老方的博客

发布 *1.0*

2019 年 10 月 08 日

BookMark Online

CHAPTER 1

编程教室

1.1 Python 电子教程

- 简明 Python 教程
- 廖雪峰的 Python 教程
- RUNNOOB 的 Python 教程
- w3school&runoob 教程合集-1：囊括世面各种语言
- w3school&runoob 教程合集-2：囊括世面各种语言
- 现代魔法学院：Python 教室
- Python 中文学习大本营

1.2 Linux 入门

- 本人 Linux 学习笔记：持续更新
- LINUX：命令行的艺术 (Github)
- 基础学习 | 鸟哥的私房菜
- 基础学习 | 每天一个 Linux 命令
- 基础学习 | 《Linux 云计算从入门到精通》系列实战笔记

1.3 云计算入门

- CloudMan：致力于云计算学习和实践
- 每天 5 分钟玩转 OpenStack
- 每天 5 分钟玩转 Docker 容器技术

1.4 数据库入门

- 数据库 | MySQL50 题练习建库脚本
- 数据库 | MySQL 50 题练习答案

1.5 正则表达式入门

- 正则表达式 | 语法 - 基础教材
- 正则表达式 | Everything 的使用
- Git | 廖雪峰的 Git 教程
- Git | Git-Book 教程

1.6 集群管理

- 虚拟化管理之 virsh 命令 - 官方文档
- crmsh 命令官方文档 - 英文
- corosync pacemaker crmsh crm 实例详解
- HAProxy 使用手册（官方）- 英文
- HAProxy 使用手册（翻译）- 中文

1.7 Python 参考文档

- PEP8 代码规范
- Python 语言参考 - 中文
- Python 语言参考 - 英文
- Python 标准库文档 - 中文

- Python 标准库文档 - 英文
- Python 资源大全中文版：Awesome-Python
- Python / C API 参考手册
- Python Cookbook 3rd Edition Documentation
- Twisted 与异步编程入门 - 中文文档

CHAPTER 2

在线工具

2.1 编程相关

- 在线 Markdown | Md2All
- 在线 Markdown | Cmd Markdown
- 在线编程 - 1：包含各种主流语言
- 在线编程 - 2：包含各种主流语言
- PDF | Smallpdf.com – 您所有 PDF 问题的免费解决方案
- Pycharm |破解下载及激活方法
- Pycharm |激活 Licence Server
- 代码美化 | Carbon
- 变量命名 | CODELF
- 代码美化 | JSON
- Base64 编码
- Linux |命令大全
- 数据库 | 库表可视化设计
- IP 地址 | ipv4 计算器
- IP 地址 | ipv6 计算器

- 文件中转 | 7 天无限下载
- 项目推荐 | HelloGithub
- 工具集合 - MIKU
- 工具集合 - 小森林导航
- Rufus - 轻松创建 U 盘启动盘
- Screen To Gif - 轻松制作 gif 动画
- USER.ME - 在线 PS/CAD/PPT/EXCEL/AI/XMind/Visio
- 五笔拆字图解：解决你的五笔问题
- SM.MS - Simple Free Image Hosting
- 集思会：一个电子书免费推送网站
- 书伴：教你如何玩转 Kindle 的网站

2.2 精品软件

- 精品绿色便携软件
- 我最喜欢的软件 Windows 版 - 小众软件
- MSDN, 我告诉你：微软产品下载中心
- 大眼仔旭 - 爱软件爱汉化爱分享
- Windows Apps That Amaze Us：Windows 绝赞应用
- 精品 Mac 软件分享

2.3 信息查询

- 中国人民银行征信中心
- 国家企业信用信息公示系统
- TinEye Reverse Image Search
- 站长工具 - 站长之家
- 去查网
- GPSspg 查询网
- 工业和信息化部 ICP/IP 地址/域名信息备案管理系统
- 程序员的工具箱

- 全球国家 IP 地址段
- 网站测速：全国地区
- TinEye - 最专业的以图搜图网站

2.4 在线资源

- Linux ｜离线包大全
- Ubuntu ｜中文维基
- CentOS ｜系统镜像（各版本）
- Linux ｜网易镜像源大全
- Centos ｜清华大学开源软件镜像站
- Ubuntu ｜清华大学开源软件镜像站
- Python ｜各版本 Python 下载源

2.5 设计素材

- IconFinder
- easyIcon
- Iconfont-阿里巴巴矢量图标库
- Icons for everything - Noun Project
- Font Awesome，一套绝佳的图标字体库和 CSS 框架
- iSlide：制作高大上的 PPT

CHAPTER 3

centos command not found

3.1 command not found 解决办法

当不知道某个命令是哪个包装时，可以在已经有这个命令的主机上用下面的命令确定是哪个安装包安装的

yum whatprovides 命令路径或者命令的绝对路径

```
[root@db14 ~]# yum whatprovides /usr/sbin/ss
已加载插件: fastestmirror
Loading mirror speeds from cached hostfile
 * base: mirrors.aliyun.com
 * elrepo: mirrors.tuna.tsinghua.edu.cn
 * extras: mirrors.aliyun.com
 * updates: mirrors.aliyun.com
iproute-4.11.0-14.el7.x86_64 : Advanced IP routing and network device configuration tools
源     : base
匹配来源:
文件名     : /usr/sbin/ss



iproute-4.11.0-14.el7_6.2.x86_64 : Advanced IP routing and network device configuration␣
↪tools
源     : updates
```

(下页继续)

(续上页)

```
匹配来源:
文件名     : /usr/sbin/ss
```

3.2 常见 command not found

```
ss:bash:command not found

yum install iproute -y

ifconfig:bash:command not found

yum install net-tools -y

vim:bash:command not found

yum install vim -y

sar:bash:command not found

yum install sysstat

brctl: command not found

yum install bridge-utils -y

arp: command not found
net-tools-2.0-0.24.20131004git.el7.x86_64 : Basic networking tools
```

CHAPTER 4

centos7 fdisk 磁盘分区格式化挂载

以 /dev/sdb 分一个区为例

```
fdisk -l  # 查看没有分区的硬盘

fdisk /dev/sdb
n，建立分区
p，建立主分区
回车(使用默认分区号)
回车(使用默认起始扇区)
回车(使用默认结束扇区，如需指定分区大小可输入如 +20G)，
w，保存

partprobe # 重读分区表

mkfs -t xfs /dev/sdb1  # 格式化

# 临时挂载
mkdir /mnt/data
mount  /dev/sdb1 /mnt/data

# 修改/etc/fstab，使分区自动挂载
vim /etc/fstab 添加
/dev/sdb1 /mnt/data     xfs     defaults    0 0
```

(下页继续)

(续上页)

```
# 也可以使用 lsblk -f 或者 blkid /dev/sdb1 查看磁盘的 uuid 来挂载
UUID=f502fa32-96bf-48e8-bdf0-166c1e74f8fa    /mnt/data      xfs      defaults     0 0

mount -a
```

CHAPTER 5

git 学习

5.1 git 安装

下载安装

```
#windows
https://git-scm.com/download/win
#mac
https://git-scm.com/download/mac
#linux
apt install git 或 yum -y install git
```

编译安装

https://github.com/git/git/releases

```
yum install curl-devel expat-devel  openssl-devel zlib-devel gcc perl-ExtUtils-MakeMaker
yum  remove git
wget https://github.com/git/git/archive/v2.23.0.tar.gz
tar xf v2.23.0.tar.gz
cd git-2.23.0/
make prefix=/usr/local/git all
make prefix=/usr/local/git install
echo "export PATH=$PATH:/usr/local/git/bin" >> /etc/bashrc
```

(下页继续)

(续上页)

```
source /etc/bashrc
git --version
```

5.1.1 安装 zsh 和 oh-my-zsh

安装 zsh 的目的是为了安装 oh-my-zsh，因为 oh-my-zsh 是基于 zsh 的一个主题。

安装 oh-my-zsh 的目的是为了更快速的学习 Git 的命令行操作，它能够给我们在输入一些 Git 命令时候提供很大的方便。

```
#mac
brew install zsh

#linux
apt install zsh 或 yum -y install zsh

# 安装 oh-my-zsh
sh -c "$(curl -fsSL https://raw.github.com/robbyrussell/oh-my-zsh/master/tools/install.
↪sh)"
```

5.2 常用命令

要将代码推送到服务器通常会经历五个步骤：更新、检查，提交暂存，正式提交，推送。即 pull -> status -> add -> commit -> push

```
# 初始化一个本地版本仓库
git init

# 查看配置信息
git config user.name
git config user.email

# 设置配置
git config --global user.name Denis
git config --global user.email 87702755@qq.com

# 命令行修改配置
git config --global --replace-all user.name Denis
git config --global --replace-all user.email 87702755@qq.com
```

(下页继续)

(续上页)

```
# 修改配置文件
vim ~/.gitconfig

# 拉取代码
git clone  https://xxxx          #https 方式，需要输入用户名和密码
git clone  git@gitee.com:xxx     #ssh 方式，需要把本机公钥添加到服务器

# 同步远程代码
git pull

# 检查改动文件
git status

# 撤销更改
git checkout 文件名

# 添加文件到缓存
git add 文件名 或者  git add .

# 提交代码
git commit . -m  " 备注内容"  或者 git commit 文件名 -m  " 备注内容"

# 推送代码
git push
```

5.3 分支学习

```
# 主分支：master，默认分支

# 新建分支
git branch 分支名

# 查看分支
git branch

# 切换分支（实际项目中，每个人都要在自己的分支上工作，最后再合并到如果要在 master 上面合并分支，需要先切回到 master
```

(下页继续)

(续上页)

```
git checkout 分支名

# 合并分支
git merge 分支名字

# 删除分支（如果分支没有合并不能删除）
git branch -d 分支名

# 强制删除（如果分支没有合并要删除可以使用）
git branch  -D 分支名字
```

5.4 开发步骤

一个 master，一个 dev

1. 新建一个 dev
2. 切换到 dev 进行开发
3. 在 dev 添加文件并且提交文件
4. 切换到 master 分支
5. 将 dev 分支合并到 master 分支 git merge dev
6. 推送 master 到服务端
7. 继续切换到 dev 进行开发

5.5 冲突解决

a 和 b 同时修改同一个文件的同一行代码就会产生冲突，如果 a 先 push，那么 b 在 push 的时候就会报错。所以，为了保险起见，只要想向服务端 push 内容，首先需要 pull 内容，pull 下来之后就会将服务端的代码和本地的代码进行合并，如果有冲突，就会显示冲突 (git diff)，如果没有冲突，那就合并成功，然后再 push 上去即可，如果有冲突，商量解决冲突即可。

```
git pull    # 下拉文件
git diff    # 查看冲突
```

5.6 参考文章

https://blog.csdn.net/ZZQHELLO2018/article/details/82354900

CHAPTER 6

kvm 常用操作

6.1 创建虚拟机

```
cp swarm03.xml vm-denis-cmdb1.xml

#name 修改为 ssd/vm-denis-cmdb1
#cpu 内核调整，内存格式改为 GIB(2 处)，容量调整，ssdname 修改为 ssd/vm-denis-cmdb1
# 如不需指定 mac 地址，则需去掉<mac address='52:54:00:4F:BB:CF'/>，如需指定则设置成网络中唯一的地址

vi vm-denis-cmdb1.xml

rbd list ssd
rbd snap list ssd/CentOS7Templet
rbd clone ssd/CentOS7Templet@base ssd/vm-denis-cmdb1  # 复制快照
rbd resize  ssd/vm-denis-cmdb1 --size 30G   或者  virsh blockresize --domian vm-denis-cmbd1 --path vda --size 30G # 调整硬盘大小
virsh define  vm-denis-cmdb1.xml # 从 XML 文件定义（但不启动）一个虚拟机
virsh start vm-denis-cmdb1

virsh list
virsh dumpxml <id>  # 查看用于 vnc view 连接的端口
```

用 vnc viewer 连接 192.168.252.82:< 端口 > 后查看 ip, 即可用 xshell 连接

6.2 登录虚拟机后操作

更改默认密码

关闭防火墙

增加公钥

关闭 sshd 里 dns

6.3 快照操作

```
virsh -list -all

 -     vm-denis-k8s1                  shut off

rbd snap create ssd/vm-denis-k8s1@snapshot1   # 创建快照（首先关闭虚拟机）
rbd snap ls ssd/vm-denis-k8s1
virsh start vm-denis-k8s1  # 启动虚拟机



rbd snap rollback ssd/vm-denis-k8s1@snapshot1  # 恢复快照（关闭虚拟机后操作）
rbd snap rm ssd/vm-denis-k8s1@snapshot1  # 删除快照
rbd snap purge ssd/vm-denis-k8s1 # 删除多个快照
```

6.4 其它操作

```
virsh destroy <id> # 停止虚拟机
virsh undefine vm-denis-cmdb1 # 取消虚拟机

virsh create vm-denis-cmdb1.xml # 一次性启动

virsh blockresize 分情况
- 如果虚拟机是开机状态下，不用调 rbd resize，直接调 virsh blockresize 就行，它会帮忙自动执行 rbd resize 或 qemu-img resize
- 如果虚拟机是关机状态，则不用调该命令，直接调底层的 resize，比如 rbd 就调 rbd resize，qcow2 就调 qemu-img resize
```

(下页继续)

(续上页)

```
rbd rm  ssd/vm-denis-cmdb1  # 删除镜像
```

6.5 调整虚拟机硬盘大小

```
# 虚拟机查看原大小
[root@test ~]# fdisk /dev/vda -l

Disk /dev/vda: 21.5 GB, 21474836480 bytes, 41943040 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0x000bc509

   Device Boot      Start         End      Blocks   Id  System
/dev/vda1   *        2048    41943039    20970496   83  Linux


# 在 82 上调整成 30G
[root@rxserver3 ~]# rbd resize  ssd/vm-denis-cmdb1 --size 30G
Resizing image: 100% complete...done.

[root@rxserver3 ~]# rbd --image ssd/vm-denis-cmdb1 info
rbd image 'vm-denis-cmdb1':
        size 30 GiB in 7680 objects
        order 22 (4 MiB objects)
        id: 7650c6b8b4567
        block_name_prefix: rbd_data.7650c6b8b4567
        format: 2
        features: layering, exclusive-lock, object-map, fast-diff, deep-flatten
        op_features:
        flags:
        create_timestamp: Wed Sep  4 17:04:55 2019
        parent: ssd/CentOS7Templet@base
        overlap: 20 GiB
```

(下页继续)

(续上页)

```
# 虚拟机 poweroff 后重启，再次查看大小

[root@localhost ~]# fdisk -l

Disk /dev/vda: 32.2 GB, 32212254720 bytes, 62914560 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0x000bc509

   Device Boot      Start         End      Blocks   Id  System
/dev/vda1   *        2048    41943039    20970496   83  Linux

# 重新分区，这是在只有一个分区的情况下

[root@localhost ~]# fdisk /dev/vda
Welcome to fdisk (util-linux 2.23.2).

Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.


Command (m for help): d
Selected partition 1
Partition 1 is deleted

Command (m for help): n
Partition type:
   p   primary (0 primary, 0 extended, 4 free)
   e   extended
Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-62914559, default 2048):
Using default value 2048
Last sector, +sectors or +size{K,M,G} (2048-62914559, default 62914559):
Using default value 62914559
Partition 1 of type Linux and of size 30 GiB is set

Command (m for help): p
```

(下页继续)

(续上页)

```
Disk /dev/vda: 32.2 GB, 32212254720 bytes, 62914560 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0x000bc509

   Device Boot      Start         End      Blocks   Id  System
/dev/vda1            2048    62914559    31456256   83  Linux

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.

WARNING: Re-reading the partition table failed with error 16: Device or resource busy.
The kernel still uses the old table. The new table will be used at
the next reboot or after you run partprobe(8) or kpartx(8)
Syncing disks.


# 重启虚拟机后操作
[root@localhost ~]# reboot

[root@localhost ~]# resize2fs /dev/vda1
resize2fs 1.42.9 (28-Dec-2013)
Filesystem at /dev/vda1 is mounted on /; on-line resizing required
old_desc_blocks = 3, new_desc_blocks = 4
The filesystem on /dev/vda1 is now 7864064 blocks long.

[root@localhost ~]# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/vda1        30G  1.1G   27G   4% /
devtmpfs        1.9G     0  1.9G   0% /dev
tmpfs           1.9G     0  1.9G   0% /dev/shm
tmpfs           1.9G  8.3M  1.9G   1% /run
tmpfs           1.9G     0  1.9G   0% /sys/fs/cgroup
tmpfs           380M     0  380M   0% /run/user/0
```

6.6 参考文章

https://blog.51cto.com/wutou/1782931

https://blog.51cto.com/speakingbaicai/1161964

https://www.cnblogs.com/chenjiahe/p/5919426.html

CHAPTER 7

mysql 学习

7.1 sql 相关网站

sqlzoo

myslq 开发规范

leetcode 数据库练习

CHAPTER 8

mysql 表连接

8.1 创建表并插入测试数据

```
CREATE TABLE Person (
    id int primary key auto_increment,
    name varchar(32) unique not null default '',
    sex bool not null default 1,
    age int not null DEFAULT 0
) ENGINE=InnoDB DEFAULT CHARSET=utf8


CREATE TABLE Address  (
    id int primary key auto_increment,
    p_id int not null,
    province varchar(32) not null default '',
    city varchar(32)  not null default ''
) ENGINE=InnoDB DEFAULT CHARSET=utf8

insert into Person(name,age,sex) value
(" 张三",1,28),
(" 李四",1,21),
(" 王五",1,33),
```

(下页继续)

(续上页)

```
(" 赵六",0,11),
(" 元七",1,64),
(" 冯八",1,56)


insert into Adress(p_id,province,city) value
(1,' 北京',' 北京'),
(2,' 上海',' 上海'),
(3,' 江苏',' 南京'),
(4,' 安徽',' 蚌埠'),
(5,' 江苏',' 徐州'),
(5,' 河北',' 雄安'),
(8,' 广州',' 惠州')
```

8.2 左连接右连接内连接

内连接: 组合两个表中的记录，返回关联字段相符的记录，也就是返回两个表的交集（阴影）部分。

左连接: left join 是 left outer join 的简写，它的全称是左外连接，是外连接中的一种。左表 (a_table) 的记录将会全部表示出来，而右表 (b_table) 只会显示符合搜索条件的记录。右表记录不足的地方均为 NULL。

右连接: right join 是 right outer join 的简写，它的全称是右外连接，是外连接中的一种。左表 (a_table) 只会显示符合搜索条件的记录，而右表 (b_table) 的记录将会全部表示出来。左表记录不足的地方均为 NULL。

```
# 内连接
SELECT name,sex,age, Address.province, Address.city FROM Person INNER JOIN Address on␣
↪Person.id = Address.p_id

# 左连接
SELECT name,sex,age, Address.province, Address.city FROM Person LEFT JOIN Address on␣
↪Person.id = Address.p_id

# 右连接
SELECT name,sex,age, Address.province, Address.city FROM Person RIGHT JOIN Address on␣
↪Person.id = Address.p_id

# 使用 AS 示例 (效果同左连接)
SELECT name,sex,age, A.province, A.city FROM Person as P INNER JOIN Address as A on P.id␣
↪= A.p_id
```

显示效果

内连接

```
108  SELECT name,sex,age, Address.province, Address.city FROM Person INNER JOIN Address on Person.id = Address.p_id
109
110
```

信息 结果 1 剖析 状态

name	sex	age	province	city
张三	28	1	北京	北京
李四	21	1	上海	上海
王五	33	1	江苏	南京
赵六	11	0	安徽	蚌埠
冯八	56	1	江苏	徐州
冯八	56	1	河北	雄安

左连接

```
110
111  SELECT name,sex,age, Address.province, Address.city FROM Person LEFT JOIN Address on Person.id = Address.p_id
112
113
114
```

信息 结果 1 剖析 状态

name	sex	age	province	city
张三	28	1	北京	北京
李四	21	1	上海	上海
王五	33	1	江苏	南京
赵六	11	0	安徽	蚌埠
冯八	56	1	江苏	徐州
冯八	56	1	河北	雄安
元七	64	1	(Null)	(Null)

右连接

```
113  SELECT name,sex,age, Address.province, Address.city FROM Person RIGHT JOIN Address on Person.id = Address.p_id
114
115
116
```

信息 结果 1 剖析 状态

name	sex	age	province	city
张三	28	1	北京	北京
李四	21	1	上海	上海
王五	33	1	江苏	南京
赵六	11	0	安徽	蚌埠
冯八	56	1	江苏	徐州
冯八	56	1	河北	雄安
(Null)	(Null)	(Null)	广州	惠州

CHAPTER 9

shell 常用脚本

9.1 centos7 新装设置 IP

ifcfg-eno1 根据自己的名称变化，$1 第一个参数为 ip 地址最后一位

使用方法: sh xx.sh 12

```
#! /bin/bash

# I 不区分大小写
sed -i 's/DHCP/none/Ig' /etc/sysconfig/network-scripts/ifcfg-eno1
sed -i 's/ONBOOT=no/ONBOOT=yes/Ig' /etc/sysconfig/network-scripts/ifcfg-eno1

cat>> /etc/sysconfig/network-scripts/ifcfg-eno1 <<EOF
IPADDR="10.255.201.$1"
PREFIX="24"
GATEWAY="10.255.201.1"
DNS1="10.255.201.1"
EOF
```

CHAPTER 10

网站性能压力测试之 ab 命令

ab 是 Apache 自带的压力测试工具。ab 非常实用，它不仅可以对 Apache 服务器进行网站访问压力测试，也可以对其它类型的服务器进行压力测试。比如 Nginx、Tomcat、IIS 等。

10.1 一 ab 原理

ab 是 apachebench 命令的缩写。

ab 的原理：ab 命令会创建多个并发访问线程，模拟多个访问者同时对某一 URL 地址进行访问。它的测试目标是基于 URL 的，因此，它既可以用来测试 apache 的负载压力，也可以测试 nginx、lighthttp、tomcat、IIS 等其它 Web 服务器的压力。

ab 命令对发出负载的计算机要求很低，它既不会占用很高 CPU，也不会占用很多内存。但却会给目标服务器造成巨大的负载，其原理类似 CC 攻击。自己测试使用也需要注意，否则一次上太多的负载。可能造成目标服务器资源耗完，严重时甚至导致死机。

10.2 二 ab 安装

```
yum install httpd-tools
```

命令执行完成后，就可以直接运行 ab。

10.3 三 ab 参数说明

-n：在测试会话中所执行的请求个数。默认时，仅执行一个请求。

-c：一次产生的请求个数。默认是一次一个。

-t：测试所进行的最大秒数。其内部隐含值是-n 50000，它可以使对服务器的测试限制在一个固定的总时间以内。默认时，没有时间限制。

-p：包含了需要 POST 的数据的文件。

-P：对一个中转代理提供 BASIC 认证信任。用户名和密码由一个: 隔开，并以 base64 编码形式发送。无论服务器是否需要 (即是否发送了 401 认证需求代码)，此字符串都会被发送。

-T：POST 数据所使用的 Content-type 头信息。

-v：设置显示信息的详细程度-4 或更大值会显示头信息，3 或更大值可以显示响应代码 (404,200 等),2 或更大值可以显示警告和其他信息。

-V：显示版本号并退出。

-w：以 HTML 表的格式输出结果。默认时，它是白色背景的两列宽度的一张表。

-i：执行 HEAD 请求，而不是 GET。

-x：设置属性的字符串。

-X：对请求使用代理服务器。

-y：设置属性的字符串。

-z：设置属性的字符串。

-C：对请求附加一个 Cookie: 行。其典型形式是 name=value 的一个参数对，此参数可以重复。

-H：对请求附加额外的头信息。此参数的典型形式是一个有效的头信息行，其中包含了以冒号分隔的字段和值的对 (如,”Accept-Encoding:zip/zop;8bit”)。

-A：对服务器提供 BASIC 认证信任。用户名和密码由一个: 隔开，并以 base64 编码形式发送。无论服务器是否需要 (即, 是否发送了 401 认证需求代码)，此字符串都会被发送。

-h：显示使用方法。

-d：不显示”percentage served within XX [ms] table”的消息 (为以前的版本提供支持)。

-e：产生一个以逗号分隔的 (CSV) 文件，其中包含了处理每个相应百分比的请求所需要 (从 1% 到 100%) 的相应百分比的 (以微妙为单位) 时间。由于这种格式已经“二进制化”，所以比’gnuplot’格式更有用。

-g：把所有测试结果写入一个’gnuplot’或者 TSV(以 Tab 分隔的) 文件。此文件可以方便地导入到 Gnuplot,IDL,Mathematica,Igor 甚至 Excel 中。其中的第一行为标题。

-i：执行 HEAD 请求，而不是 GET。

-k：启用 HTTP KeepAlive 功能，即在一个 HTTP 会话中执行多个请求。默认时，不启用 KeepAlive 功能。

-q：如果处理的请求数大于 150，ab 每处理大约 10% 或者 100 个请求时，会在 stderr 输出一个进度计数。此-q 标记可以抑制这些信息。

10.4 四 ab 性能指标

在进行性能测试过程中有几个指标比较重要：

吞吐率（Requests per second）概念：服务器并发处理能力的量化描述，单位是 reqs/s，指的是某个并发用户数下单位时间内处理的请求数。某个并发用户数下单位时间内能处理的最大请求数，称之为最大吞吐率。计算公式：总请求数 / 处理完成这些请求数所花费的时间，即 Request per second = Complete requests / Time taken for tests

并发连接数（The number of concurrent connections）概念：某个时刻服务器所接受的请求数目，简单的讲，就是一个会话。

并发用户数（The number of concurrent users，Concurrency Level）概念：要注意区分这个概念和并发连接数之间的区别，一个用户可能同时会产生多个会话，也即连接数。

用户平均请求等待时间（Time per request）计算公式：处理完成所有请求数所花费的时间/总请求数，即 Time per request=Time taken for tests/（Complete requests/Concurrency Level）

服务器平均请求等待时间（Time per request: across all concurrent requests）计算公式：处理完成所有请求数所花费的时间 / 总请求数，即 Time taken for / testsComplete requests 可以看到，它是吞吐率的倒数。同时，它也 = 用户平均请求等待时间/并发用户数，即 Time per request / Concurrency Level

10.5 五 ab 实际使用

ab 的命令参数比较多，我们经常使用的是-c 和-n 参数。

```
-n 100 表示请求总数为 100
-c 10 表示并发用户数为 10

下面表示处理 100 个请求并每次同时运行 10 次请求。

ab -n 100 -c 10 http://www.163.com/index.html

This is ApacheBench, Version 2.3 <$Revision: 1430300 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking www.163.com (be patient).....done
```

（下页继续）

(续上页)

```
Server Software:        nginx  # 被测试的 Web 服务器软件名称
Server Hostname:        www.163.com    # 请求的 URL 主机名
Server Port:            80  # 监听端口

Document Path:          /index.html  # URL 中的根绝对路径
Document Length:        697909 bytes # HTTP 响应数据的正文长度

Concurrency Level:      10  # 并发用户数，我们设置的参数之一
Time taken for tests:   3.717 seconds # 压力测试消耗的总时间
Complete requests:      100  # 总请求数量，即并发数，我们设置的参数之一
Failed requests:        0   # 表示失败的请求数量，指请求在连接服务器、发送数据等环节发生异常，以及无响应后超时的情况。含有 2XX 以外的状态码，则会在测试结果中显示另一个名为 "Non-2xx responses" 的统计项，用于统计这部分请求数，这些请求并不算在失败的请求中。
Write errors:           0   # 写入错误数
Total transferred:      69833401 bytes # 表示所有请求的响应数据长度总和，包括每个 HTTP 响应数据的头信息和正文数据的长度。不包括 HTTP 请求数据的长度，仅仅为 web 服务器流向用户 PC 的应用层数据总长度。
HTML transferred:       69790900 bytes # 表示所有请求的响应数据中正文数据的总和，也就是减去了 Total transferred 中 HTTP 响应数据中的头信息的长度。
Requests per second:    26.90 [#/sec] (mean)   # 吞吐率 29.91[#/sec](mean) ，也叫 QPS 或者每秒请求数，主要指标一
Time per request:       371.720 [ms] (mean)  # 用户平均请求等待时间，3710/100/10，  主要指标二
Time per request:       37.172 [ms] (mean, across all concurrent requests)  # 单个用户请求一次的平均时间，3717/100，  主要指标三
Transfer rate:          18346.24 [Kbytes/sec] received # 表示网络传输速度，计算公式：Total trnasferred/ Time taken for tests，统计很好的说明服务器的处理能力达到极限时，其出口宽带的需求量。

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:       11   33  14.5     32      75
Processing:   189  318  57.5    318     453
Waiting:       12   37  16.2     36     107
Total:        240  351  57.4    348     473

# 用于描述每个请求处理时间的分布情况，这个处理时间是指前面的 Time per request，即对于单个用户而言，平均每个请求的处理时间。
Percentage of the requests served within a certain time (ms)
```

(下页继续)

(续上页)

```
 50%    348
 66%    377
 75%    389
 80%    407
 90%    433
 95%    455
 98%    465
 99%    473
100%    473 (longest request)
```

对于大文件的请求测试，这个值很容易成为系统瓶颈所在。要确定该值是不是瓶颈，需要了解客户端和被测服务器之间的网络情况，包括网络带宽和网卡速度等信息。

显示结果:

```
➜  ~ ab -n 100 -c 10 http://www.163.com/index.html
This is ApacheBench, Version 2.3 <$Revision: 1430300 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking www.163.com (be patient).....done


Server Software:        nginx
Server Hostname:        www.163.com
Server Port:            80

Document Path:          /index.html
Document Length:        697521 bytes

Concurrency Level:      10
Time taken for tests:   3.343 seconds
Complete requests:      100
Failed requests:        0
Write errors:           0
Total transferred:      69794600 bytes
HTML transferred:       69752100 bytes
Requests per second:    29.91 [#/sec] (mean)
Time per request:       334.330 [ms] (mean)
Time per request:       33.433 [ms] (mean, across all concurrent requests)
Transfer rate:          20386.66 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:       11   32  12.0     32      70
Processing:   148  292  71.4    282     530
Waiting:       17   32  11.7     30      89
Total:        203  324  66.6    310     568

Percentage of the requests served within a certain time (ms)
  50%    310
  66%    346
  75%    373
  80%    394
  90%    428
  95%    434
  98%    452
  99%    568
 100%    568 (longest request)
➜  ~
```

10.6 其它使用场景

```
#ab 进行 app 接口的压测:
```

(下页继续)

(续上页)

```
ab -n 400 -c20  "http://www.xxx.com/api.php?sig=......";

将需要压测的接口，用  " " ;

#ab 进行 post 传参的压测
# 将 parm.txt 放在和 ab.exe 相同的文件夹中，parm.txt 中存放的是需要 post 格式传递的参数。-T␣
↪: post 请求的 head 头
ab -n 400 -c20  -p  parm.txt  -T "application/x-www-form-urlencoded" http://
↪localhost:3000/login
```

CHAPTER 11

运维经典面试题合集

11.1 1. 如何查看 HTTP 的并发请求数与其 TCP 连接状态？

```
netstat -n | awk '/^tcp/ {++b[$NF]} END {for(a in b) print a,b[a]}'
```

```
88 10    223.171
53 11    .125.218
19 21    .106.143
14 11    .98.4
13 12    .1.13.210
```

Linux 服务器打开文件数也是影响并发的重要一环，具体可以查看该文件配置：/etc/security/limits.conf

当然同级目录下面的 limits.d 目录下的的配置文件也需要关注，也可以使用 ulimit -n 查看当前的配置数量。

11.2 2. 查看每个 IP 地址的连接数

```
netstat -n | awk '/^tcp/ {print $5}' | awk -F: '{print $1}' | sort | uniq -c | sort -rn
```

```
[root@zabbix-server ~]# netstat -n | awk '/^tcp/ {++b[$NF]} END {for(a in b) print a,b[a]}'
ESTABLISHED 35
TIME_WAIT 233
```

11.3 3. 通过 tcpdump 查看 80 端口访问量排前 10

```
tcpdump -i eth0 -tnn dst port 80 -c 1000 | awk -F"." '{print $1"."$2"."$3"."$4}' | sort␣
↪| uniq -c | sort -nr |head -10
```

注意自己机器的网卡名称

11.4 4. 统计 access.log 中访问量前 10 的 IP

```
cat access.log | awk '{print $1}' | sort | uniq -c | sort -n -r | head -10
```

```
[root@testserver03 mal ]# cat access.log | awk '{print $1}' | sort | uniq -c | sort -n -r | head -10
    733 192.168.10.152
    711 192.168.10.231
    550 192.168.10.32
    408 192.168.10.233
    390 192.168.10.241
     99 192.168.10.98
     91 192.168.10.167
     52 192.168.10.85
     46 192.168.10.185
     45 192.168.10.100
```

可以通过这个操作判断有些非人为操作

11.5 5. 只查看/var/log 这一级目录下面的文件

```
find /var/log -maxdepth 1 -type f
```

简单的 find 命令考察，有意思的就是 -maxdepth 参数，如果不加默认会把该目录下的其它目录下的子文件也显示

这个在用于我们按照时间点删除某个目录下面的文件的时候特别有用

11.6 6. 生成 32 位的随机码

```
cat /dev/urandom | head -1 | md5sum | head -c 32
```

11.7 7. 如何查看二进制文件的内容？

一般通过 hexdump 命令查看，用法：hexdump -C 文件名，没怎么用过！

-C 是比较规范的十六进制和 ASCII 码显示

-c 是单字节字符显示

-b 单字节八进制显示

-o 是双字节八进制显示

-d 是双字节十进制显示

-x 是双字节十六进制显示

显示样式:

```
0001b250  d6 26 40 00 00 00 00 00  e6 26 40 00 00 00 00 00  |.&@......&@.....|
0001b260  f6 26 40 00 00 00 00 00  06 27 40 00 00 00 00 00  |.&@......'@.....|
0001b270  16 27 40 00 00 00 00 00  26 27 40 00 00 00 00 00  |.'@.....&'@.....|
0001b280  36 27 40 00 00 00 00 00  46 27 40 00 00 00 00 00  |6'@.....F'@.....|
```

11.8 8. ps aux 中的 VSZ 代表什么意思，RSS 代表什么意思？

VSZ：虚拟内存集，进程占用的虚拟内存空间

RSS：物理内存集，进程战用实际物理内存空间

11.9 9. 如何检测并修复 /dev/hda1？

fsck 命令

不应该用 fsck 检查已挂载的磁盘，这很可能会对磁盘造成永久性的伤害。因此在开始使用 fsck 之前，我们需要使用命令来卸载磁盘

```
umount /dev/hda1

# 检查文件系统错误并自动修复
fsck -a /dev/hda1
```

11.10 10. Linux 系统的开机启动顺序：

加载 BIOS –> 读取 MBR –> Boot Loader –> 加载内核 –> 用户层 init （根据 inittab 设定系统运行的等级：一般 3 或 5）–> init 进程执行 rc.syninit –> 启动内核模块 –> 执行不同级别运行的脚本程序 –> 执行 /etc/rc.d/rc.local –> 执行 /bin/login

11.11 11. 软连接和硬链接的区别：

软连接（符号链接），类似 windows 系统里的快捷方式硬链接，类似复制了一份，但是会跟着文件的改变而改变，但是不会因为删除而影响另一个

```
# 软链接
ln -s  123.txt 456.txt  # 可以理解给给 123.txt 创建了一个快捷方式 456.txt

# 硬连接
ln 123.txt 456.txt
```

11.12 12. FTP 的主动模式和被动模式：

FTP 有两种工作方式：PORT 方式和 PASV 方式

PORT（主动）：

1. 客户端向服务端的 FTP 端口（默认是 21）发送连接请求
2. 服务端接受连接，建立一条命令链路
3. 传送数据时，客户端在命令链路上用 PORT 命令告诉服务端自己打开的随机端口
4. 服务端从 20 端口向收到的客户端的随机端口发送连接请求，建立一条数据链路来传送数据

PASV（被动）：

1. 客户端向服务端的 FTP 端口（默认是 21）发送连接请求，服务器接受连接
2. 建立一条命令链路，当传送数据时，服务端通过命令链路自己打开随机端口
3. 客户端向服务端的随机端口发送连接请求，建立一条数据链路来传送数据

11.13 13. 显示 /etc/inittab 中以 # 开头，且后面跟了一个或者多个空白字符，而后又跟了任意非空字符的行

```
grep "^\#[[:space:]]\{1,\}.\{1,\}" /etc/inittab
```

```
[root@VMServer tmp]# grep "^\#[[:space:]]\{1,\}.\{1,\}" /etc/inittab
# inittab is no longer used when using systemd.
# ADDING CONFIGURATION HERE WILL HAVE NO EFFECT ON YOUR SYSTEM.
# Ctrl-Alt-Delete is handled by /usr/lib/systemd/system/ctrl-alt-del.target
# systemd uses 'targets' instead of runlevels. By default, there are two main targets:
# multi-user.target: analogous to runlevel 3
# graphical.target: analogous to runlevel 5
# To view current default target, run:
# systemctl get-default
# To set a default target, run:
# systemctl set-default TARGET.target
```

说明：这个一般用于剔除注释内容

11.14 14. 显示 /tmp/1.txt 中包含了: 一个数字: 的行

```
grep "^\([0-9]\).*\1$" /tmp/1.txt
```

11.15 15. 批量添加 10 个用户，用户名为 user01 - user10，密码为 user 后面跟 5 个随机字符

```
#!/bin/bash
#1 是产生 0 -10 的数并按照格式输出，2 是产生密码随机数并截取，3 是命令行添加密码

for i in `seq -f "%02g" 1 10`;do
    useradd user$i
    echo "user$i`echo $RANDOM|md5sum|cut -c 1-5`" | passwd --stdin user$i >/dev/null 2>&1
done
```

11.16 16. 判断 192.168.1.0/24 网络里，当前在线的 IP 有哪些，能 ping 通则认为在线

```
#!/bin/bash

for ip in `seq 1 255`;do
    ping -c 3 192.168.1.$ip > /dev/null 2>&1
    if [ $? -eq 0 ];then
        echo "192.168.1.$ip UP"
```

(下页继续)

(续上页)

```
    else
        echo "192.168.1.$ip DOWN"
    fi
done
```

1. 删除一个目录下的所有文件，但保留一个指定文件

```
假设这个目录是/xx/，里面有 file1,file2,file3..file10 十个文件
[root@oldboy xx]# touch file{1..10}
[root@oldboy xx]# ls
file1 file10 file2 file3 file4 file5 file6 file7 file8 file9


# 方法一：find
[root@oldboy xx]# find /xx -type f ! -name "file10"|xargs rm -f
[root@oldboy xx]# ls
file10

[root@oldboy xx]# find /xx -type f ! -name "file10" -exec rm -f {} \;
[root@oldboy xx]# ls
file10

这两种一个通过 xargs 传参，一个通过 find 的-exec 执行命令参数来完成，都算作 find 吧

# 方法二：rsync
[root@oldboy xx]# ls
file1 file10 file2 file3 file4 file5 file6 file7 file8 file9
[root@oldboy xx]# rsync -az --delete --exclude "file10" /null/ /xx/
[root@oldboy xx]# ls
file10

# 方法四
find ./ -type f|grep -v "\file10\b"|xargs rm -f


# 方法五
rm -f `ls|grep -v "\file10\b"`
```

从运维角度，任何删除性的操作都应该事先备份后在执行或者确认有备份存在。

11.17 参考文章

本面试题合集主要来自众多运维博主的文章, 主要来自:

Linux 运维工程师经典面试题合集

linux 运维面试题

CHAPTER 12

Kubernetes 权威指南 1.3 案例部署

12.1 环境

```
[root@master1 ~]# kubectl get node -o wide
NAME      STATUS   ROLES    AGE   VERSION   INTERNAL-IP      EXTERNAL-IP   OS-IMAGE␣
↪          KERNEL-VERSION                CONTAINER-RUNTIME
master1   Ready    master   91m   v1.15.4   192.168.252.79   <none>        CentOS Linux␣
↪7 (Core)   4.4.195-1.el7.elrepo.x86_64   docker://19.3.2
node1     Ready    <none>   89m   v1.15.4   192.168.252.88   <none>        CentOS Linux␣
↪7 (Core)   4.4.196-1.el7.elrepo.x86_64   docker://19.3.2
node2     Ready    <none>   81m   v1.15.4   192.168.252.89   <none>        CentOS Linux␣
↪7 (Core)   4.4.196-1.el7.elrepo.x86_64   docker://19.3.2
```

12.2 创建 mysql 服务

vim mysql-rc.yaml

```
apiVersion: v1
# 副本控制器 RC
kind: ReplicationController
metadata:
```

(下页继续)

(续上页)

```
  # rc 的名称，全局唯一
  name: mysql
spec:
  # 期待创建的 pod 个数
  replicas: 1
  selector:
    # 选择符合拥有此标签的 pod
    app: mysql
  # 根据模板定义的信息创建 pod
  template:
    metadata:
      labels:
        # pod 拥有的标签，对应上边 RC 的 selector
        app: mysql
    # 定义 Pod 细则
    spec:
      containers:
      - name: mysql
        image: mysql:5.7
        # 容器应用监听的端口号
        ports:
        - containerPort: 3306
        # 注入到容器中的环境变量
        env:
        - name: MYSQL_ROOT_PASSWORD
          value: "123456"
```

vim mysql-svc.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: mysql
spec:
  ports:
   - port: 3306
  selector:
    app: mysql
```

开始创建服务

```
kubectl apply -f myql-rc.yaml
kubectl apply -f myql-svc.yaml
```

12.3 创建 Tomcat 服务

vim myweb-rc.yml

```
apiVersion: v1
kind: ReplicationController
metadata:
  name: myweb
spec:
  replicas: 2
  selector:
    app: myweb
  template:
    metadata:
      labels:
        app: myweb
    spec:
      containers:
      - name: myweb
        image: kubeguide/tomcat-app:v1
        ports:
        - containerPort: 8080
        env:
          - name: MYSQL_SERVICE_HOST
            value: 'mysql'
          - name: MYSQL_SERVICE_PORT
            value: '3306'
```

上面文件中已用了 MYSQL_SERVICE_HOST、MYSQL_SERVICE_PORT 环境变量，mysql 正是上文中定义的 MySQL 服务名。

vim myweb-svc.yml

```
apiVersion: v1
kind: Service
metadata:
  name: myweb
```

(下页继续)

(续上页)

```
spec:
  type: NodePort
  ports:
   - port: 8080
     nodePort: 30001
  selector:
    app: myweb
~
```

开始创建服务

```
kubectl apply -f myweb-rc.yaml
kubectl apply -f myweb-svc.yaml
```

12.4 验证

```
[root@master1 ~]# kubectl get pod,svc -o wide
NAME              READY   STATUS    RESTARTS   AGE   IP           NODE    NOMINATED NODE␣
↪  READINESS GATES
pod/mysql-mpqwv   1/1     Running   0          19m   172.32.2.5   node2   <none>         ␣
↪  <none>
pod/myweb-5fw65   1/1     Running   0          17m   172.32.2.6   node2   <none>         ␣
↪  <none>
pod/myweb-jn5xv   1/1     Running   0          17m   172.32.1.6   node1   <none>         ␣
↪  <none>

NAME                 TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE   ␣
↪SELECTOR
service/kubernetes   ClusterIP   172.16.0.1       <none>        443/TCP          102m
↪<none>
service/mysql        ClusterIP   172.21.225.229   <none>        3306/TCP         39m   ␣
↪app=mysql
service/myweb        NodePort    172.25.154.78    <none>        8080:30001/TCP   32m   ␣
↪app=myweb
```

浏览器访问 http://192.168.252.79:30001/demo/ 验证

12.5 参考文章

https://blog.csdn.net/wo18237095579/article/details/89376877

CHAPTER 13

使用 kubeadm 工具快速安装 k8s 集群

13.1 机器配置

master 192.168.252.79 node1 192.168.252.88 node2 192.168.252.89

13.2 准备工作 (三台机器执行)

```
# 关闭 selinux

setenforce 0
sed --follow-symlinks -i "s/SELINUX=enforcing/SELINUX=disabled/g" /etc/selinux/config

# 关闭防火墙

systemctl stop iptables && systemctl disable iptables
systemctl stop firewalld  && systemctl disable firewalld


# 修改 hostname

hostnamectl set-hostname 主机名
```

(下页继续)

(续上页)

```
# 所有机器添加 hosts

echo "192.168.252.79 master1" >> /etc/hosts
echo "192.168.252.88 node1" >> /etc/hosts
echo "192.168.252.89 node2" >> /etc/hosts

# 安装 gcc 开发工具等
yum install -y  gcc*

yum install -y vim-enhanced wget  bash-completion  lrzsz ntpdate sysstat iftop htop␣
↪dstat lsof chkconfig unzip telnet nmap net-tools git bzip2  bind-utils

yum install -y expat-devel  pcre-devel libxml2-devel  openssl openssl-devel  bzip2-devel␣
↪ libjpeg-devel  libpng-devel  freetype-devel  libXpm-devel  libmcrypt-devel  ␣
↪libaio  libaio-devel  php-mysqlnd  mysql-devel gd-devel  gdbm-devel  glib2-devel ␣
↪libdb4-devel  libdb4-devel  libicu-devel  libxslt-devel  readline-devel  xmlrpc-
↪c  xmlrpc-c-devel curl-devel yum-utils device-mapper-persistent-data lvm2  conntrack-
↪tools

# 时间同步

ntpdate 1.cn.pool.ntp.org

echo "*/15 * * * * /usr/sbin/ntpdate 1.cn.pool.ntp.org  >/dev/null 2>&1"  >>/var/spool/
↪cron/root

# 升级/重启

yum update -y
reboot

# 配置免密登录

把公钥传去其他每台机器，当然如果借助 ansible 或者脚本之类更方便

ssh-keygen -t rsa
ssh-copy-id -i ~/.ssh/id_rsa.pub root@192.168.252.62
ssh-copy-id -i ~/.ssh/id_rsa.pub root@192.168.252.63
ssh-copy-id -i ~/.ssh/id_rsa.pub root@192.168.252.64
```

(下页继续)

(续上页)

```
# 机器参数修改

cat <<EOF | tee /etc/sysctl.d/k8s.conf
net.ipv4.ip_forward = 1
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
EOF

sysctl -p /etc/sysctl.d/k8s.conf
```

13.2.1 升级最新稳定版内核 (选做)

```
#! /bin/bash
rpm --import https://www.elrepo.org/RPM-GPG-KEY-elrepo.org
rpm -Uvh http://www.elrepo.org/elrepo-release-7.0-3.el7.elrepo.noarch.rpm
yum history new
yum -y --disablerepo=* --enablerepo=elrepo-kernel install kernel-lt   #lt 长期稳定版 ml 最新版
grub2-set-default 0
```

reboot

13.2.2 安装 docker

yum install -y yum-utils device-mapper-persistent-data lvm2 yum-config-manager –add-repo https://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo

查看最新的 Docker 版本

yum list docker-ce.x86_64 –showduplicates |sort -r

yum makecache fast yum install –setopt=obsoletes=0 docker-ce-18.06.3.ce-3.el7 -y systemctl start docker && systemctl enable docker

卸载 docker

yum remove docker docker-client docker-client-latest docker-common docker-latest docker-latest-logrotate docker-logrotate docker-selinux docker-engine-selinux docker-engine

13.2.3 安装 kubeadm 和相关工具

cat < /etc/yum.repos.d/kubernetes.repo [kubernetes] name=Kubernetes baseurl=https://mirrors.aliyun.com/kubernetes/yum/repos/kubernetes-el7-x86_64/ enabled=1 gpgcheck=0 EOF

yum install -y kubelet kubeadm kubectl –disableexcludes=Kubernetes 或者 yum install kubelet-1.15.4 kubeadm-1.15.4 kubectl-1.15.4 –disableexcludes=kubernetes

systemctl enable kubelet.service && systemctl start kubelet.service

13.3 安装 master

kubeadm config print init-defaults > init-config.yaml # 获取默认的初始化参数文件

编辑 init-config.yaml , 定制镜像仓库地址,pod 地址范围,service 地址范围, 完整内容如下

```
apiVersion: kubeadm.k8s.io/v1beta2
imageRepository: registry.cn-hangzhou.aliyuncs.com/google_containers
kind: ClusterConfiguration
kubernetesVersion: v1.15.0
networking:
  dnsDomain: cluster.local
  serviceSubnet: "172.16.0.0/16"
  podSubnet: "172.32.0.0/16"
scheduler: {}
```

```
kubeadm config images pull  --config=init.config.yaml  # 下载镜像
kubeadm init  --config=init.config.yaml # 初始化安装
```

安装成功最后底下文字

```
Then you can join any number of worker nodes by running the following on each as root:

kubeadm join 192.168.252.79:6443 --token fu6k4l.5b4jspxqxajsd2la \
    --discovery-token-ca-cert-hash␣
↪sha256:ee0836e5ac9db0780a190a19c46323d0a32909d758632703b3340a0c30b34228
```

```
# 执行操作
[root@master1 ~]# mkdir -p $HOME/.kube
[root@master1 ~]# sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
[root@master1 ~]# sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

13.3.1 去除 master 上污点

kubectl taint nodes –all node-role.kubernetes.io/master-

13.4 安装 node

kubeadm config print join-defaults > join-confog.yaml

编辑 join-confog.yaml, apiServerEndpoint 值为 master 的 ip, token 和 tlsBootstrapToken 的值来自使用 kubeadm init 安装 master 的最后一段信息

```
apiVersion: kubeadm.k8s.io/v1beta2
caCertPath: /etc/kubernetes/pki/ca.crt
discovery:
  bootstrapToken:
    apiServerEndpoint: 192.168.252.79:6443
    token: fu6k4l.5b4jspxqxajsd2la
    unsafeSkipCAVerification: true
  timeout: 5m0s
  tlsBootstrapToken: fu6k4l.5b4jspxqxajsd2la
kind: JoinConfiguration
nodeRegistration:
  criSocket: /var/run/dockershim.sock
  name: node1
  taints: null
```

```
kubeadm join --config=join-defaults.yaml # 将 node 加入集群
```

13.5 安装网络插件

wget https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml

修改 kube-flannel.yml, pod 范围要和之前 init-config.yaml 中匹配

```
  net-conf.json: |
    {
      "Network": "172.32.0.0/16",
      "Backend": {
        "Type": "vxlan"
      }
    }
```

```
kubectl apply -f kube-flannel.yml
```

13.6 验证

```
[root@master1 ~]# kubectl get -n kube-system configmap
NAME                                 DATA   AGE
coredns                              1      45m
extension-apiserver-authentication   6      45m
kube-flannel-cfg                     2      36m
kube-proxy                           2      45m
kubeadm-config                       2      45m
kubelet-config-1.15                  1      45m

[root@master1 ~]# kubectl get node
NAME      STATUS   ROLES    AGE   VERSION
master1   Ready    master   45m   v1.15.4
node1     Ready    <none>   43m   v1.15.4
node2     Ready    <none>   35m   v1.15.4

[root@master1 ~]# kubectl get -n kube-system pod
NAME                              READY   STATUS    RESTARTS   AGE
coredns-6967fb4995-dtjcj          1/1     Running   0          45m
coredns-6967fb4995-g27s2          1/1     Running   0          45m
etcd-master1                      1/1     Running   0          44m
kube-apiserver-master1            1/1     Running   0          44m
kube-controller-manager-master1   1/1     Running   0          44m
kube-flannel-ds-amd64-9245r       1/1     Running   0          35m
kube-flannel-ds-amd64-95gnl       1/1     Running   0          37m
kube-flannel-ds-amd64-r29pl       1/1     Running   0          37m
kube-proxy-6rls8                  1/1     Running   0          45m
kube-proxy-c5zs7                  1/1     Running   0          35m
kube-proxy-lhsxn                  1/1     Running   0          43m
kube-scheduler-master1            1/1     Running   0          44m
```

13.7 其它命令

```
kubeadm reset # 重置
kubectl get nodes -o jsonpath='{.items[*].spec.podCIDR}'
```

13.8 参考

https://blog.51cto.com/michaelkang/2432048

本博客仿照 MING 哥的博客完成，在此特别感谢 MING 哥的支持和帮助。

CHAPTER 14

友情链接

MING 哥的博客

MING 哥的 github