在Ubuntu操作系统里安装Docker

Docker 是一个开源的应用容器引擎，让开发者可以打包他们的应用以及依赖包到一个可移植的容器中，然后发布到任何流行的 Linux 机器上，也可以实现虚拟化。容器是完全使用沙箱机制，相互之间不会有任何接口。

1. 由于Ubuntu里apt官方库里的docker版本可能比较低，因此先用下面的命令行卸载旧版本（如果有的话）

   ```
   sudo apt-get remove docker docker-engine docker-ce docker.io
   ```

1.安装docker CE时

```
sudo apt-get install -y apt-transport-https ca-certificates curl software-
properties-common
```

2添加Docker官方GPG key, 需要切换到root账号

```
sudo su root
sudo curl -fsSL https://mirrors.aliyun.com/docker-ce/linux/ubuntu/gpg | apt-key
add -
```

3.添加最新或测试repository

```
add-apt-repository "deb [arch=amd64] https://mirrors.aliyun.com/docker-
ce/linux/ubuntu $(lsb_release -cs) stable"
```

4.更新apt包索引

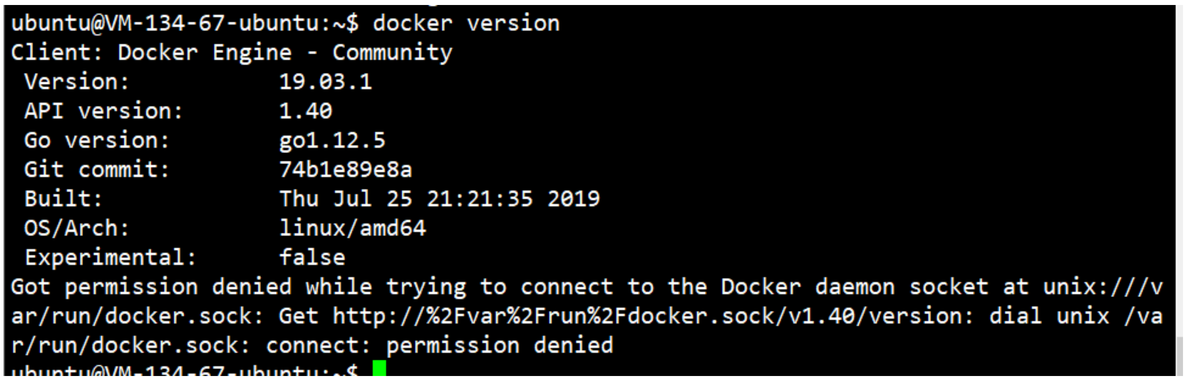
```
apt-get update
```

5.安装最新版本的Docker CE

```
apt-get install -y docker-ce
```

6.查看docker版本

```
docker version
```



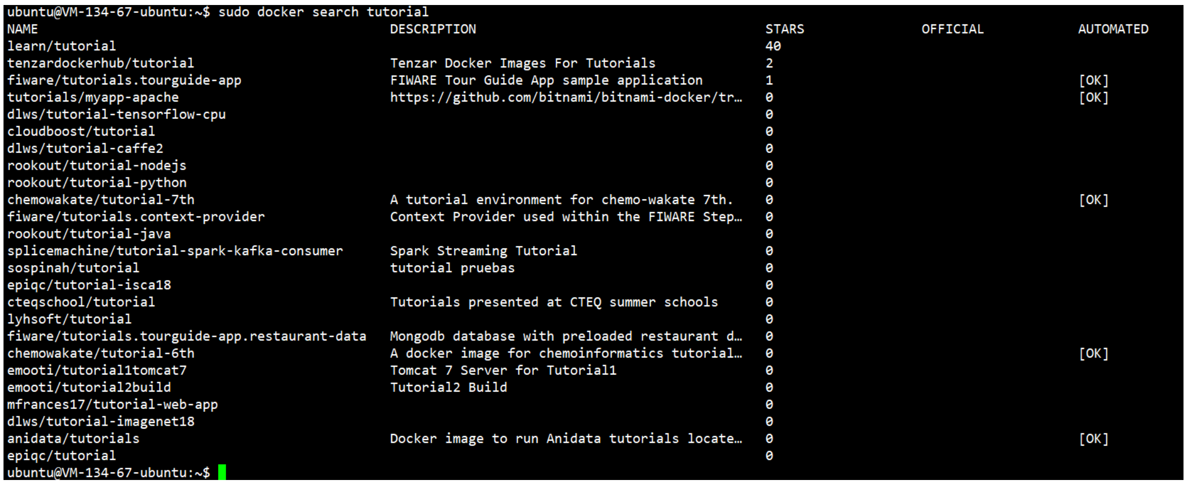
```
ubuntu@VM-134-67-ubuntu:~$ docker version
Client: Docker Engine - Community
 Version:           19.03.1
 API version:       1.40
 Go version:        go1.12.5
 Git commit:        74b1e89e8a
 Built:             Thu Jul 25 21:21:35 2019
 OS/Arch:           linux/amd64
 Experimental:      false
Got permission denied while trying to connect to the Docker daemon socket at unix:///v
ar/run/docker.sock: Get http://%2Fvar%2Frun%2Fdocker.sock/v1.40/version: dial unix /va
r/run/docker.sock: connect: permission denied
ubuntu@VM-134-67-ubuntu:~$
```

7.使用命令sudo docker run hello-world，能观察到从远程下载这个测试用的容器：Pulling from library/hello-world:然后看到打印消息：Hello from Docker! 说明Docker安装成功。

```
![1567308907618](note1.assets/1567308907618.png)
```

8.搜索可用的docker镜像

```
sudo docker search tutorial
```



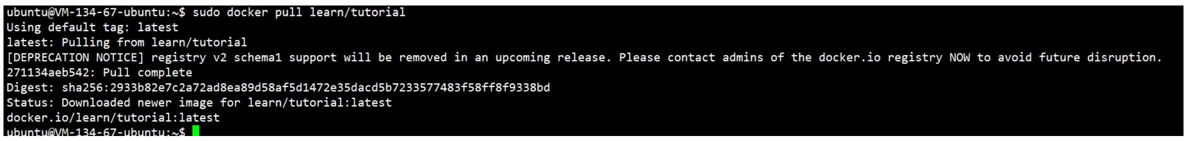
```
ubuntu@VM-134-67-ubuntu:~$ sudo docker search tutorial
NAME                                        DESCRIPTION                                     STARS               OFFICIAL            AUTOMATED
learn/tutorial                                                                              40
tenzardockerhub/tutorial                    Tenzar Docker Images For Tutorials              2
fiware/tutorials.tourguide-app              FIWARE Tour Guide App sample application        1                                       [OK]
tutorials/myapp-apache                      https://github.com/bitnami/bitnami-docker/tr…   0                                       [OK]
dlws/tutorial-tensorflow-cpu                                                                0
cloudboost/tutorial                                                                         0
dlws/tutorial-caffe2                                                                        0
rookout/tutorial-nodejs                                                                     0
rookout/tutorial-python                                                                     0
chemowakate/tutorial-7th                    A tutorial environment for chemo-wakate 7th.    0                                       [OK]
fiware/tutorials.context-provider           Context Provider used within the FIWARE Step…   0
rookout/tutorial-java                                                                       0
splicemachine/tutorial-spark-kafka-consumer Spark Streaming Tutorial                        0
sospinah/tutorial                           tutorial pruebas                                0
epiqc/tutorial-isca18                                                                       0
cteqschool/tutorial                         Tutorials presented at CTEQ summer schools      0
lyhsoft/tutorial                                                                            0
fiware/tutorials.tourguide-app.restaurant-data Mongodb database with preloaded restaurant d… 0
chemowakate/tutorial-6th                    A docker image for chemoinformatics tutorial…   0                                       [OK]
emooti/tutorial1tomcat7                     Tomcat 7 Server for Tutorial1                   0
emooti/tutorial2build                       Tutorial2 Build                                 0
mfrances17/tutorial-web-app                                                                 0
dlws/tutorial-imagenet18                                                                    0
anidata/tutorials                           Docker image to run Anidata tutorials locate…   0                                       [OK]
epiqc/tutorial                                                                              0
ubuntu@VM-134-67-ubuntu:~$
```

9.下载容器镜像

学会使用docker命令来下载镜像 下载镜像的命令非常简单，使用docker pull命令即可。(译者按：docker命令和git有一些类似的地方）。在docker的镜像索引网站上面，镜像都是按照 用户名/镜像名的方式来存储的。有一组比较特殊的镜像，比如ubuntu这类基础镜像，经过官方的验证，值得信任，可以直接用 镜像名来检索到。

目标：通过docker命令下载tutorial镜像。

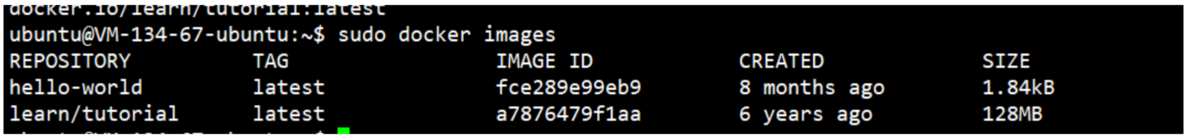
```
sudo docker pull learn/tutorial
```



```
ubuntu@VM-134-67-ubuntu:~$ sudo docker pull learn/tutorial
Using default tag: latest
latest: Pulling from learn/tutorial
[DEPRECATION NOTICE] registry v2 schema1 support will be removed in an upcoming release. Please contact admins of the docker.io registry NOW to avoid future disruption.
271134aeb542: Pull complete
Digest: sha256:2933b82e7c2a72ad8ea89d58af5d1472e35dacd5b7233577483f58ff8f9338bd
Status: Downloaded newer image for learn/tutorial:latest
docker.io/learn/tutorial:latest
ubuntu@VM-134-67-ubuntu:~$
```

10.查看镜像

```
sudo docker images
```



```
docker.io/learn/tutorial:latest
ubuntu@VM-134-67-ubuntu:~$ sudo docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
hello-world         latest              fce289e99eb9        8 months ago        1.84kB
learn/tutorial      latest              a7876479f1aa        6 years ago         128MB
```

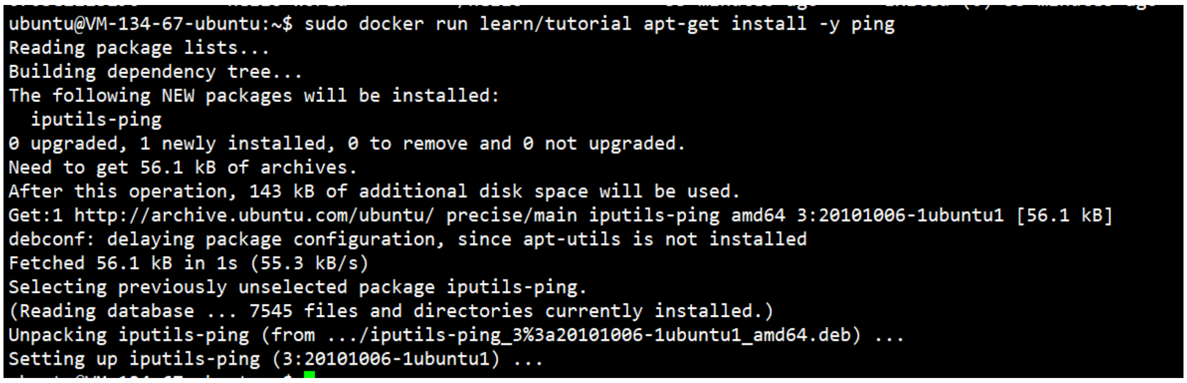
11.Docker 删除镜像

```
1. 查询镜像：docker images
    可以看到所有已经存在的镜像的 ID (IMAGE ID)
2. 查询容器：docker ps -a
    可以看到所有容器的 ID (CONTAINER ID)
3. 先删除容器：docker rm 容器ID
4. 再删除镜像：docker rmi 镜像ID
```

12.在容器中安装新的程序

下一步我们要做的事情是在容器里面安装一个简单的程序(ping)。我们之前下载的tutorial镜像是基于ubuntu的，所以你可以使用ubuntu的apt-get命令来安装ping程序： apt-get install -y ping

```
sudo docker run learn/tutorial apt-get install -y ping
```



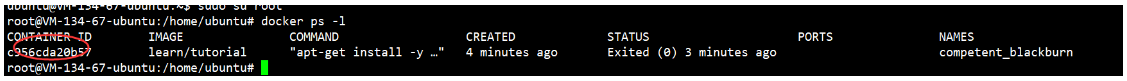
```
ubuntu@VM-134-67-ubuntu:~$ sudo docker run learn/tutorial apt-get install -y ping
Reading package lists...
Building dependency tree...
The following NEW packages will be installed:
  iputils-ping
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 56.1 kB of archives.
After this operation, 143 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu/ precise/main iputils-ping amd64 3:20101006-1ubuntu1 [56.1 kB]
debconf: delaying package configuration, since apt-utils is not installed
Fetched 56.1 kB in 1s (55.3 kB/s)
Selecting previously unselected package iputils-ping.
(Reading database ... 7545 files and directories currently installed.)
Unpacking iputils-ping (from .../iputils-ping_3%3a20101006-1ubuntu1_amd64.deb) ...
Setting up iputils-ping (3:20101006-1ubuntu1) ...
```

13.保存对容器的修改

简介：通过docker commit命令保存对容器的修改,保存对容器的修改,当你对某一个容器做了修改之后（通过在容器中运行某一个命令），可以把对容器的修改保存下来，这样下次可以从保存后的最新状态运行该容器。docker中保存状态的过程称之为committing，它保存的新旧状态之间的区别，从而产生一个新的版本。

1. 首先使用 **docker ps -l** 命令获得安装完ping命令之后容器的id

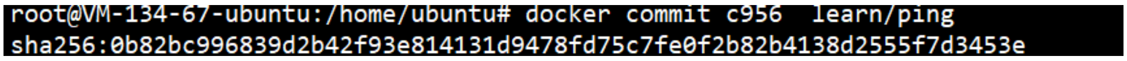
```
提示：
    1. 运行docker commit，可以查看该命令的参数列表。
    2. 你需要指定要提交保存容器的ID。(译者按：通过docker ps -l 命令获得)
    3. 无需拷贝完整的id，通常来讲最开始的三至四个字母即可区分。（译者按：非常类似git里面的版本号)
```



```
ubuntu@VM-134-67-ubuntu:~$ sudo su root
root@VM-134-67-ubuntu:/home/ubuntu# docker ps -l
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS                     PORTS               NAMES
c956cda20b57        learn/tutorial      "apt-get install -y …"   4 minutes ago       Exited (0) 3 minutes ago                       competent_blackburn
root@VM-134-67-ubuntu:/home/ubuntu#
```

2. 提交

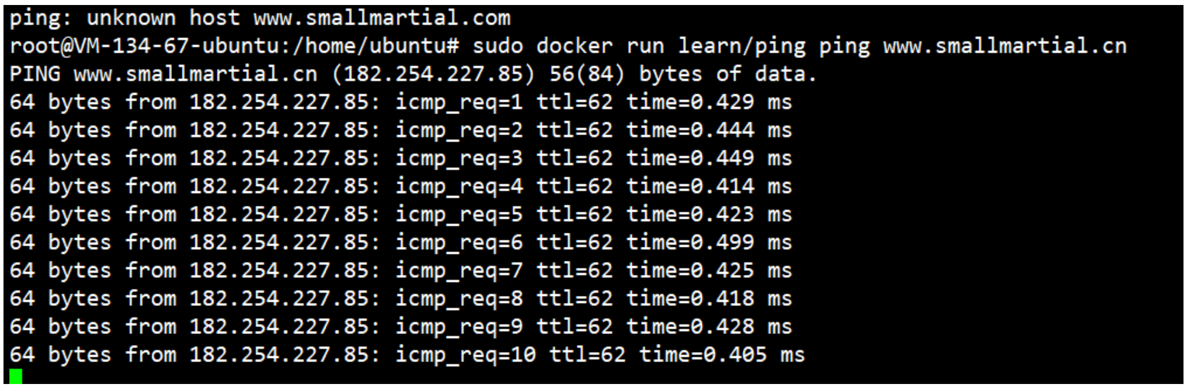
```
docker commit c956  learn/ping
```



```
root@VM-134-67-ubuntu:/home/ubuntu# docker commit c956  learn/ping
sha256:0b82bc996839d2b42f93e814131d9478fd75c7fe0f2b82b4138d2555f7d3453e
```

14.运行新的镜像

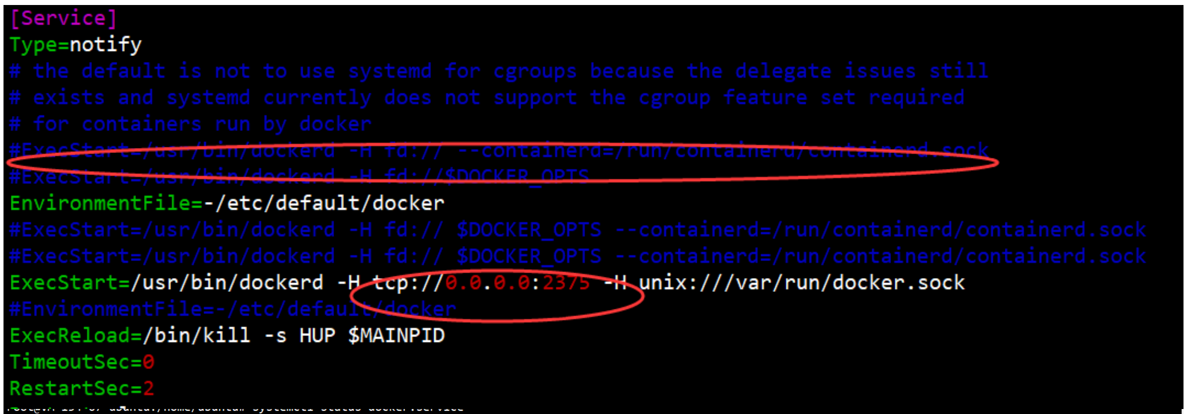
```
docker run learn/ping ping www.smallmartial.cn
```



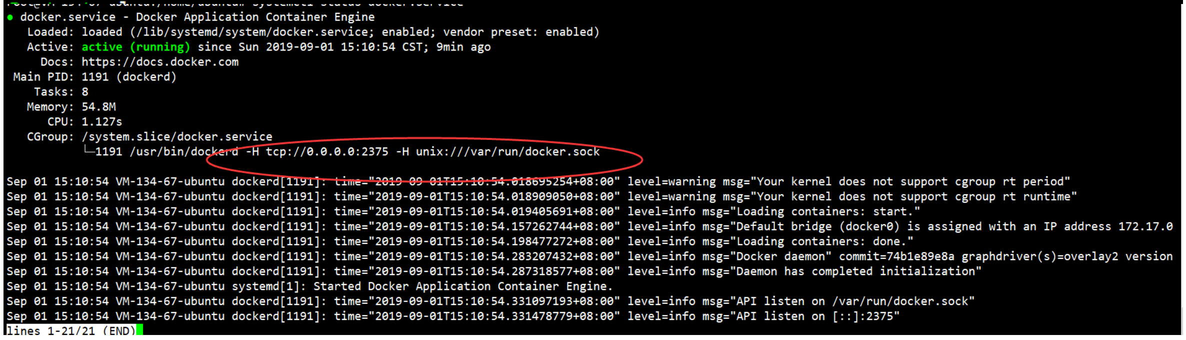
```
ping: unknown host www.smallmartial.com
root@VM-134-67-ubuntu:/home/ubuntu# sudo docker run learn/ping ping www.smallmartial.cn
PING www.smallmartial.cn (182.254.227.85) 56(84) bytes of data.
64 bytes from 182.254.227.85: icmp_req=1 ttl=62 time=0.429 ms
64 bytes from 182.254.227.85: icmp_req=2 ttl=62 time=0.444 ms
64 bytes from 182.254.227.85: icmp_req=3 ttl=62 time=0.449 ms
64 bytes from 182.254.227.85: icmp_req=4 ttl=62 time=0.414 ms
64 bytes from 182.254.227.85: icmp_req=5 ttl=62 time=0.423 ms
64 bytes from 182.254.227.85: icmp_req=6 ttl=62 time=0.499 ms
64 bytes from 182.254.227.85: icmp_req=7 ttl=62 time=0.425 ms
64 bytes from 182.254.227.85: icmp_req=8 ttl=62 time=0.418 ms
64 bytes from 182.254.227.85: icmp_req=9 ttl=62 time=0.428 ms
64 bytes from 182.254.227.85: icmp_req=10 ttl=62 time=0.405 ms
```

15.Ubuntu 18 下修改docker 配置文件不生效问题解决:

```
vim /lib/systemd/system/docker.service
#修改之后  执行以下代码
sudo systemctl daemon-reload
sudo service docker restart
#查看状态
systemctl status docker.service
```



```
[Service]
Type=notify
# the default is not to use systemd for cgroups because the delegate issues still
# exists and systemd currently does not support the cgroup feature set required
# for containers run by docker
#ExecStart=/usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
#ExecStart=/usr/bin/dockerd -H fd://$DOCKER_OPTS
EnvironmentFile=-/etc/default/docker
#ExecStart=/usr/bin/dockerd -H fd:// $DOCKER_OPTS --containerd=/run/containerd/containerd.sock
#ExecStart=/usr/bin/dockerd -H fd:// $DOCKER_OPTS --containerd=/run/containerd/containerd.sock
ExecStart=/usr/bin/dockerd -H tcp://0.0.0.0:2375 -H unix:///var/run/docker.sock
#EnvironmentFile=-/etc/default/docker
ExecReload=/bin/kill -s HUP $MAINPID
TimeoutSec=0
RestartSec=2
```



```
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
   Active: active (running) since Sun 2019-09-01 15:10:54 CST; 9min ago
     Docs: https://docs.docker.com
 Main PID: 1191 (dockerd)
    Tasks: 8
   Memory: 54.8M
      CPU: 1.127s
   CGroup: /system.slice/docker.service
           └─1191 /usr/bin/dockerd -H tcp://0.0.0.0:2375 -H unix:///var/run/docker.sock

Sep 01 15:10:54 VM-134-67-ubuntu dockerd[1191]: time="2019-09-01T15:10:54.018695254+08:00" level=warning msg="Your kernel does not support cgroup rt period"
Sep 01 15:10:54 VM-134-67-ubuntu dockerd[1191]: time="2019-09-01T15:10:54.018909050+08:00" level=warning msg="Your kernel does not support cgroup rt runtime"
Sep 01 15:10:54 VM-134-67-ubuntu dockerd[1191]: time="2019-09-01T15:10:54.019405691+08:00" level=info msg="Loading containers: start."
Sep 01 15:10:54 VM-134-67-ubuntu dockerd[1191]: time="2019-09-01T15:10:54.157262744+08:00" level=info msg="Default bridge (docker0) is assigned with an IP address 172.17.0
Sep 01 15:10:54 VM-134-67-ubuntu dockerd[1191]: time="2019-09-01T15:10:54.198477272+08:00" level=info msg="Loading containers: done."
Sep 01 15:10:54 VM-134-67-ubuntu dockerd[1191]: time="2019-09-01T15:10:54.283207432+08:00" level=info msg="Docker daemon" commit=74b1e89e8a graphdriver(s)=overlay2 version
Sep 01 15:10:54 VM-134-67-ubuntu dockerd[1191]: time="2019-09-01T15:10:54.287318577+08:00" level=info msg="Daemon has completed initialization"
Sep 01 15:10:54 VM-134-67-ubuntu systemd[1]: Started Docker Application Container Engine.
Sep 01 15:10:54 VM-134-67-ubuntu dockerd[1191]: time="2019-09-01T15:10:54.331097193+08:00" level=info msg="API listen on /var/run/docker.sock"
Sep 01 15:10:54 VM-134-67-ubuntu dockerd[1191]: time="2019-09-01T15:10:54.331478779+08:00" level=info msg="API listen on [::]:2375"
lines 1-21/21 (END)
```

```
源地址
ExecStart=/usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
修改为
ExecStart=/usr/bin/dockerd -H tcp://0.0.0.0:2375 -H unix:///var/run/docker.sock
```

`tcp://0.0.0.0:2375`对外访问端口