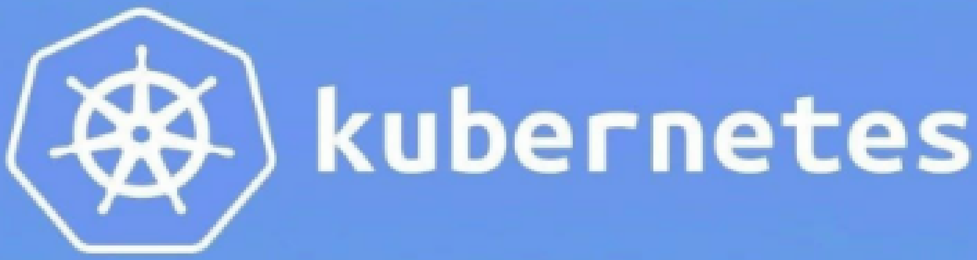
# Netkiller Virtualization 手札

陈景峰 著



知乎专栏 https://www.zhihu.com/column/netkiller

# Netkiller Container 手札

目录

范例清单

# Netkiller Container 手札

## Virtualization、Docker、Kubernetes、KVM、Vagrant、OpenVZ、VirtualBox ...

ISBN#

**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>

2015-07-14

电子书最近一次更新于 2023-04-05 18:45:23

Netkiller 手札系列电子书 http://www.netkiller.cn
Netkiller Virtualization 手札
陈景峰 著
kubernetes
知乎专栏 https://www.zhihu.com/column/netkiller

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 ”

$Date$

---

# 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 Virtualization 手札
陈景峰 著
kubernetes
知乎专栏 https://www.zhihu.com/column/netkiller

《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 (ㄔㄣ ㄐㄧㄥ 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 在线安裝电子书

[http://netkiller.sourceforge.net/pub/repo/](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

# 第 1 章 Docker

https://www.docker.com

## 1. 安装 Docker

### 1.1. Rocky Linux 9.0 / AlmiLinux 9.0 / CentOS 8 Stream

安装 Docker

```
[root@netkiller ~]# dnf config-manager --add-
repo=https://download.docker.com/linux/centos/docker-ce.repo
Adding repo from:
https://download.docker.com/linux/centos/docker-ce.repo

[root@netkiller ~]# dnf install -y docker-ce docker-compose-
plugin

[root@netkiller ~]# systemctl enable docker
[root@netkiller ~]# systemctl start docker
```

```
[root@netkiller ~]# docker -v
Docker version 19.03.12, build 48a66213fe
```

添加容器管理员

```
GID=$(egrep -o 'docker:x:([0-9]+)' /etc/group | egrep -o '([0-
9]+)')
```

```
adduser -u ${GID} -g ${GID} -G wheel -c "Container
Administrator" docker
```

```
[root@netkiller ~]# id docker
uid=986(docker) gid=986(docker) groups=986(docker),10(wheel)
```

配置 sudo 无需密码

```
cat > /etc/sudoers.d/docker <<-EOF
docker    ALL=(ALL)    NOPASSWD: ALL
EOF
```

检查 sudo 是否工作正常

```
[root@netkiller ~]# su - docker
Last login: Mon Mar 21 15:43:39 CST 2022 on pts/3

[docker@netkiller ~]$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS
NAMES

[docker@iZt4nazp2u494r8p1dr1zdZ ~]$ sudo ls /sbin
```

### docker-compose 2.x

正常情况使用 docker-compose-plugin 安装

```
[root@netkiller ~]# dnf install -y docker-compose-plugin
```

如需手工安装

```
DOCKER_CONFIG=${DOCKER_CONFIG:-$HOME/.docker}
mkdir -p $DOCKER_CONFIG/cli-plugins
curl -SL
https://github.com/docker/compose/releases/download/v2.2.3/dock
er-compose-linux-x86_64 -o $DOCKER_CONFIG/cli-plugins/docker-
compose
chmod +x $DOCKER_CONFIG/cli-plugins/docker-compose
```

使用 docker compose version 命令查看版本好，确认 docker compose 被成功安装

```
[root@netkiller ~]# docker compose version
Docker Compose version v2.6.0

[root@netkiller ~]# alias docker-compose='docker compose'
[root@netkiller ~]# docker-compose version
Docker Compose version v2.6.0
```

### 切换镜像

```
[root@netkiller ~]# cat << EOF > /etc/docker/daemon.json
>
> {
>   "registry-mirrors": [
>     "https://hub-mirror.c.163.com",
>     "https://mirror.baidubce.com",
>     "https://docker.mirrors.ustc.edu.cn/"
>   ]
> }
>
> EOF

[root@netkiller ~]# cat /etc/docker/daemon.json

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

[root@netkiller ~]# systemctl restart docker

[root@netkiller ~]# docker info
Client:
 Context:    default
 Debug Mode: false
 Plugins:
  app: Docker App (Docker Inc., v0.9.1-beta3)
  buildx: Build with BuildKit (Docker Inc., v0.5.1-docker)
  scan: Docker Scan (Docker Inc., v0.8.0)

Server:
 Containers: 0
  Running: 0
  Paused: 0
  Stopped: 0
 Images: 0
 Server Version: 20.10.7
 Storage Driver: overlay2
  Backing Filesystem: xfs
  Supports d_type: true
  Native Overlay Diff: true
  userxattr: false
 Logging Driver: json-file
 Cgroup Driver: cgroupfs
 Cgroup Version: 1
 Plugins:
  Volume: local
  Network: bridge host ipvlan macvlan null overlay
  Log: awslogs fluentd gcplogs gelf journald json-file local
logentries splunk syslog
 Swarm: inactive
 Runtimes: io.containerd.runc.v2 io.containerd.runtime.v1.linux
runc
 Default Runtime: runc
 Init Binary: docker-init
 containerd version: e25210fe30a0a703442421b0f60afac609f950a3
 runc version: v1.0.1-0-g4144b63
 init version: de40ad0
 Security Options:
  seccomp
   Profile: default
 Kernel Version: 4.18.0-326.el8.x86_64
 Operating System: CentOS Stream 8
 OSType: linux
 Architecture: x86_64
 CPUs: 4
 Total Memory: 7.514GiB
 Name: netkiller
 ID:
5GBU:CMWS:VIVP:TREZ:Y5AP:OGOW:EABK:NP4R:AWUA:S4J2:2YQ2:U7MT
 Docker Root Dir: /var/lib/docker
 Debug Mode: false
 Registry: https://index.docker.io/v1/
 Labels:
 Experimental: false
 Insecure Registries:
  127.0.0.0/8
 Registry Mirrors:
  https://hub-mirror.c.163.com/
  https://mirror.baidubce.com/
  https://docker.mirrors.ustc.edu.cn/
 Live Restore Enabled: false
```

## 1.2. Ubuntu docker-ce

从官方网站获得最新社区版

```
#!/bin/bash

sudo apt update

sudo apt remove docker docker-engine docker.io containerd runc

sudo apt install \
    apt-transport-https \
    ca-certificates \
    curl \
    gnupg \
    lsb-release

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo
gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg

echo \
  "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-
archive-keyring.gpg] https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) stable" | sudo tee
/etc/apt/sources.list.d/docker.list > /dev/null


<!-- sudo add-apt-repository \
   "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
   $(lsb_release -cs) \
   stable" -->

sudo apt update
sudo apt install docker-ce docker-ce-cli containerd.io

apt-cache madison docker-ce
```

查看 docker 运行状态

```
root@production:~# systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled;
vendor preset: enabled)
   Active: active (running) since Tue 2021-08-17 11:25:04 CST;
57s ago
     Docs: https://docs.docker.com
 Main PID: 7379 (dockerd)
   CGroup: /system.slice/docker.service
           └─7379 /usr/bin/dockerd -H fd:// --
containerd=/run/containerd/containerd.sock

Aug 17 11:25:04 production dockerd[7379]: time="2021-08-
17T11:25:04.708262132+08:00" level=info msg="ClientConn
switching balancer to \"pick_first\"" module=grpc
Aug 17 11:25:04 production dockerd[7379]: time="2021-08-
17T11:25:04.742384618+08:00" level=warning msg="Your kernel
does not support swap memory limit"
Aug 17 11:25:04 production dockerd[7379]: time="2021-08-
17T11:25:04.742397707+08:00" level=warning msg="Your kernel
does not support CPU realtime scheduler"
Aug 17 11:25:04 production dockerd[7379]: time="2021-08-
17T11:25:04.742489785+08:00" level=info msg="Loading
containers: start."
Aug 17 11:25:04 production dockerd[7379]: time="2021-08-
17T11:25:04.811316570+08:00" level=info msg="Default bridge
(docker0) is assigned with an IP address 172.18.0.0/16. Daemon
option --bip can be used
Aug 17 11:25:04 production dockerd[7379]: time="2021-08-
17T11:25:04.836024290+08:00" level=info msg="Loading
containers: done."
Aug 17 11:25:04 production dockerd[7379]: time="2021-08-
17T11:25:04.858428922+08:00" level=info msg="Docker daemon"
commit=b0f5bc3 graphdriver(s)=overlay2 version=20.10.7
Aug 17 11:25:04 production dockerd[7379]: time="2021-08-
17T11:25:04.858470910+08:00" level=info msg="Daemon has
completed initialization"
Aug 17 11:25:04 production systemd[1]: Started Docker
Application Container Engine.
Aug 17 11:25:04 production dockerd[7379]: time="2021-08-
17T11:25:04.875279830+08:00" level=info msg="API listen on
/var/run/docker.sock"
```

启动参数配置 /etc/default/docker

```
neo@ubuntu:~$ cat /etc/default/docker
# Docker Upstart and SysVinit configuration file

#
# THIS FILE DOES NOT APPLY TO SYSTEMD
#
#   Please see the documentation for "systemd drop-ins":
#   https://docs.docker.com/engine/admin/systemd/
#

# Customize location of Docker binary (especially for
development testing).
#DOCKERD="/usr/local/bin/dockerd"

# Use DOCKER_OPTS to modify the daemon startup options.
#DOCKER_OPTS="--dns 8.8.8.8 --dns 8.8.4.4"

# If you need Docker to use an HTTP proxy, it can also be
specified here.
#export http_proxy="http://127.0.0.1:3128/"

# This is also a handy place to tweak where Docker's temporary
files go.
#export DOCKER_TMPDIR="/mnt/bigdrive/docker-tmp"
```

启动脚本 /etc/init/docker.conf

```
neo@ubuntu:~$ sudo cat /etc/init/docker.conf
[sudo] password for neo:
description "Docker daemon"

start on (filesystem and net-device-up IFACE!=lo)
stop on runlevel [!2345]

limit nofile 524288 1048576

# Having non-zero limits causes performance problems due to
```

```
accounting overhead
# in the kernel. We recommend using cgroups to do container-
local accounting.
limit nproc unlimited unlimited

respawn

kill timeout 20

pre-start script
        # see also https://github.com/tianon/cgroupfs-
mount/blob/master/cgroupfs-mount
        if grep -v '^#' /etc/fstab | grep -q cgroup \
                || [ ! -e /proc/cgroups ] \
                || [ ! -d /sys/fs/cgroup ]; then
                exit 0
        fi
        if ! mountpoint -q /sys/fs/cgroup; then
                mount -t tmpfs -o uid=0,gid=0,mode=0755 cgroup
/sys/fs/cgroup
        fi
        (
                cd /sys/fs/cgroup
                for sys in $(awk '!/^#/ { if ($4 == 1) print $1
}' /proc/cgroups); do
                        mkdir -p $sys
                        if ! mountpoint -q $sys; then
                                if ! mount -n -t cgroup -o $sys
cgroup $sys; then
                                        rmdir $sys || true
                                fi
                        fi
                done
        )
end script

script
        # modify these in /etc/default/$UPSTART_JOB
(/etc/default/docker)
        DOCKERD=/usr/bin/dockerd
        DOCKER_OPTS=
        if [ -f /etc/default/$UPSTART_JOB ]; then
                . /etc/default/$UPSTART_JOB
        fi
        exec "$DOCKERD" $DOCKER_OPTS --raw-logs
```

```
end script

# Don't emit "started" event until docker.sock is ready.
# See https://github.com/docker/docker/issues/6647
post-start script
        DOCKER_OPTS=
        DOCKER_SOCKET=
        if [ -f /etc/default/$UPSTART_JOB ]; then
                . /etc/default/$UPSTART_JOB
        fi

        if ! printf "%s" "$DOCKER_OPTS" | grep -qE -e '-H|--
host'; then
                DOCKER_SOCKET=/var/run/docker.sock
        else
                DOCKER_SOCKET=$(printf "%s" "$DOCKER_OPTS" |
grep -oP -e '(-H|--host)\W*unix://\K(\S+)' | sed 1q)
        fi

        if [ -n "$DOCKER_SOCKET" ]; then
                while ! [ -e "$DOCKER_SOCKET" ]; do
                        initctl status $UPSTART_JOB | grep -qE
"(stop|respawn)/" && exit 1
                        echo "Waiting for $DOCKER_SOCKET"
                        sleep 0.1
                done
                echo "$DOCKER_SOCKET is up"
        fi
end script
```

### 1.3. 测试 Docker

```
neo@MacBook-Pro ~ % docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
1b930d010525: Pull complete
Digest:
sha256:2557e3c07ed1e38f26e389462d03ed943586f744621577a99efb7732
4b0fe535
```

```
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working
correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the
Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image
which runs the
    executable that produces the output you are currently
reading.
 4. The Docker daemon streamed that output to the Docker
client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu
container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker
ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/

neo@MacBook-Pro ~ % docker image ls
REPOSITORY                                  TAG
IMAGE ID           CREATED             SIZE
hello-world                                 latest
fce289e99eb9       2 months ago        1.84kB

neo@MacBook-Pro ~ % docker container ls --all
CONTAINER ID       IMAGE               COMMAND
CREATED             STATUS                          PORTS
NAMES
ea694b443e9e       hello-world         "/hello"
About a minute ago   Exited (0) About a minute ago
dreamy_feistel
```

## 1.4. 重置 Docker

```
docker stop $(docker ps -a -q)
docker rm -f $(docker ps -a -q)
docker rmi -f $(docker images -q)
docker volume rm $(docker volume ls -q)
```

## 1.5. 早起版本

### CentOS 7 docker-ce

下载 containerd.io https://download.docker.com/linux/centos/7/x86_64/stable/Packages/

```
[root@netkiller ~]# yum install
https://download.docker.com/linux/centos/7/x86_64/stable/Packag
es/containerd.io-1.2.13-3.2.el7.x86_64.rpm
```

从官方网站获得最新社区版

```
yum install -y yum-utils
yum-config-manager --add-repo
https://download.docker.com/linux/centos/docker-ce.repo
yum makecache fast
yum -y install docker-ce

systemctl start docker
```

测试安装是否成功

```
docker run hello-world
```

**CentOS 6**

```
yum install docker-io
service docker start
chkconfig docker on
docker pull centos:latest
docker images centos
```

test

```
docker run -i -t centos /bin/bash
```

**Ubuntu**

Ubuntu 默认版本

```
$ sudo apt update
$ sudo apt install docker.io
$ sudo ln -sf /usr/bin/docker.io /usr/local/bin/docker
$ sudo sed -i '$acomplete -F _docker docker'
```

```
/etc/bash_completion.d/docker.io
```

```
$ sudo docker run -i -t ubuntu /bin/bash
```

# 2. Portainer - Docker 图形管理界面

Portainer 是一个轻量级的 Docker 管理界面，官方提供了 Demo 演示地址

## 2.1. 安装

Server 服务器安装

```
docker volume create portainer_data
docker run -d -p 8000:8000 -p 9000:9000 --name=portainer --restart=always -v
/var/run/docker.sock:/var/run/docker.sock -v portainer_data:/data
portainer/portainer-ce
```

Agent 代理安装

```
docker run -d -p 9001:9001 --name portainer_agent --restart=always -v
/var/run/docker.sock:/var/run/docker.sock -v
/var/lib/docker/volumes:/var/lib/docker/volumes portainer/agent
```

使用 docker-compose 安装

```
version: '3.9'

services:
  portainer:
    image: portainer/portainer-ce
    container_name: prtainer
    restart: always
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
      - portainter:/data
    ports:
      - 8000:8000
      - 9000:9000

  portainer-agent:
    image: portainer/agent
    container_name: portainer-agent
    restart: always
    volumes:
        - /var/run/docker.sock:/var/run/docker.sock
        - /var/lib/docker/volumes:/var/lib/docker/volumes
    ports:
        - 9001:9001

volumes:
  portainter:
```

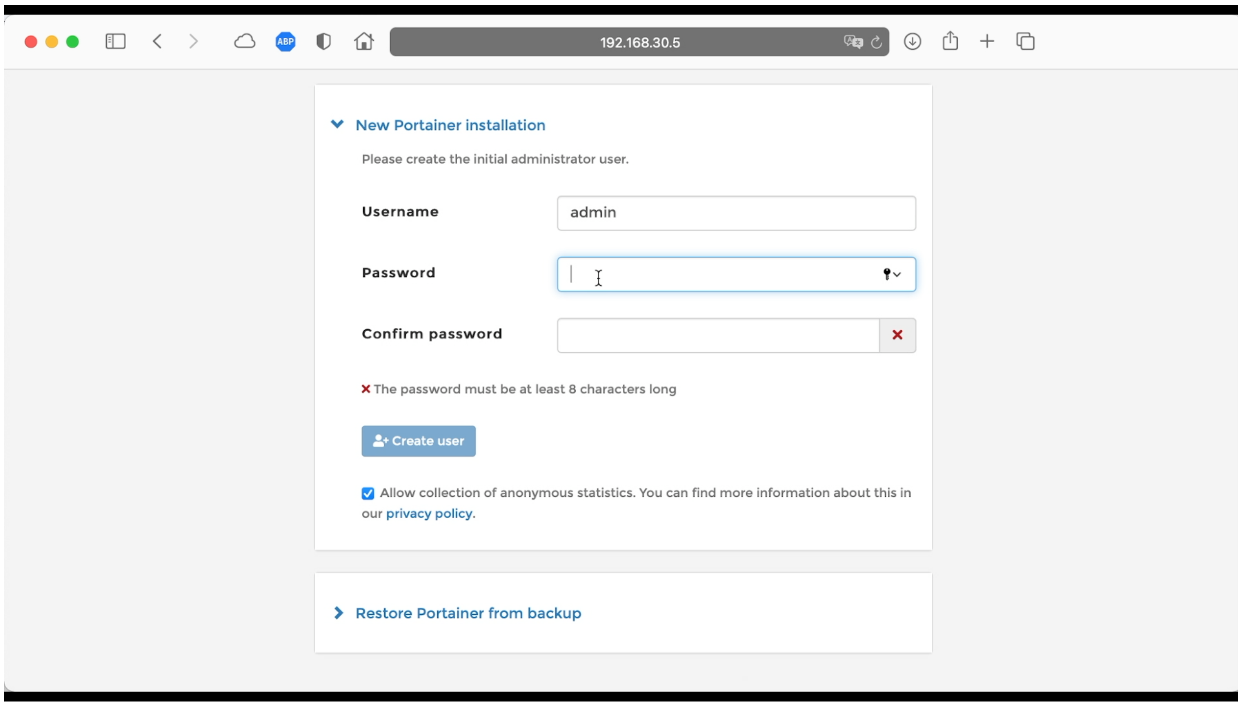
```
第一台管理服务器，启动管理界面：
[root@netkiller portainer]# docker-compose up -d portainer

第二台开发环境服务器，启动代理：
[root@development portainer]# docker-compose up -d portainer-agent

第三台测试环境服务器，启动代理
[root@testing portainer]# docker-compose up -d portainer-agent
```

## 2.2. 配置 Portainer

设置管理员密码，创建用户



Please create the initial administrator user.

Username: admin

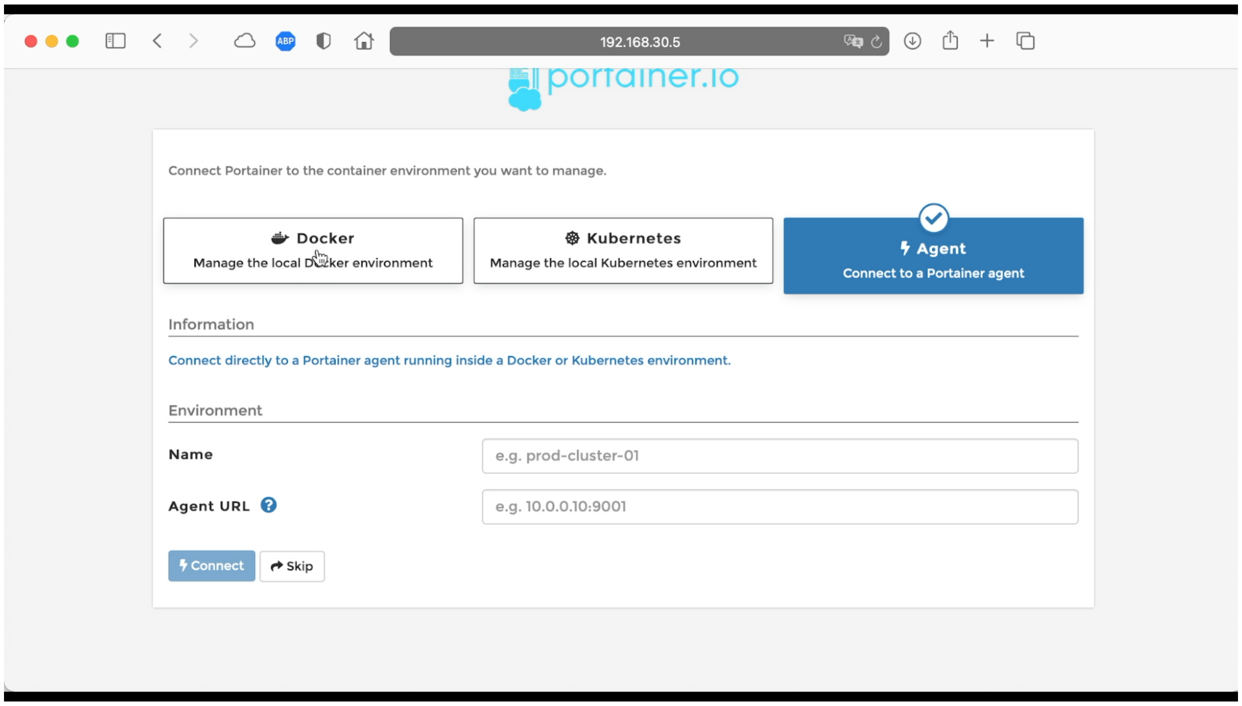
Password:

Confirm password:

✕ The password must be at least 8 characters long

✓ Allow collection of anonymous statistics. You can find more information about this in our privacy policy.

当前界面中有三个选项，分别是 Docker（本地 Docker），Kubernetes, Agent(代理)


Connect Portainer to the container environment you want to manage.

Docker
Manage the local Docker environment
Kubernetes
Manage the local Kubernetes environment
Agent
Connect to a Portainer agent

Information

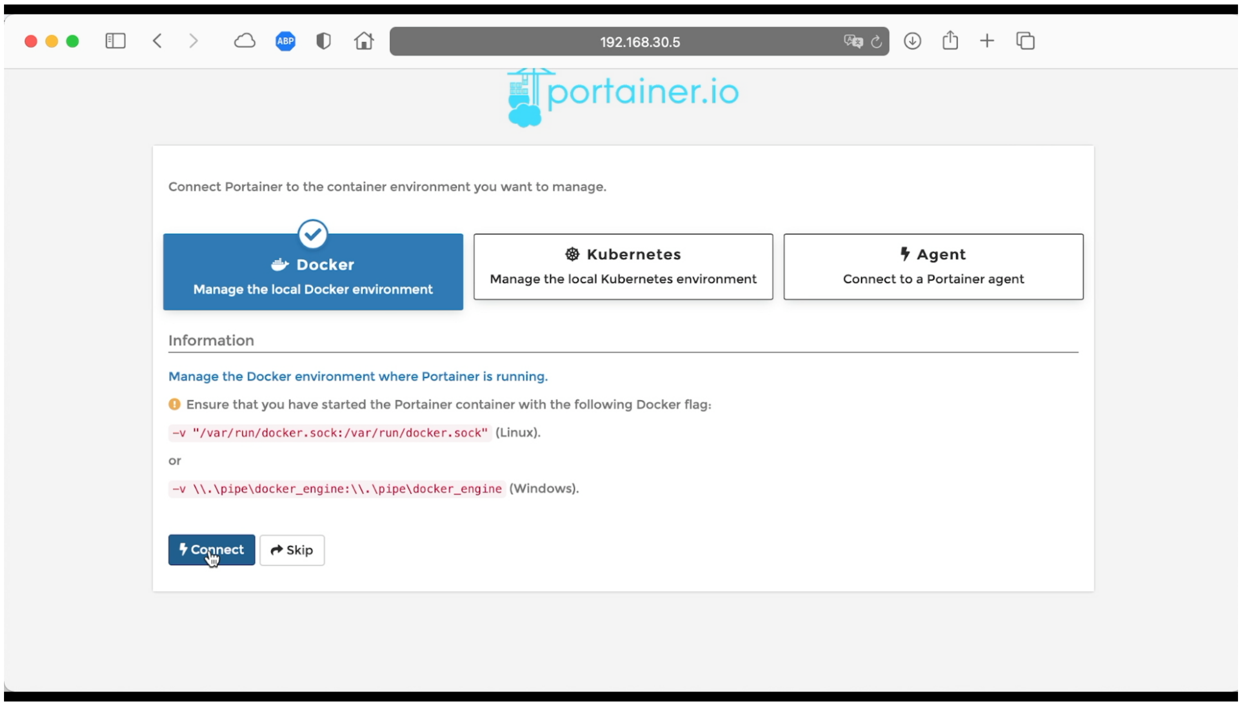
Connect directly to a Portainer agent running inside a Docker or Kubernetes environment.

Environment

Name: e.g. prod-cluster-01

Agent URL: e.g. 10.0.0.10:9001

添加本地 Docker，通过 UNIX SOCK 链接，通常是 /var/run/docker.sock


Connect Portainer to the container environment you want to manage.

Docker
Manage the local Docker environment
Kubernetes
Manage the local Kubernetes environment
Agent
Connect to a Portainer agent

Information

Manage the Docker environment where Portainer is running.

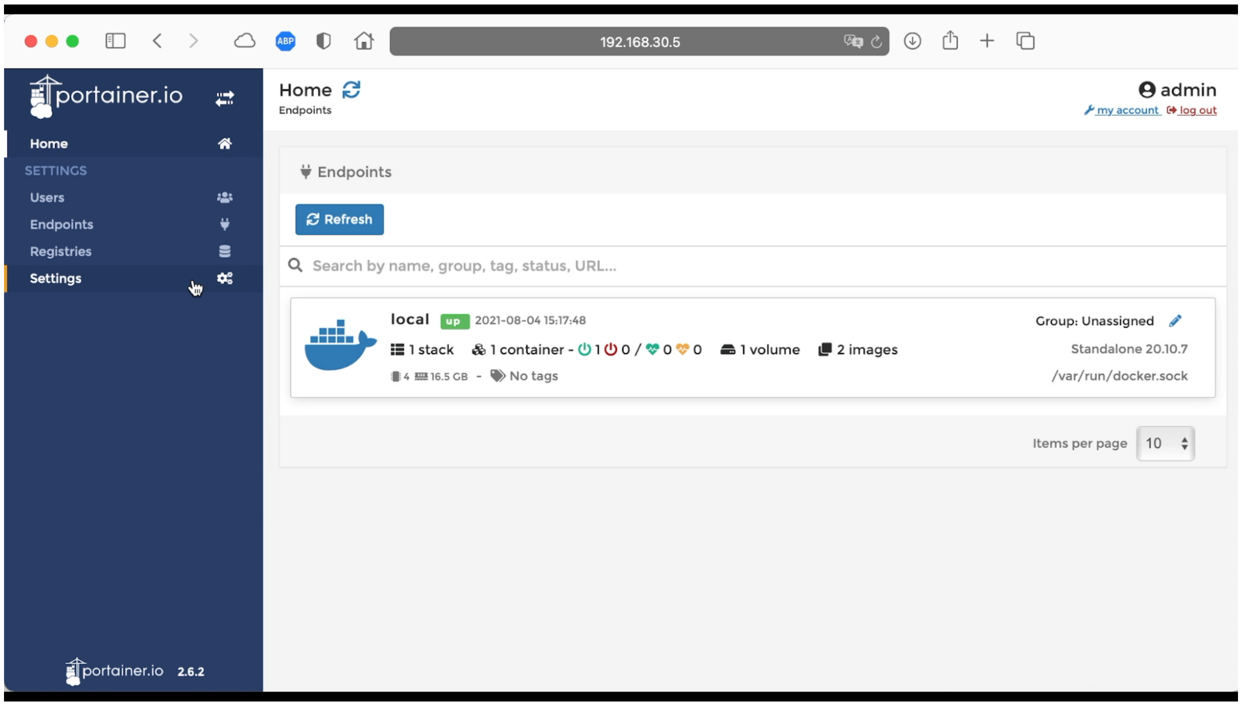
Ensure that you have started the Portainer container with the following Docker flag:

`-v "/var/run/docker.sock:/var/run/docker.sock"` (Linux).

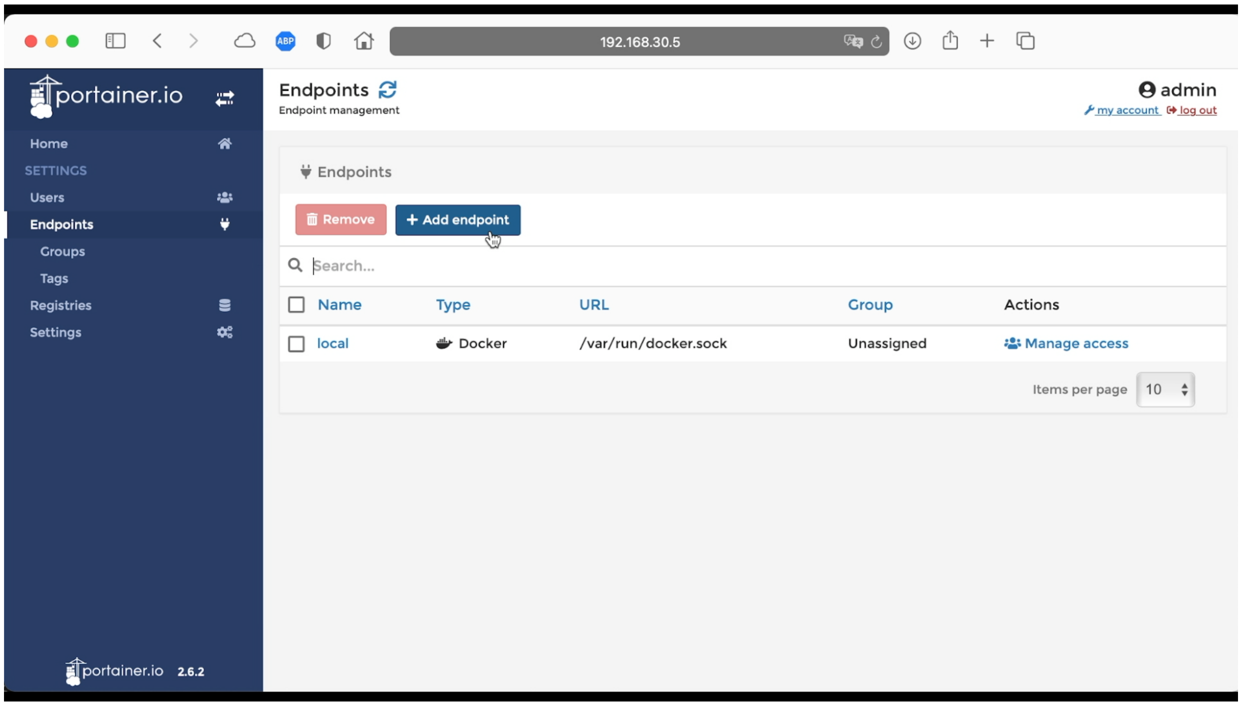
or

`-v \\.\pipe\docker_engine:\\.\pipe\docker_engine` (Windows).

点击 Connect 按钮就可以建立链接

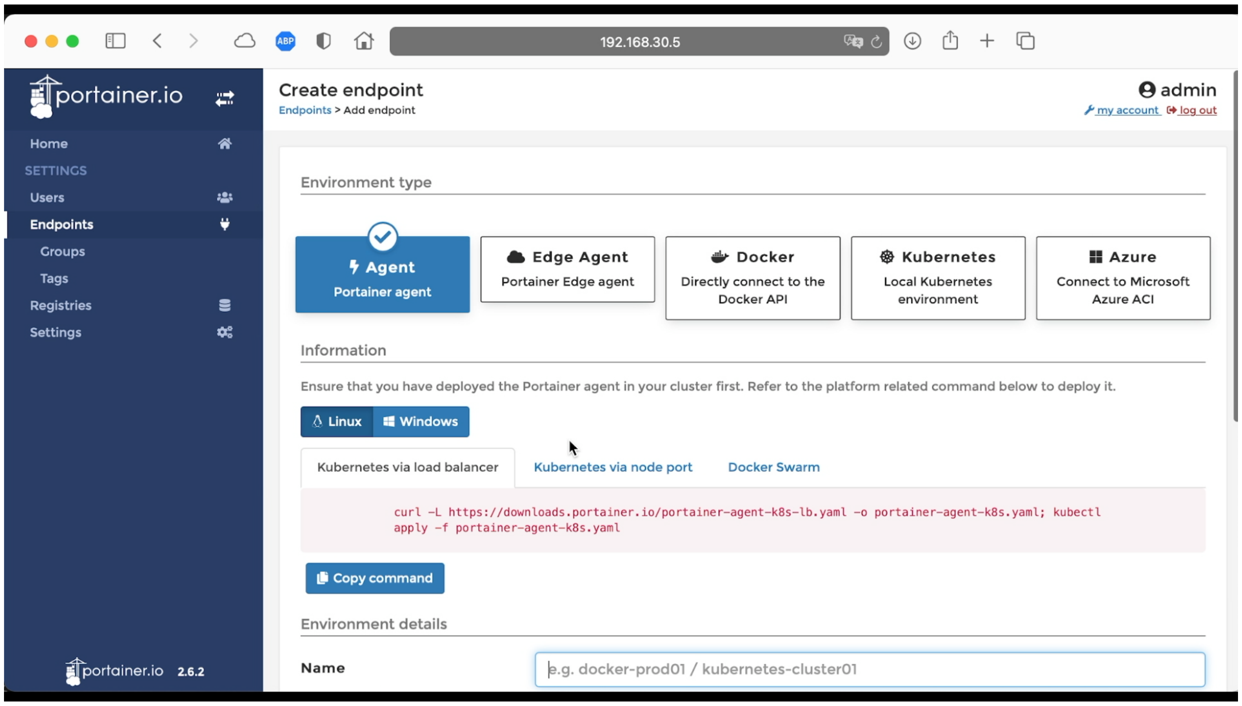


添加代理 Docker，左边菜单点击 Endpoints，然后点击 Add endpoint



| Name | Type | URL | Group | Actions |
|---|---|---|---|---|
| local | Docker | /var/run/docker.sock | Unassigned | Manage access |

选择 Agent



Create endpoint

Endpoints > Add endpoint

Environment type

- **Agent** — Portainer agent
- **Edge Agent** — Portainer Edge agent
- **Docker** — Directly connect to the Docker API
- **Kubernetes** — Local Kubernetes environment
- **Azure** — Connect to Microsoft Azure ACI

Information

Ensure that you have deployed the Portainer agent in your cluster first. Refer to the platform related command below to deploy it.

Linux | Windows

Kubernetes via load balancer | Kubernetes via node port | Docker Swarm

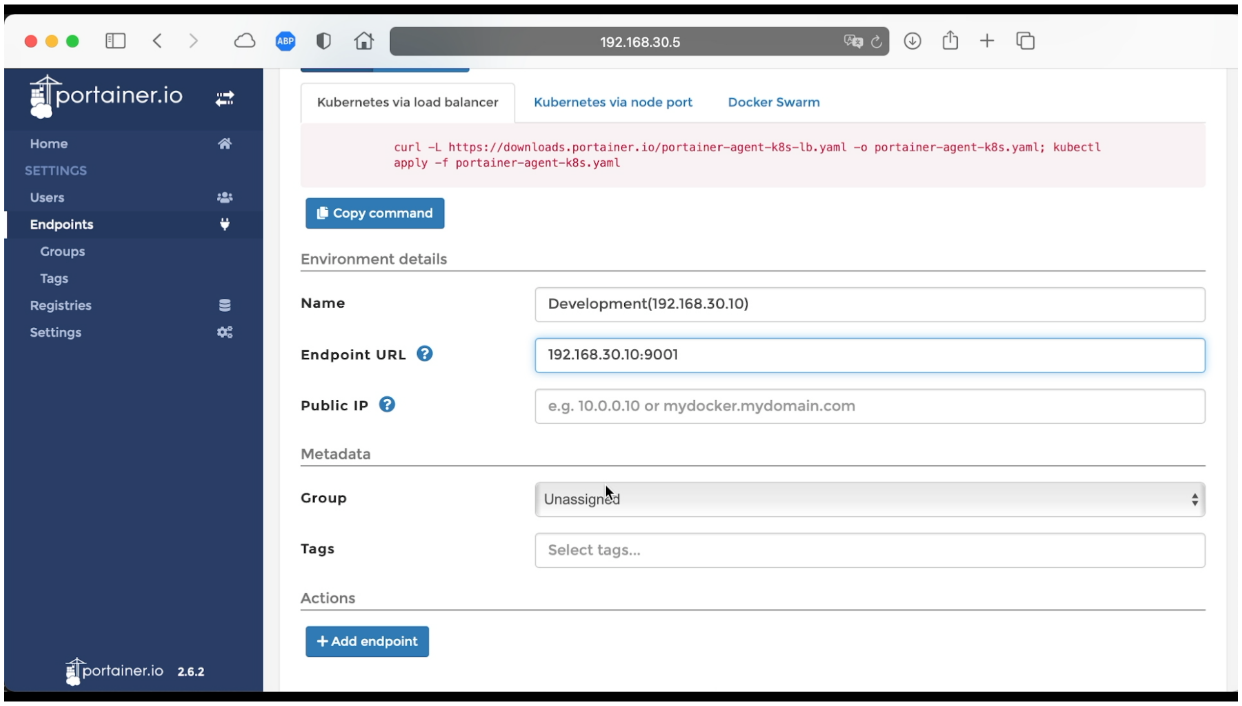
```
curl -L https://downloads.portainer.io/portainer-agent-k8s-lb.yaml -o portainer-agent-k8s.yaml; kubectl
apply -f portainer-agent-k8s.yaml
```

Copy command

Environment details

Name: e.g. docker-prod01 / kubernetes-cluster01

Name 给代理起个名气，Endpoint URL 输入代理的IP地址和端口号，Group 是分组（可不选），最后点击 Add endpoint 按钮。



Kubernetes via load balancer | Kubernetes via node port | Docker Swarm

```
curl -L https://downloads.portainer.io/portainer-agent-k8s-lb.yaml -o portainer-agent-k8s.yaml; kubectl
apply -f portainer-agent-k8s.yaml
```

Copy command

Environment details

| | |
|---|---|
| Name | Development(192.168.30.10) |
| Endpoint URL | 192.168.30.10:9001 |
| Public IP | e.g. 10.0.0.10 or mydocker.mydomain.com |

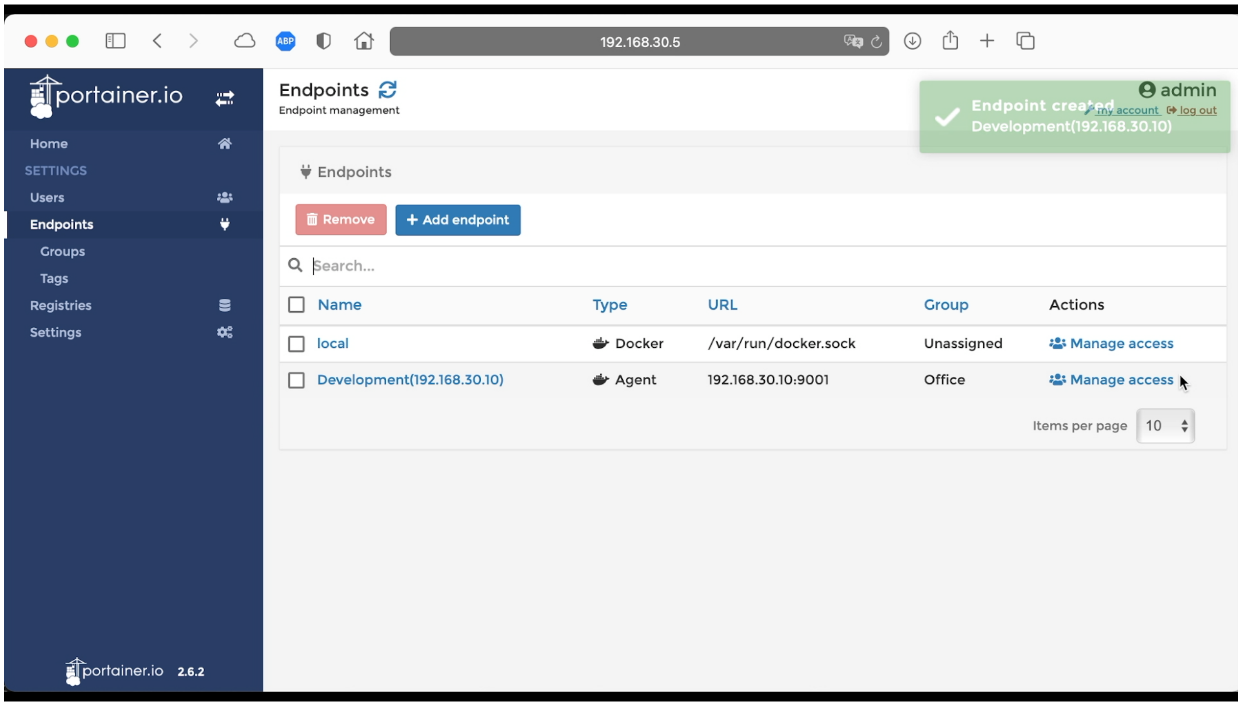
Metadata

| | |
|---|---|
| Group | Unassigned |
| Tags | Select tags... |

Actions

+ Add endpoint

完成代理的添加



Endpoints
Endpoint management

Endpoint created
Development(192.168.30.10)

Endpoints

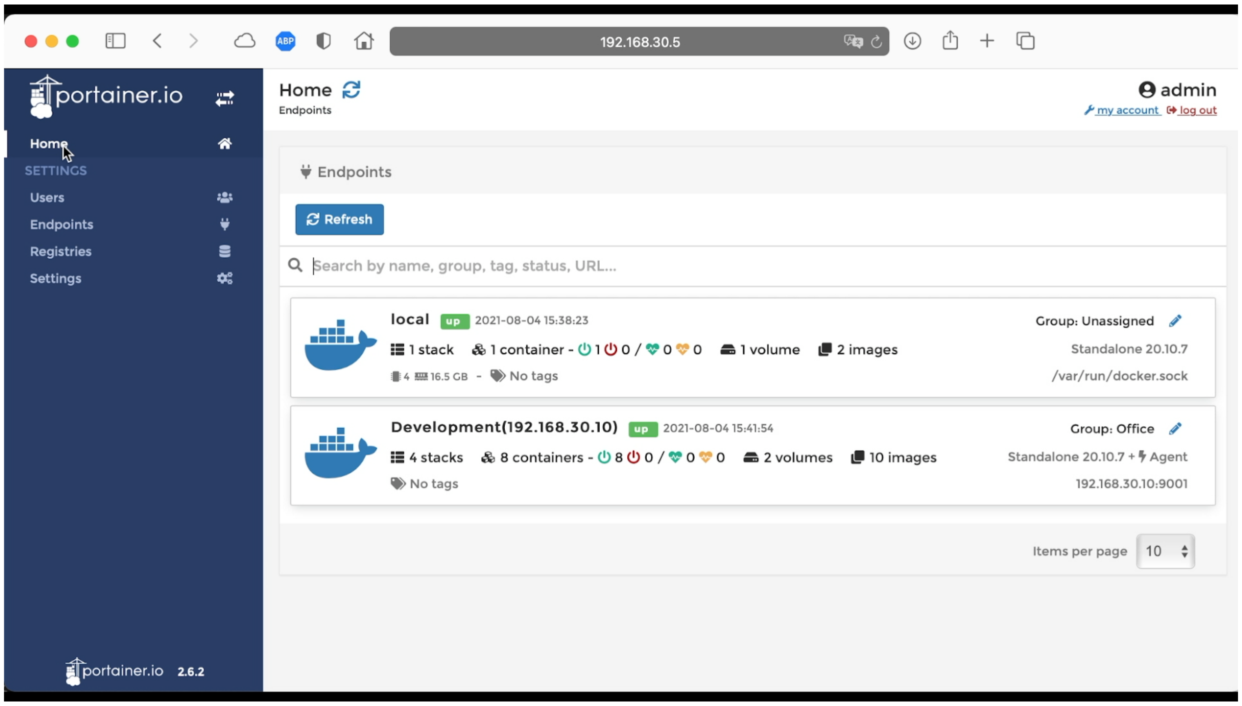
Remove | Add endpoint

Search...

| Name | Type | URL | Group | Actions |
|---|---|---|---|---|
| local | Docker | /var/run/docker.sock | Unassigned | Manage access |
| Development(192.168.30.10) | Agent | 192.168.30.10:9001 | Office | Manage access |

Items per page 10

回到 Home



Home
Endpoints

Endpoints

Refresh

Search by name, group, tag, status, URL...

local up 2021-08-04 15:38:23 Group: Unassigned
1 stack 1 container - 1 0 / 0 0 1 volume 2 images Standalone 20.10.7
4 16.5 GB - No tags /var/run/docker.sock

Development(192.168.30.10) up 2021-08-04 15:41:54 Group: Office
4 stacks 8 containers - 8 0 / 0 0 2 volumes 10 images Standalone 20.10.7 + Agent
No tags 192.168.30.10:9001

Items per page 10

## 2.3. 添加代理出错

portainer 错误日志

```
portainer          | 2021/08/04 07:24:14 http error: Unable to initiate
communications with endpoint (err=agent already paired with another Portainer
instance) (code=500)
portainer          | 2021/08/04 07:25:49 http error: Unable to initiate
communications with endpoint (err=agent already paired with another Portainer
instance) (code=500)
```

agent 日志

```
portainer-agent    | 2021/08/04 07:25:49 http error: Invalid request signature
(err=Unauthorized) (code=403)
portainer-agent    | 2021/08/04 07:25:49 http error: Invalid request signature
(err=Unauthorized) (code=403)
```

问题出在，重装了 portainer 先前的 agent 已经与之前的 portainer 建立链接。

解决方法，重装 agent 记得要删除卷。

```
[root@testing portainer]# docker-compose stop portainer-agent
Stopping portainer-agent ... done

[root@testing portainer]# docker-compose rm -a  portainer-agent
WARNING: --all flag is obsolete. This is now the default behavior of `docker-
compose rm`
Going to remove portainer-agent
Are you sure? [yN] y
Removing portainer-agent ... done

[root@testing portainer]# docker volume ls
DRIVER    VOLUME NAME
local     portainer_portainter

[root@testing portainer]# docker volume rm portainer_portainter
portainer_portainter

[root@testing portainer]# docker-compose up -d portainer-agent
Creating volume "portainer_portainter" with default driver
Creating portainer-agent ... done

[root@testing portainer]# docker-compose ps
     Name         Command   State                   Ports
----------------------------------------------------------------------------
portainer-agent   ./agent   Up      0.0.0.0:9001->9001/tcp,:::9001->9001/tcp
```

# 3. 配置 Docker

## 3.1. 开启远程访问

修改/etc/sysconfig/docker文件，在最后增加一行DOCKER_OPTS

vim /etc/sysconfig/docker

```
DOCKER_OPTS="-H unix:///var/run/docker.sock -H
tcp://0.0.0.0:2375"
```

修改/usr/lib/systemd/system/docker.service 在[Service]的ExexStart=下面增加一行$DOCKER_OPTS

```
[Unit]
Description=Docker Application Container Engine
Documentation=https://docs.docker.com
BindsTo=containerd.service
After=network-online.target 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
EnvironmentFile=-/etc/sysconfig/docker
ExecStart=/usr/bin/dockerd $DOCKER_OPTS
ExecReload=/bin/kill -s HUP $MAINPID
TimeoutSec=0
RestartSec=2
```

```
Restart=always

# Note that StartLimit* options were moved from "Service" to
"Unit" in systemd 229.
# Both the old, and new location are accepted by systemd 229
and up, so using the old location
# to make them work for either version of systemd.
StartLimitBurst=3

# Note that StartLimitInterval was renamed to
StartLimitIntervalSec in systemd 230.
# Both the old, and new name are accepted by systemd 230 and
up, so using the old name to make
# this option work for either version of systemd.
StartLimitInterval=60s

# Having non-zero Limit*s causes performance problems due to
accounting overhead
# in the kernel. We recommend using cgroups to do container-
local accounting.
LimitNOFILE=infinity
LimitNPROC=infinity
LimitCORE=infinity

# Comment TasksMax if your systemd version does not supports
it.
# Only systemd 226 and above support this option.
TasksMax=infinity

# 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

[Install]
WantedBy=multi-user.target
```

重启 docker

```
[root@localhost ~]# systemctl daemon-reload
[root@localhost ~]# systemctl restart docker
```

### /etc/docker/daemon.json

编辑 /etc/docker/daemon.json 文件加入

```
{
  "hosts":[
    "unix:///var/run/docker.sock",
    "tcp://0.0.0.0:2375"
  ]
}
```

重启 docker

```
[root@localhost ~]# systemctl daemon-reload
[root@localhost ~]# systemctl restart docker
```

```
$ docker -H docker.netkiller.cn:2375 info
```

```
$ export DOCKER_HOST="tcp://docker.netkiller.cn:2375"
$ docker info
```

查看端口

```
[root@localhost ~]# ss -lnt | grep 2375
LISTEN     0      1024        :::2375                    :::*
```

检查 docker 信息

```
[root@localhost ~]# curl -s http://your-docker-ip-
address:2375/info
{"ID":"YNK5:OJTT:FELN:H4DQ:AG7H:W3RE:WGLD:TOOI:32CH:S6HR:AJ45:4
VLZ","Containers":4,"ContainersRunning":0,"ContainersPaused":0,
"ContainersStopped":4,"Images":10,"Driver":"btrfs","DriverStatu
s":[["Build Version","Btrfs v4.9.1"],["Library
Version","102"]],"SystemStatus":null,"Plugins":{"Volume":
["local"],"Network":
["bridge","host","macvlan","null","overlay"],"Authorization":nu
ll,"Log":
["awslogs","fluentd","gcplogs","gelf","journald","json-
file","local","logentries","splunk","syslog"]},"MemoryLimit":tr
ue,"SwapLimit":true,"KernelMemory":true,"CpuCfsPeriod":true,"Cp
uCfsQuota":true,"CPUShares":true,"CPUSet":true,"IPv4Forwarding"
:true,"BridgeNfIptables":false,"BridgeNfIp6tables":false,"Debug
":false,"NFd":23,"OomKillDisable":true,"NGoroutines":37,"System
Time":"2019-01-24T23:30:56.230913047-
05:00","LoggingDriver":"json-
file","CgroupDriver":"cgroupfs","NEventsListener":0,"KernelVers
ion":"3.10.0-693.el7.x86_64","OperatingSystem":"CentOS Linux 7
(Core)","OSType":"linux","Architecture":"x86_64","IndexServerAd
dress":"https://index.docker.io/v1/","RegistryConfig":
{"AllowNondistributableArtifactsCIDRs":
[],"AllowNondistributableArtifactsHostnames":
[],"InsecureRegistryCIDRs":["127.0.0.0/8"],"IndexConfigs":
{"docker.io":{"Name":"docker.io","Mirrors":
[],"Secure":true,"Official":true}},"Mirrors":
```

```
[]},"NCPU":2,"MemTotal":1958645760,"GenericResources":null,"Doc
kerRootDir":"/var/lib/docker","HttpProxy":"","HttpsProxy":"","N
oProxy":"","Name":"localhost.localdomain","Labels":
[],"ExperimentalBuild":false,"ServerVersion":"18.09.1","Cluster
Store":"","ClusterAdvertise":"","Runtimes":{"runc":
{"path":"runc"}},"DefaultRuntime":"runc","Swarm":
{"NodeID":"","NodeAddr":"","LocalNodeState":"inactive","Control
Available":false,"Error":"","RemoteManagers":null},"LiveRestore
Enabled":false,"Isolation":"","InitBinary":"docker-
init","ContainerdCommit":
{"ID":"9754871865f7fe2f4e74d43e2fc7ccd237edcbce","Expected":"97
54871865f7fe2f4e74d43e2fc7ccd237edcbce"},"RuncCommit":
{"ID":"96ec2177ae841256168fcf76954f7177af9446eb","Expected":"96
ec2177ae841256168fcf76954f7177af9446eb"},"InitCommit":
{"ID":"fec3683","Expected":"fec3683"},"SecurityOptions":
["name=seccomp,profile=default"],"ProductLicense":"Community
Engine","Warnings":["WARNING: API is accessible on
http://0.0.0.0:2375 without encryption.\n         Access to the
remote API is equivalent to root access on the host. Refer\n
to the 'Docker daemon attack surface' section in the
documentation for\n         more information:
https://docs.docker.com/engine/security/security/#docker-
daemon-attack-surface","WARNING: bridge-nf-call-iptables is
disabled","WARNING: bridge-nf-call-ip6tables is disabled"]}
```

```
$ docker -H 192.168.10.11:2375 info
```

```
DOCKER_HOST=tcp://192.168.57.110:2376
```

## 配置SSL证书

```
{
  "tlsverify": true,
  "tlscert": "/etc/docker/server-cert.pem",
  "tlskey": "/etc/docker/server-key.pem",
  "tlscacert": "/etc/docker/ca.pem",
  "hosts":[
    "unix:///var/run/docker.sock",
    "tcp://0.0.0.0:2376"
  ]
}
```

```
$ docker --tlsverify \
  --tlscacert=/Users/neo/test/ca.pem \
  --tlscert=/Users/neo/test/cert.pem \
  --tlskey=/Users/neo/test/key.pem \
  -H=192.168.57.110:2376 \
  info
```

我们可以把 ca.pem cert.pem key.pem 三个文件放入客户端 ~/.docker 中，然后配置环境变量就可以简化命令了

```
$ export DOCKER_HOST=tcp://192.168.5.10:2376
DOCKER_TLS_VERIFY=1
$ docker info
```

## 通过 SSH 连接远程 Docker

```
export DOCKER_HOST=ssh://docker-user@host1.example.com
```

```
Neo-iMac:Shell neo$ export DOCKER_HOST=ssh://root@192.168.30.11
Neo-iMac:Shell neo$ docker info
Client:
 Context:    default
 Debug Mode: false
 Plugins:
  buildx: Build with BuildKit (Docker Inc., v0.6.3)
  compose: Docker Compose (Docker Inc., v2.0.0)
  scan: Docker Scan (Docker Inc., v0.8.0)

Server:
 Containers: 9
  Running: 7
  Paused: 0
  Stopped: 2
 Images: 12
 Server Version: 20.10.10
 Storage Driver: overlay2
  Backing Filesystem: xfs
  Supports d_type: true
  Native Overlay Diff: true
  userxattr: false
 Logging Driver: json-file
 Cgroup Driver: cgroupfs
 Cgroup Version: 1
 Plugins:
  Volume: local
  Network: bridge host ipvlan macvlan null overlay
  Log: awslogs fluentd gcplogs gelf journald json-file local
logentries splunk syslog
 Swarm: inactive
 Runtimes: io.containerd.runc.v2 io.containerd.runtime.v1.linux
runc
 Default Runtime: runc
 Init Binary: docker-init
 containerd version: 5b46e404f6b9f661a205e28d59c982d3634148f8
 runc version: v1.0.2-0-g52b36a2
 init version: de40ad0
 Security Options:
  seccomp
   Profile: default
 Kernel Version: 4.18.0-348.el8.x86_64
 Operating System: CentOS Stream 8
 OSType: linux
 Architecture: x86_64
 CPUs: 4
 Total Memory: 15.39GiB
 Name: localhost.localdomain
 ID:
UODB:ETXF:35NV:DDSK:B5QU:RTNZ:7DM4:3ABZ:RZUB:SHOE:W6EP:UK4K
 Docker Root Dir: /var/lib/docker
 Debug Mode: false
 Registry: https://index.docker.io/v1/
 Labels:
 Experimental: false
 Insecure Registries:
  127.0.0.0/8
 Registry Mirrors:
  https://registry.docker-cn.com/
  http://hub-mirror.c.163.com/
  https://docker.mirrors.ustc.edu.cn/
 Live Restore Enabled: false
```

## 3.2. 镜像配置

### 临时选择镜像

您可以在 Docker 守护进程启动时传入 --registry-mirror 参数：

```
$ docker --registry-mirror=https://registry.docker-cn.com
daemon
```

### 切换国内镜像

设置默认镜像，修改 /etc/docker/daemon.json 文件，并添加上 registry-mirrors 键值。

Docker 中国官方镜像

```
{
  "registry-mirrors": ["https://registry.docker-cn.com"]
}
```

设置多个镜像

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

```
"registry-mirrors": ["https://mirror.ccs.tencentyun.com"]
```

## 3.3. DNS

/etc/docker/daemon.json

```
{
        "dns": ["8.8.8.8","114.114.114.114"]
}
```

## 3.4. ulimit 资源

/etc/docker/daemon.json

```
"default-ulimits": { "nofile": { "Name": "nofile", "Hard":
128000, "Soft": 128000 } }
```

# 4. docker 命令

## 4.1. docker - A self-sufficient runtime for containers

### 连接远程主机

TCP 2375

```
Neo-iMac:~ neo$ docker -H 192.168.30.10:2375 info
```

SSH 方式

```
Neo-iMac:~ neo$ docker -H ssh://root@192.168.30.13 info

Client:
 Context:    default
 Debug Mode: false
 Plugins:
  buildx: Build with BuildKit (Docker Inc., v0.6.3)
  compose: Docker Compose (Docker Inc., v2.1.1)
  scan: Docker Scan (Docker Inc., 0.9.0)

Server:
 Containers: 3
  Running: 2
  Paused: 0
  Stopped: 1
 Images: 178
 Server Version: 20.10.11
 Storage Driver: overlay2
  Backing Filesystem: xfs
  Supports d_type: true
  Native Overlay Diff: true
  userxattr: false
 Logging Driver: json-file
 Cgroup Driver: cgroupfs
 Cgroup Version: 1
 Plugins:
  Volume: local
  Network: bridge host ipvlan macvlan null overlay
  Log: awslogs fluentd gcplogs gelf journald json-file local
logentries splunk syslog
 Swarm: inactive
 Runtimes: io.containerd.runtime.v1.linux runc
io.containerd.runc.v2
 Default Runtime: runc
 Init Binary: docker-init
 containerd version: 7b11cfaabd73bb80907dd23182b9347b4245eb5d
 runc version: v1.0.2-0-g52b36a2
 init version: de40ad0
 Security Options:
  seccomp
   Profile: default
 Kernel Version: 4.18.0-338.el8.x86_64
 Operating System: CentOS Stream 8
 OSType: linux
 Architecture: x86_64
 CPUs: 4
 Total Memory: 7.514GiB
 Name: localhost.localdomain
 ID:
XGEY:2L25:2GTC:LGK5:3D7D:TC5B:EBBU:5GZJ:VDZ2:S67Z:T7VK:O7WD
 Docker Root Dir: /var/lib/docker
 Debug Mode: false
 Registry: https://index.docker.io/v1/
 Labels:
 Experimental: false
 Insecure Registries:
  registry.netkiller.cn
  127.0.0.0/8
 Registry Mirrors:
  https://registry.cn-hangzhou.aliyuncs.com/
  https://docker.mirrors.ustc.edu.cn/
  https://registry.docker-cn.com/
  http://hub-mirror.c.163.com/
 Live Restore Enabled: false
```

设置 DOCKER_HOST 环境变量

```
Neo-iMac:~ neo$ export DOCKER_HOST=tcp://192.168.30.10:2375
Neo-iMac:~ neo$ docker info
Client:
 Context:    default
 Debug Mode: false
 Plugins:
  buildx: Build with BuildKit (Docker Inc., v0.6.3)
  compose: Docker Compose (Docker Inc., v2.1.1)
  scan: Docker Scan (Docker Inc., 0.9.0)

Server:
 Containers: 11
  Running: 11
  Paused: 0
  Stopped: 0
 Images: 11
 Server Version: 20.10.10
 Storage Driver: overlay2
  Backing Filesystem: xfs
  Supports d_type: true
  Native Overlay Diff: true
  userxattr: false
 Logging Driver: json-file
 Cgroup Driver: cgroupfs
 Cgroup Version: 1
 Plugins:
  Volume: local
  Network: bridge host ipvlan macvlan null overlay
  Log: awslogs fluentd gcplogs gelf journald json-file local
logentries splunk syslog
 Swarm: inactive
 Runtimes: io.containerd.runc.v2 io.containerd.runtime.v1.linux
runc
 Default Runtime: runc
 Init Binary: docker-init
 containerd version: 5b46e404f6b9f661a205e28d59c982d3634148f8
 runc version: v1.0.2-0-g52b36a2
 init version: de40ad0
 Security Options:
  seccomp
```

```
   Profile: default
 Kernel Version: 4.18.0-348.el8.x86_64
 Operating System: CentOS Stream 8
 OSType: linux
 Architecture: x86_64
 CPUs: 4
 Total Memory: 15.39GiB
 Name: testing
 ID:
5GBU:CMWS:VIVP:TREZ:Y5AP:OGOW:EABK:NP4R:AWUA:S4J2:2YQ2:U7MT
 Docker Root Dir: /var/lib/docker
 Debug Mode: false
 Registry: https://index.docker.io/v1/
 Labels:
 Experimental: false
 Insecure Registries:
  127.0.0.0/8
 Registry Mirrors:
  https://hub-mirror.c.163.com/
  https://mirror.baidubce.com/
  https://docker.mirrors.ustc.edu.cn/
 Live Restore Enabled: false
```

## 查看 docker 信息

```
neo@MacBook-Pro ~ % docker info
Containers: 9
 Running: 8
 Paused: 0
 Stopped: 1
Images: 5
Server Version: 18.09.2
Storage Driver: overlay2
 Backing Filesystem: extfs
 Supports d_type: true
 Native Overlay Diff: true
Logging Driver: json-file
Cgroup Driver: cgroupfs
Plugins:
 Volume: local
 Network: bridge host macvlan null overlay
 Log: awslogs fluentd gcplogs gelf journald json-file local
logentries splunk syslog
Swarm: inactive
Runtimes: runc
Default Runtime: runc
Init Binary: docker-init
containerd version: 9754871865f7fe2f4e74d43e2fc7ccd237edcbce
runc version: 09c8266bf2fcf9519a651b04ae54c967b9ab86ec
init version: fec3683
Security Options:
 seccomp
  Profile: default
Kernel Version: 4.9.125-linuxkit
Operating System: Docker for Mac
OSType: linux
Architecture: x86_64
CPUs: 4
Total Memory: 1.952GiB
Name: linuxkit-025000000001
ID: IT7A:OHXM:XG4E:HX53:ZMA3:GIRA:CYMP:6IJF:QKZ5:MQI4:6LU2:ZD7Z
Docker Root Dir: /var/lib/docker
Debug Mode (client): false
Debug Mode (server): true
 File Descriptors: 70
 Goroutines: 88
 System Time: 2019-03-31T04:23:51.43837431Z
 EventsListeners: 2
HTTP Proxy: gateway.docker.internal:3128
HTTPS Proxy: gateway.docker.internal:3129
Registry: https://index.docker.io/v1/
Labels:
Experimental: false
Insecure Registries:
 127.0.0.0/8
Live Restore Enabled: false
Product License: Community Engine
```

iMac

```
iMac:~ neo$ docker info
Client:
 Debug Mode: false
 Plugins:
  buildx: Build with BuildKit (Docker Inc., v0.3.1-tp-docker)
  scan: Docker Scan (Docker Inc., v0.3.3)
  app: Docker Application (Docker Inc., v0.8.0)

Server:
 Containers: 0
  Running: 0
  Paused: 0
  Stopped: 0
 Images: 0
 Server Version: 19.03.13-beta2
 Storage Driver: overlay2
  Backing Filesystem: extfs
  Supports d_type: true
  Native Overlay Diff: true
 Logging Driver: json-file
 Cgroup Driver: cgroupfs
 Plugins:
  Volume: local
  Network: bridge host ipvlan macvlan null overlay
  Log: awslogs fluentd gcplogs gelf journald json-file local
logentries splunk syslog
 Swarm: inactive
 Runtimes: runc
 Default Runtime: runc
 Init Binary: docker-init
 containerd version: 7ad184331fa3e55e52b890ea95e65ba581ae3429
 runc version: dc9208a3303feef5b3839f4323d9beb36df0a9dd
 init version: fec3683
 Security Options:
  seccomp
   Profile: default
 Kernel Version: 4.19.76-linuxkit
 Operating System: Docker Desktop
 OSType: linux
 Architecture: x86_64
 CPUs: 2
 Total Memory: 3.848GiB
 Name: docker-desktop
 ID:
LWQ5:KBRL:SE7U:SJZ4:ANS2:JEQD:5YJO:MVRG:HIEA:XDWD:LQIZ:EJPX
 Docker Root Dir: /var/lib/docker
 Debug Mode: false
 HTTP Proxy: gateway.docker.internal:3128
 HTTPS Proxy: gateway.docker.internal:3129
 Registry: https://index.docker.io/v1/
 Labels:
 Experimental: true
 Insecure Registries:
  127.0.0.0/8
 Registry Mirrors:
  https://registry.docker-cn.com/
 Live Restore Enabled: false
 Product License: Community Engine
```

### run

run

```
$ sudo docker run ubuntu:14.04 /bin/echo 'Hello world'
Hello world
```

**查看 docker run 参数**

```
pip3 install runlike
```

```
格式：runlike -p <容器名>|<容器ID>
```

```
```

**-it**

```
neo@Netkiller-iMac ~> docker run -it nginx:latest /bin/sh
```

**--restart 参数**

该参数用于指定自动重启docker容器策略，包含3个选项：no，on-failure[:times]，always，unless-stopped

```
no 默认值，表示容器退出时，docker不自动重启容器
docker run --restart=no [容器名]

on-failure 若容器的退出状态非0，则docker自动重启容器，还可以指定重启次数，若超过指定次数未能启动容器则放弃
docker run --restart=on-failure:3 [容器名]

always 容器退出时总是重启
docker run --restart=always [容器名]

unless-stopped 容器退出时总是重启，但不考虑Docker守护进程启动时就已经停止的容器
docker run --restart=unless-stopped [容器名]
```

**--privileged 让 root 具备真正的 root 权限**

```
[root@localhost ~]# docker run -t -i centos:latest bash
[root@test /]# lsblk
NAME    MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
vda     254:0    0 59.6G  0 disk
```

```
|-vda1 254:1    0 59.6G  0 part /etc/hosts
`-vda2 252:1    0    1G  0 part
[root@test /]# mount  /dev/vda2  /mnt/
mount: permission denied
```

加入 --privileged 选项后

```
[root@netkiller ~]# docker run -t -i  --privileged
centos:latest bash
[root@test /]# mount  /dev/vda2  /mnt/
```

设置环境变量

```
docker run -e VAR1=value1 --env VAR2=value2 ubuntu
docker run --env VAR1=value1 --env VAR2=value2 ubuntu
```

**DNS**

```
docker run --dns 8.8.8.8 busybox:latest
```

**add-host**

```
docker run --add-host=test.netkiller.cn:172.16.0.73
busybox:latest
```

暴漏端口

```
docker run -p 80:80 ubuntu bash
docker run -p 127.0.0.1:80:80 ubuntu bash
docker run -p 127.0.0.1:80:80/tcp ubuntu bash
```

内存资源分配

-m 或者--memory :分配内存

--memory-swap：分配临时内存

```
docker run -it -m 200M --memory-swap=400M ubuntu
```

给ubuntu分配200兆内存和400M交换分区，一般memory-swap默认是内存两倍。

**start / stop / restart**

```
sudo docker start silly_bohr
silly_bohr

$ sudo docker stop silly_bohr
silly_bohr

$ sudo docker restart silly_bohr
silly_bohr
```

更新容器参数

为容器增加 --restart 参数

```
如果容器启动时没有设置—restart参数，则通过下面命令进行更新：
docker update --restart=always [容器名]
```

**ps**

```
OPTIONS说明：
-a :显示所有的容器，包括未运行的。
-f :根据条件过滤显示的内容。
--format :指定返回值的模板文件。
-l :显示最近创建的容器。
-n :列出最近创建的n个容器。
--no-trunc :不截断输出。
-q :静默模式，只显示容器编号。
-s :显示总的文件大小。
```

```
sudo docker ps
```

```
$ sudo docker ps -l
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
84391d1de0fc ubuntu:14.04 /bin/echo Hello worl 31 minutes ago
Exit 0 romantic_ritchie
```

不截断输出，显示完整信息

正常情况下无法显示完整的 COMMAND 信息

```
neo@MacBook-Pro-Neo ~ % docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED
STATUS                 PORTS
NAMES
08252e252e11   eb705d309426   "redis-server /etc/r…"   About a
minute ago   Up About a minute   0.0.0.0:6379->6379/tcp,
:::6379->6379/tcp   redis
```

使用 --no-trunc 参数可以显示完整信息

```
neo@MacBook-Pro-Neo ~ % docker ps --no-trunc
CONTAINER ID
IMAGE
COMMAND
CREATED                     STATUS                 PORTS
NAMES
08252e252e113105568f8b60b7bcee2f47978938402e440ba6874221a162122
0
sha256:eb705d3094264a13130234869af89b635138f3d05b964ffdf6b3ee96
1f44a664   "redis-server /etc/redis.conf --requirepass
yourpassword"   About a minute ago   Up About a minute
0.0.0.0:6379->6379/tcp, :::6379->6379/tcp   redis
```

格式化输出

格式化选项(--format)

```
.ID 容器ID
.Image 镜像ID
.Command Quoted command
.CreatedAt 创建容器的时间点.
```

```
.RunningFor 从容器创建到现在过去的时间.
.Ports 暴露的端口.
.Status 容器状态.
.Size 容器占用硬盘大小.
.Names 容器名称.
.Labels 容器所有的标签.
.Label 指定label的值 例如'{{.Label “com.docker.swarm.cpu”}}’
.Mounts 挂载到这个容器的数据卷名称
```

```
$ docker ps --format "{{.Names}}={{.ID}}"
portal=04b421501ab7
price=098f85c3c916
admin=8617cb486566
```

## kill 信号

```
docker kill -s HUP <CONTAINER ID>
```

## top

```
$ sudo docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
13b2a4a31455 ubuntu:14.04 /bin/bash 3 hours ago Up 3 hours
silly_bohr

$ sudo docker top silly_bohr
UID PID PPID C STIME TTY TIME CMD
root 23225 22908 0 12:17 pts/14 00:00:00 /bin/bash
```

## inspect

```
$ sudo docker inspect silly_bohr
[{
    "ID":
"13b2a4a3145528d087c9d1580fa78aaa52e8a9bb973c9da923bceb9f9b9e7e
5a",
    "Created": "2014-07-17T04:17:45.262480632Z",
    "Path": "/bin/bash",
    "Args": [],
    "Config": {
        "Hostname": "13b2a4a31455",
        "Domainname": "",
        "User": "",
        "Memory": 0,
        "MemorySwap": 0,
        "CpuShares": 0,
        "AttachStdin": true,
        "AttachStdout": true,
        "AttachStderr": true,
        "PortSpecs": null,
        "ExposedPorts": null,
        "Tty": true,
        "OpenStdin": true,
        "StdinOnce": true,
        "Env": [
            "HOME=/",

"PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/
bin"
        ],
        "Cmd": [
            "/bin/bash"
        ],
        "Dns": [
            "8.8.8.8",
            "8.8.4.4"
        ],
        "Image": "ubuntu",
        "Volumes": null,
        "VolumesFrom": "",
        "WorkingDir": "",
        "Entrypoint": null,
        "NetworkDisabled": false,
        "OnBuild": null
    },
    "State": {
        "Running": true,
        "Pid": 23225,
        "ExitCode": 0,
        "StartedAt": "2014-07-17T04:17:45.672269614Z",
        "FinishedAt": "0001-01-01T00:00:00Z",
        "Ghost": false
    },
    "Image":
"e54ca5efa2e962582a223ca9810f7f1b62ea9b5c3975d14a5da79d3bf6020f
37",
    "NetworkSettings": {
        "IPAddress": "172.17.0.2",
        "IPPrefixLen": 16,
        "Gateway": "172.17.42.1",
        "Bridge": "docker0",
        "PortMapping": null,
        "Ports": {}
    },
    "ResolvConfPath":
"/var/lib/docker/containers/13b2a4a3145528d087c9d1580fa78aaa52e
8a9bb973c9da923bceb9f9b9e7e5a/resolv.conf",
    "HostnamePath":
"/var/lib/docker/containers/13b2a4a3145528d087c9d1580fa78aaa52e
8a9bb973c9da923bceb9f9b9e7e5a/hostname",
    "HostsPath":
"/var/lib/docker/containers/13b2a4a3145528d087c9d1580fa78aaa52e
8a9bb973c9da923bceb9f9b9e7e5a/hosts",
    "Name": "/silly_bohr",
    "Driver": "aufs",
    "ExecDriver": "native-0.1",
    "Volumes": {},
    "VolumesRW": {},
    "HostConfig": {
        "Binds": null,
        "ContainerIDFile": "",
        "LxcConf": [],
        "Privileged": false,
        "PortBindings": {},
        "Links": null,
        "PublishAllPorts": false
    }
}]
```

获取容器名称

```
neo@MacBook-Pro ~ % docker inspect --format='{{.Name}}'
$(docker ps -aq)
/redis-cli
/cluster_redisslave3_1
/cluster_redismaster3_1
/cluster_redismaster2_1
/cluster_redisslave2_1
/cluster_redismaster1_1
/cluster_redisslave1_1
/cluster_redis-image_1
/devel_eureka_1
/devel_config_1
/quizzical_heisenberg

neo@MacBook-Pro ~ % docker inspect --format='{{.Name}}'
$(docker ps -aq)|cut -d"/" -f2
redis-cli
cluster_redisslave3_1
cluster_redismaster3_1
cluster_redismaster2_1
cluster_redisslave2_1
cluster_redismaster1_1
cluster_redisslave1_1
cluster_redis-image_1
devel_eureka_1
devel_config_1
quizzical_heisenberg
```

容器镜像名称

```
neo@MacBook-Pro ~ % docker inspect --format='{{.Config.Image}}'
`docker ps -a -q`
netkiller/redis:latest
netkiller/redis
netkiller/redis
netkiller/redis
netkiller/redis
netkiller/redis
netkiller/redis
netkiller/redis:latest
netkiller/eureka:latest
netkiller/config:latest
netkiller/eureka
```

**获取容器主机名 Hostname**

```
neo@MacBook-Pro ~ % docker inspect --format '{{
.Config.Hostname }}' $(docker ps -q)
dbea51159085
79126b58e92a
5d1fff33a3e1
42a58cb957d9
68904b82d071
70a20dd0396d
742313f2af46
```

**查询 IP 地址**

```
$ sudo docker inspect -f '{{ .NetworkSettings.IPAddress }}'
silly_bohr
```

```
[root@development ~]# docker ps | grep mysql
84639b1810a1   mysql:5.7                     "docker-
entrypoint.s…"   2 weeks ago    Up 22 hours
0.0.0.0:3306->3306/tcp, :::3306->3306/tcp, 33060/tcp
mysql

[root@development ~]# docker inspect -f '{{range
.NetworkSettings.Networks}}{{.IPAddress}}{{end}}' mysql
172.21.0.4
```

```
neo@MacBook-Pro ~ % docker inspect --format='{{range
.NetworkSettings.Networks}}{{.IPAddress}}{{end}}' $(docker ps -
q)

172.24.0.7
172.24.0.6
172.24.0.5
172.24.0.4
172.24.0.3
172.24.0.2
```

获取容器的MAC地址

```
neo@MacBook-Pro ~ % docker inspect --format='{{range
.NetworkSettings.Networks}}{{.MacAddress}}{{end}}' $(docker ps
-a -q)

02:42:ac:18:00:07
02:42:ac:18:00:06
02:42:ac:18:00:05
02:42:ac:18:00:04
02:42:ac:18:00:03
02:42:ac:18:00:02
```

查询子网

```
[root@development ~]# docker network ls | grep nginx
a82ea0e05c7b   nginx_default           bridge    local

[root@development ~]# docker network inspect -f '{{range
.IPAM.Config}}{{.Subnet}}{{end}}' nginx_default
172.26.0.0/16
```

容器日志

```
neo@MacBook-Pro ~ % docker inspect --format='{{.LogPath}}'
`docker ps -a -q`
/var/lib/docker/containers/dbea511590859fee80565d1c047da2443d62
f72f79627c7a97fd891b3ae41168/dbea511590859fee80565d1c047da2443d
62f72f79627c7a97fd891b3ae41168-json.log
/var/lib/docker/containers/79126b58e92adbe933d8e39966af1e19cd86
7afe509deca2689fd27e5d25dce7/79126b58e92adbe933d8e39966af1e19cd
867afe509deca2689fd27e5d25dce7-json.log
/var/lib/docker/containers/5d1fff33a3e14d409e2ef675820d68af0fdd
6d512a7db06540b02b612eb889cc/5d1fff33a3e14d409e2ef675820d68af0f
dd6d512a7db06540b02b612eb889cc-json.log
/var/lib/docker/containers/42a58cb957d965d5ac0aa5d329c6b68aa7f6
2cae096f974df99281f50c4819ab/42a58cb957d965d5ac0aa5d329c6b68aa7
f62cae096f974df99281f50c4819ab-json.log
/var/lib/docker/containers/68904b82d071b956757a54c50d95122210e8
4012542ec3cbe354b72601bf62ba/68904b82d071b956757a54c50d95122210
e84012542ec3cbe354b72601bf62ba-json.log
/var/lib/docker/containers/70a20dd0396d4b48314bfe119d71fc810fe1
7fcb174d0bfb116bb8da53bff677/70a20dd0396d4b48314bfe119d71fc810f
e17fcb174d0bfb116bb8da53bff677-json.log
/var/lib/docker/containers/742313f2af466b7b932f8562e0dc75a228c7
f815b4eb5a35dd1618d94c88bf7e/742313f2af466b7b932f8562e0dc75a228
c7f815b4eb5a35dd1618d94c88bf7e-json.log
/var/lib/docker/containers/d60dcf49c5d4c78904c442f8fb09e5d3d57a
9a2d21f6abaae7ee2d36bcc3e4a2/d60dcf49c5d4c78904c442f8fb09e5d3d5
7a9a2d21f6abaae7ee2d36bcc3e4a2-json.log
/var/lib/docker/containers/44c7ea7593838db1cea824862ee9708c7714
```

```
3d0e07d12cae0116cd8231eb2d1c/44c7ea7593838db1cea824862ee9708c77
143d0e07d12cae0116cd8231eb2d1c-json.log
/var/lib/docker/containers/ae3c930f6eca854c9dc1c2ae84b7c870d63f
3731290d347dc27fcf85c36821e5/ae3c930f6eca854c9dc1c2ae84b7c870d6
3f3731290d347dc27fcf85c36821e5-json.log
/var/lib/docker/containers/9beae3d5f5132e5f733e044d634b1e8b2650
c30151db1a8468109bbf891be674/9beae3d5f5132e5f733e044d634b1e8b26
50c30151db1a8468109bbf891be674-json.log
```

获取 **json** 配置

```
neo@MacBook-Pro ~ % docker inspect --format='{{json .Config}}'
dbea51159085 | jq
{
  "Hostname": "dbea51159085",
  "Domainname": "",
  "User": "",
  "AttachStdin": false,
  "AttachStdout": false,
  "AttachStderr": false,
  "ExposedPorts": {
    "6379/tcp": {}
  },
  "Tty": false,
  "OpenStdin": false,
  "StdinOnce": false,
  "Env": [

"PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/
bin",
    "GOSU_VERSION=1.10",
    "REDIS_VERSION=5.0.4",

"REDIS_DOWNLOAD_URL=http://download.redis.io/releases/redis-
5.0.4.tar.gz",

"REDIS_DOWNLOAD_SHA=3ce9ceff5a23f60913e1573f6dfcd4aa53b42d4a278
9e28fa53ec2bd28c987dd",
    "REDIS_PORT=6379"
  ],
  "Cmd": [
    "redis-cli"
  ],
  "Image": "netkiller/redis:latest",
  "Volumes": {
    "/data": {}
  },
  "WorkingDir": "/data",
  "Entrypoint": [
    "/docker-entrypoint.sh"
  ],
  "OnBuild": null,
  "Labels": {
    "com.docker.compose.config-hash":
"f2e8434ec82c796bceac48461d71d487ff3fb53f711220a1efb976c59bd4d6
8c",
    "com.docker.compose.container-number": "1",
    "com.docker.compose.oneoff": "False",
    "com.docker.compose.project": "cluster",
    "com.docker.compose.service": "redis-cli",
    "com.docker.compose.version": "1.23.2"
  }
}
```

函数

### 拆分和组合

```
neo@MacBook-Pro ~ % docker inspect --format '{{join
.Config.Entrypoint " , "}}' dbea51159085
/docker-entrypoint.sh

neo@MacBook-Pro ~ % docker inspect --format '{{.HostsPath}}'
dbea51159085
/var/lib/docker/containers/dbea511590859fee80565d1c047da2443d62
f72f79627c7a97fd891b3ae41168/hosts
neo@MacBook-Pro ~ % docker inspect --format '{{split .HostsPath
"/"}}' dbea51159085
[ var lib docker containers
dbea511590859fee80565d1c047da2443d62f72f79627c7a97fd891b3ae4116
8 hosts]
```

大小写转换

```
neo@MacBook-Pro ~ % docker inspect --format "{{lower .Name}}"
dbea51159085
/redis-cli
neo@MacBook-Pro ~ % docker inspect --format "{{upper .Name}}"
dbea51159085
/REDIS-CLI
```

首字母大写

```
neo@MacBook-Pro ~ % docker inspect --format "{{title
.State.Status}}" dbea51159085
Restarting
```

长度计算

```
neo@MacBook-Pro ~ % docker inspect --format '{{len .Name}}'
dbea51159085
10
```

打印字符串

```
neo@MacBook-Pro ~ % INSTANCE_ID=42a58cb957d9
```

```
neo@MacBook-Pro ~ % docker inspect --format '{{.State.Pid}}
{{.State.ExitCode}}' $INSTANCE_ID
745770

neo@MacBook-Pro ~ % docker inspect --format '{{print .State.Pid
.State.ExitCode}}' $INSTANCE_ID
74577 0

neo@MacBook-Pro ~ % docker inspect --format '{{printf "Pid:%d
ExitCode:%d" .State.Pid .State.ExitCode}}' $INSTANCE_ID
Pid:74577 ExitCode:0

neo@MacBook-Pro ~ % docker inspect --format '{{.State.Pid}}
{{print "|"}}{{.State.ExitCode}}' $INSTANCE_ID
74577|0
```

综合查询

```
neo@MacBook-Pro ~ % docker inspect --format 'Hostname:{{
.Config.Hostname }}  Name:{{.Name}} IP:{{range
.NetworkSettings.Networks}}{{.IPAddress}}{{end}}' $(docker ps -
q)
Hostname:dbea51159085  Name:/redis-cli IP:
Hostname:79126b58e92a  Name:/cluster_redisslave3_1
IP:172.24.0.7
Hostname:5d1fff33a3e1  Name:/cluster_redismaster3_1
IP:172.24.0.6
Hostname:42a58cb957d9  Name:/cluster_redismaster2_1
IP:172.24.0.5
Hostname:68904b82d071  Name:/cluster_redisslave2_1
IP:172.24.0.4
Hostname:70a20dd0396d  Name:/cluster_redismaster1_1
IP:172.24.0.3
Hostname:742313f2af46  Name:/cluster_redisslave1_1
IP:172.24.0.2
```

```
docker inspect --format '{{ .Config.Hostname }}:{{range
.NetworkSettings.Networks}}{{.IPAddress}}{{end}}' $(docker ps -
q)
```

查看 **Mount** 目录

```
[root@netkiller ~]# docker inspect gitlab | grep Mounts -A 20
        "Mounts": [
            {
                "Source": "/srv/gitlab/config",
                "Destination": "/etc/gitlab",
                "Mode": "",
                "RW": true,
                "Propagation": "rprivate"
            },
            {
                "Source": "/srv/gitlab/logs",
                "Destination": "/var/log/gitlab",
                "Mode": "",
                "RW": true,
                "Propagation": "rprivate"
            },
            {
                "Source": "/srv/gitlab/data",
                "Destination": "/var/opt/gitlab",
                "Mode": "",
                "RW": true,
                "Propagation": "rprivate"
```

## 镜像管理

查看镜像

```
$ sudo docker images
REPOSITORY TAG IMAGE ID CREATED VIRTUAL SIZE
ubuntu 14.10 58faa899733f 2 weeks ago 196 MB
ubuntu utopic 58faa899733f 2 weeks ago 196 MB
ubuntu precise ea7d6801c538 3 weeks ago 127.5 MB
ubuntu 12.04 ea7d6801c538 3 weeks ago 127.5 MB
ubuntu 12.10 c5881f11ded9 4 weeks ago 172.2 MB
ubuntu quantal c5881f11ded9 4 weeks ago 172.2 MB
ubuntu 13.04 463ff6be4238 4 weeks ago 169.4 MB
ubuntu raring 463ff6be4238 4 weeks ago 169.4 MB
ubuntu 13.10 195eb90b5349 4 weeks ago 184.7 MB
ubuntu saucy
195eb90b5349 4 weeks ago 184.7 MB
ubuntu 14.04 e54ca5efa2e9 4 weeks ago 276.5 MB
ubuntu latest e54ca5efa2e9 4 weeks ago 276.5 MB
ubuntu trusty e54ca5efa2e9 4 weeks ago 276.5 MB
ubuntu 10.04 3db9c44f4520 12 weeks ago 183 MB
ubuntu lucid 3db9c44f4520 12 weeks ago 183 MB
```

获取新镜像

```
$ sudo docker pull centos
Pulling repository centos
b7de3133ff98: Pulling dependent layers
5cc9e91966f7: Pulling fs layer
511136ea3c5a: Download complete
ef52fb1fe610: Download complete
```

批量删除镜像

```
docker rmi $(docker images --format "{{.ID}}: {{.Repository}}"
| grep fscs | cut -d: -f1)
```

删除 **<none>** 镜像

```
neo@MacBook-Pro ~/git/springcloud/webflux % docker images |
grep none | cut -f2
<none>                                       <none>
0fe48d3d68c6        About an hour ago   487MB
<none>                                       <none>
8372211e8f27        About an hour ago   487MB
<none>                                       <none>
10e486f8b7e0        About an hour ago   487MB
<none>                                       <none>
4e741a99e2f7        About an hour ago   487MB
<none>                                       <none>
ecb48c238139        About an hour ago   487MB
<none>                                       <none>
5fb2543fe938        About an hour ago   487MB
<none>                                       <none>
2638e33e8168        About an hour ago   487MB
<none>                                       <none>
447651629be0        About an hour ago   470MB
<none>                                       <none>
f66e1450b24b        About an hour ago   487MB
<none>                                       <none>
90e5e4ccedb1        2 hours ago         486MB
<none>                                       <none>
4de93b767f79        3 hours ago         486MB
<none>                                       <none>
746b7846eb74        3 hours ago         470MB
<none>                                       <none>
cb45a33c957a        3 hours ago         470MB
<none>                                       <none>
7a1e07e37dc6        3 hours ago         105MB

neo@MacBook-Pro ~/git/springcloud/webflux % docker rmi -f
$(docker images | grep none | awk '{print $3}')
Deleted:
sha256:0fe48d3d68c6e6784b6080a14a0f06eec55a29f2593b601579ffa3e3
4e0de6fe
Deleted:
```

```
sha256:14a1b072ff90eeccd14530b60576fe488917df6bf4e1e369dfc841ad
f8827e72
Deleted:
sha256:08f9d5b08dca78932767195c9188f6c32fccf6a8394ce0955ae280ca
785187c2
Deleted:
sha256:8372211e8f27dd23093b151a157b990b2d96feec2d3dd9ab38acbd66
45c423c9
Deleted:
sha256:d47c4aec3dec6beae787a1e1ab0245e69ca0e0aeaca76db2decaee3c
5be13c5c
Deleted:
sha256:e791fe1e86eeb86c4195d3558bb67025deaee36c5430fb83c60ab8c1
88774667
Deleted:
sha256:10e486f8b7e000f5deb920cdd7db4d56fceab689747eda8ba365419d
7abb7461
Deleted:
sha256:eaccd2521fab18511d5aa1e51184f25442c3e717e29e85ff255c1f4f
031ea572
Deleted:
sha256:3af7330310b481636cdf756208cac87de4704612f95af2d309aa327b
5d1fd30b
Deleted:
sha256:4e741a99e2f707b6957be436d384d087200ebd11c8673b2c0c1e8bae
f304fbfb
```

批量删除镜像

```

```

## logs

显示容器运行日志，用于排查异常情况

```
$ docker logs [OPTIONS] CONTAINER
  Options:
        --details        显示更多的信息
    -f, --follow         跟踪实时日志
        --since string   显示自某个timestamp之后的日志，或相对时间，
如42m（即42分钟）
        --tail string    从日志末尾显示多少行日志， 默认是all
    -t, --timestamps     显示时间戳
        --until string   显示自某个timestamp之前的日志，或相对时间，
如42m（即42分钟）
```

例如下面是nginx容易启动出错日志

```
[root@netkiller]# docker logs my-nginx-container
nginx: [emerg] invalid server name or wildcard "www.*.com" on
0.0.0.0:80
nginx: [emerg] invalid server name or wildcard "www.*.com" on
0.0.0.0:80
nginx: [emerg] invalid server name or wildcard "www.*.com" on
0.0.0.0:80
nginx: [emerg] invalid server name or wildcard "www.*.com" on
0.0.0.0:80
nginx: [emerg] invalid server name or wildcard "www.*.com" on
0.0.0.0:80
nginx: [emerg] invalid server name or wildcard "www.*.com" on
0.0.0.0:80
```

跟踪实时日志

```
$ docker logs -f CONTAINER_ID
```

显示时间戳

```
$ docker logs -t --since="2018-02-08" --tail=100 CONTAINER_ID
```

显示一段范围内的日志

```
$ docker logs -t --since="2019-02-08T12:20:30" --until "2019-
02-09T12:23:30" CONTAINER_ID
```

## 重置 Docker

```
docker ps -aq | xargs docker rm -f
docker images -aq | xargs docker rmi -f
```

## 仓库操作

https://docs.docker.com/engine/reference/commandline/login/

登陆到一个Docker镜像仓库，如果未指定镜像仓库地址，默认为官方仓库 Docker Hub

登陆

```
docker login -u 用户名 -p 密码
```

登陆到私有仓库

```
$ docker login localhost:8080
```

从标准输出传递密码

```
$ cat ~/my_password.txt | docker login --username foo --
password-stdin
```

注销

```
docker logout
```

## build

```
$ docker build -f /path/to/a/Dockerfile .
```

## 网络管理

```
docker network create -d bridge --subnet 172.25.0.0/16
private_network

docker run -d -v
```

```
/usr/local/etc/redis/redis.conf:/usr/local/etc/redis/redis.conf
-p 6379:6379 --network=private_network --name redis redis
redis-server /usr/local/etc/redis/redis.conf
```

## 事件信息

```
neo@MacBook-Pro-Neo ~ % docker events
2020-10-22T21:29:44.289075472+08:00 network create
8eab34642596e253eb51aa40cc4f5c4c14fb88f1bad7c8cbdeacc2ad411cdb4
4 (name=search_elastic, type=bridge)
2020-10-22T21:29:44.304732058+08:00 volume create search_data01
(driver=local)
2020-10-22T21:29:44.319023013+08:00 volume create search_data02
(driver=local)
2020-10-22T21:29:44.331507541+08:00 volume create search_data03
(driver=local)
2020-10-22T21:29:44.584989392+08:00 volume create search_data01
(driver=local)
```

## 从 docker 中复制文件

```
neo@MacBook-Pro-Neo ~ % docker cp
13acbc98fb35:/etc/nginx/nginx.conf nginx/conf
```

复制文件和目录

```
[root@localhost nginx]# docker cp nginx:/etc/nginx/nginx.conf .
[root@localhost nginx]# docker cp nginx:/etc/nginx/conf.d .
```

## 查看历史记录

```
neo@MacBook-Pro-Neo ~/workspace/Linux % docker history
prom/prometheus:latest
IMAGE          CREATED        CREATED BY
SIZE      COMMENT
267e73020447   9 days ago     /bin/sh -c #(nop)  CMD ["--
config.file=/etc/…   0B
<missing>      9 days ago     /bin/sh -c #(nop)  ENTRYPOINT
["/bin/prometh…   0B
<missing>      9 days ago     /bin/sh -c #(nop) WORKDIR
/prometheus            0B
<missing>      9 days ago     /bin/sh -c #(nop)  VOLUME
[/prometheus]          0B
<missing>      9 days ago     /bin/sh -c #(nop)  EXPOSE 9090
0B
<missing>      9 days ago     /bin/sh -c #(nop)  USER nobody
0B
<missing>      9 days ago     |2 ARCH=amd64 OS=linux /bin/sh -c
mkdir -p /…   1kB
<missing>      9 days ago     |2 ARCH=amd64 OS=linux /bin/sh -c
ln -s /usr…   70B
<missing>      9 days ago     /bin/sh -c #(nop) COPY
file:ccd2272d74b950d3…   129kB
<missing>      9 days ago     /bin/sh -c #(nop) COPY
file:e56be853b56584e3…   3.65kB
<missing>      9 days ago     /bin/sh -c #(nop) COPY
file:141c5dcfe0148c05…   11.4kB
<missing>      9 days ago     /bin/sh -c #(nop) COPY
dir:fb3645c7e168b5a4c…   19.5kB
<missing>      9 days ago     /bin/sh -c #(nop) COPY
dir:6111a57e3d623c34c…   9.04kB
<missing>      9 days ago     /bin/sh -c #(nop) COPY
file:a1aaf2bddcc0da1d…   934B
<missing>      9 days ago     /bin/sh -c #(nop) COPY
file:32c8fb6cc8e0278c…   91.1MB
<missing>      9 days ago     /bin/sh -c #(nop) COPY
file:a9b6183415409ccb…   102MB
```

```
<missing>      9 days ago     /bin/sh -c #(nop)  ARG OS=linux
0B
<missing>      9 days ago     /bin/sh -c #(nop)  ARG ARCH=amd64
0B
<missing>      9 days ago     /bin/sh -c #(nop)  LABEL
maintainer=The Prom…   0B
<missing>      3 months ago   /bin/sh -c #(nop) COPY
dir:bb5589ed25434b0b5…   1.44MB
<missing>      3 months ago   /bin/sh -c #(nop)  MAINTAINER The
Prometheus…   0B
<missing>      3 months ago   /bin/sh -c #(nop)  CMD ["sh"]
0B
<missing>      3 months ago   /bin/sh -c #(nop) ADD
file:dc794c2febce9ec5b…   1.24MB
```

使用 --no-trunc 可以查看被隐藏的部分

```
neo@MacBook-Pro-Neo ~/workspace/Linux % docker history --no-
trunc docker.io/mysql:latest
```

安全漏洞扫描

```
Neo-iMac:nginx neo$ docker scan
Usage:  docker scan [OPTIONS] IMAGE

A tool to scan your images

Options:
      --accept-license    Accept using a third party scanning
provider
      --dependency-tree   Show dependency tree with scan
results
      --exclude-base      Exclude base image from vulnerability
scanning (requires --file)
  -f, --file string       Dockerfile associated with image,
provides more detailed results
      --group-issues      Aggregate duplicated vulnerabilities
and group them to a single one (requires --json)
      --json              Output results in JSON format
      --login             Authenticate to the scan provider
using an optional token (with --token), or web base token if
empty
      --reject-license    Reject using a third party scanning
provider
      --severity string   Only report vulnerabilities of
provided level or higher (low|medium|high)
      --token string      Authentication token to login to the
third party scanning provider
      --version           Display version of the scan plugin
"docker scan" requires exactly 1 argument
```

```
Neo-iMac:nginx neo$ docker scan redis:latest
Neo-iMac:nginx neo$ docker scan 192.168.30.5/netkiller.cn/java
```

## Contexts

```
Neo-iMac:~ neo$ docker context
Manage contexts

Usage:
  docker context [command]

Available Commands:
  create      Create new context
  export      Export a context to a tar or kubeconfig file
  import      Import a context from a tar or zip file
  inspect     Display detailed information on one or more
contexts
  list        List available contexts
  rm          Remove one or more contexts
  show          Print the current context
  update        Update a context
  use           Set the default context

Flags:
  -h, --help   Help for context

Use "docker context [command] --help" for more information
about a command.
```

查看

```
Neo-iMac:~ neo$ docker context ls
NAME                TYPE                DESCRIPTION
DOCKER ENDPOINT                            KUBERNETES ENDPOINT
ORCHESTRATOR
default *           moby                Current DOCKER_HOST
based configuration   unix:///var/run/docker.sock
swarm
desktop-linux       moby
unix:///Users/neo/.docker/run/docker.sock
```

创建

```
localhost       default unix:///var/run/docker.sock
Remote host     remote  ssh://user@remotemachine
docker-in-docker        dind    tcp://127.0.0.1:2375
```

```
Neo-iMac:~ neo$ docker context create development --docker
"host=ssh://root@192.168.30.11"
development
```

```
Successfully created context "development"

Neo-iMac:~ neo$ docker context create testing --docker
"host=tcp://192.168.30.11:2376"
testing
Successfully created context "testing"
```

```
Neo-iMac:~ neo$ docker context ls
NAME                TYPE                DESCRIPTION
DOCKER ENDPOINT                             KUBERNETES ENDPOINT
ORCHESTRATOR
default *           moby                Current DOCKER_HOST
based configuration   unix:///var/run/docker.sock
swarm
desktop-linux       moby
unix:///Users/neo/.docker/run/docker.sock
development         moby
ssh://root@192.168.30.11
testing             moby
tcp://192.168.30.11:2376
```

**inspect**

```
Neo-iMac:~ neo$ docker context inspect
[
    {
        "Name": "default",
        "Metadata": {
            "StackOrchestrator": "swarm"
        },
        "Endpoints": {
            "docker": {
                "Host": "unix:///var/run/docker.sock",
                "SkipTLSVerify": false
            }
        },
        "TLSMaterial": {},
        "Storage": {
            "MetadataPath": "\u003cIN MEMORY\u003e",
            "TLSPath": "\u003cIN MEMORY\u003e"
        }
    }
]
```

**使用 context**

切换默认为 development

```
Neo-iMac:~ neo$ docker context use development
development
```

查看，注意 * 指标

```
Neo-iMac:~ neo$ docker context ls
NAME                TYPE                DESCRIPTION
DOCKER ENDPOINT                             KUBERNETES ENDPOINT
ORCHESTRATOR
default             moby                Current DOCKER_HOST
based configuration   unix:///var/run/docker.sock
swarm
desktop-linux       moby
unix:///Users/neo/.docker/run/docker.sock
development *       moby
ssh://root@192.168.30.11
testing             moby
tcp://192.168.30.11:2376
```

连接到 development 查看 ps

```
Neo-iMac:~ neo$ docker ps
CONTAINER ID   IMAGE                            COMMAND
CREATED        STATUS        PORTS
NAMES
be36eb55d2a7   openjdk:8                        "java -jar
/app/neo…"   6 days ago     Up 40 hours   0.0.0.0:8088-
>8080/tcp, :::8088->8080/tcp
api
5c6892c6d488   redis:alpine                     "docker-
entrypoint.s…"   2 months ago   Up 2 weeks    0.0.0.0:6379-
>6379/tcp, :::6379->6379/tcp
redis
9ee2a3aab354   portainer/agent                  "./agent"
3 months ago   Up 2 weeks    0.0.0.0:9001->9001/tcp, :::9001-
>9001/tcp
portainer-agent
84639b1810a1   mysql:5.7                        "docker-
entrypoint.s…"   3 months ago   Up 2 weeks    0.0.0.0:3306-
>3306/tcp, :::3306->3306/tcp, 33060/tcp
mysql
```

删除

```
Neo-iMac:~ neo$ docker context rm testing
testing
```

**--context 参数**

```
Neo-iMac:~ neo$ docker --context default ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS
NAMES

Neo-iMac:~ neo$ docker --context development ps
```

```
CONTAINER ID   IMAGE                          COMMAND
CREATED        STATUS        PORTS
NAMES
be36eb55d2a7   openjdk:8                      "java -jar
/app/neo…"   6 days ago     Up 41 hours   0.0.0.0:8088-
>8080/tcp, :::8088->8080/tcp
api
```

## 4.2. docker-compose - Define and run multi-container applications with Docker.

### Docker Compose v3

### 安装 docker-compose

#### 使用 pip 安装

```
yum install -y python-pip
pip install docker-compose
```

#### OSCM 安装

```
curl -s
https://raw.githubusercontent.com/oscm/shell/master/virtualizat
ion/docker/docker-compose.sh | bash
```

### 查看版本号

```
```

```
[root@localhost ~]# docker-compose version
docker-compose version 1.29.2, build 5becea4c
docker-py version: 5.0.0
CPython version: 3.7.10
OpenSSL version: OpenSSL 1.1.0l  10 Sep 2019
```

## 快速入门

```
[root@localhost tmp]# cat app.py
import time

import redis
from flask import Flask


app = Flask(__name__)
cache = redis.Redis(host='redis', port=6379)


def get_hit_count():
    retries = 5
    while True:
        try:
            return cache.incr('hits')
        except redis.exceptions.ConnectionError as exc:
            if retries == 0:
                raise exc
            retries -= 1
            time.sleep(0.5)


@app.route('/')
def hello():
    count = get_hit_count()
    return 'Hello World! I have been seen {}
times.\n'.format(count)

if __name__ == "__main__":
    app.run(host="0.0.0.0", debug=True)
```

## 启动

```
[root@localhost tmp]# cat requirements.txt
flask
redis
```

```
[root@localhost tmp]# cat Dockerfile
FROM python:3.4-alpine
ADD . /code
WORKDIR /code
RUN pip install -r requirements.txt
CMD ["python", "app.py"]
```

```
[root@localhost tmp]# cat docker-compose.yml
version: '2'
services:
  web:
    build: .
    ports:
     - "5000:5000"
  redis:
    image: "redis:alpine"
```

## 启动

docker-compose up

```
[root@localhost docker]# docker-compose up
```

守护进程

```
docker-compose up -d
```

启动指定服务

```
[root@localhost docker]# docker-compose up mysql
[root@localhost docker]# docker-compose up -d mysql
```

指定 yml 文件

```
$ docker-compose -f docker-compose.yml up -d
```

## 停止

停止

docker-compose down

```
[root@localhost docker]# docker-compose down
Removing docker_membersrvc_1 ... done
```

启动

## 查看进程

docker-compose ps

```
[root@localhost docker]# docker-compose ps
      Name                         Command                      State
Ports
----------------------------------------------------------------
----------------------------------------------------------------
-----
test_membersrvc_1   membersrvc                                  Up
0.0.0.0:7054->7054/tcp
test_vp0_1          sh -c sleep 5; peer node s ...   Up
0.0.0.0:7050->7050/tcp, 0.0.0.0:7051->7051/tcp, 0.0.0.0:7053-
>7053/tcp
```

## 查看日志

```
docker-compose logs -f vp0
```

查看最后100行日志

```
[www@testing api.netkiller.cn]$ sudo docker-compose logs -f --
tail=100
```

执行命令

```
docker-compose exec vp0 bash
```

运行

```
docker-compose run vp0 bash
```

## 4.3. Docker Scan

安装

```
dnf install docker-scan-plugin
```

扫描

```
docker scan nginx
```

# 5. 镜像管理

Docker 镜像地址 https://registry.hub.docker.com/

## 5.1. 搜索镜像

```
$ sudo docker search centos |more
NAME                                               DESCRIPTION
STARS     OFFICIAL   AUTOMATED
centos                                             The official
build of CentOS.                  542      [OK]
tianon/centos                                      CentOS 5 and 6,
created using rinse instea...   28
ansible/centos7-ansible                            Ansible on
Centos7                            13                   [OK]
saltstack/centos-6-minimal
7                    [OK]
blalor/centos                                      Bare-bones base
CentOS 6.5 image                7                       [OK]
steeef/graphite-centos                             CentOS 6.x with
Graphite and Carbon via ng...   6                       [OK]
ariya/centos6-teamcity-server                      TeamCity Server
8.1 on CentOS 6                 6                       [OK]
tutum/centos                                       Centos image
with SSH access. For the root...   5                       [OK]
tutum/centos-6.4                                   DEPRECATED. Use
tutum/centos:6.4 instead. ...   5                       [OK]
```

## 5.2. 获取镜像

可以使用 docker pull 命令来从官网仓库获取所需要的镜像。

```
$ sudo docker pull ubuntu:14.04
```

等同于

```
$ sudo docker pull registry.hub.docker.com/ubuntu:14.04
```

获得所有版本镜像

```
$ sudo docker pull ubuntu

$ sudo docker images
REPOSITORY          TAG                 IMAGE ID
CREATED             VIRTUAL SIZE
ubuntu              utopic              277eb4304907        3
days ago           215.6 MB
ubuntu              14.10               277eb4304907        3
days ago           215.6 MB
ubuntu              14.04               5506de2b643b        3
days ago           197.8 MB
ubuntu              trusty              5506de2b643b        3
days ago           197.8 MB
ubuntu              latest              5506de2b643b        3
days ago           197.8 MB
ubuntu              14.04.1             5506de2b643b        3
days ago           197.8 MB
ubuntu              precise             0b310e6bf058        3
days ago           116.1 MB
ubuntu              12.04.5             0b310e6bf058        3
days ago           116.1 MB
ubuntu              12.04               0b310e6bf058        3
days ago           116.1 MB
ubuntu              12.10               c5881f11ded9        4
months ago         172.1 MB
ubuntu              quantal             c5881f11ded9        4
months ago         172.1 MB
ubuntu              13.04               463ff6be4238        4
months ago         169.4 MB
```

```
ubuntu                raring               463ff6be4238        4
months ago          169.4 MB
ubuntu                13.10                195eb90b5349        4
months ago          184.6 MB
ubuntu                saucy                195eb90b5349        4
months ago          184.6 MB
ubuntu                10.04                3db9c44f4520        6
months ago          183 MB
ubuntu                lucid                3db9c44f4520        6
months ago          183 MB
```

从其他服务器获得镜像

```
$ sudo docker pull dl.dockerpool.com:5000/ubuntu:12.04
```

完成后，即可随时使用该镜像了，例如创建一个容器，让其中运行 bash 应用。

```
$ sudo docker run -t -i ubuntu:14.10 /bin/bash
```

## 5.3. 列出本地镜像

```
$ sudo docker images
REPOSITORY            TAG                  IMAGE ID
CREATED               VIRTUAL SIZE
ubuntu                14.10                277eb4304907        3
days ago            215.6 MB
ubuntu                latest               5506de2b643b        3
days ago            197.8 MB
```

## 5.4. tag

版本标签

```
docker tag ubuntu:15.10 runoob/ubuntu:v3
```

latest 标签

```
docker tag netkiller/config:10.10 netkiller/config
```

在不同仓库间打标签

```
iMac:registry neo$ docker tag
127.0.0.1:5000/netkiller/config:latest
192.168.64.2:30050/netkiller/config:latest
```

## 5.5. 保存和载入镜像

保存镜像

```
$sudo docker save -o ubuntu_14.10.tar ubuntu:14.10
```

载入镜像

```
$ sudo docker load --input ubuntu_14.10.tar
或
$ sudo docker load < ubuntu_14.10.tar
```

## 5.6. 删除本地镜像

```
$ sudo docker rmi ubuntu:12.04
Untagged: ubuntu:12.04
```

强制删除所有镜像

```
docker rmi -f $(docker images -q)
```

删除 none 标签镜像

```
docker images | grep none | awk '{ print $3; }' | xargs docker
rmi
```

## 5.7. history 镜像历史纪录

镜像历史纪录

```
# docker history centos:tomcat
IMAGE               CREATED             CREATED BY
SIZE                COMMENT
2faf9a2d2bdc        22 hours ago        /bin/sh -c #(nop)  CMD
["catalina.sh" "run"]    0 B
8e12c1e8fd89        22 hours ago        /bin/sh -c #(nop)
EXPOSE 8080/tcp              0 B
35158d8231c5        22 hours ago        /bin/sh -c #(nop)
VOLUME [/srv/tomcat/temp]    0 B
4302c5c13241        22 hours ago        /bin/sh -c #(nop)
VOLUME [/srv/tomcat/work]    0 B
53537696aa19        22 hours ago        /bin/sh -c #(nop) ADD
file:ac42f23f37092b9...   298 B
be04ba27a9ae        23 hours ago        /bin/sh -c set -x   &&
wget -O tomcat.tar....   8.75 MB
847be662a35f        5 days ago          /bin/sh -c #(nop)  ENV
TOMCAT_ASC_URL=http...   0 B
ac6550346558        5 days ago          /bin/sh -c #(nop)  ENV
TOMCAT_TGZ_URL=http...   0 B
50c12be7ca48        5 days ago          /bin/sh -c #(nop)  ENV
TOMCAT_VERSION=8.5.15    0 B
89c44758e4ae        5 days ago          /bin/sh -c #(nop)  ENV
TOMCAT_MAJOR=8           0 B
560ad98c1b23        5 days ago          /bin/sh -c yum install
-y java-1.8.0-openj...   236 MB
befeedbb7dc7        5 days ago          /bin/sh -c #(nop)
WORKDIR /srv/tomcat           0 B
c85cf394faf8        5 days ago          /bin/sh -c mkdir -p
"$CATALINA_HOME"             0 B
debf78012b2c        5 days ago          /bin/sh -c #(nop)  ENV
PATH=/srv/tomcat/bi...   0 B
ccc27f4f3bcf        5 days ago          /bin/sh -c #(nop)  ENV
CATALINA_HOME=/srv/...   0 B
8f351964d568        6 days ago          /bin/sh -c #(nop)
MAINTAINER Netkiller <n...   0 B
3bee3060bfc8        9 days ago          /bin/sh -c #(nop)  CMD
["/bin/bash"]            0 B
<missing>           9 days ago          /bin/sh -c #(nop)
LABEL name=CentOS Base ...   0 B
<missing>           9 days ago          /bin/sh -c #(nop) ADD
file:d22a9c627d1d1f3...   193 MB
```

```
docker history docker.io/mysql:5.7
docker history --no-trunc docker.io/mysql:5.7
```

```
neo@MacBook-Pro-Neo ~ % docker history
docker.elastic.co/kibana/kibana:7.9.2
IMAGE                CREATED             CREATED BY
SIZE                 COMMENT
ba296c26886a         4 weeks ago         /bin/sh -c #(nop)  CMD
["/usr/local/bin/kiba…   0B
<missing>            4 weeks ago         /bin/sh -c #(nop)
ENTRYPOINT ["/usr/local/b…   0B
<missing>            4 weeks ago         /bin/sh -c #(nop)
LABEL org.label-schema.sc…   0B
<missing>            4 weeks ago         /bin/sh -c #(nop)  USER
kibana                   0B
<missing>            4 weeks ago         /bin/sh -c groupadd --
gid 1000 kibana && use…   360kB
<missing>            4 weeks ago         /bin/sh -c find / -xdev
-perm -4000 -exec ch…   484kB
<missing>            4 weeks ago         /bin/sh -c chmod g+ws
/usr/share/kibana && f…   0B
<missing>            4 weeks ago         /bin/sh -c #(nop) COPY
--chown=1000:0file:49…   9.69kB
<missing>            4 weeks ago         /bin/sh -c #(nop) COPY
--chown=1000:0file:ea…   234B
<missing>            4 weeks ago         /bin/sh -c #(nop)  ENV
PATH=/usr/share/kiban…   0B
<missing>            4 weeks ago         /bin/sh -c #(nop)  ENV
ELASTIC_CONTAINER=true   0B
<missing>            4 weeks ago         /bin/sh -c ln -s
/usr/share/kibana /opt/kiba…   17B
<missing>            4 weeks ago         /bin/sh -c #(nop)
WORKDIR /usr/share/kibana     0B
<missing>            4 weeks ago         /bin/sh -c #(nop) COPY
--chown=1000:0dir:e8c…   941MB
<missing>            4 weeks ago         /bin/sh -c chmod +x
```

```
/usr/local/bin/dumb-init    54.7kB
<missing>           4 weeks ago         /bin/sh -c echo
"37f2c1f0372a45554f1b89924fb…   0B
<missing>           4 weeks ago         /bin/sh -c curl -L -o
/usr/local/bin/dumb-in…   75.2kB
<missing>           4 weeks ago         /bin/sh -c yum update -
y && yum install -y f…   31.1MB
<missing>           4 weeks ago         /bin/sh -c #(nop)
EXPOSE 5601                 0B
<missing>           2 months ago        /bin/sh -c #(nop)  CMD
["/bin/bash"]            0B
<missing>           2 months ago        /bin/sh -c #(nop)
LABEL org.label-schema.sc…   0B
<missing>           2 months ago        /bin/sh -c #(nop) ADD
file:61908381d3142ffba…   203MB
```

## 5.8. format 用法

```
docker images --format "{{.Repository}}:{{.Tag}}" | grep
':latest'
```

## 5.9. inspect

```
[root@netkiller ~]# docker image inspect redis:latest | grep -i
version
                "GOSU_VERSION=1.14",
                "REDIS_VERSION=7.0.4",
        "DockerVersion": "20.10.12",
                "GOSU_VERSION=1.14",
                "REDIS_VERSION=7.0.4",
```

## 5.10. 查看镜像内容

```
docker run -it --entrypoint sh <images>
```

操作演示

```
[root@netkiller ~]# docker run -it --entrypoint sh nginx:latest
# find / | more
/
/bin
/bin/bash
/bin/cat
/bin/chgrp
/bin/chmod
/bin/chown
```

# 6. 容器管理

## 6.1. 查看容器

```
iMac:netkiller neo$ docker container ls
```

## 6.2. 启动与终止容器

```
$ sudo docker run ubuntu:14.10 /bin/echo 'Hello world'
Hello world
```

进入BASH

```
$ sudo docker run -t -i ubuntu:14.10 /bin/bash
root@f8c7b2afff14:/#
```

start / stop / restart

```
sudo docker start silly_bohr
silly_bohr

$ sudo docker stop silly_bohr
silly_bohr

$ sudo docker restart silly_bohr
silly_bohr
```

```
[root@localhost ~]# docker container start registry
registry

[root@localhost ~]# docker ps
CONTAINER ID        IMAGE               COMMAND
CREATED             STATUS              PORTS
NAMES
f1e57592f82a        registry:latest     "/entrypoint.sh /etc…"
8 days ago          Up 6 seconds        0.0.0.0:5000->5000/tcp
registry

[root@localhost ~]# curl http://192.168.3.6:5000/v2/_catalog
{"repositories":[]}
```

守护进程运行

```
$ sudo docker run -d ubuntu:14.10 /bin/sh -c "while true; do
echo hello world; sleep 1; done"
4cdbb75eeabf3f1ea87bec91accdf5211639d0895e94ab94ffa1d55fb7f62e2
a
```

通过 docker ps 命令来查看容器信息

```
$ sudo docker ps
CONTAINER ID        IMAGE               COMMAND
CREATED             STATUS              PORTS
NAMES
4cdbb75eeabf        ubuntu:14.10        "/bin/sh -c 'while t
30 seconds ago      Up 28 seconds
drunk_rosalind
```

```
```

要获取容器的输出信息，可以通过 docker logs 命令。

```
$ sudo docker logs insane_babbage
```

注意：守护进程在后台运行，所以无输出，只能通过docker logs命令查看

## 6.3. 进入容器

```
$ sudo docker run -idt ubuntu:14.10
793f9805620d7e10564e0778c388640cb73b6a1aec663bf468904d72a4f219f
2

$ sudo docker ps
CONTAINER ID        IMAGE               COMMAND
CREATED             STATUS              PORTS
NAMES
793f9805620d        ubuntu:14.10        "/bin/bash"         5
seconds ago       Up 4 seconds
mad_elion

$ sudo docker attach mad_elion
root@793f9805620d:/# ls
bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc
root  run  sbin  srv  sys  tmp  usr  var
```

## 6.4. 运行容器内的命令

```
neo@MacBook-Pro-Neo ~ % docker exec prometheus id
```

```
uid=65534(nobody) gid=65534(nogroup)
```

## 6.5. 导出和导入容器

### Ubuntu

```
$ sudo docker export 7691a814370e > ubuntu.tar
```

```
                        <![CDATA[
$ cat ubuntu.tar | sudo docker import - test/ubuntu:v1.0
```

指定 URL 或者某个目录来导入，例如

```
$sudo docker import http://example.com/exampleimage.tgz
example/imagerepo
```

### Mac 导出与导入

导出

```
iMac:tmp neo$ docker export registry -o registry.tar
```

导入

```
iMac:tmp neo$ docker import registry.tar
sha256:1678c838115696f9540f168fe117ea81715b6b676497307e65d15d1a
c10d9a11
```

指定 [REPOSITORY[:TAG]]

```
iMac:tmp neo$ docker import registry.tar registry:latest
sha256:7b76bd807a47dcc60e41bf2f8268ecf69906bb14c2ebaa348c4c15aa
c716b878

iMac:tmp neo$ docker images registry
REPOSITORY          TAG                 IMAGE ID
CREATED             SIZE
registry            latest              7b76bd807a47        11
seconds ago      26.2MB
```

## 6.6. 停止所有容器

杀死所有正在运行的容器

```
docker kill $(docker ps -a -q)
```

### 信号处理

--signal, -s 向容器发送信号

发送一个SIGHUP信号

```
$ docker kill -s=SIGHUP my_container
```

你可以通过名字或数字指定自定义信号，SIG前缀是可选的，例如下面的命令是等价的：

```
$ docker kill -s=SIGHUP my_container
$ docker kill -s=HUP my_container
$ docker kill -s=1 my_container
```

## 6.7. 删除容器

使用 docker rm 来删除一个处于终止状态的容器。

```
$ sudo docker ps -a
CONTAINER ID        IMAGE               COMMAND
CREATED             STATUS                      PORTS
NAMES
f8c7b2afff14        ubuntu:14.10        "/bin/bash"
14 minutes ago      Exited (0) 2 minutes ago
agitated_fermat
0abd2e5fc251        ubuntu:14.10        "/bin/echo 'Hello wo
15 minutes ago      Exited (0) 15 minutes ago
clever_kowalevski

$ sudo docker rm clever_kowalevski
clever_kowalevski

$ sudo docker ps -a
CONTAINER ID        IMAGE               COMMAND
CREATED             STATUS                     PORTS
NAMES
f8c7b2afff14        ubuntu:14.10        "/bin/bash"         16
minutes ago      Exited (0) 5 minutes ago
```

```
agitated_fermat
```

```
$ docker rm
719f98391ecf1d6f1f153ffea1bbd84cd2dc9cf6d31d5a4f348c60d98392814
c
```

删除所有已经停止的容器

```
docker rm $(docker ps -a -q)
```

## 6.8. log-driver

日志发送到 fluentd

```
docker run --log-driver=fluentd --log-opt fluentd-
address=192.168.2.5:24220 ubuntu echo "Hello world"
```

## 6.9. 操作系统

### 设置环境变量

```
iMac:welcome neo$ docker run 127.0.0.1:5000/netkiller/welcome -
e JAVA_OPTS="-server -Xms512m -Xmx4096m"
```

### /etc/hosts 配置

```
# docker run --add-host=docker:10.180.0.1 --rm -it debian
```

向 /etc/hosts 文件内添加主机名

```
docker run -it --add-host=db.netkiller.cn:172.16.18.80 ubuntu
cat /etc/hosts
```

### sysctl

```
$ docker run --sysctl net.ipv4.ip_forward=1 someimage
```

```
docker run -itd --restart=always --net=host \
--name=centos01 --hostname=centos01 \
--sysctl kernel.msgmnb=13107200 \
--sysctl kernel.msgmni=256 \
--sysctl kernel.msgmax=65536 \
--sysctl kernel.shmmax=69719476736 \
--sysctl kernel.sem='500 256000 250 1024' \
-v /mnt/ssd:/var/lib/www \
centos:latest /bin/bash

docker exec centos01 sysctl -a |grep -E \
'kernel.msgmnb|kernel.msgmni|kernel.msgmax|kernel.shmmax|kernel
.sem'
```

**ulimits**

查看 ulimit 设置

```
$ docker run --ulimit nofile=1024:1024 --rm debian sh -c
"ulimit -n"
```

```
$ docker run -it --ulimit as=1024 fedora /bin/bash
$ docker run -d -u daemon --ulimit nproc=3 busybox top
```

```
docker run -d --ulimit nofile=20480:40960 nproc=1024:2048 nginx
```

## 6.10. 查看容器内运行的进程

```
neo@MacBook-Pro-Neo ~ % docker ps
CONTAINER ID        IMAGE
COMMAND                  CREATED             STATUS
PORTS                              NAMES
a6e33697e4bb
docker.elastic.co/elasticsearch/elasticsearch:7.9.2   "/tini --
/usr/local…"   2 minutes ago       Up 2 minutes
9200/tcp, 9300/tcp                 es02
598a6e61d4fc        docker.elastic.co/kibana/kibana:7.9.2
"/usr/local/bin/dumb…"   2 minutes ago       Up 2 minutes
0.0.0.0:5601->5601/tcp             kibana
bc125a658981
docker.elastic.co/elasticsearch/elasticsearch:7.9.2   "/tini --
/usr/local…"   2 minutes ago       Up 2 minutes
9200/tcp, 9300/tcp                 es03
```

```
d027503bee4b
docker.elastic.co/elasticsearch/elasticsearch:7.9.2   "/tini --
/usr/local…"   2 minutes ago       Up 2 minutes
0.0.0.0:9200->9200/tcp, 9300/tcp   elasticsearch

neo@MacBook-Pro-Neo ~ % docker top 598a6e61d4fc
PID                 USER                TIME
COMMAND
3077                1000                0:00
/usr/local/bin/dumb-init -- /usr/local/bin/kibana-docker
3285                1000                1:58
/usr/share/kibana/bin/../node/bin/node
/usr/share/kibana/bin/../src/cli --cpu.cgroup.path.override=/ -
-cpuacct.cgroup.path.override=/
```

## 6.11. 更新容器资源配置

```
neo@MacBook-Pro-Neo ~ % docker update kibana --cpus 1
kibana
```

## 6.12. 查看容器的退出状态

```
neo@MacBook-Pro-Neo ~ % docker wait a6e33697e4bb
0
```

## 6.13. 暂停与恢复容器

暂停容器运行

```
docker pause a6e33697e4bb
```

恢复容器运行

```
docker unpause a6e33697e4bb
```

## 6.14. 对比容器的变化

查看容器启动后，修改了镜像中哪些问题

```
neo@MacBook-Pro-Neo ~ % docker diff a6e33697e4bb
C /tmp
A /tmp/elasticsearch-14495251404334864644
A /tmp/hsperfdata_elasticsearch
A /tmp/hsperfdata_elasticsearch/6
C /usr
C /usr/share
C /usr/share/elasticsearch
C /usr/share/elasticsearch/config
A /usr/share/elasticsearch/config/elasticsearch.keystore
A /usr/share/elasticsearch/.cache
A /usr/share/elasticsearch/.cache/JNA
A /usr/share/elasticsearch/.cache/JNA/temp
C /usr/share/elasticsearch/logs
A /usr/share/elasticsearch/logs/gc.log
A /usr/share/elasticsearch/logs/gc.log.00
```

## 6.15. 查看容器状态

```
neo@MacBook-Pro-Neo ~ % docker stats
CONTAINER ID        NAME                CPU %               MEM
```

```
USAGE / LIMIT     MEM %               NET I/O             BLOCK
I/O           PIDS
a6e33697e4bb        es02                  0.68%
894.2MiB / 3.848GiB   22.69%                13.9MB / 6.95MB
98.9MB / 3.88MB     77
598a6e61d4fc        kibana                0.95%
462.8MiB / 3.848GiB   11.74%                718kB / 13MB
409MB / 4.1kB       12
bc125a658981        es03                  2.67%
889.9MiB / 3.848GiB   22.58%                1.76MB / 5.79MB
48.5MB / 3.09MB     71
d027503bee4b        elasticsearch         2.75%
928.4MiB / 3.848GiB   23.56%                24MB / 14.7MB
139MB / 8.57MB      75
```

## 6.16. 重启容器

--time, -t 10 停止容器之前需要等待的时间(秒)

```
$ docker restart [options] container [container...]
```

## 6.17. DNS

host.docker.internal

gateway.docker.internal

# 7. 卷管理

## 7.1. 列出卷

docker volume ls

```
# docker volume ls
DRIVER              VOLUME NAME
local
dbac41b6de88c75d2932d5949367b17f347f482977d508195375dbc71518ab2
7
```

## 7.2. 创建卷

```
# docker volume create --name WebVolume1
WebVolume1
```

```
# docker volume ls
DRIVER              VOLUME NAME
local               WebVolume1
local
dbac41b6de88c75d2932d5949367b17f347f482977d508195375dbc71518ab2
7
```

## 7.3. 挂在镜像

```
# docker run -ti --rm -v WebVolume1:/www ubuntu
# docker run -ti --rm -v WebVolume1:/www docker.io/centos:7
```

查看卷的挂载情况

```
# df | grep /www
/dev/vda1       20510332 7943940  11501484  41% /www
```

创建测试文件

```
# mkdir -p /www/netkiller.cn/www.netkiller.cn
# echo Helloworld >
/www/netkiller.cn/www.netkiller.cn/index.html
# cat /www/netkiller.cn/www.netkiller.cn/index.html
Helloworld
# exit
exit
```

## 7.4. 检查卷

```
# docker volume inspect WebVolume1
[
    {
        "Driver": "local",
        "Labels": {},
        "Mountpoint":
"/var/lib/docker/volumes/WebVolume1/_data",
        "Name": "WebVolume1",
        "Options": {},
        "Scope": "local"
    }
]
```

## 7.5. 删除卷

```
# docker volume create AppVolume1
# docker volume rm AppVolume1
```

## 7.6. 销毁所有未使用的卷

```
# docker volume prune
WARNING! This will remove all volumes not used by at least one
container.
Are you sure you want to continue? [y/N] y
Deleted Volumes:
WebVolume1
3fd379f8c2cf8727d2e83e84e434ea1f122016957bd7cf78a0f05b6e5a69cf2
b
app

Total reclaimed space: 11 B
```

## 7.7. 在多个容器间共享卷

容器一

```
# docker run -ti --name=Container1 -v DataVolume1:/opt/data
ubuntu
```

容器二

```
# docker run -ti --name=Container2 --volumes-from Container1
ubuntu
```

进入容器一中查看数据

```
# docker start -ai Container1
```

容器三，挂在只读卷

```
# docker run -ti --name=Container3 --volumes-from Container2:ro
ubuntu
```

删除上面三个测试容易和卷

```
# docker rm Container1 Container2 Container3
# docker volume rm DataVolume1
```

## 7.8. 容器绑定本地文件系统

### Bind mount a volume (default [])

```
# docker run -it --name mycentos1 -v /www:/tmp/test
docker.io/centos:7 /bin/bash
# docker run -d -v ~/logs:/var/log/nginx -p 80:80 -i nginx
```

## 7.9. 只读权限

/etc/redis/redis.conf:/etc/redis/redis.conf:ro 表示只读权限

```
docker run \
-p 6379:6379 \
-v /var/lib/redis:/data \
-v /etc/redis/redis.conf:/etc/redis/redis.conf:ro \
--privileged=true \
--name redis \
-d docker.io/redis:latest redis-server /etc/redis/redis.conf
```

# 8. Docker 网络管理

## 8.1. docker0 IP地址

查看 docker0 的IP地址

```
root@production:~# ifconfig docker0
docker0   Link encap:Ethernet  HWaddr 02:42:ad:68:6b:cf
          inet addr:172.18.0.1  Bcast:172.18.255.255
Mask:255.255.0.0
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

修改 docker0 的IP地址

```
root@production:~# vim /etc/docker/daemon.json
root@production:~# cat /etc/docker/daemon.json
{
  "bip":"172.100.10.1/24"
}
root@production:~# systemctl restart docker

root@production:~# ifconfig docker0
docker0   Link encap:Ethernet  HWaddr 02:42:ad:68:6b:cf
          inet addr:172.100.10.1  Bcast:172.100.10.255
Mask:255.255.255.0
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

**提示**

曾经遇到一个案例，阿里云使用172.18.0.0/16作为RDS内网IP地址，ECS安装了docker后无法链接RDS属于，因为docker修改了路由表，将docker换到其他网段后工作正常。

## 8.2. 容器指定固定IP地址

```
docker run -d --privileged -p 9000:9000 --ip 192.168.5.2 \
--restart=always \
-v /var/run/docker.sock:/var/run/docker.sock \
-v /opt/portainer:/data \
portainer/portainer
```

## 8.3. 创建子网

```
docker network create --subnet=172.32.0.0/24 web
```

## 8.4. 创建 overlay 网络

```
docker network create \
  --driver=overlay \
  --subnet=172.12.0.0/16 \
  --ip-range=172.12.0.0/16 \
  --gateway=172.12.0.1 \
  --attachable \
  test
```

```
iMac:redis neo$ docker network ls
NETWORK ID          NAME                DRIVER
SCOPE
786efe30f42d        bridge              bridge
local
51e2b21d7daa        docker_gwbridge     bridge
local
96ba0de26cd2        host                host
local
7r7k9robn0uu        ingress             overlay
swarm
cbf078a5f121        none                null
local
d851mrlkludv        redis_default       overlay
swarm
q0h9awx86ef4        registry_default    overlay
swarm
cf585ea9ceb4        registry_default    bridge
local
gvcz5y66ovrl        test                overlay
swarm
```

查看详细信息

```
iMac:redis neo$ docker network inspect test
[
    {
        "Name": "test",
        "Id": "gvcz5y66ovrlqfaxb02zx026t",
        "Created": "2020-09-26T14:07:49.037581155Z",
        "Scope": "swarm",
        "Driver": "overlay",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "172.12.0.0/16",
                    "IPRange": "172.12.0.0/16",
                    "Gateway": "172.12.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": true,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": null,
        "Options": {
            "com.docker.network.driver.overlay.vxlanid_list":
"4104"
        },
        "Labels": null
    }
]
```

## 8.5. 网络命令空间

```
[root@localhost ~]# docker inspect --format="{{ .State.Pid }}"
b279738af403
2180

[root@localhost ~]# mkdir -p /var/run/netns
[root@localhost ~]# ln -s /proc/2180/ns/net /var/run/netns/2180

[root@localhost ~]# ip netns exec 2180 ip route
default via 192.168.49.1 dev eth0
172.17.0.0/16 dev docker0 proto kernel scope link src
172.17.0.1
192.168.30.0/24 via 192.168.49.1 dev eth0
192.168.49.0/24 dev eth0 proto kernel scope link src
192.168.49.2
```

## 8.6. flannel 网络配置

```
[root@master ~]# ip -d link show flannel.1
11: flannel.1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc
noqueue state UNKNOWN mode DEFAULT group default
    link/ether c2:51:5c:09:4e:18 brd ff:ff:ff:ff:ff:ff
promiscuity 0 minmtu 68 maxmtu 65535
    vxlan id 1 local 172.18.200.5 dev enp3s0 srcport 0 0
dstport 8472 nolearning ttl auto ageing 300 udpcsum
noudp6zerocsumtx noudp6zerocsumrx addrgenmode eui64 numtxqueues
1 numrxqueues 1 gso_max_size 64000 gso_max_segs 64

[root@master ~]# cat /run/flannel/subnet.env
FLANNEL_NETWORK=10.42.0.0/16
FLANNEL_SUBNET=10.42.0.1/24
FLANNEL_MTU=1450
FLANNEL_IPMASQ=true

[root@master ~]# dockerd --bip=$FLANNEL_SUBNET --
mtu=$FLANNEL_MTU
```

```
[root@agent-1 ~]# ip -d link show flannel.1
5: flannel.1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc
noqueue state UNKNOWN mode DEFAULT group default
    link/ether 56:e0:f3:da:d5:c4 brd ff:ff:ff:ff:ff:ff
promiscuity 0 minmtu 68 maxmtu 65535
    vxlan id 1 local 172.18.200.51 dev enp3s0 srcport 0 0
dstport 8472 nolearning ttl auto ageing 300 udpcsum
noudp6zerocsumtx noudp6zerocsumrx addrgenmode eui64 numtxqueues
1 numrxqueues 1 gso_max_size 64000 gso_max_segs 64

[root@agent-1 ~]# cat /run/flannel/subnet.env
FLANNEL_NETWORK=10.42.0.0/16
FLANNEL_SUBNET=10.42.1.1/24
```

```
FLANNEL_MTU=1450
FLANNEL_IPMASQ=true

[root@agent-1 ~]# cat /etc/docker/daemon.json
{

"bip":"10.42.1.254/24",
        "ip-masq":true,
        "mtu":1472,

  "registry-mirrors": [
    "https://docker.mirrors.ustc.edu.cn/"
  ]
}

[root@agent-1 ~]# cat /usr/lib/systemd/system/docker.service
[Unit]
Description=Docker Application Container Engine
Documentation=https://docs.docker.com
After=network-online.target docker.socket firewalld.service
containerd.service
Wants=network-online.target
Requires=docker.socket containerd.service

[Service]
Type=notify
EnvironmentFile=-/run/flannel/subnet.env
# 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:// --
containerd=/run/containerd/containerd.sock --
bip=$FLANNEL_SUBNET --mtu=$FLANNEL_MTU
ExecReload=/bin/kill -s HUP $MAINPID
TimeoutSec=0
RestartSec=2
Restart=always

# Note that StartLimit* options were moved from "Service" to
"Unit" in systemd 229.
# Both the old, and new location are accepted by systemd 229
and up, so using the old location
# to make them work for either version of systemd.
```

```
StartLimitBurst=3

# Note that StartLimitInterval was renamed to
StartLimitIntervalSec in systemd 230.
# Both the old, and new name are accepted by systemd 230 and
up, so using the old name to make
# this option work for either version of systemd.
StartLimitInterval=60s

# Having non-zero Limit*s causes performance problems due to
accounting overhead
# in the kernel. We recommend using cgroups to do container-
local accounting.
LimitNOFILE=infinity
LimitNPROC=infinity
LimitCORE=infinity

# Comment TasksMax if your systemd version does not support it.
# Only systemd 226 and above support this option.
TasksMax=infinity

# 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
OOMScoreAdjust=-500

[Install]
WantedBy=multi-user.target
```

```
[root@master ~]# docker run -it --name test busybox /bin/sh
/ # ifconfig
eth0      Link encap:Ethernet  HWaddr 02:42:0A:2A:01:01
          inet addr:10.42.0.2  Bcast:10.42.1.255
Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1472  Metric:1
          RX packets:12 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:1016 (1016.0 B)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)


[root@agent-1 ~]# docker run -it --name test busybox /bin/sh
/ # ifconfig
eth0      Link encap:Ethernet  HWaddr 02:42:0A:2A:01:01
          inet addr:10.42.1.2  Bcast:10.42.1.255
Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1472  Metric:1
          RX packets:12 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:1016 (1016.0 B)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

/ # ping 10.42.0.2
```

# 9. 日志管理

## 9.1. 查看默认驱动

查看默认驱动 docker info --format '{{.LoggingDriver}}'

```
[root@testing ~]# docker info --format '{{.LoggingDriver}}'
json-file
```

查看容器日志配置

```
[root@testing ~]# docker inspect -f
'{{.HostConfig.LogConfig.Type}}' api
fluentd
```

## 9.2. Fluentd 配置

在 Docker 中安装 Fluentd

准备 test.conf 文件

```
 <source>
   @type forward
 </source>

 <match **>
   @type stdout
 </match>
```

启动 fluentd 接收日志

```
 $ docker run -it -p 24224:24224 -v
/path/to/conf/test.conf:/fluentd/etc/test.conf -e
FLUENTD_CONF=test.conf fluent/fluentd:latest
```

## 9.3. Docker 配置

运行你的程序

```
$ docker run --log-driver=fluentd your/application
```

如果是远程主机使用 fluentd-address 参数

```
docker run --log-driver=fluentd --log-opt fluentd-
address=fluentdhost:24224
docker run --log-driver=fluentd --log-opt fluentd-
address=tcp://fluentdhost:24224
docker run --log-driver=fluentd --log-opt fluentd-
address=unix:///path/to/fluentd.sock
```

以 Nginx 为例:

```
$ docker run -d \
--log-driver=fluentd \
--log-opt fluentd-address=10.10.0.1:24224 \
--log-opt tag="docker.{{.Name}}" \
nginx
```

## 9.4. docker-compose 编排

fluentd.conf

```
<source>
  @type forward
</source>

<match **>
  @type file
  path                  /var/log/fluentd/${tag}
  append                true
  <format>
    @type               single_value
    message_key         log
  </format>
  <buffer tag,time>
    @type                file
    timekey              1d
    timekey_wait         10m
    flush_mode           interval
    flush_interval       30s
  </buffer>
</match>
```

```
version: '3.9'
services:
  fluentd:
    image: fluent/fluentd:latest
    container_name: fluentd
    hostname: fluentd.netkiller.cn
    restart: always
    volumes:
      -
/opt/netkiller.cn/ops.netkiller.cn/fluentd/conf:/fluentd/etc
      - /var/log/fluentd:/var/log/fluentd
    ports:
      - "24224:24224"
      - "24224:24224/udp"
    environment:
      FLUENTD_CONF: fluentd.conf

  api:
    image: openjdk:8
    container_name: api
    restart: always
    hostname: api.netkiller.cn
    extra_hosts:
      - www.netkiller.cn:139.186.170.130
    environment:
      TZ: Asia/Shanghai
      JAVA_OPTS: -Xms1024m -Xmx4096m -XX:MetaspaceSize=128m -
XX:MaxMetaspaceSize=512m
    ports:
      - 8088:8080
    volumes:
      - /opt/netkiller.cn/api.netkiller.cn:/app
      - /opt/netkiller.cn/api.netkiller.cn/logs:/app/logs
    working_dir: /app
    #links:
    #  - fluentd
    logging:
      driver: fluentd
      options:
        fluentd-address: 192.168.30.10:24224
        tag: httpd.access
    entrypoint: java -jar /app/api.netkiller.cn.jar
    command:
      --spring.profiles.active=test
      --server.port=8080
```

## 9.5. 将日志输出到 /dev/stdout 和 /dev/stderr

```
# ls -l /var/log/nginx/
total 0
lrwxrwxrwx    1 root     root            11 Jan 31  2022
access.log -> /dev/stdout
lrwxrwxrwx    1 root     root            11 Jan 31  2022
error.log -> /dev/stderr
```

# 10. Dockerfile

## 10.1. 基于 Dockerfile 创建镜像

为什么要自己创建镜像呢？ 因为官方提供的镜像无法满足我们的需求，例如 nginx 镜像你会发现 ps, top 等等很多命令缺失。

### 创建 Dockerfile 文件

需求基于centos7镜像创建nginx stable最新版本镜像

```
############################################################
# Dockerfile to build Nginx container
# Based on centos7
############################################################

FROM centos:latest

MAINTAINER Netkiller <netkiller@msn.com>

# Install EPEL
RUN yum install -y epel-release && yum clean all

# Update RPM Packages
RUN yum -y update

# Install Nginx
RUN rpm -ivh
http://nginx.org/packages/centos/7/noarch/RPMS/nginx-release-
centos-7-0.el7.ngx.noarch.rpm
RUN yum install -y nginx
RUN yum clean all

# forward request and error logs to docker log collector
RUN ln -sf /dev/stdout /var/log/nginx/access.log
RUN ln -sf /dev/stderr /var/log/nginx/error.log

# be backwards compatible with pre-official images
```

```
#RUN ln -sf ../share/nginx /usr/local/nginx

# prepare container

# add startup script
#ADD startup.sh /startup.sh
#RUN chmod 755 /startup.sh

VOLUME ["/etc/nginx"]
VOLUME ["/usr/share/nginx/html"]
VOLUME ["/var/www"]

EXPOSE 80 443

CMD ["nginx", "-g", "daemon off;"]
```

创建镜像

```


# docker build -t "centos:nginx" .
Sending build context to Docker daemon 3.072 kB
Step 1/14 : FROM centos:latest
 ---> 3bee3060bfc8
Step 2/14 : MAINTAINER Netkiller <netkiller@msn.com>
 ---> Using cache
 ---> 8f351964d568
Step 3/14 : RUN yum install -y epel-release && yum clean all
 ---> Using cache
 ---> bf86eff77ff3
Step 4/14 : RUN yum -y update
 ---> Using cache
 ---> 4915172ac4f3
Step 5/14 : RUN rpm -ivh
http://nginx.org/packages/centos/7/noarch/RPMS/nginx-release-
centos-7-0.el7.ngx.noarch.rpm
 ---> Using cache
 ---> 4a919bd141c9
Step 6/14 : RUN yum install -y nginx
 ---> Using cache
 ---> 2718221eab8c
```

```
Step 7/14 : RUN yum clean all
 ---> Using cache
 ---> 62231a5f1d76
Step 8/14 : RUN ln -sf /dev/stdout /var/log/nginx/access.log
 ---> Using cache
 ---> 38be8f0cc782
Step 9/14 : RUN ln -sf /dev/stderr /var/log/nginx/error.log
 ---> Using cache
 ---> bbf3a468d24f
Step 10/14 : VOLUME /etc/nginx
 ---> Using cache
 ---> 919292c7ce04
Step 11/14 : VOLUME /usr/share/nginx/html
 ---> Using cache
 ---> c2aeb8ed3c1c
Step 12/14 : VOLUME /var/www
 ---> Using cache
 ---> 31849cb8a9d0
Step 13/14 : EXPOSE 80 443
 ---> Using cache
 ---> 0e3d3b4a215b
Step 14/14 : CMD nginx -g daemon off;
 ---> Using cache
 ---> d5f21e409690
Successfully built d5f21e409690
```

查看镜像

```
# docker image ls
REPOSITORY          TAG                 IMAGE ID
CREATED             SIZE
centos              nginx               d5f21e409690        4
minutes ago        364 MB
centos              latest              3bee3060bfc8        2
days ago           193 MB
nginx               latest              958a7ae9e569        8
days ago           109 MB
redis               latest              a858478874d1        2
weeks ago          184 MB
```

```
```

## 运行镜像

```
# docker run --name my-centos-nginx -d centos:nginx
ecf342ddd66d1d5f3d28c583ec852c05903ef4813fcb75295c907a6b578dea3
d

# docker ps
CONTAINER ID        IMAGE               COMMAND
CREATED             STATUS              PORTS
NAMES
ecf342ddd66d        centos:nginx        "nginx -g 'daemon ..."
23 seconds ago      Up 23 seconds       80/tcp, 443/tcp
my-centos-nginx
0df3b275bb03        nginx               "nginx -g 'daemon ..."
6 hours ago         Up 6 hours          80/tcp
my-nginx
1c4540d8617f        redis               "docker-entrypoint..."
2 days ago          Up 2 days           0.0.0.0:6379->6379/tcp
my-redis
```

## 测试 Nginx

```
[root@netkiller]~/docker/nginx# docker exec -it my-centos-nginx
/bin/bash

[root@netkiller-docker /]# ps ax
  PID TTY      STAT   TIME COMMAND
    1 ?        Ss     0:00 nginx: master process nginx -g
daemon off;
    7 ?        S      0:00 nginx: worker process
    8 ?        Ss     0:00 /bin/bash
   22 ?        R+     0:00 ps ax
```

```
[root@netkiller-docker /]# curl http://localhost
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully
installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

提交镜像

```
# docker commit my-centos-nginx netkiller/centos:nginx
sha256:9ea1851b1c9f04aa3168977f666337223d09e20983f7a2c2328e1513
2a03d224
```

```
# docker push netkiller/centos:nginx
The push refers to a repository [docker.io/netkiller/centos]
16916856eaaa: Pushed
6172d61b45f1: Pushed
db323af550f0: Pushed
232df2cfd38f: Pushed
c247a550215b: Pushed
3b5451d7989c: Pushed
e3a6f1af6a7a: Pushed
9e3cea652b37: Pushed
dc1e2dcdc7b6: Mounted from library/centos
nginx: digest: 
sha256:ad9bd1ae3a3e17dac70f32afc14baf90932949d3eaa8bebbe907726a
ca3ea336 size: 2205
```

## 10.2. 基于 Alpine 制作镜像

获取最新镜像

```
root@netkiller ~# docker pull alpine:latest
```

运行镜像，看看这个镜像，在里面模拟一次执行

```
root@netkiller ~# docker run --rm -it --name=alpine --
entrypoint=sh alpine:latest
```

进入容器，修改apk库的镜像

```
root@netkiller ~# docker run --rm -it --name=alpine --
entrypoint=sh alpine:latest
```

```
sed 's/dl-cdn.alpinelinux.org/mirrors.aliyun.com/g' -i
/etc/apk/repositories
apk update
apk add python3
```

```


FROM python:3-alpine
MAINTAINER netkiller "netkiller@msn.com"

RUN echo https://mirrors.aliyun.com/alpine/latest-stable/main/
> /etc/apk/repositories
RUN pip install --no-cache-dir flask && pip3 install python-
jenkins
RUN pip install --no-cache-dir netkiller-devops --upgrade -i
https://pypi.tuna.tsinghua.edu.cn/simple
RUN mkdir -p /data

ADD ./ /data

RUN chmod +x /data/devops
RUN rm -rf /var/cache/apk/*

WORKDIR /data

EXPOSE 8080

CMD ["python3", "app.py"]
```

## 10.3. Dockerfile 缺失的工具

工作中我们常常发现官方镜像裁剪的面目全非，里面缺失很多常用工具，这种情况给我们工作带来诸多不便。

### Debian/Ubuntu 镜像

切换镜像

https://mirrors.tuna.tsinghua.edu.cn/help/debian/

```
cat > /etc/apt/sources.list <<-EOF

deb https://mirrors.tuna.tsinghua.edu.cn/debian/ bullseye main contrib non-free
# deb-src https://mirrors.tuna.tsinghua.edu.cn/debian/ bullseye main contrib non-free
deb https://mirrors.tuna.tsinghua.edu.cn/debian/ bullseye-updates main contrib non-free
# deb-src https://mirrors.tuna.tsinghua.edu.cn/debian/ bullseye-updates main contrib non-free

deb https://mirrors.tuna.tsinghua.edu.cn/debian/ bullseye-backports main contrib non-free
# deb-src https://mirrors.tuna.tsinghua.edu.cn/debian/ bullseye-backports main contrib non-free

deb https://mirrors.tuna.tsinghua.edu.cn/debian-security bullseye-security main contrib non-free
# deb-src https://mirrors.tuna.tsinghua.edu.cn/debian-security bullseye-security main contrib non-free

EOF
```

ps,top 等系统工具

```
apt update -y && apt install -y procps
```

ping

```
apt install iputils-ping
```

telnet

```
apt install -y telnet
```

ip, ss

```
apt install -y iproute2
```

ifconfig, netstat

```
apt install -y net-tools
```

dig

```
apt install -y dnsutils
```

**CentOS**

psmisc 里面包含 ps, top 等命令

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

nslookup

```
dnf install -y bind-utils
```

```
dnf install -y net-tools
```

**alpine**

添加 apk 仓库

```
FROM python:3.9-alpine
MAINTAINER netkiller "netkiller@msn.com"

RUN echo https://mirrors.aliyun.com/alpine/latest-stable/main/
> /etc/apk/repositories
RUN pip3 install flask && pip3 install python-jenkins
RUN mkdir -p /data

ADD ./ /data

RUN chmod +x /data/devops
RUN rm -rf /var/cache/apk/*

WORKDIR /data

EXPOSE 8080
```

```
CMD ["python3", "app.py"]
```

## 10.4. Dockerfile 语法

### COPY

跨容器拷贝

```
FROM demo/test:latest as netkiller

MAINTAINER Netkiller <netkiller@msn.com>

RUN mkdir /www

COPY some/path/to/ /www/



FROM nginx:1.13-alpine

RUN rm -rf /usr/share/nginx/html/*
COPY --from=netkiller /www/ /usr/share/nginx/html/
```

--from 参数

```
# Install the base requirements for the app.
# This stage is to support development.
FROM python:alpine AS base
WORKDIR /app
COPY requirements.txt .
RUN pip install -r requirements.txt

# Run tests to validate app
FROM node:12-alpine AS app-base
```

```
WORKDIR /app
COPY app/package.json app/yarn.lock ./
RUN yarn install
COPY app/spec ./spec
COPY app/src ./src
RUN yarn test

# Clear out the node_modules and create the zip
FROM app-base AS app-zip-creator
RUN rm -rf node_modules && \
    apk add zip && \
    zip -r /app.zip /app

# Dev-ready container - actual files will be mounted in
FROM base AS dev
CMD ["mkdocs", "serve", "-a", "0.0.0.0:8000"]

# Do the actual build of the mkdocs site
FROM base AS build
COPY . .
RUN mkdocs build

# Extract the static content from the build
# and use a nginx image to serve the content
FROM nginx:alpine
COPY --from=app-zip-creator /app.zip
/usr/share/nginx/html/assets/app.zip
COPY --from=build /app/site /usr/share/nginx/html
```

## EXPOSE

EXPOSE 是声明端口，容器内运行的程序使用了什么端口

```
EXPOSE <端口1> [<端口2>...]
```

## ENTRYPOINT

从命令行传递参数给容器

```
FROM ubuntu
ENTRYPOINT [ "top", "-b" ]
```

运行下面的命令:

```
$ docker run --rm test1 -c
```

实际 Docker 内部

```
top -b -c
```

ENTRYPOINT 与 CMD 组合

```
FROM ubuntu
ENTRYPOINT [ "top", "-b" ]
CMD [ "-c" ]
```

**docker-entrypoint.sh** 文件

```
ENTRYPOINT ["docker-entrypoint.sh"]
```

你不能写成

```
ENTRYPOINT docker-entrypoint.sh
```

ENTRYPOINT docker-entrypoint.sh 会使用 sh -c 执行

```
 "/bin/sh -c /srv/docker-entrypoint.sh
/srv/rocketmq/bin/mqnamesrv"
```

而我们需要的是

```
/srv/docker-entrypoint.sh /srv/rocketmq/bin/mqnamesrv
```

所以需要写成 ENTRYPOINT ["docker-entrypoint.sh"]

## 11. 仓库

### 11.1. Docker 官方仓库

登陆仓库

登录

```
$ sudo docker login
Username: netkiller
Password:
Email: netkiller@msn.com
Login Succeeded
```

获取镜像

```
docker pull ubuntu:14.04
```

上传镜像

```
docker tag friendlyhello username/repository:tag
docker push username/repository:tag
```

### 11.2. 私有仓库

## 搭建私有仓库

搭建私有仓库只需两步

```
docker pull registry
docker run -d -p 5000:5000 -v /opt/registry:/var/lib/registry -
-name registry registry
```

操作演示

```
neo@ubuntu:~$ docker pull registry
Using default tag: latest
latest: Pulling from library/registry
169185f82c45: Pull complete
046e2d030894: Pull complete
188836fddeeb: Pull complete
832744537747: Pull complete
7ceea07e80be: Pull complete
Digest:
sha256:870474507964d8e7d8c3b53bcfa738e3356d2747a42adad26d0d81ef
4479eb1b
Status: Downloaded newer image for registry:latest

neo@ubuntu:~$ docker run -d -p 5000:5000 -v
/opt/registry:/tmp/registry registry
38a6d3b5e18e378b7765fa00374426db3a06c64f4b9219a1f85dc42a6a66ef2
8

neo@ubuntu:~$ docker ps | grep registry
38a6d3b5e18e        registry              "/entrypoint.sh
/etc…"   35 seconds ago      Up 33 seconds       0.0.0.0:5000-
>5000/tcp
```

设置允许http协议访问，有两种方式，一种是修改/etc/docker/daemon.json并添加 “insecure-registries” 项

```
{
    "registry-mirrors": ["https://registry.docker-cn.com"],
    "insecure-registries": ["127.0.0.1:5000"]
}
```

另一种方式是修改 /etc/default/docker 中加入下面内容

```
neo@ubuntu:~$ sudo vim /etc/default/docker

DOCKER_OPTS="--insecure-registry 0.0.0.0:5000"
```

修改 /lib/systemd/system/docker.service

```
# 加入
EnvironmentFile=/etc/default/docker
# 尾部加入 $DOCKER_OPTS
ExecStart=/usr/bin/dockerd -H fd:// -H
unix:///var/run/docker.sock -H tcp://0.0.0.0:2375 $DOCKER_OPTS
```

完整的例子

```
neo@ubuntu:~$ sudo vim /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
EnvironmentFile=/etc/default/docker
# for containers run by docker
ExecStart=/usr/bin/dockerd -H fd:// -H
unix:///var/run/docker.sock -H tcp://0.0.0.0:2375 $DOCKER_OPTS
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
```

重启 Docker

```
neo@ubuntu:~$ sudo systemctl daemon-reload
```

```
neo@ubuntu:~$ sudo systemctl restart docker

neo@ubuntu:~$ ps ax | grep docker
19548 ?        Ssl    0:00 /usr/bin/dockerd -H fd:// -H
unix:///var/run/docker.sock -H tcp://0.0.0.0:2375 --insecure-
registry 0.0.0.0:5000
```

验证 5000 端口可以访问

```
neo@ubuntu:~$ curl -XGET http://localhost:5000/v2/_catalog
{"repositories":[]}
```

### 推送镜像到私有仓库

本地镜像推送到远程私有仓库

```
docker pull busybox
docker tag busybox docker.netkiller.cn:5000/busybox
docker push docker.netkiller.cn:5000/busybox
```

操作演示

```
[root@localhost ~]# docker pull busybox
Using default tag: latest
latest: Pulling from library/busybox
697743189b6d: Pull complete
Digest:
sha256:061ca9704a714ee3e8b80523ec720c64f6209ad3f97c0ff7cb9ec7d1
9f15149f
Status: Downloaded newer image for busybox:latest
```

```
[root@localhost ~]# docker tag busybox
docker.netkiller.cn:5000/busybox

[root@localhost ~]# docker push
docker.netkiller.cn:5000/busybox
The push refers to repository
[docker.netkiller.cn:5000/busybox]
adab5d09ba79: Pushed
latest: digest:
sha256:4415a904b1aca178c2450fd54928ab362825e863c0ad5452fd020e92
f7a6a47e size: 527
```

查看远程私有仓库

```
[root@localhost ~]# curl -XGET
http://docker.netkiller.cn:5000/v2/_catalog
{"repositories":["busybox"]}

[root@localhost ~]# curl -XGET
http://docker.netkiller.cn:5000/v2/busybox/tags/list
{"name":"busybox","tags":["latest"]}
```

从私有仓库拉镜像

```
docker pull docker.netkiller.cn:5000/busybox
```

### 查询镜像

http://localhost:5000/v2/_catalog

如果我们想要查询私有仓库中的所有镜像，使用docker search命令：

```
docker search registry_ipaddr:5000/
```

如果要查询仓库中指定账户下的镜像，则使用如下命令：

```
docker search registry_ipaddr:5000/account/
```

操作演示

```
[root@localhost ~]# curl -XGET
http://docker.netkiller.cn:5000/v2/_catalog
{"repositories":["busybox"]}

[root@localhost ~]# curl -XGET
http://docker.netkiller.cn:5000/v2/busybox/tags/list
{"name":"busybox","tags":["latest"]}
```

**registry 镜像高级配置**

/etc/docker/registry/config.yml

```
cat config.yml

version: 0.1
log:
  fields:
    service: registry
storage:
  delete:
    enabled: true
  cache:
    blobdescriptor: inmemory
  filesystem:
    rootdirectory: /var/lib/registry
http:
  addr: :5000
  headers:
    X-Content-Type-Options: [nosniff]
health:
  storagedriver:
    enabled: true
    interval: 10s
    threshold: 3
```

私有仓库认证

创建密码文件

```
docker run --entrypoint htpasswd registry -Bbn testuser
testpassword > auth/htpasswd
```

启动 docker

```
docker run -d -p 5000:5000 --restart=always --name docker-hub \
  -v /opt/registry:/var/lib/registry \
  -v /opt/auth:/auth \
  -e "REGISTRY_AUTH=htpasswd" \
  -e "REGISTRY_AUTH_HTPASSWD_REALM=Registry Realm" \
  -e REGISTRY_AUTH_HTPASSWD_PATH=/auth/htpasswd \
  registry
```

登录

```
docker login -u testuser -p testpassword
docker.netkiller.cn:5000
```

退出

```
docker logout docker.netkiller.cn:5000
```

### registry 接口

查看仓库 http://registry:5000/v2/_catalog

```
curl -XGET http://registry:5000/v2/_catalog
```

查看镜像

```
curl -XGET http://registry:5000/v2/image_name/tags/list
```

删除镜像

```
DELETE /v2/<name>/manifests/<reference>
name:镜像名称
reference: 镜像对应sha256值
```

处理器测试

```
curl -I -X DELETE
http://registry:5000/v2/netkiller/manifests/sha256:6a67ba482a8d
d4f8143ac96b1dcffa5e45af95b8d3e37aeba72401a5afd7ab8e
```

## 11.3. Harbor

Harbor 是 Vmware 公司开源的 企业级的 Docker Registry 管理项目，它提供 Dcoker Registry 管理 WebUI，可基于角色访问控制, AD/LDAP 集成，日志审核等功能，完全的支持中文。

开源项目地址 https://github.com/vmware/harbor

# 12. Swarms

Swarm 是一组运行着Docker的机器。经过这些配置后，将节点加入到一个集群中，你仍然像之前那样运行Docker命令一样管理集群上的容器。这些命令由swarm manager在集群上执行。这些机器可以是真实的机器，也可以是虚拟机。机器加入到一个swarm后，可以称这些机器为节点(node)。

## 12.1. 管理 Swarms

帮助命令

```
neo@MacBook-Pro ~ % docker-machine
Usage: docker-machine [OPTIONS] COMMAND [arg...]

Create and manage machines running Docker.

Version: 0.16.1, build cce350d7

Author:
  Docker Machine Contributors -
<https://github.com/docker/machine>

Options:
  --debug, -D                                                    Enable
debug mode
  --storage-path, -s "/Users/neo/.docker/machine"
Configures storage path [$MACHINE_STORAGE_PATH]
  --tls-ca-cert                                                  CA to
verify remotes against [$MACHINE_TLS_CA_CERT]
  --tls-ca-key                                                   Private
key to generate certificates [$MACHINE_TLS_CA_KEY]
  --tls-client-cert                                              Client
cert to use for TLS [$MACHINE_TLS_CLIENT_CERT]
  --tls-client-key                                               Private
key used in client TLS auth [$MACHINE_TLS_CLIENT_KEY]
  --github-api-token                                             Token
to use for requests to the Github API
```

```
[$MACHINE_GITHUB_API_TOKEN]
  --native-ssh                                                  Use the
native (Go-based) SSH implementation. [$MACHINE_NATIVE_SSH]
  --bugsnag-api-token                                           BugSnag
API token for crash reporting [$MACHINE_BUGSNAG_API_TOKEN]
  --help, -h                                                    show
help
  --version, -v                                                 print
the version

Commands:
  active                Print which machine is active
  config                Print the connection config for machine
  create                Create a machine
  env                   Display the commands to set up the
environment for the Docker client
  inspect               Inspect information about a machine
  ip                    Get the IP address of a machine
  kill                  Kill a machine
  ls                    List machines
  provision             Re-provision existing machines
  regenerate-certs      Regenerate TLS Certificates for a
machine
  restart               Restart a machine
  rm                    Remove a machine
  ssh                   Log into or run a command on a machine
with SSH.
  scp                   Copy files between machines
  mount                 Mount or unmount a directory from a
machine with SSHFS.
  start                 Start a machine
  status                Get the status of a machine
  stop                  Stop a machine
  upgrade               Upgrade a machine to the latest version
of Docker
  url                   Get the URL of a machine
  version               Show the Docker Machine version or a
machine docker version
  help                  Shows a list of commands or help for
one command

Run 'docker-machine COMMAND --help' for more information on a
command.
```

```
```

### 查看 Swarms 版本

```
neo@MacBook-Pro ~ % docker-machine version
docker-machine version 0.16.1, build cce350d7
```

### 初始化 Swarms

```
neo@MacBook-Pro ~/workspace/docker/docker-compose % docker
swarm init
Swarm initialized: current node (t8gqr7wfyeis9n8wuegy4j6gn) is
now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-
5w5joob510ug74m9vfn2j1a41nox3ddh6eiyrpgonm38zaoj5c-
bo2q6tdem9ihd68gryue1b42x 192.168.65.3:2377

To add a manager to this swarm, run 'docker swarm join-token
manager' and follow the instructions.
```

### 显示 join-token

```
neo@MacBook-Pro ~ % docker swarm join-token manager
To add a manager to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-
200v95u6lkow6wyxne1ll44rhhwy1zfvawnrqo39i44sqay8vp-
```

```
1vltkdz94y79mgech56wtnj9n 192.168.65.3:2377
```

## 创建虚拟机

使用VirtualBox驱动，创建虚拟机：

```
neo@MacBook-Pro ~ % docker-machine create --driver virtualbox
vm1
neo@MacBook-Pro ~ % docker-machine create --driver virtualbox
vm2
```

## 显示虚拟机列表

```
$ docker-machine ls
```

## 设置管理节点

配置虚拟机作为manager节点，用以执行管理命令并准许其他worker加入到swarm中。

```
$ docker-machine ssh vm1 "docker swarm init --advertise-addr
<ip_address>"
```

加入到管理节点

```
$ docker-machine ssh vm2 "docker swarm join \
--token <token> \
<ip>:2377"
```

查看节点列表

```
$ docker-machine ssh vm1 "docker node ls"
```

## 环境变量

```
$ docker-machine env vm1
```

现在运行docker-machine ls来验证vm1就是当前的活跃机器，会有星号标识：

```
$ docker-machine ls
```

## 切换节点

```
eval $(docker-machine env vm1)
```

重置 shell 环境

```
neo@MacBook-Pro ~ % docker-machine env -u
unset DOCKER_TLS_VERIFY
unset DOCKER_HOST
unset DOCKER_CERT_PATH
unset DOCKER_MACHINE_NAME
# Run this command to configure your shell:
# eval $(docker-machine env -u)
```

```
eval $(docker-machine env -u)
```

启动/停止节点

```
$ docker-machine start vm1
```

```
$ docker-machine stop vm1
```

离线

```
docker swarm leave --force
```

## 12.2. Stack

stack 是一组相互关联的services，这些services之间相互依赖，并能够一起进行编排和scale。单个stack就能够定义和协调整个应用程序的功能.

Stack 使用 docker-compose.yml 部署，Stack 与 docker-compose 的区别是，Stack 无法 build 镜像，不支持 v2会v1 版本的 docker-compose.yml

创建 docker-compose.yml

```
version: "3"
services:
  web:
    # replace username/repo:tag with your name and image
details
    image: nginx
    deploy:
      replicas: 5
      restart_policy:
        condition: on-failure
      resources:
        limits:
          cpus: "0.1"
          memory: 50M
    ports:
      - "80:80"
    networks:
      - webnet
  visualizer:
    image: dockersamples/visualizer:stable
    ports:
      - "8080:8080"
    volumes:
      - "/var/run/docker.sock:/var/run/docker.sock"
    deploy:
      placement:
        constraints: [node.role == manager]
    networks:
      - webnet
networks:
  webnet:
```

部署 docker-compose.yml

```
neo@MacBook-Pro ~ % docker stack deploy -c docker-compose.yml
visualizer
Creating service visualizer_web
Creating service visualizer_visualizer
```

查看部署

```
neo@MacBook-Pro ~ % docker stack ls
NAME                SERVICES            ORCHESTRATOR
visualizer          2                   Swarm
```

```
neo@MacBook-Pro ~ % docker stack services visualizer
ID                  NAME                    MODE
REPLICAS            IMAGE                             PORTS
h6vpdk8wqr8w        visualizer_visualizer   replicated
1/1                 dockersamples/visualizer:stable   *:8080-
>8080/tcp
tm5rre8d4kni        visualizer_web          replicated
5/5                 nginx:latest                      *:80-
>80/tcp
```

```
neo@MacBook-Pro ~ % docker stack ps visualizer
ID                  NAME                                 IMAGE
NODE                    DESIRED STATE       CURRENT STATE
ERROR                              PORTS
rnkgapj5oozr        visualizer_visualizer.1
dockersamples/visualizer:stable   linuxkit-025000000001
Running             Running 24 minutes ago
msstp0uavxpf         \_ visualizer_visualizer.1
dockersamples/visualizer:stable   linuxkit-025000000001
Shutdown            Rejected 31 minutes ago   "No such image:
dockersamples/…"
1jmhrzmlsy0j         \_ visualizer_visualizer.1
dockersamples/visualizer:stable   linuxkit-025000000001
Shutdown            Rejected 31 minutes ago   "No such image:
dockersamples/…"
p7iyq0147oh0         \_ visualizer_visualizer.1
dockersamples/visualizer:stable   linuxkit-025000000001
Shutdown            Rejected 31 minutes ago   "No such image:
dockersamples/…"
jdc7cx00a994         \_ visualizer_visualizer.1
dockersamples/visualizer:stable   linuxkit-025000000001
Shutdown            Rejected 32 minutes ago   "No such image:
dockersamples/…"
pttqpa4z21id        visualizer_web.1              nginx:latest
linuxkit-025000000001   Running             Running 30 minutes
ago
rappf97c8dtb        visualizer_web.2              nginx:latest
linuxkit-025000000001   Running             Running 30 minutes
ago
t3dcjqf0fsly        visualizer_web.3              nginx:latest
linuxkit-025000000001   Running             Running 30 minutes
ago
jtztvsqccb5d        visualizer_web.4              nginx:latest
linuxkit-025000000001   Running             Running 30 minutes
ago
ldb92uky85oc        visualizer_web.5              nginx:latest
linuxkit-025000000001   Running             Running 30 minutes
ago
```

```
neo@MacBook-Pro ~ % docker node ls
```

```
ID                              HOSTNAME                STATUS
AVAILABILITY        MANAGER STATUS      ENGINE VERSION
t8gqr7wfyeis9n8wuegy4j6gn *   linuxkit-025000000001   Ready
Active              Leader              18.09.2
```

```
neo@MacBook-Pro ~ % docker service ls
ID                  NAME                    MODE
REPLICAS            IMAGE                             PORTS
h6vpdk8wqr8w        visualizer_visualizer   replicated
1/1                 dockersamples/visualizer:stable   *:8080-
>8080/tcp
tm5rre8d4kni        visualizer_web          replicated
5/5                 nginx:latest                      *:80-
>80/tcp
```

```
neo@MacBook-Pro ~ % docker stack rm visualizer
Removing service visualizer_visualizer
Removing service visualizer_web
Removing network visualizer_webnet
```

## 12.3. 服务

```
neo@MacBook-Pro ~ % docker service

Usage:  docker service COMMAND

Manage services

Commands:
  create      Create a new service
  inspect     Display detailed information on one or more
```

```
services
  logs        Fetch the logs of a service or task
  ls          List services
  ps          List the tasks of one or more services
  rm          Remove one or more services
  rollback    Revert changes to a service's configuration
  scale       Scale one or multiple replicated services
  update      Update a service

Run 'docker service COMMAND --help' for more information on a
command.
```

### 创建 Service

```
$ docker service create \
  --replicas 10 \
  --name ping_service \
  alpine ping www.netkiller.cn
```

```
$ docker service create --replicas 1 --name my-prometheus \
    --mount
type=bind,source=/tmp/prometheus.yml,destination=/etc/prometheu
s/prometheus.yml \
    --publish published=9090,target=9090,protocol=tcp \
    prom/prometheus
```

```
iMac:redis neo$ docker stack deploy -c redis.yml redis
Creating service redis_redis
```

**提示**

    --mount 不允许使用相对路径，小技巧 `pwd`/prometheus.yml

```
docker service create --replicas 1 --name my-prometheus \
    --mount 
type=bind,source=`pwd`/prometheus.yml,destination=/etc/promethe
us/prometheus.yml \
    --publish published=9090,target=9090,protocol=tcp \
    prom/prometheus
```

## 删除 Service

```
iMac:docker neo$ docker service rm prometheus
prometheus
```

如果是 stack 部署的也可以这样删除

```
iMac:redis neo$ docker stack rm redis
Removing service redis_redis
```

## inspect

```
iMac:redis neo$ docker service inspect redis_redis
[
    {
        "ID": "kpqopqq10a2yi1rdecuf1246q",
        "Version": {
            "Index": 10148
        },
        "CreatedAt": "2020-09-26T14:19:53.920458941Z",
        "UpdatedAt": "2020-09-26T14:19:53.922204086Z",
        "Spec": {
            "Name": "redis_redis",
            "Labels": {
                "com.docker.stack.image": "redis:latest",
                "com.docker.stack.namespace": "redis"
            },
            "TaskTemplate": {
                "ContainerSpec": {
                    "Image":
"redis:latest@sha256:1cfb205a988a9dae5f025c57b92e9643ec0e7ccff6
e66bc639d8a5f95bba928c",
                    "Labels": {
                        "com.docker.stack.namespace": "redis",
                        "desktop.docker.io/mounts/0/Source":
"/Users/neo/workspace/docker/docker-compose/redis/redis.conf",

"desktop.docker.io/mounts/0/SourceKind": "hostFile",
                        "desktop.docker.io/mounts/0/Target":
"/etc/redis/redis.conf"
                    },
                    "Args": [
                        "entrypoint.sh",
                        "/etc/redis/redis.conf"
                    ],
                    "Hostname": "redis",
                    "Env": [
                        "TZ=Asia/Shanghai"
                    ],
                    "Privileges": {
                        "CredentialSpec": null,
                        "SELinuxContext": null
                    },
                    "Mounts": [
                        {
                            "Type": "bind",
                            "Source":
"/host_mnt/Users/neo/workspace/docker/docker-
compose/redis/redis.conf",
                            "Target": "/etc/redis/redis.conf"
                        },
                        {
                            "Type": "bind",
                            "Source": "/var/lib/redis",
                            "Target": "/var/lib/redis"
                        },
                        {
                            "Type": "bind",
                            "Source": "/var/log/redis",
                            "Target": "/var/log/redis"
                        }
                    ],
                    "StopGracePeriod": 10000000000,
                    "DNSConfig": {},
                    "Isolation": "default"
                },
                "Resources": {
                    "Limits": {
                        "NanoCPUs": 1000000000,
                        "MemoryBytes": 536870912
                    }
                },
                "RestartPolicy": {
                    "Condition": "any",
                    "Delay": 5000000000,
                    "MaxAttempts": 0
                },
                "Placement": {
                    "Platforms": [
                        {
                            "Architecture": "amd64",
                            "OS": "linux"
                        },
                        {
                            "OS": "linux"
                        },
                        {
                            "OS": "linux"
                        },
                        {
                            "Architecture": "arm64",
                            "OS": "linux"
                        },
                        {
                            "Architecture": "386",
                            "OS": "linux"
                        },
                        {
                            "Architecture": "mips64le",
                            "OS": "linux"
                        },
                        {
                            "Architecture": "ppc64le",
                            "OS": "linux"
                        },
                        {
                            "Architecture": "s390x",
                            "OS": "linux"
                        }
                    ]
                },
                "Networks": [
                    {
                        "Target": "gvcz5y66ovrlqfaxb02zx026t",
                        "Aliases": [
                            "redis"
                        ]
                    }
                ],
                "ForceUpdate": 0,
                "Runtime": "container"
            },
            "Mode": {
                "Replicated": {
                    "Replicas": 1
                }
            },
            "UpdateConfig": {
                "Parallelism": 1,
                "Delay": 5000000000,
                "FailureAction": "pause",
                "Monitor": 10000000000,
                "MaxFailureRatio": 0.1,
                "Order": "start-first"
            },
            "RollbackConfig": {
                "Parallelism": 1,
                "FailureAction": "pause",
                "Monitor": 5000000000,
                "MaxFailureRatio": 0,
                "Order": "stop-first"
            },
            "EndpointSpec": {
                "Mode": "vip",
                "Ports": [
                    {
                        "Protocol": "tcp",
                        "TargetPort": 6379,
                        "PublishedPort": 6379,
                        "PublishMode": "ingress"
                    }
                ]
            }
        },
        "Endpoint": {
            "Spec": {
                "Mode": "vip",
                "Ports": [
                    {
                        "Protocol": "tcp",
                        "TargetPort": 6379,
                        "PublishedPort": 6379,
                        "PublishMode": "ingress"
                    }
                ]
            },
            "Ports": [
                {
                    "Protocol": "tcp",
                    "TargetPort": 6379,
                    "PublishedPort": 6379,
                    "PublishMode": "ingress"
                }
            ],
            "VirtualIPs": [
                {
                    "NetworkID": "7r7k9robn0uuojuxl1es2wdds",
                    "Addr": "10.0.0.42/24"
                },
                {
                    "NetworkID": "gvcz5y66ovrlqfaxb02zx026t",
                    "Addr": "172.12.0.2/16"
                }
            ]
        }
    }
```

```
]
```

## 12.4. swarm 卷管理

swarm 不能使用 -v /mysite:/usr/share/nginx/html 挂载卷，系统会提示

```
unknown shorthand flag: 'v' in -v
See 'docker service create --help'.
```

### Host Volumes

```
$ docker service create --name nginx \
  --mount type=bind,source=`pwd`/static-
site,target=/usr/share/nginx/html \
  -p 80:80 nginx
```

### Named Volumes

```
$ docker service create --name nginx \
  --mount type=volume,source=web,target=/usr/share/nginx/html \
  -p 80:80 nginx
```

### 共享卷

创建 NFS 数据共享卷

```
docker volume create --driver local \
    --opt type=nfs4 \
    --opt o=addr=<NFS-Server>,rw \
    --opt device=:<Shared-Path> \
    share
```

创建服务副本

```
docker service create \
  --mount type=volume,source=<Volume-Name>,destination=
<Container-Path> \
  --replicas 2 \
  <Image>
```

# 13. docker-compose.yml 容器编排

本章节介绍如何定义 docker-compose.yml 文件

首先创建项目目录

```
mkdir docker
cd docker
vim     docker-compose.yml
```

## 13.1. 版本号

```
version: '3.8'
```

## 13.2. 镜像

image: mysql:5.7 表示使用 mysql:5.7 镜像, image: mysql:latest 表示 mysql 最新版

```
services:
   db:
     image: mysql:5.7
     volumes:
       - db_data:/var/lib/mysql
     restart: always
     environment:
       MYSQL_ROOT_PASSWORD: somewordpress
       MYSQL_DATABASE: wordpress
       MYSQL_USER: wordpress
```

```
      MYSQL_PASSWORD: wordpress
```

### 13.3. 容器名称

```
  prometheus:
    image: prom/prometheus
    container_name: prometheus
```

### 13.4. 启动策略

```
restart: unless-stopped
```

### 13.5. 容器用户

```
# Define in docker-compose:

 services:
  prometheus:
    image: prom/prometheus
    user: "1000:1000"

 services:
  prometheus:
    image: prom/prometheus
    user: root

# Dockerfile

USER 1000:1000
```

```
```

## 13.6. 挂在卷

```
    volumes:
      - db_data:/var/lib/mysql
```

## 13.7. 映射端口的标签

将容器中的端口暴漏给宿主主机。

```
ports:
 - "3000"
 - "80:80"
 - "22:22"
 - "127.0.0.1:8000:8000"
```

默认 "端口:端口" 将监听 127.0.0.1 主机。如果需要将端口暴漏出去，格式是"IP:PORT:PORT"，IP地址是宿主主机的网络适配器IP地址。

## 13.8. 添加 hosts 文件

往/etc/hosts文件中添加主机名，与Docker client的--add-host类似：

```
        extra_hosts:
                - "orderer.example.com:10.130.116.8"
                - "peer0.org1.example.com:10.130.116.9"
                - "peer1.org1.example.com:10.130.116.10"
                - "peer0.org2.example.com:10.130.116.25"
```

```
              - "peer1.org2.example.com:10.130.116.27"
```

## 13.9. 网络配置

### 自定义 IPv4 子网地址

```
version: '3.9'
networks:
  default:
    driver: bridge
    ipam:
      driver: default
      config:
        - subnet: 172.88.10.0/24
          gateway: 172.88.10.1
```

### external 外部网络

创建固定网段的网络bridge2。

```
docker network create --subnet=10.16.1.0/16 --gateway=10.16.1.1
--opt "com.docker.network.bridge.name"="bridge2" bridge2
```

把bridge2网络配置导docker-compose里面。

```
networks:
  default:
    driver: bridge
  persist:
    external:
      name: bridge2
```

**配置 IPv6**

```
networks:
    frontend:
        # use the bridge driver, but enable IPv6
        driver: bridge
        driver_opts:
            com.docker.network.enable_ipv6: "true"
        ipam:
            driver: default
            config:
                - subnet: 172.16.238.0/24
                gateway: 172.16.238.1
                - subnet: "2001:3984:3989::/64"
                gateway: "2001:3984:3989::1"
```

## 13.10. links 主机别名

links的作用是在当前服务里面创建一个链接外部服务的别名。

docker-compose.yml

```
services:
  tomcat:
    image: netkiller:latest
    links:
      - mysql:db.netkiller.cn
```

这时配置文件 application.properties 就可以这样些

```
sql.mysql.jdbc-url=jdbc:mysql://db.netkiller.cn:3306/test?
characterEncoding=utf8&serverTimezone=UTC&autoReconnect=true&us
eSSL=false
sql.mysql.username=root
sql.mysql.password=abcdef
sql.mysql.driverClassName=com.mysql.jdbc.Driver
```

## 13.11. 链接外部容器

创建 development 网络

```
docker network create development --driver bridge
docker run --name redis-external --net development -d redis
```

```
version: "3.9"
networks:
  default:
    external:
      name: development
services:
  demo-external:
    image: demo:1.0
    container_name: demo-external
    restart: always
    environment:
      REDIS_HOST: redis-external
    ports:
      - 80:80
    external_links:
      - redis-external
```

```
```

测试方法，进入 demo-external 容器，然后 ping redis-external 容器

```
docker exec -it demo-external ping redis-external
```

```
[root@netkiller docker]# docker exec -it demo-external ping
redis-external
PING redis-external (172.18.0.3) 56(84) bytes of data.
64 bytes from redis-external.development (172.18.0.3):
icmp_seq=1 ttl=64 time=0.091 ms
64 bytes from redis-external.development (172.18.0.3):
icmp_seq=2 ttl=64 time=0.122 ms
64 bytes from redis-external.development (172.18.0.3):
icmp_seq=3 ttl=64 time=0.185 ms
```

## 13.12. 服务依赖

通过depends_on告诉docker-compose当前服务启动之前先要把depends_on指定的服务启动起来才行。

```
services:
  kafka:
    image: tflinux_kafka
    depends_on:
      - zookeeper
  spring:
    image: springboot
    depends_on:
      - redis
      - mysql
```

## 13.13. working_dir

```
working_dir
```

## 13.14. 设置环境变量

environment 实现容器中环境变量的定义

```
version: '3'

networks:
  basic:

services:
  tools:
    container_name: tools
    image: hyperledger/fabric-tools
    tty: true
    environment:
      - GOPATH=/opt/gopath
      - CORE_VM_ENDPOINT=unix:///host/var/run/docker.sock
      - CORE_LOGGING_LEVEL=DEBUG
      - CORE_PEER_ID=cli
      - CORE_PEER_ADDRESS=peer0.org1.example.com:7051
      - CORE_PEER_LOCALMSPID=Org1MSP
      -
CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/
fabric/peer/crypto/peerOrganizations/org1.example.com/users/Adm
in@org1.example.com/msp
      - CORE_CHAINCODE_KEEPALIVE=10
    # working_dir:
/opt/gopath/src/github.com/hyperledger/fabric/peer
    working_dir: /root/netkiller
    command: /bin/bash
    volumes:
        - /var/run/:/host/var/run/
        - ~/netkiller:/root/netkiller
        - ./chaincode/:/opt/gopath/src/github.com/
        - 
./crypto:/opt/gopath/src/github.com/hyperledger/fabric/peer/cry
pto/
    networks:
        - basic
```

## 13.15. 临时文件系统

挂载临时目录到容器:

```
tmpfs: /run
tmpfs:
  - /run
  - /tmp
```

## 13.16. 编译 Dockerfile

编译当前目录下的 Dockerfile 使用 build: .

```
version: '3'
services:
  web:
    build: .
    ports:
     - "5000:5000"
```

指定镜像名称

```
version: "3.7"
services:
  redis-image:
    build:
      context: .
      dockerfile: Dockerfile
      args:
        - node=master
    image: netkiller/redis:latest
    container_name: redis
    restart: always
    ports:
      - "6379:6379"
    networks:
      - redis
    privileged: true
    sysctls:
      net.core.somaxconn: '511'
    ulimits:
      nproc: 65535
      nofile:
        soft: 65535
        hard: 65535
```

docker-compose build redis-image 构建镜像

```
neo@MacBook-Pro ~/workspace/docker/docker-compose/redis/cluster
% docker-compose build redis-image
Building redis-image
Step 1/12 : FROM redis:latest
 ---> a55fbf438dfd
Step 2/12 : ARG node
 ---> Using cache
 ---> 4deb8fc1e1df
Step 3/12 : ENV REDIS_PORT 6379
 ---> Using cache
 ---> 5723ff2fe55c
Step 4/12 : COPY redis.conf /etc/redis/redis.conf
```

```
 ---> Using cache
 ---> daf496f8c342
Step 5/12 : COPY docker-entrypoint.sh /usr/local/bin/
 ---> Using cache
 ---> 600ae3b0c059
Step 6/12 : RUN ln -sf /usr/share/zoneinfo/Asia/Shanghai
/etc/localtime
 ---> Using cache
 ---> 630e3813bc8f
Step 7/12 : RUN echo 'Asia/Shanghai' >/etc/timezone
 ---> Using cache
 ---> 7d48350d6621
Step 8/12 : RUN echo 'echo never >
/sys/kernel/mm/transparent_hugepage/enabled' > /etc/rc.local
 ---> Using cache
 ---> c096dc75da72
Step 9/12 : RUN chmod +rw /etc/redis/redis.conf
 ---> Using cache
 ---> 25d8b0ac8893
Step 10/12 : EXPOSE $REDIS_PORT
 ---> Using cache
 ---> 99f31a88d2ff
Step 11/12 : ENTRYPOINT ["/usr/local/bin/docker-entrypoint.sh"]
 ---> Using cache
 ---> ef98f89610ae
Step 12/12 : CMD [ "redis-server", "/etc/redis/redis.conf" ]
 ---> Using cache
 ---> 095823650068

Successfully built 095823650068
Successfully tagged netkiller/redis:latest

neo@MacBook-Pro ~/workspace/docker/docker-compose/redis/cluster
% docker images | grep netkiller/redis
netkiller/redis                                   latest
095823650068        8 minutes ago       95MB
```

## 13.17. resources 硬件资源分配

```


version: "3"
```

```
services:
  node:
    build:
     context: .
      dockerfile: ./Dockerfile
    restart: always
    environment:
      - HOST=localhost
    volumes:
      - logs:/app/logs
    expose:
      - 8080
    deploy:
      resources:
        limits:
          cpus: '0.001'
          memory: 50M
        reservations:
          cpus: '0.0001'
          memory: 20M
```

**提示**

注意：启动必须加入 `--compatibility` 选项

```
docker-compose --compatibility up
```

# 14. Docker Example

## 14.1. registry

```
docker run -d -p 5000:5000 --name registry registry:latest
```

### Auth + SSL

```
iMac:registry neo$ mkdir etc
iMac:registry neo$ htpasswd -Bbn neo chen > etc/htpasswd

or

docker run --entrypoint htpasswd registry:2 -Bbn neo passw0rd >
etc/htpasswd
```

```

```

```
docker run -d \
  --restart=always \
  --name registry \
  -v `pwd`/etc:/usr/local/etc \
  -e "REGISTRY_AUTH=htpasswd" \
  -e "REGISTRY_AUTH_HTPASSWD_REALM=Registry Realm" \
  -e REGISTRY_AUTH_HTPASSWD_PATH=/usr/local/etc/htpasswd \
  -e REGISTRY_HTTP_ADDR=0.0.0.0:443 \
  -e REGISTRY_HTTP_TLS_CERTIFICATE=/usr/local/etc/domain.cer \
  -e REGISTRY_HTTP_TLS_KEY=/usr/local/etc/domaon.key \
  -p 443:443 \
  registry:2
```

## 14.2. Example Java - Spring boot with Docker

### 获取 CentOS 7 镜像

docker pull centos:7

```
# docker pull centos:7
7: Pulling from library/centos
343b09361036: Pull complete
Digest:
sha256:bba1de7c9d900a898e3cadbae040dfe8a633c06bc104a0df76ae2448
3e03c077
Status: Downloaded newer image for centos:7
```

基于 CentOS 7 运行一个容器

docker run -it --name mycentos docker.io/centos:7 /bin/bash

```
# docker run -it --name mycentos docker.io/centos:7 /bin/bash
```

运行后直接进入了容器的shell控制台默认是bash

### 安装 openjdk

```
# yum install -y java-1.8.0-openjdk

# cat >> /etc/profile.d/java.sh <<'EOF'
export JAVA_HOME=/usr/java/default
export JAVA_OPTS="-server -Xms2048m -Xmx4096m -
Djava.io.tmpdir=/tmp -Djava.security.egd=file:/dev/./urandom -
Dfile.encoding=UTF8 -Duser.timezone=GMT+08"
export CLASSPATH=$JAVA_HOME/lib:$JAVA_HOME/jre/lib:.
export PATH=$PATH:$JAVA_HOME/bin:$JAVA_HOME/jre/bin:
EOF

# source /etc/profile.d/java.sh
```

检查Java是否安装成功

```
# whereis java
java: /usr/bin/java /usr/lib/java /etc/java /usr/share/java
/usr/share/man/man1/java.1.gz

# java -version
openjdk version "1.8.0_131"
OpenJDK Runtime Environment (build 1.8.0_131-b11)
OpenJDK 64-Bit Server VM (build 25.131-b11, mixed mode)
```

创建应用程序目录

```
# mkdir -p /www/netkiller.cn/www.netkiller.cn/
```

推出当前容器

```
# exit
```

**Spring boot 包**

复制 jar 文件到Docker容器

```
docker cp /www/netkiller.cn/www.netkiller.cn/www.netkiller.cn-
0.0.1.war mycentos:/usr/local/libexec
```

**启动 Spring boot 项目**

启动容器

```
# docker start mycentos
mycentos
```

进入容器

```
# docker exec -it mycentos /bin/bash
```

如果仅仅是测试可以手动启动 Srping boot 项目

```
# cat >> /root/run.sh <<EOF
java -server -Xms2048m -Xmx8192m -jar
/usr/local/libexec/www.netkiller.cn-0.0.1.war
EOF

chmod u+x /root/run.sh
```

生产环境请使用启动脚本

```
# curl -s
https://raw.githubusercontent.com/oscm/build/master/Application
/Spring/service/springbootd -o /etc/init.d/springbootd
# chmod +x /etc/init.d/springbootd
```

编辑启动脚本 /etc/init.d/springbootd 修改下面配置项

```
##############################################
BASEDIR="/www/netkiller.cn/api.netkiller.cn"
JAVA_HOME=/srv/java
JAVA_OPTS="-server -Xms2048m -Xmx8192m -
Djava.security.egd=file:/dev/./urandom"
PACKAGE="api.netkiller.cn-0.0.2-release.jar"
CONFIG="--
spring.config.location=$BASEDIR/application.properties -
Dspring.profiles.active=production -Dserver.port=8080 -
Dlog.level=info"
USER=www
##############################################
NAME=springbootd
PROG="$JAVA_HOME/bin/java $JAVA_OPTS -jar $BASEDIR/$PACKAGE
$CONFIG"
LOGFILE=/var/tmp/$NAME.log
PIDFILE=/var/tmp/$NAME.pid
ACCESS_LOG=/var/tmp/$NAME.access.log
##############################################
```

你也可以使用 systemd 启动脚本，详见《Netkiller Java 手札》

### 基于 CentOS 7 制作 spring 镜像

docker commit mycentos springboot:1

```
# docker commit mycentos springboot:1
sha256:757d92d642d1b5a7b244f6ddf89f24a8d463d154438651c83ba51a64
4b401782
```

启动 spring boot 容器

```
# docker run -d --name springboot -p 80:8080 springboot:1
/root/run.sh
```

```
-d: 以守护进程方式启动
--name: 指定容器的名称
-p: 映射容器8080端口到宿主机的80端口
springboot:1 : 上一步制作好的springboot镜像,版本号为1
```

启动容器

```
# docker start springboot
```

停止容器

```
# docker stop springboot
```

## 14.3. Redis

http://download.redis.io/redis-stable/redis.conf

http://download.redis.io/redis-stable/sentinel.conf

### Docker 命令

#### 获取 Redis 镜像

docker pull redis

```
# docker pull redis
Using default tag: latest
latest: Pulling from library/redis
10a267c67f42: Pull complete
5b690bc4eaa6: Pull complete
4cdd94354d2a: Pull complete
71c1f30d820f: Pull complete
c54584150374: Pull complete
d1f9221193a6: Pull complete
d45bc46b48e4: Pull complete
Digest:
sha256:548a75066f3f280eb017a6ccda34c561ccf4f25459ef8e36d6ea582b
6af1decf
Status: Downloaded newer image for redis:latest
```

#### 启动一个 Redis 实例

```
# docker run --name my-redis -d redis
10207174e18f61290f9c869e6437fa787e459e07b076b82cedf800a8c37c515
d
```

查看启动情况

```
# docker ps
CONTAINER ID        IMAGE               COMMAND
CREATED             STATUS              PORTS
NAMES
10207174e18f        redis               "docker-entrypoint..."
8 minutes ago       Up 8 minutes        6379/tcp            my-
redis
```

进入 **Redis**

```
# docker run -it --link my-redis:redis --rm redis redis-cli -h
redis -p 6379
redis:6379> set name neo
OK
redis:6379> get name
"neo"
redis:6379> exit
```

启动一个 **Redis** 实例并映射 **6379** 端口

```
# docker stop my-redis
my-redis

# docker rm my-redis
my-redis

# docker run --name my-redis -d -p 6379:6379 redis
10207174e18f61290f9c869e6437fa787e459e07b076b82cedf800a8c37c515
d

# docker ps -a
```

```
CONTAINER ID        IMAGE               COMMAND
CREATED             STATUS              PORTS
NAMES
1c4540d8617f        redis               "docker-entrypoint..."
2 seconds ago       Up 1 second         0.0.0.0:6379->6379/tcp
my-redis
```

检查端口

```
# ss -lnt | grep 6379
LISTEN     0      128         :::6379                    :::*
```

维护容器

使用下面命令进入容器维护 Redis

```
# docker exec -it my-redis /bin/bash
root@1c4540d8617f:/data#

root@1c4540d8617f:/data# redis-server -v
Redis server v=3.2.9 sha=00000000:0 malloc=jemalloc-4.0.3
bits=64 build=a30533b464d1689b
```

## Docker compose

```
version: "3.7"
services:
  redis:
    image: redis:latest
    container_name: redis
    ports:
      - "6379:6379"
    volumes:
      - redis_data:/var/lib/redis
    restart: always
    networks:
      - dev

networks:
  dev:
    driver: bridge

volumes:
  redis_data:
```

```
version: '3.9'

services:
  redis:
    image: redis:alpine
    container_name: redis
    restart: always
    hostname: redis.netkiller.cn
    user: redis:redis
    privileged: true
    environment:
      - TZ=Asia/Shanghai
      - LANG=en_US.UTF-8
    ports:
      - 6379:6379
    volumes:
      - ./conf/redis.conf:/etc/redis.conf
      - redis:/var/lib/redis
      - ./logs:/var/log/redis
    entrypoint: redis-server /etc/redis.conf
    command:
      --requirepass passw0rd
volumes:
  redis:
```

```
```

确认配置生效

```
neo@MacBook-Pro-Neo ~ % docker exec -it redis redis-cli -a
passw0rd
Warning: Using a password with '-a' or '-u' option on the
command line interface may not be safe.
127.0.0.1:6379> config get dir
1) "dir"
2) "/var/lib/redis"
127.0.0.1:6379>
```

## Docker Stack

```
version: '3.8'

services:
  redis:
    image: redis:latest
    environment:
      - TZ=Asia/Shanghai
    hostname: redis
    ports:
      - 6379:6379
    networks:
      - test
    volumes:
      - data:/var/lib/redis
    configs:
      - source: config
        target: /usr/local/etc/redis.conf
        mode: 0440
    deploy:
      replicas: 1
      restart_policy:
        condition: on-failure
      resources:
        limits:
          cpus: "1"
          memory: 512M
      update_config:
        parallelism: 1
        delay: 5s
        monitor: 10s
        max_failure_ratio: 0.1
        order: start-first

configs:
  config:
    file: ./redis.conf

volumes:
  data:

networks:
   test:
     driver: overlay
```

下载 配置文件 https://redis.io/topics/config

```
iMac:redis neo$ curl -sO
https://raw.githubusercontent.com/redis/redis/6.0/redis.conf
iMac:redis neo$ egrep -v "^#|^$" redis.conf
```

修改配置文件

```
bind 0.0.0.0
logfile "/var/log/redis/redis.log"
dir /var/lib/redis
appendonly yes
```

创建 Docker 网络

```
iMac:redis neo$ docker network create \
>   --driver=overlay \
>   --subnet=172.12.0.0/16 \
>   --ip-range=172.12.0.0/16 \
>   --gateway=172.12.0.1 \
>   --attachable \
>   test
gvcz5y66ovrlqfaxb02zx026t

iMac:redis neo$ docker network ls
NETWORK ID          NAME                DRIVER
SCOPE
786efe30f42d        bridge              bridge
local
51e2b21d7daa        docker_gwbridge     bridge
local
96ba0de26cd2        host                host
local
7r7k9robn0uu        ingress             overlay
swarm
cbf078a5f121        none                null
local
d851mrlkludv        redis_default       overlay
swarm
q0h9awx86ef4        registry_default    overlay
swarm
cf585ea9ceb4        registry_default    bridge
local
gvcz5y66ovrl        test                overlay
swarm

iMac:redis neo$ docker stack deploy -c redis.yml redis
Creating network redis_default
Creating service redis_redis
```

查看服务

```
iMac:redis neo$ docker service ls
ID                   NAME                 MODE
REPLICAS             IMAGE                PORTS
1ti2ndlpdhm8         redis_redis          replicated          0/1
redis:latest         *:6379->6379/tcp
1w6xjrl0sn88         registry_registry    replicated          1/1
registry:latest      *:5000->5000/tcp
```

查看容器运行状态

```
iMac:redis neo$ docker container ls
CONTAINER ID         IMAGE                COMMAND
CREATED              STATUS               PORTS
NAMES
8407fd8fe66b         redis:latest         "docker-entrypoint.s…"
29 seconds ago       Up 29 seconds        6379/tcp
redis_redis.1.6fpqt3pdti03j9swn3x04ob9n
```

**somaxconn/overcommit_memory**

redis 日志

```
1:C 09 Aug 2021 15:13:20.270 # oO0OoO0OoO0Oo Redis is starting
oO0OoO0OoO0Oo
1:C 09 Aug 2021 15:13:20.270 # Redis version=6.2.5, bits=64,
commit=00000000, modified=0, pid=1, just started
1:C 09 Aug 2021 15:13:20.270 # Configuration loaded
1:M 09 Aug 2021 15:13:20.270 * monotonic clock: POSIX
clock_gettime
1:M 09 Aug 2021 15:13:20.270 * Running mode=standalone,
port=6379.
1:M 09 Aug 2021 15:13:20.270 # WARNING: The TCP backlog setting
of 511 cannot be enforced because /proc/sys/net/core/somaxconn
is set to the lower value of 128.
1:M 09 Aug 2021 15:13:20.270 # Server initialized
1:M 09 Aug 2021 15:13:20.270 # WARNING overcommit_memory is set
to 0! Background save may fail under low memory condition. To
fix this issue add 'vm.overcommit_memory = 1' to
/etc/sysctl.conf and then reboot or run the command 'sysctl
vm.overcommit_memory=1' for this to take effect.
1:M 09 Aug 2021 15:13:20.271 * Ready to accept connections
```

宿主主机上配置如下

```
[root@localhost ~]# cat >> /etc/sysctl.conf <<EOF

# Redis
net.core.somaxconn = 1024
vm.overcommit_memory=1
EOF
```

docker-compose.yml 中设置 net.core.somaxconn

```
[root@localhost redis]# cat docker-compose.yml
version: '3.9'

services:
  redis:
    image: redis:alpine
    container_name: redis
    restart: always
    hostname: redis.netkiller.cn
    user: redis:redis
    environment:
      - TZ=Asia/Shanghai
      - LANG=en_US.UTF-8
    ports:
      - 6379:6379
    volumes:
      - redis:/data
    sysctls:
      - net.core.somaxconn=511
    command:
      --logfile /data/redis.log
      --requirepass passw0rd
      --appendonly yes
volumes:
  redis:
```

## 14.4. Nginx

本例子使用 alpine 版本

**nginx:latest**

过程 1.1.

1.
   ```
   [root@iZj6ciilv2rcpgauqg2uuwZ]~# docker pull nginx
   Using default tag: latest
   latest: Pulling from library/nginx
   Digest:
   sha256:41ad9967ea448d7c2b203c699b429abe1ed5af331cd92533900c
   6d77490e0268
   Status: Image is up to date for nginx:latest
   ```

2. 启动容器

   ```
   docker run --name my-nginx-container -p 80:80 -d nginx
   ```

上面不能满足生产环境的需求，通常不会将数据放在容器中，我的做法如下。

```
docker rm my-nginx-container -f
docker run --name my-nginx-container \
-v /srv/nginx/nginx.conf:/etc/nginx/nginx.conf:ro \
-v /srv/nginx/conf.d:/etc/nginx/conf.d:ro \
-v /var/log/nginx:/var/log/nginx:rw \
-v /www:/www:ro \
-p 80:80 -d nginx
docker ps
```

## 安装 **Docker Nginx alpine**

过程 1.2. Docker nginx

1. 获取镜像

```
# docker pull nginx:alpine
```

2. 运行容器

```
docker run --name my-nginx-container -v
/srv/nginx:/etc/nginx:ro -v /www:/www:ro -p 80:80 -d
nginx:alpine
```

3. 
```
docker exec -it my-nginx-container /bin/bash
```

安装依赖工具

```
apt update -y && apt install -y procps iproute2
```

容器内优雅重启

首先观察一个现象，打开 linux 终端窗口，查看 nginx 进程。

```
[root@localhost ~]# ps ax | grep nginx
   6670 ?        Ss     0:00 nginx: master process
/usr/sbin/nginx
   6671 ?        S      0:00 nginx: worker process
   6672 ?        S      0:00 nginx: worker process
   6673 ?        S      0:00 nginx: worker process
   6674 ?        S      0:00 nginx: worker process
   9396 pts/0    S+     0:00 grep --color=auto nginx
```

6670 ~ 6674 都是 nginx 的进程，其中 6670 nginx: master process /usr/sbin/nginx 是父进程，用于监听 80/443 端口。6671 ～ 6674 nginx: worker process 是子进程，每个进程中又产生多线程，每个线程对应一次用户TCP请求。

6671 ~ 6674 子进程的进程ID会变化，而 6670 是不变的。 6670 父进程可以接收操作系统传递过来的信号（不懂信号的同学请恶补，信号，共享内存，管道，Socket 可以实现进程间通信），也就是我们可以告诉正在运行的进程，现在要干什么。

给 6670 进程发送 HUP 信号，nginx 就会重新读取配置文件，刷新缓存，此时 6671～6674不受影响，会继续为用户体统TCP链接服务，直到都安全Close为止。此时 6670 父进程已经完成配置的更新，6671～6674 也完成了它的使命，下一次新用户过来 nginx 就会创建新的进程，这个过程是无缝的，用户感知不到，80/443 端口始终提供服务，不会有任何用户出现中断链接的情况。

现在来演示一下，执行 reload 就会刷新配置文件，清空缓存，同时会将闲置的 nginx: worker process 关闭，并开启新的子进程。

```
[root@localhost ~]# systemctl reload nginx
[root@localhost ~]# ps ax | grep nginx
   6670 ?        Ss     0:00 nginx: master process
/usr/sbin/nginx
   6671 ?        S      0:01 nginx: worker process is shutting
down
   9403 ?        S      0:00 nginx: worker process
   9404 ?        S      0:00 nginx: worker process
   9405 ?        S      0:00 nginx: worker process
   9406 ?        S      0:00 nginx: worker process
   9408 pts/0    S+     0:00 grep --color=auto nginx
```

现在我们可以看到子进程ID的变化，9403 ～ 9406。父进程 nginx: master process /usr/sbin/nginx 的ID仍然是 6670

现在是容器中实现上面的 reload 操作。

```
[root@localhost ~]# cat docker-compose.yml
version: '3.9'
services:
  nginx:
    container_name: nginx
    restart: always
    image: nginx:latest
    ports:
      - 192.168.30.11:80:80
      - 192.168.30.11:443:443
```

```
[root@localhost ~]# docker-compose up
Starting nginx ... done
Attaching to nginx
nginx    | /docker-entrypoint.sh: /docker-entrypoint.d/ is not
empty, will attempt to perform configuration
nginx    | /docker-entrypoint.sh: Looking for shell scripts in
/docker-entrypoint.d/
nginx    | /docker-entrypoint.sh: Launching /docker-
entrypoint.d/10-listen-on-ipv6-by-default.sh
nginx    | 10-listen-on-ipv6-by-default.sh: info: IPv6 listen
already enabled
nginx    | /docker-entrypoint.sh: Launching /docker-
entrypoint.d/20-envsubst-on-templates.sh
nginx    | /docker-entrypoint.sh: Launching /docker-
entrypoint.d/30-tune-worker-processes.sh
nginx    | /docker-entrypoint.sh: Configuration complete; ready
for start up
nginx    | 2021/07/12 20:55:41 [notice] 1#1: using the "epoll"
event method
nginx    | 2021/07/12 20:55:41 [notice] 1#1: nginx/1.21.1
nginx    | 2021/07/12 20:55:41 [notice] 1#1: built by gcc 8.3.0
(Debian 8.3.0-6)
nginx    | 2021/07/12 20:55:41 [notice] 1#1: OS: Linux 4.18.0-
315.el8.x86_64
nginx    | 2021/07/12 20:55:41 [notice] 1#1:
getrlimit(RLIMIT_NOFILE): 1048576:1048576
nginx    | 2021/07/12 20:55:41 [notice] 1#1: start worker
processes
nginx    | 2021/07/12 20:55:41 [notice] 1#1: start worker
process 24
nginx    | 2021/07/12 20:55:41 [notice] 1#1: start worker
process 25
nginx    | 2021/07/12 20:55:41 [notice] 1#1: start worker
process 26
nginx    | 2021/07/12 20:55:41 [notice] 1#1: start worker
process 27
```

```
[root@localhost ~]# docker exec -it nginx bash
root@2d2637a6ac4d:/# ps ax
    PID TTY      STAT   TIME COMMAND
      1 ?        Ss     0:00 nginx: master process nginx -g
daemon off;
     24 ?        S      0:00 nginx: worker process
     25 ?        S      0:00 nginx: worker process
     26 ?        S      0:00 nginx: worker process
     27 ?        S      0:00 nginx: worker process
    623 pts/0    Ss     0:00 bash
    629 pts/0    R+     0:00 ps ax
root@2d2637a6ac4d:/#
```

reload nginx

```
[root@localhost ~]# docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED
STATUS         PORTS
NAMES
2d2637a6ac4d   nginx:latest   "/docker-entrypoint.…"   25
minutes ago   Up 5 minutes   192.168.30.11:80->80/tcp,
192.168.30.11:443->443/tcp   nginx
[root@localhost ~]# docker container exec nginx  nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is
ok
nginx: configuration file /etc/nginx/nginx.conf test is
successful
[root@localhost ~]# docker container exec nginx  nginx -s
reload
2021/07/12 21:01:41 [notice] 636#636: signal process started
```

再次查看进程

```
[root@localhost ~]# docker exec -it nginx bash
root@2d2637a6ac4d:/# ps ax
    PID TTY       STAT   TIME COMMAND
      1 ?         Ss     0:00 nginx: master process nginx -g
daemon off;
     24 ?         S      0:00 nginx: worker process
     25 ?         S      0:00 nginx: worker process
     26 ?         S      0:00 nginx: worker process
     27 ?         S      0:00 nginx: worker process
    623 pts/0     Ss     0:00 bash
    629 pts/0     R+     0:00 ps ax

root@2d2637a6ac4d:/# ps ax
    PID TTY       STAT   TIME COMMAND
      1 ?         Ss     0:00 nginx: master process nginx -g
daemon off;
    623 pts/0     Ss     0:00 bash
    642 ?         S      0:00 nginx: worker process
    643 ?         S      0:00 nginx: worker process
    644 ?         S      0:00 nginx: worker process
    645 ?         S      0:00 nginx: worker process
    646 pts/0     R+     0:00 ps ax
```

## 14.5. MySQL

```
sudo mkdir -p /opt/mysql/{data,mysql.d,docker-entrypoint-
initdb.d}
```

docker-compose.yaml

```
version: '3'

services:
  mysql:
    # 镜像名
    image: mysql:latest
    # 容器名
    container_name: mysql
    # 重启策略
    restart: always
    hostname: db.netkiller.cn
    environment:
      # 时区上海
      TZ: Asia/Shanghai
      # root 密码
      MYSQL_ROOT_PASSWORD: test
      # 初始化数据库
      MYSQL_DATABASE: test
      # 初始普通化用户
      MYSQL_USER: test
      # 用户密码
      MYSQL_PASSWORD: test
      # 映射端口
    ports:
      - 3306:3306
    volumes:
      # 挂载数据
      - ./mysql/data/:/var/lib/mysql/
      # 挂载配置
      - ./mysql/conf.d/:/etc/mysql/conf.d/
      # 挂载初始化目录
      - ./mysql/docker-entrypoint-initdb.d/:/docker-
entrypoint-initdb.d/
    command:
      --default-authentication-plugin=mysql_native_password
      --character-set-server=utf8mb4
      --collation-server=utf8mb4_general_ci
      --explicit_defaults_for_timestamp=true
      --lower_case_table_names=1
```

登陆测试

```
neo@MacBook-Pro-Neo ~ % docker exec -it mysql mysql -uroot -
```

```
ptest
mysql: [Warning] Using a password on the command line interface
can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 14
Server version: 8.0.25 MySQL Community Server - GPL

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

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>
```

## 14.6. MongoDB

```
$ docker run -d --network some-network --name mongo \
        -e MONGO_INITDB_DATABASE=test \
    -e MONGO_INITDB_ROOT_USERNAME=admin \
    -e MONGO_INITDB_ROOT_PASSWORD=secret \
    mongo

$ docker run -it --rm --network some-network mongo \
    mongo --host mongo \
        -u admin \
        -p secret \
        --authenticationDatabase admin \
        test
> db.getName();
test
```

### 使用 mongodb 用户运行

```
version: '3.9'
services:
  mongodb:
    image: mongo:latest
    container_name: mongo
    hostname: mongo.netkiller.cn
    restart: always
    user: mongodb:mongodb
    privileged: false
    volumes:
      - ./data:/data
    ports:
      - 27017:27017
    environment:
      TZ: Asia/Shanghai
      LANG: en_US.UTF-8
      MONGO_INITDB_DATABASE: admin
      MONGO_INITDB_ROOT_USERNAME: admin
      MONGO_INITDB_ROOT_PASSWORD: admin
    entrypoint: docker-entrypoint.sh mongod
    command:
      --logpath /data/mongod.log
```

```
[www@testing ~]$ sudo  cat /var/log/mongodb/mongod.log | grep
'"W"'
{"t":{"$date":"2021-08-13T19:54:20.219+08:00"},"s":"W",
"c":"ASIO",     "id":22601,   "ctx":"main","msg":"No
TransportLayer configured during NetworkInterface startup"}
{"t":{"$date":"2021-08-13T19:54:20.227+08:00"},"s":"W",
"c":"ASIO",     "id":22601,   "ctx":"main","msg":"No
TransportLayer configured during NetworkInterface startup"}
{"t":{"$date":"2021-08-13T19:54:20.851+08:00"},"s":"W",
"c":"CONTROL",  "id":22178,
"ctx":"initandlisten","msg":"/sys/kernel/mm/transparent_hugepag
e/enabled is 'always'. We suggest setting it to
'never'","tags":["startupWarnings"]}
{"t":{"$date":"2021-08-13T20:01:12.470+08:00"},"s":"W",
"c":"ASIO",     "id":22601,   "ctx":"main","msg":"No
TransportLayer configured during NetworkInterface startup"}
{"t":{"$date":"2021-08-13T20:01:12.478+08:00"},"s":"W",
"c":"ASIO",      "id":22601,   "ctx":"main","msg":"No
TransportLayer configured during NetworkInterface startup"}
{"t":{"$date":"2021-08-13T20:01:13.085+08:00"},"s":"W",
"c":"CONTROL",  "id":22178,
"ctx":"initandlisten","msg":"/sys/kernel/mm/transparent_hugepag
e/enabled is 'always'. We suggest setting it to
'never'","tags":["startupWarnings"]}
```

```
[root@testing ~]# docker exec -it mongo bash
root@mongo:/# cat /sys/kernel/mm/transparent_hugepage/enabled
[always] madvise never
root@mongo:/# cat /sys/kernel/mm/transparent_hugepage/defrag
always defer defer+madvise [madvise] never
```

```
root@mongo:/# echo never >
/sys/kernel/mm/transparent_hugepage/defrag
bash: /sys/kernel/mm/transparent_hugepage/defrag: Read-only
file system
```

```
[root@testing ~]# if test -f
/sys/kernel/mm/transparent_hugepage/enabled; then
>   echo never > /sys/kernel/mm/transparent_hugepage/enabled
> fi

[root@testing ~]# cat
/sys/kernel/mm/transparent_hugepage/enabled
always madvise [never]

[root@testing ~]# docker exec -it mongo bash
root@mongo:/# cat /sys/kernel/mm/transparent_hugepage/defrag
always defer defer+madvise [madvise] never
```

```
root@mongo:/# cat /sys/kernel/mm/transparent_hugepage/enabled
always madvise [never]
root@mongo:/# exit
exit
```

解决方案 /etc/rc.local 中加入下面脚本，CentOS 8 Stream 开启 rc.local 请参考《Netkiller Linux 手札》

```
cat <<'EOF'>> /etc/rc.local

if test -f /sys/kernel/mm/transparent_hugepage/enabled; then
        echo never >
/sys/kernel/mm/transparent_hugepage/enabled
fi

if test -f /sys/kernel/mm/transparent_hugepage/defrag; then
        echo never > /sys/kernel/mm/transparent_hugepage/defrag
fi
EOF
```

```
[root@testing ~]# systemctl restart rc-local
```

## 14.7. Node

```
version: '3.9'
services:
  node:
    image: node:latest
    container_name: node
    restart: always
    hostname: node.netkiller.cn
    extra_hosts:
      - db.netkiller.cn:192.168.10.5
      - redis.netkiller.cn:192.168.10.12
    environment:
      TZ: Asia/Shanghai
    ports:
      - 7777:7777
    volumes:
      -
/opt/netkiller.cn/www.netkiller.cn:/opt/netkiller.cn/www.netkil
ler.cn
    working_dir: /opt/netkiller.cn/www.netkiller.cn
    entrypoint: node /opt/netkiller.cn/www.netkiller.cn/main.js
```

# 15. Docker FAQ

## 15.1. 通过 IP 找容器

已知 IP 172.17.0.66 我们希望知道那个容器在使用该 IP 地址。

```
$ docker network inspect 50ddb92f378e | grep -A2 -B4  '0\.66'

"b8f2b71e5715972c910f0876a89dbd9b7000d8fb77580206091e982b2119c47b": {
                "Name": "nginx",
                "EndpointID":
"b7a3aea20619489def16f410c54ed5d857f8cd2062f2c66972f6341de8174ed8",
                "MacAddress": "02:42:ac:11:00:42",
                "IPv4Address": "172.17.0.66/16",
                "IPv6Address": ""
            },
```

## 15.2. 常用工具

查看出口IP地址

```
root@production:~# curl icanhazip.com
root@production:~# curl -4 icanhazip.com
root@production:~# curl -6 icanhazip.com

root@production:~# curl api.ipify.org
root@production:~# curl bot.whatismyipaddress.com
```

**Debian/Ubuntu**

## 15.3. 检查 Docker 是否可用

```
docker -v
docker run ubuntu /bin/echo hello world
docker stop $(docker ps -a -q)
docker rm $(docker ps -a -q)
docker rmi $(docker images -q)
```

## 15.4. Bitnami

https://github.com/bitnami

# 第 2 章 Podman

## 1. 安装 Podman

### 1.1. RockyLinux 安装 Podman

某些 Redhat 家族的 Linux 是自带 Podman，例如 Almalinux 9.0，RockyLinux没有自带 podman 需要自己安装，是方法执行下面的命令

```
[root@netkiller ~]# dnf install -y podman
```

### 1.2. Almalinux 9.0

Almalinux 9.0 自带 podman

```
systemctl enable podman
```

### 1.3. MacOS 安装 Podman

MacOS 安装方法

```
brew install podman
```

### 1.4. 初始化 Podman

初始化，启动 Podman

```
podman machine init
podman machine start
```

操作演示

```
neo@MacBook-Pro-M2 ~ % podman machine init
Downloading VM image: fedora-coreos-37.20221127.2.0-
qemu.aarch64.qcow2.xz: done
Extracting compressed file Image resized.
Machine init complete
To start your machine run:

    podman machine start

neo@MacBook-Pro-M2 ~ % podman machine start
Starting machine "podman-machine-default"
Waiting for VM ...
Mounting volume... /Users/neo:/Users/neo

This machine is currently configured in rootless mode. If your
containers
require root permissions (e.g. ports < 1024), or if you run
into compatibility
issues with non-podman clients, you can switch using the
following command:

    podman machine set --rootful

API forwarding listening on:
/Users/neo/.local/share/containers/podman/machine/podman-
machine-default/podman.sock

The system helper service is not installed; the default Docker
API socket
address can't be used by podman. If you would like to install
it run the
```

```
following commands:

    sudo /opt/homebrew/Cellar/podman/4.3.1/bin/podman-mac-
helper install
    podman machine stop; podman machine start

You can still connect Docker API clients by setting DOCKER_HOST
using the
following command in your terminal session:

    export
DOCKER_HOST='unix:///Users/neo/.local/share/containers/podman/m
achine/podman-machine-default/podman.sock'

Machine "podman-machine-default" started successfully
```

## 1.5. 让 Podman 支持 Docker Compose

启用 socket

```
systemctl enable podman.socket
systemctl start podman.socket
systemctl status podman.socket
```

验证 sock 是否正常工作

```
[root@localhost ~]# curl -H "Content-Type: application/json" --
unix-socket /run/podman/podman.sock http://localhost/_ping
OK
```

此时可以使用 docker compose

```
[root@localhost ~]# ln -s /run/podman/podman.sock
/var/run/docker.sock
```

## 1.6. 配置 Podman

/etc/containers/registries.conf

## 1.7.

```
$ podman pull maven
$ podman run -v ~/.m2:/root/.m2 \
-v /root/bottleneck:/root/bottleneck \
-w /root/bottleneck \
maven:latest \
mvn package
```

# 2. podman 管理

## 2.1. 虚拟机管理

```
$ podman machine init   # 初始化
$ podman machine start  # 启动 podman VM
$ podman machine stop   # 停止VM
$ podman machine list   # 罗列VM
$ podman machine rm             # 删除VM
$ podman machine ssh    # 通过SSH 进入VM，在终端进行操作
```

### 管理 Podman 系统

```
$ podman system --help

neo@MacBook-Pro-Neo ~> podman system connection list
Name                         URI
Identity                                       Default
podman-machine-default
ssh://core@localhost:59590/run/user/501/podman/podman.sock
/Users/neo/.ssh/podman-machine-default  true
podman-machine-default-root
ssh://root@localhost:59590/run/podman/podman.sock
/Users/neo/.ssh/podman-machine-default  false

$ podman system df
    TYPE           TOTAL       ACTIVE      SIZE
RECLAIMABLE
    Images         29          0           8.931GB     8.931GB
(100%)
    Containers     0           0           0B          0B (0%)
    Local Volumes  1           0           0B          0B (0%)

$ podman system info
```

```

```

## 2.2. 镜像管理

获取镜像

```
[root@localhost ~]# podman pull busybox
Resolved "busybox" as an alias
(/etc/containers/registries.conf.d/000-shortnames.conf)
Trying to pull docker.io/library/busybox:latest...
Getting image source signatures
Copying blob 45a0cdc5c8d3 done
Copying config 334e4a014c done
Writing manifest to image destination
Storing signatures
334e4a014c81bd4050daa78c7dfd2ae87855e9052721c164ea9d9d9a416ebdd
3
```

查看镜像

```
[root@localhost ~]# podman image ls
REPOSITORY                   TAG         IMAGE ID      CREATED
SIZE
docker.io/library/busybox  latest      334e4a014c81  13 days
ago  5.09 MB
```

## 2.3. Registry

```
mkdir -p /var/lib/registry
podman run --privileged -d --name registry -p 5000:5000 -v
```

```
/var/lib/registry:/var/lib/registry --restart=always registry:2
```

修改 /etc/containers/registries.conf 配置文件

```
registries = []
改为
registries = ['localhost:5000']
```

## 3. 按例

### 3.1. podman run 用法

```
podman run -v ~/.m2:/root/.m2 -v
/root/bottleneck:/root/bottleneck -w /root/bottleneck
maven:latest mvn package
```

### 3.2. mysql

```
podman pull mysql
```

```
neo@MacBook-Pro-M2 ~ % podman volume create mysql
mysql

neo@MacBook-Pro-M2 ~ % podman volume ls
DRIVER      VOLUME NAME
local       mysql

neo@MacBook-Pro-M2 ~ % podman run \
-p 3306:3306 \
-e MYSQL_ROOT_PASSWORD=chen \
-v mysql:/var/lib/mysql:rw \
-v /etc/localtime:/etc/localtime:ro \
--name mysql \
-d mysql

neo@MacBook-Pro-M2 ~ % podman exec -it mysql bash
bash-4.4# mysql -h 127.0.0.1 -uroot -pchen
mysql: [Warning] Using a password on the command line interface
can be insecure.
```

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.31 MySQL Community Server - GPL

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

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>
```

如果你想修改密码

```
alter user 'root'@'%' identified with mysql_native_password by
'密码';
```

## 3.3. 制作镜像

```
[root@localhost Maven]# podman pull maven:3-openjdk-18
[root@localhost Maven]# podman run -it --rm --name maven --
entrypoint=sh maven:3-openjdk-18 -c "cat
/usr/share/maven/conf/settings.xml" > settings.xml
[root@localhost Maven]# dos2unix settings.xml
```

修改 settings.xml 文件，加入国内镜像

```
[root@localhost Maven]# cat settings.xml
<?xml version="1.0" encoding="UTF-8"?>

<!--
Licensed to the Apache Software Foundation (ASF) under one
or more contributor license agreements.  See the NOTICE file
distributed with this work for additional information
regarding copyright ownership.  The ASF licenses this file
to you under the Apache License, Version 2.0 (the
"License"); you may not use this file except in compliance
with the License.  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing,
software distributed under the License is distributed on an
"AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
KIND, either express or implied.  See the License for the
specific language governing permissions and limitations
under the License.
-->

<!--
    | This is the configuration file for Maven. It can be
specified at two levels:
    |
    |  1. User Level. This settings.xml file provides
configuration for a single user,
    |                 and is normally provided in
${user.home}/.m2/settings.xml.
    |
    |                 NOTE: This location can be overridden
with the CLI option:
    |
    |                 -s /path/to/user/settings.xml
    |
    |  2. Global Level. This settings.xml file provides
configuration for all Maven
    |                 users on a machine (assuming they're all
using the same Maven
    |                 installation). It's normally provided in
    |                 ${maven.conf}/settings.xml.
    |
    |                 NOTE: This location can be overridden
with the CLI option:
    |
    |                 -gs /path/to/global/settings.xml
    |
    | The sections in this sample file are intended to give you
a running start at
    | getting the most out of your Maven installation. Where
appropriate, the default
    | values (values used when the setting is not specified)
are provided.
    |
    |-->
<settings xmlns="http://maven.apache.org/SETTINGS/1.2.0"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"

xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.2.0
https://maven.apache.org/xsd/settings-1.2.0.xsd">
    <!-- localRepository
     | The path to the local repository maven will use to store
artifacts.
     |
     | Default: ${user.home}/.m2/repository
    <localRepository>/path/to/local/repo</localRepository>
    -->

    <!-- interactiveMode
     | This will determine whether maven prompts you when it
needs input. If set to false,
     | maven will use a sensible default value, perhaps based on
some other setting, for
     | the parameter in question.
     |
     | Default: true
    <interactiveMode>true</interactiveMode>
    -->

    <!-- offline
     | Determines whether maven should attempt to connect to the
network when executing a build.
     | This will have an effect on artifact downloads, artifact
deployment, and others.
     |
     | Default: false
    <offline>false</offline>
    -->
    <!-- pluginGroups
    | This is a list of additional group identifiers that will
be searched when resolving plugins by their prefix, i.e.
    | when invoking a command line like "mvn prefix:goal".
Maven will automatically add the group identifiers
    | "org.apache.maven.plugins" and "org.codehaus.mojo" if
these are not already contained in the list.
    |-->
    <pluginGroups>
    <!-- pluginGroup
        | Specifies a further group identifier to use for
plugin lookup.
    <pluginGroup>com.your.plugins</pluginGroup>
    -->
    </pluginGroups>

    <!-- proxies
    | This is a list of proxies which can be used on this
machine to connect to the network.
    | Unless otherwise specified (by system property or
command-line switch), the first proxy
    | specification in this list marked as active will be used.
    |-->
    <proxies>
    <!-- proxy
        | Specification for one proxy, to be used in connecting
to the network.
        |
    <proxy>
        <id>optional</id>
        <active>true</active>
        <protocol>http</protocol>
        <username>proxyuser</username>
        <password>proxypass</password>
        <host>proxy.host.net</host>
        <port>80</port>
        <nonProxyHosts>local.net|some.host.com</nonProxyHosts>
    </proxy>
    -->
    </proxies>

    <!-- servers
    | This is a list of authentication profiles, keyed by the
server-id used within the system.
```

```
    | Authentication profiles can be used whenever maven must
make a connection to a remote server.
    |-->
    <servers>
    <!-- server
        | Specifies the authentication information to use when
connecting to a particular server, identified by
        | a unique name within the system (referred to by the
'id' attribute below).
        |
        | NOTE: You should either specify username/password OR
privateKey/passphrase, since these pairings are
        |       used together.
        |
    <server>
        <id>deploymentRepo</id>
        <username>repouser</username>
        <password>repopwd</password>
    </server>
    -->

    <!-- Another sample, using keys to authenticate.
    <server>
        <id>siteServer</id>
        <privateKey>/path/to/private/key</privateKey>
        <passphrase>optional; leave empty if not used.
</passphrase>
    </server>
    -->
    </servers>

    <!-- mirrors
    | This is a list of mirrors to be used in downloading
artifacts from remote repositories.
    |
    | It works like this: a POM may declare a repository to use
in resolving certain artifacts.
    | However, this repository may have problems with heavy
traffic at times, so people have mirrored
    | it to several places.
    |
    | That repository definition will have a unique id, so we
can create a mirror reference for that
    | repository, to be used as an alternate download site. The
mirror site will be the preferred
```

```
    | server for that repository.
    |-->
    <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>
    -->
    <mirror>
        <id>aliyunmaven</id>
        <mirrorOf>*</mirrorOf>
        <name>aliyun</name>
        <url>https://maven.aliyun.com/repository/public</url>
    </mirror>
    </mirrors>

    <!-- profiles
    | This is a list of profiles which can be activated in a
variety of ways, and which can modify
    | the build process. Profiles provided in the settings.xml
are intended to provide local machine-
    | specific paths and repository locations which allow the
build to work in the local environment.
    |
    | For example, if you have an integration testing plugin -
like cactus - that needs to know where
    | your Tomcat instance is installed, you can provide a
```

```
variable here such that the variable is
    | dereferenced during the build process to configure the
cactus plugin.
    |
    | As noted above, profiles can be activated in a variety of
ways. One way - the activeProfiles
    | section of this document (settings.xml) - will be
discussed later. Another way essentially
    | relies on the detection of a system property, either
matching a particular value for the property,
    | or merely testing its existence. Profiles can also be
activated by JDK version prefix, where a
    | value of '1.4' might activate a profile when the build is
executed on a JDK version of '1.4.2_07'.
    | Finally, the list of active profiles can be specified
directly from the command line.
    |
    | NOTE: For profiles defined in the settings.xml, you are
restricted to specifying only artifact
    |       repositories, plugin repositories, and free-form
properties to be used as configuration
    |       variables for plugins in the POM.
    |
    |-->
    <profiles>
    <!-- profile
        | Specifies a set of introductions to the build
process, to be activated using one or more of the
        | mechanisms described above. For inheritance purposes,
and to activate profiles via <activatedProfiles/>
        | or the command line, profiles have to have an ID that
is unique.
        |
        | An encouraged best practice for profile
identification is to use a consistent naming convention
        | for profiles, such as 'env-dev', 'env-test', 'env-
production', 'user-jdcasey', 'user-brett', etc.
        | This will make it more intuitive to understand what
the set of introduced profiles is attempting
        | to accomplish, particularly when you only have a list
of profile id's for debug.
        |
        | This profile example uses the JDK version to trigger
activation, and provides a JDK-specific repo.
    <profile>
```

```
        <id>jdk-1.4</id>

        <activation>
        <jdk>1.4</jdk>
        </activation>

        <repositories>
        <repository>
            <id>jdk14</id>
            <name>Repository for JDK 1.4 builds</name>
            <url>http://www.myhost.com/maven/jdk14</url>
            <layout>default</layout>
            <snapshotPolicy>always</snapshotPolicy>
        </repository>
        </repositories>
    </profile>
    -->

    <!--
        | Here is another profile, activated by the system
property 'target-env' with a value of 'dev',
        | which provides a specific path to the Tomcat
instance. To use this, your plugin configuration
        | might hypothetically look like:
        |
        | ...
        | <plugin>
        |   <groupId>org.myco.myplugins</groupId>
        |   <artifactId>myplugin</artifactId>
        |
        |   <configuration>
        |     <tomcatLocation>${tomcatPath}</tomcatLocation>
        |   </configuration>
        | </plugin>
        | ...
        |
        | NOTE: If you just wanted to inject this configuration
whenever someone set 'target-env' to
        |       anything, you could just leave off the <value/>
inside the activation-property.
        |
    <profile>
        <id>env-dev</id>

        <activation>
        <property>
            <name>target-env</name>
            <value>dev</value>
        </property>
        </activation>

        <properties>
        <tomcatPath>/path/to/tomcat/instance</tomcatPath>
        </properties>
    </profile>
    -->
    </profiles>

    <!-- activeProfiles
    | List of profiles that are active for all builds.
    |
    <activeProfiles>
    <activeProfile>alwaysActiveProfile</activeProfile>
    <activeProfile>anotherAlwaysActiveProfile</activeProfile>
    </activeProfiles>
    -->
</settings>
```

创建 Dockerfile 文件

```
[root@localhost Maven]# cat Dockerfile
FROM maven:3-openjdk-18

COPY settings.xml /root/.m2/settings.xml
```

制作 Maven 镜像

```
[root@localhost Maven]# podman build -t
"docker.io/netkiller/maven:3-openjdk-18" .

[root@localhost Maven]# podman image ls | grep maven
```

```
docker.io/netkiller/maven       3-openjdk-18  3951f6d3aa19  50
seconds ago  829 MB
docker.io/library/maven         latest        0f909120a578  3
weeks ago     543 MB
docker.io/library/maven         3-openjdk-18  1e86120a0116  3
weeks ago     829 MB

[root@localhost Maven]# podman login docker.io/netkiller
Username: netkiller
Password:
Login Succeeded!

[root@localhost Maven]# podman push
docker.io/netkiller/maven:3-openjdk-18
```

使用自制的 Maven 镜像

```
[root@localhost ~]# podman run -it --rm --name maven -v
~/.m2:/root/.m2 -v /root/bottleneck:/root/bottleneck -w
/root/bottleneck docker.io/netkiller/maven:3-openjdk-18 mvn
package
[INFO] Scanning for projects...
[INFO]
[INFO] ----------------------< cn.netkiller:bottleneck >-------
----------------
[INFO] Building bottleneck 0.0.1-SNAPSHOT
[INFO] --------------------------------[ jar ]-----------------
----------------
[INFO]
[INFO] --- maven-resources-plugin:3.3.0:resources (default-
resources) @ bottleneck ---
[INFO] Copying 1 resource
[INFO] Copying 4 resources
[INFO]
[INFO] --- maven-compiler-plugin:3.10.1:compile (default-
compile) @ bottleneck ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 8 source files to
/root/bottleneck/target/classeskm
[INFO]
```

```
[INFO] --- maven-resources-plugin:3.3.0:testResources (default-
testResources) @ bottleneck ---
[INFO] skip non existing resourceDirectory
/root/bottleneck/src/test/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.10.1:testCompile (default-
testCompile) @ bottleneck ---
[INFO] No sources to compile
[INFO]
[INFO] --- maven-surefire-plugin:2.22.2:test (default-test) @
bottleneck ---
[INFO] Tests are skipped.
[INFO]
[INFO] --- maven-jar-plugin:3.3.0:jar (default-jar) @
bottleneck ---
[INFO] Building jar: /root/bottleneck/target/bottleneck-0.0.1-
SNAPSHOT.jar
[INFO]
[INFO] --- spring-boot-maven-plugin:3.0.1:repackage (repackage)
@ bottleneck ---
[INFO] Replacing main artifact with repackaged archive
[INFO] --------------------------------------------------------
----------------
[INFO] BUILD SUCCESS
[INFO] --------------------------------------------------------
----------------
[INFO] Total time:  1.546 s
[INFO] Finished at: 2023-01-01T11:58:11Z
[INFO] --------------------------------------------------------
----------------
```

# 部分 I. Kubernetes

# 第 3 章 Minikube

## 1. CentOS 8 安装 minikube

### 1.1. CentOS

执行下面命令检查服务器是否开启虚拟化技术

```
egrep --color 'vmx|svm' /proc/cpuinfo
```

如果没有任何输出，请重启服务器进入 BIOS 启用 VT-X 或 AMD-v

```
curl -LO
https://storage.googleapis.com/minikube/releases/latest/minikub
e-linux-amd64 \
  && install minikube-linux-amd64 /usr/local/bin/minikube
```

尝试运行 minikube 如果输出帮助信息表示安装成功

```
[root@localhost ~]# minikube version
minikube version: v1.13.0
commit: 0c5e9de4ca6f9c55147ae7f90af97eff5befef5f-dirty
```

```
echo "1" > /proc/sys/net/bridge/bridge-nf-call-iptables
```

dnf 安装 kubectl

```
cat <<EOF > /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-
el7-x86_64
enabled=1
gpgcheck=1
repo_gpgcheck=1
gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg
https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg
EOF
```

```
[root@localhost ~]# dnf install kubectl
```

二进制安装 kubectl

```
curl -LO "https://storage.googleapis.com/kubernetes-
release/release/$(curl -s
https://storage.googleapis.com/kubernetes-
release/release/stable.txt)/bin/linux/amd64/kubectl" \
        && install kubectl /usr/local/bin/kubectl
```

## 无虚拟机

如果你不想安装虚拟机

```
adduser docker
su - docker
sudo usermod -aG docker $USER && newgrp docker
```

```
[docker@localhost ~]$ minikube start --driver=docker
* minikube v1.13.0 on Centos 8.2.2004
* Using the docker driver based on user configuration

X Requested memory allocation (1694MB) is less than the
recommended minimum 2000MB. Deployments may fail.


X The requested memory allocation of 1694MiB does not leave
room for system overhead (total system memory: 1694MiB). You
may face stability issues.
* Suggestion: Start minikube with less memory allocated:
'minikube start --memory=1694mb'

* Starting control plane node minikube in cluster minikube
* Pulling base image ...
* Downloading Kubernetes v1.19.0 preload ...
    > preloaded-images-k8s-v6-v1.19.0-docker-overlay2-
amd64.tar.lz4: 486.28 MiB
```

## 1.2. Mac OS

检查硬件是否支持虚拟化

```
iMac:Linux neo$ sysctl -a | grep -E --color
'machdep.cpu.features|VMX'
machdep.cpu.features: FPU VME DE PSE TSC MSR PAE MCE CX8 APIC
SEP MTRR PGE MCA CMOV PAT PSE36 CLFSH DS ACPI MMX FXSR SSE SSE2
SS HTT TM PBE SSE3 PCLMULQDQ DTES64 MON DSCPL VMX SMX EST TM2
```

```
SSSE3 CX16 TPR PDCM SSE4.1 SSE4.2 x2APIC POPCNT AES PCID XSAVE
OSXSAVE TSCTMR AVX1.0
```

```
$ brew install hyperkit
$ brew install minikube
$ brew install kubectl
$ brew install kubernetes-helm
```

```
neo@MacBook-Pro-Neo ~ % minikube start
😄  minikube v1.13.1 on Darwin 11.0
🆕  Kubernetes 1.19.2 is now available. If you would like to
upgrade, specify: --kubernetes-version=v1.19.2
✨  Using the hyperkit driver based on existing profile
👍  Starting control plane node minikube in cluster minikube
🔄  Restarting existing hyperkit VM for "minikube" ...
❗  This VM is having trouble accessing https://k8s.gcr.io
💡  To pull new external images, you may need to configure a
proxy:
https://minikube.sigs.k8s.io/docs/reference/networking/proxy/
🐳  Preparing Kubernetes v1.19.0 on Docker 19.03.12 ...
🔎  Verifying Kubernetes components...
🌟  Enabled addons: dashboard, default-storageclass, storage-
provisioner
🏄  Done! kubectl is now configured to use "minikube" by
default
```

有些老系统可能不支持 hyperkit，需要virtualbox。

```
$ brew cask install virtualbox
$ minikube start —vm-driver=virtualbox
$ minikube dashboard
```

检查 minikube 启动状态

```
Neo-iMac:~ neo$ docker container inspect minikube --format=
{{.State.Status}}
running
```

## 2. Quickstart

启动

```
minikube start
```

运行一个 echoserver 镜像

```
kubectl run hello-minikube --image=k8s.gcr.io/echoserver:1.4 --
port=8080
kubectl expose deployment hello-minikube --type=NodePort
minikube service hello-minikube
```

查询 echoserver 访问地址

```
minikube service hello-minikube --url
```

在浏览器中访问查询到的网址

停止并删除镜像

```
minikube stop
minikube delete
```

**例 3.1. minikube 操作演示**

快速开始使用 minikube 运行一个镜像

```
[root@localhost ~]# kubectl run hello-minikube --
image=k8s.gcr.io/echoserver:1.4 --port=8080
kubectl run --generator=deployment/apps.v1 is DEPRECATED and
will be removed in a future version. Use kubectl run --
generator=run-pod/v1 or kubectl create instead.
deployment.apps/hello-minikube created

[root@localhost ~]# kubectl expose deployment hello-minikube --
type=NodePort
service/hello-minikube exposed

[root@localhost ~]# minikube service hello-minikube
Opening kubernetes service default/hello-minikube in default
browser...

[root@localhost ~]# kubectl get pod
NAME                              READY   STATUS    RESTARTS
AGE
hello-minikube-5c856cbf98-6vfvp   1/1     Running   0
6m59s

[root@localhost ~]# minikube service hello-minikube --url
http://172.16.0.121:30436

[root@localhost ~]# curl http://172.16.0.121:30436
CLIENT VALUES:
client_address=172.17.0.1
command=GET
real path=/
query=nil
request_version=1.1
request_uri=http://172.16.0.121:8080/

SERVER VALUES:
server_version=nginx: 1.10.0 - lua: 10001

HEADERS RECEIVED:
accept=*/*
```

```
host=172.16.0.121:30436
user-agent=curl/7.29.0
BODY:
-no body in request-
```

## 3. minikube 命令

```
[root@localhost ~]# minikube
Minikube is a CLI tool that provisions and manages single-node
Kubernetes clusters optimized for development workflows.

Usage:
  minikube [command]

Available Commands:
  addons         Modify minikube's kubernetes addons
  cache          Add or delete an image from the local cache.
  completion     Outputs minikube shell completion for the
given shell (bash or zsh)
  config         Modify minikube config
  dashboard      Access the kubernetes dashboard running within
the minikube cluster
  delete         Deletes a local kubernetes cluster
  docker-env     Sets up docker env variables; similar to
'$(docker-machine env)'
  help           Help about any command
  ip             Retrieves the IP address of the running
cluster
  logs           Gets the logs of the running instance, used
for debugging minikube, not user code
  mount          Mounts the specified directory into minikube
  profile        Profile sets the current minikube profile
  service        Gets the kubernetes URL(s) for the specified
service in your local cluster
  ssh            Log into or run a command on a machine with
SSH; similar to 'docker-machine ssh'
  ssh-key        Retrieve the ssh identity key path of the
specified cluster
  start          Starts a local kubernetes cluster
  status         Gets the status of a local kubernetes cluster
  stop           Stops a running local kubernetes cluster
  tunnel         tunnel makes services of type LoadBalancer
accessible on localhost
  update-check   Print current and latest version number
  update-context Verify the IP address of the running cluster
```

```
in kubeconfig.
  version        Print the version of minikube

Flags:
      --alsologtostderr                  log to standard error
as well as files
  -b, --bootstrapper string              The name of the
cluster bootstrapper that will set up the kubernetes cluster.
(default "kubeadm")
  -h, --help                             help for minikube
      --log_backtrace_at traceLocation   when logging hits line
file:N, emit a stack trace (default :0)
      --log_dir string                   If non-empty, write
log files in this directory
      --logtostderr                      log to standard error
instead of files
  -p, --profile string                   The name of the
minikube VM being used.
                                                This can be
modified to allow for multiple minikube instances to be run
independently (default "minikube")
      --stderrthreshold severity         logs at or above this
threshold go to stderr (default 2)
  -v, --v Level                          log level for V logs
      --vmodule moduleSpec               comma-separated list
of pattern=N settings for file-filtered logging

Use "minikube [command] --help" for more information about a
command.
```

## 3.1. minikube ip 地址

```
[docker@localhost ~]$ minikube ip
192.168.58.2
```

```
kubectl get nodes -o
```

```
jsonpath='{.items[*].status.addresses[].address}'
```

## 3.2. 启动 minikube

### 虚拟机驱动

--vm-driver=none

```
minikube start --vm-driver=none
```

### 开启GPU

```
minikube start --vm-driver kvm2 --gpu
```

### 日志输出级别

指定日志输出级别

```
minikube start --v=7
```

### CPU 和 内存分配

```
minikube start --memory 8000 --cpus 2
```

## 指定 kubernetes 版本

```
minikube start --memory 8000 --cpus 2 --kubernetes-version
v1.6.0
```

## 配置启动项

```
minikube start --extra-config=apiserver.v=10 --extra-
config=kubelet.max-pods=100
```

## 指定 registry-mirror 镜像

```
minikube start --registry-mirror=https://registry.docker-cn.com

minikube start --image-mirror-country=cn --registry-
mirror="https://docker.mirrors.ustc.edu.cn" --insecure-
registry="127.0.0.1:5000"

minikube start --image-mirror-country=cn --registry-
mirror="https://docker.mirrors.ustc.edu.cn" --insecure-
registry="192.168.0.0/24"
```

## 指定下载镜像

```
minikube start --image-mirror-country=cn --image-
repository=registry.cn-hangzhou.aliyuncs.com/google_containers
```

```
# 从阿里云下载 virtualbox 镜像
minikube start --vm-driver='virtualbox' --image-mirror-country
cn \
    --iso-url=https://kubernetes.oss-cn-
hangzhou.aliyuncs.com/minikube/iso/minikube-v1.9.0.iso \
    --registry-mirror=https://docker.mirrors.ustc.edu.cn

minikube start --vm-driver=virtualbox \
--image-mirror-country cn \
--registry-mirror=https://docker.mirrors.ustc.edu.cn \
--image-repository=registry.aliyuncs.com/google_containers \
--insecure-registry=192.168.0.10:5000 //访问宿主机的私有docker仓库
```

### Enabling Unsafe Sysctls

```
minikube start --extra-config="kubelet.allowed-unsafe-
sysctls=kernel.msg*,net.core.somaxconn".
```

### 使用 CRI-O 容易

```
minikube start --container-runtime=cri-o --vm-driver=none
```

启动演示

```
iMac:~ neo$ minikube start --container-runtime=cri-o
😄  Darwin 10.13.6 上的 minikube v1.15.0
🆕  Kubernetes 1.19.4 is now available. If you would like to
upgrade, specify: --kubernetes-version=v1.19.4
✨  根据现有的配置文件使用 hyperkit 驱动程序
👍  Starting control plane node minikube in cluster minikube
🔄  Restarting existing hyperkit VM for "minikube" ...
🎁  正在 CRI-O 1.17.3 中准备 Kubernetes v1.19.2…
🔗  Configuring bridge CNI (Container Networking Interface) ...
🔎  Verifying Kubernetes components...
🌟  Enabled addons: storage-provisioner, dashboard, default-
storageclass
🏄  Done! kubectl is now configured to use "minikube" cluster
and "" namespace by default
```

## 3.3. 停止 minikube

```
[root@localhost ~]# minikube stop
Stopping local Kubernetes cluster...
Machine stopped.
```

## 3.4. Docker 环境变量

```
neo@MacBook-Pro-Neo ~ % minikube docker-env
export DOCKER_TLS_VERIFY="1"
export DOCKER_HOST="tcp://192.168.64.3:2376"
export DOCKER_CERT_PATH="/Users/neo/.minikube/certs"
export MINIKUBE_ACTIVE_DOCKERD="minikube"

# To point your shell to minikube's docker-daemon, run:
# eval $(minikube -p minikube docker-env)
```

设置环境变量

```
# eval $(minikube docker-env)
# eval $(minikube -p minikube docker-env)
```

## 3.5. SSH

```
neo@MacBook-Pro-Neo ~ % minikube ssh
                         _             _            
            _         _ ( )           ( )           
  ___ ___  (_)  ___  (_)| |/')  _   _ | |_      __  
/' _ ` _ `\| |/' _ `\| || , <  ( ) ( )| '_`\  /'__`\
| ( ) ( ) || || ( ) || || |\`\ | (_) || |_) )(  ___/
(_) (_) (_)(_)(_) (_)(_)(_) (_)`\___/'(_,__/'`\____)

$
```

## 3.6. 缓存镜像

```
# cache a image into $HOME/.minikube/cache/images

$ minikube cache add ubuntu:16.04
$ minikube cache add redis:3

# list cached images
$ minikube cache list
redis:3
ubuntu:16.04

# delete cached images
$ minikube cache delete ubuntu:16.04
$ minikube cache delete $(minikube cache list)
```

```
```

## 3.7. 清理 minikube

```
minikube delete
rm ~/.minikube
minikube start
```

## 3.8. Kubernetes 控制面板

Dashboard是基于Web的Kubernetes管理界面。使用下面的命令启动：

```
minikube dashboard
```

查询控制面板访问地址

```
$ minikube dashboard --url
http://192.168.3.14:30000
```

## 3.9. service

列出所有服务

```
Neo-iMac:~ neo$ minikube service list
|----------------------|------------------------------------|--
```

```
------------|-----|
|      NAMESPACE       |                NAME                 |
TARGET PORT  | URL |
|----------------------|------------------------------------|--
------------|-----|
| default              | kubernetes                         |
No node port |
| default              | nginx                              |
80 |      |
| ingress-nginx        | ingress-nginx-controller           |
http/80      |      |
|                      |                                    |
https/443    |      |
| ingress-nginx        | ingress-nginx-controller-admission |
No node port |
| kube-system          | kube-dns                           |
No node port |
| kubernetes-dashboard | dashboard-metrics-scraper          |
No node port |
| kubernetes-dashboard | kubernetes-dashboard               |
No node port |
|----------------------|------------------------------------|--
------------|-----|
```

查看指定服务

```
Neo-iMac:~ neo$ minikube service nginx
|-----------|-------|-------------|---------------------------|
| NAMESPACE | NAME  | TARGET PORT |            URL            |
|-----------|-------|-------------|---------------------------|
| default   | nginx |          80 | http://192.168.49.2:30330 |
|-----------|-------|-------------|---------------------------|
🏃  Starting tunnel for service nginx.
|-----------|-------|-------------|------------------------|
| NAMESPACE | NAME  | TARGET PORT |          URL           |
|-----------|-------|-------------|------------------------|
| default   | nginx |             | http://127.0.0.1:55815 |
|-----------|-------|-------------|------------------------|
🎉  Opening service default/nginx in default browser...
❗  Because you are using a Docker driver on darwin, the
```

```
terminal needs to be open to run it.
```

查看服务的网址

```
[root@localhost ~]# minikube service hello-minikube --url
http://172.16.0.121:30436
```

## 3.10. 查看日志

```
minikube logs -v10
```

## 3.11. 查看 Docker 环境变量

minikube docker-env

```
Neo-iMac:~ neo$ minikube docker-env
export DOCKER_TLS_VERIFY="1"
export DOCKER_HOST="tcp://127.0.0.1:54734"
export DOCKER_CERT_PATH="/Users/neo/.minikube/certs"
export MINIKUBE_ACTIVE_DOCKERD="minikube"

# To point your shell to minikube's docker-daemon, run:
# eval $(minikube -p minikube docker-env)
```

## 3.12. profile

```
minikube profile demo
minikube start -p demo --memory=8192 --cpus=6 --disk-size=50g
```

## 3.13. addons

查看所有插件

```
iMac:registry neo$ minikube addons list
|-----------------------------|----------|--------------|
|         ADDON NAME          | PROFILE  |    STATUS    |
|-----------------------------|----------|--------------|
| ambassador                  | minikube | disabled     |
| dashboard                   | minikube | enabled ✅   |
| default-storageclass        | minikube | enabled ✅   |
| efk                         | minikube | disabled     |
| freshpod                    | minikube | disabled     |
| gcp-auth                    | minikube | disabled     |
| gvisor                      | minikube | disabled     |
| helm-tiller                 | minikube | disabled     |
| ingress                     | minikube | disabled     |
| ingress-dns                 | minikube | disabled     |
| istio                       | minikube | disabled     |
| istio-provisioner           | minikube | disabled     |
| kubevirt                    | minikube | disabled     |
| logviewer                   | minikube | disabled     |
| metallb                     | minikube | disabled     |
| metrics-server              | minikube | disabled     |
| nvidia-driver-installer     | minikube | disabled     |
| nvidia-gpu-device-plugin    | minikube | disabled     |
| olm                         | minikube | disabled     |
| pod-security-policy         | minikube | disabled     |
| registry                    | minikube | disabled     |
| registry-aliases            | minikube | disabled     |
| registry-creds              | minikube | disabled     |
| storage-provisioner         | minikube | enabled ✅   |
| storage-provisioner-gluster | minikube | disabled     |
|-----------------------------|----------|--------------|
```

## 启用 addons

```
minikube addons enable heapster
minikube addons enable ingress
```

启用 WebUI

```
[root@localhost ~]# minikube addons enable dashboard
dashboard was successfully enabled
[root@localhost ~]# minikube addons list | grep dashboard
- dashboard: enabled
```

## 查看 addons 列表

```
[root@localhost ~]# minikube addons list
- addon-manager: enabled
- dashboard: enabled
- default-storageclass: enabled
- efk: disabled
- freshpod: disabled
- gvisor: disabled
- heapster: disabled
- ingress: disabled
- kube-dns: disabled
- metrics-server: disabled
- nvidia-driver-installer: disabled
- nvidia-gpu-device-plugin: disabled
- registry: disabled
- registry-creds: disabled
- storage-provisioner: enabled
- storage-provisioner-gluster: disabled
```

### dashboard

```
Neo-iMac:~ neo$ minikube addons enable dashboard
    ▪ Using image registry.cn-hangzhou.aliyuncs.com/google_containers/metrics-scraper:v1.0.7
    ▪ Using image registry.cn-hangzhou.aliyuncs.com/google_containers/dashboard:v2.3.1
💡  Some dashboard features require the metrics-server addon. To enable all features please run:

	minikube addons enable metrics-server	


🌟  The 'dashboard' addon is enabled
```

```
Neo-iMac:~ neo$ minikube dashboard
🤔  Verifying dashboard health ...
🚀  Launching proxy ...
🤔  Verifying proxy health ...
🎉  Opening http://127.0.0.1:62433/api/v1/namespaces/kubernetes-dashboard/services/http:kubernetes-dashboard:/proxy/ in your default browser...
```

### 开启 registry 私有库

```
# enable the registry addon
```

```
$ minikube addons enable registry

$ minikube start

# use the minikube docker daemon from the host
$ eval $(minikube docker-env)

# get the ip of the registry endpoint
$ kubectl -n kube-system get svc registry -o jsonpath="
{.spec.clusterIP}"
10.0.0.240
```

```
{
  "insecure-registries" : ["10.0.0.240"]
}
```

```
$ minikube ssh
$ docker pull busybox
$ docker tag busybox 10.0.0.240/busybox

# or

# build and push to insecure registry
$ docker build -t 10.0.0.240/busybox .
$ docker push 10.0.0.240/busybox
```

### 启用 ingress

```
Neo-iMac:~ neo$ minikube addons enable ingress
💡  After the addon is enabled, please run "minikube tunnel"
and your ingress resources would be available at "127.0.0.1"
```

```
    ▪ Using image registry.cn-
hangzhou.aliyuncs.com/google_containers/kube-webhook-
certgen:v1.1.1
    ▪ Using image registry.cn-
hangzhou.aliyuncs.com/google_containers/kube-webhook-
certgen:v1.1.1
    ▪ Using image registry.cn-
hangzhou.aliyuncs.com/google_containers/nginx-ingress-
controller:v1.0.4
🔎  Verifying ingress addon...
🌟  The 'ingress' addon is enabled
```

运行一个简单的demo

```
运行 nginx 服务
kubectl run nginx --image=nginx --port=80
暴露服务
kubectl expose deployment nginx --port=80 --target-port=80

创建ingress
yaml 定义 ingress.yaml

apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: nginx
spec:
  rules:
  - host: www.netkiller.cn
    http:
      paths:
      - path: /
        backend:
          serviceName: nginx
          servicePort: 80

运行
kubectl apply -f ingress.yaml

配置本机host获取minikube ip
```

```
[docker@localhost ~]$ minikube ip
192.168.58.2

配置 /etc/hosts 文件
192.168.58.2 www.netkiller.cn
```

访问 http://www.netkiller.cn

## 3.14. SSH

--vm-driver=none 不支持 ssh

```
[root@localhost ~]# minikube ssh
'none' driver does not support 'minikube ssh' command
```

## 3.15. 查看IP地址

```
[root@localhost ~]# minikube ip
172.16.0.121
```

## 3.16. 镜像管理

```
neo@MacBook-Pro-Neo ~ % minikube image ls
registry.cn-hangzhou.aliyuncs.com/google_containers/storage-provisioner:v5
registry.cn-hangzhou.aliyuncs.com/google_containers/pause:3.2
registry.cn-hangzhou.aliyuncs.com/google_containers/metrics-scraper:v1.0.4
registry.cn-hangzhou.aliyuncs.com/google_containers/kube-
scheduler:v1.20.7
registry.cn-hangzhou.aliyuncs.com/google_containers/kube-
proxy:v1.20.7
registry.cn-hangzhou.aliyuncs.com/google_containers/kube-
controller-manager:v1.20.7
registry.cn-hangzhou.aliyuncs.com/google_containers/kube-
apiserver:v1.20.7
registry.cn-
hangzhou.aliyuncs.com/google_containers/etcd:3.4.13-0
registry.cn-
hangzhou.aliyuncs.com/google_containers/dashboard:v2.1.0
registry.cn-
hangzhou.aliyuncs.com/google_containers/coredns:1.7.0
docker.io/netkiller/flask:latest
```

## 3.17. kubectl

```
neo@MacBook-Pro-Neo ~ % minikube kubectl -- get pods -A
    > kubectl.sha256: 64 B / 64 B [--------------------------]
100.00% ? p/s 0s
    > kubectl: 44.08 MiB / 44.08 MiB [---------------] 100.00%
5.30 MiB p/s 8.5s
NAMESPACE       NAME
READY   STATUS              RESTARTS   AGE
ingress-nginx   ingress-nginx-admission-create-vzk2b        0/1
ImagePullBackOff    0          118d
ingress-nginx   ingress-nginx-admission-patch-65b85         0/1
ImagePullBackOff    0          118d
ingress-nginx   ingress-nginx-controller-7f79776f95-ncqkn   0/1
ContainerCreating   0          118d
kube-system     coredns-54d67798b7-cnjgw                    1/1
Running             2          121d
kube-system     etcd-minikube                               1/1
Running             2          121d
kube-system     kube-apiserver-minikube                     1/1
Running             2          121d
kube-system     kube-controller-manager-minikube            1/1
Running             2          121d
kube-system     kube-proxy-tr8fd                            1/1
```

```
Running             2          121d
kube-system     kube-scheduler-minikube                     1/1
Running             2          121d
kube-system     storage-provisioner                         1/1
Running             2          121d
```

## 4. Minikube 案例演示

# 5. FAQ

## 5.1. This computer doesn't have VT-X/AMD-v enabled. Enabling it in the BIOS is mandatory

检查一下 BIOS 是否开启 VT-X/AMD-v

如果在虚拟机安装 Minikube 也会遇到这个问题。 可以使用 --vm-driver=none 参数启动。

```
neo@ubuntu:~$ sudo minikube start --vm-driver=none
```

## 5.2. ERROR FileContent--proc-sys-net-bridge-bridge-nf-call-iptables

解决方法

```
echo "1" > /proc/sys/net/bridge/bridge-nf-call-iptables
```

然后在 minikube start

## 5.3. ERROR ImagePull

[ERROR ImagePull]: failed to pull image k8s.gcr.io/pause:3.1: output: 3.1: Pulling from pause Get https://k8s.gcr.io/v2/pause/manifests/sha256:59eec8837a4d942cc19a52b8c09ea75121acc38114a2c68b98983ce9356b8610: net/http: TLS handshake timeout

更换镜像再重试

```
[root@localhost ~]# minikube start --vm-driver=none --registry-
mirror=https://registry.docker-cn.com
```

## 5.4. 证书已存在错误

启动提示如下错误，一般出现这种错误是因为 minikube stop, minikube delete 后再重启 minikube start

```
error execution phase kubeconfig/admin: a kubeconfig file
"/etc/kubernetes/admin.conf" exists already but has got the
wrong CA cert
error execution phase kubeconfig/kubelet: a kubeconfig file
"/etc/kubernetes/kubelet.conf" exists already but has got the
wrong CA cert
error execution phase kubeconfig/controller-manager: a
kubeconfig file "/etc/kubernetes/controller-manager.conf"
exists already but has got the wrong CA cert
error execution phase kubeconfig/scheduler: a kubeconfig file
"/etc/kubernetes/scheduler.conf" exists already but has got the
wrong CA cert
```

解决方法

```
[root@localhost ~]# mv /etc/kubernetes/admin.conf
/etc/kubernetes/admin.conf.backup
[root@localhost ~]# mv /etc/kubernetes/kubelet.conf
/etc/kubernetes/kubelet.conf.backup
[root@localhost ~]# mv /etc/kubernetes/controller-manager.conf
/etc/kubernetes/controller-manager.conf.backup
[root@localhost ~]# mv /etc/kubernetes/scheduler.conf
```

```
/etc/kubernetes/scheduler.conf.backup
```

现在启动 minikube start 不会再出错

```
[root@localhost ~]# minikube start --vm-driver=none
Starting local Kubernetes v1.13.2 cluster...
Starting VM...
Getting VM IP address...
Moving files into cluster...
Setting up certs...
Connecting to cluster...
Setting up kubeconfig...
Stopping extra container runtimes...
Starting cluster components...
Verifying kubelet health ...
Verifying apiserver health ...
Kubectl is now configured to use the cluster.
===================
WARNING: IT IS RECOMMENDED NOT TO RUN THE NONE DRIVER ON
PERSONAL WORKSTATIONS
        The 'none' driver will run an insecure kubernetes
apiserver as root that may leave the host vulnerable to CSRF
attacks

When using the none driver, the kubectl config and credentials
generated will be root owned and will appear in the root home
directory.
You will need to move the files to the appropriate location and
then set the correct permissions.  An example of this is below:

        sudo mv /root/.kube $HOME/.kube # this will write over
any previous configuration
        sudo chown -R $USER $HOME/.kube
        sudo chgrp -R $USER $HOME/.kube

        sudo mv /root/.minikube $HOME/.minikube # this will
write over any previous configuration
        sudo chown -R $USER $HOME/.minikube
        sudo chgrp -R $USER $HOME/.minikube
```

```
This can also be done automatically by setting the env var
CHANGE_MINIKUBE_NONE_USER=true
Loading cached images from config file.


Everything looks great. Please enjoy minikube!
```

## 5.5. http: server gave HTTP response to HTTPS client

问题原因，使用私有 registry 由于没有 HTTPS 导致 kubectl 使用 https 去访问私有 registry.

```
Failed to pull image
"192.168.3.85:5000/netkiller/config:latest": rpc error: code =
Unknown desc = Error response from daemon: Get
https://192.168.3.85:5000/v2/: http: server gave HTTP response
to HTTPS client
```

minikube 并不会使用 docker 配置文件中的 insecure-registry 配置项

解决办法

```
minikube start --insecure-registry=127.0.0.1:5000
```

或指定网段

```
minikube start --insecure-registry "10.0.0.0/24"
```

## 5.6. provided port is not in the valid range. The range of valid ports is 30000-32767

```
iMac:kubernetes neo$ kubectl create -f redis/redis.yml
configmap/redis-config created
deployment.apps/redis created
The Service "redis" is invalid: spec.ports[0].nodePort: Invalid
value: 6379: provided port is not in the valid range. The range
of valid ports is 30000-32767
```

编辑kube-apiserver.yaml文件

```
$ minikube ssh
$ sudo vi /etc/kubernetes/manifests/kube-apiserver.yaml
```

增加kube-apiserver的启动配置项

```
--service-node-port-range=1024-65535
```

```
$ sudo cat /etc/kubernetes/manifests/kube-apiserver.yaml
apiVersion: v1
kind: Pod
metadata:
  annotations:
    kubeadm.kubernetes.io/kube-apiserver.advertise-
address.endpoint: 192.168.64.5:8443
  creationTimestamp: null
  labels:
    component: kube-apiserver
    tier: control-plane
  name: kube-apiserver
  namespace: kube-system
spec:
  containers:
  - command:
    - kube-apiserver
    - --advertise-address=192.168.64.5
    - --allow-privileged=true
    - --authorization-mode=Node,RBAC
    - --client-ca-file=/var/lib/minikube/certs/ca.crt
    - --enable-admission-
plugins=NamespaceLifecycle,LimitRanger,ServiceAccount,DefaultSt
orageClass,DefaultTolerationSeconds,NodeRestriction,MutatingAdm
issionWebhook,ValidatingAdmissionWebhook,ResourceQuota
    - --enable-bootstrap-token-auth=true
    - --etcd-cafile=/var/lib/minikube/certs/etcd/ca.crt
    - --etcd-certfile=/var/lib/minikube/certs/apiserver-etcd-
client.crt
    - --etcd-keyfile=/var/lib/minikube/certs/apiserver-etcd-
client.key
    - --etcd-servers=https://127.0.0.1:2379
    - --insecure-port=0
    - --kubelet-client-
certificate=/var/lib/minikube/certs/apiserver-kubelet-
client.crt
    - --kubelet-client-key=/var/lib/minikube/certs/apiserver-
kubelet-client.key
    - --kubelet-preferred-address-
types=InternalIP,ExternalIP,Hostname
    - --proxy-client-cert-file=/var/lib/minikube/certs/front-
proxy-client.crt
    - --proxy-client-key-file=/var/lib/minikube/certs/front-
proxy-client.key
    - --requestheader-allowed-names=front-proxy-client
    - --requestheader-client-ca-
file=/var/lib/minikube/certs/front-proxy-ca.crt
    - --requestheader-extra-headers-prefix=X-Remote-Extra-
    - --requestheader-group-headers=X-Remote-Group
    - --requestheader-username-headers=X-Remote-User
    - --secure-port=8443
    - --service-account-key-file=/var/lib/minikube/certs/sa.pub
    - --service-cluster-ip-range=10.10.0.0/24
    - --service-node-port-range=1024-65535
    - --tls-cert-file=/var/lib/minikube/certs/apiserver.crt
    - --tls-private-key-
file=/var/lib/minikube/certs/apiserver.key
    image: registry.cn-
hangzhou.aliyuncs.com/google_containers/kube-apiserver:v1.19.2
    imagePullPolicy: IfNotPresent
    livenessProbe:
      failureThreshold: 8
      httpGet:
        host: 192.168.64.5
        path: /livez
        port: 8443
        scheme: HTTPS
      initialDelaySeconds: 10
      periodSeconds: 10
      timeoutSeconds: 15
    name: kube-apiserver
    readinessProbe:
      failureThreshold: 3
      httpGet:
        host: 192.168.64.5
        path: /readyz
        port: 8443
        scheme: HTTPS
      periodSeconds: 1
      timeoutSeconds: 15
    resources:
      requests:
        cpu: 250m
    startupProbe:
      failureThreshold: 24
      httpGet:
        host: 192.168.64.5
        path: /livez
        port: 8443
        scheme: HTTPS
      initialDelaySeconds: 10
      periodSeconds: 10
      timeoutSeconds: 15
    volumeMounts:
    - mountPath: /etc/ssl/certs
      name: ca-certs
      readOnly: true
    - mountPath: /var/lib/minikube/certs
      name: k8s-certs
      readOnly: true
    - mountPath: /usr/share/ca-certificates
      name: usr-share-ca-certificates
      readOnly: true
  hostNetwork: true
  priorityClassName: system-node-critical
  volumes:
  - hostPath:
      path: /etc/ssl/certs
      type: DirectoryOrCreate
    name: ca-certs
  - hostPath:
      path: /var/lib/minikube/certs
      type: DirectoryOrCreate
    name: k8s-certs
  - hostPath:
      path: /usr/share/ca-certificates
      type: DirectoryOrCreate
    name: usr-share-ca-certificates
status: {}
```

```
sudo systemctl restart kubelet
```

## 5.7. Exiting due to MK_ENABLE: run callbacks: running callbacks: [verifying registry addon pods : timed out waiting for the condition: timed out waiting for the condition]

```
iMac:~ neo$ minikube addons enable registry
🔎  Verifying registry addon...

❌  Exiting due to MK_ENABLE: run callbacks: running callbacks:
[verifying registry addon pods : timed out waiting for the
condition: timed out waiting for the condition]

😿  If the above advice does not help, please let us know:
```

```
👉  https://github.com/kubernetes/minikube/issues/new/choose
```

### 5.8. Exiting due to SVC_URL_TIMEOUT: http://127.0.0.1:11068/api/v1/namespaces/kubernetes-dashboard/services/http:kubernetes-dashboard:/proxy/ is not accessible: Temporary Error: unexpected response code: 503

```
minikube dashboard --alsologtostderr -v=1
```

```
[docker@localhost ~]$ kubectl get pods --all-namespaces | grep
dashboard
kubernetes-dashboard   dashboard-metrics-scraper-6f7955cd98-
xjzkq   0/1     ImagePullBackOff    0          11d
kubernetes-dashboard   kubernetes-dashboard-7bf64fd654-ckr7v
0/1     ImagePullBackOff    0          11d
```

```
[docker@localhost ~]$ kubectl logs --namespace=kubernetes-
dashboard kubernetes-dashboard-7bf64fd654-ckr7v
Error from server (BadRequest): container "kubernetes-
dashboard" in pod "kubernetes-dashboard-7bf64fd654-ckr7v" is
waiting to start: trying and failing to pull image
```

### 5.9. Mac minikube ip 不通，ingress 不工作

```
minikube start --image-mirror-country=cn --insecure-
```

```
registry="registry.netkiller.cn" --cache-images=true
```

```
Neo-iMac:~ neo$ kubectl get pods -n ingress-nginx
NAME                                        READY   STATUS
RESTARTS   AGE
ingress-nginx-admission-create--1-qpckk     0/1     Completed
0          18h
ingress-nginx-admission-patch--1-5x94l      0/1     Completed
0          18h
ingress-nginx-controller-78d858bdc7-nrszs   1/1     Running
1          18h

Neo-iMac:~ neo$ kubectl create deployment web --
image=nginx:latest
deployment.apps/web created

Neo-iMac:~ neo$ kubectl expose deployment web --type=NodePort -
-port=80
service/web exposed

Neo-iMac:~ neo$ kubectl get service web
NAME   TYPE       CLUSTER-IP      EXTERNAL-IP   PORT(S)
AGE
web    NodePort   10.109.55.204   <none>        8080:30857/TCP
19s

Neo-iMac:~ neo$ minikube service web --url
🏃  Starting tunnel for service web.
|-----------|------|-------------|------------------------|
| NAMESPACE | NAME | TARGET PORT |          URL           |
|-----------|------|-------------|------------------------|
| default   | web  |             | http://127.0.0.1:62956 |
|-----------|------|-------------|------------------------|
http://127.0.0.1:62956
❗  Because you are using a Docker driver on darwin, the
terminal needs to be open to run it.
```

ingress.yaml

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: nginx
  annotations:
    kubernetes.io/ingress.class: nginx
spec:
  rules:
    - host: www.netkiller.cn
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: web
                port:
                  number: 80
```

http://www.netkiller.cn 无法访问，解决方案 minikube tunnel

```
Neo-iMac:~ neo$ minikube tunnel
❗  The service/ingress example-ingress requires privileged 
ports to be exposed: [80 443]
🔑  sudo permission will be asked for it.
🏃  Starting tunnel for service example-ingress.
Password:
```

如果注意观察，在启动的时候系统已经提示：After the addon is enabled, please run "minikube tunnel" and your ingress resources would be available at "127.0.0.1"

```
Neo-iMac:nginx neo$ minikube start --image-mirror-country=cn --
```

```
insecure-registry="registry.netkiller.cn" --cache-images=true
😄   minikube v1.24.0 on Darwin 12.0.1
✨   Using the docker driver based on existing profile
👍   Starting control plane node minikube in cluster minikube
🚜   Pulling base image ...
🔄   Restarting existing docker container for "minikube" ...
🐳   Preparing Kubernetes v1.22.3 on Docker 20.10.8 ...
🔎   Verifying Kubernetes components...
💡   After the addon is enabled, please run "minikube tunnel"
and your ingress resources would be available at "127.0.0.1"
    ▪ Using image registry.cn-
hangzhou.aliyuncs.com/google_containers/dashboard:v2.3.1
    ▪ Using image registry.cn-
hangzhou.aliyuncs.com/google_containers/storage-provisioner:v5
    ▪ Using image registry.cn-
hangzhou.aliyuncs.com/google_containers/nginx-ingress-
controller:v1.0.4
    ▪ Using image registry.cn-
hangzhou.aliyuncs.com/google_containers/metrics-scraper:v1.0.7
    ▪ Using image registry.cn-
hangzhou.aliyuncs.com/google_containers/kube-webhook-
certgen:v1.1.1
    ▪ Using image registry.cn-
hangzhou.aliyuncs.com/google_containers/kube-webhook-
certgen:v1.1.1
🔎   Verifying ingress addon...
🌟   Enabled addons: dashboard, storage-provisioner, default-
storageclass, ingress
🏄   Done! kubectl is now configured to use "minikube" cluster
and "default" namespace by default
```

# 第 4 章 microk8s

https://microk8s.io

更多配置参考官网 https://github.com/ubuntu/microk8s

## 1. 安装 microk8s

latest/stable 安装最新版本

```
root@kubernetes:~# snap install microk8s --classic --
channel=latest/stable
microk8s v1.21.3 from Canonical✓ installed
```

查看安装情况

```
root@kubernetes:~# snap list
Name      Version   Rev    Tracking       Publisher   Notes
core18    20210722  2128   latest/stable  canonical✓  base
lxd       4.0.7     21029  4.0/stable/…   canonical✓  -
microk8s  v1.21.3   2346   latest/stable  canonical✓  classic
snapd     2.51.4    12883  latest/stable  canonical✓  snapd
```

```
root@kubernetes:~# microk8s start
Started.
```

启用或禁用 microk8s

```
snap disable microk8s   # 禁用
snap enable microk8s    # 启用
```

卸载

```
microk8s.reset
snap remove microk8s
```

安装 VirtualBox

```
neo@ubuntu:~$ sudo apt install -y virtualbox
```

## 1.1. 安装指定版本

```
root@kubernetes:~# snap info microk8s
name:      microk8s
summary:   Lightweight Kubernetes for workstations and
appliances
publisher: Canonical✓
store-url: https://snapcraft.io/microk8s
contact:   https://github.com/ubuntu/microk8s
license:   unset
description: |
  MicroK8s is the smallest, simplest, pure production
Kubernetes for clusters, laptops, IoT and
  Edge, on Intel and ARM. One command installs a single-node
K8s cluster with carefully selected
  add-ons on Linux, Windows and macOS.  MicroK8s requires no
configuration, supports automatic
```

```
  updates and GPU acceleration. Use it for offline development,
prototyping, testing, to build your
  CI/CD pipeline or your IoT apps.
commands:
  - microk8s.add-node
  - microk8s.cilium
  - microk8s.config
  - microk8s.ctr
  - microk8s.dashboard-proxy
  - microk8s.dbctl
  - microk8s.disable
  - microk8s.enable
  - microk8s.helm
  - microk8s.helm3
  - microk8s.inspect
  - microk8s.istioctl
  - microk8s.join
  - microk8s.juju
  - microk8s.kubectl
  - microk8s.leave
  - microk8s.linkerd
  - microk8s
  - microk8s.refresh-certs
  - microk8s.remove-node
  - microk8s.reset
  - microk8s.start
  - microk8s.status
  - microk8s.stop
services:
  microk8s.daemon-apiserver:            simple, enabled,
inactive
  microk8s.daemon-apiserver-kicker:     simple, enabled, active
  microk8s.daemon-cluster-agent:        simple, enabled, active
  microk8s.daemon-containerd:           simple, enabled, active
  microk8s.daemon-control-plane-kicker: simple, enabled,
inactive
  microk8s.daemon-controller-manager:   simple, enabled,
inactive
  microk8s.daemon-etcd:                 simple, enabled,
inactive
  microk8s.daemon-flanneld:             simple, enabled,
inactive
  microk8s.daemon-kubelet:              simple, enabled,
inactive
  microk8s.daemon-kubelite:             simple, enabled, active
  microk8s.daemon-proxy:                simple, enabled,
inactive
  microk8s.daemon-scheduler:            simple, enabled,
inactive
snap-id:      EaXqgt1lyCaxKaQCU349mlodBkDCXRcg
tracking:     latest/stable
refresh-date: today at 07:54 UTC
channels:
  1.21/stable:      v1.21.3  2021-07-27 (2346) 191MB classic
  1.21/candidate:   v1.21.4  2021-08-20 (2407) 191MB classic
  1.21/beta:        v1.21.4  2021-08-20 (2407) 191MB classic
  1.21/edge:        v1.21.4  2021-08-23 (2427) 191MB classic
  latest/stable:    v1.21.3  2021-07-28 (2346) 191MB classic
  latest/candidate: v1.22.1  2021-08-20 (2424) 195MB classic
  latest/beta:      v1.22.1  2021-08-20 (2424) 195MB classic
  latest/edge:      v1.22.1  2021-08-27 (2451) 195MB classic
  dqlite/stable:    –
  dqlite/candidate: –
  dqlite/beta:      –
  dqlite/edge:      v1.16.2  2019-11-07 (1038) 189MB classic
  1.22/stable:      v1.22.0  2021-08-13 (2399) 195MB classic
  1.22/candidate:   v1.22.1  2021-08-27 (2450) 195MB classic
  1.22/beta:        v1.22.1  2021-08-27 (2450) 195MB classic
  1.22/edge:        v1.22.1  2021-08-27 (2450) 195MB classic
  1.20/stable:      v1.20.9  2021-08-01 (2361) 221MB classic
  1.20/candidate:   v1.20.10 2021-08-19 (2409) 221MB classic
  1.20/beta:        v1.20.10 2021-08-19 (2409) 221MB classic
  1.20/edge:        v1.20.10 2021-08-12 (2409) 221MB classic
  1.19/stable:      v1.19.13 2021-07-26 (2339) 216MB classic
  1.19/candidate:   v1.19.14 2021-08-19 (2408) 216MB classic
  1.19/beta:        v1.19.14 2021-08-19 (2408) 216MB classic
  1.19/edge:        v1.19.14 2021-08-12 (2408) 216MB classic
  1.18/stable:      v1.18.20 2021-07-12 (2271) 198MB classic
  1.18/candidate:   v1.18.20 2021-07-12 (2271) 198MB classic
  1.18/beta:        v1.18.20 2021-07-12 (2271) 198MB classic
  1.18/edge:        v1.18.20 2021-06-16 (2271) 198MB classic
  1.17/stable:      v1.17.17 2021-01-15 (1916) 177MB classic
  1.17/candidate:   v1.17.17 2021-01-14 (1916) 177MB classic
  1.17/beta:        v1.17.17 2021-01-14 (1916) 177MB classic
  1.17/edge:        v1.17.17 2021-01-13 (1916) 177MB classic
  1.16/stable:      v1.16.15 2020-09-12 (1671) 179MB classic
  1.16/candidate:   v1.16.15 2020-09-04 (1671) 179MB classic
  1.16/beta:        v1.16.15 2020-09-04 (1671) 179MB classic
  1.16/edge:        v1.16.15 2020-09-02 (1671) 179MB classic
  1.15/stable:      v1.15.11 2020-03-27 (1301) 171MB classic
  1.15/candidate:   v1.15.11 2020-03-27 (1301) 171MB classic
  1.15/beta:        v1.15.11 2020-03-27 (1301) 171MB classic
  1.15/edge:        v1.15.11 2020-03-26 (1301) 171MB classic
  1.14/stable:      v1.14.10 2020-01-06 (1120) 217MB classic
  1.14/candidate:   ↑
  1.14/beta:        ↑
  1.14/edge:        v1.14.10 2020-03-26 (1303) 217MB classic
  1.13/stable:      v1.13.6  2019-06-06  (581) 237MB classic
  1.13/candidate:   ↑
  1.13/beta:        ↑
  1.13/edge:        ↑
  1.12/stable:      v1.12.9  2019-06-06  (612) 259MB classic
  1.12/candidate:   ↑
  1.12/beta:        ↑
  1.12/edge:        ↑
  1.11/stable:      v1.11.10 2019-05-10  (557) 258MB classic
  1.11/candidate:   ↑
  1.11/beta:        ↑
  1.11/edge:        ↑
  1.10/stable:      v1.10.13 2019-04-22  (546) 222MB classic
  1.10/candidate:   ↑
  1.10/beta:        ↑
  1.10/edge:        ↑
installed:          v1.21.3             (2346) 191MB classic
```

```
snap install microk8s --channel=1.14/beta --classic
```

## 2. 组件管理

```
root@kubernetes:~# microk8s enable ADDON -- --help
Addon ADDON does not yet have a help message.
For more information about it, visit
https://microk8s.io/docs/addons
```

启用组件

```
microk8s enable dashboard dns ingress istio registry storage
```

microk8s 只是最精简的安装，所以只有 api-server, controller-manager, scheduler, kubelet, cni, kube-proxy 被安装运行。额外的服务比如 kube-dns, dashboard 可以通过 microk8s.enable 启动

可用的扩展

```
dns
dashboard
storage
ingress
gpu
istio
registry
metrics-server
```

### 2.1. dns

```
microk8s.enable dns
禁用
microk8s.disable dns
```

## 2.2. dashboard

```
microk8s enable dashboard
```

```
root@kubernetes:~# microk8s enable dashboard
Enabling Kubernetes Dashboard
Addon metrics-server is already enabled.
Applying manifest
serviceaccount/kubernetes-dashboard created
service/kubernetes-dashboard created
secret/kubernetes-dashboard-certs created
secret/kubernetes-dashboard-csrf created
secret/kubernetes-dashboard-key-holder created
configmap/kubernetes-dashboard-settings created
role.rbac.authorization.k8s.io/kubernetes-dashboard created
clusterrole.rbac.authorization.k8s.io/kubernetes-dashboard
created
rolebinding.rbac.authorization.k8s.io/kubernetes-dashboard
created
clusterrolebinding.rbac.authorization.k8s.io/kubernetes-
dashboard created
deployment.apps/kubernetes-dashboard created
service/dashboard-metrics-scraper created
deployment.apps/dashboard-metrics-scraper created

If RBAC is not enabled access the dashboard using the default
token retrieved with:

token=$(microk8s kubectl -n kube-system get secret | grep
default-token | cut -d " " -f1)
microk8s kubectl -n kube-system describe secret $token

In an RBAC enabled setup (microk8s enable RBAC) you need to
create a user with restricted
permissions as shown in:
https://github.com/kubernetes/dashboard/blob/master/docs/user/a
ccess-control/creating-sample-user.md
```

```
microk8s dashboard-proxy
```

## 3. kubectl

为了不和已经安装的 kubectl 产生冲突，microk8s 有自己的 microk8s.kubectl 命令

```
microk8s.kubectl get services
```

如果本地没有 kubectl 命令可以增加一个别名

```
snap alias microk8s.kubectl kubectl
```

取消别名

```
snap unalias kubectl
```

API 服务监听 8080 端口

```
microk8s.kubectl config view
```

# 4. Kubernetes Addons

## 4.1.

```
root@kubernetes:~# microk8s kubectl get all --all-namespaces
NAMESPACE     NAME
READY   STATUS     RESTARTS   AGE
kube-system   pod/calico-kube-controllers-f7868dd95-xrt2w
0/1     Pending    0          83m
kube-system   pod/metrics-server-8bbfb4bdb-6m92q
0/1     Pending    0          74m
kube-system   pod/calico-node-vpsbv
0/1     Init:0/3   0          83m
kube-system   pod/kubernetes-dashboard-85fd7f45cb-w824z
0/1     Pending    0          114s
kube-system   pod/dashboard-metrics-scraper-78d7698477-g5b5k
0/1     Pending    0          114s

NAMESPACE     NAME                                TYPE
CLUSTER-IP       EXTERNAL-IP   PORT(S)    AGE
default       service/kubernetes                  ClusterIP
10.152.183.1     <none>        443/TCP    83m
kube-system   service/metrics-server              ClusterIP
10.152.183.99    <none>        443/TCP    74m
kube-system   service/kubernetes-dashboard        ClusterIP
10.152.183.225   <none>        443/TCP    114s
kube-system   service/dashboard-metrics-scraper   ClusterIP
10.152.183.11    <none>        8000/TCP   114s

NAMESPACE     NAME                         DESIRED   CURRENT
READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR            AGE
kube-system   daemonset.apps/calico-node   1         1
0       1            0           kubernetes.io/os=linux   83m

NAMESPACE     NAME                                      READY
UP-TO-DATE   AVAILABLE   AGE
kube-system   deployment.apps/calico-kube-controllers   0/1
1            0           83m
kube-system   deployment.apps/metrics-server            0/1
1            0           74m
```

```
kube-system   deployment.apps/kubernetes-dashboard        0/1
1            0           114s
kube-system   deployment.apps/dashboard-metrics-scraper   0/1
1            0           114s

NAMESPACE     NAME
DESIRED   CURRENT   READY   AGE
kube-system   replicaset.apps/calico-kube-controllers-f7868dd95
1         1         0       83m
kube-system   replicaset.apps/metrics-server-8bbfb4bdb
1         1         0       74m
kube-system   replicaset.apps/kubernetes-dashboard-85fd7f45cb
1         1         0       114s
kube-system   replicaset.apps/dashboard-metrics-scraper-
78d7698477   1         1         0       114s
```

# 第 5 章 Kubernetes 集群管理

## *kubectl - controls the Kubernetes cluster manager.*

kubectl是Kubernetes的命令行管理工具

```
kubectl controls the Kubernetes cluster manager.

Find more information at:
https://kubernetes.io/docs/reference/kubectl/overview/

Basic Commands (Beginner):
  create         Create a resource from a file or from stdin.
  expose         Take a replication controller, service,
deployment or pod and expose it as a new Kubernetes Service
  run            Run a particular image on the cluster
  set            Set specific features on objects

Basic Commands (Intermediate):
  explain        Documentation of resources
  get            Display one or many resources
  edit           Edit a resource on the server
  delete         Delete resources by filenames, stdin,
resources and names, or by resources and label selector

Deploy Commands:
  rollout        Manage the rollout of a resource
  scale          Set a new size for a Deployment, ReplicaSet,
Replication Controller, or Job
  autoscale      Auto-scale a Deployment, ReplicaSet, or
ReplicationController

Cluster Management Commands:
  certificate    Modify certificate resources.
  cluster-info   Display cluster info
  top            Display Resource (CPU/Memory/Storage) usage.
  cordon         Mark node as unschedulable
  uncordon       Mark node as schedulable
  drain          Drain node in preparation for maintenance
  taint          Update the taints on one or more nodes

Troubleshooting and Debugging Commands:
  describe       Show details of a specific resource or group
of resources
  logs           Print the logs for a container in a pod
  attach         Attach to a running container
  exec           Execute a command in a container
  port-forward   Forward one or more local ports to a pod
  proxy          Run a proxy to the Kubernetes API server
  cp             Copy files and directories to and from
containers.
  auth           Inspect authorization

Advanced Commands:
  diff           Diff live version against would-be applied
version
  apply          Apply a configuration to a resource by
filename or stdin
  patch          Update field(s) of a resource using strategic
merge patch
  replace        Replace a resource by filename or stdin
  wait           Experimental: Wait for a specific condition on
one or many resources.
  convert        Convert config files between different API
versions

Settings Commands:
  label          Update the labels on a resource
  annotate       Update the annotations on a resource
  completion     Output shell completion code for the specified
shell (bash or zsh)

Other Commands:
  api-resources  Print the supported API resources on the
server
  api-versions   Print the supported API versions on the
server, in the form of "group/version"
  config         Modify kubeconfig files
  plugin         Provides utilities for interacting with
plugins.
  version        Print the client and server version
information

Usage:
  kubectl [flags] [options]

Use "kubectl <command> --help" for more information about a
given command.
Use "kubectl options" for a list of global command-line options
(applies to all commands).
```

# 1. 配置

## 1.1. KUBECONFIG

KUBECONFIG 环境变量

## 1.2. use-context

```
[root@netkiller ~]# kubectl config view
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: DATA+OMITTED
    server: https://127.0.0.1:6445
  name: k3d-mycluster
contexts:
- context:
    cluster: k3d-mycluster
    user: admin@k3d-mycluster
  name: k3d-mycluster
current-context: k3d-mycluster
kind: Config
preferences: {}
users:
- name: admin@k3d-mycluster
  user:
    client-certificate-data: REDACTED
    client-key-data: REDACTED
```

```
$ kubectl config use-context
```

## 2. 如何从 docker 过渡到 kubectl 命令

docker run 命令

```
$ docker run -d --restart=always -e DOMAIN=cluster --name nginx
-p 80:80 nginx
```

kubectl 命令

```
$ kubectl run --image=nginx nginx-app --port=80 --
env="DOMAIN=cluster"
$ kubectl expose deployment nginx-app --port=80 --name=nginx-
http
```

docker exec 命令

```
$ docker run -t -i ubuntu:14.10 /bin/bash
```

kubectl 命令

```
$ kubectl exec -ti nginx-app-5jyvm -- /bin/sh
```

docker ps 命令

```
$ docker ps
```

kubectl 命令

```
$ kubectl get pods
NAME                        READY   STATUS    RESTARTS   AGE
mongodba-6d5d6ddf64-jw4fv   1/1     Running   0          16h

# kubectl exec -it mongodba-6d5d6ddf64-jw4fv bash
```

# 3. 查看信息

## 3.1. api-versions

```
iMac:springboot neo$ kubectl api-versions
admissionregistration.k8s.io/v1
admissionregistration.k8s.io/v1beta1
apiextensions.k8s.io/v1
apiextensions.k8s.io/v1beta1
apiregistration.k8s.io/v1
apiregistration.k8s.io/v1beta1
apps/v1
authentication.k8s.io/v1
authentication.k8s.io/v1beta1
authorization.k8s.io/v1
authorization.k8s.io/v1beta1
autoscaling/v1
autoscaling/v2beta1
autoscaling/v2beta2
batch/v1
batch/v1beta1
certificates.k8s.io/v1
certificates.k8s.io/v1beta1
coordination.k8s.io/v1
coordination.k8s.io/v1beta1
discovery.k8s.io/v1beta1
events.k8s.io/v1
events.k8s.io/v1beta1
extensions/v1beta1
networking.k8s.io/v1
networking.k8s.io/v1beta1
node.k8s.io/v1beta1
policy/v1beta1
rbac.authorization.k8s.io/v1
rbac.authorization.k8s.io/v1beta1
scheduling.k8s.io/v1
scheduling.k8s.io/v1beta1
storage.k8s.io/v1
storage.k8s.io/v1beta1
v1
```

```
```

## 3.2. 节点

```
[root@localhost ~]# kubectl get nodes
NAME       STATUS   ROLES    AGE   VERSION
minikube   Ready    master   23m   v1.13.2
```

### nodes

```
[root@localhost ~]# kubectl get nodes
NAME       STATUS   ROLES    AGE    VERSION
minikube   Ready    master   119m   v1.13.2
```

```
iMac:~ neo$ kubectl get node
NAME       STATUS   ROLES    AGE   VERSION
minikube   Ready    master   42h   v1.19.0

iMac:~ neo$ kubectl get node -o wide
NAME       STATUS   ROLES    AGE   VERSION   INTERNAL-IP
EXTERNAL-IP   OS-IMAGE              KERNEL-VERSION
CONTAINER-RUNTIME
minikube   Ready    master   42h   v1.19.0   192.168.64.2
<none>        Buildroot 2019.02.11   4.19.114
docker://19.3.12
```

## 3.3. 查询集群状态

```
[root@localhost ~]# kubectl get cs
NAME                 STATUS    MESSAGE              ERROR
controller-manager   Healthy   ok
scheduler            Healthy   ok
etcd-0               Healthy   {"health": "true"}
```

## 3.4. config

```
[root@localhost ~]# kubectl config view
apiVersion: v1
clusters:
- cluster:
    certificate-authority: /root/.minikube/ca.crt
    server: https://172.16.0.121:8443
  name: minikube
contexts:
- context:
    cluster: minikube
    user: minikube
  name: minikube
current-context: minikube
kind: Config
preferences: {}
users:
- name: minikube
  user:
    client-certificate: /root/.minikube/client.crt
    client-key: /root/.minikube/client.key
```

```
iMac:~ neo$ kubectl config view
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: DATA+OMITTED
    server: https://kubernetes.docker.internal:6443
  name: docker-desktop
- cluster:
    certificate-authority: /Users/neo/.minikube/ca.crt
    server: https://192.168.64.2:8443
  name: minikube
contexts:
- context:
    cluster: docker-desktop
    user: docker-desktop
  name: docker-desktop
- context:
    cluster: minikube
    user: minikube
  name: minikube
current-context: minikube
kind: Config
preferences: {}
users:
- name: docker-desktop
  user:
    client-certificate-data: REDACTED
    client-key-data: REDACTED
- name: minikube
  user:
    client-certificate:
/Users/neo/.minikube/profiles/minikube/client.crt
    client-key:
/Users/neo/.minikube/profiles/minikube/client.key
```

**use-context**

如果之前用其他方式运行Kubernetes，如 minikube, mircok8s 等等，可以使用下面命令切换。

```
$ kubectl config use-context docker-for-desktop
```

## 3.5. cluster-info

```
[root@localhost ~]# kubectl cluster-info
Kubernetes master is running at https://172.16.0.121:8443
KubeDNS is running at
https://172.16.0.121:8443/api/v1/namespaces/kube-
system/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl
cluster-info dump'.
```

# 4. namespace 命名空间

## 4.1. 查看命名空间

```
root@netkiller ~# kubectl get ns
NAME                 STATUS   AGE
default              Active   197d
kube-system          Active   197d
kube-public          Active   197d
kube-node-lease      Active   197d
longhorn-system      Active   195d
test                 Active   163d
gitlab               Active   156d
dev                  Active   155d
training             Active   133d
project              Active   24h

root@netkiller ~# kubectl get namespace
NAME                 STATUS   AGE
default              Active   197d
kube-system          Active   197d
kube-public          Active   197d
kube-node-lease      Active   197d
longhorn-system      Active   195d
test                 Active   163d
gitlab               Active   156d
dev                  Active   155d
training             Active   133d
project              Active   24h
```

## 4.2. 创建命名空间

```
$ kubectl create namespace new-namespace
```

## 4.3. 使用 yaml 创建命名空间

创建 jenkins-namespace.yaml

```
apiVersion: v1
kind: Namespace
metadata:
  name: jenkins-project
```

```
$ kubectl create -f jenkins-namespace.yaml
namespace ”jenkins-project“ created
```

## 4.4. 删除命名空间

```
root@netkiller ~# kubectl delete namespace new-namespace
namespace "new-namespace" deleted
```

## 5. label 标签

label 用于识别对象，管理关联关系等目的，如Pod、Service、Deployment、Node的关联。

```
kubectl label nodes <node-name> <label-key>=<label-value>
```

打标签，例如 disk-type=ssd

```
[root@master ~]# kubectl label nodes agent-1 disk-type=ssd
node/agent-1 labeled
```

查看标签

```
[root@master ~]# kubectl get node --show-labels
NAME          STATUS   ROLES    AGE   VERSION   LABELS
master   Ready    master   42d   v1.17.4
beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,kuber
netes.io/arch=amd64,kubernetes.io/hostname=master,kubernetes.io
/os=linux,node-role.kubernetes.io/master=
agent-1   Ready    <none>   42d   v1.17.4
beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,disk-
type=ssd,kubernetes.io/arch=amd64,kubernetes.io/hostname=agent-
1,kubernetes.io/os=linux
agent-2   Ready    <none>   42d   v1.17.4
beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,kuber
netes.io/arch=amd64,kubernetes.io/hostname=agent-
2,kubernetes.io/os=linux
```

删除标签

```
[root@master ~]# kubectl label nodes agent-1 disk-type-
node/agent-1 unlabeled
```

# 6. 服务管理

## 6.1. 列出服务

```
[root@localhost ~]# kubectl get service
NAME                TYPE        CLUSTER-IP     EXTERNAL-IP
PORT(S)             AGE
hello-minikube      NodePort    10.109.33.86   <none>
8080:30436/TCP      134m
kubernetes          ClusterIP   10.96.0.1      <none>
443/TCP             147m
```

排序

```
iMac:kubernetes neo$ kubectl get services --sort-
by=.metadata.name
NAME          TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)
AGE
kubernetes    ClusterIP   10.96.0.1        <none>        443/TCP
121m
my-service    ClusterIP   10.106.157.143   <none>
80/TCP,443/TCP   9m43s
```

## 6.2. 创建服务

创建 service.yaml 文件

```
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    app: MyApp
  ports:
  - name: http
    protocol: TCP
    port: 80
    targetPort: 80
  - name: https
    protocol: TCP
    port: 443
    targetPort: 443
```

```
iMac:kubernetes neo$ kubectl create -f service.yaml
service/my-service created
```

查看服务

```
iMac:kubernetes neo$ kubectl get service
NAME         TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)
AGE
kubernetes   ClusterIP   10.96.0.1        <none>        443/TCP
113m
my-service   ClusterIP   10.106.157.143   <none>
80/TCP,443/TCP   64s
```

查看 service 后端代理的 pod 的 ip，这里没有挂载 pod 所以显示 none

```
iMac:kubernetes neo$ kubectl get endpoints my-service
```

```
NAME         ENDPOINTS   AGE
my-service   <none>      2m20s
```

## 6.3. 查看服务详细信息

```
iMac:kubernetes neo$ kubectl describe service/registry
Name:                     registry
Namespace:                default
Labels:                   app=registry
Annotations:              <none>
Selector:                 app=registry
Type:                     NodePort
IP:                       10.10.0.188
Port:                     registry  5000/TCP
TargetPort:               5000/TCP
NodePort:                 registry  32050/TCP
Endpoints:                172.17.0.6:5000
Session Affinity:         None
External Traffic Policy:  Cluster
Events:                   <none>
```

查看服务

```
        > kubectl get service
        NAME                    TYPE           CLUSTER-IP
EXTERNAL-IP   PORT(S)                         AGE
        kubernetes              ClusterIP      10.43.0.1
<none>        443/TCP                         4d13h
        nacos                   ClusterIP      10.43.175.40
<none>        8848/TCP,9848/TCP,9555/TCP      4d13h
        redis                   NodePort       10.43.129.224
<none>        6379:31436/TCP                  42h
        kube-explorer           ClusterIP      10.43.208.84
<none>        80/TCP                          36h
        elasticsearch           ClusterIP      10.43.241.136
```

```
<none>        9200/TCP,9300/TCP             13h
        elasticsearch-data   ClusterIP      10.43.39.228
<none>        9300/TCP                      13h
        kibana               ClusterIP      10.43.193.15
<none>        80/TCP                        13h
        mysql                ExternalName   <none>
master        3306/TCP                      6m24s
        mongo                ExternalName   <none>
master        27017/TCP                     6m24s

        > kubectl get service -o wide
        NAME                 TYPE           CLUSTER-IP
EXTERNAL-IP   PORT(S)                       AGE      SELECTOR
        kubernetes           ClusterIP      10.43.0.1
<none>        443/TCP                       4d13h    <none>
        nacos                ClusterIP      10.43.175.40
<none>        8848/TCP,9848/TCP,9555/TCP    4d13h    app=nacos
        redis                NodePort       10.43.129.224
<none>        6379:31436/TCP                42h      app=redis
        kube-explorer        ClusterIP      10.43.208.84
<none>        80/TCP                        36h      app=kube-
explorer
        elasticsearch        ClusterIP      10.43.241.136
<none>        9200/TCP,9300/TCP             13h
app=elasticsearch,role=master
        elasticsearch-data   ClusterIP      10.43.39.228
<none>        9300/TCP                      13h
app=elasticsearch,role=data
        kibana               ClusterIP      10.43.193.15
<none>        80/TCP                        13h      app=kibana
        mysql                ExternalName   <none>
master        3306/TCP                      6m45s    <none>
        mongo                ExternalName   <none>
master        27017/TCP                     6m45s    <none>
```

## 6.4. 更新服务

```
kubectl replace -f service.yaml --force
```

## 6.5. 删除服务

```
kubectl delete service hello-minikube
```

## 6.6. clusterip

语法

```
$ kubectl create service clusterip NAME [--tcp=<port>:
<targetPort>] [--dry-run]
```

演示

```
kubectl create service clusterip my-service --tcp=5678:8080
```

headless 模式

```
kubectl create service clusterip my-service --clusterip="None"
```

### selector

```
        apiVersion: v1
        kind: Service
        metadata:
          name: spring-cloud-config-server
          namespace: default
          labels:
                app: springboot
        spec:
          ports: web
          - port: 8888
                targetPort: web
          clusterIP: 10.10.0.1
          selector:
                app: spring-cloud-config-server
```

## 6.7. 设置外部IP

报漏 80.11.12.10:80 地址

```
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
        app: MyApp
  ports:
        - name: http
          protocol: TCP
          port: 80
          targetPort: 9376
  externalIPs:
        - 80.11.12.10
```

## 6.8. externalname

语法

```
$ kubectl create service externalname NAME --external-name
external.name [--dry-run]
```

演示

```
kubectl create service externalname my-externalname --external-
name bar.com
```

绑定外部域名

```
apiVersion: v1
kind: Service
metadata:
  name: my-service
  namespace: prod
spec:
  type: ExternalName
  externalName: my.database.example.com
```

应用案例，在master节点宿主主机上安装了mysql和mongo地址，pod链接他们可以使用宿主IP链接，或者写 master 主机名。

我认为更好的方法使用使用 Service 做一层映射，然后使用统一容器域名访问 mysql.default.svc.cluster.local，mongo.default.svc.cluster.local

```
metadata:
    name: mysql
    namespace: default
  spec:
    ports:
          - name: mysql
            protocol: TCP
            port: 3306
            targetPort: 3306
    type: ExternalName
    externalName: master
  apiVersion: v1
  kind: Service
  ---
  metadata:
    name: mongo
    namespace: default
  spec:
    ports:
          - name: mongo
            protocol: TCP
            port: 27017
            targetPort: 27017
    type: ExternalName
    externalName: master
  apiVersion: v1
  kind: Service
```

## 6.9. loadbalancer

语法

```
$ kubectl create service loadbalancer NAME [--
tcp=port:targetPort] [--dry-run]
```

演示

```
kubectl create service loadbalancer my-lb --tcp=5678:8080
```

**LoadBalancer YAML**

一般 HTTP 服务通过 ingress 对外报漏服务，TCP 的 Socket 服务可以使用 LoadBalancer 进行报漏

```
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
        app: MyApp
  ports:
        - protocol: TCP
          port: 80
          targetPort: 9376
  clusterIP: 10.0.171.239
  type: LoadBalancer
status:
  loadBalancer:
        ingress:
        - ip: 192.0.2.127
```

```
apiVersion: v1
kind: Service
metadata:
  name: example-service
spec:
  selector:
        app: example
  ports:
        - port: 8765
          targetPort: 9376
    type: LoadBalancer
```

## 6.10. nodeport

语法

```
$ kubectl create service nodeport NAME [--tcp=port:targetPort]
[--dry-run]
```

演示

```
kubectl create service nodeport my-nodeport --tcp=5678:8080
```

### NodePort YAML

```
    apiVersion: v1
    kind: Service
    metadata:
      name: my-service
    spec:
      type: NodePort
      selector:
            app: MyApp
      ports:
              # By default and for convenience, the
`targetPort` is set to the same value as the `port` field.
            - port: 80
              targetPort: 80
```

```
                  # Optional field
                  # By default and for convenience, the
Kubernetes control plane will allocate a port from a range
(default: 30000-32767)
                  nodePort: 30007
```

## 6.11. Example

```


apiVersion: v1
kind: Service
metadata:
  name: registry
  namespace: default
  labels:
    app: registry
spec:
  type: NodePort
  selector:
    app: registry
  ports:
  - name: registry
    port: 5000
    nodePort: 30050
    protocol: TCP
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: registry
  namespace: default
  labels:
    app: registry
spec:
  replicas: 1
  selector:
    matchLabels:
      app: registry
  template:
    metadata:
      labels:
```

```
      app: registry
  spec:
    containers:
    - name: registry
      image: registry:latest
      resources:
        limits:
          cpu: 100m
          memory: 100Mi
      env:
      - name: REGISTRY_HTTP_ADDR
        value: :5000
      - name: REGISTRY_STORAGE_FILESYSTEM_ROOTDIRECTORY
        value: /var/lib/registry
      ports:
      - containerPort: 5000
        name: registry
        protocol: TCP
```

## 7. serviceaccount

语法

```
$ kubectl create serviceaccount NAME [--dry-run]
```

演示

```
kubectl create serviceaccount my-service-account
```

```
apiVersion: v1
kind: ServiceAccount
metadata:
 labels:
   app: elasticsearch
 name: elasticsearch
 namespace: elastic
```

# 8. 部署管理

```
kubectl create -f
https://raw.githubusercontent.com/kubernetes/dashboard/master/s
rc/deploy/recommended/kubernetes-dashboard.yaml
kubectl get pods --namespace=kube-system
```

## 8.1. Pod 管理

Pod 状态说明

Pod 状态:

- Pending：Pod已经被创建，但还没有完成调度，或者说有一个或多个镜像正处于从远程仓库下载的过程。处在这个阶段的Pod可能正在写数据到etcd中、调度、pull镜像或启动容器。
- Pending：Pod已经被创建，但还没有完成调度，或者说有一个或多个镜像正处于从远程仓库下载的过程。处在这个阶段的Pod可能正在写数据到etcd中、调度、pull镜像或启动容器。
- Running：该Pod已经绑定到了一个节点上，Pod中所有的容器都已被创建。至少有一个容器正在运行，或者正处于启动或重启状态。
- Succeeded：Pod中的所有的容器已经正常的执行后退出，并且不会自动重启，一般会是在部署job的时候会出现。
- Failed：Pod中的所有容器都已终止了，并且至少有一个容器是因为失败终止。也就是说，容器以非0状态退出或者被系统终止。
- Unkonwn：APIServer无法正常获取到Pod对象的状态信息，通常是由于其无法与所在工作节点的kubelet通信所致。

Pod 错误的详细的说明

```
状态                                                描述
CrashLoopBackOff                容器退出，kubelet正在将它重启
InvalidImageName                无法解析镜像名称
ImageInspectError               无法校验镜像
ErrImageNeverPull               策略禁止拉取镜像
ImagePullBackOff                正在重试拉取
RegistryUnavailable             连接不到镜像中心
ErrImagePull                    通用的拉取镜像出错
CreateContainerConfigError      不能创建kubelet使用的容器配置
CreateContainerError    创建容器失败
m.internalLifecycle.PreStartContainer   执行hook报错
RunContainerError               启动容器失败
PostStartHookError              执行hook报错
ContainersNotInitialized        容器没有初始化完毕
ContainersNotRead               容器没有准备完毕
ContainerCreating               容器创建中
PodInitializing pod     初始化中
DockerDaemonNotReady    docker还没有完全启动
NetworkPluginNotReady   网络插件还没有完全启动
```

## 查看 POD 状态

```
kubectl get pod <pod-name> -o wide
kubectl get pods --all-namespaces
```

查看默认命名空间下的 pod

```
[root@localhost ~]# kubectl get pod
NAME                                  READY    STATUS     RESTARTS
AGE
hello-minikube-5c856cbf98-6vfvp   1/1      Running    0
6m59s
```

查看所有命名空间下的 Pod

```
[root@localhost ~]# kubectl get pods --all-namespaces
NAMESPACE     NAME                                   READY
STATUS    RESTARTS   AGE
default       hello-minikube-5c856cbf98-6vfvp        1/1
Running   1          4d18h
kube-system   coredns-86c58d9df4-2rfqf               1/1
Running   51         4d18h
kube-system   coredns-86c58d9df4-wkb7l               1/1
Running   49         4d18h
kube-system   etcd-minikube                          1/1
Running   12         4d18h
kube-system   kube-addon-manager-minikube            1/1
Running   11         4d18h
kube-system   kube-apiserver-minikube                1/1
Running   74         4d18h
kube-system   kube-controller-manager-minikube       1/1
Running   31         4d18h
kube-system   kube-proxy-brrdd                       1/1
Running   1          4d18h
kube-system   kube-scheduler-minikube                1/1
Running   31         4d18h
kube-system   kubernetes-dashboard-ccc79bfc9-dxcq2   1/1
Running   7          4d17h
kube-system   storage-provisioner                    1/1
Running   2          4d18h
```

```
iMac:~ neo$ kubectl get pods --output=wide
NAME                          READY   STATUS             RESTARTS
AGE   IP           NODE       NOMINATED NODE   READINESS GATES
registry-65854b565b-bkhvq   0/1     ImagePullBackOff   0
18m   172.17.0.4   minikube   <none>           <none>
```

查看pod标签

```
kubectl get pods --show-labels
```

查看指定标签的pod

```
kubectl get pods -l run=nginx
```

指定命名空间

```
[root@localhost ~]# kubectl get pod --namespace=kube-system
NAME                                   READY   STATUS
RESTARTS   AGE
coredns-86c58d9df4-2rfqf               1/1     Running   0
40m
coredns-86c58d9df4-wkb7l               1/1     Running   0
40m
etcd-minikube                          1/1     Running   0
40m
kube-addon-manager-minikube            1/1     Running   0
41m
kube-apiserver-minikube                1/1     Running   2
40m
kube-controller-manager-minikube       1/1     Running   6
40m
kube-proxy-brrdd                       1/1     Running   0
40m
kube-scheduler-minikube                1/1     Running   5
41m
kubernetes-dashboard-ccc79bfc9-dxcq2   1/1     Running   5
16m
storage-provisioner                    1/1     Running   0
39m
```

格式化输出

```
neo@Netkiller-iMac ~> kubectl get pods -l app=nacos -o
jsonpath='{.items[0].metadata.name}'
nacos-0⏎
```

查看 **pod** 下面容器

```
root@logging ~# kubectl --kubeconfig=/home/prod/.kube/config -n
netkiller get pod neo-6787cfcb9-8s8pp -o jsonpath="
{.spec.containers[*].name}"
filebeat neo
```

## 运行 POD

```
iMac:kubernetes neo$ kubectl run registry --
image=registry:latest
```

```
kubectl run busybox --image=busybox --command -- ping
www.netkiller.cn
```

```
kubectl run nginx --replicas=3 --labels="app=example" --
image=nginx:latest --port=80
```

```
kubectl run busybox --rm=true --image=busybox --restart=Never -
it
```

## 删除 pod

```
kubectl delete -n default pod registry
kubectl delete -n default pod counter
```

## 查看 Pod 的事件

```
kubectl describe pod <pod-name>
```

```
iMac:~ neo$ kubectl describe pod springboot
Name:           springboot
Namespace:      default
Priority:       0
Node:           minikube/192.168.64.2
Start Time:     Mon, 21 Sep 2020 16:17:03 +0800
Labels:         run=springboot
Annotations:    <none>
Status:         Pending
IP:
IPs:            <none>
Containers:
  springboot:
    Container ID:
    Image:          127.0.0.1:5000/netkiller/config:latest
    Image ID:
    Port:           8888/TCP
    Host Port:      0/TCP
    State:          Waiting
      Reason:       ContainerCreating
    Ready:          False
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from
default-token-fhfn8 (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             False
  ContainersReady   False
  PodScheduled      True
Volumes:
  default-token-fhfn8:
    Type:        Secret (a volume populated by a Secret)
    SecretName:  default-token-fhfn8
    Optional:    false
QoS Class:       BestEffort
Node-Selectors:  <none>
Tolerations:     node.kubernetes.io/not-ready:NoExecute
op=Exists for 300s
                 node.kubernetes.io/unreachable:NoExecute
op=Exists for 300s
Events:
  Type    Reason     Age   From                 Message
  ----    ------     ----  ----                 -------
  Normal  Scheduled  80s   default-scheduler    Successfully
assigned default/springboot to minikube
  Normal  Pulling    79s   kubelet              Pulling image
"127.0.0.1:5000/netkiller/config:latest"
```

### Taint（污点）和 Toleration（容忍）

其目的是分配 pod 在集群间的调度，Taint 和 toleration 相互配合，可以用来避免 pod 被分配到某个节点上。这跟节点亲和性作用相反。

给 node 节点设置 label，通过给 pod 设置 nodeSelector 将 pod 调度到匹配标签的节点上。

如果设置 toleration 应用于 pod 上，则表示 pod 可以被调度到 taint 的节点上。

**Taint（污点）设置**

设置污点: kubectl taint node [node] key=value:[effect]

effect 参数

1. NoSchedule： 不能被调度。
2. PreferNoSchedule：尽量不要调度。
3. NoExecute：不允许该节点有 Pod。

在 shenzhen 节点上设置Taint，键为key，值为value，effect是NoSchedule。

```
kubectl taint nodes shenzhen key=value:NoSchedule
```

这意味着除非pod只有明确声明toleration可以容忍这个Taint，否则就不会被调度到该节点。

```
apiVersion: v1
kind: Pod
metadata:
  name: pod-taints
spec:
  tolerations:
  - key: "key"
    operator: "Equal"
    value: "value"
    effect: "NoSchedule"
  containers:
    - name: pod-taints
      image: busybox:latest
```

**Toleration（容忍）调度**

key 存在即可匹配

```
spec:
  tolerations:
  - key: "key"
    operator: "Exists"
    effect: "NoSchedule"
```

key 必须存在，并且值等 value

```
spec:
  tolerations:
  - key: "key"
    operator: "Equal"
    value: "value"
    effect: "NoSchedule"
```

在pod上设置多个toleration：

```
spec:
  tolerations:
  - key: "key1"
    operator: "Equal"
    value: "value1"
    effect: "NoSchedule"
  - key: "key2"
    operator: "Equal"
    value: "value2"
    effect: "NoExecute"
```

如果给node加上Taint effect=NoExecute的，该节点上的没有设置toleration的pod都会被立刻驱逐，设置 tolerationSeconds 后会给 Pod 一个宽限期。

```
spec:
  tolerations:
  - key: "key"
    operator: "Equal"
    value: "value"
    effect: "NoSchedule"
    tolerationSeconds: 3600
```

使用场景

例如有些节点上挂了SSD，给redis,mongodb,mysql 使用，有些节点上安装了显卡GPU。就可以使用 taint

```
kubectl taint nodes shenzhen special=true:NoSchedule
kubectl taint nodes guangdong special=true:PreferNoSchedule
```

**Pod**

```
apiVersion: v1
kind: Pod
metadata:
  name: counter
spec:
  containers:
  - name: count
    image: busybox
    args: [/bin/sh, -c, 'i=0; while true; do echo "$i:
$(date)"; i=$((i+1)); sleep 1; done']
```

创建 pod

```
iMac:kubernetes neo$ kubectl create -f pod.yaml
pod/counter created

iMac:kubernetes neo$ kubectl logs counter
0: Sun Oct  4 12:32:44 UTC 2020
1: Sun Oct  4 12:32:45 UTC 2020
2: Sun Oct  4 12:32:46 UTC 2020
3: Sun Oct  4 12:32:47 UTC 2020
4: Sun Oct  4 12:32:48 UTC 2020
5: Sun Oct  4 12:32:49 UTC 2020
6: Sun Oct  4 12:32:50 UTC 2020
7: Sun Oct  4 12:32:51 UTC 2020
8: Sun Oct  4 12:32:52 UTC 2020
9: Sun Oct  4 12:32:53 UTC 2020
```

### 容器编排

#### 镜像拉取策略

imagePullPolicy: Always 总是拉取

imagePullPolicy: IfNotPresent 默认值,本地有则使用本地镜像,不拉取

imagePullPolicy: Never 只使用本地镜像，从不拉取

指定主机名

```
apiVersion: v1
kind: Pod
metadata:
  name: hostaliases-pod
spec:
  restartPolicy: Never
  hostAliases:
  - ip: "127.0.0.1"
    hostnames:
    - "foo.local"
    - "bar.local"
  - ip: "10.1.2.3"
    hostnames:
    - "foo.remote"
    - "bar.remote"
  containers:
  - name: cat-hosts
    image: busybox
    command:
    - cat
    args:
    - "/etc/hosts"
```

环境变量

```
apiVersion: v1
kind: Pod
metadata:
  name: envars-fieldref
```

```
spec:
  containers:
    - name: test-container
      image: k8s.gcr.io/busybox
      command: [ "sh", "-c"]
      args:
      - while true; do
          echo -en '\n';
          printenv NODE_NAME POD_NAME POD_NAMESPACE;
          printenv POD_IP POD_SERVICE_ACCOUNT;
          sleep 10;
        done;
      env:
        - name: NODE_NAME
          valueFrom:
            fieldRef:
              fieldPath: spec.nodeName
        - name: POD_NAME
          valueFrom:
            fieldRef:
              fieldPath: metadata.name
        - name: POD_NAMESPACE
          valueFrom:
            fieldRef:
              fieldPath: metadata.namespace
        - name: POD_IP
          valueFrom:
            fieldRef:
              fieldPath: status.podIP
        - name: POD_SERVICE_ACCOUNT
          valueFrom:
            fieldRef:
              fieldPath: spec.serviceAccountName
  restartPolicy: Never
```

```
apiVersion: v1
kind: Pod
metadata:
  name: envars-resourcefieldref
spec:
```

```
containers:
  - name: test-container
    image: k8s.gcr.io/busybox:1.24
    command: [ "sh", "-c"]
    args:
    - while true; do
        echo -en '\n';
        printenv CPU_REQUEST CPU_LIMIT;
        printenv MEM_REQUEST MEM_LIMIT;
        sleep 10;
      done;
    resources:
      requests:
        memory: "32Mi"
        cpu: "125m"
      limits:
        memory: "64Mi"
        cpu: "250m"
    env:
      - name: CPU_REQUEST
        valueFrom:
          resourceFieldRef:
            containerName: test-container
            resource: requests.cpu
      - name: CPU_LIMIT
        valueFrom:
          resourceFieldRef:
            containerName: test-container
            resource: limits.cpu
      - name: MEM_REQUEST
        valueFrom:
          resourceFieldRef:
            containerName: test-container
            resource: requests.memory
      - name: MEM_LIMIT
        valueFrom:
          resourceFieldRef:
            containerName: test-container
            resource: limits.memory
restartPolicy: Never
```

就绪探针

```
        readinessProbe:
          exec:
            command:
            - cat
            - /tmp/healthy
          initialDelaySeconds: 10         #10s之后开始第一次探测
          periodSeconds: 5                #第一次探测之后每隔5s探测
一次
```

**securityContext**

**sysctls**

```
kubelet --allowed-unsafe-sysctls \
  'kernel.msg*,net.core.somaxconn' ...
```

```
apiVersion: v1
kind: Pod
metadata:
  name: sysctl-example
spec:
  securityContext:
    sysctls:
    - name: kernel.shm_rmid_forced
      value: "0"
    - name: net.core.somaxconn
      value: "1024"
    - name: kernel.msgmax
      value: "65536"
```

**runAsUser**

allowPrivilegeEscalation 表示是否继承父进程权限，runAsUser 表示使用 UID 1000 的用户运行

```
apiVersion: v1
kind: Pod
metadata:
  name: security-context-demo
spec:
  securityContext:
    runAsUser: 1000
  containers:
  - name: sec-ctx-demo
    image: busybox:latest
    securityContext:
      runAsUser: 1000
      allowPrivilegeEscalation: false
```

```
   spec:
     securityContext:
        runAsUser: 1000
        fsGroup: 2000
        runAsNonRoot: true
```

**security.alpha.kubernetes.io/sysctls**

security.alpha.kubernetes.io/sysctls

```
apiVersion: v1
kind: Pod
metadata:
  name: sysctl-example
  annotations:
    security.alpha.kubernetes.io/sysctls: 
kernel.shm_rmid_forced=1
spec:
```

unsafe-sysctls

```
apiVersion: v1
kind: Pod
metadata:
  name: sysctl-example
  annotations:
    security.alpha.kubernetes.io/unsafe-sysctls: 
net.core.somaxconn=65535                  #使用unsafe sysctl，设置
最大连接数
spec:
  securityContext:
    privileged: true                                            
#开启privileged权限
```

**nodeName 选择节点**

首先查看节点名称

```
[root@master ~]# kubectl get node
NAME       STATUS   ROLES                    AGE     VERSION
agent-1    Ready    <none>                   2d13h   v1.24.4+k3s1
master     Ready    control-plane,master     2d13h   v1.24.4+k3s1
agent-2    Ready    <none>                   13h     v1.24.4+k3s1
```

使用 nodeName: master 选择节点

```yaml
metadata:
  name: redis
  labels:
    app: redis
spec:
  replicas: 1
  serviceName: redis
  selector:
    matchLabels:
      app: redis
  template:
    metadata:
      labels:
        app: redis
    spec:
      containers:
        - name: redis
          image: redis:latest
          ports:
            - containerPort: 6379
          volumeMounts:
            - name: data
              mountPath: /data
            - name: config
              mountPath: /usr/local/etc/redis.conf
              subPath: redis.conf
          livenessProbe:
            tcpSocket:
              port: 6379
            initialDelaySeconds: 60
            failureThreshold: 3
            periodSeconds: 10
            successThreshold: 1
            timeoutSeconds: 5
          readinessProbe:
            tcpSocket:
              port: 6379
            initialDelaySeconds: 5
            failureThreshold: 3
            periodSeconds: 10
            successThreshold: 1
            timeoutSeconds: 5
      volumes:
        - name: data
          persistentVolumeClaim:
            claimName: redis
        - name: config
          configMap:
            name: redis
      nodeName: master
  volumeClaimTemplates:
    - metadata:
        name: data
      spec:
        accessModes:
          - ReadWriteOnce
        storageClassName: longhorn
        resources:
          requests:
            storage: 2Gi
apiVersion: apps/v1
kind: StatefulSet
```

**nodeSelector 选择节点**

首先给节点打标签，例如 disk-type=ssd

```
[root@master ~]# kubectl label nodes agent-1 disk-type=ssd
node/agent-1 labeled
```

查看标签

```
[root@master ~]# kubectl get node --show-labels
NAME         STATUS   ROLES    AGE   VERSION   LABELS
master   Ready    master   42d   v1.17.4
beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,kuber
netes.io/arch=amd64,kubernetes.io/hostname=master,kubernetes.io
```

```
/os=linux,node-role.kubernetes.io/master=
agent-1   Ready    <none>   42d   v1.17.4
beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,disk-
type=ssd,kubernetes.io/arch=amd64,kubernetes.io/hostname=agent-
1,kubernetes.io/os=linux
agent-2   Ready    <none>   42d   v1.17.4
beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,kuber
netes.io/arch=amd64,kubernetes.io/hostname=agent-
2,kubernetes.io/os=linux
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: busybox
  labels:
    app: busybox
spec:
  replicas: 5
  selector:
    matchLabels:
      app: busybox
  template:
    metadata:
      labels:
        app: busybox
    spec:
      containers:
      - name: busybox
        image: busybox
        imagePullPolicy: IfNotPresent
        ports:
          - containerPort: 80
      # 指定标签节点
      nodeSelector:
        disk-type: ssd
```

删除标签

```
[root@master ~]# kubectl label nodes agent-1 disk-type-
node/agent-1 unlabeled
```

**nodeAffinity** 选择节点

```
nodeAffinity可对应的两种策略:
preferredDuringScheduling(IgnoredDuringExecution /
RequiredDuringExecution) 软策略
requiredDuringScheduling(IgnoredDuringExecution /
RequiredDuringExecution) 硬策略

operator 表达式
In: label的值在某个列表中
NotIn: label的值不在某个列表中
Exists: 某个label存在
DoesNotExist: 某个label不存在
Gt: label的值大于某个值（字符串比较）
Lt: label的值小于某个值（字符串比较）
```

**Taint**（污点）和 **Toleration**（容忍）

```

```

**strategy**

滚动升级策略:

超过期望的Pod数量:1

不可用Pod最大数量:0

```
strategy:
  rollingUpdate:
    maxSurge: 1
    maxUnavailable: 0
  type: RollingUpdate
```

```
strategy:
    type: RollingUpdate
    rollingUpdate: {
      maxUnavailable: 25%
      maxSurge: 25%
```

## 8.2. expose

```
kubectl expose deployment nginx --port=88 --target-port=80 --
type=NodePort --name=nginx-service
kubectl describe service nginx-service
```

```
将服务暴露出去，在服务前面加一个负载均衡，因为pod可能分布在不同的结点上。
—port：暴露出去的端口
—type=NodePort：使用结点+端口方式访问服务
—target-port：容器的端口
—name：创建service指定的名称
```

```
kubectl expose deployment nginx --port=80 --target-port=8080 --
type=NodePort
```

```
kubectl expose deployment nginx --port=80 --target-port=8080 --
type=LoadBalancer
```

## 8.3.

```
kubectl create deployment registry --image=registry:latest
kubectl get deploy
```

## 8.4. 删除 deployment

```
kubectl delete deployment hello-minikube
```

## 8.5. 资源管理

```
kubectl scale -n default deployment nginx --replicas=1
kubectl scale deployment springbootdemo --replicas=4
kubectl scale deployment nginx --replicas=10
```

## 8.6. rollout

查看发布历史

```
kubectl rollout history deployment/nginx
```

指定版本号

```
kubectl rollout history deployment/nginx --revision=3
```

查看部署状态

```
kubectl rollout status deployment/nginx
```

回滚到上一个版本

```
kubectl rollout undo deployment/nginx-deployment
```

回滚到指定版本

```
kubectl rollout undo deployment/nginx-deployment --to-
revision=3
```

## 9. 查看 pod 日志

```
kubectl logs <pod-name>
kubectl logs --previous <pod-name>
kubectl logs -l app=your-app-name | grep "xxx"
kubectl logs --selector role=cool-app | grep "xxx"
```

## 10. endpoints

```
Neo-iMac:kubernetes neo$ rancher kubectl get endpoints nginx
NAME    ENDPOINTS                                      AGE
nginx   10.42.0.19:80,10.42.0.20:80,10.42.0.21:80   3m56s
```

## 11. 执行 Shell

进入容器内部.

```
$ kubectl get pods
NAME                           READY   STATUS    RESTARTS   AGE
mongodba-6d5d6ddf64-jw4fv      1/1     Running   0          16h

$ kubectl exec -it mongodba-6d5d6ddf64-jw4fv bash
```

```
kubectl run busybox --image=busybox:latest

iMac:kubernetes neo$ kubectl exec -it busybox -- nslookup
www.netkiller.cn
Server:         10.10.0.10
Address:        10.10.0.10:53

Non-authoritative answer:
www.netkiller.cn        canonical name = netkiller.github.io
Name:   netkiller.github.io
Address: 185.199.110.153
Name:   netkiller.github.io
Address: 185.199.108.153
Name:   netkiller.github.io
Address: 185.199.111.153
Name:   netkiller.github.io
Address: 185.199.109.153

*** Can't find www.netkiller.cn: No answer
```

## 12. edit

```
kubectl edit --namespace=kube-system rc kubernetes-dashboard
```

## 13. port-forward 端口映射

```
$ kubectl port-forward svc/demo 8080:8080
```

# 14. secret 密钥管理

## 14.1. 获取 Token

```
[gitlab-runner@agent-5 ~]$ kubectl get secrets -n gitlab -o
jsonpath="{.items[?
(@.metadata.annotations['kubernetes\.io/service-
account\.name']=='gitlab-runner')].data.token}" | base64 -d
eyJhbGciOiJSUzI1NiIsImtpZCI6IktCOHRvYlZOLXFPRmEyb1JWdlQxSzBvN0t
vZF9HNFBGRnlraDR5UU1jakkifQ.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2V
hY2NvdW50Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1lc3BhY2
UiOiJnaXRsYWIiLCJrdWJlcm5ldGVzLmlvL3NlcnZpY2VhY2NvdW50L3NlY3Jld
C5uYW1lIjoiZ2l0bGFiLXJ1bm5lci10b2tlbiIsImt1YmVybmV0ZXMuaW8vc2Vy
dmljZWFjY291bnQvc2VydmljZS1hY2NvdW50Lm5hbWUiOiJnaXRsYWItcnVubmV
yIiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9zZXJ2aWNlLWFjY291bn
QudWlkIjoiN2NiOGY2MzctNzliNC00NzliLWFmMDMtNGE4ZDZkOWIzZjM5Iiwic
3ViIjoic3lzdGVtOnNlcnZpY2VhY2NvdW50OmdpdGxhYjpnaXRsYWItcnVubmVy
In0.pU4-
8D4szeL8iud1SvesdN7nV7L3GLaNsa2UbsxkGQ4SDGN85zKTXJl6MtqDsuJB9HB
UlOTMnyEa0gCbgHOJlR3fd2HcegitrRLeybvUuotniiLpCPO7vAO-
oS5Fej7oUFBXqZJYIx-
xMbFoyt3rnGs273c_yE8avI8EGdEPNhOWRgF_GZBYstvwiEjO2IUDWbutzCTtGl
oPvJ5Ur0s7drLJkCQvT2nod5tSSnY5R0lpNyD2FodkFR28KU1EgFoHUnH_ERtUA
S5qObIETWSwm5SmCnd2Ogjh70DDxmIHSU-
saFU0zSqPpZ1oX9hgO9YMkcJXPHOEnqIVEagZ5CSf2w
```

## 14.2. 创建 Secret

```
$ cat <<EOF | kubectl create -f -
apiVersion: v1
kind: Secret
metadata:
  name: mysecret
type: Opaque
data:
  password: $(echo "passw0rd" | base64)
  username: $(echo "neo" | base64)
EOF
```

## 14.3. Private Registry 用户认证

```
kubectl create secret docker-registry docker-hub \
--docker-server=https://index.docker.io/v1/ \
--docker-username=netkiller \
--docker-password=password \
--docker-email=netkiller@msn.com
```

```
iMac:spring neo$ kubectl get secret
NAME                   TYPE
DATA   AGE
default-token-fhfn8    kubernetes.io/service-account-token   3
2d23h
docker-hub             kubernetes.io/dockerconfigjson        1
15s
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: springboot
spec:
  replicas: 3
  selector:
    matchLabels:
      app: springboot
  template:
    metadata:
      labels:
        app: springboot
    spec:
      containers:
      - name: springboot
        image: netkiller/config:latest
        imagePullPolicy: IfNotPresent
        ports:
        - containerPort: 8888
      imagePullSecrets:
        - name: docker-hub
```

```
kubectl delete -n default secret docker-hub
```

## 14.4. 配置TLS SSL

```
# 证书生成
mkdir cert && cd cert

# 生成 CA 自签证书

openssl genrsa -out ca-key.pem 2048
openssl req -x509 -new -nodes -key ca-key.pem -days 10000 -out
ca.pem -subj "/CN=kube-ca"

# 编辑 openssl 配置
cp /etc/pki/tls/openssl.cnf .
vim openssl.cnf

[req]
req_extensions = v3_req # 注释删掉
# 新增下面配置是
[ v3_req ]
basicConstraints = CA:FALSE
keyUsage = nonRepudiation, digitalSignature, keyEncipherment
subjectAltName = @alt_names
[alt_names]
```

```
DNS.1 = ns.netkiller.cn

# 生成证书
openssl genrsa -out ingress-key.pem 2048
openssl req -new -key ingress-key.pem -out ingress.csr -subj
"/CN=www.netkiller.cn" -config openssl.cnf
openssl x509 -req -in ingress.csr -CA ca.pem -CAkey ca-key.pem
-CAcreateserial -out ingress.pem -days 365 -extensions v3_req -
extfile openssl.cnf
```

```
kubectl create secret tls ingress-secret --namespace=kube-
system --key cert/ingress-key.pem --cert cert/ingress.pem
```

# 15. ConfigMap

ConfigMap 用于保存配置数据的键值，也可以用来保存配置文件。

## 15.1. 创建 Key-Value 配置项

从key-value字符串创建ConfigMap

```
neo@MacBook-Pro-Neo ~ % kubectl create configmap config --from-literal=nickname=netkiller
configmap/config created
```

```
neo@MacBook-Pro-Neo ~ % kubectl get configmap config -o go-template='{{.data}}'
map[nickname:netkiller]
```

创建多个KV对

```
neo@MacBook-Pro-Neo ~ % kubectl create configmap user --from-literal=username=neo --from-literal=nickname=netkiller --from-literal=age=35
configmap/user created

neo@MacBook-Pro-Neo ~ % kubectl get configmap user -o go-template='{{.data}}'
map[age:35 nickname:netkiller username:neo]%
```

```
neo@MacBook-Pro-Neo ~ % kubectl create configmap db-config --
from-literal=db.host=172.16.0.10 --from-literal=db.port='3306'
configmap/db-config created
neo@MacBook-Pro-Neo ~ % kubectl describe configmap db-config
Name:         db-config
Namespace:    default
Labels:       <none>
Annotations:  <none>

Data
====
db.port:
----
3306
db.host:
----
172.16.0.10
Events:  <none>
```

## 15.2. 从文件创建 ConfigMap

```
neo@MacBook-Pro-Neo ~ % kubectl create configmap passwd --from-
file=/etc/passwd
configmap/passwd created

neo@MacBook-Pro-Neo ~ % kubectl describe configmap passwd
Name:         passwd
Namespace:    default
Labels:       <none>
Annotations:  <none>

Data
====
passwd:
----
##
# User Database
#
```

```
# Note that this file is consulted directly only when the
system is running
# in single-user mode.  At other times this information is
provided by
# Open Directory.
#
# See the opendirectoryd(8) man page for additional information
about
# Open Directory.
##
nobody:*:-2:-2:Unprivileged User:/var/empty:/usr/bin/false
root:*:0:0:System Administrator:/var/root:/bin/sh
daemon:*:1:1:System Services:/var/root:/usr/bin/false
_uucp:*:4:4:Unix to Unix Copy
Protocol:/var/spool/uucp:/usr/sbin/uucico
_taskgated:*:13:13:Task Gate Daemon:/var/empty:/usr/bin/false
_networkd:*:24:24:Network Services:/var/networkd:/usr/bin/false
_installassistant:*:25:25:Install
Assistant:/var/empty:/usr/bin/false
_lp:*:26:26:Printing Services:/var/spool/cups:/usr/bin/false
_postfix:*:27:27:Postfix Mail
Server:/var/spool/postfix:/usr/bin/false
_scsd:*:31:31:Service Configuration
Service:/var/empty:/usr/bin/false
_ces:*:32:32:Certificate Enrollment
Service:/var/empty:/usr/bin/false
_appstore:*:33:33:Mac App Store
Service:/var/db/appstore:/usr/bin/false
_mcxalr:*:54:54:MCX AppLaunch:/var/empty:/usr/bin/false
_appleevents:*:55:55:AppleEvents
Daemon:/var/empty:/usr/bin/false
_geod:*:56:56:Geo Services Daemon:/var/db/geod:/usr/bin/false
_devdocs:*:59:59:Developer
Documentation:/var/empty:/usr/bin/false
_sandbox:*:60:60:Seatbelt:/var/empty:/usr/bin/false
_mdnsresponder:*:65:65:mDNSResponder:/var/empty:/usr/bin/false
_ard:*:67:67:Apple Remote Desktop:/var/empty:/usr/bin/false
_www:*:70:70:World Wide Web
Server:/Library/WebServer:/usr/bin/false
_eppc:*:71:71:Apple Events User:/var/empty:/usr/bin/false
_cvs:*:72:72:CVS Server:/var/empty:/usr/bin/false
_svn:*:73:73:SVN Server:/var/empty:/usr/bin/false
_mysql:*:74:74:MySQL Server:/var/empty:/usr/bin/false
_sshd:*:75:75:sshd Privilege
separation:/var/empty:/usr/bin/false
```

```
_qtss:*:76:76:QuickTime Streaming
Server:/var/empty:/usr/bin/false
_cyrus:*:77:6:Cyrus Administrator:/var/imap:/usr/bin/false
_mailman:*:78:78:Mailman List Server:/var/empty:/usr/bin/false
_appserver:*:79:79:Application Server:/var/empty:/usr/bin/false
_clamav:*:82:82:ClamAV Daemon:/var/virusmails:/usr/bin/false
_amavisd:*:83:83:AMaViS Daemon:/var/virusmails:/usr/bin/false
_jabber:*:84:84:Jabber XMPP Server:/var/empty:/usr/bin/false
_appowner:*:87:87:Application Owner:/var/empty:/usr/bin/false
_windowserver:*:88:88:WindowServer:/var/empty:/usr/bin/false
_spotlight:*:89:89:Spotlight:/var/empty:/usr/bin/false
_tokend:*:91:91:Token Daemon:/var/empty:/usr/bin/false
_securityagent:*:92:92:SecurityAgent:/var/db/securityagent:/usr
/bin/false
_calendar:*:93:93:Calendar:/var/empty:/usr/bin/false
_teamsserver:*:94:94:TeamsServer:/var/teamsserver:/usr/bin/fals
e
_update_sharing:*:95:-2:Update
Sharing:/var/empty:/usr/bin/false
_installer:*:96:-2:Installer:/var/empty:/usr/bin/false
_atsserver:*:97:97:ATS Server:/var/empty:/usr/bin/false
_ftp:*:98:-2:FTP Daemon:/var/empty:/usr/bin/false
_unknown:*:99:99:Unknown User:/var/empty:/usr/bin/false
_softwareupdate:*:200:200:Software Update
Service:/var/db/softwareupdate:/usr/bin/false
_coreaudiod:*:202:202:Core Audio
Daemon:/var/empty:/usr/bin/false
_screensaver:*:203:203:Screensaver:/var/empty:/usr/bin/false
_locationd:*:205:205:Location
Daemon:/var/db/locationd:/usr/bin/false
_trustevaluationagent:*:208:208:Trust Evaluation
Agent:/var/empty:/usr/bin/false
_timezone:*:210:210:AutoTimeZoneDaemon:/var/empty:/usr/bin/fals
e
_lda:*:211:211:Local Delivery Agent:/var/empty:/usr/bin/false
_cvmsroot:*:212:212:CVMS Root:/var/empty:/usr/bin/false
_usbmuxd:*:213:213:iPhone OS Device
Helper:/var/db/lockdown:/usr/bin/false
_dovecot:*:214:6:Dovecot
Administrator:/var/empty:/usr/bin/false
_dpaudio:*:215:215:DP Audio:/var/empty:/usr/bin/false
_postgres:*:216:216:PostgreSQL Server:/var/empty:/usr/bin/false
_krbtgt:*:217:-2:Kerberos Ticket Granting
Ticket:/var/empty:/usr/bin/false
_kadmin_admin:*:218:-2:Kerberos Admin
```

```
Service:/var/empty:/usr/bin/false
_kadmin_changepw:*:219:-2:Kerberos Change Password
Service:/var/empty:/usr/bin/false
_devicemgr:*:220:220:Device Management
Server:/var/empty:/usr/bin/false
_webauthserver:*:221:221:Web Auth
Server:/var/empty:/usr/bin/false
_netbios:*:222:222:NetBIOS:/var/empty:/usr/bin/false
_warmd:*:224:224:Warm Daemon:/var/empty:/usr/bin/false
_dovenull:*:227:227:Dovecot
Authentication:/var/empty:/usr/bin/false
_netstatistics:*:228:228:Network Statistics
Daemon:/var/empty:/usr/bin/false
_avbdeviced:*:229:-2:Ethernet AVB Device
Daemon:/var/empty:/usr/bin/false
_krb_krbtgt:*:230:-2:Open Directory Kerberos Ticket Granting
Ticket:/var/empty:/usr/bin/false
_krb_kadmin:*:231:-2:Open Directory Kerberos Admin
Service:/var/empty:/usr/bin/false
_krb_changepw:*:232:-2:Open Directory Kerberos Change Password
Service:/var/empty:/usr/bin/false
_krb_kerberos:*:233:-2:Open Directory
Kerberos:/var/empty:/usr/bin/false
_krb_anonymous:*:234:-2:Open Directory Kerberos
Anonymous:/var/empty:/usr/bin/false
_assetcache:*:235:235:Asset Cache
Service:/var/empty:/usr/bin/false
_coremediaiod:*:236:236:Core Media IO
Daemon:/var/empty:/usr/bin/false
_launchservicesd:*:239:239:_launchservicesd:/var/empty:/usr/bin
/false
_iconservices:*:240:240:IconServices:/var/empty:/usr/bin/false
_distnote:*:241:241:DistNote:/var/empty:/usr/bin/false
_nsurlsessiond:*:242:242:NSURLSession
Daemon:/var/db/nsurlsessiond:/usr/bin/false
_displaypolicyd:*:244:244:Display Policy
Daemon:/var/empty:/usr/bin/false
_astris:*:245:245:Astris Services:/var/db/astris:/usr/bin/false
_krbfast:*:246:-2:Kerberos FAST
Account:/var/empty:/usr/bin/false
_gamecontrollerd:*:247:247:Game Controller
Daemon:/var/empty:/usr/bin/false
_mbsetupuser:*:248:248:Setup User:/var/setup:/bin/bash
_ondemand:*:249:249:On Demand Resource
Daemon:/var/db/ondemand:/usr/bin/false
```

```
_xserverdocs:*:251:251:macOS Server Documents
Service:/var/empty:/usr/bin/false
_wwwproxy:*:252:252:WWW Proxy:/var/empty:/usr/bin/false
_mobileasset:*:253:253:MobileAsset User:/var/ma:/usr/bin/false
_findmydevice:*:254:254:Find My Device
Daemon:/var/db/findmydevice:/usr/bin/false
_datadetectors:*:257:257:DataDetectors:/var/db/datadetectors:/u
sr/bin/false
_captiveagent:*:258:258:captiveagent:/var/empty:/usr/bin/false
_ctkd:*:259:259:ctkd Account:/var/empty:/usr/bin/false
_applepay:*:260:260:applepay
Account:/var/db/applepay:/usr/bin/false
_hidd:*:261:261:HID Service User:/var/db/hidd:/usr/bin/false
_cmiodalassistants:*:262:262:CoreMedia IO Assistants
User:/var/db/cmiodalassistants:/usr/bin/false
_analyticsd:*:263:263:Analytics
Daemon:/var/db/analyticsd:/usr/bin/false
_fpsd:*:265:265:FPS Daemon:/var/db/fpsd:/usr/bin/false
_timed:*:266:266:Time Sync Daemon:/var/db/timed:/usr/bin/false
_nearbyd:*:268:268:Proximity and Ranging
Daemon:/var/db/nearbyd:/usr/bin/false
_reportmemoryexception:*:269:269:ReportMemoryException:/var/db/
reportmemoryexception:/usr/bin/false
_driverkit:*:270:270:DriverKit:/var/empty:/usr/bin/false
_diskimagesiod:*:271:271:DiskImages IO
Daemon:/var/db/diskimagesiod:/usr/bin/false
_logd:*:272:272:Log Daemon:/var/db/diagnostics:/usr/bin/false
_appinstalld:*:273:273:App Install
Daemon:/var/db/appinstalld:/usr/bin/false
_installcoordinationd:*:274:274:Install Coordination
Daemon:/var/db/installcoordinationd:/usr/bin/false
_demod:*:275:275:Demo Daemon:/var/empty:/usr/bin/false
_rmd:*:277:277:Remote Management
Daemon:/var/db/rmd:/usr/bin/false
_fud:*:278:278:Firmware Update
Daemon:/var/db/fud:/usr/bin/false
_knowledgegraphd:*:279:279:Knowledge Graph
Daemon:/var/db/knowledgegraphd:/usr/bin/false
_coreml:*:280:280:CoreML Services:/var/empty:/usr/bin/false
_oahd:*:441:441:OAH Daemon:/var/empty:/usr/bin/false

Events:  <none>
```

处理多个文件

```
neo@MacBook-Pro-Neo ~ % kubectl create configmap apache-httpd -
-from-file=/etc/apache2/httpd.conf --from-
file=/etc/apache2/extra/httpd-vhosts.conf
configmap/apache-httpd created
```

处理目录内的所有文件

```
neo@MacBook-Pro-Neo ~ % kubectl create configmap apache-httpd-
users --from-file=/etc/apache2/users
configmap/apache-httpd-users created
```

## 15.3. 从环境变量文件创建 ConfigMap

```
cat <<EOF > /tmp/test.env
username=neo
nickname=netkiller
age=38
sex=Y
EOF
```

```
neo@MacBook-Pro-Neo ~ % cat <<EOF > /tmp/test.env
username=neo
nickname=netkiller
age=38
sex=Y
EOF
neo@MacBook-Pro-Neo ~ % cat /tmp/test.env
```

```
username=neo
nickname=netkiller
age=38
sex=Y
neo@MacBook-Pro-Neo ~ % kubectl create configmap env-config --
from-env-file=/tmp/test.env
configmap/env-config created
```

## 15.4. 查看 ConfigMap

```
neo@MacBook-Pro-Neo ~ % kubectl get configmap
NAME                 DATA   AGE
config               1      52s
```

```
neo@MacBook-Pro-Neo ~ % kubectl describe configmap config
Name:         config
Namespace:    default
Labels:       <none>
Annotations:  <none>

Data
====
nickname:
----
netkiller
Events:  <none>
```

```
neo@MacBook-Pro-Neo ~ % kubectl get configmap config -o yaml
apiVersion: v1
data:
  nickname: netkiller
```

```
kind: ConfigMap
metadata:
  creationTimestamp: "2020-10-02T05:05:59Z"
  managedFields:
  - apiVersion: v1
    fieldsType: FieldsV1
    fieldsV1:
      f:data:
        .: {}
        f:nickname: {}
    manager: kubectl-create
    operation: Update
    time: "2020-10-02T05:05:59Z"
  name: config
  namespace: default
  resourceVersion: "18065"
  selfLink: /api/v1/namespaces/default/configmaps/config
  uid: 35381fa6-681b-417a-afc1-f45fdff5406d
```

```
neo@MacBook-Pro-Neo ~ % kubectl get configmap user -o json
{
    "apiVersion": "v1",
    "data": {
        "age": "35",
        "nickname": "netkiller",
        "username": "neo"
    },
    "kind": "ConfigMap",
    "metadata": {
        "creationTimestamp": "2020-10-02T05:13:09Z",
        "managedFields": [
            {
                "apiVersion": "v1",
                "fieldsType": "FieldsV1",
                "fieldsV1": {
                    "f:data": {
                        ".": {},
                        "f:age": {},
                        "f:nickname": {},
                        "f:username": {}
                    }
                },
                "manager": "kubectl-create",
                "operation": "Update",
                "time": "2020-10-02T05:13:09Z"
            }
        ],
        "name": "user",
        "namespace": "default",
        "resourceVersion": "18381",
        "selfLink":
"/api/v1/namespaces/default/configmaps/user",
        "uid": "51e3aa61-21cf-4ed1-871c-ac7119aec7a1"
    }
}
```

## 15.5. 删除 ConfigMap

```
neo@MacBook-Pro-Neo ~ % kubectl delete -n default configmap
config
configmap "config" deleted
```

## 15.6. ConfigMap

### Key-Value 配置

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: db-config
  namespace: default
data:
  db.host: 172.16.0.10
  db.port: '3306'
  db.user: neo
  db.pass: chen
```

创建配置

```
neo@MacBook-Pro-Neo ~/tmp/kubernetes % kubectl create -f key-
value.yaml
configmap/db-config created
```

将配置项保存到文件

```
apiVersion: v1
kind: Pod
metadata:
  name: test-pod
spec:
  containers:
    - name: test-container
      image: gcr.io/google_containers/busybox
      command: [ "/bin/sh", "-c", "cat
/usr/local/etc/config/db.host" ]
      volumeMounts:
      - name: config-volume
        mountPath: /usr/local/etc/config
  volumes:
    - name: config-volume
      configMap:
        name: db-config
  restartPolicy: Never
```

定义多组配置项

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: spring-cloud-config
  namespace: default
data:
  config: |
    spring.security.user=config
    spring.security.user=passw0rd
  euerka: |
    spring.security.user=eureka
    spring.security.user=passw0rd
  gateway: |
    spring.security.user=gateway
    spring.security.user=passw0rd
```

**Secret**

制作私钥证书

```
openssl genrsa -out ingress.key 2048
```

制作公钥证书

```
openssl req -new -x509 -days 3650 -key ingress.key -out
ingress.crt
```

生成 BASE64

```
neo@MacBook-Pro-Neo ~/workspace/devops/demo % base64
ingress.crt
LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURhRENDQWxBQ0NRRFdsVG0
x……
neo@MacBook-Pro-Neo ~/workspace/devops/demo % base64
ingress.key
LS0tLS1CRUdJTiBSU0EgUFJJVkFURSBLRVktLS0tLQpNSUlFb3dJQkFBS0NBUUV
B……
```

```
apiVersion: v1
kind: Secret
metadata:
  name: tls
  namespace: development
data:
  tls.crt:
LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURhRENDQWxBQ0NRRFdsV
G0x……
  tls.key:
LS0tLS1CRUdJTiBSU0EgUFJJVkFURSBLRVktLS0tLQpNSUlFb3dJQkFBS0NBU
UVB……
```

## 环境变量

envFrom 可将 ConfigMap 中的配置项定义为容器环境变量

```
apiVersion: v1
kind: Pod
metadata:
  name: neo-test-pod
spec:
  containers:
    - name: test-container
      image: k8s.gcr.io/busybox
      command: [ "/bin/sh", "-c", "env" ]
      envFrom:
      - configMapRef:
          name: special-config
  restartPolicy: Never
```

引用单个配置项使用 valueFrom

```
neo@MacBook-Pro-Neo ~/tmp/kubernetes % cat key-value.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: db-config
  namespace: default
data:
  db.host: 172.16.0.10
  db.port: '3306'
  db.user: neo
  db.pass: chen
---
apiVersion: v1
kind: Pod
metadata:
  name: test-pod
spec:
  containers:
    - name: test-container
      image: busybox
      command: [ "/bin/sh", "-c", "env" ]
      env:
        - name: DBHOST
          valueFrom:
            configMapKeyRef:
              name: db-config
              key: db.host
        - name: DBPORT
          valueFrom:
            configMapKeyRef:
              name: db-config
              key: db.port
  restartPolicy: Never

neo@MacBook-Pro-Neo ~/tmp/kubernetes % kubectl create -f key-
value.yaml
configmap/db-config created
pod/test-pod created
```

## 配置文件

### 定义配置

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: redis-config
  labels:
    app: redis
data:
  redis.conf: |-
    pidfile /var/lib/redis/redis.pid
    dir /var/lib/redis
    port 6379
    bind 0.0.0.0
    appendonly yes
    protected-mode no
    requirepass 123456
```

### 引用配置

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: redis
  labels:
    app: redis
spec:
  replicas: 1
  selector:
    matchLabels:
      app: redis
  template:
    metadata:
      labels:
        app: redis
    spec:
      containers:
      - name: redis
        image: redis:5.0.8
        command:
          - "sh"
          - "-c"
          - "redis-server /usr/local/etc/redis/redis.conf"
        ports:
        - containerPort: 6379
        resources:
          limits:
            cpu: 1000m
            memory: 1024Mi
          requests:
            cpu: 1000m
            memory: 1024Mi
        livenessProbe:
          tcpSocket:
            port: 6379
          initialDelaySeconds: 300
          timeoutSeconds: 1
          periodSeconds: 10
          successThreshold: 1
          failureThreshold: 3
        readinessProbe:
          tcpSocket:
            port: 6379
          initialDelaySeconds: 5
          timeoutSeconds: 1
          periodSeconds: 10
          successThreshold: 1
          failureThreshold: 3
        volumeMounts:
```

```
        - name: data
          mountPath: /data
        - name: config
          mountPath:  /usr/local/etc/redis/redis.conf
          subPath: redis.conf
      volumes:
      - name: data
        persistentVolumeClaim:
          claimName: redis
      - name: config
        configMap:
          name: redis-config
```

```
apiVersion: v1
kind: Pod
metadata:
  name: test-pod
spec:
  containers:
    - name: test-container
      image: gcr.io/google_containers/busybox
      command: [ "/bin/sh","-c","find /etc/config/" ]
      volumeMounts:
      - name: config-volume
        mountPath: /etc/config
  volumes:
    - name: config-volume
      configMap:
        name: special-config
        items:
        - key: special.how
          path: path/to/special-key
  restartPolicy: Never
```

# 16. Job/CronJob

## 16.1. CronJob

```
kubectl run hello --schedule="*/1 * * * *" --restart=OnFailure
--image=busybox -- /bin/sh -c "date; echo Hello from the
Kubernetes cluster"

kubectl delete cronjob hello
```

## 16.2. Job

执行单词任务

.spec.completions 标志Job结束需要成功运行的Pod个数，默认为1

.spec.parallelism 标志并行运行的Pod的个数，默认为1

.spec.activeDeadlineSeconds 标志失败Pod的重试最大时间，超过这个时间不会继续重试

```
apiVersion: batch/v1
kind: Job
metadata:
  name: busybox
spec:
  completions: 1
  parallelism: 1
  template:
    metadata:
      name: busybox
    spec:
      containers:
```

```
      - name: busybox
        image: busybox
        command: ["echo", "hello"]
      restartPolicy: Never
```

```
$ kubectl create -f job.yaml
job "busybox" created
$ pods=$(kubectl get pods --selector=job-name=busybox --
output=jsonpath={.items..metadata.name})
$ kubectl logs $pods
```

## 计划任务

.spec.schedule 指定任务运行周期，格式同Cron

.spec.startingDeadlineSeconds 指定任务开始的截止期限

.spec.concurrencyPolicy 指定任务的并发策略，支持Allow、Forbid和Replace三个选项

```
apiVersion: batch/v2alpha1
kind: CronJob
metadata:
  name: hello
spec:
  schedule: "*/1 * * * *"
  jobTemplate:
    spec:
      template:
        spec:
          containers:
          - name: hello
            image: busybox
            args:
```

```
          - /bin/sh
          - -c
          - date; echo Hello from the Kubernetes cluster
        restartPolicy: OnFailure
```

# 17. explain

## 17.1. ingress

```
iMac:kubernetes neo$ kubectl explain ingress
KIND:     Ingress
VERSION:  extensions/v1beta1

DESCRIPTION:
     Ingress is a collection of rules that allow inbound
connections to reach
     the endpoints defined by a backend. An Ingress can be
configured to give
     services externally-reachable urls, load balance traffic,
terminate SSL,
     offer name based virtual hosting etc. DEPRECATED - This
group version of
     Ingress is deprecated by networking.k8s.io/v1beta1
Ingress. See the release
     notes for more information.

FIELDS:
   apiVersion   <string>
     APIVersion defines the versioned schema of this
representation of an
     object. Servers should convert recognized schemas to the
latest internal
     value, and may reject unrecognized values. More info:
     https://git.k8s.io/community/contributors/devel/sig-
architecture/api-conventions.md#resources

   kind <string>
     Kind is a string value representing the REST resource this
object
     represents. Servers may infer this from the endpoint the
client submits
     requests to. Cannot be updated. In CamelCase. More info:
     https://git.k8s.io/community/contributors/devel/sig-
architecture/api-conventions.md#types-kinds
```

```
   metadata     <Object>
     Standard object's metadata. More info:
     https://git.k8s.io/community/contributors/devel/sig-
architecture/api-conventions.md#metadata

   spec <Object>
     Spec is the desired state of the Ingress. More info:
     https://git.k8s.io/community/contributors/devel/sig-
architecture/api-conventions.md#spec-and-status

   status       <Object>
     Status is the current state of the Ingress. More info:
     https://git.k8s.io/community/contributors/devel/sig-
architecture/api-conventions.md#spec-and-status
```

查看 ingress.spec 配置清单

```
iMac:kubernetes neo$ kubectl explain ingress.spec
KIND:     Ingress
VERSION:  extensions/v1beta1

RESOURCE: spec <Object>

DESCRIPTION:
     Spec is the desired state of the Ingress. More info:
     https://git.k8s.io/community/contributors/devel/sig-
architecture/api-conventions.md#spec-and-status

     IngressSpec describes the Ingress the user wishes to
exist.

FIELDS:
   backend      <Object>
     A default backend capable of servicing requests that don't
match any rule.
     At least one of 'backend' or 'rules' must be specified.
This field is
     optional to allow the loadbalancer controller or
defaulting logic to
     specify a global default.
   ingressClassName     <string>
     IngressClassName is the name of the IngressClass cluster
resource. The
     associated IngressClass defines which controller will
implement the
     resource. This replaces the deprecated
`kubernetes.io/ingress.class`
     annotation. For backwards compatibility, when that
annotation is set, it
     must be given precedence over this field. The controller
may emit a warning
     if the field and annotation have different values.
Implementations of this
     API should ignore Ingresses without a class specified. An
IngressClass
     resource may be marked as default, which can be used to
set a default value
     for this field. For more information, refer to the
IngressClass
     documentation.

   rules        <[]Object>
     A list of host rules used to configure the Ingress. If
unspecified, or no
     rule matches, all traffic is sent to the default backend.

   tls  <[]Object>
     TLS configuration. Currently the Ingress only supports a
single TLS port,
     443. If multiple members of this list specify different
hosts, they will be
     multiplexed on the same port according to the hostname
specified through
     the SNI TLS extension, if the ingress controller
fulfilling the ingress
     supports SNI.
```

# 18. 操作系统资源配置

## 18.1. sysctls

```
kubelet --experimental-allowed-unsafe-sysctls
'kernel.msg*,kernel.shmmax,kernel.sem,net.ipv4.route.min_pmtu'
```

## 19. 端口转发

将 本地 0.0.0.0:27017 端口转发到 service 端口

```
neo@Netkiller-iMac ~> kubectl port-forward --address 0.0.0.0
service/mongo 27017
Forwarding from 0.0.0.0:27017 -> 27017
```

## 20. 更新镜像

更新资源对象的容器镜像

可使用资源对象包括（不区分大小写）： pod (po)、replicationcontroller (rc)、deployment (deploy)、daemonset (ds)、job、replicaset (rs)

```
kubectl set image deployment/nginx nginx=nginx:1.20.0
kubectl set image deployment/nginx busybox=busybox
nginx=nginx:1.10.1
```

携带参数

```
kubectl set image deployments,rc nginx=nginx:1.9.1 --all
```

使用通配符

```
kubectl set image daemonset abc *=nginx:1.9.1
```

## 21. 复制文件

```
kubectl cp netkiller/job-executor-77fc6b4db-
5dzxz:logs/info.2022-07-29.log Downloads/info.2022-07-29.log -c
job-executor
```

```
kubectl cp Downloads/myfile netkiller/job-executor-77fc6b4db-
5dzxz:/tmp/myfile -c job-executor
```

# 22. describe

## 22.1. storageclasses.storage.k8s.io

```
[root@master ~]# kubectl describe storageclasses.storage.k8s.io
Name:                  longhorn-storage
IsDefaultClass:        No
Annotations:           <none>
Provisioner:           driver.longhorn.io
Parameters:
diskSelector=hdd,numberOfReplicas=2,staleReplicaTimeout=2880
AllowVolumeExpansion:  True
MountOptions:          <none>
ReclaimPolicy:         Delete
VolumeBindingMode:     Immediate
Events:                <none>


Name:            longhorn
IsDefaultClass:  No
Annotations:     longhorn.io/last-applied-configmap=kind:
StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: longhorn
  annotations:
    storageclass.kubernetes.io/is-default-class: "true"
provisioner: driver.longhorn.io
allowVolumeExpansion: true
reclaimPolicy: "Delete"
volumeBindingMode: Immediate
parameters:
  numberOfReplicas: "3"
  staleReplicaTimeout: "30"
  fromBackup: ""
  fsType: "ext4"
  dataLocality: "disabled"
,storageclass.beta.kubernetes.io/is-default-
class=false,storageclass.kubernetes.io/is-default-class=false
Provisioner:           driver.longhorn.io
```

```
Parameters:
dataLocality=disabled,fromBackup=,fsType=ext4,numberOfReplicas=
3,staleReplicaTimeout=30
AllowVolumeExpansion:  True
MountOptions:          <none>
ReclaimPolicy:         Delete
VolumeBindingMode:     Immediate
Events:                <none>


Name:                  local-path
IsDefaultClass:        Yes
Annotations:
objectset.rio.cattle.io/applied=H4sIAAAAAAAA/4yRT+vUMBCGv4rMua1
bu1tKwIOu7EUEQdDzNJlux6aZkkwry7LfXbIqrIffn2PyZN7hfXIFXPg7xcQSwE
BSiXimaupSxfJ2q6GAiYMDA9/+oKPHlKCAmRQdKoK5AoYgisoSUj5K/5OsJtIqs
lQWVT3lNM4xUDzJ5VegWJ63CQxMTXogW128+czBvf/gnIQXIwLOBAa8WPTl30qv
GkoL2jw5rT2V6ZKUZij+SbG5eZVRDKR0F8SpdDTg6rW8YzCgcSW4FeCxJ/+sjxH
TCAbqrhmag20Pw9DbZtfu210z7JuhPnQ719m2w3cOe7fPof81W1DHfLlE2Th/IE
UwEDHYkWJe8PCsgJgL8PxVPNsLGPhEnjRr2cSvM33k4Dicv4jLC34g60niiWPSo
4S0zhTh9jsAAP//ytgh5S0CAAA,objectset.rio.cattle.io/id=,objectse
t.rio.cattle.io/owner-gvk=k3s.cattle.io/v1,
Kind=Addon,objectset.rio.cattle.io/owner-name=local-
storage,objectset.rio.cattle.io/owner-namespace=kube-
system,storageclass.beta.kubernetes.io/is-default-
class=true,storageclass.kubernetes.io/is-default-class=true
Provisioner:           rancher.io/local-path
Parameters:            <none>
AllowVolumeExpansion:  <unset>
MountOptions:          <none>
ReclaimPolicy:         Delete
VolumeBindingMode:     WaitForFirstConsumer
Events:                <none>
```

## 22.2. pod

```
[root@master ~]# kubectl describe pvc
Name:          elasticsearch-elasticsearch-data-0
Namespace:     default
StorageClass:  local-path
```

```
Status:         Bound
Volume:         pvc-a2ebce5a-9ae1-46e9-ae9f-8840027bf5d8
Labels:         app=elasticsearch
                role=data
Annotations:    pv.kubernetes.io/bind-completed: yes
                pv.kubernetes.io/bound-by-controller: yes
                volume.beta.kubernetes.io/storage-provisioner:
rancher.io/local-path
                volume.kubernetes.io/selected-node: agent-1
                volume.kubernetes.io/storage-provisioner:
rancher.io/local-path
Finalizers:     [kubernetes.io/pvc-protection]
Capacity:       1Gi
Access Modes:   RWO
VolumeMode:     Filesystem
Used By:        elasticsearch-data-0
Events:         <none>


Name:           elasticsearch-elasticsearch-data-1
Namespace:      default
StorageClass:   local-path
Status:         Bound
Volume:         pvc-f0d9d5df-9704-44a7-93ff-8a4f431af226
Labels:         app=elasticsearch
                role=data
Annotations:    pv.kubernetes.io/bind-completed: yes
                pv.kubernetes.io/bound-by-controller: yes
                volume.beta.kubernetes.io/storage-provisioner:
rancher.io/local-path
                volume.kubernetes.io/selected-node: master
                volume.kubernetes.io/storage-provisioner:
rancher.io/local-path
Finalizers:     [kubernetes.io/pvc-protection]
Capacity:       1Gi
Access Modes:   RWO
VolumeMode:     Filesystem
Used By:        elasticsearch-data-1
Events:         <none>


Name:           elasticsearch-elasticsearch-data-2
Namespace:      default
StorageClass:   local-path
Status:         Bound
```

```
Volume:         pvc-722cce94-b2c5-457a-8e01-9a2a52b12128
Labels:         app=elasticsearch
                role=data
Annotations:    pv.kubernetes.io/bind-completed: yes
                pv.kubernetes.io/bound-by-controller: yes
                volume.beta.kubernetes.io/storage-provisioner:
rancher.io/local-path
                volume.kubernetes.io/selected-node: agent-1
                volume.kubernetes.io/storage-provisioner:
rancher.io/local-path
Finalizers:     [kubernetes.io/pvc-protection]
Capacity:       1Gi
Access Modes:   RWO
VolumeMode:     Filesystem
Used By:        elasticsearch-data-2
Events:         <none>


Name:           longhorn-volv-pvc
Namespace:      default
StorageClass:   longhorn
Status:         Bound
Volume:         pvc-5dc3ae33-9f86-4650-82ba-a7b681963adc
Labels:         <none>
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
Finalizers:     [kubernetes.io/pvc-protection]
Capacity:       2Gi
Access Modes:   RWO
VolumeMode:     Filesystem
Used By:        volume-test
Events:         <none>


Name:           redis
Namespace:      default
StorageClass:   local-path
Status:         Pending
Volume:
Labels:         <none>
Annotations:    <none>
```

```
Finalizers:    [kubernetes.io/pvc-protection]
Capacity:
Access Modes:
VolumeMode:    Filesystem
Used By:       redis-0
Events:
  Type    Reason                Age                     From
Message
  ----    ------                ----                    ----
-------
  Normal  WaitForFirstConsumer  29s (x481 over 120m)
persistentvolume-controller  waiting for first consumer to be
created before binding
[root@master ~]#
```

## 23. clusterrolebinding

```
kubectl create clusterrolebinding cluster-admin-binding --
clusterrole cluster-admin --user [USER ACCOUNT]
```

# 24. Volume

```
PersistentVolume 的访问模式（accessModes）有三种：

ReadWriteOnce（RWO）：是最基本的方式，可读可写，但只支持被单个节点挂载。
ReadOnlyMany（ROX）：可以以只读的方式被多个节点挂载。
ReadWriteMany（RWX）：这种存储可以以读写的方式被多个节点共享。不是每一种
存储都支持这三种方式，像共享方式，目前支持的还比较少，比较常用的是 NFS。在
PVC 绑定 PV 时通常根据两个条件来绑定，一个是存储的大小，另一个就是访问模
式。


PersistentVolume 的回收策略（persistentVolumeReclaimPolicy，即 PVC
释放卷的时候 PV 该如何操作）也有三种

Retain，不清理，保留 Volume（需要手动清理）
Recycle，删除数据，即 rm -rf /thevolume/*（只有 NFS 和 HostPath 支
持）
Delete，删除存储资源，比如删除 AWS EBS 卷（只有 AWS EBS, GCE PD,
Azure Disk 和 Cinder 支持）
```

## 24.1. local

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: example-pv
spec:
  capacity:
    storage: 100Gi
  # volumeMode field requires BlockVolume Alpha feature gate to
be enabled.
  volumeMode: Filesystem
  accessModes:
  - ReadWriteOnce
  persistentVolumeReclaimPolicy: Delete
  storageClassName: local-storage
  local:
    path: /mnt/disks/ssd1
  nodeAffinity:
    required:
      nodeSelectorTerms:
      - matchExpressions:
        - key: kubernetes.io/hostname
          operator: In
          values:
          - example-node
```

案例

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: local-volume
provisioner: kubernetes.io/no-provisioner
volumeBindingMode: WaitForFirstConsumer
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: netkiller-local-pv
spec:
  capacity:
    storage: 1Gi
  accessModes:
  - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  storageClassName: local-volume
  local:
    path: /tmp/neo
  nodeAffinity:
    required:
      nodeSelectorTerms:
      - matchExpressions:
```

部署 POD

```
        - key: kubernetes.io/hostname
          operator: In
          values:
          - minikube
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: netkiller-pvc
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: local-volume
---
kind: Pod
apiVersion: v1
metadata:
  name: busybox
  namespace: default
spec:
  containers:
    - name: busybox
      image: busybox:latest
      # image:
registry.netkiller.cn:5000/netkiller/welcome:latest
      imagePullPolicy: IfNotPresent
      command:
        - sleep
        - "3600"
      volumeMounts:
      - mountPath: "/srv"
        name: mypd
  restartPolicy: Always
  volumes:
    - name: mypd
      persistentVolumeClaim:
        claimName: netkiller-pvc
```

部署 POD

```
iMac:kubernetes neo$ kubectl create -f
example/volume/local.yaml
storageclass.storage.k8s.io/local-volume created
persistentvolume/netkiller-local-pv created
persistentvolumeclaim/netkiller-pvc created
pod/busybox created
```

查看POD状态

```
iMac:kubernetes neo$ kubectl get pod
NAME      READY   STATUS    RESTARTS   AGE
busybox   1/1     Running   0          2m28s
```

进入POD查看local卷的挂载情况，同时创建一个测试文件。

```
iMac:kubernetes neo$ kubectl exec -it busybox sh
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed
in a future version. Use kubectl exec [POD] -- [COMMAND]
instead.
/ # mount | grep /srv
tmpfs on /srv type tmpfs (rw)

/ # echo helloworld > /srv/netkiller
/ # cat /srv/netkiller
helloworld
```

进入宿主主机查看挂载目录

```
$ cat /tmp/neo/netkiller
```

```
helloworld
```

# 25. Ingress

正常情况 Service 只是暴露了端口，这个端口是可以对外访问的，但是80端口只有一个，很多 Service 都要使用 80端口，这时就需要使用虚拟主机技术。

多个 Service 共同使用一个 80 端口，通过域名区分业务。这就是 Ingress 存在的意义。

## 25.1. 端口

```
+----------+  Ingress   +---------+    Pod    +----------+
| internet | ---------> | Service | --------> | Pod Node |
+----------+            +---------+           +----------+
```

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: springboot
spec:
  backend:
    service:
      name: springboot
      port:
        number: 80
```

## 25.2. URI 规则

```
                  Ingress        / ---> /api --> api-
service:8080
www.netkiller.cn ----------> |  ---> /usr --> usr-service:8080
                             \ ---> /img --> img-service:8080
```

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: uri-ingress
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
  - host: www.netkiller.cn
    http:
      paths:
      - path: /api
        backend:
          serviceName: api-service
          servicePort: 8080
      - path: /usr
        backend:
          serviceName: usr-service
          servicePort: 8080
      - path: /img
        backend:
          serviceName: img-service
          servicePort: 8080
```

## 25.3. vhost 虚拟主机

```
www.netkiller.cn --|     Ingress      |-> www.netkiller.cn
www:80
                   | --------------> |
img.netkiller.cn --|                  |-> img.netkiller.cn
img:80
```

```
```

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: vhost-ingress
spec:
  rules:
  - host: www.netkiller.cn
    http:
      paths:
      - backend:
          serviceName: www
          servicePort: 80
  - host: img.netkiller.cn
    http:
      paths:
      - backend:
          serviceName: img
          servicePort: 80
```

## 25.4. rewrite

```
http://www.netkiller.cn/1100 => /article/1100
```

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: rewrite-ingress
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /article/$1
spec:
  rules:
```

```
  - host: www.netkiller.cn
    http:
      paths:
        # 可以有多个（可以正则）
        - path: /($/.*)
          backend:
            serviceName: article
            servicePort: 80
```

## 25.5. annotations 配置

### HTTP 跳转到 HTTPS

```
# 该注解只在配置了HTTPS之后才会生效进行跳转
nginx.ingress.kubernetes.io/ssl-redirect: "true"

# 强制跳转到https，不论是否配置了https证书
nginx.ingress.kubernetes.io/force-ssl-redirect: "true"
```

### server-snippet

server-snippet 可以让你直接编排 Nginx 配置

```
nginx.ingress.kubernetes.io/server-snippet: |
    rewrite /api/($|.*) /api/v2/$1 break;
    rewrite /img/($|.*) /img/thumbnail/$1 break;
```

## 25.6. 金丝雀发布（灰度发布）

三种annotation按匹配优先级顺序：

```
canary-by-header > canary-by-cookie > canary-weight
```

准备服务

```
# Release Version
apiVersion: v1
kind: Service
metadata:
    name: hello-service
    labels:
    app: hello-service
spec:
ports:
- port: 80
    protocol: TCP
selector:
    app: hello-service
---
# canary Version
apiVersion: v1
kind: Service
metadata:
    name: canary-hello-service
    labels:
    app: canary-hello-service
spec:
ports:
- port: 80
    protocol: TCP
selector:
    app: canary-hello-service
```

方案一，权重分配

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: canary
  annotations:
    kubernetes.io/ingress.class: nginx
    nginx.ingress.kubernetes.io/canary: "true"
    nginx.ingress.kubernetes.io/canary-weight: "30"
spec:
  rules:
  - host: canary.netkiller.cn
    http:
      paths:
      - backend:
          serviceName: canary-hello-service
```

```
$ for i in $(seq 1 10); do curl http://canary.netkiller.cn;
echo '\n'; done
```

### 通过HTTP头开启灰度发布

```
annotations:
  kubernetes.io/ingress.class: nginx
  nginx.ingress.kubernetes.io/canary: "true"
  nginx.ingress.kubernetes.io/canary-by-header: "canary"
```

```
$ for i in $(seq 1 5); do curl -H 'canary:always'
http://canary.netkiller.cn; echo '\n'; done
```

```
annotations:
  kubernetes.io/ingress.class: nginx
  nginx.ingress.kubernetes.io/canary: "true"
  nginx.ingress.kubernetes.io/canary-by-header: "canary"
  nginx.ingress.kubernetes.io/canary-by-header-value: "true"
```

```
$ for i in $(seq 1 5); do curl -H 'canary:true'
http://canary.netkiller.cn; echo '\n'; done
```

### 通过 Cookie 开启

```
annotations:
  kubernetes.io/ingress.class: nginx
  nginx.ingress.kubernetes.io/canary: "true"
  nginx.ingress.kubernetes.io/canary-by-cookie: "canary"
```

```
$ for i in $(seq 1 5); do curl -b 'canary=always'
http://canary.netkiller.cn; echo '\n'; done
```

## 25.7. 管理 Ingress

```
# 查看已有配置
kubectl describe ingress test

# 修改配置
kubectl edit ingress test

# 来重新载入配置
kubectl replace -f ingress.yaml
```

# 第 6 章 kubectl example

## 1. 私有 registry

```
kubectl create deployment registry --image=registry:latest
kubectl expose deployment registry --port=5000 --target-
port=5000
kubectl delete -n default deployment registry
```

```
iMac:registry neo$ docker pull nginx:latest
iMac:registry neo$ docker tag nginx:latest
192.168.64.2:30050/nginx:latest
iMac:registry neo$ docker push 192.168.64.2:30050/nginx:latest



kubectl create deployment nginx --
image=192.168.64.2:30050/nginx:latest
kubectl expose deployment nginx --port=80 --target-port=30080 -
-type=NodePort

kubectl create deployment busybox --image=docker.io/busybox
kubectl create deployment busybox --image=busybox
kubectl create deployment welcome --
image=127.0.0.1:5000/netkiller/welcome

docker tag busybox:latest 192.168.64.6:32070/busybox:latest
docker push 192.168.64.6:32070/busybox:latest
```

## 2. mongodb

```
kubectl run mongodb  --image=docker.io/mongo  --
env="p='27017:27017'" --env="v='/opt/mongodb:/data'"
kubectl expose deployment mongodb --port=27017 --target-
port=27017
```

## 3. tomcat

```
kubectl create deployment hello-minikube --image=tomcat:8.0
kubectl expose deployment hello-minikube --type=NodePort --
port=80
minikube service hello-minikube --url
```

# 第 7 章 istio

## 1. 启动 istio

下面的例子是在 default 命名空间启用 istio。

```
$ kubectl label namespace default istio-injection=enabled
namespace/default labeled
```

## 2. 禁用 istio

如果在该namespace下创建pod，不想要使用istio-proxy，可以在创建的pod中annotations 配置项声明禁用 istio

```
apiVersion: v1
kind: Pod
metadata:
  annotations:
    sidecar.istio.io/inject: "false"
```

# 第 8 章 Kubeapps

Kubeapps is a web-based UI for deploying and managing applications in Kubernetes clusters

https://kubeapps.com

# 第 9 章 Helm - The package manager for Kubernetes

https://helm.sh

## 1. 安装 Helm

### 1.1. AlmaLinux

CURL 安装

```
curl
https://raw.githubusercontent.com/helm/helm/main/scripts/get-
helm-3 | bash
```

二进制安装

```
cd /usr/local/src/
wget https://get.helm.sh/helm-v3.9.4-linux-amd64.tar.gz
tar zxvf helm-v3.9.4-linux-amd64.tar.gz
mv linux-amd64 /srv/helm-v3.9.4
alternatives --install /usr/local/bin/helm helm /srv/helm-
v3.9.4/helm 10
```

### 1.2. Rocky Linux

```
[root@netkiller ~]# dnf install -y snapd
[root@netkiller ~]# ln -s /var/lib/snapd/snap /snap
```

```
[root@netkiller ~]# systemctl enable --now snapd.socket
[root@netkiller ~]# systemctl start --now snapd.socket
[root@netkiller ~]# snap install helm --classic
```

```
cat >> /etc/profile.d/snap.sh <<EOF
export PATH=$PATH:/snap/bin
EOF
source /etc/profile.d/snap.sh
```

### 1.3. Ubuntu

```
snap install helm --classic
```

### 1.4. Mac

homebrew 安装 Helm

```
iMac:~ neo$ brew install helm

iMac:~ neo$ helm version
version.BuildInfo{Version:"v3.3.3",
GitCommit:"55e3ca022e40fe200fbc855938995f40b2a68ce0",
GitTreeState:"dirty", GoVersion:"go1.15.2"}
```

旧版本

```
brew install kubernetes-helm
```

## 2. 快速开始

```
# 初始化本地，并将 Tiller 安装到 Kubernetes cluster
$ helm init

# 更新本地 charts repo
$ helm repo update

# 安装 mysql chart
$ helm install --name my-mysql stable/mysql

# 删除 mysql
$ helm delete my-mysql

# 删除 mysql 并释放该名字以便后续使用
$ helm delete --purge my-mysql
```

# 3. Helm 命令

## 3.1. 初始化 Helm

```
neo@MacBook-Pro ~ % helm init
Creating /Users/neo/.helm
Creating /Users/neo/.helm/repository
Creating /Users/neo/.helm/repository/cache
Creating /Users/neo/.helm/repository/local
Creating /Users/neo/.helm/plugins
Creating /Users/neo/.helm/starters
Creating /Users/neo/.helm/cache/archive
Creating /Users/neo/.helm/repository/repositories.yaml
Adding stable repo with URL: https://kubernetes-
charts.storage.googleapis.com
Adding local repo with URL: http://127.0.0.1:8879/charts
$HELM_HOME has been configured at /Users/neo/.helm.
Warning: Tiller is already installed in the cluster.
(Use --client-only to suppress this message, or --upgrade to
upgrade Tiller to the current version.)
Happy Helming!
```

## 3.2. 查看仓库列表

查看当前的 Charts 包仓库

```
neo@MacBook-Pro ~ % helm repo list
NAME    URL
stable  https://kubernetes-charts.storage.googleapis.com
local   http://127.0.0.1:8879/charts
```

更新仓库

```
neo@MacBook-Pro ~ % helm repo update
Hang tight while we grab the latest from your chart
repositories...
...Skip local chart repository
...Unable to get an update from the "stable" chart repository
(https://kubernetes-charts.storage.googleapis.com):
        unexpected EOF
Update Complete. ⎈ Happy Helming!⎈
```

### 3.3. 搜索

在stable仓库搜索 redis应用

```
neo@MacBook-Pro ~ % helm search stable/redis
NAME             CHART VERSION   APP VERSION     DESCRIPTION
stable/redis     6.4.3           4.0.14          Open source,
advanced key-value store. It is often referr...
stable/redis-ha 3.3.3            5.0.3           Highly
available Kubernetes implementation of Redis
```

### 3.4. 查看包信息

查看包详细信息与帮助手册

```
neo@MacBook-Pro ~ % helm inspect stable/redis
```

### 3.5. 安装

```
$ helm install stable/redis
$ helm install --name=redis stable/redis
```

```
neo@MacBook-Pro ~ % helm install stable/redis
NAME:   vested-termite
LAST DEPLOYED: Sun Mar 31 17:46:02 2019
NAMESPACE: default
STATUS: DEPLOYED

RESOURCES:
==> v1/ConfigMap
NAME                         DATA  AGE
vested-termite-redis         3     0s
vested-termite-redis-health  3     0s

==> v1/Pod(related)
NAME                                          READY  STATUS
RESTARTS  AGE
vested-termite-redis-master-0                 0/1    Pending
0         0s
vested-termite-redis-slave-57584f877-8njkc  0/1
ContainerCreating  0         0s

==> v1/Secret
NAME                  TYPE    DATA  AGE
vested-termite-redis  Opaque  1     0s

==> v1/Service
NAME                         TYPE       CLUSTER-IP
EXTERNAL-IP  PORT(S)   AGE
vested-termite-redis-master  ClusterIP  10.98.194.187  <none>
6379/TCP  0s
vested-termite-redis-slave   ClusterIP  10.111.85.208  <none>
6379/TCP  0s

==> v1beta1/Deployment
NAME                        READY  UP-TO-DATE  AVAILABLE  AGE
vested-termite-redis-slave  0/1    1           0          0s
```

```
==> v1beta2/StatefulSet
NAME                         READY  AGE
vested-termite-redis-master  0/1    0s


NOTES:
** Please be patient while the chart is being deployed **
Redis can be accessed via port 6379 on the following DNS names
from within your cluster:

vested-termite-redis-master.default.svc.cluster.local for
read/write operations
vested-termite-redis-slave.default.svc.cluster.local for read-
only operations


To get your password run:

    export REDIS_PASSWORD=$(kubectl get secret --namespace
default vested-termite-redis -o jsonpath="{.data.redis-
password}" | base64 --decode)

To connect to your Redis server:

1. Run a Redis pod that you can use as a client:

   kubectl run --namespace default vested-termite-redis-client
--rm --tty -i --restart='Never' \
    --env REDIS_PASSWORD=$REDIS_PASSWORD \
   --image docker.io/bitnami/redis:4.0.14 -- bash

2. Connect using the Redis CLI:
   redis-cli -h vested-termite-redis-master -a $REDIS_PASSWORD
   redis-cli -h vested-termite-redis-slave -a $REDIS_PASSWORD

To connect to your database from outside the cluster execute
the following commands:

    kubectl port-forward --namespace default svc/vested-
termite-redis 6379:6379 &
    redis-cli -h 127.0.0.1 -p 6379 -a $REDIS_PASSWORD
```

## 3.6. 列表

```
neo@MacBook-Pro ~ % helm list
NAME            REVISION        UPDATED
STATUS          CHART           APP VERSION     NAMESPACE
vested-termite  1               Sun Mar 31 17:46:02 2019
DEPLOYED        redis-6.4.3     4.0.14          default
```

## 3.7. 删除

```
helm ls --all
helm delete --purge redis
```

## 3.8. 升级

```
helm upgrade -f redis-ha-values-upgrade.yaml redis-ha
stable/redis-ha
```

## 3.9. 回滚

```
helm rollback redis-ha 1
```

## 3.10. 查看状态

```
neo@MacBook-Pro ~ % helm list
NAME            REVISION        UPDATED
STATUS          CHART           APP VERSION     NAMESPACE
vested-termite  1               Sun Mar 31 17:46:02 2019
DEPLOYED        redis-6.4.3     4.0.14          default


neo@MacBook-Pro ~ % helm status vested-termite
LAST DEPLOYED: Sun Mar 31 17:46:02 2019
NAMESPACE: default
STATUS: DEPLOYED

RESOURCES:
==> v1/ConfigMap
NAME                           DATA  AGE
vested-termite-redis           3     111m
vested-termite-redis-health    3     111m

==> v1/Pod(related)
NAME                                          READY  STATUS
RESTARTS  AGE
vested-termite-redis-master-0                 1/1    Running  0
111m
vested-termite-redis-slave-57584f877-8njkc   1/1    Running  0
111m

==> v1/Secret
NAME                  TYPE    DATA  AGE
vested-termite-redis  Opaque  1     111m

==> v1/Service
NAME                           TYPE       CLUSTER-IP
EXTERNAL-IP  PORT(S)   AGE
vested-termite-redis-master  ClusterIP  10.98.194.187  <none>
6379/TCP  111m
vested-termite-redis-slave   ClusterIP  10.111.85.208  <none>
6379/TCP  111m

==> v1beta1/Deployment
NAME                          READY  UP-TO-DATE  AVAILABLE  AGE
vested-termite-redis-slave  1/1    1           1          111m
```

```
==> v1beta2/StatefulSet
NAME                         READY  AGE
vested-termite-redis-master  1/1    111m


NOTES:
** Please be patient while the chart is being deployed **
Redis can be accessed via port 6379 on the following DNS names
from within your cluster:

vested-termite-redis-master.default.svc.cluster.local for
read/write operations
vested-termite-redis-slave.default.svc.cluster.local for read-
only operations


To get your password run:

    export REDIS_PASSWORD=$(kubectl get secret --namespace
default vested-termite-redis -o jsonpath="{.data.redis-
password}" | base64 --decode)

To connect to your Redis server:

1. Run a Redis pod that you can use as a client:

   kubectl run --namespace default vested-termite-redis-client
--rm --tty -i --restart='Never' \
    --env REDIS_PASSWORD=$REDIS_PASSWORD \
   --image docker.io/bitnami/redis:4.0.14 -- bash

2. Connect using the Redis CLI:
   redis-cli -h vested-termite-redis-master -a $REDIS_PASSWORD
   redis-cli -h vested-termite-redis-slave -a $REDIS_PASSWORD

To connect to your database from outside the cluster execute
the following commands:

    kubectl port-forward --namespace default svc/vested-
termite-redis 6379:6379 &
    redis-cli -h 127.0.0.1 -p 6379 -a $REDIS_PASSWORD
```

## 4. ingress-nginx

```
helm repo add ingress-nginx
https://kubernetes.github.io/ingress-nginx
helm repo update
```

安装 ingress-nginx 并且设置为默认 ingress

```
helm upgrade --install ingress-nginx ingress-nginx/ingress-
nginx \
--namespace ingress-nginx --set
controller.service.type=LoadBalancer \
--set controller.ingressClassResource.default=true \
--set controller.watchIngressWithoutClass=true \
--create-namespace
```

让Nginx获取客户端IP地址，找到spec下的externalTrafficPolicy，把值改为Local。

```
kubectl edit service/ingress-nginx-controller --namespace
ingress-nginx
```

## 5. elastic

```
helm repo add elastic https://helm.elastic.co
```

## 6. Helm The package manager for Kubernetes

https://helm.sh

# 7. Helm Faq

# 第 10 章 Rancher - Multi-Cluster Kubernetes Management

***Rancher is open-source software for delivering Kubernetes-as-a-Service.***

## 1. 安装 Rancher

### 1.1. Rancher Server

#### Docker 安装

如果只是学习，可以安装最新版

```
docker run -d --privileged --restart=unless-stopped -p 80:80 -p 443:443 --
name=rancher rancher/rancher:latest
```

稳定版

```
docker run -d --privileged --restart=unless-stopped -p 80:80 -p 443:443 -v
/var/lib/rancher/:/var/lib/rancher/ --name=rancher rancher/rancher:stable
```

审计日志

```
docker run -d --restart=unless-stopped -p 80:80 -p 443:443 -v
/var/lib/rancher/:/var/lib/rancher/ -v /var/log/auditlog:/var/log/auditlog --
name=rancher rancher/rancher:stable
```

##### 防火墙配置

防火墙放行 etcd

```
iptables -I INPUT -s 172.16.0.0/0 -p tcp --dport 2379 -j ACCEPT
iptables -I INPUT -s 172.16.0.0/0 -p tcp --dport 2380 -j ACCEPT
```

```
systemctl restart firewalld
systemctl enable firewalld

iptables -A INPUT -p tcp --dport 6443 -j ACCEPT
iptables -A INPUT -p tcp --dport 2379 -j ACCEPT
iptables -A INPUT -p tcp --dport 2380 -j ACCEPT
iptables -A INPUT -p tcp --dport 10250 -j ACCEPT

firewall-cmd --zone=public --add-port=6443/tcp --permanent
firewall-cmd --zone=public --add-port=2379/tcp --permanent
firewall-cmd --zone=public --add-port=2380/tcp --permanent
firewall-cmd --zone=public --add-port=10250/tcp --permanent
firewall-cmd --reload
```

从阿里云安装

```
docker run -itd -p 80:80 -p 443:443 \
    --restart=unless-stopped \
    -e CATTLE_AGENT_IMAGE="registry.cn-hangzhou.aliyuncs.com/rancher/rancher-
agent:v2.4.2" \
    registry.cn-hangzhou.aliyuncs.com/rancher/rancher
```

仅用 unsupported-storage-drivers

```
[root@localhost ~]# docker run -d --privileged --restart=unless-stopped -p 8080:80
-p 8443:443 --name=rancher --env unsupported-storage-drivers=true
rancher/rancher:stable
[root@localhost ~]# docker run -d --privileged --restart=unless-stopped -p 8080:80
-p 8443:443 --name=rancher rancher/rancher:stable  --features=unsupported-storage-
drivers=true
```

**Helm 安装 Rancher**

安装 k3s

```
hostnamectl set-hostname master
curl -sfL https://rancher-mirror.oss-cn-beijing.aliyuncs.com/k3s/k3s-install.sh |
INSTALL_K3S_MIRROR=cn sh -
```

安装最新版

```
helm repo add rancher-latest https://releases.rancher.com/server-charts/latest
```

安装用于生产环境的稳定版

```
helm repo add rancher-stable https://releases.rancher.com/server-charts/stable
```

创建命名空间

```
kubectl create namespace cattle-system
```

安装 cert-manager

```
kubectl apply -f https://github.com/cert-manager/cert-
manager/releases/download/v1.7.1/cert-manager.crds.yaml

helm repo add jetstack https://charts.jetstack.io

helm repo update

helm install cert-manager jetstack/cert-manager \
  --namespace cert-manager \
  --create-namespace \
  --version v1.7.1
```

```
helm install rancher rancher-stable/rancher \
  --create-namespace \
  --namespace cattle-system \
  --set hostname=rancher.netkiller.cn \
  --set ingress.tls.source=letsEncrypt \
  --set bootstrapPassword=admin \
  --set replicas=1 \
  --set systemDefaultRegistry=registry.cn-hangzhou.aliyuncs.com
```

**Mac 安装**

```
Neo-iMac:~ neo$ brew install rancher-cli

Neo-iMac:~ neo$ rancher -v
rancher version 2.4.13
```

进入容器

```
$ docker exec -it rancher /bin/bash
```

### Web UI

安装完之后运行下面命令查看密码

```
[root@localhost ~]# docker logs rancher 2>&1 | grep "Bootstrap Password:"
2021/11/26 10:27:14 [INFO] Bootstrap Password:
wkz68vmmx4gqfwxwzq4vxrzl5zgjqxlmxkfwkdltmpkxl5clqc9dw9
```

浏览器输入 https://your-ip-address 即可进入WebUI

设置密码

### SSL 证书

第一种方式

```
docker run -d -p 8443:443 -v /srv/rancher/cacerts.pem:/etc/rancher/ssl/cacerts.pem
-v /srv/rancher/key.pem:/etc/rancher/ssl/key.pem -v
/srv/rancher/cert.crt:/etc/rancher/ssl/cert.pem rancher/rancher:latest
```

第二种方式

```
docker run -d --name rancher-server rancher/rancher:latest
docker run -d --name=nginx --restart=unless-stopped -p 80:80 -p 443:443 -v
/your_certificates:/your_certificates -v
/etc/nginx.conf:/etc/nginx/conf.d/default.conf --link=rancher-server nginx:1.11
```

## 1.2. Rancher Kubernetes Engine (RKE) 2

### Server

```
curl -sfL https://get.rke2.io | sh -
```

```
systemctl enable rke2-server.service
systemctl start rke2-server.service
```

### Linux Agent（Worker）

```
curl -sfL https://get.rke2.io | INSTALL_RKE2_TYPE="agent" sh -
```

```
systemctl enable rke2-agent.service
```

配置 rke2-agent 服务

```
mkdir -p /etc/rancher/rke2/
vim /etc/rancher/rke2/config.yaml

server: https://<server>:9345
token: <token from server node>
```

```
systemctl start rke2-agent.service
```

## 1.3. Rancher Kubernetes Engine (RKE) 1

https://github.com/rancher/rke/releases

https://rancher.com/an-introduction-to-rke/

### 安装 RKE

#### v1.3.2

```
cd /usr/local/src/
wget https://github.com/rancher/rke/releases/download/v1.3.2/rke_linux-amd64
mkdir -p /srv/rancher/bin
install rke_linux-amd64 /srv/rancher/bin/
```

#### v0.1.17

```
[root@localhost ~]# wget
https://github.com/rancher/rke/releases/download/v0.1.17/rke
[root@localhost ~]# chmod +x rke
[root@localhost ~]# ./rke --version
rke version v0.1.17
```

### 配置 RKE

```
[root@localhost ~]# /srv/rancher/bin/rke_linux-amd64 config
[+] Cluster Level SSH Private Key Path [~/.ssh/id_rsa]:
```

### 启动 RKE

```
[root@localhost ~]# /srv/rancher/bin/rke_linux-amd64 up
```

## 1.4. Rancher CLI

### 二进制安装

http://mirror.cnrancher.com

```
cd /usr/local/src
wget http://rancher-mirror.cnrancher.com/cli/v2.4.13/rancher-linux-amd64-
v2.4.13.tar.xz
tar Jxvf rancher-linux-amd64-v2.4.13.tar.xz
install rancher-v2.4.13/rancher /usr/local/bin/
```

```
[root@localhost src]# rancher
Rancher CLI, managing containers one UTF-8 character at a time

Usage: rancher [OPTIONS] COMMAND [arg...]

Version: v2.4.13

Options:
  --debug                     Debug logging
  --config value, -c value    Path to rancher config (default: "/root/.rancher")
[$RANCHER_CONFIG_DIR]
  --help, -h                  show help
  --version, -v               print the version

Commands:
  apps, [app]                                         Operations with apps. Uses
helm. Flags prepended with "helm" can also be accurately described by helm
documentation.
  catalog                                             Operations with catalogs
  clusters, [cluster]                                 Operations on clusters
  context                                             Operations for the context
  globaldns                                           Operations on global DNS
providers and entries
  inspect                                             View details of resources
  kubectl                                             Run kubectl commands
  login, [l]                                          Login to a Rancher server
  multiclusterapps, [multiclusterapp mcapps mcapp]  Operations with multi-cluster
apps
  namespaces, [namespace]                             Operations on namespaces
  nodes, [node]                                       Operations on nodes
  projects, [project]                                 Operations on projects
  ps                                                  Show workloads in a project
  server                                              Operations for the server
  settings, [setting]                                 Show settings for the current
server
  ssh                                                 SSH into a node
  up                                                  apply compose config
  wait                                                Wait for resources cluster,
app, project, multiClusterApp
  token                                               Authenticate and generate new
kubeconfig token
  help, [h]                                           Shows a list of commands or
help for one command

Run 'rancher COMMAND --help' for more information on a command.
```

## 1.5. rancher-compose

Rancher Compose是一个多主机版本的Docker Compose

下载地址： https://github.com/rancher/rancher-compose/releases

**v0.12.5**

```
cd /tmp

wget https://github.com/rancher/rancher-compose/releases/download/v0.12.5/rancher-
compose-linux-amd64-v0.12.5.tar.xz
tar Jxvf rancher-compose-linux-amd64-v0.12.5.tar.xz
mv ./rancher-compose-v0.12.5/rancher-compose /usr/local/bin/

cd
```

# 2. 快速入门

https://www.cnrancher.com/docs/rancher/v2.x/cn/overview/quick-start-guide/

## 2.1. API

## 3. Rancher Compose

Rancher Compose 工具的工作方式是跟 Docker Compose 的工作方式是相似的，Docker Compose 不能远程部署，Rancher Compose 可以部署到指定URL的 Rancher 上。

```
[root@localhost ~]# rancher-compose
Usage: rancher-compose [OPTIONS] COMMAND [arg...]

Docker-compose to Rancher

Version: v0.12.5

Author:
  Rancher Labs, Inc.

Options:
  --verbose, --debug
  --file value, -f value           Specify one or more alternate compose files
(default: docker-compose.yml) [$COMPOSE_FILE]
  --project-name value, -p value   Specify an alternate project name (default:
directory name) [$COMPOSE_PROJECT_NAME]
  --url value                      Specify the Rancher API endpoint URL
[$RANCHER_URL]
  --access-key value               Specify Rancher API access key
[$RANCHER_ACCESS_KEY]
  --secret-key value               Specify Rancher API secret key
[$RANCHER_SECRET_KEY]
  --rancher-file value, -r value   Specify an alternate Rancher compose file
(default: rancher-compose.yml)
  --env-file value, -e value       Specify a file from which to read environment
variables
  --bindings-file value, -b value  Specify a file from which to read bindings
  --help, -h                       show help
  --version, -v                    print the version

Commands:
  create      Create all services but do not start
  up          Bring all services up
  start       Start services
  logs        Get service logs
  restart     Restart services
  stop, down  Stop services
  scale       Scale services
  rm          Delete services
  pull        Pulls images for services
  upgrade     Perform rolling upgrade between services
  help        Shows a list of commands or help for one command

Run 'rancher-compose COMMAND --help' for more information on a command.
```

## 3.1. Rancher Compose 命令

**提示**

Rancher Compose 目前不支持 V3 版的 Docker Compose

为 RANCHER COMPOSE 设置 RANCHER SERVER

```
# Set the url that Rancher is on
$ export RANCHER_URL=http://server_ip/
# Set the access key, i.e. username
$ export RANCHER_ACCESS_KEY=<username_of_environment_api_key>
# Set the secret key, i.e. password
$ export RANCHER_SECRET_KEY=<password_of_environment_api_key>
```

如果你不想设置环境变量，那么你需要在Rancher Compose 命令中手动送入这些变量:

```
$ rancher-compose --url http://server_ip --access-key
<username_of_environment_api_key> --secret-key <password_of_environment_api_key>
up
```

Rancher Compose 支持所有 Docker Compose 支持的命令

```
Name            Description
create          创建所有服务但不启动
up              启动所有服务
start           启动服务
logs            输出服务日志
restart         重启服务
stop, down      停止服务
scale           缩放服务
rm              删除服务
pull            拉取所有服务的镜像
upgrade         服务之间进行滚动升级
help, h         输出命令列表或者指定命令的帮助列表
```

RANCHER COMPOSE 选项

```
无论何时你使用 Rancher Compose 命令，这些不同的选项你都可以使用

Name    Description
```

```
--verbose, --debug
--file, -f [–file option –file option]  指定一个compose 文件 (默认: docker-compose.yml) [$COMPOSE_FILE]
--project-name, -p      指定一个项目名称 (默认: directory name)
--url                           执行 Rancher API接口 URL [$RANCHER_URL]
--access-key            指定 Rancher API access key [$RANCHER_ACCESS_KEY]
--secret-key            指定 Rancher API secret key [$RANCHER_SECRET_KEY]
--rancher-file, -r      指定一个 Rancher Compose 文件 (默认: rancher-compose.yml)
--env-file, -e          指定一个环境变量配置文件
--help, -h                      输出帮助文本
--version, -v           输出 Rancher Compose 版本
```

## 3.2. 操作演示

API

准备 docker-compose.yml 文件

```
rancher-compose --url https://rancher.netkiller.cn/v3 --access-key token-pk9n2 --secret-key p2twn42xps9nmh74qm5k5fhfn8rxqhlwv7q9hzcvbvqk5tsqwdh4tc up
```

# 4. Rancher CLI

帮助信息

```
[root@localhost ~]# rancher
Rancher CLI, managing containers one UTF-8 character at a time

Usage: rancher [OPTIONS] COMMAND [arg...]

Version: v2.4.13

Options:
  --debug                     Debug logging
  --config value, -c value   Path to rancher config (default:
"/root/.rancher") [$RANCHER_CONFIG_DIR]
  --help, -h                  show help
  --version, -v               print the version

Commands:
  apps, [app]                                             Operations
with apps. Uses helm. Flags prepended with "helm" can also be
accurately described by helm documentation.
  catalog                                                 Operations
with catalogs
  clusters, [cluster]                                     Operations
on clusters
  context                                                 Operations
for the context
  globaldns                                               Operations
on global DNS providers and entries
  inspect                                                 View
details of resources
  kubectl                                                 Run kubectl
commands
  login, [l]                                              Login to a
Rancher server
  multiclusterapps, [multiclusterapp mcapps mcapp]  Operations
with multi-cluster apps
  namespaces, [namespace]                                 Operations
on namespaces
```

```
  nodes, [node]                                                Operations
on nodes
  projects, [project]                                          Operations
on projects
  ps                                                           Show
workloads in a project
  server                                                       Operations
for the server
  settings, [setting]                                          Show
settings for the current server
  ssh                                                          SSH into a
node
  up                                                           apply
compose config
  wait                                                         Wait for
resources cluster, app, project, multiClusterApp
  token
Authenticate and generate new kubeconfig token
  help, [h]                                                    Shows a
list of commands or help for one command

Run 'rancher COMMAND --help' for more information on a command.
```

## 4.1. 登陆 Rancher

链接到 Rancher

```
$ rancher login https://<SERVER_URL> --token <BEARER_TOKEN>
```

登陆演示

```
[root@localhost ~]# rancher login https://192.168.30.13 --token
token-
5q6kw:8b7w2hj85z7cwkwhhvjlp2rw5ls5n8d4gj7vj74jbdch9gv4dzq9km
The authenticity of server 'https://192.168.30.13' can't be
```

```
established.
Cert chain is : [Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 5708461865883058034 (0x4f3887d281d2bf72)
    Signature Algorithm: ECDSA-SHA256
        Issuer: O=dynamiclistener-org,CN=dynamiclistener-ca
        Validity
            Not Before: Nov 29 07:00:54 2021 UTC
            Not After : Nov 29 08:53:00 2022 UTC
        Subject: O=dynamic,CN=dynamic
        Subject Public Key Info:
            Public Key Algorithm: ECDSA
                Public-Key: (256 bit)
                X:

1c:f4:1d:86:32:a7:57:6c:d5:6c:59:86:18:b9:9f:

40:10:e2:f2:99:96:04:96:10:d4:88:82:2c:06:5c:
                    e7:7c
                Y:

16:86:d8:41:0a:f3:c3:f0:e7:0c:29:a4:69:e0:b2:

41:34:73:a6:78:58:e0:a0:df:84:4d:c9:9e:83:3f:
                    bd:fd
                Curve: P-256
        X509v3 extensions:
            X509v3 Key Usage: critical
                Digital Signature, Key Encipherment
            X509v3 Extended Key Usage:
                TLS Web Server Authentication
            X509v3 Authority Key Identifier:

keyid:3D:40:3F:96:30:78:9F:C1:84:1F:94:E0:A2:4D:1C:E1:69:3D:F3:
E4
            X509v3 Subject Alternative Name:
                DNS:localhost, DNS:rancher.cattle-system
                IP Address:127.0.0.1, IP Address:172.19.0.3, IP
Address:192.168.30.13

    Signature Algorithm: ECDSA-SHA256
         30:45:02:21:00:e5:f1:e7:2d:14:fc:25:1f:5c:ea:ce:9a:8d:
         7a:95:e2:d8:bc:64:7a:38:83:3e:84:bc:2e:c7:83:5c:44:5f:
         21:02:20:7c:91:46:fe:2f:bc:f9:18:41:e7:8d:70:0b:1b:c7:
```

```
        e3:c2:b3:12:c5:4f:44:ef:fa:00:15:88:6c:3a:c2:e1:23
]
Do you want to continue connecting (yes/no)? yes
INFO[0002] Saving config to /root/.rancher/cli2.json
```

配置文件

```
[root@localhost ~]# cat /root/.rancher/cli2.json | jq
{
  "Servers": {
    "rancherDefault": {
      "accessKey": "token-5q6kw",
      "secretKey":
"8b7w2hj85z7cwkwhhvjlp2rw5ls5n8d4gj7vj74jbdch9gv4dzq9km",
      "tokenKey": "token-
5q6kw:8b7w2hj85z7cwkwhhvjlp2rw5ls5n8d4gj7vj74jbdch9gv4dzq9km",
      "url": "https://192.168.30.13",
      "project": "local:p-8rzzk",
      "cacert": "-----BEGIN CERTIFICATE-----
\nMIIBpzCCAU2gAwIBAgIBADAKBggqhkjOPQQDAjA7MRwwGgYDVQQKExNkeW5hb
Wlj\nbGlzdGVuZXItb3JnMRswGQYDVQQDExJkeW5hbWljbGlzdGVuZXItY2EwHh
cNMjEx\nMTI5MDcwMDU0WhcNMzExMTI3MDcwMDU0WjA7MRwwGgYDVQQKExNkeW5
hbWljbGlz\ndGVuZXItb3JnMRswGQYDVQQDExJkeW5hbWljbGlzdGVuZXItY2Ew
WTATBgcqhkjO\nPQIBBggqhkjOPQMBBwNCAARppCv2i2N7k6tF4DWBaJAHhOdwC
1SMfymJaj8LUwOP\nfGsMhpLVlI/6Go7FIRPAIkGxoPqc0CeayxrcGun0R66Ao0
IwQDAOBgNVHQ8BAf8E\nBAMCAqQwDwYDVR0TAQH/BAUwAwEB/zAdBgNVHQ4EFgQ
UPUA/ljB4n8GEH5Tgok0c\n4Wk98+QwCgYIKoZIzj0EAwIDSAAwRQIhAJn4aRTO
GsJCaQllCXzDw/vl3o3AmY0a\nqTSMjPRo91vMAiBTnYJMP92NZUoqVV6tG8H+P
dsTK/QeTSHm1m4iju1JBg==\n-----END CERTIFICATE-----",
      "kubeCredentials": null,
      "kubeConfigs": null
    }
  },
  "CurrentServer": "rancherDefault"
}
```

## 4.2. 查看集群

```
[root@localhost ~]# rancher clusters
CURRENT   ID        STATE     NAME      PROVIDER   NODES
CPU       RAM            PODS
*         local     active    local     Unknown    1
0.10/4    0.07/7.51 GB   5/110
```

## 4.3. 查看节点

```
[root@localhost ~]# rancher nodes
ID                     NAME         STATE     POOL
DESCRIPTION
local:machine-5p4pj   local-node   active
```

## 4.4. catalog

```
[root@localhost ~]# rancher catalog
ID        NAME      URL
BRANCH    KIND
helm      helm      https://kubernetes-
charts.storage.googleapis.com/   master    helm
library   library   https://git.rancher.io/charts
master    helm
```

## 4.5. 查看设置

```
[root@localhost ~]# rancher settings
ID                NAME              VALUE
agent-image       agent-image       rancher/rancher-agent:v2.1.6
```

```
api-ui-version   api-ui-version   1.1.6
cacerts          cacerts          -----BEGIN CERTIFICATE-----
MIIC7jCCAdagAwIBAgIBADANBgkqhkiG9w0BAQsFADAoMRIwEAYDVQQKEwl0aGU
t
cmFuY2gxEjAQBgNVBAMTCWNhdHRsZS1jYTAeFw0xOTAzMTkwODUxNTNaFw0yOTA
z
MTYwODUxNTNaMCgxEjAQBgNVBAoTCXRoZS1yYW5jaDESMBAGA1UEAxMJY2F0dGx
l
LWNhMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA2j/x0F+VpdPHv6c
e
zKYAcGeGDjHfv8YL4Q6NpO4m6N3z3WwC9e9qNq062TGWml3q3xIu0ll229vTXYZ
G
YaW7hdIYdNcgE4d2DSFiM0rV2CCiBheAidcvGWTmVuRqDaH7+ofxUeuz940osjc
Y
GKYkugUnPA9n6cXRF8KF9a6d6t2Kcwqyd3A5c5ld+lPsu2u6lbJhJArdGWmi8Ii
q
CpkgmPyabCJhpF/YRtLfZ6+mQ0SpcapAuVvXiSGyHjnXykxywthSnTHgSJp48SV
7
XCYJx5skU4rqKOWRgwfgQLWnLdV6kWLTH7EE+aiBwt2lygZUR3Ekpr3rXe7Q+dH
h
ygOYVwIDAQABoyMwITAOBgNVHQ8BAf8EBAMCAqQwDwYDVR0TAQH/BAUwAwEB/zA
N
BgkqhkiG9w0BAQsFAAOCAQEAMfDWlobAEGKvhLW380JA93IcafbQGgTLyhBglqw
F
B4SBj56ZTki2mZrccUZXYKzIPTRwY39cnBakjkkczm4Hkci3Ag+4hz9g5mJWAa/
H
mYrxNEdUJNiih7RNwBne0MaLSHH1MjBfmCSExCJkqlXuD4XXY7dJ05ZQ6urWB2Z
I
lC7oqwGUxnvDSEMONHLTNQy+5yA+jSae9holJ5kpvEq6vE9A1PoUg4/leHZXsI5
L
h+gDJX+WbAn5rdyDB0F4XJxn/glQPGxFNib8EUGt4b58re4x9A8ZaVbzL+KEKrS
1
7QO13jU95Cy5+FA5GKO3YILrkvCFIoEaRe83jlbiQZSSaw==
-----END CERTIFICATE-----
cli-url-darwin             cli-url-darwin
https://releases.rancher.com/cli2/v2.0.6/rancher-darwin-amd64-
v2.0.6.tar.gz
cli-url-linux              cli-url-linux
https://releases.rancher.com/cli2/v2.0.6/rancher-linux-amd64-
v2.0.6.tar.gz
cli-url-windows            cli-url-windows
https://releases.rancher.com/cli2/v2.0.6/rancher-windows-386-
v2.0.6.zip
engine-install-url         engine-install-url
https://releases.rancher.com/install-docker/17.03.sh
```

```
engine-iso-url              engine-iso-url
https://releases.rancher.com/os/latest/rancheros-vmware.iso
engine-newest-version       engine-newest-version       v17.12.0
engine-supported-range      engine-supported-range      ~v1.11.2 ||
~v1.12.0 || ~v1.13.0 || ~v17.03.0
first-login                 first-login                 false
helm-version                helm-version                v2.10.0-
rancher5
ingress-ip-domain           ingress-ip-domain           xip.io
install-uuid                install-uuid                6002fd6a-
f4ae-454b-a17b-f90c64aafa2a
k8s-version                 k8s-version                 v1.11.6-
rancher1-1
k8s-version-to-images       k8s-version-to-images       {"v1.10.12-
rancher1-1":null,"v1.11.6-rancher1-1":null,"v1.12.4-rancher1-
1":null,"v1.9.7-rancher2-2":null}
machine-version             machine-version             v0.15.0-
rancher1-1
namespace                   namespace
peer-service                peer-service
rdns-base-url               rdns-base-url
https://api.lb.rancher.cloud/v1
rke-version                 rke-version                 v0.1.15
server-image                server-image
rancher/rancher
server-url                  server-url
https://192.168.0.157
server-version              server-version              v2.1.6
system-default-registry     system-default-registry
system-namespaces           system-namespaces           kube-
system,kube-public,cattle-system,cattle-alerting,cattle-
logging,cattle-pipeline,ingress-nginx
telemetry-opt               telemetry-opt               in
telemetry-uid               telemetry-uid               bf1dd7d1-
e0ed-475e-9dfe-e9af2d71f9b3
ui-feedback-form            ui-feedback-form
ui-index                    ui-index
https://releases.rancher.com/ui/latest2/index.html
ui-path                     ui-path
/usr/share/rancher/ui
ui-pl                       ui-pl                       rancher
whitelist-domain            whitelist-domain
forums.rancher.com
windows-agent-image         windows-agent-image
rancher/rancher-agent:v2.1.6-nanoserver-1803
```

## 4.6. rancher kubectl

```
[root@localhost ~]# rancher kubectl get pods --all-namespaces
NAMESPACE                    NAME
READY   STATUS    RESTARTS   AGE
cattle-fleet-local-system   fleet-agent-59b74595c-xgnjg
1/1     Running   5          129m
cattle-fleet-system         fleet-controller-66cc4c6b5b-xswdl
1/1     Running   5          131m
cattle-fleet-system         gitjob-5778966b7c-jqdtj
1/1     Running   5          131m
cattle-system               rancher-webhook-6979fbd4bf-gs8vk
1/1     Running   5          129m
kube-system                 coredns-7448499f4d-4n2vt
1/1     Running   5          134m
```

# 5. K3s

autok3s/k3s/k3d 三种封装，安装最简单的是 autok3s，其次是 k3d，如果喜欢蒸腾就安装原生 k3s。

## 5.1. AutoK3s

https://github.com/cnrancher/autok3s

挂载 iptables 内核模块，否则 traefik slb 和 service 起不来

```
modprobe ip_tables
```

```
cat > /etc/modules-load.d/k3s.conf <<-EOF
ip_tables
ip_conntrack
br_netfilter
EOF
```

设置主机名

```
hostnamectl set-hostname master
```

安装 AutoK3s

```
docker run -itd --name=autok3s --restart=unless-stopped --
```

```
net=host -v /var/run/docker.sock:/var/run/docker.sock
cnrancher/autok3s:v0.5.2
```

安装 AutoK3s 命令行

```
curl -sS https://rancher-mirror.oss-cn-
beijing.aliyuncs.com/autok3s/install.sh  |
INSTALL_AUTOK3S_MIRROR=cn sh
```

首次运行

```
[root@master ~]# autok3s
? This is the very first time using autok3s,
  would you like to share metrics with us?
  You can always your mind with telemetry command Yes

               ,        ,
  ,------------|'------'|             _        _     _____
 / .           '-'    |-             | |      | |   |____ |
 \\/|             |    |   __ _ _   _| |_ ___ | | __   / / ___
   |   .________.'----'   / _  | | | | __/ _ \| |/ /   \ \/ __|
   |   |        |    |   | (_| | |_| | || (_) |   <.___/ /\__ \
   \\___/        \\___/   \__,_|\__,_|\__\___/|_|\_\____/ |___/

Usage:
  autok3s [flags]
  autok3s [command]

Available Commands:
  completion  Generate completion script
  create      Create a K3s cluster
  delete      Delete a K3s cluster
  describe    Show details of a specific resource
  explorer    Enable kube-explorer for K3s cluster
  help        Help about any command
  join        Join one or more K3s node(s) to an existing
```

```
cluster
  kubectl     Kubectl controls the Kubernetes cluster manager
  list        Display all K3s clusters
  serve       Run as daemon and serve HTTP/HTTPS request
  ssh         Connect to a K3s node through SSH
  telemetry   Telemetry status for autok3s
  upgrade     Upgrade a K3s cluster to specified version
  version     Display autok3s version

Flags:
  -d, --debug                          Enable log debug level
  -h, --help                           help for autok3s
      --log-flush-frequency duration   Maximum number of
seconds between log flushes (default 5s)

Global Environments:
  AUTOK3S_CONFIG  Path to the cfg file to use for CLI requests
(default ~/.autok3s)
  AUTOK3S_RETRY   The number of retries waiting for the desired
state (default 20)

Use "autok3s [command] --help" for more information about a
command.
```

如果你想卸载它

```
Creating uninstall script /usr/local/bin/autok3s-uninstall.sh
kubectl --kubeconfig /etc/rancher/k3s/k3s.yaml get pods --all-
namespaces
```

## 命令行创建集群

创建 k3d 集群

```
autok3s create --provider k3d --master 1 --name test --worker 1
```

```
--api-port 0.0.0.0:6443 --image rancher/k3s:v1.21.7-k3s1
```

### 私有镜像库

指定私有镜像库

```
autok3s create --provider  k3d --master  1 --name  test --
worker  1 --api-port  0.0.0.0:6443 --image
rancher/k3s:v1.21.7-k3s1 --registry
https://registry.netkiller.cn
```

https://rancher.com/docs/k3s/latest/en/installation/private-registry/

### 暴漏 80/443

给宿主主机暴漏 ingress 80/443 端口

```
autok3s create --provider k3d --master 1 --name test --token
0ab46344f7f62488f771f1332feeabf6 --worker 1 --k3s-install-
script https://get.k3s.io --api-port 172.18.200.5:6443 --image
rancher/k3s:v1.21.7-k3s1 --ports '80:80@loadbalancer' --ports
'443:443@loadbalancer'
```

验证集群是否工作正常

```
l
kubectl create service clusterip nginx --tcp=80:80

cat <<EOF | kubectl apply -f -
```

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: nginx
  annotations:
    ingress.kubernetes.io/ssl-redirect: "false"
spec:
  rules:
  - http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: nginx
            port:
              number: 80
EOF
```

默认 ingress 地址是 br 网桥的

```
[root@master ~]# ip addr | grep  br-
4: br-2ad0dd2291af: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500
qdisc noqueue state UP group default
    inet 172.19.0.1/16 brd 172.19.255.255 scope global br-
2ad0dd2291af
```

```
# Run kubectl commands inside here
# e.g. kubectl get all
> kubectl get ingress
NAME    CLASS    HOSTS   ADDRESS                     PORTS   AGE
nginx   <none>   *       172.19.0.2,172.19.0.3       80      4m18s
```

我们已经将 80/443 暴漏给了宿主主机，所以可以直接用宿主主机 IP访问 kubernetes 集群

```
[root@master ~]# curl http://localhost
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully
installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

## 扩展本地存储

服务器是OS安装在一块 256G 的 SSD 上，默认本地存储路径是 /var/lib/rancher/k3s/storage，我们需要扩展本地存储的空间容量，有两个方案：

将 1TB 硬盘挂载到 /var/lib/rancher/k3s/storage，另一种方案，由于 1TB硬盘已经在使用，并且挂载到了 /opt 目录，这时我们使用 --

volumes '/opt/kubernetes:/var/lib/rancher/k3s/storage' 将 /var/lib/rancher/k3s/storage 挂载到 /opt/kubernetes 目录。

```
autok3s create --provider k3d --master 1 --name dev --token
7fc4b9a088a3c02ed9f3285359f1d322 --worker 1 --k3s-install-
script https://get.k3s.io --api-port 0.0.0.0:26080 --image
rancher/k3s:v1.21.7-k3s1 --volumes
'/opt/kubernetes:/var/lib/rancher/k3s/storage'
```

配置节点路径映射，修改 local-path-config

```
config.json: |-
        {
                "nodePathMap":[
                {

"node":"DEFAULT_PATH_FOR_NON_LISTED_NODES",
                        "paths":["/opt/local-path-provisioner"]
                },
                {
                        "node":"yasker-lp-dev1",
                        "paths":["/opt/local-path-provisioner",
"/data1"]
                },
                {
                        "node":"yasker-lp-dev3",
                        "paths":[]
                }
                ]
        }
```

**Agent 代理安装**

```
hostnamectl set-hostname node1
```

查看 Master Token

```
[docker@master ~]$ docker ps | egrep "k3d.*server" |grep -v lb
12b9c210b858   rancher/k3s:v1.21.7-k3s1   "/bin/k3d-
entrypoint…"   2 days ago          Up 2 days           k3d-
test-server-0

[docker@master ~]$ docker exec -it k3d-test-server-0 cat
/var/lib/rancher/k3s/server/node-token
K1083de74aba3f4fe80d744ab2a506d037165f4c475d0ca3636d48a371aac6e
f0ac::server:0ab46344f7f62488f771f1332feeabf6
```

在节点服务器安装代理

```
SERVER=172.18.200.5
TOKEN=K1083de74aba3f4fe80d744ab2a506d037165f4c475d0ca3636d48a37
1aac6ef0ac::server:0ab46344f7f62488f771f1332feeabf6
curl -sfL https://rancher-mirror.oss-cn-
beijing.aliyuncs.com/k3s/k3s-install.sh | INSTALL_K3S_MIRROR=cn
K3S_URL=https://${SERVER}:6443 K3S_TOKEN=${TOKEN} sh -
systemctl enable k3s-agent
```

加入集群

```
K3S_TOKEN="K104fddbe58cad213694b0346db17ae060fc0974e7cfdbb9063a
a1309363de16996::server:0ab46344f7f62488f771f1332feeabf6"
K3S_URL="https://172.18.200.5:6443"
```

```
curl -sfL https://rancher-mirror.oss-cn-
beijing.aliyuncs.com/k3s/k3s-install.sh | INSTALL_K3S_MIRROR=cn
K3S_URL=${K3S_URL} K3S_TOKEN=${K3S_TOKEN} sh -s - --docker
```

回到 Master 查看节点

```
[root@master ~]# kubectl get node
NAME                    STATUS   ROLES                  AGE
VERSION
localhost.localdomain   Ready    control-plane,master   28m
v1.24.4+k3s1
node1                   Ready    <none>                 117s
v1.24.4+k3s1
```

如果此前已经安装了 K3s，需要手工加入 Master

```
k3s agent --server https://10.12.1.40:6443 --token
"K1083de74aba3f4fe80d744ab2a506d037165f4c475d0ca3636d48a371aac6
ef0ac::server:0ab46344f7f62488f771f1332feeabf6"
```

也可以修改环境变量配置文件

```
[root@node1 ~]# cat /etc/systemd/system/k3s-agent.service.env
K3S_TOKEN="K1083de74aba3f4fe80d744ab2a506d037165f4c475d0ca3636d
48a371aac6ef0ac::server:0ab46344f7f62488f771f1332feeabf6"
K3S_URL="https://172.18.200.5:6443"
```

```
> kubectl describe nodes agent-1
Name:               agent-1
Roles:              <none>
Labels:             beta.kubernetes.io/arch=amd64
                    beta.kubernetes.io/instance-type=k3s
                    beta.kubernetes.io/os=linux
                    egress.k3s.io/cluster=true
                    kubernetes.io/arch=amd64
                    kubernetes.io/hostname=agent-1
                    kubernetes.io/os=linux
                    node.kubernetes.io/instance-type=k3s
Annotations:        flannel.alpha.coreos.com/backend-data:
{"VNI":1,"VtepMAC":"0e:14:1e:7c:fc:e9"}
                    flannel.alpha.coreos.com/backend-type:
vxlan
                    flannel.alpha.coreos.com/kube-subnet-
manager: true
                    flannel.alpha.coreos.com/public-ip:
172.18.200.51
                    k3s.io/hostname: agent-1
                    k3s.io/internal-ip: 172.18.200.51
                    k3s.io/node-args: ["agent"]
                    k3s.io/node-config-hash:
HJIVMRMG74UTQMXBAZD4NLDPY3FZHN7PYGB7RA7CUGXEDUTUTBTQ====
                    k3s.io/node-env:

{"K3S_DATA_DIR":"/var/lib/rancher/k3s/data/577968fa3d58539cc426
5245941b7be688833e6bf5ad7869fa2afe02f15f1cd2","K3S_TOKEN":"****
****","K3S_U...
                    node.alpha.kubernetes.io/ttl: 0
                    volumes.kubernetes.io/controller-managed-
attach-detach: true
CreationTimestamp:  Tue, 06 Sep 2022 17:33:21 +0000
Taints:             <none>
Unschedulable:      false
Lease:
  HolderIdentity:  agent-1
  AcquireTime:     <unset>
  RenewTime:       Wed, 07 Sep 2022 18:40:08 +0000
Conditions:
  Type             Status  LastHeartbeatTime
LastTransitionTime               Reason
Message
  ----             ------  -----------------                 --
----------------               ------                        --
```

```
-----
  MemoryPressure   False   Wed, 07 Sep 2022 18:35:57 +0000
Wed, 07 Sep 2022 03:48:43 +0000   KubeletHasSufficientMemory
kubelet has sufficient memory available
  DiskPressure     False   Wed, 07 Sep 2022 18:35:57 +0000
Wed, 07 Sep 2022 03:48:43 +0000   KubeletHasNoDiskPressure
kubelet has no disk pressure
  PIDPressure      False   Wed, 07 Sep 2022 18:35:57 +0000
Wed, 07 Sep 2022 03:48:43 +0000   KubeletHasSufficientPID
kubelet has sufficient PID available
  Ready            True    Wed, 07 Sep 2022 18:35:57 +0000
Wed, 07 Sep 2022 03:48:43 +0000   KubeletReady
kubelet is posting ready status
Addresses:
  InternalIP:  172.18.200.51
  Hostname:    agent-1
Capacity:
  cpu:                 16
  ephemeral-storage:  181197372Ki
  hugepages-1Gi:      0
  hugepages-2Mi:      0
  memory:             65237592Ki
  pods:               110
Allocatable:
  cpu:                 16
  ephemeral-storage:  176268803344
  hugepages-1Gi:      0
  hugepages-2Mi:      0
  memory:             65237592Ki
  pods:               110
System Info:
  Machine ID:                 bfc31b708a794f8bad984bd60770ed0f
  System UUID:                1514a1f0-c451-11eb-8522-
ac3ccdeb3900
  Boot ID:                    5c0c8375-220a-4abd-8a6d-
7debafc6a331
  Kernel Version:             5.14.0-70.22.1.el9_0.x86_64
  OS Image:                   AlmaLinux 9.0 (Emerald Puma)
  Operating System:           linux
  Architecture:               amd64
  Container Runtime Version:  containerd://1.6.6-k3s1
  Kubelet Version:            v1.24.4+k3s1
  Kube-Proxy Version:         v1.24.4+k3s1
PodCIDR:                      10.42.2.0/24
PodCIDRs:                     10.42.2.0/24
```

```
ProviderID:                   k3s://agent-1
Non-terminated Pods:          (11 in total)
  Namespace                   Name                          CPU
Requests  CPU Limits  Memory Requests  Memory Limits  AGE
  ---------                   ----                          ------
------  ----------  ---------------  -------------  ---
  kube-system                 svclb-traefik-hhvvv           0 (0%)
0 (0%)      0 (0%)           0 (0%)         25h
  default                     nacos-0                       0 (0%)
0 (0%)      0 (0%)           0 (0%)         14h
  default                     nacos-1                       0 (0%)
0 (0%)      0 (0%)           0 (0%)         14h
  default                     elasticsearch-data-1          0 (0%)
0 (0%)      0 (0%)           0 (0%)         36m
  default                     nginx-565785f75c-gmblp        0 (0%)
0 (0%)      0 (0%)           0 (0%)         35m
  default                     nginx-565785f75c-lhhcl        0 (0%)
0 (0%)      0 (0%)           0 (0%)         30m
  default                     nginx-565785f75c-rpc4k        0 (0%)
0 (0%)      0 (0%)           0 (0%)         29m
  default                     nginx-565785f75c-fr2s7        0 (0%)
0 (0%)      0 (0%)           0 (0%)         29m
  default                     nginx-565785f75c-5rjj9        0 (0%)
0 (0%)      0 (0%)           0 (0%)         29m
  default                     nginx-565785f75c-2bc9p        0 (0%)
0 (0%)      0 (0%)           0 (0%)         28m
  default                     quickstart-es-default-0       100m
(0%)     100m (0%)   2Gi (3%)         2Gi (3%)       10h
Allocated resources:
  (Total limits may be over 100 percent, i.e., overcommitted.)
  Resource                   Requests   Limits
  --------                   --------   ------
  cpu                        100m (0%)  100m (0%)
  memory                     2Gi (3%)   2Gi (3%)
  ephemeral-storage          0 (0%)     0 (0%)
  hugepages-1Gi              0 (0%)     0 (0%)
  hugepages-2Mi              0 (0%)     0 (0%)
Events:                      <none>
```

## 5.2. 安装 K3s（Docker 模式）

**Server**

设置主机名

```
hostnamectl set-hostname master
```

Docker 方式安装

```
curl -sfL https://rancher-mirror.oss-cn-
beijing.aliyuncs.com/k3s/k3s-install.sh | INSTALL_K3S_MIRROR=cn
sh -s - --docker
```

**Agent**

设置主机名

```
hostnamectl set-hostname agent-1
```

前往 master 查看 Token

```
[root@master ~]# cat /var/lib/rancher/k3s/server/node-token
K10b614928142836a5262a802c0d3056f0047f057c895373651b723697a261b
128b::server:1d436565a84f8e4bdd434b17752a2071
```

在 Agent 节点服务器执行下面命令，加入 master 集群（Docker 方式）

```
K3S_TOKEN="K10b614928142836a5262a802c0d3056f0047f057c895373651b
723697a261b128b::server:1d436565a84f8e4bdd434b17752a2071"
K3S_URL="https://172.18.200.5:6443"
curl -sfL https://rancher-mirror.oss-cn-
beijing.aliyuncs.com/k3s/k3s-install.sh | INSTALL_K3S_MIRROR=cn
K3S_URL=${K3S_URL} K3S_TOKEN=${K3S_TOKEN} sh -s - --docker
```

前往 master 查看节点

```
[root@master ~]# kubectl get node -o wide
NAME       STATUS     ROLES                  AGE   VERSION
INTERNAL-IP     EXTERNAL-IP   OS-IMAGE
KERNEL-VERSION              CONTAINER-RUNTIME
agent-1    Ready      <none>                 2d    v1.24.4+k3s1
172.18.200.51   <none>        AlmaLinux 9.0 (Emerald Puma)
5.14.0-70.22.1.el9_0.x86_64   docker://20.10.17
master     Ready      control-plane,master   2d    v1.24.4+k3s1
172.18.200.5    <none>        AlmaLinux 9.0 (Emerald Puma)
5.14.0-70.22.1.el9_0.x86_64   docker://20.10.17
agent-2    NotReady   <none>                 6s    v1.24.4+k3s1
172.18.200.52   <none>        AlmaLinux 9.0 (Emerald Puma)
5.14.0-70.13.1.el9_0.x86_64   docker://20.10.18
```

## 安装 **kube-explorer**

https://github.com/cnrancher/kube-explorer

```
docker rm -f kube-explorer
docker run -itd --name=kube-explorer --restart=unless-stopped -
```

```
-net=host -v
/etc/rancher/k3s/k3s.yaml:/etc/rancher/k3s/k3s.yaml:ro -e
KUBECONFIG=/etc/rancher/k3s/k3s.yaml cnrancher/kube-
explorer:latest
```

https://127.0.0.1:9443/dashboard/

## 5.3. 安装 K3s（VM 模式）

K3S 的安装方式有多种，官方提供的 k3s-install.sh，还有第三方的 k3d 和 k3sup

**Server 服务安装**

设置主机名

```
hostnamectl set-hostname master
```

运行在虚拟机之下

```
curl -sfL https://get.k3s.io | sh -
```

国内镜像

```
curl -sfL http://rancher-mirror.cnrancher.com/k3s/k3s-
install.sh | INSTALL_K3S_MIRROR=cn sh -
systemctl enable k3s
```

查看节点启动状态

```
[root@master ~]# kubectl get node
NAME                     STATUS   ROLES                    AGE
VERSION
localhost.localdomain    Ready    control-plane,master     28m
v1.24.4+k3s1
```

查看节点 Pod 状态

```
kubectl --kubeconfig /etc/rancher/k3s/k3s.yaml get pods --all-
namespaces
```

## Agent 代理安装

设置主机名

```
hostnamectl set-hostname node1
```

查看 Master Token

```
[root@master ~]# kubectl get node
NAME                     STATUS   ROLES                    AGE
VERSION
localhost.localdomain    Ready    control-plane,master     28m
v1.24.4+k3s1

[root@master ~]# cat /var/lib/rancher/k3s/server/node-token
```

```
K1000ba39a142b3712d2ffb1459a63f6a7f58b082aeb53406dab15d8cee0f3c
2ff0::server:5713047feb086388c19663f69cccc966
```

在节点服务器安装代理

```
SERVER=172.18.200.5
TOKEN=K1000ba39a142b3712d2ffb1459a63f6a7f58b082aeb53406dab15d8c
ee0f3c2ff0::server:5713047feb086388c19663f69cccc966
curl -sfL https://rancher-mirror.oss-cn-
beijing.aliyuncs.com/k3s/k3s-install.sh | INSTALL_K3S_MIRROR=cn
K3S_URL=https://${SERVER}:6443 K3S_TOKEN=${TOKEN} sh -
systemctl enable k3s-agent
```

回到 Master 查看节点

```
[root@master ~]# kubectl get node
NAME                    STATUS   ROLES                  AGE
VERSION
localhost.localdomain   Ready    control-plane,master   28m
v1.24.4+k3s1
node1                   Ready    <none>                 117s
v1.24.4+k3s1

[root@master ~]# kubectl get nodes -o wide
NAME      STATUS   ROLES                  AGE   VERSION
INTERNAL-IP     EXTERNAL-IP   OS-IMAGE
KERNEL-VERSION                CONTAINER-RUNTIME
master    Ready    control-plane,master   22h   v1.24.4+k3s1
172.18.200.5    <none>        AlmaLinux 9.0 (Emerald Puma)
5.14.0-70.22.1.el9_0.x86_64   docker://20.10.17
agent-1   Ready    <none>                 22h   v1.24.4+k3s1
172.18.200.51   <none>        AlmaLinux 9.0 (Emerald Puma)
5.14.0-70.22.1.el9_0.x86_64   docker://20.10.17
```

## 5.4. k3d

k3d is a lightweight wrapper to run k3s (Rancher Lab’s minimal Kubernetes distribution) in docker.

### 安装 k3d

Mac 安装 k3d

```
Neo-iMac:~ neo$ brew install k3d
```

Linux 安装 k3d

**wget -q -O - https://raw.githubusercontent.com/k3d-io/k3d/main/install.sh | bash**

```
[root@netkiller ~]# wget -q -O -
https://raw.githubusercontent.com/k3d-io/k3d/main/install.sh |
bash
Preparing to install k3d into /usr/local/bin
k3d installed into /usr/local/bin/k3d
Run 'k3d --help' to see what you can do with it.
```

### 创建集群

创建并启动集群

```
Neo-iMac:~ neo$ k3d cluster create mycluster
INFO[0000] Prep: Network
INFO[0000] Created network 'k3d-mycluster'
```

```
INFO[0000] Created volume 'k3d-mycluster-images'
INFO[0000] Starting new tools node...
INFO[0001] Creating node 'k3d-mycluster-server-0'
INFO[0006] Pulling image 'docker.io/rancher/k3d-tools:5.2.2'
INFO[0006] Pulling image 'docker.io/rancher/k3s:v1.21.7-k3s1'
INFO[0016] Starting Node 'k3d-mycluster-tools'
INFO[0036] Creating LoadBalancer 'k3d-mycluster-serverlb'
INFO[0041] Pulling image 'docker.io/rancher/k3d-proxy:5.2.2'
INFO[0057] Using the k3d-tools node to gather environment
information
INFO[0058] Starting cluster 'mycluster'
INFO[0058] Starting servers...
INFO[0059] Starting Node 'k3d-mycluster-server-0'
INFO[0078] All agents already running.
INFO[0078] Starting helpers...
INFO[0079] Starting Node 'k3d-mycluster-serverlb'
INFO[0087] Injecting '192.168.65.2 host.k3d.internal' into
/etc/hosts of all nodes...
INFO[0087] Injecting records for host.k3d.internal and for 2
network members into CoreDNS configmap...
INFO[0088] Cluster 'mycluster' created successfully!
INFO[0088] You can now use it like this:
kubectl cluster-info
```

映射80端口

```
k3d cluster create mycluster --api-port 127.0.0.1:6445 --
servers 3 --agents 2 --port '80:80@loadbalancer'
```

```
Neo-iMac:~ neo$ k3d cluster create mycluster --api-port
127.0.0.1:6445 --servers 3 --agents 2 --port
'80:80@loadbalancer'
INFO[0000] portmapping '80:80' targets the loadbalancer:
defaulting to [servers:*:proxy agents:*:proxy]
INFO[0000] Prep: Network
INFO[0000] Created network 'k3d-mycluster'
```

```
INFO[0000] Created volume 'k3d-mycluster-images'
INFO[0000] Creating initializing server node
INFO[0000] Creating node 'k3d-mycluster-server-0'
INFO[0000] Starting new tools node...
INFO[0001] Starting Node 'k3d-mycluster-tools'
INFO[0002] Creating node 'k3d-mycluster-server-1'
INFO[0003] Creating node 'k3d-mycluster-server-2'
INFO[0004] Creating node 'k3d-mycluster-agent-0'
INFO[0005] Creating node 'k3d-mycluster-agent-1'
INFO[0005] Creating LoadBalancer 'k3d-mycluster-serverlb'
INFO[0005] Using the k3d-tools node to gather environment
information
INFO[0007] Starting cluster 'mycluster'
INFO[0007] Starting the initializing server...
INFO[0007] Starting Node 'k3d-mycluster-server-0'
INFO[0012] Starting servers...
INFO[0013] Starting Node 'k3d-mycluster-server-1'
INFO[0045] Starting Node 'k3d-mycluster-server-2'
INFO[0069] Starting agents...
INFO[0070] Starting Node 'k3d-mycluster-agent-1'
INFO[0070] Starting Node 'k3d-mycluster-agent-0'
INFO[0081] Starting helpers...
INFO[0081] Starting Node 'k3d-mycluster-serverlb'
INFO[0089] Injecting '192.168.65.2 host.k3d.internal' into
/etc/hosts of all nodes...
INFO[0089] Injecting records for host.k3d.internal and for 6
network members into CoreDNS configmap...
INFO[0090] Cluster 'mycluster' created successfully!
INFO[0091] You can now use it like this:
kubectl cluster-info
```

除了使用命令，还可以使用 yaml 配置文件创建集群

```
apiVersion: k3d.io/v1alpha2
kind: Simple
name: mycluster
servers: 1
agents: 2
kubeAPI:
  hostPort: "6443" # same as `--api-port '6443'`
```

```
ports:
  - port: 8080:80  # same as `--port '8080:80@loadbalancer'`
    nodeFilters:
      - loadbalancer
  - port: 8443:443 # same as `--port '8443:443@loadbalancer'`
    nodeFilters:
      - loadbalancer
```

```
$ k3d cluster create --config /path/to/mycluster.yaml
```

查看信息

```
Neo-iMac:~ neo$ k3d cluster list
NAME        SERVERS   AGENTS   LOADBALANCER
mycluster   3/3       2/2      true
```

查看集群信息

```
Neo-iMac:~ neo$ kubectl cluster-info
Kubernetes control plane is running at https://0.0.0.0:60268
CoreDNS is running at
https://0.0.0.0:60268/api/v1/namespaces/kube-
system/services/kube-dns:dns/proxy
Metrics-server is running at
https://0.0.0.0:60268/api/v1/namespaces/kube-
system/services/https:metrics-server:/proxy

To further debug and diagnose cluster problems, use 'kubectl
cluster-info dump'.
Neo-iMac:~ neo$
```

```
```

查看节点

```
Neo-iMac:~ neo$ kubectl get nodes
NAME                     STATUS   ROLES                  AGE
VERSION
k3d-mycluster-server-0   Ready    control-plane,master   2m10s
v1.21.7+k3s1
```

## 删除集群

删除集群

```
Neo-iMac:~ neo$ k3d cluster delete mycluster
INFO[0000] Deleting cluster 'mycluster'
INFO[0002] Deleting cluster network 'k3d-mycluster'
INFO[0003] Deleting image volume 'k3d-mycluster-images'
INFO[0003] Removing cluster details from default kubeconfig...
INFO[0003] Removing standalone kubeconfig file (if there is
one)...
INFO[0003] Successfully deleted cluster mycluster!
```

## 演示

**部署 nginx**

```
kubectl create deployment nginx --image=nginx:alpine
kubectl create service clusterip nginx --tcp=80:80
```

```
cat <<EOF | kubectl apply -f -
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: nginx
  annotations:
    ingress.kubernetes.io/ssl-redirect: "false"
spec:
  rules:
  - http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: nginx
            port:
              number: 80
EOF
```

操作演示

```
Neo-iMac:~ neo$ kubectl create deployment nginx --
image=nginx:alpine
deployment.apps/nginx created
Neo-iMac:~ neo$ kubectl create service clusterip nginx --
tcp=80:80
service/nginx created
Neo-iMac:~ neo$ cat <<EOF | kubectl apply -f -
> apiVersion: networking.k8s.io/v1
> kind: Ingress
> metadata:
>   name: nginx
>   annotations:
>     ingress.kubernetes.io/ssl-redirect: "false"
> spec:
>   rules:
>   - http:
>       paths:
>       - path: /
>         pathType: Prefix
>         backend:
>           service:
>             name: nginx
>             port:
>               number: 80
> EOF
ingress.networking.k8s.io/nginx created
```

使用浏览器或者CURL命令访问 http://localhost

```
Neo-iMac:~ neo$ curl http://localhost
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully
installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

## 配置文件

导出集群配置文件

```
Netkiller-iMac:~ neo$ k3d kubeconfig write mycluster
/Users/neo/.k3d/kubeconfig-mycluster.yaml
Netkiller-iMac:~ neo$ cat /Users/neo/.k3d/kubeconfig-
mycluster.yaml
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data:
LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUJkakNDQVIyZ0F3SUJBZ0l
CQURBS0JnZ3Foa2pPUFFRREFqQWpNU0V3SHdZRFZRUUREQmhyTTNNdGMyVnkKZG
1WeUxXTmhRREUyTkRFME16WTVNelV3SGhjTk1qSXdNVEEyTURJME1qRTFXaGNOT
XpJd01UQTBNREkwTWpFMQpXakFqTVNFd0h3WURWUVFEREJock0zTXRjMlZ5ZG1W
eUxXTmhRREUyTkRFME16WTVNelV3V1RBVEJnY3Foa2pPPClBRSUJCZ2dxaGtqT1B
RTUJCd05DQUFUQVZKN01XdVY3dzA5dGZybUswbDAybkxOcjFiaGpXM1hIZEgrQU
tCdWEKREFBZ3UrNHF4dVdyNHBkbGpraVNrL3ZZMEJjVWJMZ1RkemJnSEY4UnA1O
VpvMEl3UURBT0JnTlZIUThCQWY4RQpCQU1DQXFRd0R3WURWUjBUQVFIL0JBVXdB
d0VCL3pBZEJnTlZIUTRFRmdRVUZ2UXVRTVBjeStrbTFla2pqaUtUCmRoZ1c4TjB
3Q2dZSUtvWkl6ajBFQXdJRFJ3QXdSQUlnVGMvZDBHWjN5aWRuZ2dXamZGWnowc0
R6V3diVXkzV0IKVmZYamZ1Tis3UjRDSUJ4ZmttSUs1Z1NTL0RNUjltc0VxYUsxZ
VNGTEl2bHZuNXhaeE53RDJoUlgKLS0tLS1FTkQgQ0VSVElGSUNBVEUtLS0tLQo=
    server: https://127.0.0.1:6445
  name: k3d-mycluster
contexts:
- context:
    cluster: k3d-mycluster
    user: admin@k3d-mycluster
  name: k3d-mycluster
current-context: k3d-mycluster
kind: Config
preferences: {}
users:
- name: admin@k3d-mycluster
  user:
    client-certificate-data:
LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUJrVENDQVRlZ0F3SUJBZ0l
JVnR3SGsxWDlUam93Q2dZSUtvWkl6ajBFQXdJd0l6RWhNQjhHQTFVRUF3d1kKYX
pOekxXTnNhV1Z1ZEMxallVQXhOalF4TkRNMk9UTTFNQjRYRFRJeU1ERXdOakF5T
kRJeE5Wb1hEVEl6TURFdwpOakF5TkRJMU0xb3dNREVYTUJVR0ExVUVDaE1PYzNs
```

```
emRHVnRPbTFoYzNSbGNuTXhGVEFUQmdOVkJBTVRESE41CmMzUmxiVHBoWkcxcGJ
qQlpNQk1HQnlxR1NNNDlBZ0VHQ0NxR1NNNDlBd0VIQTBJQUJCcFNScmNGMW9VQU
FCRW4Kb2hZM1haWmpoMUhkNks0eEtXVUpsc3A2blR0UzNFbDJJQjZrUmZIcGNwa
DdjQ3NaUnFvV2RsT1MxdlFtNGM3VgplNVZ6aEY2alNEQkdNQTRHQTFVZER3RUIv
d1FFQXdJRm9EQVRCZ05WSFNVRUREQUtCZ2dyQmdFRkJRY0RBakFmCkJnTlZIU01
FR0RBV2dCVFhrTVpDYnJXVTNKQmxIb0t2Z0F4MDF6TUJUVEFLQmdncWhrak9QUV
FEQWdOSUFEQkYKQWlFQTFIQ0M1OUlaS3FieVQ2MExSS2pvcWNWMFJiK3BWZ1FLd
U1aR3YxZXFvOGdDSUZFMjB6OTg1ZStnR3dGYQppK3FkenFYQTVKU2FrV05naVE0
TUZLcExpVDI3Ci0tLS0tRU5EIENFUlRJRklDQVRFLS0tLS0KLS0tLS1CRUdJTiB
DRVJUSUZJQ0FURS0tLS0tCk1JSUJlRENDQVIyZ0F3SUJBZ0lCQURBS0JnZ3Foa2
pPUFFRREFqQWpNU0V3SHdZRFZRUUREQmhyTTNNdFkyeHHAKWlc1MExXTmhRREUyT
kRFME16WTVNelV3SGhjTk1qSXdNVEEyTURJME1qRTFXaGNOTXpJd01UQTBNREkw
TWpFMQpXakFqTVNFd0h3WURWUVFEREJock0zTXRZMnhwWlc1MExXTmhRREUyTkR
FME16WTVNelV3V1RBVEJnY3Foa2pPClBRSUJCZ2dxaGtqT1BRTUJCd05DQUFTd0
c2dk9tay8vL01jNlUwU3BLZm9ERFM1NDNkQnZSdzVZUnNlZmpmWm0KT01BQUNRb
kViYS9QY0FGc2ZIUlBWWU9HczRnWTQ3TVlDbzF3L2swV3had3lvMEl3UURBT0Jn
TlZIUThCQWY4RQpCQU1DQXFRd0R3WURWUjBUQVFIL0JBVXdBd0VCL3pBZEJnTlZ
IUTRFRmdRVTE1REdRbTYxbE55UVpSNkNyNEFNCmROY3pBVTB3Q2dZSUtvWkl6aj
BFQXdJRFNRQXdSZ0loQUtQcjE3T0lDNk94a1hBYnpxUGl2R0QwZkptVjFmTnIKV
FNzc2IvMktWMjh4QWlFQTFEUVlHU2F0V3R6Y2tFdk1JNnYzeTcyQ2hwdDZWMHZU
dWNEWWJsOWxRVFU9Ci0tLS0tRU5EIENFUlRJRklDQVRFLS0tLS0K
    client-key-data:
LS0tLS1CRUdJTiBFQyBQUklWQVRFIEtFWS0tLS0tCk1IY0NBUUVFSUxjTWt1aW9
mTHo1Z1lUZGVrWmlsOEhTZVMzSXVONHVHUGU2VXFxRWJkN0dvQW9HQ0NxR1NNND
kKQXdFSG9VUURRZ0FFR2xKR3R3WFdoUUFBRVNlaUZqZGRsbU9IVWQzb3JqRXBaU
W1XeW5xZE8xTGNTWFlnSHFSRgo4ZWx5bUh0d0t4bEdxaFoyVTVMVzlDYmh6dFY3
bFhPRVhnPT0KLS0tLS1FTkQgRUMgUFJJVkFURSBLRVktLS0tLQo=
```

## 镜像管理

### 导入本地镜像

```
Netkiller-iMac:~ neo$ docker image ls | grep netkiller
netkiller
openjdk8        52e22fa28d43    3 weeks ago    552MB
```

将本地 netkiller:openjdk8 镜像导入到 mycluster 中

```
Netkiller-iMac:~ neo$ k3d image import netkiller:openjdk8 -c
mycluster
INFO[0000] Importing image(s) into cluster 'mycluster'
INFO[0000] Loading 1 image(s) from runtime into nodes...
INFO[0051] Importing images '[netkiller:openjdk8]' into node
'k3d-mycluster-server-0'...
INFO[0050] Importing images '[netkiller:openjdk8]' into node
'k3d-mycluster-server-2'...
INFO[0050] Importing images '[netkiller:openjdk8]' into node
'k3d-mycluster-agent-1'...
INFO[0050] Importing images '[netkiller:openjdk8]' into node
'k3d-mycluster-server-1'...
INFO[0050] Importing images '[netkiller:openjdk8]' into node
'k3d-mycluster-agent-0'...
INFO[0355] Successfully imported image(s)
INFO[0355] Successfully imported 1 image(s) into 1 cluster(s)
```

## 管理 k3d 集群

```
[root@netkiller k3d]# k3d cluster start mycluster
```

**配置 api-port 端口**

```
k3d cluster create netkiller --api-port 6443 --servers 1 --
agents 1 --port '80:80@loadbalancer' --port
'443:443@loadbalancer'
```

```
[root@netkiller ~]# cat .kube/config | grep server
    server: https://0.0.0.0:6445
```

```
[root@netkiller ~]# ss -lnt | grep 6445
LISTEN 0      1024         0.0.0.0:6445       0.0.0.0:*
```

```
[root@netkiller ~]# firewall-cmd --add-service=http --permanent
success
[root@netkiller ~]# firewall-cmd --add-service=https --
permanent
success
[root@netkiller ~]# firewall-cmd --zone=public --add-
service=kube-api --permanent
success
```

```
k3d cluster create netkiller --api-port 172.16.0.1:6443 --
servers 1 --agents 1 --port '80:80@loadbalancer' --port
'443:443@loadbalancer' --k3s-arg "--no-deploy=traefik@server:*"
```

export http_proxy="socks://127.0.0.1:1080" export
https_proxy="socks://127.0.0.1:1080"

**kubectl 管理指定集群**

```
export KUBECONFIG="$(k3d kubeconfig write netkiller)"
```

```
[root@netkiller ~]# kubectl config view
apiVersion: v1
clusters:
```

```
- cluster:
    certificate-authority-data: DATA+OMITTED
    server: https://172.18.200.10:6445
  name: k3d-netkiller
contexts:
- context:
    cluster: k3d-netkiller
    user: admin@k3d-netkiller
  name: k3d-netkiller
current-context: k3d-netkiller
kind: Config
preferences: {}
users:
- name: admin@k3d-netkiller
  user:
    client-certificate-data: REDACTED
    client-key-data: REDACTED
```

## 容器镜像库

```
neo@Netkiller-iMac ~> vim ~/.k3d/registries.yaml
mirrors:
  "registry.netkiller.cn":
    endpoint:
      - http://registry.netkiller.cn
```

```
neo@Netkiller-iMac ~> k3d cluster create mycluster --api-port
6443 --servers 1 --agents 1 --port '80:80@loadbalancer' --port
'443:443@loadbalancer' --registry-config ~/.k3d/registries.yaml
```

## traefik 配置

**增加 Redis 6379 端口**

```
neo@Netkiller-iMac ~> kubectl edit -n kube-system deployment
traefik
deployment.apps/traefik edited
```

```
    spec:
      containers:
      - args:
        - --global.checknewversion
        - --global.sendanonymoususage
        - --entrypoints.traefik.address=:9000/tcp
        - --entrypoints.web.address=:8000/tcp
        - --entrypoints.websecure.address=:8443/tcp
        - --entrypoints.redis.address=:6379/tcp
        - --entrypoints.mysql.address=:3306/tcp
        - --entrypoints.mongo.address=:27017/tcp
        - --api.dashboard=true
        - --ping=true
        - --providers.kubernetescrd
        - --providers.kubernetesingress
        - --
providers.kubernetesingress.ingressendpoint.publishedservice=ku
be-system/traefik
        - --entrypoints.websecure.http.tls=true
        image: rancher/library-traefik:2.4.8
        imagePullPolicy: IfNotPresent
        livenessProbe:
          failureThreshold: 3
          httpGet:
            path: /ping
            port: 9000
            scheme: HTTP
          initialDelaySeconds: 10
          periodSeconds: 10
          successThreshold: 1
          timeoutSeconds: 2
        name: traefik
        ports:
        - containerPort: 9000
          name: traefik
          protocol: TCP
        - containerPort: 8000
          name: web
          protocol: TCP
        - containerPort: 8443
          name: websecure
          protocol: TCP
        - containerPort: 6379
          name: redis
          protocol: TCP
        - containerPort: 3306
          name: mysql
          protocol: TCP
        - containerPort: 27017
          name: mongo
          protocol: TCP
```

args 处加入

```
- --entrypoints.redis.address=:6379/tcp
```

ports 处加入

```
              - containerPort: 6379
          name: redis
          protocol: TCP
```

```
[root@netkiller k3d]# k3d cluster edit mycluster --port-add
'6379:6379@loadbalancer'
```

```
[root@netkiller k3d]# cat redis.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: redis
spec:
  selector:
    matchLabels:
      app: redis
  template:
    metadata:
      labels:
        app: redis
    spec:
      containers:
      - name: redis
        image: redis:latest
        ports:
        - containerPort: 6379
          protocol: TCP
---
apiVersion: v1
kind: Service
metadata:
  name: redis
spec:
  ports:
  - port: 6379
    targetPort: 6379
  selector:
    app: redis
---
apiVersion: traefik.containo.us/v1alpha1
kind: IngressRouteTCP
metadata:
  name: redis
spec:
  entryPoints:
    - redis
  routes:
  - match: HostSNI(`*`)
    services:
    - name: redis
      port: 6379
```

```
[root@netkiller k3d]# kubectl apply -f redis.yaml
deployment.apps/redis created
service/redis created
ingressroutetcp.traefik.containo.us/redis created

[root@netkiller k3d]# kubectl get pods
NAME                     READY   STATUS    RESTARTS   AGE
redis-5c9986b94b-gsctv   1/1     Running   0          6m49s
[root@netkiller k3d]# kubectl exec redis-5c9986b94b-gsctv -it
-- redis-cli
127.0.0.1:6379> set nickname netkiller
OK
127.0.0.1:6379> get nickname
"netkiller"
127.0.0.1:6379>
127.0.0.1:6379> exit
```

```
[root@netkiller k3d]# dnf install redis
[root@netkiller k3d]# redis-cli -h 127.0.0.1
127.0.0.1:6379> get nickname
```

### ingress-nginx

#### 卸载 traefik

我们希望使用 nginx ingress，所以需要讲 traefik 卸载

```
kubectl -n kube-system delete helmcharts.helm.cattle.io traefik
helm uninstall traefik-crd --namespace kube-system
```

**安装 ingress-nginx**

ingress-nginx: https://kubernetes.github.io/ingress-nginx/deploy/

```
kubectl apply -f
https://raw.githubusercontent.com/kubernetes/ingress-
nginx/controller-
v1.3.0/deploy/static/provider/cloud/deploy.yaml
```

修改镜像库地址，否则无法下载

```
wget https://raw.githubusercontent.com/kubernetes/ingress-
nginx/controller-
v1.3.0/deploy/static/provider/cloud/deploy.yaml
vim deploy.yaml
:%s:registry.k8s.io/ingress-nginx/:registry.cn-
hangzhou.aliyuncs.com/google_containers/:g
:%s:registry.cn-
hangzhou.aliyuncs.com/google_containers/controller:registry.cn-
hangzhou.aliyuncs.com/google_containers/nginx-ingress-
controller:g
```

svclb-ingress-nginx-controller 启动不起来

```
neo@MacBook-Pro-Neo-3 ~ [1]> kubectl logs -n kube-system svclb-
ingress-nginx-controller-8b62cc7d-qbqtv
```

```
Defaulted container "lb-tcp-80" out of: lb-tcp-80, lb-tcp-443
+ trap exit TERM INT
+ echo 10.43.36.160
+ grep -Eq :
+ cat /proc/sys/net/ipv4/ip_forward
+ '[' 1 '!=' 1 ]
+ iptables -t nat -I PREROUTING '!' -s 10.43.36.160/32 -p TCP -
-dport 80 -j DNAT --to 10.43.36.160:80
iptables v1.8.4 (legacy): can't initialize iptables table
`nat': Table does not exist (do you need to insmod?)
Perhaps iptables or your kernel needs to be upgraded.
```

解决方法

```


root@netkiller ~ # modprobe ip_tables

root@netkiller ~# lsmod|grep iptable
iptable_nat            16384  2
ip_tables              28672  1 iptable_nat
nf_nat                 53248  4
xt_nat,nft_chain_nat,iptable_nat,xt_MASQUERADE

root@netkiller ~# kubectl get pods --all-namespaces
NAMESPACE       NAME
READY   STATUS      RESTARTS         AGE
ingress-nginx   ingress-nginx-admission-create-nqv2f
0/1     Completed   0                6m9s
ingress-nginx   ingress-nginx-admission-patch-m9hcf
0/1     Completed   1                6m9s
kube-system     metrics-server-7cd5fcb6b7-8wrqx
1/1     Running     3 (6m30s ago)    82m
ingress-nginx   ingress-nginx-controller-75d55647d-nstch
1/1     Running     0                6m9s
kube-system     coredns-d76bd69b-rgvwj
1/1     Running     3 (6m21s ago)    82m
kube-system     local-path-provisioner-6c79684f77-psmgs
1/1     Running     3 (6m21s ago)    82m
kube-system     svclb-ingress-nginx-controller-8b62cc7d-5lb8d
2/2     Running     12 (3m17s ago)   6m9s
kube-system     svclb-ingress-nginx-controller-8b62cc7d-qbqtv
```

```
2/2     Running     12 (3m20s ago)   6m9s
```

验证安装是否正确

部署 Nginx Web 服务器，用来检查 ingress

```
Neo-iMac:~ neo$ kubectl create deployment nginx --
image=nginx:alpine
deployment.apps/nginx created

Neo-iMac:~ neo$ kubectl create service clusterip nginx --
tcp=80:80
service/nginx created
```

```
cat <<EOF | kubectl apply -f -
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: nginx
  annotations:
    kubernetes.io/ingress.class: nginx
    ingress.kubernetes.io/ssl-redirect: "false"
spec:
  rules:
  - http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: nginx
            port:
              number: 80
EOF
```

```
```

## 5.5. TLS 证书

```
[root@master ~]# ll /var/lib/rancher/k3s/server/tls
total 116
-rw-r--r-- 1 root root 1173 2022-09-08 13:48 client-admin.crt
-rw------- 1 root root  227 2022-09-08 13:48 client-admin.key
-rw-r--r-- 1 root root 1178 2022-09-08 13:48 client-auth-
proxy.crt
-rw------- 1 root root  227 2022-09-08 13:48 client-auth-
proxy.key
-rw-r--r-- 1 root root  570 2022-09-08 13:48 client-ca.crt
-rw------- 1 root root  227 2022-09-08 13:48 client-ca.key
-rw-r--r-- 1 root root 1165 2022-09-08 13:48 client-
controller.crt
-rw------- 1 root root  227 2022-09-08 13:48 client-
controller.key
-rw-r--r-- 1 root root 1161 2022-09-08 13:48 client-k3s-cloud-
controller.crt
-rw------- 1 root root  227 2022-09-08 13:48 client-k3s-cloud-
controller.key
-rw-r--r-- 1 root root 1153 2022-09-08 13:48 client-k3s-
controller.crt
-rw------- 1 root root  227 2022-09-08 13:48 client-k3s-
controller.key
-rw-r--r-- 1 root root 1181 2022-09-08 13:48 client-kube-
apiserver.crt
-rw------- 1 root root  227 2022-09-08 13:48 client-kube-
apiserver.key
-rw-r--r-- 1 root root 1149 2022-09-08 13:48 client-kube-
proxy.crt
-rw------- 1 root root  227 2022-09-08 13:48 client-kube-
proxy.key
-rw------- 1 root root  227 2022-09-08 13:48 client-kubelet.key
-rw-r--r-- 1 root root 1153 2022-09-08 13:48 client-
scheduler.crt
-rw------- 1 root root  227 2022-09-08 13:48 client-
scheduler.key
-rw-r--r-- 1 root root 3789 2022-09-08 13:48 dynamic-cert.json
drwxr-xr-x 2 root root 4096 2022-09-08 13:48 etcd
```

```
-rw-r--r-- 1 root root  591 2022-09-08 13:48 request-header-
ca.crt
-rw------- 1 root root  227 2022-09-08 13:48 request-header-
ca.key
-rw-r--r-- 1 root root  570 2022-09-08 13:48 server-ca.crt
-rw------- 1 root root  227 2022-09-08 13:48 server-ca.key
-rw------- 1 root root 1675 2022-09-08 13:48 service.key
-rw-r--r-- 1 root root 1368 2022-09-08 13:48 serving-kube-
apiserver.crt
-rw------- 1 root root  227 2022-09-08 13:48 serving-kube-
apiserver.key
-rw------- 1 root root  227 2022-09-08 13:48 serving-
kubelet.key
drwx------ 2 root root   84 2022-09-08 13:48 temporary-certs
```

## 5.6. 创建 Token

```
[root@master ~]#kubectl create serviceaccount secrets
serviceaccount/gitlab created

[root@master ~]# kubectl create token secrets
eyJhbGciOiJSUzI1NiIsImtpZCI6IktCOHRvYlZOLXFPRmEyb1JWdlQxSzBvN0t
vZF9HNFBGRnlraDR5UU1jakkifQ.eyJhdWQiOlsiaHR0cHM6Ly9rdWJlcm5ldGV
zLmRlZmF1bHQuc3ZjLmNsdXN0ZXIubG9jYWwiLCJrM3MiXSwiZXhwIjoxNjY2MT
cyOTc4LCJpYXQiOjE2NjYxNjkzNzgsImlzcyI6Imh0dHBzOi8va3ViZXJuZXRlc
y5kZWZhdWx0LnN2Yy5jbHVzdGVyLmxvY2FsIiwia3ViZXJuZXRlcy5pbyI6eyJu
YW1lc3BhY2UiOiJkZWZhdWx0Iiwic2VydmljZWFjY291bnQiOnsibmFtZSI6Imd
pdGxhYiIsInVpZCI6IjAzNTdkOWIwLWY2YWEtNGFlMy05MDc0LWM2YzM5Y2Q1YT
diNiJ9fSwibmJmIjoxNjY2MTY5Mzc4LCJzdWIiOiJzeXN0ZW06c2VydmljZWFjY
291bnQ6ZGVmYXVsdDpnaXRsYWIifQ.oDWjQmVH7BOqHUp4AgjxNncfJ0Nz9oY_j
S9DU5E-geKmX5GnchC96-t0ZsdtgPiWXFbieb0aUH1wZXCrkFAuGeM-
XNDEvfhbK4UL9GiDl98KaYMjTSwXipp4bIZeSctL-
Zpc0nSKwaWdWNwxmmlC30HwMwjQPdwBgCDM8SEr9aepUuJD9rHdclKWv8NcXlLq
4t5c9sV3qEQRKbGOTnSeY3RokoAY-
tYD7FT3jzFktbkTk4SHZAKYUeILlc2eaE0cOm9N4yhl8IYZvEcrBGZV_-
Nl0XzGu5XpDrVVXlk2k2RdYQHj3Iw5l4sSFfnRVg1Q-1B45y7FJDEbXa-
tCXeRKA

[root@master ~]#
token=eyJhbGciOiJSUzI1NiIsImtpZCI6IktCOHRvYlZOLXFPRmEyb1JWdlQxS
```

```
zBvN0tvZF9HNFBGRnlraDR5UU1jakkifQ.eyJhdWQiOlsiaHR0cHM6Ly9rdWJlc
m5ldGVzLmRlZmF1bHQuc3ZjLmNsdXN0ZXIubG9jYWwiLCJrM3MiXSwiZXhwIjox
NjY2MTcyOTc4LCJpYXQiOjE2NjYxNjkzNzgsImlzcyI6Imh0dHBzOi8va3ViZXJ
uZXRlcy5kZWZhdWx0LnN2Yy5jbHVzdGVyLmxvY2FsIiwia3ViZXJuZXRlcy5pby
I6eyJuYW1lc3BhY2UiOiJkZWZhdWx0Iiwic2VydmljZWFjY291bnQiOnsibmFtZ
SI6ImdpdGxhYiIsInVpZCI6IjAzNTdkOWIwLWY2YWEtNGFlMy05MDc0LWM2YzM5
Y2Q1YTdiNiJ9fSwibmJmIjoxNjY2MTY5Mzc4LCJzdWIiOiJzeXN0ZW06c2Vydml
jZWFjY291bnQ6ZGVmYXVsdDpnaXRsYWIifQ.oDWjQmVH7BOqHUp4AgjxNncfJ0N
z9oY_jS9DU5E-geKmX5GnchC96-t0ZsdtgPiWXFbieb0aUH1wZXCrkFAuGeM-
XNDEvfhbK4UL9GiDl98KaYMjTSwXipp4bIZeSctL-
Zpc0nSKwaWdWNwxmmlC30HwMwjQPdwBgCDM8SEr9aepUuJD9rHdclKWv8NcXlLq
4t5c9sV3qEQRKbGOTnSeY3RokoAY-
tYD7FT3jzFktbkTk4SHZAKYUeILlc2eaE0cOm9N4yhl8IYZvEcrBGZV_-
Nl0XzGu5XpDrVVXlk2k2RdYQHj3Iw5l4sSFfnRVg1Q-1B45y7FJDEbXa-
tCXeRKA
[root@master ~]# curl -k https://127.0.0.1:6443/api --header
"Authorization: bearer $token"
{
  "kind": "APIVersions",
  "versions": [
    "v1"
  ],
  "serverAddressByClientCIDRs": [
    {
      "clientCIDR": "0.0.0.0/0",
      "serverAddress": "172.18.200.5:6443"
    }
  ]
}
```

## 5.7. FAQ

### ghcr.io 镜像下载问题

创建集群始终停止在这里，这是因为 ghcr.io 被墙，无法访问。

```
INFO[0004] Pulling image 'ghcr.io/k3d-io/k3d-proxy:5.4.4'
```

```

```

找一台境外VPS安装K3D并创建集群，然后讲 k3d-proxy 镜像保存为文件。

```
[docker@netkiller ~]$ docker images
REPOSITORY                  TAG             IMAGE ID
CREATED         SIZE
ghcr.io/k3d-io/k3d-proxy    5.4.4           5a963719cb39    2
weeks ago     42.4MB
ghcr.io/k3d-io/k3d-tools    5.4.4           741f01cb5093    2
weeks ago     18.7MB

[docker@netkiller ~]$ docker save 5a963719cb39 -o k3d-proxy.tar
```

复制到国内，导入镜像

```
docker load --input k3d-proxy.tar
```

**k3s 80/443 端口问题**

```
[root@master ~]# kubectl get svc --namespace=kube_system
NAME                 TYPE            CLUSTER-IP      EXTERNAL-IP
PORT(S)                         AGE
kube-dns             ClusterIP       10.43.0.10      <none>
53/UDP,53/TCP,9153/TCP          4d2h
metrics-server       ClusterIP       10.43.88.112    <none>
443/TCP                         4d2h
traefik              LoadBalancer    10.43.125.52
172.18.200.5,172.18.200.51    80:32623/TCP,443:31516/TCP    4d2h
```

本地没有 80 和 443 端口

```
[root@master ~]# ss -tnlp | egrep "80|443"
LISTEN 0      1024               *:6443            *:*
users:(("k3s-server",pid=173779,fd=17))

[root@master ~]# lsof -i :80
[root@master ~]# lsof -i :443
```

telnet 测试后可工作

```
[root@master ~]# telnet 172.18.200.5 80
Trying 172.18.200.5...
Connected to 172.18.200.5.
Escape character is '^]'.
```

80/443 是 Iptable NAT映射出来的端口

```
[root@master ~]# iptables -nL -t nat | grep traefik
# Warning: iptables-legacy tables present, use iptables-legacy
to see them
KUBE-MARK-MASQ  all  --  0.0.0.0/0            0.0.0.0/0
/* masquerade traffic for kube-system/traefik:websecure
external destinations */
KUBE-MARK-MASQ  all  --  0.0.0.0/0            0.0.0.0/0
/* masquerade traffic for kube-system/traefik:web external
destinations */
KUBE-EXT-CVG3OEGEH7H5P3HQ  tcp  --  0.0.0.0/0
0.0.0.0/0            /* kube-system/traefik:websecure */ tcp
dpt:31516
```

```
KUBE-EXT-UQMCRMJZLI3FTLDP  tcp  --  0.0.0.0/0
0.0.0.0/0            /* kube-system/traefik:web */ tcp
dpt:32623
KUBE-MARK-MASQ  all  --  10.42.2.3            0.0.0.0/0
/* kube-system/traefik:web */
DNAT       tcp  --  0.0.0.0/0            0.0.0.0/0
/* kube-system/traefik:web */ tcp to:10.42.2.3:8000
KUBE-MARK-MASQ  all  --  10.42.2.3            0.0.0.0/0
/* kube-system/traefik:websecure */
DNAT       tcp  --  0.0.0.0/0            0.0.0.0/0
/* kube-system/traefik:websecure */ tcp to:10.42.2.3:8443
KUBE-SVC-CVG3OEGEH7H5P3HQ  tcp  --  0.0.0.0/0
10.43.125.52         /* kube-system/traefik:websecure cluster
IP */ tcp dpt:443
KUBE-EXT-CVG3OEGEH7H5P3HQ  tcp  --  0.0.0.0/0
172.18.200.5         /* kube-system/traefik:websecure
loadbalancer IP */ tcp dpt:443
KUBE-EXT-CVG3OEGEH7H5P3HQ  tcp  --  0.0.0.0/0
172.18.200.51        /* kube-system/traefik:websecure
loadbalancer IP */ tcp dpt:443
KUBE-SVC-UQMCRMJZLI3FTLDP  tcp  --  0.0.0.0/0
10.43.125.52         /* kube-system/traefik:web cluster IP */
tcp dpt:80
KUBE-EXT-UQMCRMJZLI3FTLDP  tcp  --  0.0.0.0/0
172.18.200.5         /* kube-system/traefik:web loadbalancer IP
*/ tcp dpt:80
KUBE-EXT-UQMCRMJZLI3FTLDP  tcp  --  0.0.0.0/0
172.18.200.51        /* kube-system/traefik:web loadbalancer IP
*/ tcp dpt:80
KUBE-MARK-MASQ  tcp  -- !10.42.0.0/16         10.43.125.52
/* kube-system/traefik:websecure cluster IP */ tcp dpt:443
KUBE-SEP-NTYW4CRSJDKN6UYK  all  --  0.0.0.0/0
0.0.0.0/0            /* kube-system/traefik:websecure ->
10.42.2.3:8443 */
KUBE-MARK-MASQ  tcp  -- !10.42.0.0/16         10.43.125.52
/* kube-system/traefik:web cluster IP */ tcp dpt:80
KUBE-SEP-M4A3OJBNTWBZ5ISS  all  --  0.0.0.0/0
0.0.0.0/0            /* kube-system/traefik:web ->
10.42.2.3:8000 */
```

NAT 端口可以通过 nmap 扫描出来

```
[root@master ~]# nmap localhost
Starting Nmap 7.91 ( https://nmap.org ) at 2022-09-01 10:04 CST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.0000050s latency).
Other addresses for localhost (not scanned): ::1
Not shown: 996 closed ports
PORT      STATE    SERVICE
22/tcp    open     ssh
80/tcp    filtered http
443/tcp   filtered https
10010/tcp open     rxapi

Nmap done: 1 IP address (1 host up) scanned in 1.26 seconds
```

```
[root@master ~]# iptables-save | grep "CNI-DN" | grep "to-
destination"
# Warning: iptables-legacy tables present, use iptables-legacy-
save to see them
-A CNI-DN-485265bef43fea7142e9d -p tcp -m tcp --dport 80 -j
DNAT --to-destination 10.42.0.10:80
-A CNI-DN-485265bef43fea7142e9d -p tcp -m tcp --dport 443 -j
DNAT --to-destination 10.42.0.10:443
```

### flannel 不通

```
[root@netkiller ~]# systemctl disable firewalld
Removed /etc/systemd/system/multi-
user.target.wants/firewalld.service.
Removed /etc/systemd/system/dbus-
org.fedoraproject.FirewallD1.service.

[root@master ~]# ifconfig
br-6ac52d42db64: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>
```

```
mtu 1500
        inet 172.20.0.1  netmask 255.255.0.0  broadcast
172.20.255.255
        inet6 fe80::42:94ff:fefd:1fc3  prefixlen 64  scopeid
0x20<link>
        ether 02:42:94:fd:1f:c3  txqueuelen 0  (Ethernet)
        RX packets 782783  bytes 200925233 (191.6 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 625170  bytes 194933933 (185.9 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0
collisions 0

cni0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1450
        inet 10.42.0.1  netmask 255.255.255.0  broadcast
10.42.0.255
        inet6 fe80::6448:6dff:fe75:5e8d  prefixlen 64  scopeid
0x20<link>
        ether 66:48:6d:75:5e:8d  txqueuelen 1000  (Ethernet)
        RX packets 2049669  bytes 371281787 (354.0 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2235678  bytes 334579428 (319.0 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0
collisions 0

docker0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 172.16.0.1  netmask 255.255.255.0  broadcast
172.16.0.255
        inet6 fe80::42:4cff:fe70:883  prefixlen 64  scopeid
0x20<link>
        ether 02:42:4c:70:08:83  txqueuelen 0  (Ethernet)
        RX packets 14  bytes 616 (616.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 8  bytes 788 (788.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0
collisions 0

enp3s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.18.200.5  netmask 255.255.255.0  broadcast
172.18.200.255
        inet6 fe80::2ef0:5dff:fec7:387  prefixlen 64  scopeid
0x20<link>
        ether 2c:f0:5d:c7:03:87  txqueuelen 1000  (Ethernet)
        RX packets 782783  bytes 200925233 (191.6 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 625171  bytes 194934547 (185.9 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0
collisions 0

flannel.1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1450
        inet 10.42.0.0  netmask 255.255.255.255  broadcast
0.0.0.0
        inet6 fe80::c051:5cff:fe09:4e18  prefixlen 64  scopeid
0x20<link>
        ether c2:51:5c:09:4e:18  txqueuelen 0  (Ethernet)
        RX packets 180007  bytes 21310049 (20.3 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 222507  bytes 39026179 (37.2 MiB)
        TX errors 0  dropped 5 overruns 0  carrier 0
collisions 0


[root@master ~]# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric
Ref    Use Iface
0.0.0.0         172.18.200.254  0.0.0.0         UG    100    0
0 enp3s0
10.42.0.0       0.0.0.0         255.255.255.0   U     0      0
0 cni0
10.42.1.0       10.42.1.0       255.255.255.0   UG    0      0
0 flannel.1
172.16.0.0      0.0.0.0         255.255.255.0   U     0      0
0 docker0
172.18.200.0    0.0.0.0         255.255.255.0   U     100    0
0 enp3s0
172.20.0.0      0.0.0.0         255.255.0.0     U     0      0
0 br-6ac52d42db64

[root@master ~]# cat /proc/sys/net/ipv4/ip_forward
1
[root@master ~]# sysctl net.ipv4.ip_forward
net.ipv4.ip_forward = 1

[root@master ~]# kubectl get pods -o wide
NAME                         READY   STATUS    RESTARTS        AGE
IP            NODE      NOMINATED NODE   READINESS GATES
nacos-1                      1/1     Running   5 (12h ago)     35h
10.42.0.50    master    <none>           <none>
elasticsearch-data-1         1/1     Running   5 (12h ago)     35h
10.42.0.44    master    <none>           <none>
```

```
nacos-2                  1/1     Running   7 (6m39s ago)   35h
10.42.1.49   agent-1   <none>           <none>
nacos-0                  1/1     Running   7 (6m32s ago)   35h
10.42.1.50   agent-1   <none>           <none>
elasticsearch-master-0   1/1     Running   6 (6m32s ago)   35h
10.42.1.47   agent-1   <none>           <none>
busybox                  0/1     Error     0               11h
10.42.1.46   agent-1   <none>           <none>
elasticsearch-data-2     1/1     Running   6 (6m32s ago)   35h
10.42.1.48   agent-1   <none>           <none>
elasticsearch-data-0     1/1     Running   6 (6m32s ago)   35h
10.42.1.51   agent-1   <none>           <none>
[root@master ~]# ping 10.42.0.50
PING 10.42.0.50 (10.42.0.50) 56(84) bytes of data.
64 bytes from 10.42.0.50: icmp_seq=1 ttl=64 time=0.039 ms
64 bytes from 10.42.0.50: icmp_seq=2 ttl=64 time=0.031 ms
64 bytes from 10.42.0.50: icmp_seq=3 ttl=64 time=0.042 ms
64 bytes from 10.42.0.50: icmp_seq=4 ttl=64 time=0.038 ms
^C
--- 10.42.0.50 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3054ms
rtt min/avg/max/mdev = 0.031/0.037/0.042/0.004 ms

[root@master ~]# kubectl get pods -o wide
NAME                     READY   STATUS    RESTARTS      AGE
IP           NODE      NOMINATED NODE   READINESS GATES
nacos-1                  1/1     Running   5 (12h ago)   35h
10.42.0.50   master    <none>           <none>
elasticsearch-data-1     1/1     Running   5 (12h ago)   35h
10.42.0.44   master    <none>           <none>
nacos-2                  1/1     Running   7 (29m ago)   35h
10.42.1.49   agent-1   <none>           <none>
nacos-0                  1/1     Running   7 (29m ago)   35h
10.42.1.50   agent-1   <none>           <none>
elasticsearch-master-0   1/1     Running   6 (29m ago)   35h
10.42.1.47   agent-1   <none>           <none>
busybox                  0/1     Error     0             11h
10.42.1.46   agent-1   <none>           <none>
elasticsearch-data-2     1/1     Running   6 (29m ago)   35h
10.42.1.48   agent-1   <none>           <none>
elasticsearch-data-0     1/1     Running   6 (29m ago)   35h
10.42.1.51   agent-1   <none>           <none>
[root@master ~]# ping 10.42.1.51 -c 5
PING 10.42.1.51 (10.42.1.51) 56(84) bytes of data.
64 bytes from 10.42.1.51: icmp_seq=1 ttl=63 time=0.402 ms
```

```
64 bytes from 10.42.1.51: icmp_seq=2 ttl=63 time=0.171 ms
64 bytes from 10.42.1.51: icmp_seq=3 ttl=63 time=0.170 ms
64 bytes from 10.42.1.51: icmp_seq=4 ttl=63 time=0.410 ms
64 bytes from 10.42.1.51: icmp_seq=5 ttl=63 time=0.414 ms

--- 10.42.1.51 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4105ms
rtt min/avg/max/mdev = 0.170/0.313/0.414/0.116 ms

[root@agent-1 ~]# ping 10.42.0.50 -c 5
PING 10.42.0.50 (10.42.0.50) 56(84) bytes of data.
64 bytes from 10.42.0.50: icmp_seq=1 ttl=63 time=0.154 ms
64 bytes from 10.42.0.50: icmp_seq=2 ttl=63 time=0.206 ms
64 bytes from 10.42.0.50: icmp_seq=3 ttl=63 time=0.213 ms
64 bytes from 10.42.0.50: icmp_seq=4 ttl=63 time=0.218 ms
64 bytes from 10.42.0.50: icmp_seq=5 ttl=63 time=0.220 ms

--- 10.42.0.50 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4125ms
rtt min/avg/max/mdev = 0.154/0.202/0.220/0.024 ms

[root@master ~]# kubectl exec -it nacos-1 -- ping nacos-
0.nacos.default.svc.cluster.local -c 5
PING nacos-0.nacos.default.svc.cluster.local (10.42.1.50)
56(84) bytes of data.
64 bytes from nacos-0.nacos.default.svc.cluster.local
(10.42.1.50): icmp_seq=1 ttl=62 time=0.440 ms
64 bytes from nacos-0.nacos.default.svc.cluster.local
(10.42.1.50): icmp_seq=2 ttl=62 time=0.429 ms
64 bytes from nacos-0.nacos.default.svc.cluster.local
(10.42.1.50): icmp_seq=3 ttl=62 time=0.431 ms
64 bytes from nacos-0.nacos.default.svc.cluster.local
(10.42.1.50): icmp_seq=4 ttl=62 time=0.343 ms
64 bytes from nacos-0.nacos.default.svc.cluster.local
(10.42.1.50): icmp_seq=5 ttl=62 time=0.229 ms

--- nacos-0.nacos.default.svc.cluster.local ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4127ms
rtt min/avg/max/mdev = 0.229/0.374/0.440/0.082 ms
[root@master ~]# kubectl exec -it nacos-2 -- ping nacos-
0.nacos.default.svc.cluster.local -c 5
PING nacos-0.nacos.default.svc.cluster.local (10.42.1.50)
56(84) bytes of data.
64 bytes from nacos-0.nacos.default.svc.cluster.local
(10.42.1.50): icmp_seq=1 ttl=64 time=0.053 ms
```

```
64 bytes from nacos-0.nacos.default.svc.cluster.local
(10.42.1.50): icmp_seq=2 ttl=64 time=0.039 ms
64 bytes from nacos-0.nacos.default.svc.cluster.local
(10.42.1.50): icmp_seq=3 ttl=64 time=0.038 ms
64 bytes from nacos-0.nacos.default.svc.cluster.local
(10.42.1.50): icmp_seq=4 ttl=64 time=0.077 ms
64 bytes from nacos-0.nacos.default.svc.cluster.local
(10.42.1.50): icmp_seq=5 ttl=64 time=0.039 ms

--- nacos-0.nacos.default.svc.cluster.local ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4113ms
rtt min/avg/max/mdev = 0.038/0.049/0.077/0.015 ms

[root@master ~]# kubectl delete pod busybox
[root@master ~]# kubectl run -i --tty busybox --image=busybox -
-restart=Never
If you don't see a command prompt, try pressing enter.
/ # ping nacos-0.nacos.default.svc.cluster.local -c 3
PING nacos-0.nacos.default.svc.cluster.local (10.42.1.50): 56
data bytes
64 bytes from 10.42.1.50: seq=0 ttl=64 time=0.052 ms
64 bytes from 10.42.1.50: seq=1 ttl=64 time=0.049 ms
64 bytes from 10.42.1.50: seq=2 ttl=64 time=0.047 ms

--- nacos-0.nacos.default.svc.cluster.local ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 0.047/0.049/0.052 ms
/ #
```

### Failed to allocate directory watch: Too many open files

```
[root@netkiller ~]# ulimit -a
real-time non-blocking time  (microseconds, -R) unlimited
core file size              (blocks, -c) 0
data seg size               (kbytes, -d) unlimited
scheduling priority                 (-e) 0
file size                   (blocks, -f) unlimited
pending signals                     (-i) 254690
```

```
max locked memory           (kbytes, -l) 64
max memory size             (kbytes, -m) unlimited
open files                          (-n) 6553500
pipe size                (512 bytes, -p) 8
POSIX message queues         (bytes, -q) 819200
real-time priority                  (-r) 0
stack size                  (kbytes, -s) 8192
cpu time                   (seconds, -t) unlimited
max user processes                  (-u) 254690
virtual memory              (kbytes, -v) unlimited
file locks                          (-x) unlimited
```

```
[root@netkiller ~]# sysctl fs.inotify.max_user_instances
fs.inotify.max_user_watches
fs.inotify.max_user_instances = 128
fs.inotify.max_user_watches = 508881

[root@netkiller ~]# sysctl -w
fs.inotify.max_user_watches=5088800
fs.inotify.max_user_watches = 5088800

[root@netkiller ~]# sysctl -w
fs.inotify.max_user_instances=4096
fs.inotify.max_user_instances = 4096
```

# 6. Rancher Demo

## 6.1. Rancher 部署 Nginx

准备编排脚本

```
[root@localhost ~]# cat nginx.yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx
  labels:
    app: nginx
spec:
  ports:
  - port: 88
    targetPort: 80
  selector:
    app: nginx
  type: NodePort
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
```

```
        image: nginx:latest
        ports:
        - containerPort: 80
```

部署

```
[root@localhost ~]# rancher kubectl create -f nginx.yaml
service/nginx created
deployment.apps/nginx created
```

查看状态

```
[root@localhost ~]# rancher kubectl get deployment -n default
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
nginx   3/3     3            3           113s

[root@localhost ~]# rancher kubectl get service -n default
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)
AGE
kubernetes   ClusterIP   10.43.0.1       <none>        443/TCP
156m
nginx        NodePort    10.43.111.205   <none>
88:32646/TCP   119s

[root@localhost ~]# rancher kubectl get pods -n default
NAME                      READY   STATUS              RESTARTS
AGE
nginx-585449566-kd2mk     0/1     ContainerCreating   0
14s
nginx-585449566-mdl8n     0/1     ContainerCreating   0
14s
nginx-585449566-v8s5k     0/1     ContainerCreating   0
14s
```

```
[root@localhost ~]# rancher kubectl describe services nginx
Name:                     nginx
Namespace:                default
Labels:                   app=nginx
Annotations:              field.cattle.io/publicEndpoints:
[{"port":32646,"protocol":"TCP","serviceName":"default:nginx","
allNodes":true}]
Selector:                 app=nginx
Type:                     NodePort
IP Family Policy:         SingleStack
IP Families:              IPv4
IP:                       10.43.111.205
IPs:                      10.43.111.205
Port:                     <unset>  88/TCP
TargetPort:               80/TCP
NodePort:                 <unset>  32646/TCP
Endpoints:
10.42.0.40:80,10.42.0.41:80,10.42.0.42:80
Session Affinity:         None
External Traffic Policy:  Cluster
Events:                   <none>
```

## 6.2. local-path-provisioner

https://github.com/rancher/local-path-provisioner

local-path 即 pod 销毁之后，数据仍然存储在磁盘上，实验过程：

```
kubectl create -f
https://raw.githubusercontent.com/rancher/local-path-
provisioner/master/examples/pvc/pvc.yaml
kubectl create -f
https://raw.githubusercontent.com/rancher/local-path-
provisioner/master/examples/pod/pod.yaml
```

```
kubectl exec volume-test -- sh -c "echo local-path-test >
/data/test"
```

```
kubectl delete -f
https://raw.githubusercontent.com/rancher/local-path-
provisioner/master/examples/pod/pod.yaml
kubectl create -f
https://raw.githubusercontent.com/rancher/local-path-
provisioner/master/examples/pod/pod.yaml
```

```
$ kubectl exec volume-test -- sh -c "cat /data/test"
local-path-test
```

```
kubectl delete -f
https://raw.githubusercontent.com/rancher/local-path-
provisioner/master/examples/pod/pod.yaml
kubectl delete -f
https://raw.githubusercontent.com/rancher/local-path-
provisioner/master/examples/pvc/pvc.yaml
```

# 7. Longhorn

https://longhorn.io/docs/

## 7.1. 安装 Longhorn

```
[root@master ~]# dnf install -y jq
[root@master ~]# dnf install -y iscsi-initiator-utils

kubectl apply -f
https://raw.githubusercontent.com/longhorn/longhorn/v1.3.1/depl
oy/longhorn.yaml
```

检查环境

```
[root@master ~]# curl -sSfL
https://raw.githubusercontent.com/longhorn/longhorn/v1.3.1/scri
pts/environment_check.sh | bash
[INFO]  Required dependencies are installed.
[INFO]  Waiting for longhorn-environment-check pods to become
ready (0/3)...
[INFO]  All longhorn-environment-check pods are ready (3/3).
[ERROR] nfs-utils is not found in agent-2.
[ERROR] nfs-utils is not found in agent-1.
[ERROR] nfs-utils is not found in master.
[ERROR] Please install missing packages.
[INFO]  Cleaning up longhorn-environment-check pods...
[INFO]  Cleanup completed.
```

由于我不需要 NFS 所以没有安装 nfs-utils

## 7.2. 选择磁盘类型

首先要给磁盘打上标签，才能使用这个功能

```
[root@master ~]# lsblk
NAME         MAJ:MIN RM   SIZE RO TYPE MOUNTPOINTS
sda            8:0    0 931.5G  0 disk
`-sda1         8:1    0 931.5G  0 part /opt
nvme0n1      259:0    0 238.5G  0 disk
|-nvme0n1p1 259:1    0   600M  0 part /boot/efi
|-nvme0n1p2 259:2    0     1G  0 part /boot
|-nvme0n1p3 259:3    0    64G  0 part [SWAP]
`-nvme0n1p4 259:4    0 172.9G  0 part /

[root@master ~]# ls /opt/longhorn/
longhorn-disk.cfg  replicas
```

/opt/longhorn/ 被打了 HDD 标签

```
allowVolumeExpansion: true
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  annotations:
    field.cattle.io/description: 硬盘存储
  name: longhorn-storage
parameters:
  diskSelector: hdd
  numberOfReplicas: "3"
  staleReplicaTimeout: "2880"
provisioner: driver.longhorn.io
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

选择多个标签 diskSelector: "ssd,fast"

## 7.3. 节点选择

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: longhorn
provisioner: driver.longhorn.io
allowVolumeExpansion: true
parameters:
  numberOfReplicas: "2"
  staleReplicaTimeout: "2880"
  fromBackup: ""
#  diskSelector: "ssd,fast"
  nodeSelector: "storage,fast"
#  recurringJobs: '[{"name":"snap", "task":"snapshot",
"cron":"*/1 * * * *", "retain":1},
#                   {"name":"backup", "task":"backup",
"cron":"*/2 * * * *", "retain":1,
#                    "labels": {"interval":"2m"}}]'
```

## 7.4. FAQ

### FailedAttachVolume

```
Type    Reason  Updated         Message
Warning FailedAttachVolume      8 hours ago
AttachVolume.Attach failed for volume "pvc-03796772-abeb-4042-
8e5e-63a9b21da0f7" : rpc error: code = DeadlineExceeded desc =
volume pvc-03796772-abeb-4042-8e5e-63a9b21da0f7 failed to
attach to node master
```

# 8. FAQ

## 8.1. 调试 Rancher 查看日志

```
neo@ubuntu:~$ docker logs -f rancher
```

```
$ curl -L http://127.0.0.1:2379/health

{"health": "true"}
```

## 8.2. [network] Host [rancher.netkiller.cn] is not able to connect to the following ports: [rancher.netkiller.cn:2379]. Please check network policies and firewall rules

提示错误

[network] Host [rancher.netkiller.cn] is not able to connect to the following ports: [rancher.netkiller.cn:2379]. Please check network policies and firewall rules

排查

```
$ docker logs -f share-mnt

Error response from daemon: {"message":"No such container: kubelet"}
Error: failed to start containers: kubelet
```

```
neo@m-1d41c853af58:~$ snap list
Name      Version          Rev    Tracking   Publisher   Notes
core      16-2.37.4        6531   stable     canonical✓  core
go        1.12             3318   stable     mwhudson    classic
kubectl   1.13.4           780    stable     canonical✓  classic
lxd       3.11             10343  stable/…   canonical✓  -
microk8s  v1.14.0-beta.1   442    1.14/beta  canonical✓  classic

neo@m-1d41c853af58:~$ snap remove microk8s kubectl lxd
error: access denied (try with sudo)

neo@m-1d41c853af58:~$ sudo snap remove microk8s kubectl lxd
sudo: unable to resolve host m-1d41c853af58: Invalid argument
microk8s removed
kubectl removed
lxd removed
```

## 8.3. cgroups v2

```
检查操作系统是否支持 cgroups v2

grep cgroup2 /proc/filesystems

启用 cgroups v2 内核参数

systemd.unified_cgroup_hierarchy=1

回到 cgroups v1

sudo grubby --update-kernel=ALL --
args="systemd.unified_cgroup_hierarchy=0"
```

# 第 11 章 netkiller 容器编排工具

## 1. 安装 netkiller-devops

```
pip3 install netkiller-devops
```

## 2. 使用 python 优雅地编排 Docker 容器

用 Python 替代 docker compose 编排容器

docker compose 是 docker 的容器编排工具，它是基于 YAML 配置，YAML 是一种配置文件格式，支持传递环境变量，但是对于复杂的容器编排显得力不从心。

于是我便开发这个程序，可以像写程序一样编排 docker ，可以充分发挥程序猿的想象力。

```
pip install netkiller-devops
```

快速入门，首先我们参照这个 docker-compose.yaml 脚本，转换成 python 脚本。

```
version: '3.9'
services:
  nginx:
    container_name: nginx
    environment:
    - TZ=Asia/Shanghai
    extra_hosts:
    - db.netkiller.cn:127.0.0.1
    - cache.netkiller.cn:127.0.0.1
    - api.netkiller.cn:127.0.0.1
    hostname: www.netkiller.cn
    image: nginx:latest
    ports:
    - 80:80
    - 443:443
    restart: always
    volumes:
    - /tmp:/tmp
```

```
```

转换成 python 语言之后

```


from netkiller.docker import *

service =  Services('nginx')
service.image('nginx:latest')
service.container_name('nginx')
service.restart('always')
service.hostname('www.netkiller.cn')
service.extra_hosts(['db.netkiller.cn:127.0.0.1','cache.netki
ller.cn:127.0.0.1','api.netkiller.cn:127.0.0.1'])
service.environment(['TZ=Asia/Shanghai'])
service.ports(['80:80','443:443'])
service.volumes(['/tmp:/tmp'])
# service.debug()
# print(service.dump())

compose = Composes('development')
compose.version('3.9')
compose.services(service)
# print (compose.debug())
print(compose.dump())
compose.save()
```

怎么样，只是换了另一种写法，并没有难度。下面我们就系统学习，如何使用 python 编排 docker 容器

实际上程序最终还是会转化做 docker-compose 脚本执行。这种写法的有点是更灵活，你可以在程序中使用 if, while, 链接数据库，等等操作，可以做更复杂的容器编排。

## 2.1. 安装依赖库

```
neo@MacBook-Pro-Neo ~ % pip install netkiller-devops
```

确认是否安装成功

```
neo@MacBook-Pro-Neo ~ % pip show netkiller-devops
Name: netkiller-devops
Version: 0.2.4
Summary: DevOps of useful deployment and automation
Home-page: https://github.com/oscm/devops
Author: Neo Chen
Author-email: netkiller@msn.com
License: BSD
Location: /usr/local/lib/python3.9/site-packages
Requires: pyttsx3, requests, redis, pyyaml
Required-by:
```

## 2.2. 创建一个 Services

```
from netkiller.docker import *

service =  Services('nginx')
service.image('nginx:latest')
service.container_name('nginx')
service.restart('always')
service.hostname('www.netkiller.cn')
service.extra_hosts(['db.netkiller.cn:127.0.0.1','cache.netki
ller.cn:127.0.0.1','api.netkiller.cn:127.0.0.1'])
service.environment(['TZ=Asia/Shanghai'])
service.ports(['80:80','443:443'])
service.volumes(['/tmp:/tmp'])
# service.debug()
print(service.dump())
```

运行结果

```
nginx:
  container_name: nginx
  environment:
  - TZ=Asia/Shanghai
  extra_hosts:
  - db.netkiller.cn:127.0.0.1
  - cache.netkiller.cn:127.0.0.1
  - api.netkiller.cn:127.0.0.1
  hostname: www.netkiller.cn
  image: nginx:latest
  ports:
  - 80:80
  - 443:443
  restart: always
  volumes:
  - /tmp:/tmp
```

来一个复杂的演示

```
for i in range(10) :
    cluster =  Services('nginx-'+str(i))
    cluster.image('nginx:latest').container_name('nginx-
'+str(i)).restart('always').hostname('www'+str(i)+'.netkiller.c
n')
    cluster.ports(['8{port}:80'.format(port=i)])
    print(cluster.dump())
```

运行结果

```
nginx-0:
  container_name: nginx-0
  hostname: www0.netkiller.cn
  image: nginx:latest
  ports:
  - 80:80
  restart: always

nginx-1:
  container_name: nginx-1
  hostname: www1.netkiller.cn
  image: nginx:latest
  ports:
  - 81:80
  restart: always

nginx-2:
  container_name: nginx-2
  hostname: www2.netkiller.cn
  image: nginx:latest
  ports:
  - 82:80
  restart: always

nginx-3:
  container_name: nginx-3
  hostname: www3.netkiller.cn
  image: nginx:latest
  ports:
  - 83:80
  restart: always

nginx-4:
  container_name: nginx-4
  hostname: www4.netkiller.cn
  image: nginx:latest
  ports:
  - 84:80
  restart: always

nginx-5:
  container_name: nginx-5
  hostname: www5.netkiller.cn
  image: nginx:latest
  ports:
  - 85:80
  restart: always

nginx-6:
  container_name: nginx-6
  hostname: www6.netkiller.cn
  image: nginx:latest
  ports:
  - 86:80
  restart: always

nginx-7:
  container_name: nginx-7
  hostname: www7.netkiller.cn
  image: nginx:latest
  ports:
  - 87:80
  restart: always

nginx-8:
  container_name: nginx-8
  hostname: www8.netkiller.cn
  image: nginx:latest
  ports:
  - 88:80
  restart: always

nginx-9:
  container_name: nginx-9
  hostname: www9.netkiller.cn
  image: nginx:latest
  ports:
  - 89:80
  restart: always
```

## 2.3. 创建 Composes

Services 对象创建服务，让服务工作还需要 Composes 对象。

```
from netkiller.docker import *

service =  Services('nginx')
service.image('nginx:latest')
service.container_name('nginx')
service.restart('always')
service.hostname('www.netkiller.cn')
service.extra_hosts(['db.netkiller.cn:127.0.0.1','cache.netki
ller.cn:127.0.0.1','api.netkiller.cn:127.0.0.1'])
service.environment(['TZ=Asia/Shanghai'])
service.ports(['80:80','443:443'])
service.volumes(['/tmp:/tmp'])

compose = Composes('development')
compose.version('3.9')
compose.services(service)
# print (compose.debug())
print(compose.dump())
compose.save()
# compose.save('/tmp/docker-compose.yaml')
```

运行结果

```
services:
  nginx:
    container_name: nginx
    environment:
    - TZ=Asia/Shanghai
    extra_hosts:
    - db.netkiller.cn:127.0.0.1
    - cache.netkiller.cn:127.0.0.1
    - api.netkiller.cn:127.0.0.1
    hostname: www.netkiller.cn
    image: nginx:latest
    ports:
    - 80:80
    - 443:443
    restart: always
    volumes:
    - /tmp:/tmp
version: '3.9'
```

这已经是一个完善的 docker-compose 脚本了。使用 save 可以保存为 yaml 文件，这是使用 docker-compose -f development.yaml up 就可以启动容器了。

Composes 对象同时也携带了完善的 docker-compose 命令和参数，用于自我管理容器。

compose.up() 创建容器

```
compose = Composes('development')
compose.version('3.9')
compose.services(service)
compose.up()
```

compose.start() 启动已存在的容器

```
compose = Composes('development')
compose.version('3.9')
compose.services(service)
compose.start()
```

compose.stop() 停止已存在的容器

```
compose = Composes('development')
compose.version('3.9')
compose.services(service)
compose.stop()
```

compose.restart() 重启已存在的容器

```
compose = Composes('development')
compose.version('3.9')
compose.services(service)
compose.restart()
```

compose.rm() 销毁已存在的容器

```
compose = Composes('development')
compose.version('3.9')
compose.services(service)
compose.rm()
```

compose.logs() 查看容器日志

```
compose = Composes('development')
compose.version('3.9')
compose.services(service)
compose.logs()
```

compose.ps() 查看容器运行状态

```
compose = Composes('development')
```

```
compose.version('3.9')
compose.services(service)
compose.ps()
```

## 2.4. 容器管理

Docker 对象是让我们摆脱 docker-compose 这个命令，它将接管 docker-compose 这个命令，进行自我管理。

```
#!/usr/bin/python3
#-*- coding: utf-8 -*-
##############################################
# Home  : http://netkiller.github.io
# Author: Neo <netkiller@msn.com>
# Upgrade: 2021-09-05
##############################################
try:
        import os,  sys
        module = 
os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
        sys.path.insert(0,module)
        from netkiller.docker import *
except ImportError as err:
        print("%s" %(err))

nginx =  Services('nginx')
nginx.image('nginx:latest')
nginx.container_name('nginx')
nginx.restart('always')
nginx.hostname('www.netkiller.cn')
nginx.environment(['TA=Asia/Shanghai'])
nginx.ports(['80:80'])

compose = Composes('development')
compose.version('3.9')
compose.services(nginx)
compose.workdir('/tmp/compose')

if __name__ == '__main__':
        try:
                docker = Docker()
                docker.environment(compose)
                docker.main()
        except KeyboardInterrupt:
                print ("Crtl+C Pressed. Shutting down.")
```

运行结果

```
neo@MacBook-Pro-Neo ~ % python3 docker.py
Usage: docker.py [options]
up|rm|start|stop|restart|logs|top|images|exec <service>

Options:
  -h, --help          show this help message and exit
  --debug             debug mode
  -d, --daemon        run as daemon
  --logfile=LOGFILE   logs file.
  -l, --list          following logging
  -f, --follow        following logging
  -c, --compose       show docker compose
  -e, --export        export docker compose

Homepage: http://www.netkiller.cn       Author: Neo
<netkiller@msn.com>
```

Docker 对象提供了与 docker-compose 对等的参数，用法也基本相通。例如

```
python3 docker.py up = docker-compose up
python3 docker.py up -d nginx = docker-compose up -d nginx
python3 docker.py restart nginx = docker-compose restart nginx

python3 docker.py ps = docker-compose ps
python3 docker.py logs nginx = docker-compose logs nginx
```

使用 -c 可以查看 compose yaml 脚本，使用 -e 可以导出 docker compose yaml

## 2.5. 演示例子

### Redis 主从配置

**例 11.1. Redis Master/Slave**

```
from netkiller.docker import *

image = 'redis:latest'
requirepass='11223344'

compose = Composes('redis-master-slave')
compose.version('3.9')

master =  Services('master')
master.image(image)
master.container_name('master')
master.restart('always')
master.environment(['TZ=Asia/Shanghai'])
master.ports('6379:6379')
master.volumes(['/tmp/master:/data'])
master.sysctls(['net.core.somaxconn=1024'])
master.command([
        '--requirepass '+requirepass,
        '--appendonly yes'])
# master.debug()
# print(master.dump())
compose.services(master)


for i in range(5) :
    slave =  Services('slave-'+str(i))
    slave.image(image).container_name('slave-
'+str(i)).restart('always')
```

```
slave.ports(['638{port}:6379'.format(port=i)]).environment(['TZ
=Asia/Shanghai'])
    slave.volumes(['/tmp/slave{n}:/data'.format(n=i)])
    slave.sysctls(['net.core.somaxconn=1024']).command([
        '--slaveof master 6379',
        '--masterauth '+requirepass,
        '--requirepass ' + requirepass,
        '--appendonly yes'
    ])

    # print(cluster.dump())
    compose.services(slave)

# print (compose.debug())
print(compose.dump())
# compose.save()
compose.up()
```

## 2.6. 使用 Python 编排 Dockerfile

```
from netkiller.docker import *

# 实例化 Dockerfile() 对象
nginx = Dockerfile()

# 基于什么镜像
nginx.image('nginx:latest')

# 配置挂载卷
nginx.volume(['/etc/nginx','/var/log/nginx','/opt'])

# 运行脚本
nginx.run('apt update -y && apt install -y procps')

# 暴漏端口
nginx.expose(['80','443'])

# 设置工作目录
```

```
nginx.workdir('/opt')

# 打印 Dockerfile
nginx.show()
```

运行结果

```
FROM nginx:latest
VOLUME ["/etc/nginx","/var/log/nginx","/opt"]
RUN apt update -y && apt install -y procps
EXPOSE 80 443
WORKDIR /opt
```

另一种写法

```
from netkiller.docker import *

nginx = Dockerfile()
nginx.image('nginx:latest').volume(['/etc/nginx','/var/log/ng
inx']).run('apt update -y && apt install -y
procps').expose(['80','443']).workdir('/opt')
nginx.render()
nginx.save('/tmp/Dockerfile')
```

构建 Docker 镜像

```
from netkiller.docker import *

# 编排 Docker 镜像
dockerfile = Dockerfile()
```

```
dockerfile.image('openjdk:8').volume(['/srv']).run(
    'apt update -y && apt install -y procps net-tools
iputils-ping iproute2 telnet'
).expose(['80', '443']).workdir('/srv')

# 通过 Service 设置镜像名称是 netkiller:openjdk8
image = Services('image')
image.build(dockerfile)
image.image('netkiller:openjdk8')

# 构建镜像
demo = Composes('demo')
demo.version('3.9')
demo.services(image)
demo.build()
```

完整演示

```
#!/usr/bin/python3
#-*- coding: utf-8 -*-
##############################################
# Home  : http://netkiller.github.io
# Author: Neo <netkiller@msn.com>
# Upgrade: 2021-11-17
##############################################
try:
        import os,  sys
        module =
os.path.dirname(os.path.dirname(os.path.dirname(os.path.abspath
(__file__))))
        print(module)
        sys.path.insert(0,module)
        from netkiller.docker import *
except ImportError as err:
        print("%s" %(err))

dockerfile = Dockerfile()
#
dockerfile.label({'org.opencontainers.image.authors':'netkiller
'})
```

```
dockerfile.image('openjdk:8-alpine')
# dockerfile.image('openjdk:8')
dockerfile.env({'ROCKETMQ_VERSION':'4.9.2','ROCKETMQ_HOME':'/sr
v/rocketmq', 'PATH':'${ROCKETMQ_HOME}/bin:$PATH'}) #
'JAVA_OPT':'"${JAVA_OPT} -server -Xms512m -Xmx2048m -Xmn128m"'
dockerfile.arg({'user':'rocketmq', 'group':'nogroup'})
dockerfile.run('wget
https://dlcdn.apache.org/rocketmq/4.9.2/rocketmq-all-4.9.2-bin-
release.zip && unzip rocketmq-all-4.9.2-bin-release.zip')
dockerfile.run('mv rocketmq-4.9.2 /srv/rocketmq-4.9.2 && rm -rf
rocketmq-all-4.9.2-bin-release.zip')
dockerfile.run('ln -s /srv/rocketmq-${ROCKETMQ_VERSION}
/srv/rocketmq')
dockerfile.run('adduser -S -D ${user}')
dockerfile.run(['chown ${user}:${group} -R
/srv/rocketmq-${ROCKETMQ_VERSION}'])
dockerfile.expose(['9876'])
dockerfile.expose(['10909','10911','10912'])
dockerfile.copy('docker-entrypoint.sh','/srv/docker-
entrypoint.sh')
dockerfile.run('chmod a+x /srv/docker-entrypoint.sh')
dockerfile.entrypoint('["/srv/docker-entrypoint.sh"]')
dockerfile.workdir('${ROCKETMQ_HOME}')
# dockerfile.render()
# dockerfile.save('/tmp/Dockerfile')

rocketmq = Services('rocketmq')
rocketmq.build(dockerfile).image('registry.netkiller.cn/rocketm
q/rocketmq:4.9.2').container_name('rocketmq')
# rocketmq.entrypoint('/srv/rocketmq/bin/mqnamesrv')
#
rocketmq.ports('9876:9876').command('/srv/rocketmq/bin/mqnamesr
v')

dockerfile = Dockerfile()
dockerfile.image('registry.netkiller.cn/rocketmq/rocketmq:4.9.2
')
dockerfile.run('ln -s /srv/rocketmq-${ROCKETMQ_VERSION}
/srv/mqnamesrv')
dockerfile.cmd('/srv/mqnamesrv/bin/mqnamesrv')
dockerfile.workdir('/srv/mqnamesrv')
dockerfile.user('rocketmq:nogroup')
dockerfile.volume([
        '/home/rocketmq/logs/rocketmqlogs'
])
```

```
mqnamesrv = Services('mqnamesrv')
mqnamesrv.build(dockerfile).image('registry.netkiller.cn/rocket
mq/mqnamesrv:4.9.2').container_name('mqnamesrv').ports('9876:98
76')
mqnamesrv.command('mqnamesrv')

dockerfile = Dockerfile()
dockerfile.image('registry.netkiller.cn/rocketmq/rocketmq:4.9.2
')
dockerfile.run('ln -s /srv/rocketmq-${ROCKETMQ_VERSION}
/srv/mqbroker')
dockerfile.cmd('/srv/rocketmq/bin/mqbroker')
dockerfile.workdir('/srv/mqbroker')
dockerfile.user('rocketmq:nogroup')
dockerfile.volume([
        '/home/rocketmq/logs/rocketmqlogs'
])

mqbroker = Services('mqbroker')
mqbroker.build(dockerfile).image('registry.netkiller.cn/rocketm
q/mqbroker:4.9.2').container_name('mqbroker').ports(['10909:109
09','10911:10911','10912:10912'])
mqbroker.command('mqbroker -n mqnamesrv:9876 -c
/srv/rocketmq/conf/broker.conf')
mqbroker.volumes(['/tmp/logs:/home/rocketmq/logs/rocketmqlogs']
)

composes = Composes('test')
composes.version('3.9')
composes.services(rocketmq)
composes.services(mqnamesrv)
composes.services(mqbroker)


# cat >> /srv/docker-entrypoint.sh <<'EOF'
# EOF

entrypoint='''#!/bin/sh
if [ "$1" = 'mqnamesrv' ]; then
        exec /srv/rocketmq/bin/mqnamesrv
fi
exec "$@"
'''
```

```
if __name__ == '__main__':
        try:
                docker = 
Docker({'DOCKER_HOST':'ssh://root@192.168.30.11','NAMESRV_ADDR'
:'localhost:9876'})
                docker.createfile('rocketmq/rocketmq/docker-
entrypoint.sh',entrypoint)
                docker.environment(composes)
                docker.main()
        except KeyboardInterrupt:
                print ("Crtl+C Pressed. Shutting down.")
```

运行

```
python3 demo.py -e test -b rocketmq
```

## 2.7.

```
#!/usr/bin/python3
#-*- coding: utf-8 -*-
##############################################
# Home  : http://netkiller.github.io
# Author: Neo <netkiller@msn.com>
# Upgrade: 2022-08-19
##############################################
try:
        import os, sys
        from netkiller.docker import *
except ImportError as err:
        print("%s" %(err))

#extra_hosts = [
#    'mongo.netkiller.cn:172.17.195.17',
'eos.netkiller.cn:172.17.15.17',
#    'cfca.netkiller.cn:172.17.15.17'
#]

# 解决时区问题，只能制作新镜像，并且在镜像中增加 tzdata
dockerfile = Dockerfile()
dockerfile.image('openresty/openresty:alpine').run(
    'apk add -U tzdata',
    'cp /usr/share/zoneinfo/Asia/Shanghai /etc/localtime'
)
openresty = Services('openresty')
openresty.build(dockerfile)
openresty.image('openresty:alpine')
openresty.container_name('openresty')
openresty.restart('always')
openresty.hostname('www.netkiller.cn')
#openresty.extra_hosts(extra_hosts)
#
service.extra_hosts(['db.netkiller.cn:127.0.0.1','cache.netki
ller.cn:127.0.0.1','api.netkiller.cn:127.0.0.1'])
openresty.environment(['TZ=Asia/Shanghai'])
openresty.ports(['80:80','443:443'])
#openresty.depends_on('test')
openresty.working_dir('/usr/local/openresty')
openresty.volumes(
        [
        '/var/log/openresty:/usr/local/openresty/nginx/logs',
        ]
)

development = Composes('development')
development.workdir('/var/tmp/development')
development.version('3.9')
development.services(openresty)


if __name__ == '__main__':
    try:
        docker = Docker(
        #       {'DOCKER_HOST': 'ssh://root@192.168.30.11'}
        )
        #docker.sysctl({'neo': '1'})
        docker.environment(development)
        docker.main()
    except KeyboardInterrupt:
        print("Crtl+C Pressed. Shutting down.")
```

## 2.8. logstash

```
[root@netkiller log]# cat /srv/logstash/bin/logstash
#!/usr/bin/python3
# -*- coding: utf-8 -*-
##############################################
# Home  : http://netkiller.github.io
# Author: Neo <netkiller@msn.com>
# Upgrade: 2023-01-11
##############################################
import os
import sys
try:
        module =
os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
        sys.path.insert(0, module)
        from netkiller.docker import *
except ImportError as err:
        print("%s" % (err))

project = 'logstash'

# extra_hosts = [
#     'mongo.netkiller.cn:172.17.195.17',
'eos.netkiller.cn:172.17.15.17',
#     'cfca.netkiller.cn:172.17.15.17'
# ]

dockerfile = Dockerfile()
dockerfile.image('docker.elastic.co/logstash/logstash:8.6.0').r
un(
        ['apk add -U tzdata', 'rm -f
/usr/share/logstash/pipeline/logstash.conf']
).copy('pipeline/',
'/usr/share/logstash/pipeline/').copy('config/',
'/usr/share/logstash/config/').workdir('/usr/share/logstash')

logstash = Services(project)
```

```
# openresty.image('openresty/openresty:alpine')
# openresty.build(dockerfile)
logstash.image('docker.elastic.co/logstash/logstash:8.6.0')
logstash.container_name(project)
logstash.restart('always')
# logstash.hostname('www.netkiller.cn')
# openrelogstashsty.extra_hosts(extra_hosts)
logstash.extra_hosts(['elasticsearch:127.0.0.1'])
logstash.environment(['TZ=Asia/Shanghai','XPACK_MONITORING_ENAB
LED=false','LOG_LEVEL=info'])
logstash.ports(['12201:12201/udp', '12201:12201/tcp'])
#logstash.ports(['12201:12201','4567:4567'])
# openresty.depends_on('test')
logstash.working_dir('/usr/share/logstash')
logstash.user('root')
logstash.volumes(
        [

'/srv/logstash/pipeline/:/usr/share/logstash/pipeline/',

#'/srv/logstash/config/logstash.yml:/usr/share/logstash/config/
logstash.yml:rw',

'/srv/logstash/logs/:/usr/share/logstash/logs/',
                '/opt/log/:/opt/log/',
                '/proc:/proc','/sys:/sys'
        ]
).privileged()

development = Composes('development')
development.workdir('/var/tmp/development')
development.version('3.9')
development.services(logstash)

if __name__ == '__main__':
        try:
                docker = Docker(
                        # {'DOCKER_HOST':
'ssh://root@192.168.30.11'}
                )
                # docker.sysctl({'neo': '1'})
                docker.environment(development)
                docker.main()
        except KeyboardInterrupt:
                print("Crtl+C Pressed. Shutting down.")
```

pipeline

```
[root@netkiller log]# cat /srv/logstash/pipeline/config.conf
input {
        tcp {
                port => 4567
                codec => json_lines
        }
        gelf {
                port => 12201
                use_udp => true
                use_tcp => true
        }
}

filter {
        ruby {
                code => "event.set('datetime', 
event.get('@timestamp').time.localtime.strftime('%Y-%m-%d 
%H:%M:%S'))"
        }
}

output {
        if [marker] {
                file {
                        path => "/opt/log/%{environment}/%
{service}/%{marker}.%{+yyyy}-%{+MM}-%{+dd}.log"
                        codec => line { format => "[%
{datetime}] %{level} %{message}"}
                }
        } else {
                file {
                        path => "/opt/log/%{environment}/%
{service}/spring.%{+yyyy}-%{+MM}-%{+dd}.log"
                        codec => line { format => "[%
{datetime}] [%{host}:%{source_host}] [%{level}] (%{class}.%
{method}:%{line}) - %{message}"}
```

```
                }
        }
        file {
                path => "/opt/log/%{environment}/%
{service}/spring.%{+yyyy}-%{+MM}-%{+dd}.json.gz"
                codec => json_lines
                gzip => true
        }

        if "ERROR" in [level] {
                http {
                        url =>
"https://oapi.dingtalk.com/robot/send?
access_token=f9257740a95b0b052e69c699400ea0ec06ae40fa5db316613f
084b0162de90f8"
                        http_method => "post"
                        content_type => "application/json;
charset=utf-8"
                        format => "message"
                        message => '{"msgtype":"text","text":
{"content":"Logger: %{host}[%{source_host}] - %{message}"}}'
                }
        }
        if "WARN" in [level] {
                http {
                        url =>
"https://oapi.dingtalk.com/robot/send?
access_token=d6602c6fbe68d31f791968a12201a6980f36b47250f39a57a1
17582afca7678b"
                        http_method => "post"
                        content_type => "application/json;
charset=utf-8"
                        format => "message"
                        message => '{"msgtype":"text","text":
{"content":"Logger: %{host}[%{source_host}] - %{message}"}}'
                }
        }
}
```

# 3. 使用 Python 优雅地编排 Kubernetes

## 3.1. 快速演示编排Nginx

你还用 yaml编排 kubernetes 吗？你是否意识到YAML的局限性，例如你无法定义变量，不能循环重复内容，不能跟高级语言互动，于是你转向了 HELM， helm 提供模版技术，可以在模版中实现包含引用，定义变量，循环等等操作，但也仅此而已。 YAML 和 HELM 方案更多是给运维人员准备的，对开发并不友好，那么有没有更好的解决方案呢？

我用 python 写的一个工具吧 netkiller-devops，安装方法

```
pip install netkiller-devops
```

下面编排一个 nginx 给大家演示一下。运行环境使用 macOS + k3d

**提示**

k3s 是由 Rancher Labs 推出的一款轻量级 Kubernetes 发行版，满足在边缘计算环境中运行在 x86、ARM64 处理器上的小型、易于管理的 Kubernetes 集群日益增长的需求。

k3s 除了在边缘计算领域的应用外，在研发侧的表现也十分出色。我们可以快速在本地拉起一个轻量级的 k8s 集群，而 k3d 则是 k3s 社区创建的一个小工具，可以在一个 docker 进程中运行整个 k3s 集群，相比直接使用 k3s 运行在本地，更好管理和部署。

安装 k3d

```
brew install k3d
```

启动集群

```
k3d cluster create mycluster --api-port 6443 --servers 1 --
agents 1 --port '80:80@loadbalancer' --port
'443:443@loadbalancer'
```

现在创建一个 python 文件 例如 nginx.py 把下面内容复制进去

```
import os, sys

module = os.path.dirname(

os.path.dirname(os.path.dirname(os.path.abspath(__file__))))
print(module)
sys.path.insert(0, module)
from netkiller.kubernetes import *

namespace = Namespace()
namespace.metadata.name('development')
namespace.metadata.namespace('development')
# namespace.debug()

service = Service()
service.metadata().name('nginx')
service.metadata().namespace('development')
service.spec().selector({'app': 'nginx'})
service.spec().type('NodePort')
service.spec().ports([{
    'name': 'http',
    'protocol': 'TCP',
    'port': 80,
    'targetPort': 80
}])
```

```
deployment = Deployment()
deployment.apiVersion('apiVersion: apps/v1')
deployment.metadata().name('nginx').labels({'app':
'nginx'}).namespace('development')
deployment.spec().replicas(2)
deployment.spec().selector({'matchLabels': {'app': 'nginx'}})
deployment.spec().template().metadata().labels({'app':
'nginx'})
deployment.spec().template().spec().containers().name('nginx'
).image(
    'nginx:latest').ports([{
        'containerPort': 80
    }])
# deployment.debug()

ingress = Ingress()
ingress.apiVersion('networking.k8s.io/v1')
ingress.metadata().name('nginx')
ingress.metadata().namespace('development')
ingress.metadata().annotations({'ingress.kubernetes.io/ssl-
redirect': "false"})
ingress.spec().rules([{
    # 'host': 'www.netkiller.cn',
    'http': {
        'paths': [{
            'path': '/',
            'pathType': 'Prefix',
            'backend': {
                'service': {
                    'name': 'nginx',
                    'port': {
                        'number': 80
                    }
                }
            }
        }]
    }
}])

# ingress.debug()

compose = Compose('development')
compose.add(namespace)
compose.add(service)
compose.add(deployment)
compose.add(ingress)
# compose.debug()
# compose.yaml()
# compose.save('/tmp/test.yaml')

kubernetes = Kubernetes()
kubernetes.compose(compose)
# kubernetes.debug()
# print(kubernetes.dump())
kubernetes.main()
```

查看帮助信息 /usr/bin/python3 nginx.py -h

```
➜  devops git:(master) ✗ /usr/bin/python3 nginx.py -h
Usage: nginx.py [options] <command>

Options:
  -h, --help            show this help message and exit
  -e development|testing|production, --
environment=development|testing|production
                        environment
  -l, --list            print service of environment

  Cluster Management Commands:
    -g, --get           Display one or many resources
    -c, --create        Create a resource from a file or from
stdin
    -d, --delete        Delete resources by filenames, stdin,
resources and
                        names, or by resources and label
selector
    -r, --replace       Replace a resource by filename or stdin

  Namespace:
    -n, --namespace     Display namespace
    -s, --service       Display service

  Others:
    --logfile=LOGFILE   logs file.
    -y, --yaml          show yaml compose
    --export             export docker compose
    --debug              debug mode
    -v, --version        print version information
```

现在开始部署 nginx 使用参数 -c，命令 /usr/bin/python3 nginx.py -c

```
➜  devops git:(master) ✗ /usr/bin/python3 nginx.py -c
namespace/development created
service/nginx created
deployment.apps/nginx created
ingress.networking.k8s.io/nginx created
```

查看部署状态

```
➜  devops git:(master) ✗ kubectl get namespace
NAME                 STATUS   AGE
default              Active   3h15m
kube-system          Active   3h15m
kube-public          Active   3h15m
kube-node-lease      Active   3h15m
development          Active   21m

➜  devops git:(master) ✗ kubectl get service -n development
NAME    TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)
AGE
nginx   NodePort    10.43.19.13   <none>        80:31258/TCP
21m

➜  devops git:(master) ✗ kubectl get deployment -n development
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
nginx   2/2     2            2           21m

➜  devops git:(master) ✗ kubectl get ingress -n development
NAME    CLASS    HOSTS   ADDRESS                 PORTS   AGE
nginx   <none>   *       172.23.0.2,172.23.0.3   80      21m
```

```
```

检验 nginx 启动情况

```
➜  devops git:(master) ✗ curl http://localhost
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully
installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

## 3.2. 创建命名空间

```
import os, sys
from netkiller.kubernetes import *

print("=" * 40, "Namespace", "=" * 40)
```

```
namespaces = []
environment = ['development','testing','production']
for name in environment :
    namespace = Namespace(name)
    namespace.metadata().name(name)
    namespace.metadata().namespace(name)
    # namespace.debug()
    namespaces.append(namespace)

compose = Compose('development')
for ns in namespaces :
    compose.add(ns)

# compose.debug()
# compose.save('/tmp/test.yaml')
# compose.delete()
compose.create()
```

## 3.3. ConfigMap/Secret 编排演示

ConfigMap 实例

```
from netkiller.kubernetes import *

config = ConfigMap()
config.apiVersion('v1')
config.metadata().name('test').namespace('test')
config.data({'host':'localhost','port':3306,'user':'root','pa
ss':'123456'})
config.data({'redis.conf':pss(
    'pidfile /var/lib/redis/redis.pid\n'
    'dir /var/lib/redis\n'
    'port 6379\n'
    'bind 0.0.0.0\n'
    'appendonly yes\n'
    'protected-mode no\n'
    'requirepass 123456\n'
    )
    })
```

```
config.data({'dbhost':'localhost','dbport':3306,'dbuser':'roo
t','dbpass':'123456'}).data({'mysql.cnf':pss('''\
mysql.db = devops
mysql.host = 127.0.0.1
mysql.user = root
mysql.pwd  = root123
mysql.port = 3306
''')})
config.json()
config.debug()
```

输出结果

```
metadata:
  name: test
  namespace: test
data:
  host: localhost
  port: 3306
  user: root
  pass: '123456'
  redis.conf: |
    pidfile /var/lib/redis/redis.pid
    dir /var/lib/redis
    port 6379
    bind 0.0.0.0
    appendonly yes
    protected-mode no
    requirepass 123456
  dbhost: localhost
  dbport: 3306
  dbuser: root
  dbpass: '123456'
  mysql.cnf: |
    mysql.db = devops
    mysql.host = 127.0.0.1
    mysql.user = root
    mysql.pwd  = root123
    mysql.port = 3306
apiVersion: v1
```

```
kind: ConfigMap
```

Secret 实例

```
secret = Secret()
secret.metadata().name('tls').namespace('development')
secret.data({'tls.crt':' ','tls.key':' '})
secret.type('kubernetes.io/tls')
secret.debug()
```

Secret 运行结果

```
metadata:
  name: tls
  namespace: development
data:
  tls.crt: ' '
  tls.key: ' '
type: kubernetes.io/tls
apiVersion: v1
kind: Secret
```

从文件创建 ConfigMap

```
from netkiller.kubernetes import *

print("=" * 40, "ConfigMap", "=" * 40)
config = ConfigMap()
config.apiVersion('v1')
config.metadata().name('test').namespace('default')
config.from_file('redis.conf',
'/etc/redis/redis.conf').from_file('nginx.conf','/etc/nginx/n
ginx.conf')
```

从环境变量文件创建 ConfigMap

```
config = ConfigMap('test')
config.apiVersion('v1')
config.metadata().name('test').namespace('test')
config.from_env_file('config.env')
config.debug()
```

```
neo@Netkiller-iMac ~/w/d/d/k8s (master) [1]> cat config.env
key=value
dev.logfile=/tmp/logfile.log
dev.tmpdir=/tmp
```

运行结果

```
neo@Netkiller-iMac ~/w/d/d/k8s (master)> python3
/Users/neo/workspace/devops/demo/k8s/demo.py
metadata:
  name: test
  namespace: test
data:
  key: value
  dev.logfile: /tmp/logfile.log
  dev.tmpdir: /tmp
apiVersion: v1
```

```
kind: ConfigMap
```

## 3.4. Pod 挂载 ConfigMap 编排演示

```
from netkiller.kubernetes import *

print("=" * 40, "ConfigMap", "=" * 40)
config = ConfigMap()
config.apiVersion('v1')
config.metadata().name('test').namespace('default')
config.data({'redis.conf':pss(
    'pidfile /var/lib/redis/redis.pid\n'
    'dir /var/lib/redis\n'
    'port 6379\n'
    'bind 0.0.0.0\n'
    'appendonly yes\n'
    'protected-mode no\n'
    'requirepass 123456\n'
    )
    })
config.debug()

print("=" * 40, "Pod", "=" * 40)

pod = Pod()
pod.metadata().name('busybox')
pod.spec().containers().name('test').image('busybox').command
([ "/bin/sh","-c","cat /tmp/config/redis.conf"
]).volumeMounts([{'name':'config-
volume','mountPath':'/tmp/config/redis.conf','subPath':'redis
.conf'}])
pod.spec().volumes().name('config-
volume').configMap({'name':'test'}) # , 'items':
[{'key':'redis.conf','path':'keys'}]
pod.debug()

print("=" * 40, "Compose", "=" * 40)
compose = Compose('development')
# compose.add(namespace)
```

```
compose.add(config)
compose.add(pod)

compose.delete()
compose.create()

print("=" * 40, "Busybox", "=" * 40)
os.system("sleep 10 && kubectl logs busybox")
```

生成 yaml 内容

```
metadata:
  name: test
  namespace: default
data:
  redis.conf: |
    pidfile /var/lib/redis/redis.pid
    dir /var/lib/redis
    port 6379
    bind 0.0.0.0
    appendonly yes
    protected-mode no
    requirepass 123456
apiVersion: v1
kind: ConfigMap
---
metadata:
  name: busybox
spec:
  containers:
    - name: test
      image: busybox
      command:
        - /bin/sh
        - -c
        - cat /tmp/config/redis.conf
      volumeMounts:
        - name: config-volume
          mountPath: /tmp/config/redis.conf
          subPath: redis.conf
  volumes:
    - name: config-volume
      configMap:
        name: test
apiVersion: v1
kind: Pod
```

运行结果

```
configmap "test" deleted
pod "busybox" deleted
configmap/test created
pod/busybox created
======================================== Busybox
========================================
pidfile /var/lib/redis/redis.pid
dir /var/lib/redis
port 6379
bind 0.0.0.0
appendonly yes
protected-mode no
requirepass 123456
```

## 3.5. Pod 挂载 ConfigMap 设置环境变量

```
import os,sys
sys.path.insert(0, '/Users/neo/workspace/devops')
from netkiller.kubernetes import *

print("=" * 40, "ConfigMap", "=" * 40)
config = ConfigMap()
config.apiVersion('v1')
config.metadata().name('test').namespace('default')
config.data({'host':'localhost','port':'3306','user':'root','
pass':'123456'})
config.from_file('nginx.conf',
'/etc/nginx/nginx.conf').from_env_file('redis.conf','redis.en
v')

pod = Pod()
pod.metadata().name('busybox')
pod.spec().containers().name('test').image('busybox').command
([ "/bin/sh","-c","env" ]).env([{'name':'DBHOST','valueFrom':
{'configMapKeyRef':{'name':'test','key':'host'}}}])

compose = Compose('development')
compose.add(config)
compose.add(pod)
compose.delete()
compose.create()

print("=" * 40, "Busybox", "=" * 40)
os.system("sleep 10 && kubectl logs busybox")
```

输出结果

```
configmap "test" deleted
pod "busybox" deleted
configmap/test created
pod/busybox created
======================================== Busybox
========================================
KUBERNETES_PORT=tcp://10.43.0.1:443
KUBERNETES_SERVICE_PORT=443
HOSTNAME=busybox
SHLVL=1
HOME=/root
DBHOST=localhost
KUBERNETES_PORT_443_TCP_ADDR=10.43.0.1
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/b
in
KUBERNETES_PORT_443_TCP_PORT=443
KUBERNETES_PORT_443_TCP_PROTO=tcp
```

```
KUBERNETES_SERVICE_PORT_HTTPS=443
KUBERNETES_PORT_443_TCP=tcp://10.43.0.1:443
KUBERNETES_SERVICE_HOST=10.43.0.1
PWD=/
```

DBHOST=localhost

## 3.6. Ingress 挂载 SSL 证书

准备 SSL 证书，如果你没有，可以使用下面命令创建

```
制作私钥证书
openssl genrsa -out ingress.key 2048

制作公钥证书
openssl req -new -x509 -days 3650 -key ingress.key -out
ingress.crt

mkdir -p cert/private
cp ingress.crt cert/netkiller.cn.crt
cp ingress.key cert/private/netkiller.cn.key
```

编排脚本

```
import sys
sys.path.insert(0, '/Users/neo/workspace/devops')
from netkiller.kubernetes import *

namespace = 'default'

# namespace = Namespace()
# namespace.metadata().name(namespace)
# namespace.metadata().namespace(namespace)
# namespace.debug()
```

```
secret = Secret('ingress-secret')
secret.metadata().name('tls').namespace(namespace)
# secret.data({'tls.crt':' ','tls.key':' '})
secret.cert('cert/netkiller.cn.crt')
secret.key('cert/private/netkiller.cn.key')
secret.type('kubernetes.io/tls')
# secret.save()
# secret.debug()
# exit()

service = Service()
service.metadata().name('nginx')
service.metadata().namespace(namespace)
service.spec().selector({'app': 'nginx'})
service.spec().type('NodePort')
service.spec().ports([{
    'name': 'http',
    'protocol': 'TCP',
    'port': 80,
    'targetPort': 80
}])

deployment = Deployment()
deployment.apiVersion('apps/v1')

deployment.metadata().name('nginx').labels(
    {'app': 'nginx'}).namespace(namespace)
deployment.spec().replicas(1)
deployment.spec().selector({'matchLabels': {'app': 'nginx'}})
deployment.spec().template().metadata().labels({'app':
'nginx'})
deployment.spec().template().spec().containers().name('nginx'
).image(
    'nginx:latest').ports([{
        'containerPort': 80
    }])
# deployment.debug()
# deployment.json()

ingress = Ingress()
ingress.apiVersion('networking.k8s.io/v1')
ingress.metadata().name('nginx')
ingress.metadata().namespace(namespace)
ingress.metadata().annotations({'ingress.kubernetes.io/ssl-
redirect': "true"})
ingress.spec().tls([{'hosts':
['www.netkiller.cn','admin.netkiller.cn'],'secretName':'tls'}
])
ingress.spec().rules([{
    'host': 'www.netkiller.cn',
    'http': {
        'paths': [{
            'path': '/',
            'pathType': 'Prefix',
            'backend': {
                'service': {
                    'name': 'nginx',
                    'port': {
                        'number': 80
                    }
                }
            }
        }]
    }
}])

# ingress.debug()

print("=" * 40, "Compose", "=" * 40)
compose = Compose('development')
# compose.add(namespace)
compose.add(secret)
compose.add(service)
compose.add(deployment)
compose.add(ingress)
# compose.debug()
# compose.save('/tmp/test.yaml')
compose.delete()
compose.create()

print("=" * 40, "Busybox", "=" * 40)
os.system("sleep 5")
for cmd in ['kubectl get secret tls','kubectl get
pods','kubectl get service','kubectl get deployment','kubectl
get ingress'] :
    os.system(cmd)
    print("-" * 50)
```

启动后使用 openssl 检查证书

```
neo@Netkiller-iMac ~> openssl s_client -connect
www.netkiller.cn:443
CONNECTED(00000003)
depth=0 CN = TRAEFIK DEFAULT CERT
verify error:num=20:unable to get local issuer certificate
verify return:1
depth=0 CN = TRAEFIK DEFAULT CERT
verify error:num=21:unable to verify the first certificate
verify return:1
---
Certificate chain
 0 s:/CN=TRAEFIK DEFAULT CERT
   i:/CN=TRAEFIK DEFAULT CERT
---
Server certificate
-----BEGIN CERTIFICATE-----
MIIDXjCCAkagAwIBAgIRAPLS5GFlqTUbZuNxXxu9SGEwDQYJKoZIhvcNAQELBQA
w
HzEdMBsGA1UEAxMUVFJBRUZJSyBERUZBVUxUIENFUlQwHhcNMjIwMTE0MDQwNDU
2
WhcNMjMwMTE0MDQwNDU2WjAfMR0wGwYDVQQDExRUUkFFRklLIERFRkFVTFQgQ0V
S
VDCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBALtuaTUNs89KKUm6dG8
M
JUcdqsNLsG0a369O+VjSSgJnrYb9BL8ZTCTYTu44y8cepH+mMdq1SVmDpXwyMVP
u
CuXYDnrK2n6Zdv9T9K59pKOu08GoRmF7kmxmA8d4UGbDR5D01AEjOLvd8EKzRJq
i
tB8KP5KEjdVUQYB7ZUy3EHSsfyM+grN/XbWn0Sfj7VGWnUBS+WG9Huvi+vgHwU5
W
r+JL5ojsWw7q6glG45x3iIjqYNaVWqRwuSoH905AIA9Q2mCpRjNNQJL1sUYxHFf
d
mYlOW47ovKIw/OR48lqlwZy8/YblDveIn66kEAF7Y3EGDQuUB21lSW6q7qNum7l
q
S5MCAwEAAaOBlDCBkTAOBgNVHQ8BAf8EBAMCA7gwEwYDVR0lBAwwCgYIKwYBBQU
H
AwEwDAYDVR0TAQH/BAIwADBcBgNVHREEVTBTglE0NWNjOThiNDQ0MTlmOTM2ODc
```

```
w
YTU5YTZkN2EyZWRhZC5lMWIyMDRmZTVjMTlhZGJjNWE4NjE3NjA0YzkxNGI4OS5
0
cmFlZmlrLmRlZmF1bHQwDQYJKoZIhvcNAQELBQADggEBAG+BrjgG0Z8j4/G08eC
J
elVpUaxCXzWEC6KgPmQPpgYGh98PcrZNe4E/FnaKJ9pjtA7NpG8Y2Ke+D3D8H+M
Q
hutT9+XtGRU93zxpT3SVxJLHQnx3511s0jAfj3sCxyvuv17bT+q8C0KjQf9k6HM
T
X/oBsND0HXrDbdsUK4f2sCdmql0CK/uAj0ibjfjajfCc5Ve5hQw1a5x2StCvQZA
B
6TO8YQpFR+TeIbyclr++tYLBBocl0E3nXFommYPt2zxiY1K129fNPRfmq+yKbuz
V
4u1KLRWIUJnab6Ue7ezJLCNT5T0bVXSG089yeaB/MdPRVkbAMHXF+AxQDUu9iZx
+
8Aw=
-----END CERTIFICATE-----
subject=/CN=TRAEFIK DEFAULT CERT
issuer=/CN=TRAEFIK DEFAULT CERT
---
No client certificate CA names sent
Server Temp Key: ECDH, X25519, 253 bits
---
SSL handshake has read 1454 bytes and written 289 bytes
---
New, TLSv1/SSLv3, Cipher is ECDHE-RSA-AES256-GCM-SHA384
Server public key is 2048 bit
Secure Renegotiation IS supported
Compression: NONE
Expansion: NONE
No ALPN negotiated
SSL-Session:
    Protocol  : TLSv1.2
    Cipher    : ECDHE-RSA-AES256-GCM-SHA384
    Session-ID:
0A39917DAE8C45B5495FA7CDEF733CF524A117E070B37428C984550AB938299
3
    Session-ID-ctx:
    Master-Key:
F9F5464856CE3D12437AC45843A07732C1A313E99240F6C8AAD6A8BEC957786
237846A687B62C5A4A6362FD738B68F2D
    TLS session ticket:
    0000 - ca 1d cc 1f fa ea 48 88-f2 d8 b2 94 ac 32 d0 f4
......H......2..
    0010 - 4f ad 8c de 17 49 97 c8-7f 73 2d 3d 04 86 86 f0
```

```
O....I...s-=....
    0020 - 9c 51 e3 60 50 c6 ab 70-3d a6 8a a5 5c 50 c7 04
.Q.`P..p=...\P..
    0030 - 89 93 89 a6 d5 c5 73 ac-2a 3f f6 1c 7b 26 5f 70
......s.*?..{&_p
    0040 - 0b 27 ae bd 5b 37 b0 f4-76 79 5d 9d 90 10 f5 24
.'..[7..vy]....$
    0050 - ef 64 04 4b cd ad c3 83-2b f3 a4 37 6a 83 f8 ce
.d.K....+..7j...
    0060 - 6e 18 e3 72 64 a9 c1 6c-7d 24 9a 1d f6 b7 76 d7
n..rd..l}$....v.
    0070 - 68 ee 8f 76 27 06 bf 84-4d 6d 33 f3 b7 c5 4e d4
h..v'...Mm3...N.
    0080 - 32                                                2

    Start Time: 1642133830
    Timeout   : 7200 (sec)
    Verify return code: 21 (unable to verify the first
certificate)
---
```

证书载入正确，就可以使用 curl 命令或者Safari测试了

```
neo@Netkiller-iMac ~> curl https://www.netkiller.cn
```

如果是自签名证书，需要使用 -k 参数

```
neo@Netkiller-iMac ~> curl -k https://www.netkiller.cn
```

## 3.7. StatefulSet 部署 Redis

```
import sys
sys.path.insert(0, '/Users/neo/workspace/devops')
from netkiller.kubernetes import *
namespace = 'default'

config = ConfigMap('redis')
config.metadata().name('redis').namespace(namespace).labels({
'app': 'redis'})
config.data({'redis.conf': pss('''\
pidfile /var/lib/redis/redis.pid
dir /data
port 6379
bind 0.0.0.0
appendonly yes
protected-mode yes
requirepass passw0rd
maxmemory 2mb
maxmemory-policy allkeys-lru
''')})
# config.debug()


statefulSet = StatefulSet()
statefulSet.metadata().name('redis')
statefulSet.spec().replicas(1)
statefulSet.spec().serviceName('redis')
statefulSet.spec().selector({'matchLabels': {'app':
'redis'}})
statefulSet.spec().template().metadata().labels({'app':
'redis'})
#
statefulSet.spec().template().spec().initContainers().name('b
usybox').image('busybox').command(['sh','-c','mkdir -p
/var/lib/redis && echo 2048 > /proc/sys/net/core/somaxconn &&
echo never >
/sys/kernel/mm/transparent_hugepage/enabled']).volumeMounts([
#         {'name': 'data', 'mountPath': '/var/lib/redis'}])
statefulSet.spec().template().spec().containers().name('redis
').image(
    'redis:latest').command(['sh', '-c', 'redis-server
/usr/local/etc/redis.conf']).ports([{
        'name': 'redis',
        'protocol': 'TCP',
        'containerPort': 6379
    }]).volumeMounts([
        {'name': 'config', 'mountPath':
'/usr/local/etc/redis.conf',
            'subPath': 'redis.conf'},
        {'name': 'data', 'mountPath': '/data',
            'subPath': 'default.conf'}
    ]).imagePullPolicy('IfNotPresent')
statefulSet.spec().template().spec().volumes().name(
    'config').configMap({'name': 'redis'})
statefulSet.spec().template().spec().volumes().name(
    'data').hostPath({'path': '/var/lib/redis'})
# statefulSet.debug()
# exit()

service = Service()
service.metadata().name('redis')
service.metadata().namespace(namespace).labels({'app':
'redis'})
service.spec().selector({'app': 'redis'})
# service.spec().type('NodePort')
service.spec().ports([{
    # 'name': 'redis',
    # 'protocol': 'TCP',
    'port': 6379,
    'targetPort': 6379
}])

ingress = IngressRouteTCP()
ingress.metadata().name('redis')
ingress.metadata().namespace(namespace)
ingress.spec().entryPoints(['redis'])
ingress.spec().routes([{
    'match': 'HostSNI(`*`)',
    'services': [{
        'name': 'redis',
        'port': 6379
    }]
}])
# ingress.debug()

print("=" * 40, "Compose", "=" * 40)
compose = Compose('development')
# compose.add(namespace)
```

```
compose.add(config)
compose.add(statefulSet)
compose.add(service)
compose.add(ingress)
compose.debug()
# compose.save()
compose.delete()
compose.create()
```

检查 redis 是否工作正常

```
neo@Netkiller-iMac ~> kubectl get pods
NAME                       READY   STATUS             RESTARTS
AGE
nginx-88c84c4d8-gb5rg      1/1     Running            1
3d16h
redis-0                    1/1     Running            0
14h
busybox                    0/1     CrashLoopBackOff   256
21h

neo@Netkiller-iMac ~> kubectl exec -it "redis-0" bash
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed
in a future version. Use kubectl exec [POD] -- [COMMAND]
instead.
root@redis-0:/data# redis-cli -a passw0rd
Warning: Using a password with '-a' or '-u' option on the
command line interface may not be safe.
127.0.0.1:6379> set nickname netkiller
OK
127.0.0.1:6379> get nickname
"netkiller"
127.0.0.1:6379>
```

## 3.8. StorageClass

```
storageClass = StorageClass('local-storage')
storageClass.metadata().name('local-storage')
storageClass.provisioner('kubernetes.io/no-provisioner')
storageClass.volumeBindingMode('WaitForFirstConsumer')
# storageClass.json()
# storageClass.debug()
```

```
persistentVolume = PersistentVolume()
persistentVolume.metadata().name('redis').annotations({'pv.ku
bernetes.io/provisioned-by': 'rancher.io/local-path'})
persistentVolume.spec().capacity({'storage': '1Gi'})
persistentVolume.spec().accessModes(['ReadWriteOnce'])
persistentVolume.spec().persistentVolumeReclaimPolicy('Retain
')
persistentVolume.spec().storageClassName('local-path')
# persistentVolume.spec().local('/opt/redis')
persistentVolume.spec().hostPath({'path':
'/var/lib/rancher/k3s/storage/redis', 'type':
'DirectoryOrCreate'})
persistentVolume.spec().nodeAffinity({
    'required':{
      'nodeSelectorTerms':[
       {'matchExpressions':[
         {'key': 'kubernetes.io/hostname',
          'operator': 'In',
          'values':['node1']
            }]
        }]
    }
})
```

## 3.9. 部署 MySQL 到 kubernetes

```
from netkiller.kubernetes import *
namespace = 'default'

config = ConfigMap('mysql')
config.metadata().name('mysql').namespace(namespace).labels({
'app': 'mysql'})
config.data({'mysql.cnf': pss('''\
[mysqld]
max_connections=2048
max_execution_time=120
connect_timeout=120
max_allowed_packet=32M
net_read_timeout=120
net_write_timeout=120
# --wait_timeout=60
# --interactive_timeout=60

sql_mode=STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERR
OR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION
character-set-server=utf8mb4
collation-server=utf8mb4_general_ci
explicit_defaults_for_timestamp=true
max_execution_time=0
''')})
config.data({'MYSQL_ROOT_PASSWORD': '123456',
'MYSQL_DATABASE': 'test',
            'MYSQL_USER': 'test', 'MYSQL_PASSWORD': 'test'})
# config.debug()


storageClassName = 'manual'
persistentVolume = PersistentVolume('mysql-pv')
persistentVolume.metadata().name(
    'mysql-pv').labels({'type': 'local'})
persistentVolume.spec().storageClassName(storageClassName)
persistentVolume.spec().capacity({'storage':
'2Gi'}).accessModes(
    ['ReadWriteOnce']).hostPath({'path': "/var/lib/mysql"})
persistentVolume.debug()

persistentVolumeClaim = PersistentVolumeClaim('mysql-pvc')
persistentVolumeClaim.metadata().name('mysql-pvc')
persistentVolumeClaim.spec().storageClassName(storageClassNam
e)
persistentVolumeClaim.spec().resources({'requests':
{'storage':'2Gi'}})
persistentVolumeClaim.spec().accessModes(
    ['ReadWriteOnce'])
persistentVolumeClaim.debug()
# exit()


statefulSet = StatefulSet()
statefulSet.metadata().name('mysql')
statefulSet.spec().replicas(1)
statefulSet.spec().serviceName('mysql')
statefulSet.spec().selector({'matchLabels': {'app':
'mysql'}})
statefulSet.spec().template().metadata().labels({'app':
'mysql'})
statefulSet.spec().replicas(1)
statefulSet.spec().template().spec().containers().name('mysql
').image(
    'mysql:latest').ports([{
        'name': 'mysql',
        'protocol': 'TCP',
        'containerPort': 3306
    }]).env([{'name': 'MYSQL_ROOT_PASSWORD', 'value':
'123456'}]).volumeMounts([
        {'name': 'config', 'mountPath':
'/etc/mysql/conf.d/mysql.cnf',
            'subPath': 'mysql.cnf'},
        {'name': 'data', 'mountPath': '/var/lib/mysql'}
    ]).imagePullPolicy('IfNotPresent')
statefulSet.spec().template().spec().volumes().name(
    'config').configMap({'name': 'mysql'})
statefulSet.spec().template().spec().volumes().name(
    'data').persistentVolumeClaim('mysql-pvc')
# statefulSet.debug()

service = Service()
service.metadata().name('mysql')
service.metadata().namespace(namespace).labels({'app':
'mysql'})
service.spec().selector({'app': 'mysql'})
service.spec().type('NodePort')
service.spec().ports([{
    'name': 'mysql',
    'protocol': 'TCP',
    'port': 3306,
    'targetPort': 3306
}])

print("=" * 40, "Compose", "=" * 40)
compose = Compose('development')
# compose.add(namespace)
compose.add(config)
compose.add(persistentVolume)
compose.add(persistentVolumeClaim)
compose.add(statefulSet)
compose.add(service)
compose.debug()
# compose.save()
compose.delete()
compose.create()
```

```
neo@Netkiller-iMac ~> kubectl get pods
NAME                        READY   STATUS             RESTARTS
AGE
nginx-88c84c4d8-gb5rg   1/1     Running            1
4d
redis-0                 1/1     Running            0
22h
mysql-0                 1/1     Running            0
9m11s
busybox                 0/1     CrashLoopBackOff   346
29h

neo@Netkiller-iMac ~> kubectl get service
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)
AGE
kubernetes   ClusterIP   10.43.0.1       <none>        443/TCP
12d
nginx        NodePort    10.43.125.134   <none>
80:31656/TCP     4d
redis        ClusterIP   10.43.91.64     <none>        6379/TCP
22h
```

```
mysql        NodePort    10.43.198.188   <none>
3306:32322/TCP   9m22s

neo@Netkiller-iMac ~ [1]> kubectl exec mysql-0 -it bash
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed
in a future version. Use kubectl exec [POD] -- [COMMAND]
instead.

root@mysql-0:/# mysql -uroot -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 9
Server version: 8.0.27 MySQL Community Server - GPL

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

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> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
4 rows in set (0.00 sec)

mysql> create database test;
Query OK, 1 row affected (0.16 sec)

mysql> exit
Bye
root@mysql-0:/#
```

## 3.10. MongoDB

```
import sys
sys.path.insert(0, '/Users/neo/workspace/devops')
from netkiller.kubernetes import *
namespace = 'default'

config = ConfigMap('mongo')
config.metadata().name('mongo').namespace(namespace).labels({
'app': 'mongo'})
config.data({'mongod.cnf': pss('''\
# mongod.conf

# for documentation of all options, see:
#   http://docs.mongodb.org/manual/reference/configuration-
options/

# Where and how to store data.
storage:
  dbPath: /var/lib/mongo
  journal:
    enabled: true
#  engine:
#  wiredTiger:

# where to write logging data.
systemLog:
  destination: file
  logAppend: true
  path: /var/log/mongodb/mongod.log

# network interfaces
net:
  port: 27017
  bindIp: 0.0.0.0


# how the process runs
processManagement:
  timeZoneInfo: /usr/share/zoneinfo
security:
  authorization: enabled

#operationProfiling:

#replication:

#sharding:

## Enterprise-Only Options:

#auditLog:

#snmp:
''')})
config.data({'mongo_ROOT_PASSWORD': '123456',
'mongo_DATABASE': 'test',
            'mongo_USER': 'test', 'mongo_PASSWORD': 'test'})
# config.debug()


storageClassName = 'manual'
persistentVolume = PersistentVolume('mongo-pv')
persistentVolume.metadata().name(
    'mongo-pv').labels({'type': 'local'})
persistentVolume.spec().storageClassName(storageClassName)
persistentVolume.spec().capacity({'storage':
'2Gi'}).accessModes(
    ['ReadWriteOnce']).hostPath({'path': "/var/lib/mongodb"})
persistentVolume.debug()

persistentVolumeClaim = PersistentVolumeClaim('mongo-pvc')
persistentVolumeClaim.metadata().name('mongo-pvc')
persistentVolumeClaim.spec().storageClassName(storageClassNam
e)
persistentVolumeClaim.spec().resources({'requests':
{'storage':'2Gi'}})
persistentVolumeClaim.spec().accessModes(
    ['ReadWriteOnce'])
persistentVolumeClaim.debug()
# exit()


statefulSet = StatefulSet()
statefulSet.metadata().name('mongo')
statefulSet.spec().replicas(1)
statefulSet.spec().serviceName('mongo')
statefulSet.spec().selector({'matchLabels': {'app':
'mongo'}})
statefulSet.spec().template().metadata().labels({'app':
'mongo'})
statefulSet.spec().replicas(1)
statefulSet.spec().template().spec().containers().name('mongo
').image(
    'mongo:latest').ports([{
        'name': 'mongo',
        'protocol': 'TCP',
        'containerPort': 27017
    }]).env([
        {'name': 'TZ', 'value': 'Asia/Shanghai'},
        {'name': 'LANG', 'value': 'en_US.UTF-8'},
        {'name': 'MONGO_INITDB_DATABASE', 'value': 'admin'},
        {'name': 'MONGO_INITDB_ROOT_USERNAME', 'value':
'admin'},
        {'name': 'MONGO_INITDB_ROOT_PASSWORD', 'value':
'A8nWiX7vitsqOsqoWVnTtv4BDG6uMbexYX9s'}
    ]).volumeMounts([
        {'name': 'config', 'mountPath': '/etc/mongod.conf',
            'subPath': 'mongo.cnf'},
        {'name': 'data', 'mountPath': '/var/lib/mongodb'}
    ]).imagePullPolicy('IfNotPresent')
statefulSet.spec().template().spec().volumes().name(
    'config').configMap({'name': 'mongo'})
statefulSet.spec().template().spec().volumes().name(
    'data').persistentVolumeClaim('mongo-pvc')
# statefulSet.debug()
# exit()

service = Service()
service.metadata().name('mongo')
service.metadata().namespace(namespace).labels({'app':
'mongo'})
service.spec().selector({'app': 'mongo'})
service.spec().type('NodePort')
service.spec().ports([{
    'name': 'mongo',
    'protocol': 'TCP',
    'port': 27017,
    'targetPort': 27017
}])
```

```
ingress = IngressRouteTCP()
ingress.metadata().name('mongo')
ingress.metadata().namespace(namespace)
ingress.spec().entryPoints(['mongo'])
ingress.spec().routes([{
    'match': 'HostSNI(`*`)',
    'services': [{
        'name': 'mongo',
        'port': 27017,
    }]
}])
# ingress.debug()

print("=" * 40, "Compose", "=" * 40)
compose = Compose('development')
# compose.add(namespace)
compose.add(config)
compose.add(persistentVolume)
compose.add(persistentVolumeClaim)
compose.add(statefulSet)
compose.add(service)
compose.add(ingress)
compose.debug()
# compose.save()
compose.delete()
compose.create()
```

进入容器，检查是否工作正常

```
neo@Netkiller-iMac ~> kubectl get all
NAME                            READY   STATUS              RESTARTS
AGE
pod/mongo-0                     1/1     Running             0
164m
pod/mysql-0                     1/1     Running             0
149m
pod/nginx-88c84c4d8-dwz9x       1/1     Running             0
147m
pod/redis-0                     1/1     Running             0
```

```
132m
pod/busybox                 0/1     CrashLoopBackOff   436
2d2h

NAME                 TYPE        CLUSTER-IP      EXTERNAL-IP
PORT(S)           AGE
service/kubernetes   ClusterIP   10.43.0.1       <none>
443/TCP           13d
service/mongo        NodePort    10.43.135.49    <none>
27017:32598/TCP   164m
service/mysql        NodePort    10.43.186.2     <none>
3306:32440/TCP    149m
service/nginx        NodePort    10.43.235.124   <none>
80:32124/TCP      147m
service/redis        NodePort    10.43.134.73    <none>
6379:30376/TCP    133m

NAME                    READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/nginx   1/1     1            1           147m

NAME                              DESIRED   CURRENT   READY
AGE
replicaset.apps/nginx-88c84c4d8   1         1         1
147m

NAME                     READY   AGE
statefulset.apps/mongo   1/1     164m
statefulset.apps/mysql   1/1     149m
statefulset.apps/redis   1/1     133m

neo@Netkiller-iMac ~> kubectl exec -it mongo-0 -- bash
root@mongo-0:/# ps ax
  PID TTY      STAT   TIME COMMAND
    1 ?        Ssl    1:43 mongod --auth --bind_ip_all
  133 pts/0    Ss     0:00 bash
  141 pts/0    R+     0:00 ps ax


root@mongo-0:/# mongosh
mongodb://admin:A8nWiX7vitsqOsqoWVnTtv4BDG6uMbexYX9s@localhost/
admin


Current Mongosh Log ID: 61e7acde14e7858c6d5dfcf6
Connecting to:
```

```
mongodb://<credentials>@localhost/admin?
directConnection=true&serverSelectionTimeoutMS=2000
Using MongoDB:          5.0.5
Using Mongosh:          1.1.7

For mongosh info see: https://docs.mongodb.com/mongodb-shell/


To help improve our products, anonymous usage data is collected
and sent to MongoDB periodically
(https://www.mongodb.com/legal/privacy-policy).
You can opt-out by running the disableTelemetry() command.

------
   The server generated these startup warnings when booting:
   2022-01-19T11:30:22.969+08:00: Using the XFS filesystem is
strongly recommended with the WiredTiger storage engine. See
http://dochub.mongodb.org/core/prodnotes-filesystem
------

admin>
Browserslist: caniuse-lite is outdated. Please run:
npx browserslist@latest --update-db

Why you should do it regularly:
https://github.com/browserslist/browserslist#browsers-data-
updating

admin> use test
switched to db test
test> db.createCollection("mycollection")
{ ok: 1 }
test> exit
root@mongo-0:/# exit
exit
```

端口转发

```


neo@Netkiller-iMac ~> kubectl port-forward --address 0.0.0.0
```

```
service/mongo 27017
Forwarding from 0.0.0.0:27017 -> 27017
```

远程登陆

```
[root@gitlab ~]# mongo
mongodb://admin:A8nWiX7vitsqOsqoWVnTtv4BDG6uMbexYX9s@192.168.30
.131/admin
MongoDB shell version v5.0.5
connecting to: mongodb://192.168.30.131:27017/admin?
compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("22b2d5ec-9643-492e-
93df-12bb81ba21f4") }
MongoDB server version: 5.0.5
================
Warning: the "mongo" shell has been superseded by "mongosh",
which delivers improved usability and compatibility.The "mongo"
shell has been deprecated and will be removed in
an upcoming release.
For installation instructions, see
https://docs.mongodb.com/mongodb-shell/install/
================
---
The server generated these startup warnings when booting:
        2022-01-19T11:30:22.969+08:00: Using the XFS filesystem
is strongly recommended with the WiredTiger storage engine. See
http://dochub.mongodb.org/core/prodnotes-filesystem
---
---
        Enable MongoDB's free cloud-based monitoring service,
which will then receive and display
        metrics about your deployment (disk utilization, CPU,
operation statistics, etc).

        The monitoring data will be available on a MongoDB
website with a unique URL accessible to you
        and anyone you share the URL with. MongoDB may use this
information to make product
        improvements and to suggest MongoDB products and
deployment options to you.
```

```
	To enable free monitoring, run the following command:
db.enableFreeMonitoring()
	To permanently disable this reminder, run the following
command: db.disableFreeMonitoring()
---
> show databases
admin   0.000GB
config  0.000GB
local   0.000GB
test    0.000GB
> use test
switched to db test
> show tables
mycollection
> exit
bye
```

## 3.11. Nacos

### 单节点部署

```
import sys
sys.path.insert(0, '/Users/neo/workspace/devops')
from netkiller.kubernetes import *

namespace = 'default'

# namespace = Namespace()
# namespace.metadata().name(namespace)
# namespace.metadata().namespace(namespace)
# namespace.debug()

config = ConfigMap('nacos')
config.apiVersion('v1')
config.metadata().name('nacos').namespace(namespace)
config.from_file('custom.properties',
'nacos/init.d/custom.properties')
```

```
config.data({'application.properties':pss('''\
    # spring
    server.servlet.contextPath=/nacos
    server.contextPath=/nacos
    server.port=8848
    spring.datasource.platform=mysql
    # nacos.cmdb.dumpTaskInterval=3600
    # nacos.cmdb.eventTaskInterval=10
    # nacos.cmdb.labelTaskInterval=300
    # nacos.cmdb.loadDataAtStart=false
    db.num=1
    # db.url.0=jdbc:mysql://mysql-0.mysql:3306/nacos?
characterEncoding=utf8&connectTimeout=30000&socketTimeout=300
00&autoReconnect=true&useSSL=false&serverTimezone=GMT%2B8
    # db.url.1=jdbc:mysql://mysql-0.mysql:3306/nacos?
characterEncoding=utf8&connectTimeout=30000&socketTimeout=300
00&autoReconnect=true&useSSL=false&serverTimezone=GMT%2B8
    db.url.0=jdbc:mysql://192.168.30.12:3306/nacos?
characterEncoding=utf8&connectTimeout=30000&socketTimeout=300
00&autoReconnect=true&useSSL=false&serverTimezone=GMT%2B8
    db.url.1=jdbc:mysql://192.168.30.12:3306/nacos?
characterEncoding=utf8&connectTimeout=30000&socketTimeout=300
00&autoReconnect=true&useSSL=false&serverTimezone=GMT%2B8
    # db.url.1=jdbc:mysql://mysql-
0.mysql.default.svc.cluster.local:3306/nacos?
characterEncoding=utf8&connectTimeout=3000&socketTimeout=3000
&autoReconnect=true&useSSL=false&serverTimezone=Asia/Shanghai
    db.user=nacos
    db.password=nacos
    ### The auth system to use, currently only 'nacos' is
supported:
    nacos.core.auth.system.type=nacos


    ### The token expiration in seconds:

nacos.core.auth.default.token.expire.seconds=${NACOS_AUTH_TOK
EN_EXPIRE_SECONDS:18000}

    ### The default token:

nacos.core.auth.default.token.secret.key=${NACOS_AUTH_TOKEN:S
ecretKey012345678901234567890123456789012345678901234567890123
3456789}
```

```
    ### Turn on/off caching of auth information. By turning
on this switch, the update of auth information would have a
15 seconds delay.

nacos.core.auth.caching.enabled=${NACOS_AUTH_CACHE_ENABLE:fal
se}

nacos.core.auth.enable.userAgentAuthWhite=${NACOS_AUTH_USER_A
GENT_AUTH_WHITE_ENABLE:false}

nacos.core.auth.server.identity.key=${NACOS_AUTH_IDENTITY_KEY
:serverIdentity}

nacos.core.auth.server.identity.value=${NACOS_AUTH_IDENTITY_V
ALUE:security}

server.tomcat.accesslog.enabled=${TOMCAT_ACCESSLOG_ENABLED:fa
lse}
    server.tomcat.accesslog.pattern=%h %l %u %t "%r" %s %b %D
    # default current work dir
    server.tomcat.basedir=
    ## spring security config
    ### turn off security

nacos.security.ignore.urls=${NACOS_SECURITY_IGNORE_URLS:/,/er
ror,/**/*.css,/**/*.js,/**/*.html,/**/*.map,/**/*.svg,/**/*.p
ng,/**/*.ico,/console-
fe/public/**,/v1/auth/**,/v1/console/health/**,/actuator/**,/
v1/console/server/**}
    # metrics for elastic search
    management.metrics.export.elastic.enabled=false
    management.metrics.export.influx.enabled=false

    nacos.naming.distro.taskDispatchThreadCount=10
    nacos.naming.distro.taskDispatchPeriod=200
    nacos.naming.distro.batchSyncKeyCount=1000
    nacos.naming.distro.initDataRatio=0.9
    nacos.naming.distro.syncRetryDelay=5000
    nacos.naming.data.warmup=true
'''
)})
# config.save()
# config.debug()

# statefulSet = StatefulSet()
```

```
deployment = StatefulSet()
deployment.apiVersion('apps/v1')
deployment.metadata().name('nacos').labels(
    {'app': 'nacos'}).namespace(namespace)
deployment.spec().replicas(1)
deployment.spec().serviceName('nacos')
deployment.spec().selector({'matchLabels': {'app': 'nacos'}})
deployment.spec().template().metadata().labels({'app':
'nacos'})
deployment.spec().template().spec().containers().name('nacos'
).image(
    'nacos/nacos-server:2.0.3').env([
        {'name': 'TZ', 'value': 'Asia/Shanghai'},
        {'name': 'LANG', 'value': 'en_US.UTF-8'},
        {'name': 'PREFER_HOST_MODE', 'value': 'hostname'},
        {'name': 'MODE', 'value': 'standalone'},
        {'name': 'SPRING_DATASOURCE_PLATFORM', 'value':
'mysql'},
        {'name': 'JVM_XMX', 'value': '4g'},
        {'name': 'NACOS_DEBUG', 'value': 'true'},
        {'name': 'TOMCAT_ACCESSLOG_ENABLED', 'value':
'true'},
    ]).ports([
        {'containerPort': 8848},
        {'containerPort': 9848},
        {'containerPort': 9555}
    ]).volumeMounts([
        {'name': 'config', 'mountPath':
'/home/nacos/conf/custom.properties', 'subPath':
'custom.properties'},
        {'name': 'config', 'mountPath':
'/home/nacos/conf/application.properties', 'subPath':
'application.properties'}
]).resources({'limits':{'memory': "4Gi"}, 'requests':
{'memory': "2Gi"}})
#
deployment.spec().template().spec().securityContext({'sysctls
':[{'name': 'fs.file-max', 'value': '60000'}]})
deployment.spec().template().spec().volumes().name(
    'config').configMap({'name': 'nacos'})
# deployment.debug()
# deployment.json()

service = Service()
service.metadata().name('nacos')
service.metadata().namespace(namespace)
service.spec().selector({'app': 'nacos'})
service.spec().type('ClusterIP')
service.spec().ports([
    {'name': 'http', 'protocol': 'TCP', 'port': 8848,
'targetPort': 8848},
    {'name': 'rpc', 'protocol': 'TCP', 'port': 9848,
'targetPort': 9848},
    # {'name': 'http', 'protocol': 'TCP', 'port': 9555,
'targetPort': 9555}
])

print("=" * 40, "Compose", "=" * 40)
compose = Compose('development')
# compose.add(namespace)
compose.add(config)
compose.add(deployment)
compose.add(service)
# compose.debug()
compose.save()
compose.delete()
compose.create()

print("=" * 40, "Busybox", "=" * 40)
os.system("sleep 5")
for cmd in ['kubectl get secret tls', 'kubectl get
configmap', 'kubectl get pods', 'kubectl get service',
'kubectl get deployment', 'kubectl get ingress']:
    os.system(cmd)
    print("-" * 50)
```

### 集群部署

```
import sys
sys.path.insert(0, '/Users/neo/workspace/devops')
from netkiller.kubernetes import *

namespace = 'default'
```

```
# namespace = Namespace()
# namespace.metadata().name(namespace)
# namespace.metadata().namespace(namespace)
# namespace.debug()

config = ConfigMap('nacos')
config.apiVersion('v1')
config.metadata().name('nacos').namespace(namespace)
config.from_file('custom.properties',
'nacos/init.d/custom.properties')
config.data({'application.properties':pss('''\
    # spring
    server.servlet.contextPath=/nacos
    server.contextPath=/nacos
    server.port=8848
    spring.datasource.platform=mysql
    # nacos.cmdb.dumpTaskInterval=3600
    # nacos.cmdb.eventTaskInterval=10
    # nacos.cmdb.labelTaskInterval=300
    # nacos.cmdb.loadDataAtStart=false
    db.num=1
    # db.url.0=jdbc:mysql://mysql-0.mysql:3306/nacos?
characterEncoding=utf8&connectTimeout=30000&socketTimeout=300
00&autoReconnect=true&useSSL=false&serverTimezone=GMT%2B8
    # db.url.1=jdbc:mysql://mysql-0.mysql:3306/nacos?
characterEncoding=utf8&connectTimeout=30000&socketTimeout=300
00&autoReconnect=true&useSSL=false&serverTimezone=GMT%2B8
    # db.url.1=jdbc:mysql://mysql-
0.mysql.default.svc.cluster.local:3306/nacos?
characterEncoding=utf8&connectTimeout=3000&socketTimeout=3000
&autoReconnect=true&useSSL=false&serverTimezone=Asia/Shanghai
    db.user=nacos
    db.password=nacos
    ### The auth system to use, currently only 'nacos' is
supported:
    nacos.core.auth.system.type=nacos


    ### The token expiration in seconds:

nacos.core.auth.default.token.expire.seconds=${NACOS_AUTH_TOK
EN_EXPIRE_SECONDS:18000}

    ### The default token:
```

```
nacos.core.auth.default.token.secret.key=${NACOS_AUTH_TOKEN:S
ecretKey012345678901234567890123456789012345678901234567890123456789012
3456789}

    ### Turn on/off caching of auth information. By turning
on this switch, the update of auth information would have a
15 seconds delay.

nacos.core.auth.caching.enabled=${NACOS_AUTH_CACHE_ENABLE:fal
se}

nacos.core.auth.enable.userAgentAuthWhite=${NACOS_AUTH_USER_A
GENT_AUTH_WHITE_ENABLE:false}

nacos.core.auth.server.identity.key=${NACOS_AUTH_IDENTITY_KEY
:serverIdentity}

nacos.core.auth.server.identity.value=${NACOS_AUTH_IDENTITY_V
ALUE:security}

server.tomcat.accesslog.enabled=${TOMCAT_ACCESSLOG_ENABLED:fa
lse}
    server.tomcat.accesslog.pattern=%h %l %u %t "%r" %s %b %D
    # default current work dir
    server.tomcat.basedir=
    ## spring security config
    ### turn off security

nacos.security.ignore.urls=${NACOS_SECURITY_IGNORE_URLS:/,/er
ror,/**/*.css,/**/*.js,/**/*.html,/**/*.map,/**/*.svg,/**/*.p
ng,/**/*.ico,/console-
fe/public/**,/v1/auth/**,/v1/console/health/**,/actuator/**,/
v1/console/server/**}
    # metrics for elastic search
    management.metrics.export.elastic.enabled=false
    management.metrics.export.influx.enabled=false

    nacos.naming.distro.taskDispatchThreadCount=10
    nacos.naming.distro.taskDispatchPeriod=200
    nacos.naming.distro.batchSyncKeyCount=1000
    nacos.naming.distro.initDataRatio=0.9
    nacos.naming.distro.syncRetryDelay=5000
    nacos.naming.data.warmup=true
'''
```

```
)})
# config.save()
# config.debug()

statefulSet = StatefulSet()
statefulSet = StatefulSet()
statefulSet.apiVersion('apps/v1')
statefulSet.metadata().name('nacos').labels(
    {'app': 'nacos'}).namespace(namespace)
statefulSet.spec().replicas(3)
statefulSet.spec().serviceName('nacos')
statefulSet.spec().selector({'matchLabels': {'app':
'nacos'}})
statefulSet.spec().template().metadata().labels({'app':
'nacos'})
statefulSet.spec().template().spec().containers().name('nacos
').image(
    'nacos/nacos-server:latest').env([
        {'name': 'TZ', 'value': 'Asia/Shanghai'},
        {'name': 'LANG', 'value': 'en_US.UTF-8'},
        {'name': 'PREFER_HOST_MODE', 'value': 'hostname'},
        # {'name': 'MODE', 'value': 'standalone'},

        {'name': 'MODE', 'value': 'cluster'},
        {'name': 'NACOS_REPLICAS', 'value': '3'},
        {'name': 'NACOS_SERVERS', 'value': 'nacos-
0.nacos.default.svc.cluster.local:8848 nacos-
1.nacos.default.svc.cluster.local:8848 nacos-
2.nacos.default.svc.cluster.local:8848'},


        {'name': 'SPRING_DATASOURCE_PLATFORM', 'value':
'mysql'},
        {'name': 'MYSQL_SERVICE_HOST', 'value': 'mysql-
0.mysql.default.svc.cluster.local'},
        {'name': 'MYSQL_SERVICE_PORT', 'value': '3306'},
        {'name': 'MYSQL_SERVICE_DB_NAME', 'value': 'nacos'},
        {'name': 'MYSQL_SERVICE_USER', 'value': 'nacos'},
        {'name': 'MYSQL_SERVICE_PASSWORD', 'value': 'nacos'},
        {'name': 'MYSQL_SERVICE_DB_PARAM', 'value':
'characterEncoding=utf8&connectTimeout=1000&socketTimeout=300
0&autoReconnect=true&useSSL=false&serverTimezone=Asia/Shangha
i'},
        {'name': 'JVM_XMX', 'value': '4g'},
        {'name': 'NACOS_DEBUG', 'value': 'true'},
        {'name': 'TOMCAT_ACCESSLOG_ENABLED', 'value':
'true'},
    ]).ports([
        {'containerPort': 8848},
        {'containerPort': 9848},
        {'containerPort': 9555}
    ]).volumeMounts([
        {'name': 'config', 'mountPath':
'/home/nacos/conf/custom.properties', 'subPath':
'custom.properties'},
        # {'name': 'config', 'mountPath':
'/home/nacos/conf/application.properties', 'subPath':
'application.properties'}
]).resources({'limits':{'memory': "4Gi"}, 'requests':
{'memory': "2Gi"}})
#
statefulSet.spec().template().spec().securityContext({'sysctl
s':[{'name': 'fs.file-max', 'value': '60000'}]})
statefulSet.spec().template().spec().volumes().name(
    'config').configMap({'name': 'nacos'})
statefulSet.debug()
# statefulSet.json()

service = Service()
service.metadata().name('nacos')
service.metadata().namespace(namespace)
service.spec().selector({'app': 'nacos'})
service.spec().type('ClusterIP')
service.spec().ports([
    {'name': 'http', 'protocol': 'TCP', 'port': 8848,
'targetPort': 8848},
    {'name': 'rpc', 'protocol': 'TCP', 'port': 9848,
'targetPort': 9848},
    # {'name': 'http', 'protocol': 'TCP', 'port': 9555,
'targetPort': 9555}
])

print("=" * 40, "Compose", "=" * 40)
compose = Compose('development')
# compose.add(namespace)
compose.add(config)
compose.add(statefulSet)
compose.add(service)
# compose.debug()
compose.save()
compose.delete()
compose.create()

print("=" * 40, "Busybox", "=" * 40)
os.system("sleep 5")
for cmd in ['kubectl get secret tls', 'kubectl get
configmap', 'kubectl get pods', 'kubectl get service',
'kubectl get statefulset', 'kubectl get ingress']:
    os.system(cmd)
    print("-" * 50)
```

## Ingress 部署

```
ingress = Ingress()
ingress.apiVersion('networking.k8s.io/v1')
ingress.metadata().name('nginx')
ingress.metadata().namespace(namespace)
ingress.metadata().annotations({'ingress.kubernetes.io/ssl-
redirect': "true"})
ingress.spec().tls(
    [{'hosts': ['www.netkiller.cn',
'job.netkiller.cn','admin.netkiller.cn','nacos.netkiller.cn',
'test.netkiller.cn','cloud.netkiller.cn'],
'secretName':'tls'}])
ingress.spec().rules([
    {
        'host': 'www.netkiller.cn',
        'http': {
            'paths': [{
                'path': '/',
                'pathType': 'Prefix',
                'backend': {
                    'service': {
                        'name': 'nginx',
                        'port': {
                            'number': 80
                        }
                    }
                }
            }]
        },
    },
    {
        'host': 'nacos.netkiller.cn',
        'http': {
            'paths': [{
                'path': '/',
                'pathType': 'Prefix',
                'backend': {
                    'service': {
                        'name': 'nacos',
                        'port': {
                            'number': 8848
                        }
                    }
                }
            }]
        },
    }
])
```

测试地址 https://nacos.netkiller.cn/nacos/

## 3.12. Redis

```
import sys, os

sys.path.insert(0, '/Users/neo/workspace/devops')
from netkiller.kubernetes import *

namespace = 'default'

config = ConfigMap('redis')
config.apiVersion('v1')
config.metadata().name('redis').namespace(namespace)
# config.from_file('redis.conf', 'redis.conf')
config.data({
    'redis.conf':
    pss('''\
    pidfile /var/lib/redis/redis.pid
    dir /data
    port 6379
    bind 0.0.0.0
    appendonly yes
    protected-mode yes
    requirepass passw0rd
    maxmemory 2mb
    maxmemory-policy allkeys-lru
''')
})

# config.debug()

persistentVolumeClaim = PersistentVolumeClaim()
persistentVolumeClaim.metadata().name('redis')
# persistentVolumeClaim.metadata().labels({'app': 'redis',
'type': 'longhorn'})
# persistentVolumeClaim.spec().storageClassName('longhorn')
persistentVolumeClaim.spec().storageClassName('local-path')
persistentVolumeClaim.spec().accessModes(['ReadWriteOnce'])
persistentVolumeClaim.spec().resources({'requests':
{'storage': '2Gi'}})

limits = {
    'limits': {
        'cpu': '200m',
        'memory': '2Gi'
    },
    'requests': {
        'cpu': '200m',
        'memory': '1Gi'
    }
}

livenessProbe = {
    'tcpSocket': {
        'port': 6379
    },
    'initialDelaySeconds': 30,
    'failureThreshold': 3,
    'periodSeconds': 10,
    'successThreshold': 1,
    'timeoutSeconds': 5
}
readinessProbe = {
    'tcpSocket': {
        'port': 6379
    },
    'initialDelaySeconds': 5,
    'failureThreshold': 3,
    'periodSeconds': 10,
    'successThreshold': 1,
    'timeoutSeconds': 5
}

statefulSet = StatefulSet()
statefulSet.metadata().name('redis').labels({'app': 'redis'})
statefulSet.spec().replicas(1)
statefulSet.spec().serviceName('redis')
statefulSet.spec().selector({'matchLabels': {'app':
'redis'}})
statefulSet.spec().template().metadata().labels({'app':
'redis'})
# statefulSet.spec().template().spec().nodeName('master')
statefulSet.spec().template().spec().containers(
).name('redis').image('redis:latest').ports([{
    'containerPort': 6379
}]).volumeMounts([
    {
        'name': 'data',
        'mountPath': '/data'
    },
    {
        'name': 'config',
        'mountPath': '/usr/local/etc/redis.conf',
        'subPath': 'redis.conf'
    },
]).resources(None).livenessProbe(livenessProbe).readinessProb
e(readinessProbe)
# .command(            ["sh -c redis-server
/usr/local/etc/redis.conf"])
statefulSet.spec().template().spec().volumes([{
    'name': 'data',
    'persistentVolumeClaim': {
        'claimName': 'redis'
    }
}, {
    'name': 'config',
    'configMap': {
        'name': 'redis'
    }
}])
# statefulSet.spec().volumeClaimTemplates([{
#       'metadata':{'name': 'data'},
#     'spec':{
#       'accessModes': [ "ReadWriteOnce" ],
#       'storageClassName': "local-path",
#       'resources':{'requests':{'storage': '2Gi'}}
#       }
# }])

service = Service()
service.metadata().name('redis')
service.metadata().namespace(namespace)
service.spec().selector({'app': 'redis'})
service.spec().type('NodePort')
service.spec().ports([{
    'name': 'redis',
    'protocol': 'TCP',
    'port': 6379,
    'targetPort': 6379
}])
# service.debug()

compose = Compose('development')
compose.add(config)
compose.add(persistentVolumeClaim)
compose.add(statefulSet)
compose.add(service)
# compose.debug()

# kubeconfig = '/Volumes/Data/kubernetes/test'
kubeconfig = os.path.expanduser('~/workspace/ops/k3s.yaml')

kubernetes = Kubernetes(kubeconfig)
kubernetes.compose(compose)
kubernetes.main()
```

## 3.13. Kubernetes 部署 kube-explorer 图形化界面

```
import os
import sys
import time

sys.path.insert(0, '/Users/neo/workspace/devops')

from netkiller.kubernetes import *

namespace = 'default'
name = 'kube-explorer'
labels = {'app': name}
annotations = {}
replicas = 1
containerPort = 80
image = 'cnrancher/kube-explorer:latest'
monitor = '/dashboard'
livenessProbe = {}
readinessProbe = {}
limits = {}

compose = Compose('test', 'k3s.yaml')

config = ConfigMap()
config.metadata().name(name).namespace(namespace)
config.from_file('k3s.yaml', 'k3s.yaml')
compose.add(config)

deployment = Deployment()
deployment.metadata().name(name).labels(labels).namespace(nam
espace)
deployment.metadata().annotations(annotations)
deployment.spec().replicas(replicas)
deployment.spec().progressDeadlineSeconds(10)
deployment.spec().revisionHistoryLimit(10)
deployment.spec().selector({'matchLabels': {'app': name}})
#
deployment.spec().strategy().type('RollingUpdate').rollingUpd
ate(1, 0)
deployment.spec().template().metadata().labels({'app': name})
```

```
livenessProbe = {
    'failureThreshold': 3,
    'httpGet': {
        'path': monitor,
        'port': containerPort,
        'scheme': 'HTTP'
    },
    'initialDelaySeconds': 60,
    'periodSeconds': 10,
    'successThreshold': 1,
    'timeoutSeconds': 5
}
readinessProbe = {
    'failureThreshold': 3,
    'httpGet': {
        'path': monitor,
        'port': containerPort,
        'scheme': 'HTTP'
    },
    'initialDelaySeconds': 30,
    'periodSeconds': 10,
    'successThreshold': 1,
    'timeoutSeconds': 5
}

# limits = {'limits': {
#       # 'cpu': '500m',
#       'memory': '1Gi'}, 'requests': {
#               # 'cpu': '500m',
#       'memory': '1Gi'}}

deployment.spec().template().spec().containers().name(name).i
mage(image).ports(
    [{
        'containerPort': containerPort
    }]).imagePullPolicy('IfNotPresent').volumeMounts([
        {
            'name': 'config',
            'mountPath': '/etc/rancher/k3s/k3s.yaml',
            'subPath': 'k3s.yaml'
        },

]).resources(limits).livenessProbe(livenessProbe).readinessPr
obe(
        readinessProbe).env([
            # {
            #     'name': 'CONTEXT',
            #     'value': '/dashboard'
            # },
            {
                'name': 'KUBECONFIG',
                'value': '/etc/rancher/k3s/k3s.yaml'
            },
        ]).command([
            'kube-explorer', '--
kubeconfig=/etc/rancher/k3s/k3s.yaml',
            '--http-listen-port=80', '--https-listen-port=0'
        ])
# ,'--ui-path=/dashboard'
# --context value              [$CONTEXT]
deployment.spec().template().spec().restartPolicy(Define.rest
artPolicy.Always)
# deployment.spec().template().spec().nodeSelector({'group':
'backup'})
#
deployment.spec().template().spec().dnsPolicy(Define.dnsPolic
y.ClusterFirst)
deployment.spec().template().spec().volumes([{
    'name': 'config',
    'configMap': {
        'name': name
    }
}])
compose.add(deployment)

service = Service()
service.metadata().namespace(namespace)
service.spec().selector({'app': name})
service.metadata().name(name)
service.spec().type(Define.Service.ClusterIP)
service.spec().ports({
    'name': 'http',
    'protocol': 'TCP',
    'port': 80,
    'targetPort': containerPort
})
compose.add(service)

ingress = Ingress()
```

```
ingress.apiVersion('networking.k8s.io/v1')
ingress.metadata().name(name)
ingress.metadata().namespace(namespace)
#
ingress.metadata().annotations({'kubernetes.io/ingress.class'
: 'nginx'})
pathType = Define.Ingress.pathType.Prefix

ingress.spec().rules([{
    # 'host': vhost['host'],
    'http': {
        'paths': [{
            'path': '/dashboard/',
            'pathType': pathType,
            'backend': {
                'service': {
                    'name': name,
                    'port': {
                        'number': 80
                    }
                }
            }
        }, {
            'path': '/v1/',
            'pathType': pathType,
            'backend': {
                'service': {
                    'name': name,
                    'port': {
                        'number': 80
                    }
                }
            }
        }, {
            'path': '/k8s/',
            'pathType': pathType,
            'backend': {
                'service': {
                    'name': name,
                    'port': {
                        'number': 80
                    }
                }
            }
        }, {
            'path': '/apis/',
            'pathType': pathType,
            'backend': {
                'service': {
                    'name': name,
                    'port': {
                        'number': 80
                    }
                }
            }
        }, {
            'path': '/api/',
            'pathType': pathType,
            'backend': {
                'service': {
                    'name': name,
                    'port': {
                        'number': 80
                    }
                }
            }
        }]
    }
}])

compose.add(ingress)

kubernetes = Kubernetes()
kubernetes.compose(compose)

# kubernetes.debug()
# kubernetes.environment({'test': 'k3s.yaml', 'grey':
'grey.yaml'})
kubernetes.main()
```

## 3.14. ELK

### Elasticsearch

```
from doctest import master
import sys, os

sys.path.insert(0, '/Users/neo/workspace/devops')
from netkiller.kubernetes import *

# https://blog.csdn.net/weihua831/article/details/126172591
# https://www.jianshu.com/p/05c93cf45971

namespace = 'default'
# image = 'docker.elastic.co/elasticsearch/elasticsearch:8.4.1'
image = 'elasticsearch:8.4.1'

compose = Compose('development')

config = ConfigMap('elasticsearch')
config.apiVersion('v1')
config.metadata().name('elasticsearch').namespace(namespace).la
bels({
    'app':
    'elasticsearch',
    'role':
    'master'
})
# config.from_file('redis.conf', 'redis.conf')
config.data({
    'elasticsearch.yml':
    pss('''\
cluster.name: kubernetes-cluster
node.name: ${HOSTNAME}
discovery.seed_hosts:
  - elasticsearch-master-0
cluster.initial_master_nodes:
  - elasticsearch-master-
0.elasticsearch.default.svc.cluster.local
  - elasticsearch-data-0.elasticsearch-
data.default.svc.cluster.local
  - elasticsearch-data-1.elasticsearch-
data.default.svc.cluster.local
  - elasticsearch-data-2.elasticsearch-
data.default.svc.cluster.local
```

```
network.host: 0.0.0.0
transport.profiles.default.port: 9300

xpack.security.enabled: false
xpack.monitoring.collection.enabled: true
''')
})
config.debug()
compose.add(config)

service = Service()
service.metadata().name('elasticsearch')
service.metadata().namespace(namespace)
service.spec().selector({'app': 'elasticsearch', 'role':
'master'})
# service.spec().type('NodePort')
service.spec().ports([{
    'name': 'restful',
    'protocol': 'TCP',
    'port': 9200,
    'targetPort': 9200
}, {
    'name': 'transport',
    'protocol': 'TCP',
    'port': 9300,
    'targetPort': 9300
}])
# service.debug()
compose.add(service)

service = Service()
service.metadata().name('elasticsearch-data').labels({
    'app': 'elasticsearch',
    'role': 'data'
})
service.metadata().namespace(namespace)
service.spec().selector({'app': 'elasticsearch', 'role':
'data'})
# service.spec().type('NodePort')
service.spec().ports([
    # {'name': 'restful', 'protocol': 'TCP', 'port': 9200,
'targetPort': 9200},
    {
        'name': 'transport',
        'protocol': 'TCP',
        'port': 9300,
        'targetPort': 9300
    }
])
# service.debug()
compose.add(service)

limits = {
    'limits': {
        # 'cpu': '500m',
        'memory': '1Gi'
    },
    'requests': {
        # 'cpu': '500m',
        'memory': '1Gi'
    }
}

env = [
    {
        'name': 'TZ',
        'value': 'Asia/Shanghai'
    },
    {
        'name': 'LANG',
        'value': 'en_US.UTF-8'
    },
    {
        'name': 'cluster.name',
        'value': 'kubernetes-cluster'
    },
    {
        'name': 'node.name',
        'valueFrom': {
            'fieldRef': {
                'fieldPath': 'metadata.name'
            }
        }
    },
    {
        'name': 'cluster.initial_master_nodes',
        'value': 'elasticsearch-master-0,elasticsearch-master-
1'
    },
    {
        'name':
        'discovery.seed_hosts',
        'value':
        'elasticsearch-master-
0.elasticsearch.default.svc.cluster.local,elasticsearch-data-
0.elasticsearch-data.default.svc.cluster.local,elasticsearch-
data-1.elasticsearch-
data.default.svc.cluster.local,elasticsearch-data-
2.elasticsearch-data.default.svc.cluster.local'
    },
    {
        'name': 'xpack.security.enabled',
        'value': 'false'
    },
    {
        'name': 'ES_JAVA_OPTS',
        'value': '-Xms2048m -Xmx2048m'
    },
    {
        'name': 'RLIMIT_MEMLOCK',
        'value': 'unlimited'
    },
]

deployment = StatefulSet()
deployment.metadata().name('elasticsearch-master').labels({
    'app': 'elasticsearch',
    'role': 'master'
}).annotations({
    # 'security.kubernetes.io/sysctls': 'vm.swappiness=0',
    'security.kubernetes.io/sysctls':
'vm.max_map_count=262144',
    # 'security.kubernetes.io/sysctls':
'vm.overcommit_memory=1'
})
deployment.spec().replicas(2).revisionHistoryLimit(10)
deployment.spec().serviceName('elasticsearch')
deployment.spec().selector(
    {'matchLabels': {
        'app': 'elasticsearch',
        'role': 'master'
    }})
deployment.spec().template().metadata().labels({
    'app': 'elasticsearch',
    'role': 'master'
```

```
})
deployment.spec().template().spec().initContainers(
).name('sysctl').image(image).imagePullPolicy('IfNotPresent').s
ecurityContext({
    'privileged':
    True,
    'runAsUser':
    0
}).command([
    "/bin/bash",
    "-c",
    "sysctl -w vm.max_map_count=262144 -w vm.swappiness=0 -w
vm.overcommit_memory=1",
])
deployment.spec().template().spec().containers(
).name('elasticsearch-
master').image(image).resources(None).ports([
    {
        'name': 'restful',
        'protocol': 'TCP',
        'containerPort': 9200
    },
    {
        'name': 'transport',
        'protocol': 'TCP',
        'containerPort': 9300
    },
]).volumeMounts([
    #     {
    #     'name': 'config',
    #     'mountPath':
'/usr/share/elasticsearch/config/elasticsearch.yml',
    #     'subPath': 'elasticsearch.yml'
    # },
    {
        'name': 'elasticsearch',
        'mountPath': '/usr/share/elasticsearch/data'
    }
]).env(env).securityContext({'privileged': True})
deployment.spec().template().spec().volumes([{
    'name': 'config',
    'configMap': {
        'name': 'elasticsearch'
    }
}, {
    'name': 'elasticsearch',
    'emptyDir': {}
}])
# deployment.debug()
compose.add(deployment)

livenessProbe = {
    'tcpSocket': {
        'port': 9300
    },
    'initialDelaySeconds': 60,
    'failureThreshold': 3,
    'periodSeconds': 10,
    'successThreshold': 1,
    'timeoutSeconds': 5
}
readinessProbe = {
    'tcpSocket': {
        'port': 9300
    },
    'initialDelaySeconds': 5,
    'failureThreshold': 3,
    'periodSeconds': 10,
    'successThreshold': 1,
    'timeoutSeconds': 5
}

statefulSet = StatefulSet()
statefulSet.metadata().name('elasticsearch-data').labels({
    'app': 'elasticsearch',
    'role': 'data'
}).annotations({
    # 'security.kubernetes.io/sysctls': 'vm.swappiness=0',
    'security.kubernetes.io/sysctls':
'vm.max_map_count=262144',
    # 'security.kubernetes.io/sysctls':
'vm.overcommit_memory=1'
})
statefulSet.spec().replicas(3).revisionHistoryLimit(10)
statefulSet.spec().serviceName('elasticsearch-data')
statefulSet.spec().selector(
    {'matchLabels': {
        'app': 'elasticsearch',
        'role': 'data'
    }})
```

```
statefulSet.spec().template().metadata().labels({
    'app': 'elasticsearch',
    'role': 'data'
})
statefulSet.spec().template().spec().initContainers(
).name('sysctl').image(image).imagePullPolicy('IfNotPresent').s
ecurityContext({
    'privileged':
    True,
    'runAsUser':
    0
}).command([
    "/bin/bash",
    "-c",
    "sysctl -w vm.max_map_count=262144 -w vm.swappiness=0 -w
vm.overcommit_memory=1",
])
statefulSet.spec().template().spec().containers(
).name('elasticsearch-data').image(image).ports([
    # {'name': 'restful', 'protocol': 'TCP', 'containerPort':
9200},
    {
        'name': 'transport',
        'protocol': 'TCP',
        'containerPort': 9300
    }
]).volumeMounts([
#     {
#     'name': 'config',
#     'mountPath':
'/usr/share/elasticsearch/config/elasticsearch.yml',
#     'subPath': 'elasticsearch.yml'
# },
{
    'name': 'elasticsearch',
    'mountPath': '/usr/share/elasticsearch/data'
}]).env(env).securityContext({
    'privileged': True
}).resources(None).livenessProbe(livenessProbe).readinessProbe(
readinessProbe)
statefulSet.spec().template().spec().volumes([
    {
        'name': 'config',
        'configMap': {
            'name': 'elasticsearch'
        }
    }
])
#
statefulSet.spec().volumeClaimTemplates('a').metadata().name('e
lasticsearch')
#
statefulSet.spec().volumeClaimTemplates('a').spec().resources({
'requests':{'storage':
'1Gi'}}).accessModes(['ReadWriteOnce']).storageClassName('local
-path')
statefulSet.spec().volumeClaimTemplates([{
    'metadata': {
        'name': 'elasticsearch'
    },
    'spec': {
        'accessModes': ["ReadWriteOnce"],
        #   'storageClassName': "longhorn-storage",
        'storageClassName': "local-path",
        'resources': {
            'requests': {
                'storage': '100Gi'
            }
        }
    }
}])
# statefulSet.debug()
compose.add(statefulSet)

ingress = Ingress()
ingress.apiVersion('networking.k8s.io/v1')
ingress.metadata().name('elasticsearch').labels({
    'app': 'elasticsearch',
    'role': 'master'
})
ingress.metadata().namespace(namespace)
#
ingress.metadata().annotations({'kubernetes.io/ingress.class':
'nginx'})
ingress.spec().rules([{
    'host': 'es.netkiller.cn',
    'http': {
        'paths': [{
            'pathType': Define.Ingress.pathType.Prefix,
            'path': '/',
            'backend': {
                'service': {
                    'name': 'elasticsearch',
                    'port': {
                        'number': 9200
                    }
                }
            }
        }]
    }
}])
# ingress.debug()
compose.add(ingress)
# compose.debug()

# kubeconfig = '/Volumes/Data/kubernetes/test'
# kubeconfig =  os.path.expanduser('~/workspace/opsk3d-
test.yaml')
kubeconfig =
os.path.expanduser('~/workspace/ops/ensd/k3s.yaml')

kubernetes = Kubernetes(kubeconfig)
kubernetes.compose(compose)
kubernetes.main()
```

## Kibana

```
import sys, os

sys.path.insert(0, '/Users/neo/workspace/devops')
from netkiller.kubernetes import *

namespace = 'default'

config = ConfigMap('kibana')
config.apiVersion('v1')
config.metadata().name('kibana').namespace(namespace)
# config.from_file('redis.conf', 'redis.conf')
config.data({
    'kibana.yml':
    pss('''\
server.name: kibana
server.host: "0"
server.basePath: "/kibana"
monitoring.ui.container.elasticsearch.enabled: true
xpack.security.enabled: true
elasticsearch.hosts: [ "http://elasticsearch:9200" ]
elasticsearch.username: elastic
elasticsearch.password: I3KEj0MhUmGxKyd510MhUmGxKydSt
''')
})

limits = {
    'limits': {
        'cpu': '200m',
        'memory': '2Gi'
    },
    'requests': {
        'cpu': '200m',
        'memory': '1Gi'
    }
}

livenessProbe = {
    'tcpSocket': {
        'port': 6379
    },
    'initialDelaySeconds': 30,
    'failureThreshold': 3,
    'periodSeconds': 10,
    'successThreshold': 1,
    'timeoutSeconds': 5
}
readinessProbe = {
    'tcpSocket': {
        'port': 6379
    },
    'initialDelaySeconds': 5,
    'failureThreshold': 3,
    'periodSeconds': 10,
    'successThreshold': 1,
    'timeoutSeconds': 5
}
```

```
deployment = Deployment()
deployment.metadata().name('kibana').labels({
    'app': 'kibana'
}).namespace(namespace)
deployment.spec().replicas(1)
deployment.spec().revisionHistoryLimit(10)
# deployment.spec().serviceName('redis')
deployment.spec().selector({'matchLabels': {'app': 'kibana'}})
deployment.spec().strategy().type('RollingUpdate').rollingUpdat
e('25%','25%')
deployment.spec().template().metadata().labels({'app':
'kibana'})
deployment.spec().template().spec().containers().name('kibana')
.image(
    'kibana:8.4.1').ports([{
        'name': 'http',
        'containerPort': 5601,
        'protocol': 'TCP'
    }]).env([
        {
            'name': 'TZ',
            'value': 'Asia/Shanghai'
        },
        {
            'name': 'ELASTICSEARCH_HOSTS',
            'value':
'http://elasticsearch.default.svc.cluster.local:9200'
        },
    ])
deployment.spec().template().spec().tolerations([{
    'key': 'node-role.kubernetes.io/master',
    'effect': 'NoSchedule'
}])
# .volumeMounts([
    # {
    #     'name': 'config',
    #     'mountPath': '/usr/share/kibana/config/kibana.yml',
    #     'subPath': 'kibana.yml'
    # },
# ])
#
.resources(None).livenessProbe(livenessProbe).readinessProbe(re
adinessProbe)

# deployment.spec().template().spec().volumes([{
```

```
#     'name': 'config',
#     'configMap': {
#         'name': 'kibana'
#     }
# }])

service = Service()
service.metadata().name('kibana')
service.metadata().namespace(namespace)
service.spec().selector({'app': 'kibana'})
service.spec().type('ClusterIP')
service.spec().ports([{
    'name': 'http',
    'protocol': 'TCP',
    'port': 80,
    'targetPort': 5601
}])
# service.debug()

ingress = Ingress()
ingress.apiVersion('networking.k8s.io/v1')
ingress.metadata().name('kibana').labels({
    'app': 'kibana',
})
ingress.metadata().namespace(namespace)
#
ingress.metadata().annotations({'kubernetes.io/ingress.class':
'nginx'})
ingress.spec().rules([
    {
        'host': 'kibana.netkiller.cn',
        'http': {
            'paths': [{
                'pathType': Define.Ingress.pathType.Prefix,
                'path': '/',
                'backend': {
                    'service': {
                        'name': 'kibana',
                        'port': {
                            'number': 80
                        }
                    }
                }
            }]
        }
    }
])

compose = Compose('development')
compose.add(config)
compose.add(deployment)
compose.add(service)
compose.add(ingress)
# compose.debug()

# kubeconfig = '/Volumes/Data/kubernetes/test'
kubeconfig =
os.path.expanduser('~/workspace/ops/ensd/k3s.yaml')

kubernetes = Kubernetes(kubeconfig)
kubernetes.compose(compose)
kubernetes.main()
```

验证是否工作正常

```
neo@MacBook-Pro-Neo ~> curl -s -X GET
"http://es.netkiller.cn/_cat/nodes?v=true&pretty"
ip           heap.percent ram.percent cpu load_1m load_5m
load_15m node.role   master name
10.42.2.95             24          19   0    3.79    1.89
0.84 cdfhilmrstw -      elasticsearch-data-2
10.42.1.221            19          20   0    0.03    0.13
0.21 cdfhilmrstw -      elasticsearch-data-0
10.42.0.186            20          41   0    0.01    0.14
0.19 cdfhilmrstw -      elasticsearch-data-1
10.42.2.94             21          19   0    3.79    1.89
0.84 cdfhilmrstw -      elasticsearch-master-0
10.42.1.220            34          20   0    0.03    0.13
0.21 cdfhilmrstw *      elasticsearch-master-1
```

```
neo@MacBook-Pro-Neo ~> curl -s -X GET
"http://es.netkiller.cn/_cat/health?v&pretty"
epoch      timestamp cluster            status node.total
node.data shards pri relo init unassign pending_tasks
max_task_wait_time active_shards_percent
1662963543 06:19:03  kubernetes-cluster green           5
5      8   4    0    0        0             0
-                100.0%
```

## 3.15. 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
}])
# service.debug()

# persistentVolumeClaim = PersistentVolumeClaim()
# persistentVolumeClaim.metadata().name('sonarqube')
# persistentVolumeClaim.metadata().namespace(namespace)
# persistentVolumeClaim.metadata().labels({'app': 'sonarqube',
'type': 'longhorn'})
# persistentVolumeClaim.spec().storageClassName('longhorn')
# persistentVolumeClaim.spec().accessModes(['ReadWriteOnce'])
# persistentVolumeClaim.spec().resources({'requests':
{'storage': '2Gi'}})
```

```
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().nodeName('master')

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(
#     {
```

```
#     'limits': {
#         'cpu': '500m',
#         'memory': '2Gi'
#     },
#     'requests': {
#         'cpu': '500m',
#         'memory': '2Gi'
#     }
# }
).livenessProbe(
#     {
#     'httpGet': {
#         'port': 9000,
#         'path': '/'
#     },
#     'initialDelaySeconds': 30,
#     'failureThreshold': 3,
#     'periodSeconds': 10,
#     'successThreshold': 1,
#     'timeoutSeconds': 5
# }
).readinessProbe(
#     {
#     'httpGet': {
#         'port': 9000,
#         'path': '/'
#     },
#     'initialDelaySeconds': 5,
#     'failureThreshold': 3,
#     'periodSeconds': 10,
#     'successThreshold': 1,
#     'timeoutSeconds': 5
# }
).volumeMounts([
    {
        'name': 'sonarqube',
        'mountPath': '/opt/sonarqube/data',
        'subPath' : 'data'
    },
    {
        'name': 'sonarqube',
        'mountPath': '/opt/sonarqube/extensions',
        'subPath' : 'extensions'
    },
]).securityContext({'privileged': True})
```

```
# .args(['--appendonly yes','--requirepass sonarqubepass2021'])
# .command(["sh -c sonarqube-server
/usr/local/etc/sonarqube.conf"])
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.metadata().annotations({'kubernetes.io/ingress.class':
'nginx'})
ingress.spec().rules([
{
    'host': 'sonarqube.netkiller.cn',
    'http':{
        'paths': [{
            'pathType': Define.Ingress.pathType.Prefix,
            'path': '/',
            'backend':{
                'service':{
                    'name':'sonarqube',
                    'port':{'number': 80}
                }
            }}]}
},{
    'http':{
        'paths': [{
            'pathType': Define.Ingress.pathType.Prefix,
            'path': '/sonarqube',
            'backend':{
                'service':{
                    'name':'sonarqube',
                    'port':{'number': 80}
                }
            }}]}
}

])

compose = Compose('development')

# compose.add(persistentVolumeClaim)
compose.add(service)
compose.add(statefulSet)
compose.add(ingress)
# compose.debug()

kubeconfig = '/Users/neo/workspace/kubernetes/office.yaml'
# kubeconfig = os.path.expanduser('~/workspace/ops/k3s.yaml')

kubernetes = Kubernetes(kubeconfig)
kubernetes.compose(compose)
kubernetes.main()
```

# 第 12 章 Virtual Machine(虚拟机)

## 1. Kernel-based Virtual Machine(KVM)

http://wiki.centos.org/HowTos/KVM

### 1.1. kvm install usage yum

确认处理器是否支持KVM

```
egrep 'vmx|svm' /proc/cpuinfo
```

对当前系统做一个全面升级

```
sudo yum update
sudo yum upgrade
```

Installing

如果你不想安装Virtualization组，想单独安装需要的软件，可是使用下面命令

```
# yum install qemu-kvm libvirt virt-install bridge-utils
```

确认kvm已经安装

**lsmod | grep kvm**

```
# lsmod | grep kvm
kvm_intel              138567  0
```

```
kvm                    441119  1 kvm_intel
```

Create the disk image

**qemu-img create -f qcow2 disk.img 5G**

or

**dd if=/dev/zero of=disk.img bs=1G count=5**

```
# qemu-img create -f qcow2 disk.img 5G
Formatting 'disk.img', fmt=qcow2, size=5242880 kB

# dd if=/dev/zero of=disk.img bs=1G count=5
5+0 records in
5+0 records out
5368709120 bytes (5.4 GB) copied, 61.0353 seconds, 88.0 MB/s
```

Creating a virtual machine

```
/usr/libexec/qemu-kvm -hda disk.img -cdrom archlinux-2009.08-
core-x86_64.iso -m 512 -boot d
```

如果你不在localhost上安装OS,你需要指定vnc,这样你可以远程连接到kvm

```
[root@scientific ~]# /usr/libexec/qemu-kvm disk.img -cdrom
rhel-server-5.6-x86_64-dvd.iso -m 8000 -boot d -vnc :1
```

```
[root@scientific ~]# yum install -y virt-manager virt-top virt-
v2v virt-viewer
or
[root@scientific ~]# yum groupinstall 'Virtualization'
```

## brctl / tunctl

```
[root@scientific ~]# yum install -y tunctl
```

DHCP

```
brctl addbr br0
ifconfig eth0 0.0.0.0
brctl addif br0 eth0
dhclient br0
tunctl -b -u root
ifconfig tap0 up
brctl addif br0 tap0
```

STATIC IP Address

```
brctl addbr br0
ifconfig eth0 0.0.0.0
brctl addif br0 eth0
ifconfig br0 up
tunctl -b -u root
ifconfig tap0 up
brctl addif br0 tap0

ifconfig br0 192.168.1.120 netmask 255.255.255.0 up
ip route add default via 192.168.3.1 dev br0
```

```
[root@scientific ~]# ip route
192.168.3.0/24 dev br0  proto kernel  scope link  src
192.168.3.43
192.168.3.0/24 dev tap0  proto kernel  scope link  src
192.168.3.21

default via 192.168.3.1 dev br0
[root@scientific ~]# brctl show
bridge name     bridge id               STP enabled
```

```
interfaces
br0             8000.4ea7e4cf4633       no              eth0
                                                        tap0
br06499         8000.000000000000       no
```

启动KVM

**指定网络参数 -net nic -net tap,ifname=tap0,script=no**

```
/usr/libexec/qemu-kvm -hda disk.img -m 8000 -net nic -net
tap,ifname=tap0,script=no -vnc :1

/usr/libexec/qemu-kvm -hda disk.img -m 8000 -net nic -net
tap,ifname=tap0,script=no -nographic -daemonize
```

**virt-install**

```
yum install -y libvirt python-virtinst virt-manager
```

命令行安装

```
sudo virt-install --connect qemu:///system -n Ubuntu32 -r 512 -
-vcpus=1 -f /dev/sda3 -s 9 -c Desktop/ubuntu-10.10-desktop-
i386.iso --vnc --noautoconsole --os-type linux --os-variant
generic26 --accelerate --network=bridge:virbr0 --hvm
sudo virt-install --connect qemu:///system -n Ubuntu32 -r 512 -
-vcpus=1 -f ~/ubuntu32.qcow2 -s 12 -c esktop/ubuntu-10.10-
desktop-i386.iso --vnc --noautoconsole --os-type linux --os-
variant generic26 --accelerate --network=bridge:br0 --hvm
```

进入GUI工具

```
virsh -c qemu:///system list
```

```
sudo virt-manager
```

## 1.2. Ubuntu

确认你的CPU是否支持KVM

```
egrep '(vmx|svm)' —color=always /proc/cpuinfo
```

```
sudo apt-get install kvm libvirt-bin ubuntu-vm-builder  bridge-
utils kvm-pxe uml-utilities
```

kvm gui

```
sudo apt-get install ubuntu-virt-server ubuntu-virt-mgmt
ubuntu-vm-builder python-vm-builder kvm-pxe
```

## 1.3. CentOS 6.2

```
# yum groupinstall Virtualization
# yum groupinstall "Virtualization Client"
# yum groupinstall "Virtualization Platform"

# /etc/init.d/libvirtd start
Starting libvirtd daemon:                                  [
OK  ]
```

## 1.4. Scientific Linux Virtualization

```
[root@scientific ~]# yum groupinstall 'Virtualization'
'Virtualization Client' 'Virtualization Platform'
```

```
'Virtualization Tools'
```

## 1.5. libvirt

### virsh

```
$ sudo virsh -c qemu:///system list
 Id Name                 State
----------------------------------
  1 Ubuntu               running
  2 Ubuntu-Server        running

# virsh list
 Id    Name                           State
----------------------------------------------------
 1     Ubuntu                         running
 2     CentOS6.4                      running
```

```
# virsh

显示虚拟机列表:
virsh # list --all

启动虚拟机:
virsh # start [name]

关闭虚拟机:
virsh # shutdown [name]

重启虚拟机:
virsh # reboot [name]

指定虚拟机开机自动启动:
virsh # autostart [name]
```

### 例 12.1. virsh

```
virsh # list --all
 Id    Name                           State
----------------------------------------------------
 -     CentOS6.4                      shut off
 -     FreeBSD                        shut off
 -     Test                           shut off
 -     Ubuntu                         shut off
 -     www                            shut off

virsh # start Ubuntu
Domain Ubuntu started

virsh # list --all
 Id    Name                           State
----------------------------------------------------
 1     Ubuntu                         running
 -     CentOS6.4                      shut off
 -     FreeBSD                        shut off
 -     Test                           shut off
 -     www                            shut off

virsh # quit
```

**console**

```
# virsh list
 Id    Name                           State
----------------------------------------------------
 2     monitor                        running

# virsh console monitor
Connected to domain monitor
Escape character is ^]
```

Ctrl + ] 推出 console

**dumpxml**

dump 虚拟机配置文件

```
virsh dumpxml Test
```

### Virtual Machine Manager

## 1.6. FAQ

### No hypervisor options were found for this connection

Error: No hypervisor options were found for this connection

```
[root@r910 etc]# grep kvm /var/log/messages
Jun 21 15:28:05 r910 udevd[803]: specified group 'kvm' unknown
Jun 21 15:28:05 r910 udevd[803]: specified group 'kvm' unknown
Jun 21 15:28:07 r910 kernel: kvm: disabled by bios
Jun 21 15:28:07 r910 yum: Installed: 2:qemu-kvm-0.12.1.2-2.1
13.el6_0.8.x86_64
Jun 21 15:58:27 r910 kernel: kvm: disabled by bios
Jun 21 16:48:08 r910 kernel: kvm: disabled by bios
Jun 21 17:15:42 r910 yum: Erased: qemu-kvm
Jun 21 17:20:00 r910 kernel: kvm: disabled by bios
Jun 21 17:20:00 r910 yum: Installed: 2:qemu-kvm-0.12.1.2-2.1
13.el6_0.8.x86_64
```

进入BIOS启用虚拟化

### 如何判断当前服务器是实体机还是虚拟机

```
# lspci
00:00.0 Host bridge: Intel Corporation 440BX/ZX/DX -
82443BX/ZX/DX Host bridge (rev 01)
00:01.0 PCI bridge: Intel Corporation 440BX/ZX/DX -
82443BX/ZX/DX AGP bridge (rev 01)
00:07.0 ISA bridge: Intel Corporation 82371AB/EB/MB PIIX4 ISA
```

```
(rev 08)
00:07.1 IDE interface: Intel Corporation 82371AB/EB/MB PIIX4
IDE (rev 01)
00:07.3 Bridge: Intel Corporation 82371AB/EB/MB PIIX4 ACPI (rev
08)
00:07.7 System peripheral: VMware Virtual Machine Communication
Interface (rev 10)
00:0f.0 VGA compatible controller: VMware SVGA II Adapter
00:10.0 SCSI storage controller: LSI Logic / Symbios Logic
53c1030 PCI-X Fusion-MPT Dual Ultra320 SCSI (rev 01)
00:11.0 PCI bridge: VMware PCI bridge (rev 02)
00:15.0 PCI bridge: VMware PCI Express Root Port (rev 01)
00:15.1 PCI bridge: VMware PCI Express Root Port (rev 01)
00:15.2 PCI bridge: VMware PCI Express Root Port (rev 01)
00:15.3 PCI bridge: VMware PCI Express Root Port (rev 01)
00:15.4 PCI bridge: VMware PCI Express Root Port (rev 01)
00:15.5 PCI bridge: VMware PCI Express Root Port (rev 01)
00:15.6 PCI bridge: VMware PCI Express Root Port (rev 01)
00:15.7 PCI bridge: VMware PCI Express Root Port (rev 01)
00:16.0 PCI bridge: VMware PCI Express Root Port (rev 01)
00:16.1 PCI bridge: VMware PCI Express Root Port (rev 01)
00:16.2 PCI bridge: VMware PCI Express Root Port (rev 01)
00:16.3 PCI bridge: VMware PCI Express Root Port (rev 01)
00:16.4 PCI bridge: VMware PCI Express Root Port (rev 01)
00:16.5 PCI bridge: VMware PCI Express Root Port (rev 01)
00:16.6 PCI bridge: VMware PCI Express Root Port (rev 01)
00:16.7 PCI bridge: VMware PCI Express Root Port (rev 01)
00:17.0 PCI bridge: VMware PCI Express Root Port (rev 01)
00:17.1 PCI bridge: VMware PCI Express Root Port (rev 01)
00:17.2 PCI bridge: VMware PCI Express Root Port (rev 01)
00:17.3 PCI bridge: VMware PCI Express Root Port (rev 01)
00:17.4 PCI bridge: VMware PCI Express Root Port (rev 01)
00:17.5 PCI bridge: VMware PCI Express Root Port (rev 01)
00:17.6 PCI bridge: VMware PCI Express Root Port (rev 01)
00:17.7 PCI bridge: VMware PCI Express Root Port (rev 01)
00:18.0 PCI bridge: VMware PCI Express Root Port (rev 01)
00:18.1 PCI bridge: VMware PCI Express Root Port (rev 01)
00:18.2 PCI bridge: VMware PCI Express Root Port (rev 01)
00:18.3 PCI bridge: VMware PCI Express Root Port (rev 01)
00:18.4 PCI bridge: VMware PCI Express Root Port (rev 01)
00:18.5 PCI bridge: VMware PCI Express Root Port (rev 01)
00:18.6 PCI bridge: VMware PCI Express Root Port (rev 01)
00:18.7 PCI bridge: VMware PCI Express Root Port (rev 01)
03:00.0 Ethernet controller: VMware VMXNET3 Ethernet Controller
```

```
(rev 01)
```

```
# dmesg | grep vm
kvm-clock: Using msrs 4b564d01 and 4b564d00
kvm-clock: cpu 0, msr 0:1c28841, boot clock
kvm-clock: cpu 0, msr 0:2216841, primary cpu clock
kvm-stealtime: cpu 0, msr 220e880
kvm-clock: cpu 1, msr 0:2316841, secondary cpu clock
kvm-stealtime: cpu 1, msr 230e880
sizeof(vma)=200 bytes
Switching to clocksource kvm-clock
```

# 2. Xen

## 2.1. install

```
[root@development ~]# xm list
Name                                          ID Mem(MiB) VCPUs 
State   Time(s)
Domain-0                                       0     1735     2 r--
---   1194.1
```

create a virtual harddisk

```
[root@development ~]# mkdir /srv/vm/
[root@development ~]# dd if=/dev/zero of=/srv/vm/centos.img 
bs=1M count=4096
4096+0 records in
4096+0 records out
4294967296 bytes (4.3 GB) copied, 49.2547 seconds, 87.2 MB/s
```

ubuntu

```
[root@development ~]# virt-install -n centos -r 256 -f 
/srv/vm/centos.img --nographics -l ftp://192.168.3.9/pub/
```

## 2.2. Manager

list

```
[root@development ~]# xm list
Name                                          ID Mem(MiB) VCPUs 
State   Time(s)
Domain-0                                       0     1726     2 r--
---   5686.6
```

```
centos                                    6      127     1 -b-
---     74.3
```

start

```
[root@development ~]# virsh  start centos
Domain centos started
```

reboot

```
[root@development ~]# xm reboot centos
```

shutdown

```
[root@development ~]# xm shutdown centos
```

console

```
[root@development ~]# xm console centos
```

config

```
[root@development ~]# cat /etc/xen/centos
name = "centos"
uuid = "a6a3f200-bcbb-cdbd-c06e-9e71f739310f"
maxmem = 128
memory = 128
vcpus = 1
bootloader = "/usr/bin/pygrub"
on_poweroff = "destroy"
on_reboot = "restart"
on_crash = "restart"
```

```
disk = [ "tap:aio:/srv/vm/centos.img,xvda,w" ]
vif = [ "mac=00:16:36:5d:41:d0,bridge=xenbr0,script=vif-bridge"
]
```

Automatically starting domains

```
[root@development ~]# mv /etc/xen/centos /etc/xen/auto
```

## 3. OpenVZ

### 3.1. 安装OpenVZ

过程 12.1. OpenVZ 安装步骤

1. 获得OpenVZ yum安装源

```
# cd /etc/yum.repos.d
# wget http://download.openvz.org/openvz.repo
# rpm --import  http://download.openvz.org/RPM-GPG-Key-
OpenVZ
```

2. 安装OpenVZ核心以及头文件

```
# yum install ovzkernel[-flavor]
```

3. 修改启动所使用的内核为OpenVZ内核，使OpenVZ内核为默认启动内核

```
# vim /etc/grub.conf
```

将类似下面的内容

```
title Fedora Core (2.6.8-022stab029.1)
       root (hd0,0)
       kernel /vmlinuz-2.6.8-022stab029.1 ro root=/dev/sda5
quiet rhgb vga=0x31B
       initrd /initrd-2.6.8-022stab029.1.img
```

修改为类似这样

```
title OpenVZ (2.6.8-022stab029.1)
        root (hd0,0)
        kernel /vmlinuz-2.6.8-022stab029.1 ro
root=/dev/sda5
        initrd /initrd-2.6.8-022stab029.1.img
```

或直接在里面寻找类似开头为

```
title CentOS (2.6.18-194.3.1.el5.028stab069.6)
```

的项目，并且把default改为他的下标，下标从0开始

4. 修改LINUX网络配置文件

```
/etc/sysctl.conf
# On Hardware Node we generally need
# packet forwarding enabled and proxy arp disabled
net.ipv4.ip_forward = 1 #修改

net.ipv6.conf.default.forwarding = 1 #添加
net.ipv6.conf.all.forwarding = 1 #添加
net.ipv4.conf.default.proxy_arp = 0 #添加

# Enables source route verification
net.ipv4.conf.all.rp_filter = 1 #修改

# Enables the magic-sysrq key
kernel.sysrq = 1 #修改

# We do not want all our interfaces to send redirects
net.ipv4.conf.default.send_redirects = 1 #添加
net.ipv4.conf.all.send_redirects = 0 #添加
```

5. 关闭SELinux

```
# lokkit --selinux=disabled
```

```
SELINUX=disabled
```

6. 重启LINUX

```
# reboot
```

7. 安装OpenVZ管理工具

```
# yum install vzctl
# yum install vzquota
# yum install vzyum
```

   用到什么工具就安装什么工具，具体可以使用# yum search vz*搜索一下

8. 启动OpenVZ服务

```
# /sbin/service vz start
```

## 3.2. 使用OpenVZ & 建立VPS

由于VZ是半虚拟化的，所以VZ和VM不同的是VZ需要系统模板，而不是VM那样只需要一个ISO文件就可以安装

### 安装操作系统模板

1. 搜索系统模板

```
# yum search vztmpl
```

2. 在搜索出来的结果中选用你想安装的操作系统

```
# yum install vztmpl-centos-4 -y
```

3. 为操作系统模板建立缓存

   在我装的最小化CENTOS中，此步要下载很多包，需要很长时间完成

```
# vzpkgcache
```

该命令将建立centos-4-i386-minimal.tar.gz和centos-4-i386-default.tar.gz文件 或

```
# vzpkgcache centos-4-i386-minimal
```

建立 centos-4-i386-minimal.tar.gz

```
# vzpkgcache centos-4-i386-default
```

建立 centos-4-i386-default.tar.gz

出现Cache file centos-4-i386-default.tar.gz [120M] created.表示创建成功

**注意：本次步骤可能会出现如下错误**

```
cp: cannot stat `/etc/sysconfig/vz-scripts//ve-
```

```
vps.basic.conf-sample': No such file or directory
ERROR: Can't copy VPS config
```

解决方法：进入/etc/sysconfig/vz-scripts/目录，将ve.basic.conf-sample 拷贝一份重命名为ve-vps.basic.conf-sample

查看系统中已经存在的操作系统缓存

```
# vzpkgls
```

### 创建OpenVZ操作系统节点（VPS）

1. 准备配置文件

   平分主机系统资源（当然，如果你对配置文件的修改很熟悉也可以自己定制）

   ```
   cd /etc/sysconfig/vz-scripts/
   vzsplit -n 3 -f vps.zenw.org
   ```

   这样，系统资源就被平均分成了3分，并且产生了一个配置文件示例

2. 验证配置文件有效性

   ```
   vzcfgvalidate ve-vps.zenw.org.conf-sample
   ```

3. 创建VPS节点

   ```
   vzctl create 100 --ostemplate centos-4-i386-minimal --
   config vps.zenw.org
   ```

其中100是该节点的编号，可以自己定义

4. 配置该VPS

```
设置VPS的hostname
vzctl set 100 --hostname zenw.org --save
设置VPS的ip
vzctl set 100 --ipadd 192.168.xxx.xxx --save
设置VPS的管理员帐号和密码
vzctl set 100 --userpasswd root:xxxxxxxxx
设置VPS的DNS服务器
vzctl set 100 --nameserver 8.8.8.8 --save
设置VPS自启动
vzctl set 100 --onboot yes --save
启动VPS节点
vzctl start 100
执行VPS内部的命令（这里是开启VPS的ssh服务）
vzctl exec 100 service sshd start
加入VPS节点
vzctl enter 100
停止VPS节点
vzctl stop 100
```

## 3.3. 设置VPS参数

1. 修改VPS节点的配置文件

```
vim /etc/sysconfig/vz-scripts/100.conf
在文件中添加或修改 DISK_QUOTA=no

重启VPS节点
vzctl restart 100
查看当前磁盘大小
vzctl exec 100 df
设置磁盘大小
vzctl set 100 --diskinodes 75000000:79000000 --save
vzctl set 100 --quotatime 600 --save
查看修改后的磁盘大小
vzctl exec 100 df
vzctl exec 100 stat -f /
vzctl set 100 --quotaugidlimit 100 --save
vzctl restart 100

vzctl exec 100 rpm -q quota

vzyum 100 install quota

vzquota stat 100 -t
```

2. 为VPS节点安装yum工具或其他工具

```
vzyum 100 install <软件名称>
vzyum 100 install yum
```

另外,如果vzctl enter进入节点时出现错误，或无法ssh节点，需要运行以下命令： vzctl exec 112 "cd /dev; /sbin/MAKEDEV pty; /sbin/MAKEDEV tty; /sbin/MAKEDEV generic"

# 4. vagrant - Tool for building and distributing virtualized development environments

https://www.vagrantup.com/downloads.html

## 4.1. vagrant for windows

下一步

下一步

下一步

安装

下一步

完成

重启

# 5. 虚拟机管理

## 5.1. Proxmox - Open-source virtualization management platform Proxmox VE

## 5.2. OpenStack

## 5.3. CloudStack

## 5.4. OpenNode

## 5.5. OpenNEbula