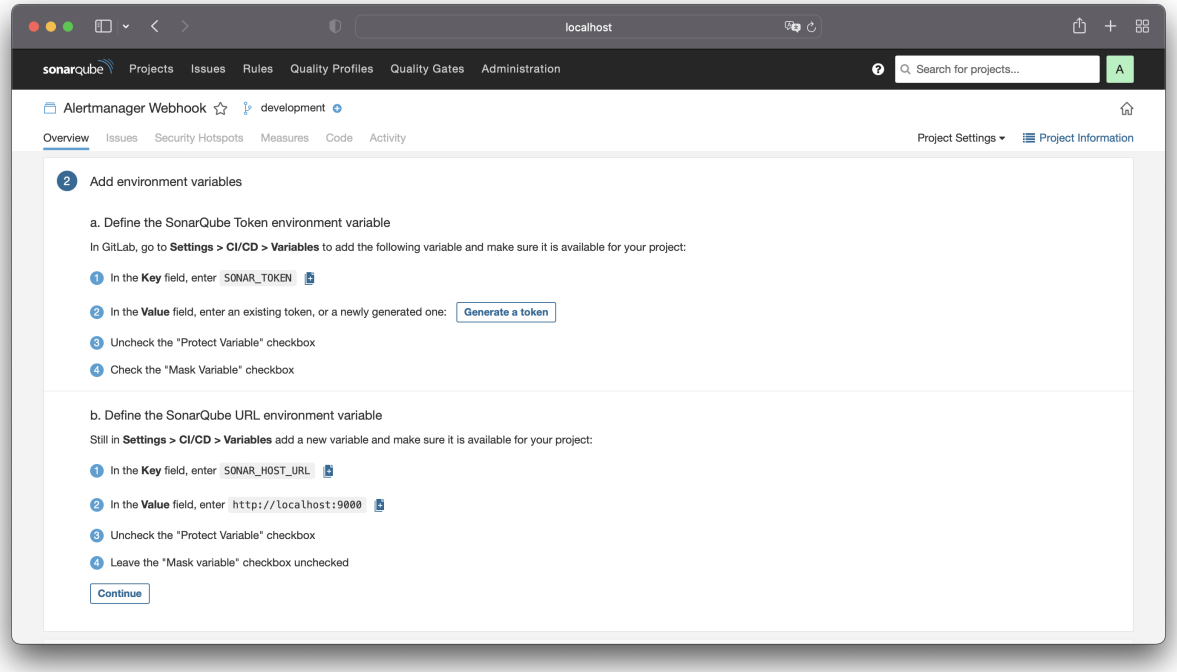
        </properties>

        <dependencies>
                <dependency>
                        <groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-web</artifactId>
                </dependency>

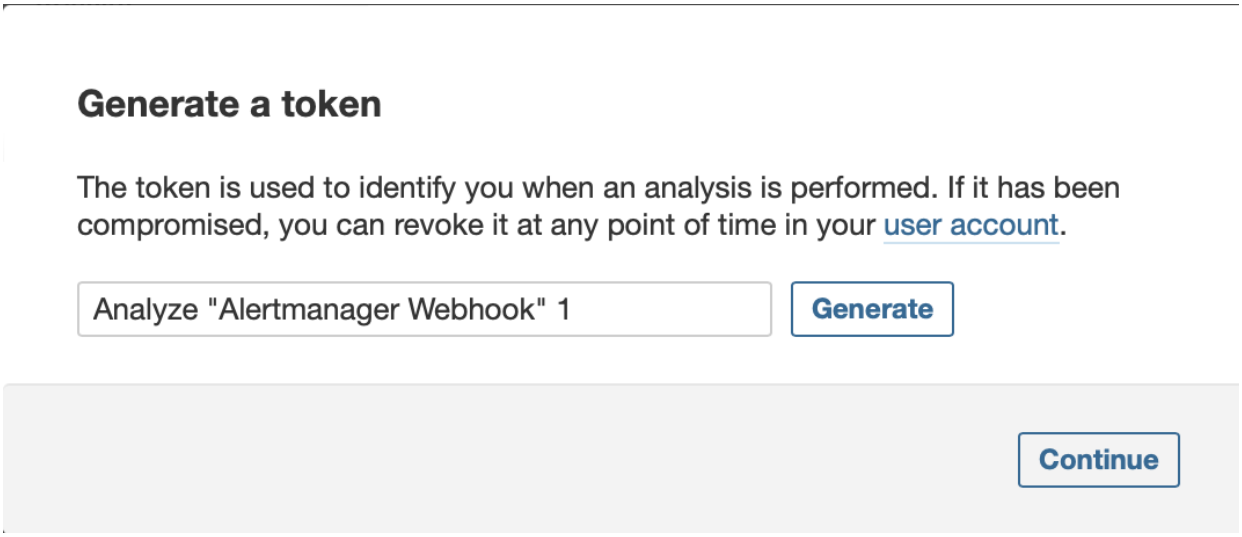
                <dependency>
                        <groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-test</artifactId>
                        <scope>test</scope>
                </dependency>
        </dependencies>

        <build>
                <plugins>
                        <plugin>
                                <groupId>org.springframework.boot</groupId>
                                <artifactId>spring-boot-maven-plugin</artifactId>
                        </plugin>
                </plugins>
        </build>

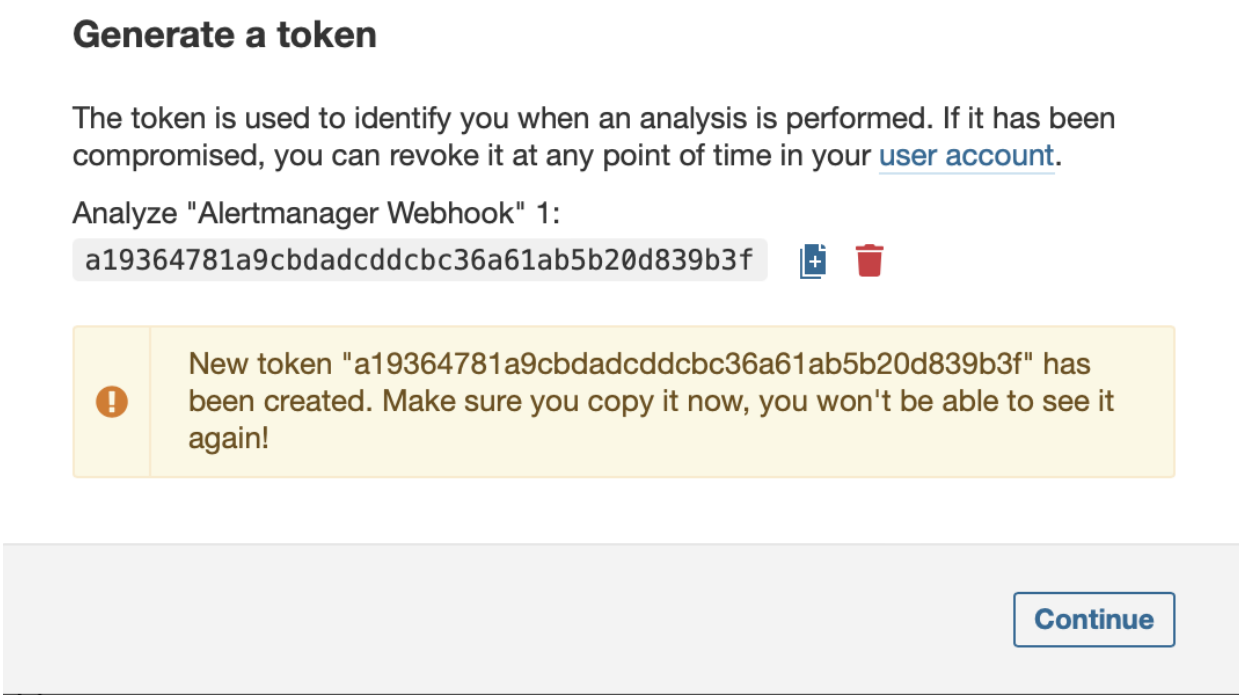
</project>
```




配置 Gitlab 环境变量，点击 “Generate a token” 按钮，生成 SONAR_TOKEN




点击 “Generate” 按钮

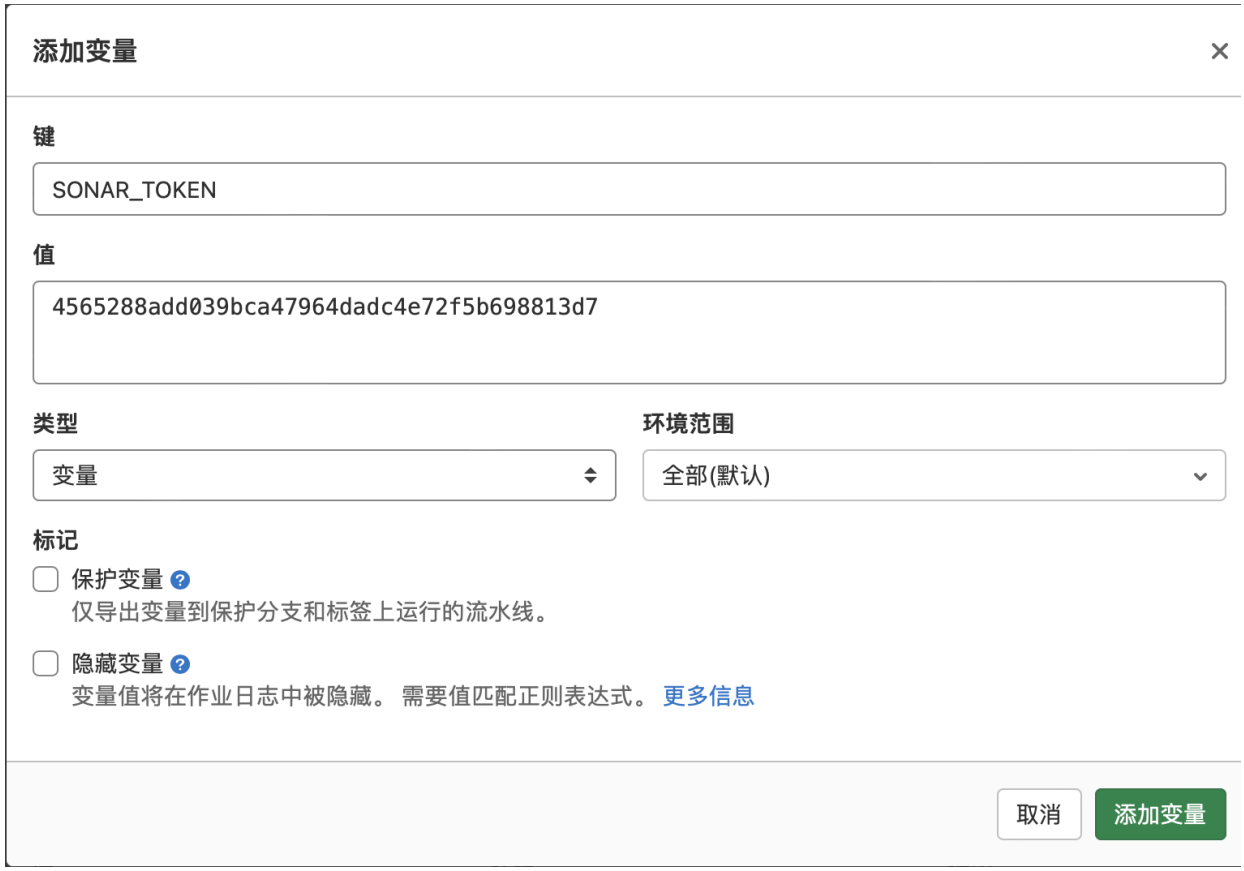



点击加号“➕”图标复制SONAR_TOKEN

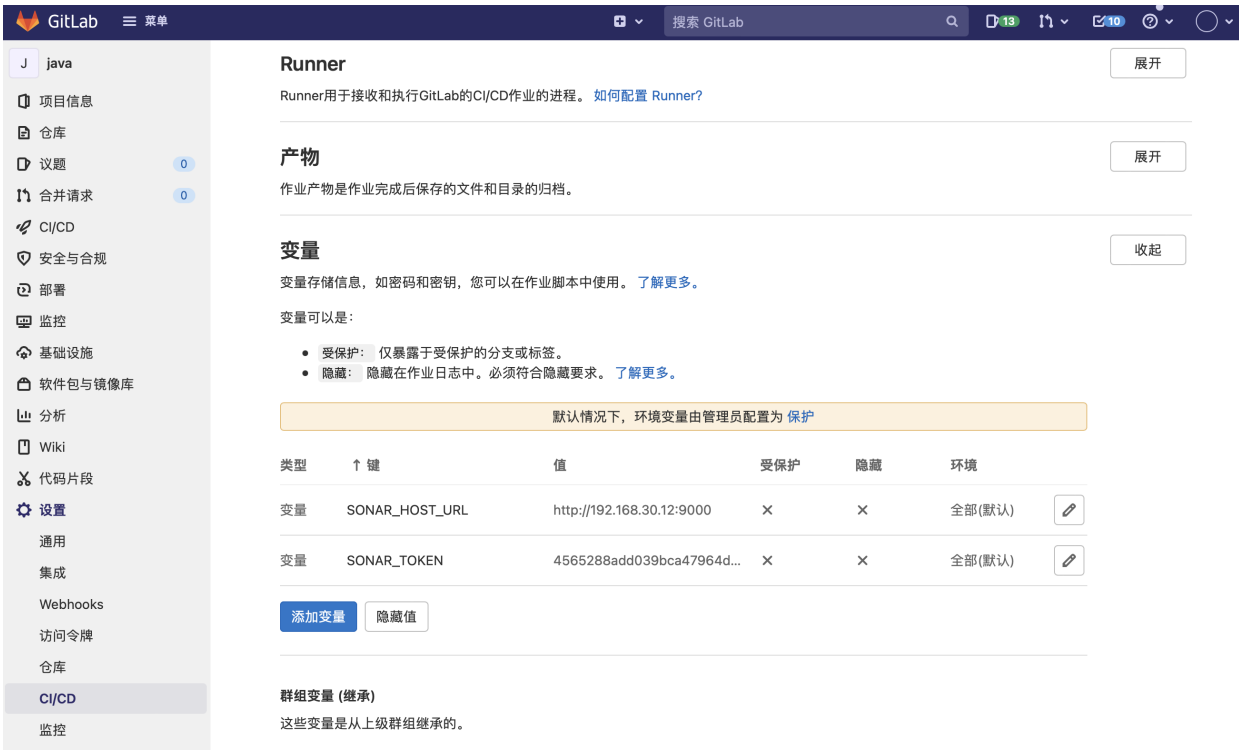
现在切换到 Gitlab 窗口，进入项目 - 设置 - CI/CD，展开 “变量”



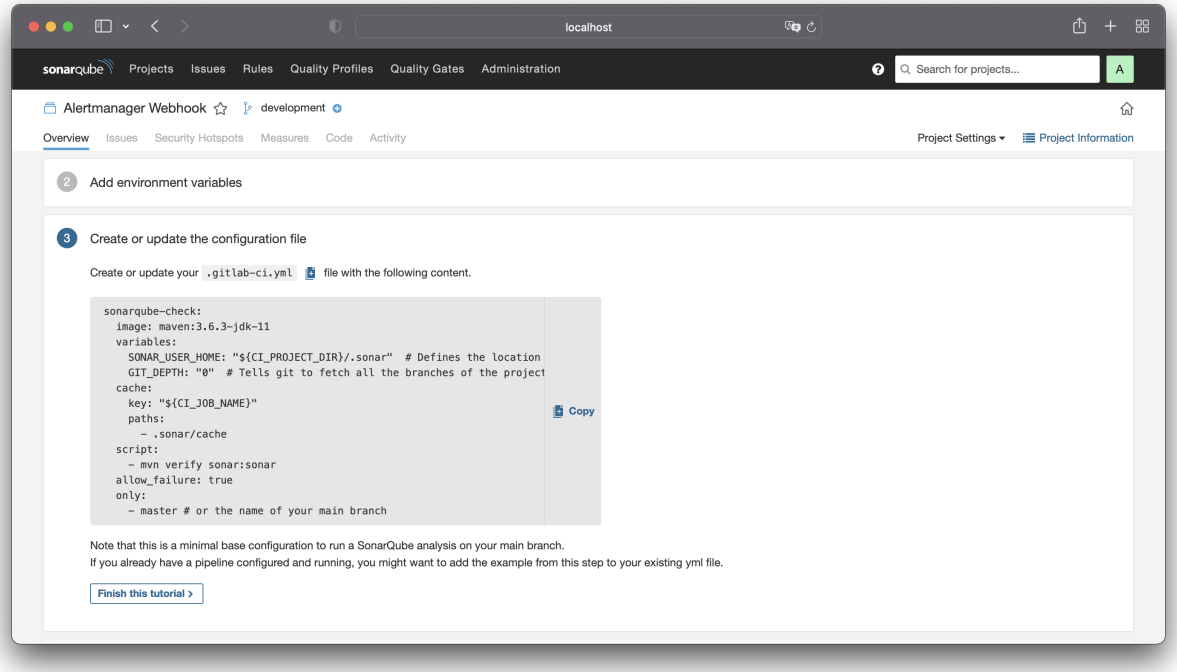

点击 “添加变量” 按钮，从 SonarQube 窗口复制并添加变量 SONAR_TOKEN 和 SONAR_HOST_URL




添加完成后，点击“显示值”按钮，检查变量是否正确



点击即 “Continue” 按钮



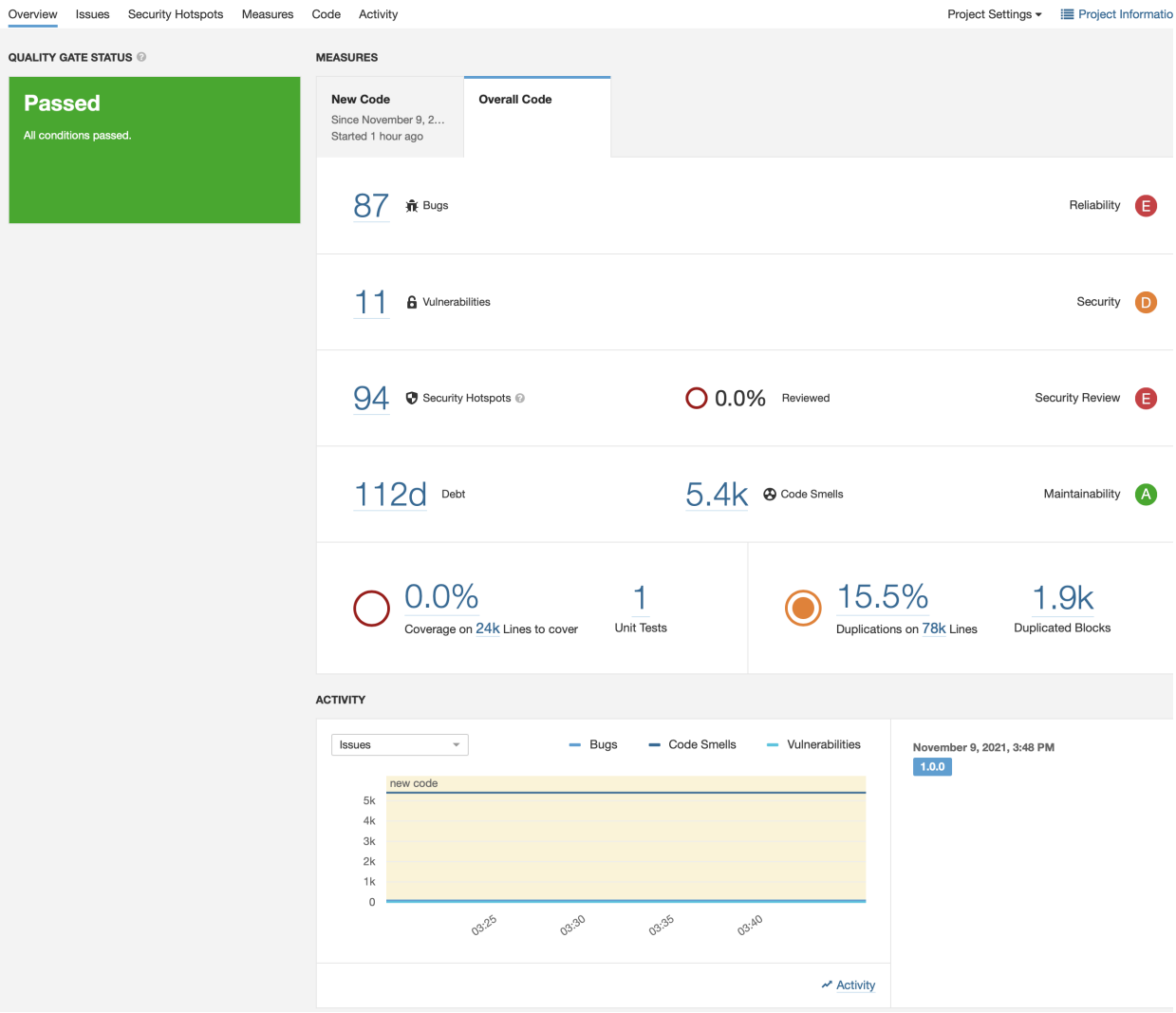

复制内容，并添加到 .gitlab-ci.yml 文件中

**提示**

注意：你的项目必须使用 `Java 11` 以上的版本，否则会出错，具体请看 `FAQ` 章节。

所有工作完成之后，点击 “Finish this tutorial” 按钮，SonarQube 窗口放在那里不用管它。

现在提交和推送代码，然后盯着流水线，如果不出错，SonarQube 就会生成下面这样的报告




## 2.4. SonarScanner

```
sonar-scanner \
  -Dsonar.projectKey=aabbcc \
  -Dsonar.sources=. \
  -Dsonar.host.url=http://localhost:9000 \
  -Dsonar.login=161e6f54add09c966518fa45d2860bad3ebf9774
```

### Node.js

https://www.npmjs.com/package/sonarqube-scanner

创建 sonar.js 文件

```
const sonarqubeScanner = require('sonarqube-scanner');
```

```
sonarqubeScanner({
        serverUrl: 'http://192.168.30.20:9000',
        token: '880300b52817bae1fe26de51fb36b6da47c40edd',
        options : {
                'sonar.projectName': 'admin.netkiller.cn',
                'sonar.sources': '.',
                'sonar.inclusions' : 'src/**'
        },
}, () => {});
```

package.json

```
{
  "name": "netkiller",
  "version": "1.0.0",
  "description": "http://www.netkiller.cn",
  "author": "Neo Chen",
  "license": "MIT",
  "scripts": {
    "sonar": "node sonar.js"
  },
  "dependencies": {
    "sonarqube-scanner": "^2.8.1"
  }
}
```

```
[gitlab-runner@gitlab admin.netkiller.cn]$ npm run sonar

> netkiller@2.3.0 sonar /home/gitlab-runner/admin.netkiller.cn
> node sonar.js

[18:39:26] Starting analysis...
[18:39:26] Getting info from "package.json" file
[18:39:26] Checking if executable exists: /home/gitlab-runner/.sonar/native-sonar-
scanner/sonar-scanner-4.5.0.2216-linux/bin/sonar-scanner
[18:39:26] Platform binaries for SonarScanner found. Using it.
INFO: Scanner configuration file: /home/gitlab-runner/.sonar/native-sonar-scanner/sonar-
scanner-4.5.0.2216-linux/conf/sonar-scanner.properties
INFO: Project root configuration file: NONE
INFO: SonarScanner 4.5.0.2216
INFO: Java 11.0.3 AdoptOpenJDK (64-bit)
INFO: Linux 4.18.0-338.el8.x86_64 amd64
INFO: User cache: /home/gitlab-runner/.sonar/cache
INFO: Scanner configuration file: /home/gitlab-runner/.sonar/native-sonar-scanner/sonar-
scanner-4.5.0.2216-linux/conf/sonar-scanner.properties
INFO: Project root configuration file: NONE
INFO: Analyzing on SonarQube server 9.1.0
INFO: Default locale: "en_US", source code encoding: "US-ASCII" (analysis is platform
dependent)
INFO: Load global settings
INFO: Load global settings (done) | time=126ms
INFO: Server id: 243B8A4D-AXz-jVsGB3jmSUHEudyb
INFO: User cache: /home/gitlab-runner/.sonar/cache
INFO: Load/download plugins
INFO: Load plugins index
```

```
INFO: Load plugins index (done) | time=64ms
INFO: Load/download plugins (done) | time=120ms
INFO: Process project properties
INFO: Process project properties (done) | time=8ms
INFO: Execute project builders
INFO: Execute project builders (done) | time=1ms
INFO: Project key: netkiller
INFO: Base dir: /home/gitlab-runner/admin.netkiller.cn
INFO: Working dir: /home/gitlab-runner/admin.netkiller.cn/.scannerwork
INFO: Load project settings for component key: 'netkiller'
INFO: Load project settings for component key: 'netkiller' (done) | time=72ms
INFO: Load quality profiles
INFO: Load quality profiles (done) | time=216ms
INFO: Load active rules
INFO: Load active rules (done) | time=4596ms
INFO: Indexing files...
INFO: Project configuration:
INFO:   Included sources: src/**
INFO:   Excluded sources: node_modules/**, bower_components/**, jspm_packages/**, typings/**,
lib-cov/**
INFO: Load project repositories
INFO: Load project repositories (done) | time=71ms
INFO: 460 files indexed
INFO: 889 files ignored because of inclusion/exclusion patterns
INFO: 0 files ignored because of scm ignore settings
INFO: Quality profile for css: Sonar way
INFO: Quality profile for js: Sonar way
INFO: ------------- Run sensors on module admin.netkiller.cn
INFO: Load metrics repository
INFO: Load metrics repository (done) | time=48ms
INFO: Sensor CSS Metrics [cssfamily]
INFO: Sensor CSS Metrics [cssfamily] (done) | time=109ms
INFO: Sensor CSS Rules [cssfamily]
INFO: 203 source files to be analyzed
INFO: 203/203 source files have been analyzed
INFO: Sensor CSS Rules [cssfamily] (done) | time=2819ms
INFO: Sensor JaCoCo XML Report Importer [jacoco]
INFO: 'sonar.coverage.jacoco.xmlReportPaths' is not defined. Using default locations:
target/site/jacoco/jacoco.xml,target/site/jacoco-
it/jacoco.xml,build/reports/jacoco/test/jacocoTestReport.xml
INFO: No report imported, no coverage information will be imported by JaCoCo XML Report
Importer
INFO: Sensor JaCoCo XML Report Importer [jacoco] (done) | time=4ms
INFO: Sensor JavaScript analysis [javascript]
WARN: You are using Node.js version 10, which reached end-of-life. Support for this version
will be dropped in future release, please upgrade Node.js to more recent version.
INFO: 304 source files to be analyzed
INFO: 30/304 files analyzed, current file: src/views/fcms/LoanIn/ScreenCustomers/index.vue
INFO: 87/304 files analyzed, current file: src/views/fcms/confingManage/warnConfig/index.vue
INFO: 153/304 files analyzed, current file: src/views/tdms/components/BusinessRisk.vue
INFO: 211/304 files analyzed, current file: src/views/fcms/LoanIn/LoanModel/modal.vue
INFO: 275/304 files analyzed, current file: src/views/system/post/index.vue
INFO: 304/304 source files have been analyzed
INFO: Sensor JavaScript analysis [javascript] (done) | time=57807ms
INFO: Sensor TypeScript analysis [javascript]
INFO: No input files found for analysis
INFO: Sensor TypeScript analysis [javascript] (done) | time=7ms
INFO: Sensor C# Project Type Information [csharp]
INFO: Sensor C# Project Type Information [csharp] (done) | time=1ms
INFO: Sensor C# Analysis Log [csharp]
INFO: Sensor C# Analysis Log [csharp] (done) | time=9ms
INFO: Sensor C# Properties [csharp]
INFO: Sensor C# Properties [csharp] (done) | time=0ms
INFO: Sensor JavaXmlSensor [java]
INFO: Sensor JavaXmlSensor [java] (done) | time=2ms
INFO: Sensor HTML [web]
INFO: Sensor HTML [web] (done) | time=479ms
```

```
INFO: Sensor VB.NET Project Type Information [vbnet]
INFO: Sensor VB.NET Project Type Information [vbnet] (done) | time=3ms
INFO: Sensor VB.NET Analysis Log [vbnet]
INFO: Sensor VB.NET Analysis Log [vbnet] (done) | time=13ms
INFO: Sensor VB.NET Properties [vbnet]
INFO: Sensor VB.NET Properties [vbnet] (done) | time=0ms
INFO: ------------- Run sensors on project
INFO: Sensor Zero Coverage Sensor
INFO: Sensor Zero Coverage Sensor (done) | time=68ms
INFO: CPD Executor 16 files had no CPD blocks
INFO: CPD Executor Calculating CPD for 288 files
INFO: CPD Executor CPD calculation finished (done) | time=269ms
INFO: Analysis report generated in 127ms, dir size=4.0 MB
INFO: Analysis report compressed in 400ms, zip size=1.7 MB
INFO: Analysis report uploaded in 792ms
INFO: ANALYSIS SUCCESSFUL, you can browse http://192.168.30.20:9000/dashboard?id=netkiller
INFO: Note that you will be able to access the updated dashboard once the server has processed
the submitted analysis report
INFO: More about the report processing at http://192.168.30.20:9000/api/ce/task?
id=AX0ESRKaT19KeT2iVgTn
INFO: Analysis total time: 1:20.455 s
INFO: ------------------------------------------------------------------------
INFO: EXECUTION SUCCESS
INFO: ------------------------------------------------------------------------
INFO: Total time: 1:21.380s
INFO: Final Memory: 13M/50M
INFO: ------------------------------------------------------------------------
```

# 3. FAQ

## 3.1. bootstrap check failure [1] of [1]: max virtual memory areas vm.max_map_count [65530] is too low, increase to at least [262144]

```
sonarqube | ERROR: [1] bootstrap checks failed. You must
address the points described in the following [1] lines before
starting Elasticsearch.
sonarqube | bootstrap check failure [1] of [1]: max virtual
memory areas vm.max_map_count [65530] is too low, increase to
at least [262144]
sonarqube | ERROR: Elasticsearch did not exit normally - check
the logs at /opt/sonarqube/logs/sonarqube.log
```

/etc/sysctl.conf 增加配置项

```
vm.max_map_count=655360
```

## 3.2. failed: An API incompatibility was encountered while executing org.sonarsource.scanner.maven:sonar-maven-plugin:3.9.0.2155:sonar: java.lang.UnsupportedClassVersionError: org/sonar/batch/bootstrapper/EnvironmentInformation has been compiled by a more recent version of the Java Runtime (class file version 55.0), this version of the Java Runtime only recognizes class file versions up to 52.0

```
[ERROR] Failed to execute goal
org.sonarsource.scanner.maven:sonar-maven-
plugin:3.9.0.2155:sonar (default-cli) on project demo:
Execution default-cli of goal
org.sonarsource.scanner.maven:sonar-maven-
plugin:3.9.0.2155:sonar failed: An API incompatibility was
encountered while executing
org.sonarsource.scanner.maven:sonar-maven-
plugin:3.9.0.2155:sonar:
java.lang.UnsupportedClassVersionError:
org/sonar/batch/bootstrapper/EnvironmentInformation has been
compiled by a more recent version of the Java Runtime (class
file version 55.0), this version of the Java Runtime only
recognizes class file versions up to 52.0
```

问题分析，SonarQube 系统是建立在 Java 11 的基础之上，而我们自己的代码是 Java 1.8，所以在 mvn package 的时候可以编译成功，但是在执行 mvn verify sonar:sonar 的时候 sonar-maven-plugin 需要 Java 11，所以会报错。

```
JDK Version     Bytecode Version
Java 1.0        45.0
Java 1.1        45.3
Java 1.2        46.0
Java 1.3        47.0
Java 1.4        48.0
Java 5          49.0
Java 6          50.0
Java 7          51.0
Java 8          52.0
Java 9          53.0
Java 10         54.0
Java 11         55.0
Java 12         56.0
Java 13         57.0
Java 14         58.0
Java 15         59.0
```

```
Java 16         60.0
Java 17         61.0
Java 18         62.0
```

更换 JDK 版本可以解决

如果你的代码无法工作在 Java 11 之上，就需要解决编译使用 Java 8，执行 sonar 时使用 Java 11，你需要安装两个JDK

```
[root@localhost ~]# dnf install java-11-openjdk java-11-
openjdk-devel
[root@localhost ~]# dnf install java-1.8.0-openjdk java-1.8.0-
openjdk-devel
```

注意安装顺序，先安装 Java 11 再安装 Java 8，这样系统默认Java 是 1.8

```
[root@localhost ~]# java -version
openjdk version "1.8.0_312"
OpenJDK Runtime Environment (build 1.8.0_312-b07)
OpenJDK 64-Bit Server VM (build 25.312-b07, mixed mode)
```

编译方法

```
[root@localhost ~]# java -version
openjdk version "1.8.0_312"
OpenJDK Runtime Environment (build 1.8.0_312-b07)
OpenJDK 64-Bit Server VM (build 25.312-b07, mixed mode)
[root@localhost ~]# mvn clean package
[root@localhost ~]# mvn verify
```

```
[root@localhost ~]# export JAVA_HOME=/usr/lib/jvm/java-11-
openjdk
[root@localhost ~]# PATH=$JAVA_HOME/bin:$PATH
[root@localhost ~]# java -version
openjdk version "11.0.13" 2021-10-19 LTS
OpenJDK Runtime Environment 18.9 (build 11.0.13+8-LTS)
OpenJDK 64-Bit Server VM 18.9 (build 11.0.13+8-LTS, mixed mode,
sharing)
[root@localhost ~]# mvn sonar:sonar -
Dsonar.projectKey=api.netkiller.cn_AX0DhnglXpSwMKevAarP \
  -Dsonar.host.url=http://192.168.30.12:9000 \
  -Dsonar.login=161e6f54add09c966518fa45d2860bad3ebf9774
```

修改 Gitlab 持续集成 .gitlab-ci.yml 文件

```
sonarqube-check:
  # image: maven:3.6.3-jdk-11
  variables:
    SONAR_USER_HOME: "${CI_PROJECT_DIR}/.sonar"  # Defines the
location of the analysis task cache
    GIT_DEPTH: "0"  # Tells git to fetch all the branches of
the project, required by the analysis task
  cache:
    key: "${CI_JOB_NAME}"
    paths:
      - .sonar/cache
  tags:
    - shell
  before_script:
    - rm -rf doc sql
  after_script:
    - wechat -t 1 api.netkiller.cn $CI_COMMIT_BRANCH 分支代码质量
和安全漏洞扫描完毕 http://192.168.30.12:9000/dashboard?
id=api.netkiller.cn_AX0DhnglXpSwMKevAarP
  script:
    - mvn verify
    - export JAVA_HOME=/usr/lib/jvm/java-11-openjdk
    - export PATH=$JAVA_HOME/bin:$PATH
    - mvn sonar:sonar -
```

```
Dsonar.projectKey=api.netkiller.cn_AX0DhnglXpSwMKevAarP
  allow_failure: true
  only:
    - testing
```

注意：这里没有使用 docker 构建，我个人比较喜欢 Shell 执行器，它的速度比 docker 快

## 3.3. [ERROR] An unknown compilation problem occurred

由于 SonarQube 使用的是 OpenJDK 11，编译代码是 1.8 会出现下面错误

```
[ERROR] Failed to execute goal org.apache.maven.plugins:maven-
compiler-plugin:3.1:compile (default-compile) on project
netkiller-common: Compilation failure
582[ERROR] An unknown compilation problem occurred
```

配置 maven-compiler-plugin 插件，指定JDK版本

```
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.8.1</version>
                <configuration>
                    <source>1.8</source>
                    <target>1.8</target>
                    <encoding>UTF-8</encoding>
                </configuration>
            </plugin>
        </plugins>
```

## 3.4. can't have 2 modules with the following key

错误日志

```
[ERROR] Failed to execute goal
org.sonarsource.scanner.maven:sonar-maven-
plugin:3.9.0.2155:sonar (default-cli) on project
api.netkiller.cn: Project
'api.netkiller.cn_AX0DhnglXpSwMKevAarP' can't have 2 modules
with the following key: api.netkiller.cn_AX0DhnglXpSwMKevAarP -
> [Help 1]
```

出错原因，Maven 使用了 module 结构

```
Project
   |- pom.xml
   |- module-1
   |      |- pom.xml
   |- module-2
   |      |- pom.xml
```

父 pom.xml 中添加了

```
        <properties>
                <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
                <project.reporting.outputEncoding>UTF-
8</project.reporting.outputEncoding>
                <java.version>1.8</java.version>

<sonar.projectKey>api.netkiller.cn_AX0DhnglXpSwMKevAarP</sonar.
projectKey>
```

```
<sonar.qualitygate.wait>true</sonar.qualitygate.wait>

        </properties>
```

module-1 和 module-2 会继承 parent 中的 properties 定义。

解决方案，注释 sonar.projectKey

```
        <properties>
                <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
                <project.reporting.outputEncoding>UTF-
8</project.reporting.outputEncoding>
                <java.version>1.8</java.version>

                <!--
<sonar.projectKey>api.netkiller.cn_AX0DhnglXpSwMKevAarP</sonar.
projectKey> -->

<sonar.qualitygate.wait>true</sonar.qualitygate.wait>

        </properties>
```

修改 Gitlab 持续集成 .gitlab-ci.yml 文件

mvn verify sonar:sonar -
Dsonar.projectKey=api.netkiller.cn_AX0DhnglXpSwMKevAarP

```
stages:
  - build
  - test
  - deploy

build-job:
```

```
  stage: build
  tags:
    - shell
  script:
    - echo "Compiling the code..."
    - mvn package
    - echo "Compile complete."

# unit-test-job:
#   stage: test
#   script:
#     - echo "Running unit tests... This will take about 60
seconds."
#     - echo "Code coverage is 90%"

# lint-test-job:
#   stage: test
#   script:
#     - echo "Linting code... This will take about 10 seconds."
#     - echo "No lint issues found."

sonarqube-check:
  image: maven:3.6.3-jdk-11
  variables:
    SONAR_USER_HOME: "${CI_PROJECT_DIR}/.sonar"  # Defines the
location of the analysis task cache
    GIT_DEPTH: "0"  # Tells git to fetch all the branches of
the project, required by the analysis task
  cache:
    key: "${CI_JOB_NAME}"
    paths:
      - .sonar/cache
  tags:
    - docker
  script:
    - mvn verify sonar:sonar -
Dsonar.projectKey=api.netkiller.cn_AX0DhnglXpSwMKevAarP
  allow_failure: true
#   only:
    # - master # or the name of your main branch


deploy-job:
  stage: deploy
  script:
```

```
    - echo "Deploying application..."
    - echo "Application successfully deployed."
```

还有一种解决方案，我没有测试过

```
<sonar.projectKey>your_projectKey</sonar.projectKey>
<sonar.moduleKey>${artifactId}</sonar.moduleKey>
```

## 3.5. Kubernetes 运行 sonar-scanner

sonar-project.properties

```
# must be unique in a given SonarQube instance
sonar.projectKey=ejy-finance-admin
sonar.host.url=http://sonarqube.default.svc.cluster.local
sonar.login=sqp_84b8d5f75bdc9dd20bf9339fb6dd5d4cda5c152d
# --- optional properties ---

# defaults to project key
sonar.projectName=ejy-finance-admin
# defaults to 'not provided'
sonar.projectVersion=1.0

# Path is relative to the sonar-project.properties file.
Defaults to .
sonar.sources=.

# Encoding of the source code. Default is default system
encoding
sonar.sourceEncoding=UTF-8
```

.gitlab-ci.yml

```
stages:
  - build
  - check
  - docker
  - deploy

variables:
  DOCKER_REGISTRY: registry.netkiller.cn
  IMAGE:
$DOCKER_REGISTRY/$CI_COMMIT_BRANCH/$CI_PROJECT_NAME:$CI_COMMIT_
SHORT_SHA-$CI_PIPELINE_ID

cache:
  key: ${CI_COMMIT_REF_SLUG}
  paths:
    - node_modules/
    - dist/

build-job:
  stage: build
  image: node:14.17-alpine
  before_script:
    - npm install --registry=https://registry.npm.taobao.org
  script:
    - npm run build:${CI_COMMIT_REF_SLUG}
  after_script:
    - tar zcvf $CI_PROJECT_NAME.tgz dist
    - ls -l $CI_PROJECT_NAME.tgz
  only:
    - office
    - dev
    - test
  tags:
    - kubernetes
  artifacts:
    name: "$CI_PROJECT_NAME"
    paths:
      - $CI_PROJECT_NAME.tgz

sonarqube-check:
  stage: check
  image: sonarsource/sonar-scanner-cli:latest
```

```
  variables:
    SONAR_USER_HOME: "${CI_PROJECT_DIR}/.sonar"
    GIT_DEPTH: "0"
  cache:
    key: "${CI_JOB_NAME}"
    paths:
      - .sonar/cache
  script:
    - sonar-scanner
  allow_failure: true
  only:
    - master
    - office
  tags:
    - kubernetes

build-docker:
  stage: docker
  image: docker:latest
  before_script:
    - echo "$CI_REGISTRY_PASSWORD" | docker login
$DOCKER_REGISTRY --username $CI_REGISTRY_USER --password-stdin
    - ls -l dist
  after_script:
    - docker images | grep $CI_PROJECT_NAME
    - docker image rm $IMAGE
  script:
    - docker build -t $IMAGE -f Dockerfile .
    - docker push $IMAGE
  only:
    - office
    - dev
    - test
  tags:
    - kubernetes

deploy-job:
  stage: deploy
  variables:
    GIT_STRATEGY: none
  dependencies: []
  before_script:
    - kubectl -n ${CI_COMMIT_BRANCH} get pod | grep
$CI_PROJECT_NAME
  script:
```

```
    - kubectl set image deployment/${CI_PROJECT_NAME}
${CI_PROJECT_NAME}=${IMAGE} -n ${CI_COMMIT_BRANCH}
  after_script:
    - kubectl -n ${CI_COMMIT_BRANCH} get pod | grep
$CI_PROJECT_NAME
  only:
    - dev
    - test
  tags:
    - shell
  environment:
    name: $CI_COMMIT_BRANCH
    url: $CI_COMMIT_BRANCH.netkiller.cn/$CI_PROJECT_NAME

deploy-office:
  stage: deploy
  image: docker:latest
  # cache: []
  variables:
    GIT_STRATEGY: none
    PORT: 1850
  before_script:
    - docker login -u "$CI_REGISTRY_USER" -p
"$CI_REGISTRY_PASSWORD" $DOCKER_REGISTRY
  script:
    - docker rm -f $CI_PROJECT_NAME
    - docker run --restart always -d --name $CI_PROJECT_NAME -v
/mnt/logs/$CI_PROJECT_NAME:/var/log/nginx -p $PORT:80 $IMAGE
  after_script:
    - docker ps -a | grep -w $CI_PROJECT_NAME
  only:
    - office
  tags:
    - office
  environment:
    name: office
    url: www.netkiller.cn
```

# 第 14 章 Dagger

***Dagger: a new way to create CI/CD pipelines!***

Dagger 是 Docker 公司推出的一个 DevOps 产品

# 第 15 章 持续集成工具

# 1. Code Review

## 1.1. Phabricator - an open source, software engineering platform

**Phabricator is a collection of open source web applications that help software companies build better software.**

Phabricator 是 Facebook 开源的Code Review 工具

## 1.2. Gerrit

Gerrit 是 Google 开发的Code Review 工具

## 1.3. TeamCity

# 2. Nexus Repository OSS

https://www.sonatype.com/download-oss-sonatype

```
wget https://www.sonatype.com/oss-thank-you-tar.gz
```

## 2.1. 安装 Nexus

### Docker

```
docker run -d -p 8081:8081 --restart=always --name nexus
sonatype/nexus3
```

## 2.2. Nexus UI

http://localhost:8081/ 登陆用户名 admin 默认密码 admin123

## 2.3. maven 设置

maven在settings.xml中配置如下，下次maven就会通过访问电脑上的私服来获取jar包

```
<mirrors>
    <mirror>
      <id>nexus</id>
      <mirrorOf>*</mirrorOf>
      <url>http://localhost:8081/repository/maven-public/</url>
    </mirror>
```

```
</mirrors>
```

## 2.4. Node.js

输入命令登陆远程仓库

```
npm login --registry=http://nexus.netkiller.cn/repository/npm/
```

在项目中输入

```
npm pack
```

上传

```
npm publish --
registry=http://nexus.netkiller.cn/repository/npm/
```

## 2.5. Ruby

安装 nexus 包

```
$ gem install nexus
```

打包

```
gem build project.gemspec
```

上传，系统会提示上传URL

```
gem nexus project-1.0.0.gem
```

# 第 16 章 Open Source Requirements Management Tool

http://sourceforge.net/projects/osrmt/

```
<client directory>\v1_50\client\

copy connection.mysql.xml connection.xml
```

```
<client directory>\v1_50\client\upgrade.bat

Select configuration option 1,2,3 or 4
1) Define a new connection
2) Test the connection
3) Save the new connection
4) Initialize a new database
5) Upgrade 1.3 to 1.4 database
6) Migrate database contents
7) Export language file
8) Import language file
0) Exit
Enter option number [Exit]:


Enter option number [Exit]: 4
initializing database defined in connection.xml:
jdbc:mysql://localhost/osrmt?
useUnicode=true&characterEncoding=UTF-8
osrmt
mysql
Target correct? Y|N [Y]: y
Empty schema located - initialize and populate schema? [Y]:
```

# 第 17 章 TRAC

http://trac.edgewall.org

# 1. Ubuntu 安装

## 1.1. source code

过程 17.1. TRAC - source

1. setup.py

```
wget http://peak.telecommunity.com/dist/ez_setup.py
python ez_setup.py
```

2. Trac

```
wget http://download.edgewall.org/trac/Trac-1.1.1.tar.gz
tar zxvf Trac-1.1.1.tar.gz
cd Trac-1.1.1
sudo python ./setup.py install
cd ..
```

## 1.2. easy_install

过程 17.2. TRAC - easy_install

1. easy_install

```
$ sudo apt-get install python-setuptools
```

2. Installing Trac

```
sudo easy_install Pygments
sudo easy_install Genshi
sudo easy_install Trac
```

ClearSilver

```
sudo apt-get install python-clearsilver
```

python svn

```
                                                    sudo apt-
get install python-svn python-svn-dbg
```

create svn repos

```
$ svnadmin create /home/netkiller/repos
```

## 1.3. Apache httpd

```
# cat /etc/httpd/conf.d/trac.conf
<VirtualHost *:80>
  # Change this to the domain which points to your host, i.e.
the domain
  # you set as "phabricator.base-uri".
  ServerName trac.repo

  <Location />
    SetHandler mod_python
    PythonInterpreter main_interpreter
```

```
    PythonHandler trac.web.modpython_frontend
    PythonOption TracEnv /gitroot/trac/default
    PythonOption TracUriRoot /
  </Location>
# Replace all occurrences of /srv/trac with your trac root
below
# and uncomment the respective SetEnv and PythonOption
directives.
#  <LocationMatch /cgi-bin/trac\.f?cgi>
#       SetEnv TRAC_ENV /srv/trac
#  </LocationMatch>
#  <IfModule mod_python.c>
#    <Location /cgi-bin/trac.cgi>
#      SetHandler mod_python
#      PythonHandler trac.web.modpython_frontend
#      #PythonOption TracEnv /srv/trac
#    </Location>
#  </IfModule>
</VirtualHost>
```

# 2. CentOS 安装

http://trac.edgewall.org/

```
[root@development ~]# yum install python-setuptools
[root@development ~]# easy_install Trac

[root@development ~]# trac-admin /var/www/myproject initenv
```

## 2.1. trac.ini

subversion 仓库配置

```
                                vim /srv/conf/trac.ini

                                repository_dir =
/svnroot/example.com
```

## 2.2. standalone

```
tracd -s --port 8000 /var/www/myproject
```

multiple projects

```
tracd --port 8000 /var/www/trac/project1/
/var/www/trac/project2 ...
or
tracd --port 8000 -e /var/www/trac/
```

## 2.3. Using Authentication

Using Authentication

To create a .passwd file using htdigest:

```
htdigest -c /var/www/trac/.passwd localhost neo
```

then for additional users:

```
htdigest /var/www/trac/.passwd localhost netkiller
```

bind ip

```
tracd -d --host 192.168.3.9 --port 8000 --
auth=*,/srv/trac/.passwd,localhost -e /srv/trac
```

```
$ tracd -p 8080 \
   --auth=project1,/path/to/users.htdigest,mycompany.com \
   --auth=project2,/path/to/users.htdigest,mycompany.com \
   /path/to/project1 /path/to/project2

tracd -p 8000 \
   --auth=*,/var/www/trac/.passwd,localhost \
   -e /var/www/trac/
```

## 2.4. trac-admin

```
# trac-admin /srv/example help
trac-admin - The Trac Administration Console 0.12.3
```

```
Usage: trac-admin </path/to/projenv> [command [subcommand]
[option ...]]

Invoking trac-admin without command starts interactive mode.

help                 Show documentation
initenv              Create and initialize a new environment
attachment add       Attach a file to a resource
attachment export    Export an attachment from a resource to a
file or stdout
attachment list      List attachments of a resource
attachment remove    Remove an attachment from a resource
changeset added      Notify trac about changesets added to a
repository
changeset modified   Notify trac about changesets modified in a
repository
component add        Add a new component
component chown      Change component ownership
component list       Show available components
component remove     Remove/uninstall a component
component rename     Rename a component
config get           Get the value of the given option in
"trac.ini"
config remove        Remove the specified option from
"trac.ini"
config set           Set the value for the given option in
"trac.ini"
deploy               Extract static resources from Trac and all
plugins
hotcopy              Make a hot backup copy of an environment
milestone add        Add milestone
milestone completed  Set milestone complete date
milestone due        Set milestone due date
milestone list       Show milestones
milestone remove     Remove milestone
milestone rename     Rename milestone
permission add       Add a new permission rule
permission list      List permission rules
permission remove    Remove a permission rule
priority add         Add a priority value option
priority change      Change a priority value
priority list        Show possible ticket priorities
priority order       Move a priority value up or down in the
list
priority remove      Remove a priority value
```

```
repository add       Add a source repository
repository alias     Create an alias for a repository
repository list      List source repositories
repository remove    Remove a source repository
repository resync    Re-synchronize trac with repositories
repository set       Set an attribute of a repository
repository sync      Resume synchronization of repositories
resolution add       Add a resolution value option
resolution change    Change a resolution value
resolution list      Show possible ticket resolutions
resolution order     Move a resolution value up or down in the
list
resolution remove    Remove a resolution value
session add          Create a session for the given sid
session delete       Delete the session of the specified sid
session list         List the name and email for the given sids
session purge        Purge all anonymous sessions older than
the given age
session set          Set the name or email attribute of the
given sid
severity add         Add a severity value option
severity change      Change a severity value
severity list        Show possible ticket severities
severity order       Move a severity value up or down in the
list
severity remove      Remove a severity value
ticket remove        Remove ticket
ticket_type add      Add a ticket type
ticket_type change   Change a ticket type
ticket_type list     Show possible ticket types
ticket_type order    Move a ticket type up or down in the list
ticket_type remove   Remove a ticket type
upgrade              Upgrade database to current version
version add          Add version
version list         Show versions
version remove       Remove version
version rename       Rename version
version time         Set version date
wiki dump            Export wiki pages to files named by title
wiki export          Export wiki page to file or stdout
wiki import          Import wiki page from file or stdin
wiki list            List wiki pages
wiki load            Import wiki pages from files
wiki remove          Remove wiki page
wiki rename          Rename wiki page
```

```
wiki replace          Replace the content of wiki pages from
files (DANGEROUS!)
wiki upgrade          Upgrade default wiki pages to current
version
```

## Permissions

```
BROWSER_VIEW

CHANGESET_VIEW

CONFIG_VIEW

EMAIL_VIEW

FILE_VIEW


LOG_VIEW

MILESTONE_ADMIN

MILESTONE_CREATE

MILESTONE_DELETE


MILESTONE_MODIFY

MILESTONE_VIEW

PERMISSION_ADMIN

PERMISSION_GRANT


PERMISSION_REVOKE

REPORT_ADMIN
```

```
REPORT_CREATE

REPORT_DELETE


REPORT_MODIFY

REPORT_SQL_VIEW

REPORT_VIEW

ROADMAP_ADMIN

ROADMAP_VIEW


SEARCH_VIEW

TICKET_ADMIN

TICKET_APPEND

TICKET_CHGPROP

TICKET_CREATE


TICKET_EDIT_CC

TICKET_EDIT_COMMENT

TICKET_EDIT_DESCRIPTION


TICKET_MODIFY

TICKET_VIEW

TIMELINE_VIEW

TRAC_ADMIN


VERSIONCONTROL_ADMIN
```

```
WIKI_ADMIN

WIKI_CREATE

WIKI_DELETE

WIKI_MODIFY


WIKI_RENAME

WIKI_VIEW
```

admin

```
$ trac-admin /path/to/projenv permission add neo TICKET_ADMIN
TRAC_ADMIN  WIKI_ADMIN
```

group

```
$ trac-admin /path/to/projenv permission add admin
MILESTONE_ADMIN PERMISSION_ADMIN REPORT_ADMIN ROADMAP_ADMIN
TICKET_ADMIN TRAC_ADMIN VERSIONCONTROL_ADMIN WIKI_ADMIN

$ trac-admin /path/to/projenv permission add developer
WIKI_ADMIN
$ trac-admin /path/to/projenv permission add developer
REPORT_ADMIN
$ trac-admin /path/to/projenv permission add developer
TICKET_ADMIN
```

user

```
$ trac-admin /path/to/projenv permission add bob developer
$ trac-admin /path/to/projenv permission add john developer
```

**Resync**

```
# trac-admin /srv/example repository resync '(default)'
```

旧版本trac: trac-admin /srv/trac/neo resync

# 3. Project Environment

## 3.1. Sqlite

1. Creating a Project Environment

```
$ trac-admin /home/netkiller/projectenv initenv

Creating a new Trac environment at
/home/netkiller/projectenv

Trac will first ask a few questions about your environment
in order to initalize and prepare the project database.

 Please enter the name of your project.
 This name will be used in page titles and descriptions.

Project Name [My Project]>

 Please specify the connection string for the database to
use.
 By default, a local SQLite database is created in the
environment
 directory. It is also possible to use an already existing
 PostgreSQL database (check the Trac documentation for the
exact
 connection string syntax).

Database connection string [sqlite:db/trac.db]>

 Please specify the type of version control system,
 By default, it will be svn.

 If you don't want to use Trac with version control
integration,
 choose the default here and don't specify a repository
directory.
 in the next question.
```

```
Repository type [svn]>

 Please specify the absolute path to the version control
 repository, or leave it blank to use Trac without a
repository.
 You can also set the repository location later.

Path to repository [/path/to/repos]> /home/netkiller/repos

 Please enter location of Trac page templates.
 Default is the location of the site-wide templates
installed with Trac.

Templates directory [/usr/share/trac/templates]>

Creating and Initializing Project
 Installing default wiki pages
 /usr/share/trac/wiki-default/TracIni => TracIni
 /usr/share/trac/wiki-default/TracSupport => TracSupport
 /usr/share/trac/wiki-default/WikiStart => WikiStart
 /usr/share/trac/wiki-default/TitleIndex => TitleIndex
 /usr/share/trac/wiki-default/TracModPython =>
TracModPython
 /usr/share/trac/wiki-default/TracInterfaceCustomization =>
TracInterfaceCustomization
 /usr/share/trac/wiki-default/WikiDeletePage =>
WikiDeletePage
 /usr/share/trac/wiki-default/TracTicketsCustomFields =>
TracTicketsCustomFields
 /usr/share/trac/wiki-default/TracChangeset =>
TracChangeset
 /usr/share/trac/wiki-default/TracLogging => TracLogging
 /usr/share/trac/wiki-default/TracSyntaxColoring =>
TracSyntaxColoring
 /usr/share/trac/wiki-default/TracImport => TracImport
 /usr/share/trac/wiki-default/TracTimeline => TracTimeline
 /usr/share/trac/wiki-default/TracAdmin => TracAdmin
 /usr/share/trac/wiki-default/InterWiki => InterWiki
 /usr/share/trac/wiki-default/WikiPageNames =>
WikiPageNames
 /usr/share/trac/wiki-default/TracNotification =>
TracNotification
 /usr/share/trac/wiki-default/TracFastCgi => TracFastCgi
 /usr/share/trac/wiki-default/InterTrac => InterTrac
 /usr/share/trac/wiki-default/TracUnicode => TracUnicode
```

```
 /usr/share/trac/wiki-default/TracGuide => TracGuide
 /usr/share/trac/wiki-default/TracRevisionLog =>
TracRevisionLog
 /usr/share/trac/wiki-default/TracBrowser => TracBrowser
 /usr/share/trac/wiki-default/WikiRestructuredText =>
WikiRestructuredText
 /usr/share/trac/wiki-default/TracLinks => TracLinks
 /usr/share/trac/wiki-default/TracInstall => TracInstall
 /usr/share/trac/wiki-default/TracPermissions =>
TracPermissions
 /usr/share/trac/wiki-default/WikiMacros => WikiMacros
 /usr/share/trac/wiki-default/TracQuery => TracQuery
 /usr/share/trac/wiki-default/TracBackup => TracBackup
 /usr/share/trac/wiki-default/TracWiki => TracWiki
 /usr/share/trac/wiki-default/SandBox => SandBox
 /usr/share/trac/wiki-default/TracRoadmap => TracRoadmap
 /usr/share/trac/wiki-default/TracAccessibility =>
TracAccessibility
 /usr/share/trac/wiki-default/TracSearch => TracSearch
 /usr/share/trac/wiki-default/TracPlugins => TracPlugins
 /usr/share/trac/wiki-default/RecentChanges =>
RecentChanges
 /usr/share/trac/wiki-default/WikiNewPage => WikiNewPage
 /usr/share/trac/wiki-default/TracCgi => TracCgi
 /usr/share/trac/wiki-default/TracRss => TracRss
 /usr/share/trac/wiki-default/CamelCase => CamelCase
 /usr/share/trac/wiki-default/WikiFormatting =>
WikiFormatting
 /usr/share/trac/wiki-default/TracTickets => TracTickets
 /usr/share/trac/wiki-default/TracStandalone =>
TracStandalone
 /usr/share/trac/wiki-default/InterMapTxt => InterMapTxt
 /usr/share/trac/wiki-default/TracReports => TracReports
 /usr/share/trac/wiki-default/WikiHtml => WikiHtml
 /usr/share/trac/wiki-default/WikiProcessors =>
WikiProcessors
 /usr/share/trac/wiki-default/TracUpgrade => TracUpgrade
 /usr/share/trac/wiki-default/TracEnvironment =>
TracEnvironment
 /usr/share/trac/wiki-default/WikiRestructuredTextLinks =>
WikiRestructuredTextLinks

Warning:

You should install the SVN bindings
```

```
----------------------------------------------------------
----------
Project environment for 'My Project' created.

You may now configure the environment by editing the file:

  /home/netkiller/projectenv/conf/trac.ini

If you'd like to take this new project environment for a
test drive,
try running the Trac standalone web server `tracd`:

  tracd --port 8000 /home/netkiller/projectenv

Then point your browser to
http://localhost:8000/projectenv.
There you can also browse the documentation for your
installed
version of Trac, including information on further setup
(such as
deploying Trac to a real web server).

The latest documentation can also always be found on the
project
website:

  http://trac.edgewall.org/

Congratulations!
```

2. Running the Standalone Server

```
tracd --port 8000 /home/netkiller/projectenv
```

3. testing

   http://192.168.1.7:8000/projectenv/

4. auth

```
sudo apt-get install apache2-utils

$ htdigest -c /home/neo/trac/conf/passwd.digest localhost
neo
Adding password for neo in realm localhost.
New password:
Re-type new password:

$ htdigest /home/neo/trac/conf/passwd.digest localhost
nchen
Adding user nchen in realm localhost
New password:
Re-type new password:

$ trac-admin /home/neo/trac permission add admin TRAC_ADMIN
$ trac-admin /home/neo/trac permission add netkiller admin

$ trac-admin /home/neo/trac permission add developer
TICKET_ADMIN
$ trac-admin /home/neo/trac permission add nchen developer
$ trac-admin /home/neo/trac permission add neo developer

$ trac-admin /home/neo/trac permission list
User           Action
------------------------------
admin          TRAC_ADMIN
anonymous      BROWSER_VIEW
anonymous      CHANGESET_VIEW
anonymous      FILE_VIEW
anonymous      LOG_VIEW
anonymous      MILESTONE_VIEW
anonymous      REPORT_SQL_VIEW
anonymous      REPORT_VIEW
anonymous      ROADMAP_VIEW
anonymous      SEARCH_VIEW
anonymous      TICKET_VIEW
anonymous      TIMELINE_VIEW
anonymous      WIKI_VIEW
authenticated  TICKET_CREATE
authenticated  TICKET_MODIFY
authenticated  WIKI_CREATE
```

```
authenticated  WIKI_MODIFY
developer      TICKET_ADMIN
nchen          developer
neo            developer
netkiller      admin
```

5. daemon

```
$ tracd -d -s --port 8000 /home/netkiller/projectenv
$ tracd -d -s --port 8000 --auth
trac,/home/neo/trac/conf/passwd.digest,localhost
/home/neo/trac
```

## 3.2. MySQL

```
GRANT ALL PRIVILEGES ON trac.* TO trac@'localhost' IDENTIFIED
BY 'password' WITH GRANT OPTION;
CREATE DATABASE IF NOT EXISTS trac default charset utf8 COLLATE
utf8_general_ci;
```

```
Database connection string [sqlite:db/trac.db]>
mysql://trac:password@localhost:3306/trac
```

下面开始创建项目

```
# trac-admin /home/git/trac initenv
Creating a new Trac environment at /home/git/trac

Trac will first ask a few questions about your environment
in order to initialize and prepare the project database.

 Please enter the name of your project.
 This name will be used in page titles and descriptions.
```

```
Project Name [My Project]>

 Please specify the connection string for the database to use.
 By default, a local SQLite database is created in the
environment
 directory. It is also possible to use an already existing
 PostgreSQL database (check the Trac documentation for the
exact
 connection string syntax).

Database connection string [sqlite:db/trac.db]>
mysql://trac:trac@localhost:3306/trac

Creating and Initializing Project
 Installing default wiki pages
  TracRepositoryAdmin imported from /root/.python-eggs/Trac-
1.1.1-py2.6.egg-tmp/trac/wiki/default-pages/TracRepositoryAdmin
  TracNavigation imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/TracNavigation
  TracUpgrade imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/TracUpgrade
  TracRevisionLog imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/TracRevisionLog
  TracTickets imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/TracTickets
  TracIni imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/TracIni
  PageTemplates imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/PageTemplates
  TracTimeline imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/TracTimeline
  TracAccessibility imported from /root/.python-eggs/Trac-
1.1.1-py2.6.egg-tmp/trac/wiki/default-pages/TracAccessibility
  WikiHtml imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/WikiHtml
  SandBox imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/SandBox
  TracImport imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/TracImport
  TracPlugins imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/TracPlugins
  TracRoadmap imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/TracRoadmap
  TracAdmin imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/TracAdmin
```

```
  TracBatchModify imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/TracBatchModify
  TracBrowser imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/TracBrowser
  InterWiki imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/InterWiki
  WikiRestructuredText imported from /root/.python-eggs/Trac-
1.1.1-py2.6.egg-tmp/trac/wiki/default-
pages/WikiRestructuredText
  WikiProcessors imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/WikiProcessors
  WikiNewPage imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/WikiNewPage
  TracEnvironment imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/TracEnvironment
  TracLogging imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/TracLogging
  TracSupport imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/TracSupport
  TracNotification imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/TracNotification
  TracGuide imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/TracGuide
  WikiStart imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/WikiStart
  TracWorkflow imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/TracWorkflow
  TracRss imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/TracRss
  TracLinks imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/TracLinks
  InterMapTxt imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/InterMapTxt
  WikiPageNames imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/WikiPageNames
  WikiFormatting imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/WikiFormatting
  WikiRestructuredTextLinks imported from /root/.python-
eggs/Trac-1.1.1-py2.6.egg-tmp/trac/wiki/default-
pages/WikiRestructuredTextLinks
  TracUnicode imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/TracUnicode
  TracChangeset imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/TracChangeset
  TitleIndex imported from /root/.python-eggs/Trac-1.1.1-
```

```
py2.6.egg-tmp/trac/wiki/default-pages/TitleIndex
  WikiDeletePage imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/WikiDeletePage
  TracReports imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/TracReports
  TracWiki imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/TracWiki
  RecentChanges imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/RecentChanges
  TracBackup imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/TracBackup
  TracModPython imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/TracModPython
  TracSearch imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/TracSearch
  TracModWSGI imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/TracModWSGI
  TracTicketsCustomFields imported from /root/.python-
eggs/Trac-1.1.1-py2.6.egg-tmp/trac/wiki/default-
pages/TracTicketsCustomFields
  TracQuery imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/TracQuery
  TracStandalone imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/TracStandalone
  InterTrac imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/InterTrac
  TracFineGrainedPermissions imported from /root/.python-
eggs/Trac-1.1.1-py2.6.egg-tmp/trac/wiki/default-
pages/TracFineGrainedPermissions
  TracInterfaceCustomization imported from /root/.python-
eggs/Trac-1.1.1-py2.6.egg-tmp/trac/wiki/default-
pages/TracInterfaceCustomization
  TracCgi imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/TracCgi
  TracFastCgi imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/TracFastCgi
  TracPermissions imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/TracPermissions
  TracInstall imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/TracInstall
  TracSyntaxColoring imported from /root/.python-eggs/Trac-
1.1.1-py2.6.egg-tmp/trac/wiki/default-pages/TracSyntaxColoring
  CamelCase imported from /root/.python-eggs/Trac-1.1.1-
py2.6.egg-tmp/trac/wiki/default-pages/CamelCase
  WikiMacros imported from /root/.python-eggs/Trac-1.1.1-
```

```
py2.6.egg-tmp/trac/wiki/default-pages/WikiMacros

------------------------------------------------------------
------
Project environment for 'My Project' created.

You may now configure the environment by editing the file:

  /home/git/trac/conf/trac.ini

If you'd like to take this new project environment for a test
drive,
try running the Trac standalone web server `tracd`:

  tracd --port 8000 /home/git/trac

Then point your browser to http://localhost:8000/trac.
There you can also browse the documentation for your installed
version of Trac, including information on further setup (such
as
deploying Trac to a real web server).

The latest documentation can also always be found on the
project
website:

  http://trac.edgewall.org/

Congratulations!
```

## 3.3. Plugin

### AccountManagerPlugin

http://trac-hacks.org/wiki/AccountManagerPlugin

```
cd accountmanagerplugin/
python setup.py install
python setup.py bdist_egg
```

```
cp dist/TracAccountManager-0.4.4-py2.6.egg
/home/git/trac/plugins/
```

## Subtickets

http://trac-hacks.org/wiki/SubticketsPlugin

# 4. trac.ini

## 4.1. repository

```
[trac]
repository_dir = /opt/svnroot/neo
repository_sync_per_request = (default)
repository_type = svn
```

svn 仓库地址 repository_dir

## 4.2. attachment 附件配置

上传附件尺寸控制

```
[attachment]
max_size=262144
```

# 5. trac-admin

权限组定制

```
trac-admin /opt/trac permission add admin TRAC_ADMIN
trac-admin /opt/trac permission add Development TICKET_ADMIN
trac-admin /opt/trac permission add Operations TICKET_ADMIN
```

权限初始化

```
trac-admin /opt/trac permission add mgmt admin
trac-admin /opt/trac permission add neo Development
trac-admin /opt/trac permission add ken Operations

trac-admin /opt/trac permission list
```

```

```

## 5.1. adduser script

```
#!/bin/bash

user=$1
trac-admin /opt/trac permission add $user Operations
htdigest -c /opt/trac/conf/passwd.digest localhost $user
```

# 6. Trac 项目管理

Trac 初始化步骤

1. 首先进入Admin，初始化TRAC
2. 使用Wiki创建项目页
3. 创建Milestones
4. 创建Ticket

## 6.1. Administration

### General

安装后首先分配权限

过程 17.3. Permissions 设置

1. 我习惯于 创建一个 developer 组和 adminstrator 组

   然后创建用户隶属于 developer 组

2. 创建用户隶属于developer组

### Ticket System

过程 17.4. Ticket System 设置

1. 设置 Components

   例如电商项目，这里可以设置，注册登录，用户中心，购物车，物流配送等等

2. 设置 Milestones

   Roadmap->Milestone->Add new milestone

我一般是一个月一个里程碑

3. 设置 Priorities

   我一般设置为:

   新特性（优先）, 不限期, 立即执行, 当日完成, 本周完成, 本月完成

4. Resolutions

   任务完成, 无效BUG, 重复, 待测试, 待发布

5. Severities

   严重错误, 次要错误, 文字错误, 不合理

6. Ticket Types

   Ticket Types 初始化
   1. 开发
   2. 测试
   3. 运维
   4. 设计
   5. 需求
   6. 事件

   7. BUG

7. Versions

   不多说了 1.0， 1。1 或者 1.0.1

**Version Control**

Repositories

默认支持 Subversion, 创建一个仓库记得不要忘记创建下面三个目录 1.branches, 2.tags, 3.trunk

```
trunk                    主干
branches                 在下面再创建两个目录development,testing
tags                     当项目Release 后会在此处打一个标记
```

Git 不需要这三个目录，我习惯上会创建几个分支

```
master                   主干
development              开发分支
testing                  测试分支
```

关于版本库项目目录，我习惯与使用该项目对应的域名作为项目目录

```
/example.com
/example.com/www.exampe.com
/example.com/images.exampe.com
/example.com/user.exampe.com
/example.com/admin.exampe.com
```

## 6.2. Wiki

过程 17.5. Wiki 使用方法

1. 项目成员页，里面要包含所有项目程序的联系方式

```
|name|telephone|cellphone|ext|im|email|
|Neo|13122993040||||
```

2. 需求页面



## 6.3. Timeline

可以看到每时每刻的项目变化，包括Wiki, Ticket, 以及代码提交

## 6.4. Roadmap

Roadmap 中的里程碑页，也可以加以利用，我喜欢将一个里程碑分解为多个Ticket 然后在该页面体现，包括整体上的工作安排等等，使用表格来安排Ticket日程，一定程度上弥补了TRAC没有甘特图的不足，

## 6.5. Ticket

过程 17.6. Ticket 使用方法

1. New Ticket

   新建Ticket, Ticket 可以理解为任务。

2. 将Ticket 分配给团队成员

   受到Ticket后，一定要更改Ticket 为 accept ， 这时在View Tickets 中将会看到该Ticket已经分配，

3. 编码过程

   这里有一个特别的规定，提交代码（包括Subversion与Git）注释中必须这样写:

```
svn ci -m "Ticket #123 - xxxxxxxxxxxxxxxxxx"
git commit -a -m "Ticket #123 - xxxxxxxxxxxxxxxxxx"
```

格式：Ticket #123 - 你的注释

这样写的好处是，在Timeline 中可以直接点击 Ticket 编号直接进入Ticket

```
10:54 AM Ticket #462 (添加一个支付方式) reopened by neo

4:51 PM Changeset in admin.example.com [01a0c4] by neo
<neo.chan@example.com>
Ticket #452 - 用户登录日志
```

4. Add Comment

回复Ticket，上面提交后悔产生一个Subversion版本号，按照下面格式写，然后提交

```
Changesets: r1, [1] or changeset:1
```

这样就可以实现，进入Ticket即可看到做了哪些代码提交与改动，一目了然。

Git 写法

```
[changeset:af212a]
[changeset:7a03c65500c4b96859a27bf5be2901e4ec42afdd]
```

如果 Repositories 中有多个项目写法如下

```
[changeset:af212a/www.example.com]
```

# 7. FAQ

## 7.1. TracError: Cannot load Python bindings for MySQL

检查 MySQLdb 是否安装

```
# /usr/bin/python -c 'import MySQLdb'
Traceback (most recent call last):
  File "<string>", line 1, in <module>
ImportError: No module named MySQLdb
```

安装MySQLdb

```
# yum install python-devel
# pip install MySQL-python
```

或者

```
# yum install python-devel
# easy_install MySQL-python
```

再次测试，如果不出现任何提示表示成功。

```
# /usr/bin/python -c 'import MySQLdb'
```

# 8. Apache Bloodhound

Apache Bloodhound 是基于 Trac 的项目管理软件

# 第 18 章 Redmine

http://www.redmine.org/

redmine 一键安装包

# 1. CentOS 安装

安装MySQL数据库

```
curl -s
https://raw.githubusercontent.com/oscm/shell/master/database/mysql/mysql.server.sh | bash
curl -s
https://raw.githubusercontent.com/oscm/shell/master/database/mysql/mysql.devel.sh | bash
```

创建数据库账号

```
CREATE DATABASE redmine CHARACTER SET utf8;
GRANT ALL PRIVILEGES ON redmine.* TO 'redmine'@'localhost'
IDENTIFIED BY 'my_password';
```

安装

```
yum install -y ruby rubygems ruby-devel ImageMagick-devel

cd /usr/local/src/
wget http://www.redmine.org/releases/redmine-3.3.0.tar.gz
tar zxf redmine-3.3.0.tar.gz
mv redmine-3.3.0 /srv/
ln -s /srv/redmine-3.3.0 /srv/redmine
cd /srv/redmine
```

```
cat >> config/database.yml <<EOF
production:
  adapter: mysql2
  database: redmine
  host: localhost
  username: redmine
  password: my_password
  encoding: utf8
EOF

gem install bundler
bundle install --without development test
bundle exec rake generate_secret_token

RAILS_ENV=production bundle exec rake db:migrate
#bundle exec rake redmine:load_default_data
RAILS_ENV=production REDMINE_LANG=zh bundle exec rake
redmine:load_default_data

mkdir -p tmp tmp/pdf public/plugin_assets
sudo chown -R redmine:redmine files log tmp
public/plugin_assets
sudo chmod -R 755 files log tmp public/plugin_assets

bundle exec rails server webrick -e production
```

默认用户名与密码 login: admin，password: admin

# 2. Redmine 运行

```
# rails server -h
Usage: rails server [mongrel, thin etc] [options]
    -p, --port=port                  Runs Rails on the
specified port.
                                     Default: 3000
    -b, --binding=IP                 Binds Rails to the
specified IP.
                                     Default: localhost
    -c, --config=file                Uses a custom rackup
configuration.
    -d, --daemon                     Runs server as a Daemon.
    -u, --debugger                   Enables the debugger.
    -e, --environment=name           Specifies the environment
to run this server under (test/development/production).
                                     Default: development
    -P, --pid=pid                    Specifies the PID file.
                                     Default:
tmp/pids/server.pid

    -h, --help                       Shows this help message.
```

绑定监听地址 -b

```
# bundle exec rails server webrick -e production -b 0.0.0.0
```

守护进程 -d

# 3. 插件

## 3.1. workflow

http://www.redmine.org/plugins/redmine_workflow_enhancements

# 第 19 章 项目管理工具

# 1. 禅道

创建数据目录

```
[root@netkiller ~]# mkdir -p /opt/zentao/{pms,data}
```

```
[root@netkiller ~]# docker run --name zentao -p 80:80 --
network=zentaonet --ip 172.172.172.172 --mac-address
02:42:ac:11:00:00 -v /opt/zentao/pms:/www/zentaopms -v
/opt/zentao/data:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=passw0rd
-d easysoft/zentao:latest

[root@netkiller ~]# docker run --name zentao -p 80:80 -v
/opt/zentao/pms:/www/zentaopms -v
/opt/zentao/data:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=passw0rd
-d easysoft/zentao:latest
```

# 2. TUTOS

TUTOS is a tool to manage the organizational needs of small groups, teams, departments ...

http://www.tutos.org/

过程 19.1. TUTOS

1. extract

```
                                        tar jxvf TUTOS-php-
1.3.20070317.tar.bz2
                                        sudo mv tutos
/www/htdocs/
```

2. database

```
netkiller@shenzhen:/www/htdocs/tutos$ mysqladmin -uroot -p
create tutos

netkiller@shenzhen:/www/htdocs/tutos$ mysql -uroot -p
                                        Enter password:
                                        Welcome to the
MySQL monitor. Commands end with ; or \g.
                                        Your MySQL
connection id is 846
                                        Server version:
5.0.45 Source distribution

                                        Type 'help;' or
'\h' for help. Type '\c' to clear the buffer.

                                        mysql> grant all on
tutos.* to tutos@% identified by "chen";
                                        Query OK, 0 rows
```

```
affected (0.05 sec)

                                        mysql> grant all on
tutos.* to tutos@localhost identified by "chen";
                                        Query
                                        OK, 0 rows affected
(0.00 sec)

                                        mysql> FLUSH
PRIVILEGES;
                                        Query OK, 0 rows
affected (0.00 sec)

                                        mysql> quit
                                        Bye


netkiller@shenzhen:/www/htdocs/tutos$ mysqladmin -uroot -p
reload
```

3. config

```
                                        mkdir
/www/htdocs/tutos/repository
```

http://192.168.1.7/tutos/php/admin/scheme.php

or

```
                                        cp
config_default.pinc config.php
```

<?php
# remove this line when finsihed with config
$tutos['CCSID'] = "10880f50567242006bf2c1a2c0b8b350";

```
#
# sessionpath
#
$tutos[sessionpath] = "/tmp";
#
# the next lines are a database definition
#
$tutos[dbname][0]     = "tutos";
$tutos[dbhost][0]     = "localhost";
$tutos[dbport][0]     = "5432";
$tutos[dbuser][0]     = "tutos";
$tutos[dbpasswd][0]   = "chen";
$tutos[dbtype][0]     = "2";
$tutos[dbalias][0]    = "Mysql database";
$tutos[cryptpw][0]    = "";
$tutos[repository][0] = "repository";
$tutos[dbprefix][0]   = "";
#
# MAIL
#
$tutos[mailmode] = "2";
$tutos[sendmail] = "/usr/lib/sendmail";
$tutos[smtphost] = "localhost";
#
# demo mode
#
$tutos[demo] = 0;
#
# debug mode
#
$tutos[debug] = 0;
$tutos[errlog] = "/tmp/debug.out";
#
$tutos[jpgraph] = "/www/htdocs/tutos/php/admin/jpgraph";
#
# EOF
```

?>

sudo apt-get install perl libnet-ssleay-perl openssl libauthen-pam-perl libpam-runtime libio-pty-perl libmd5-perl

4. login

   http://192.168.1.7/tutos/php/mytutos.php

   User: superuser Password: tutos

# 部分 III. 软件版本控制

# 第 20 章 Git - Fast Version Control System

## *distributed revision control system*

homepage: http://git.or.cz/index.html

过程 20.1. Git

1. install

```
sudo apt-get install git-core
```

2. config

```
$ git-config --global user.name neo
$ git-config --global user.email openunix@163.com
```

3. Initializ

```
$ mkdir repository
$ cd repository/

/repository$ git-init-db
Initialized empty Git repository in .git/
```

to check .gitconfig file

```
$ cat ~/.gitconfig
[user]
        name = chen
```

```
        email = openunix@163.com
```

# 1. Repositories 仓库管理

## 1.1. initial setup

```
Tell git who you are:

$ git config user.name "FirstName LastName"
$ git config user.email "user@example.com"

If you have many git repositories under your current user, you
can set this for all of them


$ git config --global user.name "FirstName LastName"
$ git config --global user.email "user@example.com"

If you want pretty colors, you can setup the following for
branch, status, and diff commands:

$ git config --global color.branch "auto"
$ git config --global color.status "auto"
$ git config --global color.diff "auto"

Or, to turn all color options on (with git 1.5.5+), use:

$ git config --global color.ui "auto"
To enable aut-detection for number of threads to use (good for
multi-CPU or multi-core computers) for packing repositories,
use:


$ git config --global pack.threads "0"
To disable the rename detection limit (which is set "pretty
low" according to Linus, "just to not cause problems for people
who have less memory in their machines than kernel developers
tend to have"), use:

$ git config --global   diff.renamelimit "0"
```

## 1.2. 克隆代码

克隆到指定目录

```
→  workspace git clone
http://neo@192.168.30.5/netkiller.cn/api.netkiller.cn.git
tmp.netkiller.cn
```

克隆单分支（非 master）

```
 git clone -b 分支名 https://xxx.git
 git clone --branch <branchname> <remote-repo-url>
```

```
$ git clone git://github.com/git/hello-world.git
$ cd hello-world
$ (edit files)
$ git add (files)
$ git commit -m 'Explain what I changed'
```

恢复文件

```
[root@grey lua]# git status
On branch grey
Your branch is up to date with 'origin/grey'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
```

```
  (use "git restore <file>..." to discard changes in working
directory)
        modified:   grey.lua

Untracked files:
  (use "git add <file>..." to include in what will be
committed)
        cache.lua
        flush.lua

no changes added to commit (use "git add" and/or "git commit -
a")

[root@grey lua]# git restore grey.lua
[root@grey lua]# git status
On branch grey
Your branch is up to date with 'origin/grey'.

Untracked files:
  (use "git add <file>..." to include in what will be
committed)
        cache.lua
        flush.lua

nothing added to commit but untracked files present (use "git
add" to track)
```

## 1.3. 切换分支

### git-checkout - Checkout and switch to a branch

### checkout master

```
$ git checkout master
Switched to branch "master"
```

**checkout 分支**

```
$ git branch
* master
  mybranch

$ git checkout mybranch
Switched to branch "mybranch"

$ git branch
  master
* mybranch
```

**通过 checkout 找回丢失的文件**

setup.py 不经意间被删除，找到丢失那一刻的提交是 fda886b0ae1526020c366cea2b747b3ceda18ff6，通过 checkout 检出该文件

```
git checkout fda886b0ae1526020c366cea2b747b3ceda18ff6 --
setup.py
```

重新添加到版本库中

```
git add setup.py
git commit -a -m '还原丢失文件'
git push
```

**checkout 所有远程分支**

```
for branch in $(git branch -r | grep -v HEAD) ; do
# git checkout -b ${branch#*/} $branch;
git checkout ${branch#*/};
git pull;
done
```

## 使用 ours 与 theirs 解决冲突

发生冲突是文件内会出现

```
<<<<<<<HEAD
<ours contents>
=======
<theirs contents>
>>>>>>>
```

使用 --ours 或 --theirs 来选择保留那一方的文件，例如：

```
git checkout --theirs <fileName>
```

冲突解决步骤

```
$ git checkout --ours <fileName>
$ git add <fileName>   //标记该冲突已解决
$ git rebase --continue
$ git status
$ git commit -a -m '冲突已经处理'
$ git push
```

使用远程分支强行覆盖本地分支

方法一

```
neo@MacBook-Pro-M2 ~/w/netkiller (master) > git fetch --all
neo@MacBook-Pro-M2 ~/w/netkiller (master) > git reset --hard
origin/master
```

方法二

```
git pull --force <远程主机> <远程分支>:<本地分支>
```

```
neo@MacBook-Pro-M2 ~/w/netkiller (master) > git pull --force
origin master:master
```

## 1.4. git-add - Add file contents to the index

```
$ echo 'hello world!!!'> newfile
$ git-add newfile
```

## 1.5. Creating and Commiting

```
$ cd (project-directory)
$ git init
$ (add some files)
$ git add .
$ git commit -m 'Initial commit'
```

### git-commit - Record changes to the repository

```
$ git-commit -m 'add a new file' newfile
Created initial commit f6fda79: add a new file
 1 files changed, 1 insertions(+), 0 deletions(-)
 create mode 100644 newfile
```

## 1.6. Status

```
$ git clone git://10.10.0.5/example.git
Cloning into example...
remote: Counting objects: 5, done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 1), reused 0 (delta 0)
Receiving objects: 100% (5/5), done.
Resolving deltas: 100% (1/1), done.

neo@neo-OptiPlex-380:~/tmp$ cd example/

neo@neo-OptiPlex-380:~/tmp/example$ git status
# On branch master
nothing to commit (working directory clean)

neo@neo-OptiPlex-380:~/tmp/example$ ls
test1  test2  test3  test4

neo@neo-OptiPlex-380:~/tmp/example$ echo hello > test1

neo@neo-OptiPlex-380:~/tmp/example$ git status
# On branch master
```

```
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in
working directory)
#
#       modified:   test1
#
no changes added to commit (use "git add" and/or "git commit -
a")
```

### git-status - Show the working tree status

```
$ git-status newfile
# On branch master
#
# Initial commit
#
# Changes to be committed:
#   (use "git rm --cached <file>..." to unstage)
#
#       new file: newfile
#
```

## 1.7. Diff

```
neo@neo-OptiPlex-380:~/tmp/example$ git diff
diff --git a/test1 b/test1
index e69de29..ce01362 100644
--- a/test1
+++ b/test1
@@ -0,0 +1 @@
+hello
```

比较 nqp-cc/src/QASTCompilerMAST.nqp 文件 当前版本与 211ab0b19f25b8c81685a97540f4b1491eb17504 版本的区别

```
git diff 211ab0b19f25b8c81685a97540f4b1491eb17504 -- nqp-
cc/src/QASTCompilerMAST.nqp
```

**--name-only 仅显示文件名**

```
git diff --name-only
```

## 1.8. Push

```
$ git clone git://10.10.0.5/example.git
$ cd example
$ (edit files)
$ git add (files)
$ git commit -m 'Explain what I changed'

$ git push origin master
Counting objects: 5, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 278 bytes, done.
Total 3 (delta 0), reused 0 (delta 0)
To git://10.10.0.5/example.git
   27f8417..b088cc3  master -> master
```

## 1.9. Pull

```
$ git pull
remote: Counting objects: 5, done.
remote: Compressing objects: 100% (2/2), done.
```

```
remote: Total 3 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
From git://10.10.0.5/example
   27f8417..b088cc3  master     -> origin/master
Updating 27f8417..b088cc3
Fast-forward
 test1 |    1 +
 1 files changed, 1 insertions(+), 0 deletions(-)
```

## 1.10. fetch

```
$ git fetch
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 2 (delta 1), reused 0 (delta 0)
Unpacking objects: 100% (2/2), done.
From git://10.10.0.5/example
   b088cc3..7e8c17d  master     -> origin/master
```

## 1.11. Creating a Patch

```
$ git clone git://github.com/git/hello-world.git
$ cd hello-world
$ (edit files)
$ git add (files)
$ git commit -m 'Explain what I changed'
$ git format-patch origin/master
```

## 1.12. reset

重置到上一个版本

```
git log
git reset --hard HEAD^
```

```
git log
git push -f
```

## 还原文件

```
$ git checkout <commit> --filename
$ git reset filename
```

# 2. 分支管理

## Manipulating branches

git-branch - List, create, or delete branches

## 2.1. 查看本地分支

```
$ git branch
* master
```

查看远程分支

```
git branch -a
```

## 2.2. 创建分支

```
$ git branch development
$ git branch
* master
  development
```

机遇分支创建分支

```
$ git checkout -b feature develop
$ git push --set-upstream origin feature
```

```
```

## 2.3. 删除分支

```
$ git branch -d staging
Deleted branch staging.

$ git branch
* master
```

## 2.4. 切换分支

```
$ git branch
* master
  testing

$ git checkout testing
Switched to branch "testing"

$ git branch
  master
* testing
```

## 2.5. 重命名分支

```


neo@MacBook-Pro-Neo ~/workspace/api.netkiller.cn % git checkout
test
neo@MacBook-Pro-Neo ~/workspace/api.netkiller.cn % git branch -
m test testing
neo@MacBook-Pro-Neo ~/workspace/api.netkiller.cn % git push --
delete origin test
neo@MacBook-Pro-Neo ~/workspace/api.netkiller.cn % git push
origin testing


```

## 2.6. git-show-branch - Show branches and their commits

```
$ git-show-branch
! [master] add a new file
 * [mybranch] add a new file
--
+* [master] add a new file
```

# 3. git log

```
git log --graph --pretty=format:'%Cred%h%Creset -
%C(yellow)%d%Creset %s %Cgreen(%cr) %C(bold blue)<%an>%Creset'
--abbrev-commit

git log --graph --pretty=format:'%Cred%h%Creset -
%C(yellow)%d%Creset %s %Cgreen(%ai) %C(bold blue)<%an>%Creset'
--abbrev-commit
```

## 3.1. hash-object

使用git命令计算文件的 sha-1 值

```
neo@MacBook-Pro ~ % echo 'test content' | git hash-object --
stdin
d670460b4b4aece5915caf5c68d12f560a9fe3e4
```

## 3.2. 一行显示 --oneline

```
Neo-iMac:test.netkiller.cn neo$ git log --name-status
commit 120f1bb6ff391c6b9b24899804f0292370873485 (HEAD -> main)
Author: 陈景峰 <neo@netkiller.cn>
Date:   Thu Dec 2 04:10:16 2021 +0000

    Initial commit

A       README.md
Neo-iMac:test.netkiller.cn neo$ git log --name-status --oneline
120f1bb (HEAD -> main) Initial commit
```

```
A       README.md
```

```
Neo-iMac:www.netkiller.cn neo$ git log --name-status --oneline
--graph
* 7ca7fb7 (HEAD -> main, origin/main, origin/HEAD) sign
| M     README.md
* ba9a9a6 更新.gitlab-ci.yml文件
| M     .gitlab-ci.yml
* 8af932e 更新.gitlab-ci.yml文件
| M     .gitlab-ci.yml
* 6fe467b Update app.js
| M     app.js
* a019da0 更新.gitlab-ci.yml文件
| M     .gitlab-ci.yml
* 65afb8b 更新.gitlab-ci.yml文件
| M     .gitlab-ci.yml
* 061c78d 更新.gitlab-ci.yml文件
| A     .gitlab-ci.yml
* 149daf5 Add new file
| A     app.js
* e927196 更新README.md
  A     README.md
```

## 3.3. 查看文件历史记录

```
neo@MacBook-Pro-Neo ~/workspace/devops % git log -- setup.py
```

diff 风格

```
neo@MacBook-Pro-Neo ~/workspace/devops % git log -p -- setup.py
```

```
commit abe282e68ad81e0e72cb8c700ba5c4db87c647a4
Author: neo <netkiller@msn.com>
Date:   Thu Sep 30 14:07:02 2021 +0800

    voice

diff --git a/setup.py b/setup.py
deleted file mode 100644
index 08f9d08..0000000
--- a/setup.py
+++ /dev/null
@@ -1,59 +0,0 @@
-import os,sys
-from setuptools import setup,find_packages
-sys.path.insert(0, os.path.abspath('lib'))
-from netkiller import __version__, __author__
-
-with open("README.md", "r") as fh:
-  long_description = fh.read()
-
-setup(
-       name="netkiller-devops",
-       version="0.2.4",
```

oneline 风格

```
neo@MacBook-Pro-Neo ~/workspace/devops % git log --
pretty=oneline -- setup.py
abe282e68ad81e0e72cb8c700ba5c4db87c647a4 voice
fda886b0ae1526020c366cea2b747b3ceda18ff6 语音通知
cb2ca23a81b2384b79d7b32bb2e84782bb80edaf 企业微信通知
ac8e573123142a2856d44d13307dd4c46b134ceb fixed logging bug
1c609b9242c8f404ec4bba207dd8c9d836e591d4 docker 增加日志功能
```

## 3.4. 格式化

格式参数

```
%H: 标准长度 commit hash
%h: 缩短的 commit hash
%T: tree hash
%t: 缩短的 tree hash
%P: parent hashes
%p: 缩短的 parent hashes
%an: 作者名字
%aN: mailmap的作者名字 (.mailmap对应，详情参照git-shortlog(1)或者
git-blame(1))
%ae: 作者邮箱
%aE: 作者邮箱 (.mailmap对应，详情参照git-shortlog(1)或者git-
blame(1))
%ad: 日期 (--date= 制定的格式)
%aD: 日期, RFC2822格式
%ar: 日期, 相对格式(1 day ago)
%at: 日期, UNIX timestamp
%ai: 日期, ISO 8601 格式
%cn: 提交者名字
%cN: 提交者名字 (.mailmap对应，详情参照git-shortlog(1)或者git-
blame(1))
%ce: 提交者 email
%cE: 提交者 email (.mailmap对应，详情参照git-shortlog(1)或者git-
blame(1))
%cd: 提交日期 (--date= 制定的格式)
%cD: 提交日期, RFC2822格式
%cr: 提交日期, 相对格式(1 day ago)
%ct: 提交日期, UNIX timestamp
%ci: 提交日期, ISO 8601 格式
%d: ref名称
%e: encoding
%s: commit信息标题
%f: sanitized subject line, suitable for a filename
%b: commit信息内容
%N: commit notes
%gD: reflog selector, e.g., refs/stash@{1}
%gd: shortened reflog selector, e.g., stash@{1}
%gs: reflog subject
%Cred: 切换到红色
%Cgreen: 切换到绿色
%Cblue: 切换到蓝色
%Creset: 重设颜色
%C(...): 制定颜色, as described in color.branch.* config option
```

```
%m: left, right or boundary mark
%n: 换行
%%: a raw %
%x00: print a byte from a hex code
%w([[,[,]]]): switch line wrapping, like the -w option of git-
shortlog(1)
```

命令演示

```
git log --since="2023-02-11" --no-merges --pretty=format:"%h
%an %ai %s"
```

# 4. reflog

reflog 类似我们软件中的 Undo/Redo ，就像使用 CMD+Z / CMD + SHIFT +Z 一样进行版本的切换和回滚。reflog 日志是保存在本地的，并不会 push 到远程，这就是他与 git log 的区别。

git reflog 与 git log的区别，git log 可以显示所有提交过的版本信息，但不包括已经被删除的 commit 记录和 reset 的操作

git reflog 可以是显示所有的操作记录，包括提交，回退的操作。一般用来找出操作记录中的版本号，进行回退，常用于恢复本地的错误操作。

git reflog 用法

```
Neo-iMac:test.netkiller.cn neo$ git reflog
120f1bb (HEAD -> main) HEAD@{0}: reset: moving to 120f1bb
9fcccf0 HEAD@{1}: commit: add tmp string
de5ca5d (origin/main, origin/HEAD) HEAD@{2}: reset: moving to
HEAD
de5ca5d (origin/main, origin/HEAD) HEAD@{3}: pull: Fast-forward
120f1bb (HEAD -> main) HEAD@{4}: clone: from
192.168.30.5:netkiller.cn/test.netkiller.cn.git
```

回滚到 120f1bb

```
Neo-iMac:test.netkiller.cn neo$ git reset --hard 120f1bb
HEAD is now at 120f1bb Initial commit
```

# 5. 远程仓库

## 5.1. 查看远程地址

查看远程仓库

```
git remote show
origin
```

```
neo@MacBook-Pro-Neo ~> git remote -v
origin  git@192.168.30.5:netkiller.cn/www.netkiller.cn.git
(fetch)
origin  git@192.168.30.5:netkiller.cn/www.netkiller.cn.git
(push)
```

```
neo@MacBook-Pro-M2 ~/netkiller (dev)> git remote get-url origin
ssh://git@gitlab.netkiller.cn/galaxy/ensd-fscs.git
```

显示远程地址

```
neo@MacBook-Pro-M2 ~/w/test (master)> git ls-remote --get-url
origin
ssh://git@gitlab.netkiller.cn:/chenjingfeng/test.git
```

## 5.2. 添加远程仓库

添加远程仓库

```
git remote add origin git@localhost:example.git
```

添加多个远程仓库

```
git remote add origin git@localhost:example.git
git remote add another
https://gitcafe.com/netkiller/netkiller.gitcafe.com.git
git push origin master
git push another master
```

## 5.3. 修改 origin

```
git remote rename origin old-origin
```

修改远程仓库

```
git remote set-url origin
git@gitlab.netkiller.cn:netkiller.cn/www.netkiller.cn.git
git remote set-url origin
https://gitlab.netkiller.cn/netkiller.cn/www.netkiller.cn.git
```

## 5.4. 删除 origin

```
git remote remove origin
```

删除远程仓库

```
git remote rm origin
```

## 5.5. 仓库共享

### Setting up a git server

First we need to setup a user with a home folder. We will store all the repositories in this users home folder.

```
sudo adduser git
```

Rather than giving out the password to the git user account use ssh keys to login so that you can have multiple developers connect securely and

easily.

Next we will make a repository. For this example we will work with a repository called example. Login as the user git and add the repository.

login to remote server

```
ssh git@REMOTE_SERVER
```

once logged in

```
sudo mkdir example.git
cd example.git
sudo git --bare init
Initialized empty Git repository in /home/git/example.git/
```

That's all there is to creating a repository. Notice we named our folder with a .git extension.

Also notice the 'bare' option. By default the git repository assumes that you'll be using it as your working directory, so git stores the actual bare repository files in a .git directory alongside all the project files. Since we are setting up a remote server we don't need copies of the files on the filesystem. Instead, all we need are the deltas and binary objects of the repository. By setting 'bare' we tell git not to store the current files of the repository only the diffs. This is optional as you may have need to be able to browse the files on your remote server.

Finally all you need to do is add your files to the remote repository. We will assume you don't have any files yet.

```
mkdir example
cd example
git init
touch README
git add README
git commit -m 'first commit'
git remote add origin git@REMOTE_SERVER:example.git
git push origin master
```

replace REMOTE_SERVER with your server name or IP

# 6. git show - Show various types of objects

```
$ git show
commit f6fda79f2f550ea3b2c1b483371ed5d12499ac35
Author: chen <openunix@163.com>
Date:   Sat Nov 1 08:50:45 2008 -0400

    add a new file

diff --git a/newfile b/newfile
new file mode 100644
index 0000000..b659464
--- /dev/null
+++ b/newfile
@@ -0,0 +1 @@
+hello world!!!
```

## 6.1. 查看指定版本的文件内容

```
neo@MacBook-Pro-Neo ~/workspace/devops % git show
fda886b0ae1526020c366cea2b747b3ceda18ff6:setup.py
```

# 7. 合并分支

## 7.1. 合并分支

从 development 像 testing 分支合并

```
git checkout development
git pull
git checkout testing
git pull
git merge --no-ff "development"
git push
```

testing 分支向 master 分支合并

获取 testing 合并请求的分支

```
git fetch origin
git checkout -b "testing" "origin/testing"
```

如果此前已经执行过，使用下面命令切换分支即可，切换后 pull 代码，看看有什么新提交

```
git checkout "testing"
git pull
```

切换到 master 分支

```
git fetch origin
git checkout "master"
git branch --show-current
git merge --no-ff "testing"
```

将合并结果推送到远程

```
git push origin "master"
```

## 7.2. rebase

恢复 rebase 版本

```
git rebase
git reflog
git reset --hard 5faf0ae
git push
```

## 7.3. 合并分支解决冲突

案例，例如我们从 testing 分支向 master 分支合并代码出现冲突，该如何解决呢?

首先，两个分支拉取最新代码

```
neo@MacBook-Pro-Neo ~/workspace/api.netkiller.cn % git checkout
testing
```

```
neo@MacBook-Pro-Neo ~/workspace/api.netkiller.cn % git pull
neo@MacBook-Pro-Neo ~/workspace/api.netkiller.cn % git checkout
master
neo@MacBook-Pro-Neo ~/workspace/api.netkiller.cn % git pull
```

然后合并分支，从 testing 分支向 master 合并

```
neo@MacBook-Pro-Neo ~/workspace/api.netkiller.cn % git merge --
no-ff testing
自动合并 neo-
incar/src/main/java/com/neo/incar/utils/PaperlessConfig.java
冲突（内容）：合并冲突于 neo-
incar/src/main/java/com/neo/incar/utils/PaperlessConfig.java
自动合并失败，修正冲突然后提交修正的结果。
```

出现冲突，编辑冲突文件

```
vim neo-
incar/src/main/java/com/neo/incar/utils/PaperlessConfig.java
```

保存后重看状态

```
neo@MacBook-Pro-Neo ~/workspace/api.netkiller.cn % git status
位于分支 master
您的分支与上游分支 'origin/master' 一致。

您有尚未合并的路径。
  （解决冲突并运行 "git commit"）
  （使用 "git merge --abort" 终止合并）

要提交的变更：
```

```
        修改:      neo-admin/src/main/resources/application-
prod.yml
        修改:      neo-admin/src/main/resources/application-
test.yml
        修改:      neo-
common/src/main/java/com/neo/common/enums/IncarAttachTypeEnum.j
ava
        修改:      neo-
incar/src/main/java/com/neo/incar/service/impl/IncarAttachServi
ceImpl.java

未合并的路径:
  (使用 "git add <文件>..." 标记解决方案)
        双方修改:    neo-
incar/src/main/java/com/neo/incar/utils/PaperlessConfig.java
```

将合并的文件添加到 git

```
neo@MacBook-Pro-Neo ~/workspace/api.netkiller.cn % git add neo-
incar/src/main/java/com/neo/incar/utils/PaperlessConfig.java
neo@MacBook-Pro-Neo ~/workspace/api.netkiller.cn % git status
位于分支 master
您的分支与上游分支 'origin/master' 一致。

所有冲突已解决但您仍处于合并中。
  (使用 "git commit" 结束合并)

要提交的变更:
        修改:      neo-admin/src/main/resources/application-
prod.yml
        修改:      neo-admin/src/main/resources/application-
test.yml
        修改:      neo-
common/src/main/java/com/neo/common/enums/IncarAttachTypeEnum.j
ava
        修改:      neo-
incar/src/main/java/com/neo/incar/service/impl/IncarAttachServi
ceImpl.java
        修改:      neo-
incar/src/main/java/com/neo/incar/utils/PaperlessConfig.java
```

```

```

提交代码

```
neo@MacBook-Pro-Neo ~/workspace/api.netkiller.cn % git commit -
a -m '手工合并分支 testing -> master'
[master 3652bf8e] 手工合并分支 testing -> master
```

推送代码

```
neo@MacBook-Pro-Neo ~/workspace/api.netkiller.cn % git push
枚举对象中：1，完成.
对象计数中：100% (1/1)，完成.
写入对象中：100% (1/1)，240 字节 | 240.00 KiB/s，完成.
总共 1（差异 0），复用 0（差异 0），包复用 0
remote:
remote: To create a merge request for master, visit:
remote:
http://192.168.30.5/netkiller.cn/api.netkiller.cn/-/merge_reque
sts/new?merge_request%5Bsource_branch%5D=master
remote:
To http://192.168.30.5/netkiller.cn/api.netkiller.cn.git
   fcaefaf4..3652bf8e  master -> master
```

## 7.4. 终止合并

```
git merge --about
```

## 7.5. 合并单个文件

从 development 到 testing

```
git checkout development
git pull
checkout testing
git checkout development public/doc/UserGuide.pdf
git status
git commit -a -m '手工合并'
git push
```

从 testing 到 staging

```
git checkout staging
git pull
git checkout testing public/doc/UserGuide.pdf
git commit -a -m '手工合并'
git push
```

从 stage 到 master

```
git checkout master
git pull
git checkout staging public/doc/UserGuide.pdf
git commit -a -m '手工合并'
git push
```

## 7.6. Git 合并特定 commits 到另一个分支

用法

```
git cherry-pick [<options>] <commit-ish>...

常用options:
    --quit                  退出当前的chery-pick序列
    --continue              继续当前的chery-pick序列
    --abort                 取消当前的chery-pick序列，恢复当前分支
    -n, --no-commit         不自动提交
    -e, --edit              编辑提交信息
```

操作步骤

```
git log --oneline -3
git switch test

git log --oneline -3
git cherry-pick 2c0a8637a46c2f22eb703a9be7f09d005ed390ff
git push
git log --oneline -3
```

找到我们需要合并的 commit，我需要合并的是 2c0a8637a46c2f22eb703a9be7f09d005ed390ff

```
neo@MacBook-Pro-M2 ~/w/netkiller (master)> git log -3
commit 2c0a8637a46c2f22eb703a9be7f09d005ed390ff (HEAD ->
master, origin/master)
Author: Neo Chan <netkiller@msn.com>
Date:   Tue Mar 7 13:59:26 2023 +0800

    [TASK#12773 定时任务增加日志输出和执行结果钉钉通知]
(https://zentao.netkiller.cn/zentao/task-view-12773.html)

commit ada2c7e1c8cd1b9f306050e6ad95a7fe1406ab41
Author: Neo Chan <netkiller@msn.com>
```

```
Date:   Mon Mar 6 18:54:26 2023 +0800

    [TASK#12744 Excel 生成错误]
(https://zentao.netkiller.cn/zentao/task-view-12744.html)

commit e0b828fa3941bb7d8698322a4e4b2c96aa3fe841
(origin/branch_order_gross_profit_20230302,
origin/branch_dongguan_shell_20230302)
Merge: bf3e45a d00eefb
Author: Neo Chan <netkiller@msn.com>
Date:   Tue Feb 28 09:51:55 2023 +0800

    Merge branch 'grey'
```

切换道目的分支

```
neo@MacBook-Pro-M2 ~/w/netkiller (master)> git switch test
Switched to branch 'test'
Your branch is up to date with 'origin/test'.
neo@MacBook-Pro-M2 ~/w/netkiller (test)> git pull
Already up to date.
```

查看目的分支的日志

```
neo@MacBook-Pro-M2 ~/w/netkiller (test)> git log --oneline -3
f8bf5e1 (HEAD -> test, origin/test) [TASK#12744 Excel 生成错误]
(https://zentao.netkiller.cn/zentao/task-view-12744.html)
a42d15d Merge branch 'bugfix-online-20230214' into test
c8229b1 [BUG #0] 收入成本excel优化
```

合并代码，然后push代码

```
neo@MacBook-Pro-M2 ~/w/netkiller (test)> git cherry-pick
2c0a8637a46c2f22eb703a9be7f09d005ed390ff
[test 235aa71] [TASK#12773 定时任务增加日志输出和执行结果钉钉通知]
(https://zentao.netkiller.cn/zentao/task-view-12773.html)
 Date: Tue Mar 7 13:59:26 2023 +0800
 8 files changed, 66 insertions(+), 13 deletions(-)
neo@MacBook-Pro-M2 ~/w/netkiller (test)> git push
```

合并成功

```
neo@MacBook-Pro-M2 ~/w/netkiller (test)> git log --oneline -1
235aa71 (HEAD -> pre, origin/pre) [TASK#12773 定时任务增加日志输出
和执行结果钉钉通知](https://zentao.netkiller.cn/zentao/task-view-
12773.html)
```

# 8. 比较文件

## 8.1. 比较 SHA

```
neo@MacBook-Pro-M2 ~/w/netkiller (dev)> git diff
cef86cfdb62de992b587c07425c8e7fa927a4b4d
66da93554af5d2e2398eb474b6981aa85a91547c
```

## 8.2. 分支比较

比较两个分支的差异

```
neo@MacBook-Pro-M2 ~/w/netkiller (test)> git diff dev test --
stat
```

比较两个分支的文件差异

```
neo@MacBook-Pro-M2 ~/w/netkiller (test)> git diff dev test
src/main/resources/bootstrap.yml
```

# 9. Submodule 子模块

## 9.1. 添加模块

```
neo@MacBook-Pro ~ % cd workspace/Linux

neo@MacBook-Pro ~/workspace/Linux % git submodule add
https://github.com/netkiller/common.git common
Cloning into '/Users/neo/workspace/Linux/common'...
remote: Enumerating objects: 9, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (8/8), done.
remote: Total 185 (delta 2), reused 6 (delta 1), pack-reused
176
Receiving objects: 100% (185/185), 56.49 KiB | 163.00 KiB/s,
done.
Resolving deltas: 100% (105/105), done.
```

模块信息存储在 .gitmodules 文件中

```
neo@MacBook-Pro ~/workspace/Linux % cat .gitmodules
[submodule "common"]
        path = common
        url = https://github.com/netkiller/common.git
```

同时也添加到 .git/config 文件中

```
neo@MacBook-Pro ~/workspace/Linux % cat .git/config | tail -n 3
[submodule "common"]
        url = https://github.com/netkiller/common.git
        active = true
```

## 9.2. checkout 子模块

clone 项目，然后进入目录

```
neo@MacBook-Pro /tmp/test % git clone
https://github.com/netkiller/Linux.git
neo@MacBook-Pro /tmp/test % cd Linux
```

初始化子模块

```
neo@MacBook-Pro /tmp/test/Linux % git submodule init
Submodule 'common' (https://github.com/netkiller/common.git)
registered for path 'common'
```

更新模块

```
neo@MacBook-Pro /tmp/test/Linux % git submodule update
Cloning into '/private/tmp/test/Linux/common'...
Submodule path 'common': checked out
'cdf61a1de34590bcc80b895fdf0e90b62cfd729f'
```

## 9.3. 删除子模块

```
git rm --cached <module>
```

# 10. Git Large File Storage

https://git-lfs.github.com/

Git Large File Storage | Git Large File Storage (LFS) replaces large files such as audio samples, videos, datasets, and graphics with text pointers inside Git, while storing the file contents on a remote server like GitHub.com or GitHub Enterprise.

```
/usr/bin/ruby -e "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/master/insta
ll)"

brew install git-lfs
```

## 10.1. 安装 LFS 支持

```
git lfs install
git lfs track "*.psd"
git add .gitattributes

git add file.psd
git commit -m "Add design file"
git push origin master
```

## 10.2. LFS lock

文件锁的用途是用户可以对一个文件进行加锁，阻止其他用户同一时间对该文件进行修改操作。因为在GIT仓库中同时编辑一个文

件，会发生冲突，然而解决二进制大文件的冲突，合并操作极其困难。

```
neo@MacBook-Pro ~/workspace/java-project % git lfs lock
test.psd
Locked test.psd

neo@MacBook-Pro ~/workspace/java-project % git lfs locks
test.psd        bg7nyt  ID:55777
```

如果Push被锁的文件，提示 Remote "origin" does not support the LFS locking API

```
neo@MacBook-Pro /tmp/java % git commit -a -m 'aaa'
[master b832eb3] aaa
 1 file changed, 2 insertions(+), 2 deletions(-)
neo@MacBook-Pro /tmp/java % git push
Remote "origin" does not support the LFS locking API. Consider
disabling it with:
  $ git config
'lfs.https://github.com/bg7nyt/java.git/info/lfs.locksverify'
false
Post https://github.com/bg7nyt/java.git/info/lfs/locks/verify:
EOF
error: failed to push some refs to
'https://github.com/bg7nyt/java.git'
```

解锁后Push成功

```
neo@MacBook-Pro ~/workspace/java-project % git lfs unlock
test.psd
Unlocked test.psd
```

```
neo@MacBook-Pro /tmp/java % git push
Git LFS: (1 of 1 files) 9 B / 9 B
Counting objects: 3, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 352 bytes | 352.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local
object.
To https://github.com/bg7nyt/java.git
   b29f474..b832eb3  master -> master
```

# 11. git config

## 11.1. git config

```
$ git config --file config http.receivepack true
```

## 11.2. 查看配置

```
Neo-iMac:workspace neo$ git config --list
credential.helper=osxkeychain
user.name=Neo Chen
user.email=netkiller@msn.com
user.signingkey=netkiller@msn.com
gpg.program=gpg
commit.gpgsign=true
```

## 11.3. 编辑配置

```
git config --global --edit
```

```
git config --edit
```

## 11.4. 替换配置项

```
$  git config --global --replace-all user.email "输入你的邮箱"
$  git config --global --replace-all user.name "输入你的用户名"
```

## 11.5. 配置默认分之

```
git config --global init.defaultBranch <名称>
```

## 11.6. GPG签名

```
开启GPG签名:

git config commit.gpgsign true

关闭:

git config commit.gpgsign false
```

## 11.7. core.sshCommand

git 默认使用 id_rsa，指定私钥方法是:

```
git config core.sshCommand "ssh -i ~/.ssh/id_rsa_example -F
/dev/null"
git pull
git push
```

```
GIT_SSH_COMMAND="ssh -i ~/.ssh/id_rsa_example -F /dev/null" git
clone git@github.com:netkiller/netkiller.github.io.git
```

## 11.8. fatal: The remote end hung up unexpectedly

```
error: RPC failed; result=22, HTTP code = 413 | 18.24 MiB/s
fatal: The remote end hung up unexpectedly
```

```
git config http.postBuffer 524288000
```

## 11.9. 忽略 SSL 检查

使用自颁发 ssl 证书时需要开启，否则无法 clone 和 push

```
export GIT_SSL_NO_VERIFY=true
```

```
git config http.sslVerify "false"
```

## 11.10. 配置忽略合并文件

从一个分支向另一个分支合并，有时我们不想覆盖某个文件。

```
neo@MacBook-Pro-M2 ~/netkiller (dev)> git config
```

```
merge.ours.driver true
```

创建 .gitattributes 文件

```
neo@MacBook-Pro-M2 ~/netkiller (office)> cat .gitattributes
src/main/resources/bootstrap.yml merge=ours
```

全局配置

```
git config --global merge.ours.driver true
```

## 11.11. .gitignore

```
find ./ -type d -empty | grep -v \.git | xargs -i touch
{}/.gitignore
```

## 11.12. .gitattributes

### SVN Keywords

```
Example:

$ echo '*.txt ident' >> .gitattributes
$ echo '$Id$' > test.txt
$ git commit -a -m "test"

$ rm test.txt
```

```
$ git checkout -- test.txt
$ cat test.txt
```

```
```

```
```

```
```

### 设置文件换行符

开发者使用不同的操作系统，不同的开发工具，保存的文件换行符有三种，CR、LF、CRLF，也就是回车CR和换行的组合。

解决方案，在项目根目录下创建 .gitattributes 文件

```
*.js    eol=lf
*.jsx   eol=lf
*.json  eol=lf
```

## 11.13. 配置模版目录

配置模版之后，克隆项目会自动创建模版下的目录和文件

```
mkdir -p ~/workspace/template/hooks
```

配置模版目录

```
git config --global init.templatedir ~/workspace/template/
```

创建文件

```
curl -s
https://gitlab.ejiayou.com/chenjingfeng/zentao/-/raw/master/com
mit-msg -o ~/workspace/hooks/commit-msg
chmod +x ~/workspace/hooks/commit-msg
```

现在执行 git clone 之后你会发现 .git/hooks 目录会产生一个 commit-msg 文件

# 12. git-rev-parse - Pick out and massage parameters

## 12.1. 获得当前提交ID

```
root@netkiller /d/s/test (master)# git rev-parse HEAD
a7b5dc12c595e8abaae8800a86f93b475e833ce3
```

```
root@netkiller /d/s/test (master)# git rev-parse --short HEAD
a7b5dc1
```

# 13. git-daemon 服务器

## 13.1. git-daemon - A really simple server for git repositories

在/home/gitroot/ 上运行 git 守护进程

```
$ cd /home/gitroot
$ mkdir test.git
$ cd test.git
$ git --bare init --shared
Initialized empty shared Git repository in
/home/gitroot/test.git/
```

```
git daemon --verbose --export-all --base-path=/home/gitroot --
enable=receive-pack --reuseaddr
```

允许push,否则该仓库只能clone/pull

```
sudo git daemon --verbose --export-all --base-
path=/home/gitroot --enable=upload-pack --enable=upload-archive
--enable=receive-pack
```

或者增加配置项

```
$ git config daemon.receivepack true
$ git config --file config receive.denyCurrentBranch ignore
```

## 13.2. git-daemon-sysvinit

for a read-only repo:

```
$ sudo apt-get install git-daemon-sysvinit


$ dpkg -l git-daemon-sysvinit
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Inst/Conf-files/Unpacked/halF-conf/Half-inst/trig-
aWait/Trig-pend
|/ Err?=(none)/Reinst-required (Status,Err: uppercase=bad)
||/ Name                                  Version
Architecture             Description
+++-=====================================-
=======================-=======================-
================================================================
==================
ii  git-daemon-sysvinit                   1:1.7.10.4-1ubuntu1
all                      fast, scalable, distributed revision
control system (git-daemon service)


$ dpkg -L git-daemon-sysvinit
/.
/usr
/usr/share
/usr/share/git-core
/usr/share/git-core/sysvinit
/usr/share/git-core/sysvinit/sentinel
/usr/share/doc
/usr/share/doc/git-daemon-sysvinit
/usr/share/doc/git-daemon-sysvinit/copyright
/usr/share/doc/git-daemon-sysvinit/README.Debian
/etc
/etc/default
/etc/default/git-daemon
/etc/init.d
/etc/init.d/git-daemon
/usr/share/doc/git-daemon-sysvinit/changelog.Debian.gz
```

配置 /etc/default/git-daemon 文件

## 13.3. inet.conf / xinetd 方式启动

过程 20.2. git-daemon

1. /etc/shells

   /etc/shells 最后一行添加 '/usr/bin/git-shell'

```
$ grep git /etc/shells
/usr/bin/git-shell
```

2. add new user 'git' and 'gitroot' for git

   you need to assign shell with /usr/bin/git-shell

```
$ sudo adduser git --shell /usr/bin/git-shell
$ sudo adduser gitroot --ingroup git --shell /bin/bash
```

   /etc/passwd

```
$ grep git /etc/passwd
git:x:1001:1002:,,,:/home/git:/usr/bin/git-shell
gitroot:x:1002:1002:,,,:/home/gitroot:/bin/bash
```

3. /etc/services

```
$ grep 9418 /etc/services
git             9418/tcp                        # Git
Version Control System
```

4. /etc/inet.conf

```
$ grep git /etc/inet.conf
git     stream  tcp     nowait  nobody \
/usr/bin/git-daemon git-daemon --inetd --syslog --export-
```

```
all /home/gitroot
```

reload inetd

```
$ sudo pkill -HUP inetd
```

5. xinetd

目前的Linux逐渐使用xinetd.d替代inet.conf，如Redhat系列已经不再使用inet.conf, Ubuntu系列发行版已经不预装inet与xinetd

```
$ apt-cache search xinetd
globus-gfork-progs - Globus Toolkit - GFork Programs
rlinetd - gruesomely over-featured inetd replacement
update-inetd - inetd configuration file updater
xinetd - replacement for inetd with many enhancements

$ sudo apt-get install xinetd
```

/etc/xinetd.d/

```
$ cat /etc/xinetd.d/git
# default: off
# description: The git server offers access to git
repositories
service git
{
        disable                 = no
        type            = UNLISTED
        port            = 9418
        socket_type     = stream
        protocol                = tcp
        wait            = no
        user            = gitroot
        server          = /usr/bin/git
        server_args     = daemon --inetd --export-all --
enable=receive-pack --reuseaddr --base-path=/home/gitroot
```

```
        log_on_failure  += USERID
}
```

reload xinitd

```
$ sudo /etc/init.d/xinetd reload
* Reloading internet superserver configuration xinetd
[ OK ]
```

## 13.4. git-daemon-run

```
$ sudo apt-get install git-daemon-run
```

安装后会创建下面两个用户

```
$ cat /etc/passwd | grep git
gitlog:x:117:65534::/nonexistent:/bin/false
gitdaemon:x:118:65534::/nonexistent:/bin/false
```

git-daemon-run 包携带的文件

```
$ dpkg -L git-daemon-run
/.
/etc
/etc/sv
/etc/sv/git-daemon
/etc/sv/git-daemon/run
/etc/sv/git-daemon/log
/etc/sv/git-daemon/log/run
/usr
/usr/share
/usr/share/doc
/usr/share/doc/git-daemon-run
/usr/share/doc/git-daemon-run/changelog.gz
```

```
/usr/share/doc/git-daemon-run/changelog.Debian.gz
/usr/share/doc/git-daemon-run/README.Debian
/usr/share/doc/git-daemon-run/copyright
```

同时创建下面配置文件

```
$ find /etc/sv/git-daemon/
/etc/sv/git-daemon/
/etc/sv/git-daemon/run
/etc/sv/git-daemon/supervise
/etc/sv/git-daemon/log
/etc/sv/git-daemon/log/run
/etc/sv/git-daemon/log/supervise
```

编辑/etc/sv/git-daemon/run配置

```
$ sudo vim /etc/sv/git-daemon/run

#!/bin/sh
exec 2>&1
echo 'git-daemon starting.'
exec chpst -ugitdaemon \
"$(git --exec-path)"/git-daemon --verbose --reuseaddr \
--base-path=/var/cache /var/cache/git
```

```
git-daemon --verbose --reuseaddr \
--base-path=/var/cache /var/cache/git

改为

git-daemon --verbose --reuseaddr \
--enable=receive-pack --export-all --base-path=/opt/git
```

**提示**

```
   * 我加上了一个--export-all 使用该选项后，在git仓库中就不必创建git-daemon-export-ok文件。

   其他选项--enable=upload-pack --enable=upload-archive --enable=receive-pack
```

/etc/services 文件中加入

```
# Local services
git             9418/tcp                        # Git Version Control System
```

确认已经加入

```
$ grep 9418 /etc/services
```

启动git-daemon

```
$ sudo sv stop git-daemon

or

$ sudo runsv git-daemon
runsv git-daemon: fatal: unable to change to directory: file does not exist
```

扫描git端口，确认git-daemon已经启动

```
$ nmap localhost

Starting Nmap 5.00 ( http://nmap.org ) at 2012-01-31 10:45 CST
Warning: Hostname localhost resolves to 2 IPs. Using 127.0.0.1.
Interesting ports on localhost (127.0.0.1):
Not shown: 989 closed ports
PORT     STATE SERVICE
21/tcp   open  ftp
```

```
22/tcp   open  ssh
53/tcp   open  domain
80/tcp   open  http
111/tcp  open  rpcbind
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds
1723/tcp open  pptp
3128/tcp open  squid-http
3306/tcp open  mysql
9418/tcp open  git
```

## 13.5. Testing

```
$ sudo mkdir -p /opt/git/example.git
$ cd /opt/git/example.git
$ git init
$ sudo vim example.git/.git/config
[receive]
denyCurrentBranch = ignore

$ sudo chown gitdaemon -R /opt/git/*
$ touch git-daemon-export-ok
```

.git/config 文件应该是下面这样

```
$ cat example.git/.git/config
[core]
repositoryformatversion = 0
filemode = true
bare = false
logallrefupdates = true

[receive]
denyCurrentBranch = ignore
```

git-clone git://localhost/example.git

```
neo@deployment:/tmp$ git clone git://localhost/example.git
example.git
Cloning into example.git...
warning: You appear to have cloned an empty repository.
neo@deployment:/tmp$ cd example.git/
neo@deployment:/tmp/example.git$ echo helloworld > hello.txt
neo@deployment:/tmp/example.git$ git add hello.txt
neo@deployment:/tmp/example.git$ git commit -m 'Initial commit'
[master (root-commit) 65a0f83] Initial commit
1 files changed, 1 insertions(+), 0 deletions(-)
create mode 100644 hello.txt
```

我们添加了一些文件 push 到服务器

```
$ git push origin master
Counting objects: 3, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 214 bytes, done.
Total 3 (delta 0), reused 0 (delta 0)
To git://localhost/example.git
* [new branch]      master -> master
```

然后再git clone，可以看到文件数目

```
$ git-clone git://localhost/example.git
Cloning into example...
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0)
Receiving objects: 100% (3/3), done.
```

# 14. git-svn - Bidirectional operation between a single Subversion branch and git

```
sudo apt-get install git-svn
```

clone

```
git-svn clone -s svn://netkiller.8800.org/neo
cd neo
git gc

git commit -a
git-svn dcommit
```

从 svn 仓库更新

```
git-svn rebase
```

git-svn init svn://netkiller.8800.org/neo/public_html

# 15. Web Tools

## 15.1. viewgit

http://viewgit.sourceforge.net/

# 16. gitolite - SSH-based gatekeeper for git repositories

```
$ apt-cache search gitolite
gitolite - SSH-based gatekeeper for git repositories
```

```
sudo apt-get install gitolite
```

No adminkey given - not setting up gitolite.

## 16.1. gitolite-admin

```
git@192.168.2.1:gitolite-admin.git
```

### gitolite.conf

gitolite-admin/conf/gitolite.conf

#### staff

```
@admin          = neo
@developer      = bottle nada dick blank phabricator
@designer       = blank
@deployer       = phoenix
@tester         = jimmy
```

#### repo

```
repo gitolite-admin
```

```
    RW+     = @admin
    R       = @deployer

repo mydomain.com/www.mydomain.com
    RW+     = @admin
    RW          = @developer @designer
    R           = @deployer

repo mydomain.com/images.mydomain.com
    RW+     = @admin
    RW          = @developer @designer
    R           = @deployer

repo mydomain.com/passport.mydomain.com
    RW+     = @admin
    RW          = @developer
    R           = @deployer @designer

repo    example.com/www.example.com
  RW+       = @all

repo    @all
  RW        = @developer @designer
  R         = @agentbot @deployment @test
```

# 17. FAQ

## 17.1. 导出最后一次修改过的文件

有时我们希望把刚刚修改的文件复制出来，同时维持原有的目录结构，这样可能交给运维直接覆盖服务器上的代码。我们可以使用下面的命令完成这样的操作，而不用一个一个文件的复制。

```
git archive -o update.zip HEAD $(git diff --name-only HEAD^)
```

## 17.2. 导出指定版本区间修改过的文件

首先使用git log查看日志，找到指定的 commit ID号。

```
$ git log
commit ee808bb4b3ed6b7c0e7b24eeec39d299b6054dd0
Author: 168 <lineagelx@126.com>
Date:   Thu Oct 22 13:12:11 2015 +0800

统计代码

commit 3e68ddef50eec39acea1b0e20fe68ff2217cf62b
Author: netkiller <netkiller@msn.com>
Date:   Fri Oct 16 14:39:10 2015 +0800

页面修改

commit b111c253321fb4b9c5858302a0707ba0adc3cd07
Author: netkiller <netkiller@msn.com>
Date:   Wed Oct 14 17:51:55 2015 +0800

数据库连接

commit 4a21667a576b2f18a7db8bdcddbd3ba305554ccb
```

```
Author: netkiller <netkiller@msn.com>
Date:   Wed Oct 14 17:27:15 2015 +0800

init repo
```

导入 b111c253321fb4b9c5858302a0707ba0adc3cd07 至 ee808bb4b3ed6b7c0e7b24eeec39d299b6054dd0 间修改过的文件。

```
$ git archive -o update2.zip HEAD $(git diff --name-only
b111c253321fb4b9c5858302a0707ba0adc3cd07)
```

## 17.3. 撤销当前修改，恢复到远程最后一次提交

```
neo@MacBook-Pro-M2 ~/w/ensd-fscs (master|MERGING)> git reset --
hard origin/master
HEAD is now at d8952521 Merge branch 'revert-caebf6ee' into
'master'
```

## 17.4. 回撤提交

首先 reset 到指定的版本，根据实际情况选择 --mixed 还是 --hard

```
git reset --mixed 096392721f105686fc3cdafcb4159439a66b0e5b --
or
git reset --hard 33ba6503b4fa8eed35182262770e4eab646396cd --
```

```
```

```
git push origin --force --all
or
git push --force --progress "origin" master:master
```

## 17.5. 撤回单个文件提交

例如撤回 project/src/main/java/cn/netkiller/controller/DemoSceneController.java 到上一个版本

```
➜  api.netkiller.cn git:(testing) git log
project/src/main/java/cn/netkiller/controller/DemoSceneControll
er.java

commit b4609646ee60927fe4c1c563d07e78f63ab106ea (HEAD ->
testing, origin/testing)
Author: Neo Chen <netkiller@msn.com>
Date:   Wed Nov 17 18:49:27 2021 +0800

手工合并，临时提交

commit bc96eb68ad73d5248c8135609191c51e258edf10
Author: Tom <tom@qq.com>
Date:   Thu Oct 21 16:29:20 2021 +0800

获取激活场景

commit d564ea25bd556324f1f576357563a8ee77b3bdd9
Author: Tom <tom@qq.com>
Date:   Thu Oct 21 15:15:26 2021 +0800

获取激活场景

commit d5a40165ad24a3a021fe58c6d78e0b7d97ab3cc5
Author: Tom <tom@qq.com>
Date:   Thu Oct 21 14:43:16 2021 +0800

新增场景角色增加
```

```
commit aa98662cb9e781e328ee3d5cec23af29c81050d9
Author: Tom <tom@qq.com>
Date:   Thu Oct 21 09:55:29 2021 +0800

新增场景角色增加

commit 140d22a8d4ea7fcc775d4372e8beb6d854831512
Author: Jerry <jerry@qq.com>
Date:   Sat Oct 16 15:27:30 2021 +0800

场景接口修改

commit 2ddbb1ff933de663305db2396d99030c938c267a
Author: Tom <tom@qq.com>
Date:   Fri Oct 15 10:55:30 2021 +0800
```

只显示最后五条记录

```
➜  api.netkiller.cn git:(testing) git log -5
project/src/main/java/cn/netkiller/controller/DemoSceneControll
er.java
```

```
➜  api.netkiller.cn git:(testing) git reset
bc96eb68ad73d5248c8135609191c51e258edf10
project/src/main/java/cn/netkiller/controller/DemoSceneControll
er.java
Unstaged changes after reset:
M
project/src/main/java/cn/netkiller/controller/DemoSceneControll
er.java
```

```
➜  api.netkiller.cn git:(testing) ✗ git status
```

```
On branch testing
Your branch is up to date with 'origin/testing'.

Changes to be committed:
(use "git restore --staged <file>..." to unstage)
modified:
project/src/main/java/cn/netkiller/controller/DemoSceneControll
er.java

Changes not staged for commit:
(use "git add <file>..." to update what will be committed)
(use "git restore <file>..." to discard changes in working
directory)
modified:
project/src/main/java/cn/netkiller/controller/DemoSceneControll
er.java

➜  api.netkiller.cn git:(testing) ✗ git add
project/src/main/java/cn/netkiller/controller/DemoSceneControll
er.java
➜  api.netkiller.cn git:(testing) ✗ git commit -m '恢复到上一个版
本'
[testing 9959acd4] 恢复到上一个版本
1 file changed, 6 insertions(+), 8 deletions(-)
```

## 17.6. 合并分支中的单个

git merge 会合并两个分支中的所有文件，如果我们只想合并单个文件，可以这样做。

```
git checkout feature
git checkout --patch master file.txt
```

例如热修复生产代码，将hotfix分支的文件file.txt 合到master分支上；

```
git checkout master
git checkout hotfix file.txt
git commit -a -m '修复生产BUG'
```

这种方式相当于吧 file.txt 文件从 hotfix 分支复制到 master 分支，适合处理二进制文件。

## 17.7. 每个项目一个证书

方法一

```
[root@localhost ~]# cat .ssh/config
host git.netkiller.cn
user git
hostname git.netkiller.cn
port 22
identityfile ~/.ssh/netkiller

host github.com
HostName github.com
IdentityFile ~/.ssh/id_rsa_github
User git
```

方法二

```
$ ssh-agent sh -c 'ssh-add ~/.ssh/id_rsa; git fetch user@host'
```

## 17.8. fatal: Not possible to fast-forward, aborting.

```
$ git pull
fatal: Not possible to fast-forward, aborting.
$ git rebase
$ git push
```

## 17.9. receive.denyCurrentBranch

git push 操作提示 receive.denyCurrentBranch

```
remote: error: refusing to update checked out branch:
refs/heads/main
remote: error: By default, updating the current branch in a
non-bare repository
remote: is denied, because it will make the index and work tree
inconsistent
remote: with what you pushed, and will require 'git reset --
hard' to match
remote: the work tree to HEAD.
remote:
remote: You can set the 'receive.denyCurrentBranch'
configuration variable
remote: to 'ignore' or 'warn' in the remote repository to allow
pushing into
remote: its current branch; however, this is not recommended
unless you
remote: arranged to update its work tree to match what you
pushed in some
remote: other way.
remote:
remote: To squelch this message and still keep the default
behaviour, set
remote: 'receive.denyCurrentBranch' configuration variable to
'refuse'.
```

起因，疫情期间远程办公，将办公室内的 gitlab 所有仓库都导出来，放在一个临时服务器上，服务器创建了一个临时账号 git，仓库放在 /home/git 目录下。

开发人员使用类似地址 git@git.netkiller.cn:netkiller.cn/api.netkiller.cn 克隆代码，但是 git push 的时候出现上面错误

解决方案

进入服务器 su - git，然后在项目目录下面运行

```
git init --shared --bare
git config receive.denyCurrentBranch ignore
```

再次尝试 git push 问题解决。

## 17.10. 更新所有项目以及分支

```
for project in $(ls -1 | grep com); do

cd $project && \
for branch in $(git branch -r | grep -v HEAD) ; do
# git checkout -b ${branch#*/} $branch;
git checkout ${branch#*/};
git pull;
done;
cd ..;
done;

rsync -auzv --delete * git@netkiller.cn:/opt/backup/code/
```

## 17.11. 找回丢失的分支

```
git reflog
git checkout -b your_branch commit-id
```

# 第 21 章 Subversion

# 1. Invoking the Server

配置开发环境版本控制Subversion

Squid + Subversion 请参考Squid一节

## 1.1. Installing

### Ubuntu

过程 21.1. subversion

1. installation

   **$ sudo apt-get install subversion**

```
$ sudo apt-get install subversion
```

2. create svn group and svnroot user

```
$ sudo groupadd svn
$ sudo adduser svnroot --ingroup svn
```

3. create repository

```
$ svnadmin create /home/svnroot/test
```

4. testing

```
svnroot@netkiller:~$ svnserve -d --foreground -r
/home/svnroot/
```

check out

neo@netkiller:/tmp$ svn list svn://localhost/test

you may see some file and directory

```
neo@netkiller:/tmp$ ls test/.svn/
entries  format  prop-base  props  text-base  tmp
```

5. configure

$ vim repositories/conf/svnserve.conf

```
[general]
anon-access = read
auth-access = write
password-db = passwd
# authz-db = authz
# realm = My First Repository
```

$ vim repositories/conf/passwd

```
[users]
# harry = harryssecret
# sally = sallyssecret
neo = chen
```

如果不允许匿名用户checkout代码，配置文件这样写anon-access = none

```
[general]
anon-access = none
auth-access = write
```

6. firewall

```
$ sudo ufw allow svn
```

## CentOS 5

```
[root@development ~]# yum -y install subversion
```

**classic Unix-like xinetd daemon**

```
[root@development ~]# vim /etc/xinetd.d/subversion
service subversion
{
    disable = no
    port = 3690
    socket_type = stream
    protocol = tcp
    wait = no
    user = svnroot
    server = /usr/bin/svnserve
    server_args = -i -r /home/svnroot
}
```

firewall

```
iptables -A INPUT -p tcp -m tcp --sport 3690 -j ACCEPT
iptables -A OUTPUT -p tcp -m tcp --dport 3690 -j ACCEPT
```

**WebDav**

install webdav module

```
[root@development ~]# yum install mod_dav_svn
```

create directory

```
mkdir /var/www/repository
svnadmin create /var/www/repository
```

subversion.conf

```
[root@development ~]# vim /etc/httpd/conf.d/subversion.conf
LoadModule dav_module modules/mod_dav.so
LoadModule dav_svn_module modules/mod_dav_svn.so
LoadModule authz_svn_module modules/mod_authz_svn.so
```

vhost.conf

```
<Location />
```

```
DAV svn
SVNPath /var/www/repository
AuthType Basic
AuthName "Subversion Repository"
AuthUserFile /etc/subversion/svn-auth-file
Require valid-user
</Location>
```

auth file

```
[root@development ~]# htpasswd -c /etc/subversion/svn-auth-file
my_user_name
```

项目目录结构

–trunk #存放主线

–branches #存放分支，可修改

–tags #存放标记，不可修改

## CentOS 6

CentOS 6 默认没有安装xinetd

```
# yum install xinetd
# yum install subversion

# mkdir -p /opt/svnroot
```

xinetd 配置

```
# vim /etc/xinetd.d/svn
service svn
{
    disable = no
    port = 3690
    socket_type = stream
    protocol = tcp
    wait = no
    user = svnroot
    server = /usr/bin/svnserve
    server_args = -i -r /opt/svnroot
}

# /etc/init.d/xinetd restart
Stopping xinetd:
[FAILED]
Starting xinetd:                                           [
OK  ]

# tail /var/log/messages | grep xinetd
May  5 18:57:20 SZVM42-C1-02 yum: Installed: 2:xinetd-2.3.14-
16.el5.i386
May  5 18:59:22 SZVM42-C1-02 xinetd[4558]: Unknown user:
svnroot [file=/etc/xinetd.d/svn] [line=8]
May  5 18:59:22 SZVM42-C1-02 xinetd[4558]: Error parsing
attribute user - DISABLING SERVICE

[file=/etc/xinetd.d/svn] [line=8]
May  5 18:59:22 SZVM42-C1-02 xinetd[4558]: xinetd Version
2.3.14 started with libwrap loadavg labeled-networking

options compiled in.
May  5 18:59:22 SZVM42-C1-02 xinetd[4558]: Started working: 0
available services
```

service 名字必须与 /etc/services中定义的名字相同，否则将不能启动，同时在/var/log/message中会提示如下

```
May  4 14:33:08 www xinetd[5656]: service/protocol combination
not in /etc/services: subversion/tcp
May  4 14:33:08 www xinetd[5656]: xinetd Version 2.3.14 started
```

```
with libwrap loadavg labeled-networking options compiled in.
May  4 14:33:08 www xinetd[5656]: Started working: 0 available
services
May  4 14:33:33 www pulseaudio[21913]: sink-input.c: Failed to
create sink input: too many inputs per sink.
May  4 14:33:33 www pulseaudio[21913]: sink-input.c: Failed to
create sink input: too many inputs per sink.
May  4 14:33:33 www pulseaudio[21913]: sink-input.c: Failed to
create sink input: too many inputs per sink.
May  4 14:33:33 www pulseaudio[21913]: sink-input.c: Failed to
create sink input: too many inputs per sink.
May  4 14:33:33 www pulseaudio[21913]: sink-input.c: Failed to
create sink input: too many inputs per sink.
May  4 14:33:33 www pulseaudio[21913]: sink-input.c: Failed to
create sink input: too many inputs per sink.
May  4 14:33:33 www pulseaudio[21913]: sink-input.c: Failed to
create sink input: too many inputs per sink.
May  4 14:33:33 www pulseaudio[21913]: sink-input.c: Failed to
create sink input: too many inputs per sink.
May  4 14:33:41 www xinetd[5656]: Exiting...
May  4 14:33:41 www xinetd[5676]: xinetd Version 2.3.14 started
with libwrap loadavg labeled-networking options compiled in.
May  4 14:33:41 www xinetd[5676]: Started working: 1 available
service
```

## 1.2. standalone “daemon” process

svn daemon

```
$ svnserve --daemon --root /home/svnroot
```

### starting subversion for debian/ubuntu

/etc/init.d/subversion for debian/ubuntu

```
debian:/etc/init.d# cat subversion
#!/bin/sh
```

```
### BEGIN INIT INFO
# Provides:          subversion
# Required-Start:    $remote_fs $network
# Required-Stop:     $remote_fs $network
# Should-Start:      fam
# Should-Stop:       fam
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: Start the subversion subversion server.
### END INIT INFO

#########################
# Author: Neo <openunix@163.com>
#########################

PATH=/sbin:/bin:/usr/sbin:/usr/bin
DAEMON=/usr/bin/svnserve
NAME=subversion
DESC="subversion server"
PIDFILE=/var/run/$NAME.pid
SCRIPTNAME=/etc/init.d/$NAME
SVNROOT=/srv/svnroot
DAEMON_OPTS="-d -T -r $SVNROOT --pid-file $PIDFILE"

test -x $DAEMON || exit 0

set -e

. /lib/lsb/init-functions

case "$1" in
    start)
        log_daemon_msg "Starting $DESC" $NAME
        echo
        $DAEMON $DAEMON_OPTS
        echo `pgrep -o $NAME` > $PIDFILE > /dev/null 2>
/dev/null
        ;;
    stop)
        log_daemon_msg "Stopping $DESC" $NAME
        echo
        killall `basename $DAEMON` > /dev/null 2> /dev/null
        rm -rf $PIDFILE
        ;;
    restart)
```

```
        $0 stop
        $0 start
        ;;
    status)
        ps ax | grep $NAME
        ;;
    *)
        echo "Usage: $SCRIPTNAME {start|stop|restart|status}"
>&2
        exit 1
        ;;
esac

exit 0
```

## starting subversion daemon script for CentOS/Radhat

```
#!/bin/bash
#
# /etc/rc.d/init.d/subversion
#
# Starts the Subversion Daemon
#
# chkconfig: 345 90 10
#
# description: Subversion Daemon

# processname: svnserve

source /etc/rc.d/init.d/functions

[ -x /usr/bin/svnserve ] || exit 1

### Default variables
SYSCONFIG="/etc/sysconfig/subversion"

### Read configuration
[ -r "$SYSCONFIG" ] && source "$SYSCONFIG"
```

```
RETVAL=0
USER="svnroot"
prog="svnserve"
desc="Subversion Daemon"

start() {
   echo -n $"Starting $desc ($prog): "
   daemon --user $USER $prog -d $OPTIONS
   RETVAL=$?
   [ $RETVAL -eq 0 ] && touch /var/lock/subsys/$prog
   echo
}

stop() {
   echo -n $"Shutting down $desc ($prog): "
   killproc $prog
   RETVAL=$?
   [ $RETVAL -eq 0 ] && success || failure
   echo
   [ $RETVAL -eq 0 ] && rm -f /var/lock/subsys/$prog
   return $RETVAL
}

case "$1" in
  start)
   start
   ;;
  stop)
   stop
   ;;
  restart)
   stop
   start
   RETVAL=$?
   ;;
  condrestart)
        [ -e /var/lock/subsys/$prog ] && restart
   RETVAL=$?
   ;;
  *)
   echo $"Usage: $0 {start|stop|restart|condrestart}"
   RETVAL=1
esac

exit $RETVAL
```

```
```

/etc/sysconfig/subversion

```
# Configuration file for the Subversion service

#
# To pass additional options (for instace, -r root of directory
to server) to
# the svnserve binary at startup, set OPTIONS here.
#
#OPTIONS=
OPTIONS="--threads --root /srv/svnroot"
```

## 1.3. classic Unix-like inetd daemon

/etc/inetd.conf

```
svn stream tcp nowait svn /usr/bin/svnserve svnserve -i -r
/home/svnroot/repositories
```

xinetd.d

/etc/xinetd.d/subversion

```
$ sudo apt-get install xinetd
$ sudo vim /etc/xinetd.d/subversion

service subversion
{
    disable = no
    port = 3690
    socket_type = stream
    protocol = tcp
    wait = no
    user = svnroot
```

```
    server = /usr/bin/svnserve
    server_args = -i -r /home/svnroot
}
```

restart xinetd

```
$ sudo /etc/init.d/xinetd restart
```

## 1.4. hooks

```
$ sudo apt-get install subversion-tools
```

### post-commit

install SVN::Notify

```
perl -MCPAN -e 'install SVN::Notify'
```

```
$ sudo cp post-commit.tmpl post-commit
$ sudo chown svnroot:svn post-commit
$ sudo vim post-commit

REPOS="$1"
REV="$2"

#/usr/share/subversion/hook-scripts/commit-email.pl "$REPOS"
"$REV" openunix@163.com
/usr/share/subversion/hook-scripts/commit-email.pl "$1" "$2" --
from neo@netkiller.8800.org -h localhost  -s "[SVN]" --diff y
openunix@163.com openx@163.com
```

另一种方法

```
#!/bin/sh

REPOS="$1"
REV="$2"

/usr/local/bin/svnnotify                        \
    --repos-path    "$REPOS"                    \
    --revision      "$REV"                      \
    --subject-cx                                \
    --with-diff                                 \
    --handler       HTML::ColorDiff             \
    --to            <your e-mail address>       \
    --from          <from e-mail address>
```

```
/usr/bin/svnnotify --repos-path "$REPOS" --revision "$REV" \
--from neo@netkiller.8800.org --to openunix@163.com --smtp
localhost \
--handler "HTML::ColorDiff"  --with-diff --charset zh_CN:GB2312
-g zh_CN --svnlook /usr/bin/svnlook --subject-prefix '[SVN]'
```

如果你没有安装邮件服务器，你可以使用服务商的SMTP如163.com

```
/usr/bin/svnnotify --repos-path "$REPOS" --revision "$REV" \
--from openx@163.com --to openunix@163.com --smtp smtp.163.com
--smtp-user openunix --smtp-pass ****** \
--handler "HTML::ColorDiff"  --with-diff --charset UTF-8 --
language zh_CN --svnlook /usr/bin/svnlook --subject-prefix
'[SVN]'
```

Charset

```
REPOS="$1"
REV="$2"

svnnotify --repos-path "$REPOS" --revision "$REV" \
        --subject-cx \
        --from neo.chen@example.com \
        --to group@example.com,manager@example.com \
        --with-diff \
        --svnlook /usr/bin/svnlook \
        --subject-prefix '[SVN]' \
        --charset UTF-8  --language zh_CN
```

## 1.5. WebDav

Apache SVN

**$ sudo apt-get install libapache2-svn**

```
netkiller@neo:/etc/apache2$ sudo apt-get install libapache2-svn
```

vhost

```
<VirtualHost *>
        ServerName svn.netkiller.8800.org
        DocumentRoot /var/svn

      <Location />
                DAV svn
                SVNPath /var/svn
                AuthType Basic
                AuthName "Subversion Repository"
                AuthUserFile /etc/apache2/svn.passwd
                <LimitExcept GET PROPFIND OPTIONS REPORT>
                        Require valid-user
                </LimitExcept>
        </Location>
</VirtualHost>
```

```

```

建立密码文件

建立第一个用户需要加-c参数

```
netkiller@neo:/etc/apache2$ sudo htpasswd2 -c
/etc/apache2/svn.passwd svn
New password:
Re-type new password:
Adding password for user svn
```

输入两次密码

建立其他用户

```
sudo htpasswd2 /etc/apache2/svn.passwd otheruser
```

**davfs2 - mount a WebDAV resource as a regular file system**

install

```
$ sudo apt-get install davfs2
```

mount a webdav to directory

```
$ sudo mount.davfs https://opensvn.csie.org/netkiller
/mnt/davfs/
Please enter the username to authenticate with server
https://opensvn.csie.org/netkiller or hit enter for none.
Username: svn
Please enter the password to authenticate user svn with server
https://opensvn.csie.org/netkiller or hit enter for none.
```

```
Password:
mount.davfs: the server certificate is not trusted
  issuer:      CSIE.org, CSIE.org, Taipei, Taiwan, TW
  subject:     CSIE.org, CSIE.org, Taipei, TW
  identity:    *.csie.org
  fingerprint:
e6:05:eb:fb:69:5d:25:4e:11:3c:83:e8:7c:44:ee:bf:a9:85:a3:64
You only should accept this certificate, if you can
verify the fingerprint! The server might be faked
or there might be a man-in-the-middle-attack.
Accept certificate for this session? [y,N] y
```

test

```
$ ls davfs/
branches  lost+found  tags  trunk
```

# 2. repository 管理

## 2.1. create repository

```
$ su - svnroot
$ svnadmin create /home/svnroot/neo
```

## 2.2. user admin

```
#!/bin/bash
################################
# Author: Neo<openunix@163.com
# Home: http://netkiller.sf.net
################################
SVNROOT=/srv/svnroot/project
adduser(){
    echo $1  $2
    if [ -z $1 ]; then
        usage
    else
        local user=$1
    fi
    if [ -z $2 ]; then
        usage
    else
        local passwd=$2
    fi
    echo "$1 = $2" >> $SVNROOT/conf/passwd
}
deluser(){
    local user=$1
    if [ -z $user ]; then
        usage
    else
        ed -s $SVNROOT/conf/passwd <<EOF
/$user/
d
```

```
wq
EOF
    fi
}
list(){
    cat $SVNROOT/conf/passwd
}
usage(){
    echo $"Usage: $0 {list|add|del} username"
}
case "$1" in
    list)
        list
        ;;
    add)
        adduser $2 $3
        ;;
    del)
        deluser $2
        ;;
    restart)
        stop
        start
        ;;
    condrestart)
        condrestart
        ;;

    *)
        usage
        exit 1
esac
```

用法

```
./svnuser list
./svnuser add user passwd
./svnuser del user
```

## 2.3. authz

$ svnadmin create /home/svnroot/project

$ svnserve --daemon --root /home/svnroot/project

```
[groups]
member = neo
blog = neo,netkiller
wiki = bg7nyt,chen,jingfeng

[/]
* =

[/member]
@member = rw
* = r

[/app/blog]
@blog = rw
* =

[/app/wiki]
@blog = rw
* =

# [repository:/baz/fuz]
# @harry_and_sally = rw
# * = r
```

$ svnadmin create /home/svnroot/project1

$ svnadmin create /home/svnroot/project2

$ svnserve --daemon --root /home/svnroot

```
[groups]
member = neo
blog = neo,netkiller
```

```
wiki = bg7nyt,chen,jingfeng

[project1:/]
* =
[project2:/]
* = r

[project1:/member]
@member = rw
* = r

[project2:/app/blog]
@blog = rw
* =

[project2:/app/wiki]
@blog = rw
* = r
```

**例 21.1. authz**

```
[aliases]
# joe = /C=XZ/ST=Dessert/L=Snake City/O=Snake Oil,
Ltd./OU=Research Institute/CN=Joe Average

### This file is an example authorization file for svnserve.
### Its format is identical to that of mod_authz_svn
authorization
### files.
### As shown below each section defines authorizations for the
path and
### (optional) repository specified by the section name.
### The authorizations follow. An authorization line can refer
to:
###  - a single user,
###  - a group of users defined in a special [groups] section,
###  - an alias defined in a special [aliases] section,
###  - all authenticated users, using the '$authenticated'
token,
###  - only anonymous users, using the '$anonymous' token,
###  - anyone, using the '*' wildcard.
###
### A match can be inverted by prefixing the rule with '~'.
```

```
Rules can
### grant read ('r') access, read-write ('rw') access, or no
access
### ('').

[aliases]
# joe = /C=XZ/ST=Dessert/L=Snake City/O=Snake Oil,
Ltd./OU=Research Institute/CN=Joe Average

[groups]

manager = neo
developer = jam,john,zen
tester = eva
designer = allan
deployer = ken

[/]
@manager = rw
@developer = r
@designer = r
@deployer = r
@tester = r
* =

##############################
# Trunk
# ##########################
[/www.mydomain.com/trunk]
@manager = rw
@designer = rw
@developer = rw
@deployer = r

[/images.mydomain.com/trunk]
@designer = rw

[/myid.mydomain.com/trunk]
@designer = r

[/info.mydomain.com/trunk]
@developer = r
@designer = r

##############################
```

```
#\Branches
##############################
[/admin.mydomain.com/branches]
@developer = rw
@designer = rw

[/myid.mydomain.com/branches]
@developer = rw
@designer = rw

[/info.mydomain.com/branches]
@developer = rw
@designer = rw

[/www.mydomain.com/branches]
@developer = rw
@designer = rw

[/images.mydomain.com/branches]
@developer = rw
@designer = rw

[/log.mydomain.com/branches]
@developer = rw

[/report.mydomain.com/branches]
@developer = rw

##############################
# TAGS
# ##############################
[/myid.mydomain.com/tags]
@deployer = rw
[/admin.mydomain.com/tags]
@deployer = rw
[/info.mydomain.com/tags]
@deployer = rw
```

## 2.4. dump

```
svnadmin dump /svnroot/project | gzip > svn.gz
```

# 3. 使用Subversion

## 3.1. Initialized empty subversion repository for project

```
svn co svn://127.0.0.1/project
cd project
mkdir trunk
mkdir tags
mkdir branches
svn ci -m "Initialized empty subversion repository in
your_project"
```

## 3.2. ignore

### svn propset svn:ignore [filename] [folder]

```
$ svn propset svn:ignore 'images' .
$ svn ci -m 'Ignoring a directory called "images".'
```

```
$ svn propset svn:ignore '*' images
$ svn ci -m 'Ignoring a directory called "images".'
```

```
$ svn export spool spool-tmp
$ svn rm spool
$ svn ci -m 'Removing inadvertently added directory "spool".'
$ mv spool-tmp spool
$ svn propset svn:ignore 'spool' .
$ svn ci -m 'Ignoring a directory called "spool".'
```

.ignore

```
svn propset svn:ignore -F .cvsignore .
svn propset -R svn:ignore -F .svnignore .
```

## 3.3. 关键字替换

```
Date
        这个关键字保存了文件最后一次在版本库修改的日期，看起来类似于
$Date: 2012-08-06 17:43:09 +0800 (Mon, 06 Aug 2012) $，它也可以用
LastChangedDate来指定。
Revision
        这个关键字描述了这个文件最后一次修改的修订版本，看起来像
$Revision: 446 $，也可以通过LastChangedRevision或者Rev引用。
Author
        这个关键字描述了最后一个修改这个文件的用户，看起来类似$Author:
netkiller $，也可以用LastChangedBy来指定。
HeadURL
        这个关键字描述了这个文件在版本库最新版本的完全URL，看起来类似
$HeadURL:
https://svn.code.sf.net/p/netkiller/svn/trunk/Docbook/Version/s
ection.version.svn.xml $，可以缩写为URL。
Id
        这个关键字是其他关键字一个压缩组合，它看起来就像$Id:
section.version.svn.xml 446 2012-08-06 09:43:09Z netkiller $，可
以解释为文件calc.c上一次修改的修订版本号是148，时间是2006年7月28日，作者
是sally。
```

```
$ cat weather.txt
$Id: section.version.svn.xml 446 2012-08-06 09:43:09Z netkiller
$

$ svn propset svn:keywords "Id" weather.txt
property 'svn:keywords' set on 'weather.txt'

$ cat weather.txt
$Id: section.version.svn.xml 446 2012-08-06 09:43:09Z netkiller
$
```

设置多个关键字

```
$ svn propset svn:keywords "Author HeadURL Id Revision" -R
*.php
```

```
svn –R propset svn:keywords –F .keywords *
```

## 3.4. lock 加锁/ unlock 解锁

$ svn lock -m "LockMessage" [–force] PATH

```
$ svn lock –m “lock test file“ test.php
$ svn unlock PATH
```

## 3.5. import

```
svn import [PATH] URL
svn export URL [PATH]
```

## 3.6. export 指定版本

```
svn log file
svn export –r rxxxxx file
or
svn export –r rxxxxx file newfile
svn ci –m "restore rxxxxxx"
```

## 3.7. 修订版本关键字

```
HEAD
```

```
    版本库中最新的（或者是“最年轻的”）版本。
BASE

    工作拷贝中一个条目的修订版本号，如果这个版本在本地修改了，则“BASE版本”就是这个条目在本地未修改的版本。
COMMITTED

    项目最近修改的修订版本，与BASE相同或更早。
PREV

    一个项目最后修改版本之前的那个版本，技术上可以认为是COMMITTED -1。
```

```
$ svn cat -r PREV filename > filename
$ svn diff -r PREV filename
```

## 3.8. 恢复旧版本

svn没有恢复旧版本的直接功能，不过可以使用svn merge命令恢复。比如说当前HEAD为2，而我要恢复成1版本，怎么做?

用svn merge:

```
svn update
svn merge --revision 2:1 svn://localhost/lynn
svn commit -m "restore to revision 1"
```

svn merge --r HEAD:1 svn://localhost/lynn

# 4. branch

## 4.1. create

create a new branch using copy

```
svn cp http://www.domain.com/truck/project
http://www.domain.com/branches/project_branch_1
```

## 4.2. remove

remove

```
svn rm http://www.domain.com/branches/project_branch_1
```

## 4.3. switch

```
svn switch http://www.domain.com/branches/project_branch_2 .
```

## 4.4. merge

```
svn -r 148:149 merge svn://server/trunk branches/module
```

## 4.5. relocate

switch --relocate FROM TO [PATH...]

```
svn switch --relocate svn://192.168.3.9/neo
```

```
svn://192.168.3.5/neo .
```

# 5. FAQ

## 5.1. 递归添加文件

```
$ svn add `svn st | grep '?' | awk '{print $2}'`
```

## 5.2. 清除项目里的所有.svn目录

```
find . -type d -iname ".svn" -exec rm -rf {} \;
```

## 5.3. color diff

http://colordiff.sourceforge.net/

```
$ sudo apt-get install colordiff
```

add the following to your ~/.bashrc

```
alias svndiff='svn diff --diff-cmd=colordiff'
```

## 5.4. cvs2svn

http://cvs2svn.tigris.org/

```
[root@development ~]# cvs2svn --encoding=gb2312 --fallback-
encoding=utf_8 --existing-svnrepos --svnrepos /home/svnroot
/home/cvsroot
[root@development ~]# cvs2svn --encoding=gb2312 --fallback-
```

```
encoding=utf_8 --svnrepos /home/svnroot /home/cvsroot
```

## 5.5. Macromedia Dreamweaver MX 2004 + WebDAV +Subversion

首先进入站点管理



单击新建(New…)按钮选择站点(Site)



显示站点设置面版 Local Info 中设置



Remote Info 中设置



单击设置按钮（settings）



单击确定



单击Done完成

连接 WebDAV 服务器



单击



连接



## 5.6. 指定用户名与密码

```
svn co svn://www.example.com/repos  --username neo --password
chen;
```

# 第 22 章 cvs - Concurrent Versions System

# 1. installation

过程 22.1. install cvs

1. install

```
$ sudo apt-get install xinetd
$ sudo apt-get install cvs
```

   show the cvs version

```
$ cvs -v

Concurrent Versions System (CVS) 1.12.13 (client/server)
```

2. create cvs group and cvsroot user

```
$ sudo groupadd cvs
$ sudo adduser cvsroot --ingroup cvs
```

   change user become cvsroot

```
$ su - cvsroot
```

3. initialization 'CVSROOT'

```
$ cvs -d /home/cvsroot init
```

if you have successed, you can see CVSROOT directory in the '/home/cvsroot'

```
$ ls /home/cvsroot/
CVSROOT
```

4. authentication

   default SystemAuth=yes, you can use system user to login cvs.

   but usually, we don't used system user because it isn't security.

   SystemAuth = no

   edit '/home/cvsroot/CVSROOT/config' make sure SystemAuth = no

```
$ vim /home/cvsroot/CVSROOT/config
SystemAuth = no
```

   create passwd file

   the format is user:password:cvsroot

   you need to using htpasswd command, if you don't have, please install it as the following

```
$ sudo apt-get install apache2-utils
```

   or

```
$ perl -e 'print("userPassword:
".crypt("secret","salt")."\n");'
```

or

```
$ cat passwd
#!/usr/bin/perl
srand (time());
my $randletter = "(int (rand (26)) + (int (rand (1) + .5) %
2 ? 65 : 97))";
my $salt = sprintf ("%c%c", eval $randletter, eval
$randletter);
my $plaintext = shift; my $crypttext = crypt ($plaintext,
$salt);
print "${crypttext}\n";

$ ./passwd "mypasswd"
atfodI2Y/dcdc
```

let's using htpasswd to create a passwd

```
$ htpasswd -n neo
New password:
Re-type new password:
neo:yA50LI1BkXysY
```

copy 'neo:yA50LI1BkXysY' and add ':cvsroot' to the end

```
$ vim /home/cvsroot/CVSROOT/passwd
neo:yA50LI1BkXysY:cvsroot
nchen:GXaAkSKaQ/Hpk:cvsroot
```

5. Go into directory '/etc/xinetd.d/', and then create a cvspserver file as the following.

```
$ sudo vim /etc/xinetd.d/cvspserver

service cvspserver
{
```

```
   disable = no
   flags = REUSE
   socket_type = stream
   wait = no
   user = cvsroot
   server = /usr/bin/cvs
   server_args = -f --allow-root=/home/cvsroot pserver
   log_on_failure += USERID
}
```

6. check cvspserver in the '/etc/services'

```
$ grep cvspserver /etc/services
cvspserver      2401/tcp                        # CVS
client/server operations
cvspserver      2401/udp
```

7. restart xinetd

```
$ /etc/init.d/xinetd
Usage: /etc/init.d/xinetd {start|stop|reload|force-
reload|restart}
```

8. port

```
$ nmap localhost -p cvspserver

Starting Nmap 4.53 ( http://insecure.org ) at 2008-11-14
16:21 HKT
Interesting ports on localhost (127.0.0.1):
PORT     STATE SERVICE
2401/tcp open  cvspserver

Nmap done: 1 IP address (1 host up) scanned in 0.080
seconds
```

9. firewall

```
$ sudo ufw allow cvspserver
```

environment variable

CVSROOT=:pserver:username@ip:/home/cvsroot

```
vim .bashrc

export CVS_RSH=ssh
export CVSROOT=:pserver:neo@localhost:/home/cvsroot
```

test

```
$ cvs login
Logging in to :pserver:neo@localhost:2401/home/cvsroot
CVS password:
neo@netkiller:/tmp/test$ cvs co test
cvs checkout: Updating test
U test/.project
U test/NewFile.xml
U test/newfile.php
neo@netkiller:/tmp/test$
```

## 1.1. chroot

```
$ sudo apt-get install cvsd
```

environment variable

```
neo@netkiller:~/workspace/cvs$ export
```

```
CVSROOT=:pserver:neo@localhost:/home/cvsroot
```

ssh

```
export CVS_RSH=ssh
export CVSROOT=:ext:$USER@localhost:/home/cvsroot
```

# 2. cvs login | logout

```
neo@netkiller:~/workspace/cvs$ cvs login
Logging in to :pserver:neo@localhost:2401/home/cvsroot
CVS password:
```

logout

```
$ cvs logout
Logging out of :pserver:neo@localhost:2401/home/cvsroot
```

# 3. cvs import

cvs import -m "write some comments here" project_name vendor_tag release_tag

```
$ cvs import -m "write some comments here" project_name
vendor_tag release_tag
```

# 4. cvs checkout

```
$ cvs checkout project_name
cvs checkout: Updating project_name
```

checkout before

**cvs checkout -r release_1_0 project_name**

```
$ cvs checkout -r release_1_0 project_name
cvs checkout: Updating project_name
U project_name/file
cvs checkout: Updating project_name/dir1
U project_name/dir1/file1
cvs checkout: Updating project_name/dir2
U project_name/dir2/file1
U project_name/dir2/file2
```

# 5. cvs update

about update

```
$ cvs update
$ cvs update -r HEAD
$ cvs update -r 1.5
$ cvs update -D now
$ cvs update -D now file
```

# 6. cvs add

```
$ cd project_name/
$ touch new_file
$ cvs add new_file
cvs add: scheduling file `new_file' for addition
cvs add: use `cvs commit' to add this file permanently
```

if the file is binary

cvs add -kb new_file.gif

add a directory

```
$ mkdir dir1
$ mkdir dir2
$ touch dir1/file1
$ touch dir2/file1
$ touch dir2/file2
$ cvs add dir1
? dir1/file1
Directory /home/cvsroot/project_name/dir1 added to the
repository
$ cvs add dir2
? dir2/file1
? dir2/file2
Directory /home/cvsroot/project_name/dir2 added to the
repository
```

add mulit files

```
$ cvs add dir1/file1
$ cvs add dir2/file?
```

# 7. cvs status

```
$ cvs status dir1/file1
cvs status: use `cvs add' to create an entry for `dir1/file1'
===================================================================
====
File: file1             Status: Unknown

   Working revision:  No entry for file1
   Repository revision: No revision control file
```

# 8. cvs commit

```
$ cvs commit -m "add a new file"
cvs commit: Examining .
/home/cvsroot/project_name/new_file,v  <--  new_file
initial revision: 1.1
```

commit multi files

```
$ cvs commit -m "add a new file" dir1/* dir2/*
/home/cvsroot/project_name/dir1/file1,v  <--  dir1/file1
initial revision: 1.1
/home/cvsroot/project_name/dir2/file1,v  <--  dir2/file1
initial revision: 1.1
/home/cvsroot/project_name/dir2/file2,v  <--  dir2/file2
initial revision: 1.1
```

# 9. cvs remove

```
$ rm -rf new_file
$ cvs remove new_file
cvs remove: scheduling `new_file' for removal
cvs remove: use `cvs commit' to remove this file permanently
$ cvs commit -m "delete file" new_file
/home/cvsroot/project_name/new_file,v  <--  new_file
new revision: delete; previous revision: 1.1
```

# 10. cvs log

let me create a file, and then modify the file to make several version

```
$ touch file
$ echo helloworld > file
$ cvs add file
cvs add: scheduling file `file' for addition
cvs add: use `cvs commit' to add this file permanently
$ cvs commit -m 'add file to cvs' file
/home/cvsroot/project_name/file,v  <--  file
initial revision: 1.1
$ echo I am Neo > file
$ cvs commit -m 'add file to cvs' file
/home/cvsroot/project_name/file,v  <--  file
new revision: 1.2; previous revision: 1.1
$ echo my nickname is netkiller > file
$ cvs commit -m 'modified file' file
/home/cvsroot/project_name/file,v  <--  file
new revision: 1.3; previous revision: 1.2
$ echo I am 28 years old > file
$ cvs commit -m 'modified file' file
/home/cvsroot/project_name/file,v  <--  file
new revision: 1.4; previous revision: 1.3
```

show log message

```
$ cvs log file

RCS file: /home/cvsroot/project_name/file,v
Working file: file
head: 1.4
branch:
locks: strict
access list:
symbolic names:
```

```
keyword substitution: kv
total revisions: 4;   selected revisions: 4
description:
----------------------------
revision 1.4
date: 2008-11-24 15:42:49 +0800;  author: neo;  state: Exp;
lines: +1 -1;  commitid: V0iuptfP43iETPrt;
modified file
----------------------------
revision 1.3
date: 2008-11-24 15:42:20 +0800;  author: neo;  state: Exp;
lines: +1 -1;  commitid: YWfYHFSV10duTPrt;
modified file
----------------------------
revision 1.2
date: 2008-11-24 15:41:47 +0800;  author: neo;  state: Exp;
lines: +1 -1;  commitid: 4iRs5fm1g9diTPrt;
add file to cvs
----------------------------
revision 1.1
date: 2008-11-24 15:41:28 +0800;  author: neo;  state: Exp;
commitid: zCWkxnWxLZHbTPrt;
add file to cvs
=============================================================
==============
```

cvs log -r1.2 file

```
$ cvs log -r1.2 file

RCS file: /home/cvsroot/project_name/file,v
Working file: file
head: 2.1
branch:
locks: strict
access list:
symbolic names:
        release_1_0_patch: 1.4.0.2
        release_1_0: 1.4
keyword substitution: kv
total revisions: 5;     selected revisions: 1
```

```
description:
----------------------------
revision 1.2
date: 2008-11-24 15:41:47 +0800;  author: neo;  state: Exp;
lines: +1 -1;  commitid: 4iRs5fm1g9diTPrt;
add file to cvs
=============================================================
==============
```

# 11. cvs annotate

```
$ cvs annotate file

Annotations for file
***************
2.2          (nchen    26-Nov-08): I am Neo
2.2          (nchen    26-Nov-08): My nickname netkiller
2.3          (nchen    26-Nov-08): I'm from shenzhen
1.4          (neo      24-Nov-08): I am 28 years old
```

# 12. cvs diff

```
neo@netkiller:~/workspace/cvs/project_name$ cvs diff -r1.3 -
r1.4 file
Index: file
===================================================================
====
RCS file: /home/cvsroot/project_name/file,v
retrieving revision 1.3
retrieving revision 1.4
diff -r1.3 -r1.4
1c1
< my nickname is netkiller
---
> I am 28 years old
neo@netkiller:~/workspace/cvs/project_name$ cvs diff -r1.2 -
r1.4 file
Index: file
===================================================================
====
RCS file: /home/cvsroot/project_name/file,v
retrieving revision 1.2
retrieving revision 1.4
diff -r1.2 -r1.4
1c1
< I am Neo
---
> I am 28 years old
```

--side-by-side

```
neo@netkiller:/tmp/cvs/test/project_name$ cvs diff --side-by-
side -r1.2 -r1.4 file
Index: file
===================================================================
====
RCS file: /home/cvsroot/project_name/file,v
```

```
retrieving revision 1.2
retrieving revision 1.4
diff --side-by-side -r1.2 -r1.4
I am Neo                                                        |
I am 28 years old
```

# 13. rename file

mv file_name new_file_name && cvs remove file_name
cvs add new_file_name

```
neo@netkiller:/tmp/cvs/project_name$ mv file file.txt
neo@netkiller:/tmp/cvs/project_name$ cvs remove file
cvs remove: scheduling `file' for removal
cvs remove: use `cvs commit' to remove this file permanently
neo@netkiller:/tmp/cvs/project_name$ cvs add file.txt
cvs add: scheduling file `file.txt' for addition
cvs add: use `cvs commit' to add this file permanently
neo@netkiller:/tmp/cvs/project_name$ cvs commit -m 'rename file
to file.txt'
cvs commit: Examining .
cvs commit: Examining dir1
cvs commit: Examining dir2
/home/cvsroot/project_name/file,v  <--  file
new revision: delete; previous revision: 2.3
/home/cvsroot/project_name/file.txt,v  <--  file.txt
initial revision: 1.1
```

# 14. revision

```
neo@netkiller:~/workspace/cvs/project_name$ cvs update -r 1.2
file
U file
neo@netkiller:~/workspace/cvs/project_name$ cvs st file
===================================================================
====
File: file              Status: Up-to-date

   Working revision:  1.2
   Repository revision: 1.2
/home/cvsroot/project_name/file,v
   Commit Identifier: 4iRs5fm1g9diTPrt
   Sticky Tag:        1.2
   Sticky Date:       (none)
   Sticky Options:    (none)
```

last version

```
neo@netkiller:~/workspace/cvs/project_name$ cvs update -r HEAD
file
U file
neo@netkiller:~/workspace/cvs/project_name$ cvs st file
===================================================================
====
File: file              Status: Up-to-date

   Working revision:  1.4
   Repository revision: 1.4
/home/cvsroot/project_name/file,v
   Commit Identifier: V0iuptfP43iETPrt
   Sticky Tag:        HEAD (revision: 1.4)
   Sticky Date:       (none)
   Sticky Options:    (none)
```

# 15. cvs export

### cvs export -r release_1_0 project_name

```
$ cvs export -r release_1_0 project_name
cvs export: Updating project_name
U project_name/file
cvs export: Updating project_name/dir1
U project_name/dir1/file1
cvs export: Updating project_name/dir2
U project_name/dir2/file1
U project_name/dir2/file2
```

cvs export -D 20081126 project_name

```
$ cvs export -D 20081126 project_name
cvs export: Updating project_name
U project_name/file
cvs export: Updating project_name/dir1
U project_name/dir1/file1
cvs export: Updating project_name/dir2
U project_name/dir2/file1
U project_name/dir2/file2
```

### cvs export -D now -d nightly project_name

```
$ cvs export -D now -d nightly project_name
cvs export: Updating nightly
U nightly/file
cvs export: Updating nightly/dir1
U nightly/dir1/file1
cvs export: Updating nightly/dir2
U nightly/dir2/file1
U nightly/dir2/file2
neo@netkiller:/tmp/cvs$
```

# 16. cvs release

```
$ ls
project_name

$ cvs release -d project_name
You have [0] altered files in this repository.
Are you sure you want to release (and delete) directory
`project_name': y

$ ls
```

# 17. branch

## 17.1. milestone

set up a release number

```
$ cvs tag release_1_0
cvs tag: Tagging .
T file
cvs tag: Tagging dir1
T dir1/file1
cvs tag: Tagging dir2
T dir2/file1
T dir2/file2
```

beginning next one milestone

```
$ cvs commit -r 2

Log message unchanged or not specified
a)bort, c)ontinue, e)dit, !)reuse this message unchanged for
remaining dirs
Action: (continue) c

CVS: ----------------------------------------------------------
------------
CVS: Enter Log.  Lines beginning with `CVS:' are removed
automatically
CVS:
CVS: Committing in .
CVS:
CVS: Modified Files:
CVS:  Tag: 2
CVS:    file dir1/file1 dir2/file1 dir2/file2
CVS: ----------------------------------------------------------
------------
```

```
/home/cvsroot/project_name/file,v  <--  file
new revision: 2.1; previous revision: 1.4
/home/cvsroot/project_name/dir1/file1,v  <--  dir1/file1
new revision: 2.1; previous revision: 1.1
/home/cvsroot/project_name/dir2/file1,v  <--  dir2/file1
new revision: 2.1; previous revision: 1.1
/home/cvsroot/project_name/dir2/file2,v  <--  dir2/file2
new revision: 2.1; previous revision: 1.1
```

other user

```
$ cvs up
cvs update: Updating .
P file
cvs update: Updating dir1
U dir1/file1
cvs update: Updating dir2
U dir2/file1
U dir2/file2

$ cvs st file
===================================================================
====
File: file              Status: Up-to-date

   Working revision:    2.1
   Repository revision: 2.1
/home/cvsroot/project_name/file,v
   Commit Identifier:   SuZpTC1gCRrH2Qrt
   Sticky Tag:          (none)
   Sticky Date:         (none)
   Sticky Options:      (none)
```

## 17.2. patch branch

create a branch release_1_0_patch from release_1_0 by cvs admin

```
$ cvs rtag -b -r release_1_0 release_1_0_patch project_name
```

```
cvs rtag: Tagging project_name
cvs rtag: Tagging project_name/dir1
cvs rtag: Tagging project_name/dir2
```

checkout release_1_0_patch by other user

```
$ cvs checkout -r release_1_0_patch project_name
cvs checkout: Updating project_name
U project_name/file
cvs checkout: Updating project_name/dir1
U project_name/dir1/file1
cvs checkout: Updating project_name/dir2
U project_name/dir2/file1
U project_name/dir2/file2
```

show the status, and you can see 'Sticky Tag' is 'release_1_0_patch'

```
$ cvs st file
===================================================================
====
File: file              Status: Up-to-date

   Working revision:    1.4
   Repository revision: 1.4
/home/cvsroot/project_name/file,v
   Commit Identifier:   V0iuptfP43iETPrt
   Sticky Tag:          release_1_0_patch (branch: 1.4.2)
   Sticky Date:         (none)
   Sticky Options:      (none)
```

# 18. keywords

$Author: netkiller $
$Date: 2012-02-03 17:18:44 +0800 (Fri, 03 Feb 2012) $
$Name$
$Id: section.version.cvs.xml 340 2012-02-03 09:18:44Z netkiller $
$Header$
$Log$
$Revision: 340 $

add above keywords into a file, and then commit it.

```
$ cat file.txt
$Author: netkiller $
$Date: 2012-02-03 17:18:44 +0800 (Fri, 03 Feb 2012) $
$Name:  $
$Id: section.version.cvs.xml 340 2012-02-03 09:18:44Z netkiller
$
$Header: /home/cvsroot/project_name/file.txt,v 1.2 2008-11-27
01:33:29 nchen Exp $
$Log: file.txt,v $
Revision 1.2  2008-11-27 01:33:29  nchen
added some of keywords

$Revision: 340 $
```

# 第 23 章 Miscellaneous

建模工具

- FreeMind
- ArgoUML,StarUML

常用的项目管理工具

- TRAC
- Subversion
- Bugzilla
- TWiki

# 1. 代码托管

## 1.1. sourceforge.net

### http://netkiller.users.sourceforge.net/ 页面

使用 sftp命令连接netkiller@frs.sourceforge.net，然后切换目录cd userweb/htdocs/，上传页面文件 put index.html，sourceforge.net 支持php

```
$ sftp netkiller@frs.sourceforge.net
netkiller@frs.sourceforge.net's password:
Connected to frs.sourceforge.net.
sftp> ls -l
lrwxrwxrwx    1 root     root           28 Apr 26  2012 userweb
sftp> cd userweb/htdocs/
sftp> put /tmp/index.html
Uploading /tmp/index.html to /home/user-
web/n/ne/netkiller/htdocs/index.html
```

```
/tmp/index.html                       100%   10     0.0KB/s
00:00
sftp> put /tmp/index.php
Uploading /tmp/index.php to /home/user-
web/n/ne/netkiller/htdocs/index.php
/tmp/index.php                        100%   17     0.0KB/s
00:00
sftp> pwd
Remote working directory: /home/user-web/n/ne/netkiller/htdocs
sftp> ls
index.html  index.php
sftp> exit
```

将上面netkiller改为你注册的用户名即可

帮助:
https://sourceforge.net/apps/trac/sourceforge/wiki/Developer%20web

## 1.2. Google Code

## 1.3. GitHub

http://www.github.com/

首次操作

Global setup:

Download and install Git

```
git config --global user.name "Neo Chan"
git config --global user.email bg7nyt@gmail.com
```

Next steps:

```
mkdir neo
cd neo
git init
touch README
git add README
git commit -m 'first commit'
git remote add origin git@github.com:netkiller/neo.git
git push origin master
```

Existing Git Repo?

```
cd existing_git_repo
git remote add origin git@github.com:netkiller/neo.git
git push origin master
```

**clone 已经存在的仓库**

```
$ git clone
https://github.com/netkiller/netkiller.github.com.git

git config --global user.name "Your Name"
git config --global user.email you@example.com
git commit --amend --reset-author
```

# 2. GUI

## 2.1. TortoiseSVN

http://tortoisesvn.net/

## 2.2. TortoiseGit

http://code.google.com/p/tortoisegit/

# 3. Browser interface for CVS and SVN version control repositories

viewvc 10年前还有人再用，目前基本淘汰

```
# yum install viewvc
```

CGI 方式安装

```
# vim /etc/httpd/conf/httpd.conf
ScriptAlias /viewvc /usr/lib/python2.4/site-
packages/viewvc/bin/cgi/viewvc.cgi
Alias /viewvc-static/ "/usr/share/viewvc/templates/docroot/"
```

# 部分 IV. Configuration Management(配置管理)

## 运维自动化

表 1. 表格标题

| 名称 | 流行度 | 开发语言 | 工作模式 | 其他 |
| --- | --- | --- | --- | --- |
| Puppet | 主流 | Ruby | C/S | |
| Chef | 主流 | Ruby | C/S | |
| SaltStack | 主流(新星) | Python | C/S | |
| ansible | 一般 | Python | Server 结构无需 client | Redhat 开发 |

# 第 24 章 Ansible - SSH-based configuration management, deployment, and task execution system

http://ansible.github.com/

Ansible is a radically simple model-driven configuration management, deployment, and command execution framework.

# 1. install

```
yum install ansible
```

# 2. Getting Started

Your first commands

/etc/ansible/hosts

```
# vim /etc/ansible/hosts

192.168.2.10
192.168.2.11
192.168.2.12
192.168.2.13
192.168.2.14
192.168.2.15
```

创建SSH公钥与私钥

```
ssh-keygen
```

将公钥文件复制到目标服务器

```
ssh-copy-id root@192.168.2.10
ssh-copy-id root@192.168.2.11
ssh-copy-id root@192.168.2.12
ssh-copy-id root@192.168.2.13
ssh-copy-id root@192.168.2.14
ssh-copy-id root@192.168.2.15
```

连接与验证测试 ansible all -m ping

```
# ansible all -m ping
192.168.2.10 | success >> {
    "module": "ping",
```

```
    "ping": "pong"
}

192.168.2.13 | success >> {
    "module": "ping",
    "ping": "pong"
}

192.168.2.14 | success >> {
    "module": "ping",
    "ping": "pong"
}

192.168.2.11 | success >> {
    "module": "ping",
    "ping": "pong"
}

192.168.2.15 | success >> {
    "module": "ping",
    "ping": "pong"
}

192.168.2.12 | success >> {
    "module": "ping",
    "ping": "pong"
}
```

# 3. ansible - run a command somewhere else

```
Usage: ansible <host-pattern> [options]

Options:
  -a MODULE_ARGS, --args=MODULE_ARGS
                        module arguments
  -k, --ask-pass        ask for SSH password
  --ask-su-pass         ask for su password
  -K, --ask-sudo-pass   ask for sudo password
  --ask-vault-pass      ask for vault password
  -B SECONDS, --background=SECONDS
                        run asynchronously, failing after X
seconds
                        (default=N/A)
  -C, --check           don't make any changes; instead, try to
predict some
                        of the changes that may occur
  -c CONNECTION, --connection=CONNECTION
                        connection type to use (default=smart)
  -f FORKS, --forks=FORKS
                        specify number of parallel processes to
use
                        (default=5)
  -h, --help            show this help message and exit
  -i INVENTORY, --inventory-file=INVENTORY
                        specify inventory host file
                        (default=/etc/ansible/hosts)
  -l SUBSET, --limit=SUBSET
                        further limit selected hosts to an
additional pattern
  --list-hosts          outputs a list of matching hosts; does
not execute
                        anything else
  -m MODULE_NAME, --module-name=MODULE_NAME
                        module name to execute
(default=command)
  -M MODULE_PATH, --module-path=MODULE_PATH
                        specify path(s) to module library
                        (default=/usr/share/ansible)
```

```
  -o, --one-line        condense output
  -P POLL_INTERVAL, --poll=POLL_INTERVAL
                        set the poll interval if using -B
(default=15)
  --private-key=PRIVATE_KEY_FILE
                        use this file to authenticate the
connection
  -S, --su              run operations with su
  -R SU_USER, --su-user=SU_USER
                        run operations with su as this user
(default=root)
  -s, --sudo            run operations with sudo (nopasswd)
  -U SUDO_USER, --sudo-user=SUDO_USER
                        desired sudo user (default=root)
  -T TIMEOUT, --timeout=TIMEOUT
                        override the SSH timeout in seconds
(default=10)
  -t TREE, --tree=TREE  log output to this directory
  -u REMOTE_USER, --user=REMOTE_USER
                        connect as this user (default=root)
  --vault-password-file=VAULT_PASSWORD_FILE
                        vault password file
  -v, --verbose         verbose mode (-vvv for more, -vvvv to
enable
                        connection debugging)
  --version             show program's version number and exit
```

## 3.1. host-pattern

匹配所有主机

```
# ansible all -m ping -u root
```

匹配指定IP地址

```
# ansible 192.168.2.9 -m ping -u root

192.168.2.9 | success >> {
```

```
    "changed": false,
    "ping": "pong"
}
```

使用通配符

```
# ansible 192.168.2.? -m ping -u root

192.168.2.9 | success >> {
    "changed": false,
    "ping": "pong"
}

# ansible 192.168.2.* -m ping -u root

192.168.2.12 | success >> {
    "changed": false,
    "ping": "pong"
}

192.168.2.15 | success >> {
    "changed": false,
    "ping": "pong"
}

192.168.2.9 | success >> {
    "changed": false,
    "ping": "pong"
}

192.168.2.11 | success >> {
    "changed": false,
    "ping": "pong"
}
```

## 3.2. -a MODULE_ARGS, --args=MODULE_ARGS module arguments

```
ansible all -a 'echo hello'
```

## 3.3. -i INVENTORY, --inventory-file=INVENTORY specify inventory host file (default=/etc/ansible/hosts)

制定一个hosts文件

```
# echo '192.168.6.10' >> hosts

# cat hosts
192.168.6.10

# ansible all -a 'echo hello' -i hosts -u root
192.168.6.10 | success | rc=0 >>
hello
```

## 3.4. -m MODULE_NAME, --module-name=MODULE_NAME module name to execute (default=command)

```
$ ansible all -m ping -u neo
```

```
ansible all -m copy -a "src=hosts dest=~/hosts" -i hosts -u
vagrant
```

## 3.5. -s, --sudo run operations with sudo (nopasswd)

sudo 模式

```
# as bruce, sudoing to root
$ ansible all -m ping -u neo --sudo
# as bruce, sudoing to batman
$ ansible all -m ping -u www --sudo --sudo-user neo
```

## 3.6. -u REMOTE_USER, --user=REMOTE_USER connect as this user (default=root)

```
$ ansible all -m ping -u www
```

## 3.7. 使用实例

```
# ansible all -m yum -a "name=wget state=present"
```

# 4. ansible-doc - Show Ansible module documentation

```
# ansible-doc -l
acl                   Sets and retrieves file ACL information.
add_host              add a host (and alternatively a group) to
the ansible-playbo
airbrake_deployment   Notify airbrake about app deployments
alternatives          Manages alternative programs for common
commands
apache2_module        enables/disables a module of the Apache2
webserver
apt                   Manages apt-packages
apt_key               Add or remove an apt key
apt_repository        Add and remove APT repositories
apt_rpm               apt_rpm package manager
arista_interface      Manage physical Ethernet interfaces
arista_l2interface    Manage layer 2 interfaces
arista_lag            Manage port channel (lag) interfaces
arista_vlan           Manage VLAN resources
assemble              Assembles a configuration file from
fragments
assert                Fail with custom message
at                    Schedule the execution of a command or
script file via the a
authorized_key        Adds or removes an SSH authorized key
azure                 create or terminate a virtual machine in
azure
bigip_facts           Collect facts from F5 BIG-IP devices
bigip_monitor_http    Manages F5 BIG-IP LTM http monitors
bigip_monitor_tcp     Manages F5 BIG-IP LTM tcp monitors
bigip_node            Manages F5 BIG-IP LTM nodes
bigip_pool            Manages F5 BIG-IP LTM pools
bigip_pool_member     Manages F5 BIG-IP LTM pool members
boundary_meter        Manage boundary meters
bzr                   Deploy software (or files) from bzr
branches
campfire              Send a message to Campfire
capabilities          Manage Linux capabilities
cloudformation        create a AWS CloudFormation stack
```

```
command                Executes a command on a remote node
composer               Dependency Manager for PHP
copy                   Copies files to remote locations.
cpanm                  Manages Perl library dependencies.
cron                   Manage cron.d and crontab entries.
datadog_event          Posts events to DataDog  service
debconf                Configure a .deb package
debug                  Print statements during execution
digital_ocean          Create/delete a droplet/SSH_key in
DigitalOcean
digital_ocean_domain Create/delete a DNS record in DigitalOcean
digital_ocean_sshkey Create/delete an SSH key in DigitalOcean
django_manage          Manages a Django application.
dnsimple               Interface with dnsimple.com (a DNS hosting
service).
dnsmadeeasy            Interface with dnsmadeeasy.com (a DNS
hosting service).
docker                 manage docker containers
docker_image           manage docker images
easy_install           Installs Python libraries
ec2                    create, terminate, start or stop an
instance in ec2, return
ec2_ami                create or destroy an image in ec2, return
imageid
ec2_ami_search         Retrieve AWS AMI for a given operating
system.
ec2_asg                Create or delete AWS Autoscaling Groups
ec2_eip                associate an EC2 elastic IP with an
instance.
ec2_elb                De-registers or registers instances from
EC2 ELBs
ec2_elb_lb             Creates or destroys Amazon ELB. - Returns
information about
ec2_facts              Gathers facts about remote hosts within
ec2 (aws)
ec2_group              maintain an ec2 VPC security group.
ec2_key                maintain an ec2 key pair.
ec2_lc                 Create or delete AWS Autoscaling Launch
Configurations
ec2_metric_alarm       Create/update or delete AWS Cloudwatch
'metric alarms'
ec2_scaling_policy     Create or delete AWS scaling policies for
Autoscaling groups
ec2_snapshot           creates a snapshot from an existing volume
ec2_tag                create and remove tag(s) to ec2 resources.
```

```
ec2_vol                create and attach a volume, return volume
id and device map.
ec2_vpc                configure AWS virtual private clouds
ejabberd_user          Manages users for ejabberd servers
elasticache            Manage cache clusters in Amazon
Elasticache.
facter                 Runs the discovery program `facter' on the
remote system...
fail                   Fail with custom message
fetch                  Fetches a file from remote nodes
file                   Sets attributes of files
filesystem             Makes file system on block device
fireball               Enable fireball mode on remote node
firewalld              Manage arbitrary ports/services with
firewalld
flowdock               Send a message to a flowdock
gc_storage             This module manages objects/buckets in
Google Cloud Storage.
gce                    create or terminate GCE instances
gce_lb                 create/destroy GCE load-balancer resources
gce_net                create/destroy GCE networks and firewall
rules
gce_pd                 utilize GCE persistent disk resources
gem                    Manage Ruby gems
get_url                Downloads files from HTTP, HTTPS, or FTP
to node
git                    Deploy software (or files) from git
checkouts
github_hooks           Manages github service hooks.
glance_image           Add/Delete images from glance
group                  Add or remove groups
group_by               Create Ansible groups based on facts
grove                  Sends a notification to a grove.io channel
hg                     Manages Mercurial (hg) repositories.
hipchat                Send a message to hipchat
homebrew               Package manager for Homebrew
homebrew_cask          Install/uninstall homebrew casks.
homebrew_tap           Tap a Homebrew repository.
hostname               Manage hostname
htpasswd               manage user files for basic authentication
include_vars           Load variables from files, dynamically
within a task.
ini_file               Tweak settings in INI files
irc                    Send a message to an IRC channel
jabber                 Send a message to jabber user or chat room
```

```
jboss                deploy applications to JBoss
jira                 create and modify issues in a JIRA
instance
kernel_blacklist     Blacklist kernel modules
keystone_user        Manage OpenStack Identity (keystone)
users, tenants and role
layman               Manage Gentoo overlays
librato_annotation   create an annotation in librato
lineinfile           Ensure a particular line is in a file, or
replace an existin
linode               create / delete / stop / restart an
instance in Linode Publi
lldp                 get details reported by lldp
locale_gen           Creates of removes locales.
logentries           Module for tracking logs via
logentries.com
lvg                  Configure LVM volume groups
lvol                 Configure LVM logical volumes
macports             Package manager for MacPorts
mail                 Send an email
modprobe             Add or remove kernel modules
mongodb_user         Adds or removes a user from a MongoDB
database.
monit                Manage the state of a program monitored
via Monit
mount                Control active and configured mount points
mqtt                 Publish a message on an MQTT topic for the
IoT
mysql_db             Add or remove MySQL databases from a
remote host.
mysql_replication    Manage MySQL replication
mysql_user           Adds or removes a user from a MySQL
database.
mysql_variables      Manage MySQL global variables
nagios               Perform common tasks in Nagios related to
downtime and notif
netscaler            Manages Citrix NetScaler entities
newrelic_deployment  Notify newrelic about app deployments
nexmo                Send a SMS via nexmo
nova_compute         Create/Delete VMs from OpenStack
nova_keypair         Add/Delete key pair from nova
npm                  Manage node.js packages with npm
ohai                 Returns inventory data from `Ohai'
open_iscsi           Manage iscsi targets with open-iscsi
openbsd_pkg          Manage packages on OpenBSD.
```

```
openvswitch_bridge   Manage Open vSwitch bridges
openvswitch_port     Manage Open vSwitch ports
opkg                 Package manager for OpenWrt
osx_say              Makes an OSX computer to speak.
ovirt                oVirt/RHEV platform management
pacman               Manage packages with `pacman'
pagerduty            Create PagerDuty maintenance windows
pause                Pause playbook execution
ping                 Try to connect to host and return `pong'
on success.
pingdom              Pause/unpause Pingdom alerts
pip                  Manages Python library dependencies.
pkgin                Package manager for SmartOS
pkgng                Package manager for FreeBSD >= 9.0
pkgutil              Manage CSW-Packages on Solaris
portage              Package manager for Gentoo
portinstall          Installing packages from FreeBSD's ports
system
postgresql_db        Add or remove PostgreSQL databases from a
remote host.
postgresql_privs     Grant or revoke privileges on PostgreSQL
database objects...
postgresql_user      Adds or removes a users (roles) from a
PostgreSQL database..
quantum_floating_ip  Add/Remove floating IP from an instance
quantum_floating_ip_associate Associate or disassociate a
particular floating IP with an i
quantum_network      Creates/Removes networks from OpenStack
quantum_router       Create or Remove router from openstack
quantum_router_gateway set/unset a gateway interface for the
router with the specif
quantum_router_interface Attach/Dettach a subnet's interface to
a router
quantum_subnet       Add/Remove floating IP from an instance
rabbitmq_parameter   Adds or removes parameters to RabbitMQ
rabbitmq_plugin      Adds or removes plugins to RabbitMQ
rabbitmq_policy      Manage the state of policies in RabbitMQ.
rabbitmq_user        Adds or removes users to RabbitMQ
rabbitmq_vhost       Manage the state of a virtual host in
RabbitMQ
raw                  Executes a low-down and dirty SSH command
rax                  create / delete an instance in Rackspace
Public Cloud
rax_cbs              Manipulate Rackspace Cloud Block Storage
Volumes
```

```
rax_cbs_attachments  Manipulate Rackspace Cloud Block Storage
Volume Attachments.
rax_clb              create / delete a load balancer in
Rackspace Public Cloud...
rax_clb_nodes        add, modify and remove nodes from a
Rackspace Cloud Load Bal
rax_dns              Manage domains on Rackspace Cloud DNS
rax_dns_record       Manage DNS records on Rackspace Cloud DNS
rax_facts            Gather facts for Rackspace Cloud Servers
rax_files            Manipulate Rackspace Cloud Files
Containers
rax_files_objects    Upload, download, and delete objects in
Rackspace Cloud File
rax_identity         Load Rackspace Cloud Identity
rax_keypair          Create a keypair for use with Rackspace
Cloud Servers
rax_meta             Manipulate metadata for Rackspace Cloud
Servers
rax_network          create / delete an isolated network in
Rackspace Public Clou
rax_queue            create / delete a queue in Rackspace
Public Cloud
rax_scaling_group    Manipulate Rackspace Cloud Autoscale
Groups
rax_scaling_policy   Manipulate Rackspace Cloud Autoscale
Scaling Policy
rds                  create, delete, or modify an Amazon rds
instance
rds_param_group      manage RDS parameter groups
rds_subnet_group     manage RDS database subnet groups
redhat_subscription  Manage Red Hat Network registration and
subscriptions using
redis                Various redis commands, slave and flush
replace              Replace all instances of a particular
string in a file using
rhn_channel          Adds or removes Red Hat software channels
rhn_register         Manage Red Hat Network registration using
the `rhnreg_ks' co
riak                 This module handles some common Riak
operations
rollbar_deployment   Notify Rollbar about app deployments
route53              add or delete entries in Amazons Route53
DNS service
rpm_key              Adds or removes a gpg key from the rpm db
s3                   idempotent S3 module putting a file into
```

```
S3.
script                Runs a local script on a remote node after
transferring it..
seboolean             Toggles SELinux booleans.
selinux               Change policy and state of SELinux
service               Manage services.
set_fact              Set host facts from a task
setup                 Gathers facts about remote hosts
shell                 Execute commands in nodes.
slack                 Send Slack notifications
slurp                 Slurps a file from remote nodes
sns                   Send Amazon Simple Notification Service
(SNS) messages
stackdriver           Send code deploy and annotation events to
stackdriver
stat                  retrieve file or file system status
subversion            Deploys a subversion repository.
supervisorctl         Manage the state of a program or group of
programs running v
svr4pkg               Manage Solaris SVR4 packages
swdepot               Manage packages with swdepot package
manager (HP-UX)
synchronize           Uses rsync to make synchronizing file
paths in your playbook
sysctl                Manage entries in sysctl.conf.
template              Templates a file out to a remote server.
twilio                Sends a text message to a mobile phone
through Twilio.
typetalk              Send a message to typetalk
ufw                   Manage firewall with UFW
unarchive             Copies an archive to a remote location and
unpack it
uri                   Interacts with webservices
urpmi                 Urpmi manager
user                  Manage user accounts
virt                  Manages virtual machines supported by
libvirt
vsphere_guest         Create/delete/manage a guest VM through
VMware vSphere.
wait_for              Waits for a condition before continuing.
win_feature           Installs and uninstalls Windows Features
win_get_url           Fetches a file from a given URL
win_group             Add and remove local groups
win_msi               Installs and uninstalls Windows MSI files
win_ping              A windows version of the classic ping
```

```
module.
win_service           Manages Windows services
win_stat              returns information about a Windows file
win_user              Manages local Windows user accounts
xattr                 set/retrieve extended attributes
yum                   Manages packages with the `yum' package
manager
zfs                   Manage zfs
zypper                Manage packages on SuSE and openSuSE
zypper_repository     Add and remove Zypper repositories
```

查看模块帮助文档

```
# ansible-doc ping
> PING

  A trivial test module, this module always returns `pong' on
  successful contact. It does not make sense in playbooks, but
it is
  useful from `/usr/bin/ansible'

# Test 'webservers' status
ansible webservers -m ping
```

# 5. ansible-playbook - run an ansible playbook

定义组

```
# cat /etc/ansible/hosts
[www]
192.168.2.23
```

创建yml文件

```
# cat test.yml
---
- hosts: www
  user: root
  tasks:
  - name: no selinux
    action: command /usr/sbin/setenforce 0

  - name: no iptables
    action: service name=iptables state=stopped

  - name: made up task just to show variables work here
    action: command /bin/echo release is $release
```

执行任务

```
# ansible-playbook test.yml -u root -T 1

PLAY [www] ********************

GATHERING FACTS ********************
ok: [192.168.2.23]

TASK: [no selinux] ********************
ok: [192.168.2.23]
```

```
TASK: [no iptables] *********************
ok: [192.168.2.23]

TASK: [made up task just to show variables work here]
*********************
ok: [192.168.2.23]

PLAY RECAP ********************
192.168.2.23                  : ok=4    changed=2
unreachable=0    failed=0
```

```
# ansible-playbook update.yml --list-hosts

playbook: update.yml

  play #1 (all): host count=11
    192.168.2.10
    192.168.2.11
    192.168.2.12
    192.168.2.13
    192.168.2.15
    192.168.6.10
    192.168.6.11
    192.168.6.12
    192.168.6.13
    192.168.6.15
    192.168.2.9
```

## 5.1. 包含文件用法

我们将下面的playbook文件分成三个文件，这样更灵活。

```
---
- hosts: all
  remote_user: username
  sudo: yes

  tasks:
    - yum: name=wget state=present
```

```
      when: ansible_distribution == 'CentOS' or
ansible_distribution == 'Red Hat Enterprise Linux'
    - yum: name=gcc state=present
      when: ansible_distribution == 'CentOS' or
ansible_distribution == 'Red Hat Enterprise Linux'
```

tasks/cenos.yml

```
- yum: name=wget state=present
- yum: name=gcc state=present
```

tasks/deban.yml

```
- apt: pkg=wget state=present
- apt: pkg=gcc state=present
```

playbook.yml

```
---
- hosts: all
  remote_user: username
  sudo: yes

  tasks:
    - include: tasks/centos.yml
      when: ansible_distribution == 'CentOS' or
ansible_distribution == 'Red Hat Enterprise Linux'
    - include: tasks/debian.yml
      when: ansible_distribution == 'Debian' or
ansible_distribution == 'Ubuntu'
```

执行playbook

```
# ansible-playbook playbook.yml
```

# 第 25 章 Capistrano

# 第 26 章 Puppet

http://www.puppetlabs.com

Puppet is the leading open source platform for IT systems management

# 1. Installing Puppet CentOS 6.3

Choose a Package Source http://yum.puppetlabs.com/

```
# rpm -Uvh http://dl.fedoraproject.org/pub/epel/6/x86_64/epel-
release-6-7.noarch.rpm
# lokkit --disabled --selinux=disabled
```

Install the Puppet Master

```
yum install puppet-server -y
service puppetmaster start

chkconfig puppetmaster on
```

Install Puppet on Agent Nodes

```
yum install puppet -y
service puppet start

chkconfig puppet on
```

# 2. Puppet 签名

```
cat >> /etc/hosts <<EOD
172.16.0.1      puppet.mydomain.com puppet
172.16.0.20     www.mydomain.com www
172.16.0.21     images.mydomain.com images
EOD
```

## 2.1. Agent 节点

Node: 服务端进行认证

```
puppetd --test --server puppet
```

### 例 26.1. puppetd

```
# puppetd --test --server puppet
info: Creating a new SSL key for haproxy
warning: peer certificate won't be verified in this SSL session
info: Caching certificate for ca
warning: peer certificate won't be verified in this SSL session
warning: peer certificate won't be verified in this SSL session
info: Creating a new SSL certificate request for haproxy
info: Certificate Request fingerprint (md5):
91:ED:04:2B:13:8C:61:8F:ED:8E:10:31:CA:8E:5C:06
warning: peer certificate won't be verified in this SSL session
warning: peer certificate won't be verified in this SSL session
warning: peer certificate won't be verified in this SSL session
Exiting; no certificate found and waitforcert is disabled
```

## 2.2. Master 服务器

认证所有的客户端

```
puppetca -s -a
```

或者认证某一台客户端

```
puppetca -l
puppetca -sign www.mydomain.com
```

**例 26.2. puppetca**

```
# puppetca --list
  "haproxy" (91:ED:04:2B:13:8C:61:8F:ED:8E:10:31:CA:8E:5C:06)

# puppetca --sign haproxy
notice: Signed certificate request for haproxy
notice: Removing file Puppet::SSL::CertificateRequest haproxy
at '/var/lib/puppet/ssl/ca/requests/haproxy.pem'
```

# 3. test

## 3.1. Master

```
vim /etc/puppet/manifests/site.pp

node default { file { "/tmp/puppettest1.txt": content =>
"hello,first puppet manifest"; } }
```

## 3.2. Agent

```
# puppetd --test --server puppet
info: Caching catalog for www.mydomain.com
info: Applying configuration version '1351280410'
notice:
/Stage[main]//Node[default]/File[/tmp/puppettest1.txt]/ensure:
defined content as '{md5}886609dedc5c8a0c58f3aa8d566175cc'
info: Creating state file /var/lib/puppet/state/state.yaml
notice: Finished catalog run in 0.06 seconds
```

```
# cat /tmp/puppettest1.txt
hello,first puppet manifest
```

# 4. 配置文件

## 4.1. /etc/sysconfig/puppet

```
# The puppetmaster server
#PUPPET_SERVER=puppet

# If you wish to specify the port to connect to do so here
#PUPPET_PORT=8140

# Where to log to. Specify syslog to send log messages to the
system log.
#PUPPET_LOG=/var/log/puppet/puppet.log

# You may specify other parameters to the puppet client here
#PUPPET_EXTRA_OPTS=--waitforcert=500
```

## 4.2. /etc/puppet/fileserver.conf

```
# cat /etc/puppet/fileserver.conf

# This file consists of arbitrarily named sections/modules
# defining where files are served from and to whom

# Define a section 'files'
# Adapt the allow/deny settings to your needs. Order
# for allow/deny does not matter, allow always takes precedence
# over deny
# [files]
#  path /var/lib/puppet/files
#  allow *.example.com
#  deny *.evil.example.com
#  allow 192.168.0.0/24
#
[files]
path /var/lib/puppet/files
allow *
```

# 5. manifests

http://docs.puppetlabs.com/learning/

## 5.1. node

default 针对所有节点

```
node default {
        file {
        "/tmp/helloworld.txt": content => "hello, world";
        }
}
```

```
# cat /etc/puppet/manifests/site.pp
node default {
        file {
                "/tmp/puppettest1.txt":
                        content => "hello,first puppet
manifest";
        }
}
```

指定节点

```
# cat /etc/puppet/manifests/test.pp
node www {
    file { "/var/www/index.html":
        source => "/tmp/something",
        mode   => 666;
    }
}
```

多个节点

```
node 'www','images' {
        ...
        ...
}
```

## 5.2. group, user 用户组管理

http://docs.puppetlabs.com/references/latest/type.html#user

http://docs.puppetlabs.com/references/latest/type.html#group

如果没有指定name的话就会建立和资源名一样的用户名/组名，如果指定了name就以name指定的用户名/组名为主

### group

用户组的添加

```
node 'node1.example.com' {
#为该节点添加一个名字为test的组，并设置组ID为1000，如果不指定name的值，所创建的用户就为web。
        group { "web":
        ensure => "present",
        gid => 1000,
        name => "test";
        }
#为该节点添加一个httpd的组，并且设置ID和web一样
        group { "httpd":
        ensure => "present",
        gid => 1000,
        allowdupe => true;
        }
#为该节点删除一个apache的组。
        group { "apache":
        ensure => "absent",
        }
}
```

用户组的删除

```
node 'node1.example.com' {
#为该节点删除一个web的组。
        group { "web":
        ensure => "absent",
        }
}
```

**user**

用户的添加

```
#创建一个用户并且密码为空
user {"svn":
        ensure => "present",
        shell => "/sbin/nologin";
}

#创建一个www用户，设置用户描述为webmaster,shell为bash,
user {"www":
        ensure => "present",
        comment => "webmaster user",
        name => "www",
        shell => "/sbin/bash";
}

#创建一个gid为80的用户组:
group { "www":
        ensure => "present",
        gid => 80,
        }
```

用户的删除

```
user { "neo":
    ensure => "absent",
```

```
}
```

创建用户并指定密码

生成密码

```
# grub-md5-crypt
Password:
Retype password:
$1$ZlJ1u0$tdv/dr8pYuHh.eT47F6b70
```

```
user { "www":
    ensure => "present",
    uid => 80,
    gid => 80,
    home => "/var/www",
    shell => "/bin/bash",
    managehome => true,
        password => '$1$ZlJ1u0$tdv/dr8pYuHh.eT47F6b70';
}

file {"/var/www":
        group => 80,
        owner => 80,
        mode => 700,
        ensure => directory;
}
```

## 5.3. file

```
file { "/var/www/my/file":
    source => "/path/in/nfs/or/something",
    mode   => 666;
}
```

**ensure**

```
ensure => absent;        #absent是检测文件是否存在，如果存在则删除
ensure => present;       #present正好相反，如果不存在则创建
ensure => directory; #创建一个目录的方法
force = > true;          #删除一个目录必须加上这个参数
source => "PATH";        #指定数据来源
backup => ".backup_$uptime_seconds"; 覆盖前备份文件
```

创建目录实例

```
file { "/tmp/cache":
  owner => "www",
  group => "www",
  mode => 700,
  ensure => directory;
}
```

**source**

source 表示 agent节点上的目录

```
node www {
    file { "/var/www":
        owner => "nginx",
        group => "nginx",
        mode => 700,
        ensure => directory;
    }

    file { "/var/www/index.html":
        source => "/tmp/something",
        mode   => 666;
    }
}
```

从master上获取文件

fileserver.conf 配置如下

```
[files]
path /var/lib/puppet/files
allow *
```

site.pp配置如下

```
file { "/tmp/test.txt":
        source  =>
"puppet://puppet.example.com/files/test.txt",
    }
```

此处的files为fileserver.conf中定义模块

**owner, group, mode**

```
file
{ "/opt/testfile":
        owner => "puppet",
        group => "puppet",
        mode => 777;
}
```

## 5.4. package

```
present, installed      安装包
absent, pureged         卸载包
```

```
# start
```

```
package {
       "dnsmasq":
               ensure => installed;
       }

file {
       "/etc/resolv.conf":
               require => Service["dnsmasq"],
               content => "nameserver 127.0.0.1\n";
       }
service {
       "dnsmasq":
               ensure => running,
               pattern => "dnsmasq" ,
               require => Package["dnsmasq"];
       }
# end
```

```
package {
        "httpd":
                ensure    => installed;          安装httpd，或用
present也表示安装
        ["vim","vsftpd"]:
                ensure=>absent;                                  删除vim
和vsftpd软件，使用pureged表示彻底删除软件
}
```

```
$package_list = [ "screen", "strace", "sudo" ]
package { $package_list: ensure => "installed" }
```

```
package { "lamp":
        ensure => present,
        provider => rpm,
        source => "http://192.168.0.1/lamp.rpm";
}
```

## 5.5. service

```
service { 'sshd':
      ensure     => running,
      enable     => true,
      hasrestart => true,
      hasstatus  => true,
      subscribe  => File['/etc/ssh/sshd_config'],
}
```

## 5.6. exec

```
exec { "creates file":
        cwd => "/tmp",
#指定命令执行的目录。如果目录不存在，则命令执行失败。
        command => "/bin/echo helloworld > /tmp/hello.txt",
        user => "root",
        path =>
"/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin";
#命令执行的搜索路径。如果path没有被定义，命令需要使用绝对路径。
}
```

```
exec { “/srv/puppet/shell/test.sh”:
    cwd => “/srv/puppet”,
    timeout => 7200,
    logoutput => on_failure,
    user => root,
    path => ["/sbin", "/usr/sbin", "/usr/local/sbin",
"/usr/local/bin", "/usr/bin", "/bin",
"/usr/local/java/jre/bin"],
    require => File["/srv/puppet/shell/test.sh"]
}
```

## 5.7. cron

```
cron{ ntpdate:
      command => "/usr/sbin/ntpdate 172.16.0.1",
      user => root,
```

```
      minute =>'*/5',
      require => Package["crontabs"];
}
```

```
file { "/etc/cron.hourly/backup":
        mode => 755,
        owner => root,
        group => root,
        require => Package[mysql],
        content => template("db/backup.erb");
}
```

# 6. modules

```
$ git clone http://github.com/example42/puppet-modules.git

mv puppet-modules /etc/puppet/modules

# vi /etc/puppet/puppet.conf
...
[master]
    modulepath = /etc/puppet/modules

# /etc/init.d/puppetmaster restart
```

```
vi /etc/puppet/manifests/node.pp

node 'web.example.com' {
    include apache

    include php
    include php::pear
    include php::apc
    php::module { mysql: }
    php::module { curl: }
    php::module { gd: }
    php::module { idn: }
    php::module { imagick: }
    php::module { imap: }
    php::module { mcrypt: }
    php::module { ming: }
    php::module { ps: }
    php::module { pspell: }
    php::module { recode: }
    php::module { snmp: }
    php::module { tidy: }
    php::module { xmlrpc: }
    php::module { xsl: }
    php::module { ldap: }

    include mysql
```

```
}
```

```
puppet agent --test --server=puppet.example.com
```

# 7. firewall 配置

```
-A INPUT -p tcp -m state --state NEW --dport 8140 -j ACCEPT
```

# 8. debug

## 8.1. master

```
puppetmasterd --debug --daemonize --verbose
```

## 8.2. node

```
puppetd --test --trace --debug
```

```
# puppetd --test --trace --debug
debug: Puppet::Type::User::ProviderDirectoryservice: file
/usr/bin/dscl does not exist
debug: Puppet::Type::User::ProviderUser_role_add: file roledel
does not exist
debug: Puppet::Type::User::ProviderPw: file pw does not exist
debug: Puppet::Type::User::ProviderLdap: true value when
expecting false
debug: Failed to load library 'rubygems' for feature 'rubygems'
debug: Puppet::Type::File::ProviderMicrosoft_windows: feature
microsoft_windows is missing
debug: Failed to load library 'ldap' for feature 'ldap'
debug: /File[/var/lib/puppet/state/state.yaml]: Autorequiring
File[/var/lib/puppet/state]
debug: /File[/var/lib/puppet/state]: Autorequiring
File[/var/lib/puppet]
debug: /File[/var/lib/puppet/ssl/public_keys/info.com.pem]:
Autorequiring File[/var/lib/puppet/ssl/public_keys]
debug: /File[/var/lib/puppet/ssl]: Autorequiring
File[/var/lib/puppet]
debug: /File[/var/lib/puppet/ssl/certificate_requests]:
Autorequiring File[/var/lib/puppet/ssl]
debug: /File[/etc/puppet/puppet.conf]: Autorequiring
File[/etc/puppet]
debug: /File[/var/lib/puppet/ssl/certs]: Autorequiring
File[/var/lib/puppet/ssl]
```

```
debug: /File[/var/lib/puppet/clientbucket]: Autorequiring
File[/var/lib/puppet]
debug: /File[/var/lib/puppet/ssl/certs/ca.pem]: Autorequiring
File[/var/lib/puppet/ssl/certs]
debug: /File[/var/lib/puppet/ssl/private]: Autorequiring
File[/var/lib/puppet/ssl]
debug: /File[/var/lib/puppet/facts]: Autorequiring
File[/var/lib/puppet]
debug: /File[/var/lib/puppet/ssl/private_keys/info.com.pem]:
Autorequiring File[/var/lib/puppet/ssl/private_keys]
debug: /File[/var/lib/puppet/ssl/crl.pem]: Autorequiring
File[/var/lib/puppet/ssl]
debug: /File[/var/lib/puppet/lib]: Autorequiring
File[/var/lib/puppet]
debug: /File[/var/lib/puppet/client_yaml]: Autorequiring
File[/var/lib/puppet]
debug: /File[/var/lib/puppet/state/last_run_summary.yaml]:
Autorequiring File[/var/lib/puppet/state]
debug: /File[/var/lib/puppet/ssl/certs/info.com.pem]:
Autorequiring File[/var/lib/puppet/ssl/certs]
debug: /File[/var/lib/puppet/client_data]: Autorequiring
File[/var/lib/puppet]
debug: /File[/var/lib/puppet/ssl/public_keys]: Autorequiring
File[/var/lib/puppet/ssl]
debug: /File[/var/lib/puppet/ssl/private_keys]: Autorequiring
File[/var/lib/puppet/ssl]
debug: /File[/var/lib/puppet/state/graphs]: Autorequiring
File[/var/lib/puppet/state]
debug: /File[/var/run/puppet/agent.pid]: Autorequiring
File[/var/run/puppet]
debug: /File[/var/lib/puppet/classes.txt]: Autorequiring
File[/var/lib/puppet]
debug: /File[/var/lib/puppet/state/state.yaml]/mode: mode
changed '640' to '660'
debug: Finishing transaction 70258153162980
debug: /File[/var/lib/puppet/ssl/certs]: Autorequiring
File[/var/lib/puppet/ssl]
debug: /File[/var/lib/puppet/ssl/private_keys]: Autorequiring
File[/var/lib/puppet/ssl]
debug: /File[/var/lib/puppet/ssl/private]: Autorequiring
File[/var/lib/puppet/ssl]
debug: /File[/var/lib/puppet/ssl/crl.pem]: Autorequiring
File[/var/lib/puppet/ssl]
debug: /File[/var/lib/puppet/ssl/certs/info.com.pem]:
Autorequiring File[/var/lib/puppet/ssl/certs]
```

```
debug: /File[/var/lib/puppet/lib]: Autorequiring
File[/var/lib/puppet]
debug: /File[/var/lib/puppet/ssl/certificate_requests]:
Autorequiring File[/var/lib/puppet/ssl]
debug: /File[/var/lib/puppet/ssl/public_keys]: Autorequiring
File[/var/lib/puppet/ssl]
debug: /File[/var/lib/puppet/state]: Autorequiring
File[/var/lib/puppet]
debug: /File[/var/lib/puppet/ssl/private_keys/info.com.pem]:
Autorequiring File[/var/lib/puppet/ssl/private_keys]
debug: /File[/var/lib/puppet/ssl/certs/ca.pem]: Autorequiring
File[/var/lib/puppet/ssl/certs]
debug: /File[/var/lib/puppet/ssl]: Autorequiring
File[/var/lib/puppet]
debug: /File[/var/lib/puppet/facts]: Autorequiring
File[/var/lib/puppet]
debug: /File[/var/lib/puppet/ssl/public_keys/info.com.pem]:
Autorequiring File[/var/lib/puppet/ssl/public_keys]
debug: Finishing transaction 70258153219940
debug: Using cached certificate for ca
debug: Using cached certificate for info.com
debug: Finishing transaction 70258152746740
debug: Loaded state in 0.00 seconds
debug: Using cached certificate for ca
debug: Using cached certificate for info.com
debug: Using cached certificate_revocation_list for ca
debug: catalog supports formats: b64_zlib_yaml dot pson raw
yaml; using pson
info: Caching catalog for info.com
debug: Creating default schedules
debug: Loaded state in 0.00 seconds
info: Applying configuration version '1351280410'
debug: Finishing transaction 70258154614200
debug: Storing state
debug: Stored state in 0.00 seconds
notice: Finished catalog run in 0.02 seconds
```

# 9. FAQ

## 9.1. err: Could not request certificate: No route to host - connect(2)

```
err: Could not request certificate: Connection refused -
connect(2)
Exiting; failed to retrieve certificate and waitforcert is
disabled
```

关闭防火墙可以解决

## 9.2. No help available unless you have RDoc::usage installed

```
#  puppetmasterd --help
No help available unless you have RDoc::usage installed
```

```
# yum install ruby-rdoc
```

# 第 27 章 SaltStack

http://saltstack.com/

# 1. 安装 Salt Stack

## 1.1. 服务端安装

```
yum install salt-master
chkconfig salt-master on
service salt-master start
```

```
cp /etc/salt/master{,.original}
```

## 1.2. 客户端安装

```
yum install salt-minion
chkconfig salt-minion on
```

配置 master

```


cp /etc/salt/minion{,.original}
sed -i '12,12imaster: salt.example.org' /etc/salt/minion

cat >> /etc/hosts <<'EOF'

192.168.2.1    salt.example.org
EOF


```

```
service salt-minion start
```

## 1.3. 防火墙配置

```
-A INPUT -p tcp -m multiport --dports 4505,4506 -m state --
state NEW -j ACCEPT
```

## 1.4. key 管理

登陆master服务器，输入 salt-key 查看接入的 minion 客户端。

```
# salt-key
Accepted Keys:
Unaccepted Keys:
haproxy
Rejected Keys:
```

接受客户端 key

```
# salt-key -a haproxy
The following keys are going to be accepted:
Unaccepted Keys:
haproxy
Proceed? [n/Y] y
Key for minion haproxy accepted.
```

至此，master 与 minion 已经建立了信任关系

## 1.5. 测试

你可以运行下面命令测试你的 minion

```
salt '*' test.arg 1 "two" 3.1 txt="hello" wow='{a: 1, b:
"hello"}'
salt '*' test.arg_repr 1 "two" 3.1 txt="hello" wow='{a: 1, b:
"hello"}'
salt '*' test.collatz 3
salt '*' test.conf_test
salt '*' test.cross_test file.gid_to_group 0
salt '*' test.echo 'foo bar baz quo qux'
salt '*' test.fib 3
salt '*' test.get_opts
salt '*' test.kwarg num=1 txt="two" env='{a: 1, b: "hello"}'
salt '*' test.not_loaded
salt '*' test.outputter foobar
salt '*' test.ping
salt '*' test.provider service
salt '*' test.providers
salt '*' test.rand_sleep 60
salt '*' test.retcode 42
salt '*' test.sleep 20
salt '*' test.tty tty0 'This is a test'
salt '*' test.tty pts3 'This is a test'
salt '*' test.version
salt '*' test.versions_information
salt '*' test.versions_report
```

我通常只作ping测试

```
# salt '*' test.ping
haproxy:
    True

# salt '*' test.versions_information
haproxy:
    ----------
    Jinja2:
        unknown
    M2Crypto:
        0.20.2
    PyYAML:
        3.09
    PyZMQ:
```

```
        2.2.0.1
    Python:
        2.6.6 (r266:84292, Feb 22 2013, 00:00:18)
    Salt:
        0.16.0
    ZMQ:
        3.2.3
    msgpack-pure:
        None
    msgpack-python:
        0.1.13
    pycrypto:
        2.0.1

# salt '*' test.versions_report
haproxy:
               Salt: 0.16.0
             Python: 2.6.6 (r266:84292, Feb 22 2013, 00:00:18)
             Jinja2: unknown
           M2Crypto: 0.20.2
     msgpack-python: 0.1.13
       msgpack-pure: Not Installed
           pycrypto: 2.0.1
             PyYAML: 3.09
              PyZMQ: 2.2.0.1
                ZMQ: 3.2.3
```

单独测试某一节点

```
salt 'haproxy' test.ping
```

## 1.6. Demo

这里为你掩饰的是，将iptables文件推送到所有的服务器上。

# vim /srv/salt/top.sis

```
base:
  '*':
```

```
    - iptables
```

# vim /srv/salt/iptables.sls

```
/etc/sysconfig/iptables:
  file:
    - managed
    - source: salt://iptables
    - user: root
    - group: root
    - mode: 644
    - backup: minion
```

# vim /srv/salt/iptables

```
# Firewall configuration written by system-config-firewall
# Manual customization of this file is not recommended.
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A INPUT -p icmp -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 22 -j
ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 80 -j
ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 443 -j
ACCEPT
-A INPUT -j REJECT --reject-with icmp-host-prohibited
-A FORWARD -j REJECT --reject-with icmp-host-prohibited
COMMIT
```

单独部署iptables

```
# salt '*' state.sls iptables
```

按照 top.sls 的设置执行

```
salt '*' state.highstate -v
```

# 2. salt-key - Salt key is used to manage Salt authentication keys

查询 key 状态

```
# salt-key

Accepted Keys:
centos6.example.com
haproxy.example.com
Unaccepted Keys:
Rejected Keys:
```

查看来自 minion 的 key

```
# salt-key -L
```

接受所有key

```
# salt-key -A
```

删除 key

```
# salt-key -d haproxy

The following keys are going to be deleted:
Accepted Keys:
haproxy
Proceed? [N/y] y
```

显示 key 文件内容

```
# salt-key -p centos6.example.com
Accepted Keys:
centos6.example.com:  -----BEGIN PUBLIC KEY-----
MIICIjANBgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEAw7E3x8jok8xsaKYJq94
8
HsElTLGR1AVBNoIT2EDAZkX/Zhyl4d0BT1C9JJN6/R/9tV9yr1+p7/GbbtPr+9G
K
9ahO4S9QKtVCwyQVIaFRdZTlmHspCfE2gBgqXKNZfgCH2IILS8fJBo5GMvNgajR
i
qB3TShCV/UOBmkQ2H6p52xfhuubpAzrmdVVZOa03ASBTkNRUmE1kDIeZcIILBug
3
XbsXYPv+Uz+rujxzSa5P2eF6O9kcr1QCNYHj3pgm2VL7ALkkbzCLbiAWSLVMmct
Q
Fx/uvRjZG+Bh/nmQ4Kz33pLgiyhHNaA+dXsF8YTrpU3QHb+UvByZH0Lwf//ekRz
T
QeSW9noZGlWkYn2uSAbEuIT9kg7xwd3rB/iEqKGBJcfn5kgaxstUywqr9i+F8Dh
d
X9hs3o46WIZMFeFX237oeXZ1lHgEwioZIYtmgEebqsMl9LT3EFG5/j2EzkEbE/D
i
hTJETbm7dttSj2tcwmb+S8hBJ6YRy1qTZsTYsdgjV3GC2U9lMeGjzhKB/znwR+k
P
8BL0JnVEVVImzdTG1Y1sGRwbHXmOPGpyKCw/oLtjxw5nZuTRMeZcEd5Jj6pZBIS
v
+Pqw+KdiiHUg47sP2G7wlZpLIvyUv69/Jq0bIP4BqTi1rDvWtSuySbkQoKFEFhM
c
Nff3pZp7Clm92WXb3+LS0z0CAwEAAQ==
-----END PUBLIC KEY-----
```

key 文件存储位置

```
# ll /etc/salt/pki/master/minions
total 8
-rw-r--r-- 1 root root 800 Aug 21 14:07 centos6.example.com
-rw-r--r-- 1 root root 800 Aug 21 14:08 haproxy.example.com
```

# 3. salt 命令

精确匹配

```
# salt 'haproxy' test.ping
haproxy:
    True
```

通配符匹配

```
# salt 'ha*' test.ping
haproxy:
    True
```

运行模块，目录如下

```
# find /srv/salt/php
/srv/salt/php
/srv/salt/php/init.sls
/srv/salt/php/php-fpm.sls
/srv/salt/php/cli.sls
```

php 目录下含多个sls文件

```
salt 'centos*' state.sls php.cli
salt 'centos*' state.sls php.php-fpm
```

## 3.1. cmd

### cmd.run

```
salt '*' cmd.run 'ls -l /etc'
```

### cmd.script

```
salt '*' cmd.script salt://iptables.sh
```

## 3.2. pkg.install

```
salt '*' pkg.install vim
```

## 3.3. network.interfaces

```
salt '*' network.interfaces
```

## 3.4. salt example

### 例 27.1. salt command

```
# salt '*' sys.doc | grep "salt '*'"

salt '*' acl.delfacl user myuser /tmp/house/kitchen
salt '*' acl.delfacl default:group mygroup /tmp/house/kitchen
salt '*' acl.delfacl d:u myuser /tmp/house/kitchen
salt '*' acl.delfacl g myuser /tmp/house/kitchen
/tmp/house/livingroom
salt '*' acl.getfacl /tmp/house/kitchen
salt '*' acl.getfacl /tmp/house/kitchen /tmp/house/livingroom
salt '*' acl.addfacl user myuser rwx /tmp/house/kitchen
salt '*' acl.addfacl default:group mygroup rx
/tmp/house/kitchen
salt '*' acl.addfacl d:u myuser 7 /tmp/house/kitchen
```

```
salt '*' acl.addfacl g mygroup 0 /tmp/house/kitchen
/tmp/house/livingroom
salt '*' acl.version
salt '*' acl.wipefacls /tmp/house/kitchen
salt '*' acl.wipefacls /tmp/house/kitchen /tmp/house/livingroom
salt '*' aliases.get_target <alias>
salt '*' aliases.has_target <alias> <target>
salt '*' aliases.list_aliases
salt '*' aliases.rm_alias <alias>
salt '*' aliases.set_target <alias> <target>
salt '*' alternatives.check_installed name path
salt '*' alternatives.display <command name>
salt '*' alternatives.install name link path priority
salt '*' alternatives.remove name path
salt '*' alternatives.show_current emacs
salt '*' archive.gunzip /tmp/sourcefile.txt.gz
salt '*' archive.gunzip template=jinja
/tmp/{{grains.id}}.txt.gz
salt '*' archive.gzip /tmp/sourcefile.txt
salt '*' archive.gzip template=jinja /tmp/{{grains.id}}.txt
salt '*' archive.rar /tmp/rarfile.rar /tmp/sourcefile1
/tmp/sourcefile2
salt '*' archive.rar template=jinja /tmp/rarfile.rar
/tmp/sourcefile1 /tmp/{{grains.id}}.txt
salt '*' archive.tar cjvf /tmp/tarfile.tar.bz2 /tmp/file_1
/tmp/file_2
salt '*' archive.tar template=jinja cjvf /tmp/salt.tar.bz2
{{grains.saltpath}}
salt '*' archive.unrar /tmp/rarfile.rar /home/strongbad/ file_1
file_2
salt '*' archive.unrar template=jinja /tmp/rarfile.rar
/tmp/{{grains.id}}/ file_1 file_2
salt '*' archive.unzip /tmp/zipfile.zip /home/strongbad/ file_1
file_2
salt '*' archive.unzip template=jinja /tmp/zipfile.zip
/tmp/{{grains.id}}/ file_1 file_2
salt '*' archive.zip /tmp/zipfile.zip /tmp/sourcefile1
/tmp/sourcefile2
salt '*' archive.zip template=jinja /tmp/zipfile.zip
/tmp/sourcefile1 /tmp/{{grains.id}}.txt
salt '*' cmd.exec_code ruby 'puts "cheese"'
salt '*' cmd.has_exec cat
salt '*' cmd.retcode "file /bin/bash"
salt '*' cmd.retcode template=jinja "file
{{grains.pythonpath[0]}}/python"
```

```
salt '*' cmd.retcode "grep f"
stdin='one\ntwo\nthree\nfour\nfive\n'
salt '*' cmd.run "ls -l | awk '/foo/{print \$2}'"
salt '*' cmd.run template=jinja "ls -l /tmp/{{grains.id}} | awk
'/foo/{print \$2}'"
salt '*' cmd.run "Get-ChildItem C:\ " shell='powershell'
salt '*' cmd.run "grep f" stdin='one\ntwo\nthree\nfour\nfive\n'
salt '*' cmd.run_all "ls -l | awk '/foo/{print \$2}'"
salt '*' cmd.run_all template=jinja "ls -l /tmp/{{grains.id}} |
awk '/foo/{print \$2}'"
salt '*' cmd.run_all "grep f"
stdin='one\ntwo\nthree\nfour\nfive\n'
salt '*' cmd.run_stderr "ls -l | awk '/foo/{print \$2}'"
salt '*' cmd.run_stderr template=jinja "ls -l
/tmp/{{grains.id}} | awk '/foo/{print \$2}'"
salt '*' cmd.run_stderr "grep f"
stdin='one\ntwo\nthree\nfour\nfive\n'
salt '*' cmd.run_stdout "ls -l | awk '/foo/{print \$2}'"
salt '*' cmd.run_stdout template=jinja "ls -l
/tmp/{{grains.id}} | awk '/foo/{print \$2}'"
salt '*' cmd.run_stdout "grep f"
stdin='one\ntwo\nthree\nfour\nfive\n'
salt '*' cmd.script salt://scripts/runme.sh
salt '*' cmd.script salt://scripts/runme.sh 'arg1 arg2 "arg 3"'
salt '*' cmd.script salt://scripts/windows_task.ps1 args=' -
Input c:\tmp\infile.txt' shell='powershell'
salt '*' cmd.script salt://scripts/runme.sh
stdin='one\ntwo\nthree\nfour\nfive\n'
salt '*' cmd.script_retcode salt://scripts/runme.sh
salt '*' cmd.script_retcode salt://scripts/runme.sh
stdin='one\ntwo\nthree\nfour\nfive\n'
salt '*' cmd.which cat
salt '*' cmd.which_bin '[pip2, pip, pip-python]'
salt '*' config.backup_mode
salt '*' config.dot_vals host
salt '*' qemu.gather_bootstrap_script True
salt '*' config.get pkg:apache
salt '*' config.manage_mode
salt '*' config.option redis.host
salt '*' config.valid_fileproto salt://path/to/file
salt '*' cp.cache_dir salt://path/to/dir
salt '*' cp.cache_file salt://path/to/file
salt '*' cp.cache_files salt://pathto/file1,salt://pathto/file1
salt '*' cp.cache_local_file /etc/hosts
salt '*' cp.cache_master
```

```
salt '*' cp.get_dir salt://path/to/dir/ /minion/dest
salt '*' cp.get_file salt://path/to/file /minion/dest
salt '*' cp.get_file "salt://{{grains.os}}/vimrc" /etc/vimrc
template=jinja
salt '*' cp.get_file_str salt://my/file
salt '*' cp.get_template salt://path/to/template /minion/dest
salt '*' cp.get_url salt://my/file /tmp/mine
salt '*' cp.get_url http://www.slashdot.org /tmp/index.html
salt '*' cp.hash_file salt://path/to/file
salt '*' cp.is_cached salt://path/to/file
salt '*' cp.list_master
salt '*' cp.list_master_dirs
salt '*' cp.list_minion
salt '*' cp.list_states
salt '*' cp.push /etc/fstab
salt '*' cron.list_tab root
salt '*' cron.list_tab root
salt '*' cron.raw_cron root
salt '*' cron.rm_job root /usr/local/weekly
salt '*' cron.rm_job root /usr/bin/foo dayweek=1
salt '*' cron.rm_env root MAILTO
salt '*' cron.rm_job root /usr/local/weekly
salt '*' cron.rm_job root /usr/bin/foo dayweek=1
salt '*' cron.set_env root MAILTO user@example.com
salt '*' cron.set_job root '*' '*' '*' '*' 1 /usr/local/weekly
salt '*' cron.set_special @hourly 'echo foobar'
salt '*' cron.write_cron_file root /tmp/new_cron
salt '*' cron.write_cron_file_verbose root /tmp/new_cron
salt '*' daemontools.full_restart <service name>
salt '*' daemontools.get_all
salt '*' daemontools.reload <service name>
salt '*' daemontools.restart <service name>
salt '*' daemontools.start <service name>
salt '*' daemontools.status <service name>
salt '*' daemontools.stop <service name>
salt '*' daemontools.term <service name>
salt '*' data.cas <key> <value> <old_value>
salt '*' data.clear
salt '*' data.dump '{'eggs': 'spam'}'
salt '*' data.getval <key>
salt '*' data.getvals <key> [<key> ...]
salt '*' data.load
salt '*' data.update <key> <value>
salt '*' disk.inodeusage
salt '*' disk.usage
```

```
salt '*' django.collectstatic <settings_module>
salt '*' django.command <settings_module> <command>
salt '*' django.createsuperuser <settings_module> user
user@example.com
salt '*' django.loaddata <settings_module> <comma delimited
list of fixtures>
salt '*' django.syncdb <settings_module>
salt '*' dnsmasq.version
salt '*' dnsmasq.get_config
salt '*' dnsmasq.get_config file=/etc/dnsmasq.conf
salt '*' dnsmasq.set_config domain=mydomain.com
salt '*' dnsmasq.set_config follow=False domain=mydomain.com
salt '*' dnsmasq.set_config file=/etc/dnsmasq.conf
domain=mydomain.com
salt '*' dnsmasq.version
salt '*' dnsutil.hosts_append /etc/hosts 127.0.0.1
ad1.yuk.co,ad2.yuk.co
salt '*' dnsutil.hosts_delete /etc/hosts ad1.yuk.co
salt '*' dnsutil.hosts_delete /etc/hosts ad2.yuk.co,ad1.yuk.co
salt '*' dnsutil.parse_hosts
salt '*' event.fire 'stuff to be in the event' 'tag'
salt '*' event.fire_master 'stuff to be in the event' 'tag'
salt '*' extfs.attributes /dev/sda1
salt '*' extfs.blocks /dev/sda1
salt '*' extfs.dump /dev/sda1
salt '*' extfs.mkfs /dev/sda1 fs_type=ext4 opts='acl,noexec'
salt '*' extfs.tune /dev/sda1 force=True label=wildstallyns
opts='acl,noexec'
salt '*' file.append /etc/motd \
salt '*' file.check_file_meta /etc/httpd/conf.d/httpd.conf
salt://http/httpd.conf '{hash_type: 'md5', 'hsum': <md5sum>}'
root, root, '755' base
salt '*' file.check_hash /etc/fstab md5=<md5sum>
salt '*' file.check_managed /etc/httpd/conf.d/httpd.conf
salt://http/httpd.conf '{hash_type: 'md5', 'hsum': <md5sum>}'
root, root, '755' jinja True None None base
salt '*' file.check_perms /etc/sudoers '{}' root root 400
salt '*' file.chgrp /etc/passwd root
salt '*' file.chown /etc/passwd root root
salt '*' file.comment /etc/modules pcspkr
salt '*' file.contains /etc/crontab 'mymaintenance.sh'
salt '*' file.contains_glob /etc/foobar '*cheese*'
salt '*' file.contains_regex /etc/crontab
salt '*' file.contains_regex_multiline /etc/crontab '^maint'
salt '*' file.directory_exists /etc
```

```
salt '*' file.file_exists /etc/passwd
salt '*' file.find / type=f name=\*.bak size=+10m
salt '*' file.find /var mtime=+30d size=+10m
print=path,size,mtime
salt '*' file.find /var/log name=\*.[0-9] mtime=+30d size=+10m
delete
salt '*' file.get_diff /home/fred/.vimrc
salt://users/fred/.vimrc
salt '*' file.get_gid /etc/passwd
salt '*' file.get_group /etc/passwd
salt '*' file.get_hash /etc/shadow
salt '*' file.get_managed /etc/httpd/conf.d/httpd.conf jinja
salt://http/httpd.conf '{hash_type: 'md5', 'hsum': <md5sum>}'
root root '755' base None None
salt '*' file.get_mode /etc/passwd
salt '*' file.get_selinux_context /etc/hosts
salt '*' file.get_sum /etc/passwd sha512
salt '*' file.get_uid /etc/passwd
salt '*' file.get_user /etc/passwd
salt '*' file.gid_to_group 0
salt '*' file.group_to_gid root
salt '*' file.makedirs /opt/code
salt '*' file.makedirs_perms /opt/code
salt '*' file.manage_file /etc/httpd/conf.d/httpd.conf '{}'
salt://http/httpd.conf '{hash_type: 'md5', 'hsum': <md5sum>}'
root root '755' base ''
salt '*' file.mkdir /opt/jetty/context
salt '*' file.patch /opt/file.txt /tmp/file.txt.patch
salt '*' file.sed /etc/httpd/httpd.conf 'LogLevel warn'
'LogLevel info'
salt '*' file.remove /tmp/foo
salt '*' file.rename /path/to/src /path/to/dst
salt '*' file.restorecon /home/user/.ssh/authorized_keys
salt '*' file.sed /etc/httpd/httpd.conf 'LogLevel warn'
'LogLevel info'
salt '*' file.contains /etc/crontab 'mymaintenance.sh'
salt '*' file.set_mode /etc/passwd 0644
salt '*' file.set_selinux_context path <role> <type> <range>
salt '*' file.source_list salt://http/httpd.conf '{hash_type:
'md5', 'hsum': <md5sum>}' base
salt '*' file.stats /etc/passwd
salt '*' file.symlink /path/to/file /path/to/link
salt '*' file.touch /var/log/emptyfile
salt '*' file.uid_to_user 0
salt '*' file.uncomment /etc/hosts.deny 'ALL: PARANOID'
```

```
salt '*' file.user_to_uid root
salt '*' gem.install vagrant
salt '*' gem.sources_add http://rubygems.org/
salt '*' gem.sources_list
salt '*' gem.sources_remove http://rubygems.org/
salt '*' gem.uninstall vagrant
salt '*' gem.update vagrant
salt '*' gem.update_system
salt '*' git.add /path/to/git/repo /path/to/file
salt '*' git.archive /path/to/repo /path/to/archive.tar.gz
salt '*' git.checkout /path/to/repo somebranch user=jeff
salt '*' git.checkout /path/to/repo opts='testbranch --
conf/file1 file2'
salt '*' git.checkout /path/to/repo rev=origin/mybranch opts=--
track
salt '*' git.clone /path/to/repo
git://github.com/saltstack/salt.git
salt '*' git.clone /path/to/repo.git\
salt '*' git.commit /path/to/git/repo 'The commit message'
salt '*' git.config_get /path/to/repo user.email
salt '*' git.config_set /path/to/repo user.email me@example.com
salt '*' git.describe /path/to/repo
salt '*' git.describe /path/to/repo develop
salt '*' git.fetch /path/to/repo '--all'
salt '*' git.fetch cwd=/path/to/repo opts='--all' user=johnny
salt '*' git.init /path/to/repo.git opts='--bare'
salt '*' git.fetch /path/to/repo
salt '*' git.merge /path/to/repo @{upstream}
salt '*' git.pull /path/to/repo opts='--rebase origin master'
salt '*' git.push /path/to/git/repo remote-name
salt '*' git.rebase /path/to/repo master
salt '*' git.rebase /path/to/repo 'origin master'
salt '*' git.remote_get /path/to/repo
salt '*' git.remote_get /path/to/repo upstream
salt '*' git.remote_set /path/to/repo
remote_url=git@github.com:saltstack/salt.git
salt '*' git.remote_set /path/to/repo origin
git@github.com:saltstack/salt.git
salt '*' git.remotes /path/to/repo
salt '*' git.reset /path/to/repo master
salt '*' git.revision /path/to/repo mybranch
salt '*' git.rm /path/to/git/repo /path/to/file
salt '*' git.stash /path/to/repo master
salt '*' git.status /path/to/git/repo
salt '*' git.submodule /path/to/repo.git/sub/repo
```

```
salt '*' grains.get pkg:apache
salt '*' grains.item os
salt '*' grains.item os osrelease oscodename
salt '*' grains.item host sanitize=True
salt '*' grains.items
salt '*' grains.items sanitize=True
salt '*' grains.ls
salt '*' grains.setval key val
salt '*' group.add foo 3456
salt '*' group.chgid foo 4376
salt '*' group.delete foo
salt '*' group.getent
salt '*' group.info foo
salt '*' grub.conf
salt '*' grub.version
salt '*' hg.archive /path/to/repo output=/tmp/archive.tgz
fmt=tgz
salt '*' hg.clone /path/to/repo
https://bitbucket.org/birkenfeld/sphinx
salt '*' hg.describe /path/to/repo
salt '*' hg.pull /path/to/repo '-u'
salt '*' hg.revision /path/to/repo mybranch
salt '*' hosts.add_host <ip> <alias>
salt '*' hosts.get_alias <ip addr>
salt '*' hosts.get_ip <hostname>
salt '*' hosts.has_pair <ip> <alias>
salt '*' hosts.list_hosts
salt '*' hosts.rm_host <ip> <alias>
salt '*' hosts.set_host <ip> <alias>
salt '*' qemu_nbd.bootstrap /srv/salt-images/host.qcow 4096
qcow2
salt '*' img.mount_image /tmp/foo
salt '*' img.seed /tmp/image.qcow2
salt '*' img.umount_image /mnt/foo
salt '*' ip.apply_network_settings
salt '*' ip.build_bond bond0 mode=balance-alb
salt '*' ip.build_interface eth0 eth <settings>
salt '*' ip.build_network_settings <settings>
salt '*' ip.build_routes eth0 <settings>
salt '*' ip.down eth0
salt '*' ip.get_bond bond0
salt '*' ip.get_interface eth0
salt '*' ip.get_network_settings
salt '*' ip.get_routes eth0
salt '*' ip.up eth0
```

```
salt '*' iptables.append filter INPUT rule='-m state --state
RELATED,ESTABLISHED -j ACCEPT'
salt '*' iptables.delete filter INPUT position=3
salt '*' iptables.delete filter INPUT rule='-m state --state
RELATED,ESTABLISHED -j ACCEPT'
salt '*' iptables.flush filter
salt '*' iptables.get_policy filter INPUT
salt '*' iptables.get_rules
salt '*' iptables.get_saved_policy filter INPUT
salt '*' iptables.get_saved_policy filter INPUT
conf_file=/etc/iptables.saved
salt '*' iptables.get_saved_rules
salt '*' iptables.insert filter INPUT position=3 rule='-m state
--state RELATED,ESTABLISHED -j ACCEPT'
salt '*' iptables.save /etc/sysconfig/iptables
salt '*' iptables.set_policy filter INPUT ACCEPT
salt '*' iptables.version
salt '*' key.finger
salt '*' keyboard.get_sys
salt '*' keyboard.get_x
salt '*' keyboard.set_sys dvorak
salt '*' keyboard.set_x dvorak
salt '*' kmod.available
salt '*' kmod.check_available kvm
salt '*' kmod.is_loaded kvm
salt '*' kmod.load kvm
salt '*' kmod.lsmod
salt '*' kmod.mod_list
salt '*' kmod.remove kvm
salt '*' locale.get_locale
salt '*' locale.list_avail
salt '*' locale.set_locale 'en_US.UTF-8'
salt '*' locate.locate
salt '*' locate.stats
salt '*' locate.updatedb
salt '*' locate.version
salt '*' logrotate.set rotate 2
salt '*' logrotate.set /var/log/wtmp rotate 2
salt '*' logrotate.show_conf
salt '*' lowpkg.file_list httpd
salt '*' lowpkg.file_list httpd postfix
salt '*' lowpkg.file_list
salt '*' lowpkg.file_list httpd
salt '*' lowpkg.file_list httpd postfix
salt '*' lowpkg.file_list
```

```
salt '*' lowpkg.list_pkgs
salt '*' lowpkg.verify
salt '*' match.compound 'L@cheese,foo and *'
salt '*' match.data 'spam:eggs'
salt '*' match.glob '*'
salt '*' match.grain 'os:Ubuntu'
salt '*' match.grain_pcre 'os:Fedo.*'
salt '*' match.ipcidr '192.168.44.0/24'
salt '*' match.list 'server1,server2'
salt '*' match.pcre '.*'
salt '*' match.pillar 'cheese:foo'
salt '*' mine.get '*' network.interfaces
salt '*' mine.get 'os:Fedora' network.interfaces grain
salt '*' mine.send network.interfaces eth0
salt '*' mine.update
salt '*' monit.monitor <service name>
salt '*' monit.restart <service name>
salt '*' monit.start <service name>
salt '*' monit.stop <service name>
salt '*' monit.summary
salt '*' monit.summary <service name>
salt '*' monit.unmonitor <service name>
salt '*' mount.active
salt '*' mount.fstab
salt '*' mount.is_fuse_exec sshfs
salt '*' mount.mount /mnt/foo /dev/sdz1 True
salt '*' mount.remount /mnt/foo /dev/sdz1 True
salt '*' mount.rm_fstab /mnt/foo
salt '*' mount.set_fstab /mnt/foo /dev/sdz1 ext4
salt '*' mount.swapoff /root/swapfile
salt '*' mount.swapon /root/swapfile
salt '*' mount.swaps
salt '*' mount.umount /mnt/foo
salt '*' '*' network.arp
salt '*' network.dig archlinux.org
salt '*' network.hwaddr eth0
salt '*' network.in_subnet 10.0.0.0/16
salt '*' network.interfaces
salt '*' network.ip_addrs
salt '*' network.ip_addrs6
salt '*' network.netstat
salt '*' network.ping archlinux.org
salt '*' network.subnets
salt '*' network.traceroute archlinux.org
salt '*' nginx.configtest
```

```
salt '*' nginx.signal reload
salt '*' nginx.version
salt '*' partition.align_check /dev/sda minimal 1
salt '*' partition.check 1
salt '*' partition.cp /dev/sda 2 3
salt '*' partition.get_id /dev/sda 1
salt '*' partition.mkfs /dev/sda2 fat32
salt '*' partition.mklabel /dev/sda msdos
salt '*' partition.mkpart /dev/sda primary fat32 0 639
salt '*' partition.mkpartfs /dev/sda logical ext2 440 670
salt '*' partition.name /dev/sda 1 'My Documents'
salt '*' partition.part_list /dev/sda
salt '*' partition.part_list /dev/sda unit=s
salt '*' partition.part_list /dev/sda unit=kB
salt '*' partition.probe
salt '*' partition.probe /dev/sda
salt '*' partition.rescue /dev/sda 0 8056
salt '*' partition.resize /dev/sda 3 200 850
salt '*' partition.rm /dev/sda 5
salt '*' partition.set /dev/sda 1 boot on
salt '*' partition.set_id /dev/sda 1 83
salt '*' partition.name /dev/sda 1 boot
salt '*' pecl.install fuse
salt '*' pecl.list
salt '*' pecl.uninstall fuse
salt '*' pecl.update fuse
salt '*' pillar.data
salt '*' pillar.data key='roles'
salt '*' pillar.ext 'libvirt: _'
salt '*' pillar.get pkg:apache
salt '*' pillar.data
salt '*' pillar.data key='roles'
salt '*' pillar.raw
salt '*' pillar.raw key='roles'
salt '*' pip.freeze /home/code/path/to/virtualenv/
salt '*' pip.install <package name>,<package2 name>
salt '*' pip.install requirements=/path/to/requirements.txt
salt '*' pip.install <package name> bin_env=/path/to/virtualenv
salt '*' pip.install <package name> bin_env=/path/to/pip_bin
salt '*' pip.install markdown,django
editable=git+https://github.com/worldcompany/djangoembed.git#eg
g=djangoembed upgrade=True no_deps=True
salt '*' pip.list salt
salt '*' pip.uninstall <package name>,<package2 name>
salt '*' pip.uninstall requirements=/path/to/requirements.txt
```

```
salt '*' pip.uninstall <package name>
bin_env=/path/to/virtualenv
salt '*' pip.uninstall <package name> bin_env=/path/to/pip_bin
salt '*' pkg.latest_version <package name>
salt '*' pkg.latest_version <package name> fromrepo=epel-
testing
salt '*' pkg.latest_version <package1> <package2> <package3>
...
salt '*' pkg.clean_metadata
salt '*' pkg.compare '0.2.4-0' '<' '0.2.4.1-0'
salt '*' pkg.compare pkg1='0.2.4-0' oper='<' pkg2='0.2.4.1-0'
salt '*' pkg.del_repo myrepo
salt '*' pkg.del_repo myrepo basedir=/path/to/dir
salt '*' pkg.file_list httpd
salt '*' pkg.file_list httpd postfix
salt '*' pkg.file_list
salt '*' pkg.file_list httpd
salt '*' pkg.file_list httpd postfix
salt '*' pkg.file_list
salt '*' pkg.get_repo myrepo
salt '*' pkg.get_repo myrepo basedir=/path/to/dir
salt '*' pkg.group_diff 'Perl Support'
salt '*' pkg.group_info 'Perl Support'
salt '*' pkg.group_install groups='["Group 1", "Group 2"]'
salt '*' pkg.group_install 'My Group' skip='["foo", "bar"]'
salt '*' pkg.group_install 'My Group' include='["foo", "bar"]'
salt '*' pkg.group_list
salt '*' pkg.install <package name>
salt '*' pkg.install pkgs='["foo", "bar"]'
salt '*' pkg.install pkgs='["foo", {"bar": "1.2.3-4.el6"}]'
salt '*' pkg.install sources='[{"foo": "salt://foo.rpm"},
{"bar": "salt://bar.rpm"}]'
salt '*' pkg.latest_version <package name>
salt '*' pkg.latest_version <package name> fromrepo=epel-
testing
salt '*' pkg.latest_version <package1> <package2> <package3>
...
salt '*' pkg.list_pkgs
salt '*' pkg.list_repos
salt '*' pkg.list_upgrades
salt '*' pkg.mod_repo reponame enabled=1 gpgcheck=1
salt '*' pkg.mod_repo reponame basedir=/path/to/dir enabled=1
salt '*' pkg.mod_repo reponame baseurl=
mirrorlist=http://host.com/
salt '*' pkg.perform_cmp '0.2.4-0' '0.2.4.1-0'
```

```
salt '*' pkg.perform_cmp pkg1='0.2.4-0' pkg2='0.2.4.1-0'
salt '*' pkg.purge <package name>
salt '*' pkg.purge <package1>,<package2>,<package3>
salt '*' pkg.purge pkgs='["foo", "bar"]'
salt '*' pkg.refresh_db
salt '*' pkg.remove <package name>
salt '*' pkg.remove <package1>,<package2>,<package3>
salt '*' pkg.remove pkgs='["foo", "bar"]'
salt '*' pkg.upgrade
salt '*' pkg.upgrade_available <package name>
salt '*' pkg.verify
salt '*' pkg.version <package name>
salt '*' pkg.version <package1> <package2> <package3> ...
salt '*' pkg_resource.add_pkg '{}' bind 9
salt '*' pkg_resource.check_desired
salt '*' pkg_resource.compare
salt '*' pkg_resource.find_changes
salt '*' pkg_resource.pack_pkgs '["foo", {"bar": 1.2}, "baz"]'
salt '*' pkg_resource.pack_sources '[{"foo": "salt://foo.rpm"},
{"bar": "salt://bar.rpm"}]'
salt '*' pkg_resource.parse_targets
salt '*' pkg_resource.perform_cmp
salt '*' pkg_resource.sort_pkglist '["3.45", "2.13"]'
salt '*' pkg_resource.stringify 'vim: 7.127'
salt '*' pkg_resource.version vim
salt '*' pkg_resource.version foo bar baz
salt '*' pkg_resource.version 'python*'
salt '*' puppet.fact kernel
salt '*' puppet.facts
salt '*' puppet.noop
salt '*' puppet.noop tags=basefiles::edit,apache::server
salt '*' puppet.noop debug
salt '*' puppet.noop apply /a/b/manifest.pp
modulepath=/a/b/modules tags=basefiles::edit,apache::server
salt '*' puppet.run
salt '*' puppet.run tags=basefiles::edit,apache::server
salt '*' puppet.run debug
salt '*' puppet.run apply /a/b/manifest.pp
modulepath=/a/b/modules tags=basefiles::edit,apache::server
salt '*' quota.get_mode
salt '*' quota.off
salt '*' quota.on
salt '*' quota.report /media/data
salt '*' quota.set /media/data user=larry block-soft-
limit=1048576
```

```
salt '*' quota.set /media/data group=painters file-hard-
limit=1000
salt '*' quota.stats
salt '*' quota.warn
salt '*' rbenv.default
salt '*' rbenv.default 2.0.0-p0
salt '*' rbenv.install
salt '*' rbenv.install_ruby 2.0.0-p0
salt '*' rbenv.is_installed
salt '*' rbenv.list
salt '*' rbenv.uninstall_ruby 2.0.0-p0
salt '*' rbenv.update
salt '*' rbenv.versions
salt '*' ret.get_fun mysql network.interfaces
salt '*' ret.get_jid redis 20421104181954700505
salt '*' ret.get_jids mysql
salt '*' ret.get_minions mysql
salt '*' rvm.do 2.0.0 <command>
salt '*' rvm.gemset_copy foobar bazquo
salt '*' rvm.gemset_create 2.0.0 foobar
salt '*' rvm.gemset_delete 2.0.0 foobar
salt '*' rvm.gemset_empty 2.0.0 foobar
salt '*' rvm.gemset_list
salt '*' rvm.gemset_list_all
salt '*' rvm.get
salt '*' rvm.install
salt '*' rvm.install_ruby 1.9.3-p385
salt '*' rvm.is_installed
salt '*' rvm.list
salt '*' rvm.reinstall_ruby 1.9.3-p385
salt '*' rvm.rubygems 2.0.0 1.8.24
salt '*' rvm.set_default 2.0.0
salt '*' rvm.wrapper <ruby_string> <wrapper_prefix>
salt '*' saltutil.find_job <job id>
salt '*' saltutil.is_running state.highstate
salt '*' saltutil.kill_job <job id>
salt '*' saltutil.refresh_modules
salt '*' saltutil.refresh_pillar
salt '*' saltutil.regen_keys
salt '*' saltutil.revoke_key
salt '*' saltutil.running
salt '*' saltutil.signal_job <job id> 15
salt '*' saltutil.sync_all
salt '*' saltutil.sync_grains
salt '*' saltutil.sync_modules
```

```
salt '*' saltutil.sync_outputters
salt '*' saltutil.sync_renderers
salt '*' saltutil.sync_returners
salt '*' saltutil.sync_states
salt '*' saltutil.term_job <job id>
salt '*' saltutil.update 0.10.3
salt '*' service.disable <service name>
salt '*' service.disabled <service name>
salt '*' service.enable <service name>
salt '*' service.enabled <service name>
salt '*' service.get_all
salt '*' service.get_all limit=upstart
salt '*' service.get_all limit=sysvinit
salt '*' service.get_disabled
salt '*' service.get_disabled limit=upstart
salt '*' service.get_disabled limit=sysvinit
salt '*' service.get_enabled
salt '*' service.get_enabled limit=upstart
salt '*' service.get_enabled limit=sysvinit
salt '*' service.reload <service name>
salt '*' service.restart <service name>
salt '*' service.start <service name>
salt '*' service.status <service name>
salt '*' service.stop <service name>
salt '*' shadow.info root
salt '*' shadow.set_date username 0
salt '*' shadow.set_inactdays username 7
salt '*' shadow.set_maxdays username 90
salt '*' shadow.set_mindays username 7
salt '*' shadow.set_password root '$1$UYCIxa628.9qXjpQCjM4a..'
salt '*' shadow.set_warndays username 7
salt '*' sqlite3.fetch /root/test.db 'SELECT * FROM test;'
salt '*' sqlite3.indexes /root/test.db
salt '*' sqlite3.indices /root/test.db
salt '*' sqlite3.modify /root/test.db 'CREATE TABLE test(id
INT, testdata TEXT);'
salt '*' sqlite3.sqlite_version
salt '*' sqlite3.tables /root/test.db
salt '*' sqlite3.version
salt '*' ssh.auth_keys root
salt '*' ssh.check_key <user> <key> <enc> <comment> <options>
salt '*' root salt://ssh/keyfile
salt '*' ssh.check_known_host <user> <hostname>
key='AAAA...FAaQ=='
salt '*' ssh.get_known_host <user> <hostname>
```

```
salt '*' ssh.host_keys
salt '*' ssh.recv_known_host <hostname> enc=<enc> port=<port>
salt '*' ssh.rm_auth_key <user> <key>
salt '*' ssh.rm_known_host <user> <hostname>
salt '*' ssh.set_auth_key <user> '<key>' enc='dsa'
salt '*' ssh.set_auth_key_from_file <user>
salt://ssh_keys/<user>.id_rsa.pub
salt '*' ssh.set_known_host <user> fingerprint='xx:xx:..:xx'
enc='ssh-rsa' config='.ssh/known_hosts'
salt '*' state.clear_cache
salt '*' state.high '{"vim": {"pkg": ["installed"]}}'
salt '*' state.highstate
salt '*' state.low '{"state": "pkg", "fun": "installed",
"name": "vi"}'
salt '*' state.running
salt '*' state.show_highstate
salt '*' state.show_lowstate
salt '*' state.show_sls core,edit.vim dev
salt '*' state.show_top
salt '*' state.single pkg.installed name=vim
salt '*' state.sls core,edit.vim dev
salt '*' state.template '<Path to template on the minion>'
salt '*' state.template_str '<Template String>'
salt '*' state.top reverse_top.sls
salt '*' status.all_status
salt '*' status.cpuinfo
salt '*' status.cpustats
salt '*' status.custom
salt '*' status.diskstats
salt '*' status.diskusage [paths and/or filesystem types]
salt '*' status.diskusage         # usage for all filesystems
salt '*' status.diskusage / /tmp  # usage for / and /tmp
salt '*' status.diskusage ext?    # usage for ext[234]
filesystems
salt '*' status.diskusage / ext?  # usage for / and all ext
filesystems
salt '*' status.loadavg
salt '*' status.meminfo
salt '*' status.netdev
salt '*' status.netstats
salt '*' status.pid <sig>
salt '*' status.procs
salt '*' status.uptime
salt '*' status.vmstats
salt '*' status.w
```

```
salt '*' supervisord.add <name>
salt '*' supervisord.custom "mstop '*gunicorn*'"
salt '*' supervisord.remove <name>
salt '*' supervisord.reread
salt '*' supervisord.restart <service>
salt '*' supervisord.start <service>
salt '*' supervisord.status
salt '*' supervisord.status_raw
salt '*' supervisord.stop <service>
salt '*' supervisord.update
salt '*' sys.argspec pkg.install
salt '*' sys.argspec sys
salt '*' sys.argspec
salt '*' sys.doc
salt '*' sys.doc sys
salt '*' sys.doc sys.doc
salt '*' sys.doc network.traceroute user.info
salt '*' sys.list_functions
salt '*' sys.list_functions sys
salt '*' sys.list_functions sys user
salt '*' sys.list_modules
salt '*' sys.reload_modules
salt '*' sysctl.assign net.ipv4.ip_forward 1
salt '*' sysctl.get net.ipv4.ip_forward
salt '*' sysctl.persist net.ipv4.ip_forward 1
salt '*' sysctl.show
salt '*' system.halt
salt '*' system.init 3
salt '*' system.poweroff
salt '*' system.reboot
salt '*' system.shutdown
salt '*' test.arg 1 "two" 3.1 txt="hello" wow='{a: 1, b:
"hello"}'
salt '*' test.arg_repr 1 "two" 3.1 txt="hello" wow='{a: 1, b:
"hello"}'
salt '*' test.collatz 3
salt '*' test.conf_test
salt '*' test.cross_test file.gid_to_group 0
salt '*' test.echo 'foo bar baz quo qux'
salt '*' test.fib 3
salt '*' test.get_opts
salt '*' test.kwarg num=1 txt="two" env='{a: 1, b: "hello"}'
salt '*' test.not_loaded
salt '*' test.outputter foobar
salt '*' test.ping
```

```
salt '*' test.provider service
salt '*' test.providers
salt '*' test.rand_sleep 60
salt '*' test.retcode 42
salt '*' test.sleep 20
salt '*' test.tty tty0 'This is a test'
salt '*' test.tty pts3 'This is a test'
salt '*' test.version
salt '*' test.versions_information
salt '*' test.versions_report
salt '*' timezone.get_hwclock
salt '*' timezone.get_offset
salt '*' timezone.get_zone
salt '*' timezone.get_zonecode
salt '*' timezone.set_hwclock UTC
salt '*' timezone.set_zone 'America/Denver'
salt '*' timezone.zone_compare 'America/Denver'
salt '*' user.add name <uid> <gid> <groups> <home> <shell>
salt '*' user.chfullname foo "Foo Bar"
salt '*' user.chgid foo 4376
salt '*' user.chgroups foo wheel,root True
salt '*' user.chhome foo /home/users/foo True
salt '*' user.chhomephone foo "7735551234"
salt '*' user.chroomnumber foo 123
salt '*' user.chshell foo /bin/zsh
salt '*' user.chuid foo 4376
salt '*' user.chworkphone foo "7735550123"
salt '*' user.delete name remove=True force=True
salt '*' user.getent
salt '*' user.info root
salt '*' user.list_groups foo
salt '*' user.list_users
salt '*' virtualenv.create /path/to/new/virtualenv
```

# 4. /etc/salt/master

## 4.1. File Server settings

编辑 /etc/salt/master 文件

```
file_roots:
  base:
    - /srv/salt
```

```
mkdir /srv/salt
```

## 4.2. Pillar settings

```
pillar_roots:
  base:
    - /srv/pillar
```

```
mkdir -p /srv/pillar
```

## 4.3. Node Groups

```
nodegroups:
  group1: 'L@foo.domain.com,bar.domain.com,baz.domain.com and
bl*.domain.com'
  group2: 'G@os:Debian and foo.domain.com'
```

## 4.4. File Server Backend

```
fileserver_backend:
  - roots
```

# 5. sls 脚本

安装 vim 语法加亮插件

```
mkdir ~/.vim/
git clone https://github.com/saltstack/salt-vim.git
mv salt-vim/{ftdetect,ftplugin,syntax} ~/.vim/
```

配置.vimrc文件

```
cat >> ~/.vimrc <<EOF
set nocompatible
filetype plugin indent on

set nocompatible
set tabstop=2
set shiftwidth=2
set expandtab
EOF
```

## 5.1. pkg

pkg 负责包的安装与卸载

```
rsync:
  pkg:
    - installed
```

## 5.2. service

service 负责管理服务脚本的启用，禁用，启动，停止，重启等等工作

```
service:
  - name: rsync
  - running
  - enable: True
```

# 6. FAQ

## 6.1. Git fileserver backend is enabled in configuration but could not be loaded, is git-python installed

```
2013-09-03 11:11:18,101 [salt.loaded.int.pillar.git_pillar
][ERROR   ] Git fileserver backend is enabled in configuration
but could not be loaded, is git-python installed?
```

编辑 /etc/salt/master 配置如下

```
fileserver_backend:
  - roots
```

# 第 28 章 Chef

http://www.opscode.com/chef/

# 1. 安装 Chef

## 1.1. CentOS

---

# 第 29 章 Cobbler

http://cobbler.github.com/

# 第 30 章 Cfengine

http://www.cfengine.org/

# 第 31 章 func

https://fedorahosted.org/func/

# 第 32 章 (R)?ex Deployment & Configuration Management

http://rexify.org/

# 第 33 章 基于Web的系统管理软件

***web-based interface for system administration***

# 1. Webmin

网站

http://www.webmin.com/

过程 33.1. Webmin 安装步骤:

1. Debian Package
2. 命令:

   sudo dpkg --install webmin_1.380_all.deb

   sudo apt-get install perl libnet-ssleay-perl openssl libauthen-pam-perl libpam-runtime libio-pty-perl libmd5-perl

   Webmin install complete. You can now login to https://netkiller.8800.org:10000/ as root with your root password, or as any user who can use sudo to run commands as root.

3. script

   Usage: /etc/init.d/webmin { start | stop }

4. nmap localhost

## 1.1. webalizer

```
# apt-get install webmin-webalizer
```

## 2. ajenti

http://ajenti.org/