DaoCloud

Docker最佳实践

Agenda

一、	Docker安装
二、	应用容器化
三、	Docker常用命令
四、	容器编排

DaoCloud

Docker安装

DaoCloud

Docker 安装

https://docs.docker.com/install/linux/docker-ce/centos/#set-up-the-repository

os: centos7

docker-ce: 17.03.2.ce

```
cat <<EOF > /etc/yum.repos.d/docker-ce.repo
[docker-ce]
name = docker-ce
baseurl = https://mirrors.aliyun.com/docker-ce/linux/centos/7/x86_64/stable
gpgcheck = 0
enable = 1
EOF
```

yum -y install docker

```
yum -y install docker-ce-17.03.2.ce
systemctl start docker
systemctl enable docker
```

DaoCloud

Docker 常见错误

这是一个常见的问题，在Mac端也有这样的设置

```
root@default:~# docker pull 106.75.120.241/busybox
Using default tag: latest
Error response from daemon: Get https://106.75.120.241/v2/: x509: certificate has expired or is not yet valid
root@default:~# docker pull 192.168.130.1:5000/busybox
Using default tag: latest
Error response from daemon: Get https://192.168.130.1:5000/v2/: http: server gave HTTP response to HTTPS client
root@default:~#
```

解决:

```
cat > /etc/docker/daemon.json <<-EOF
{
  "insecure-registries":  [ "0.0.0.0/0"]
}
EOF

systemctl daemon-reload
systemctl restart docker
```

DaoCloud

应用容器化

DaoCloud

使用 Docker 运行 Tomcat + WAR 包 Java 应用

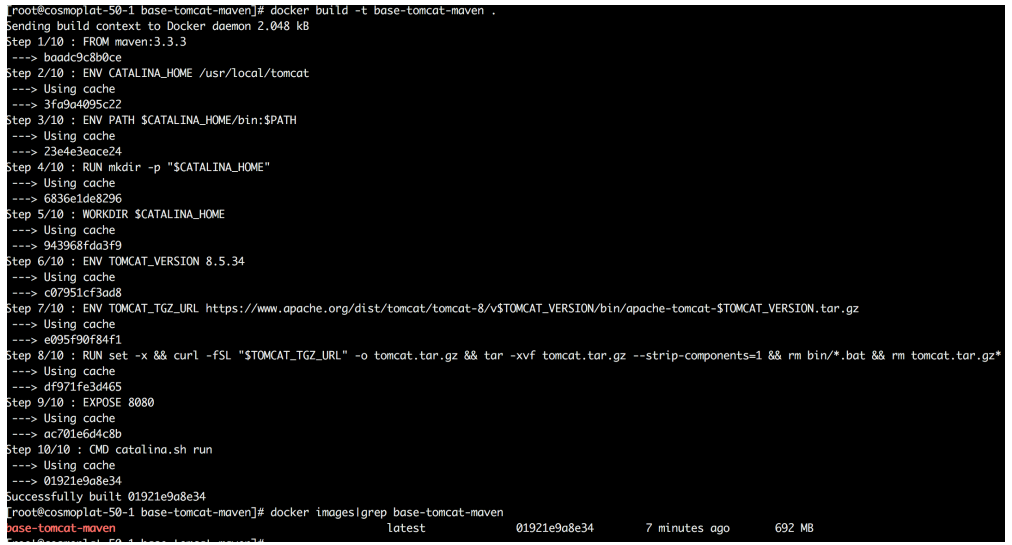
➢ **base-tomcat-maven**

mkdir base-tomcat-maven
cd base-tomcat-maven

cat > Dockerfile <<'EOF'

```
FROM maven:3.3.3
ENV CATALINA_HOME /usr/local/tomcat
ENV PATH $CATALINA_HOME/bin:$PATH
RUN mkdir -p "$CATALINA_HOME"
WORKDIR $CATALINA_HOME
ENV TOMCAT_VERSION 8.5.34
ENV TOMCAT_TGZ_URL https://www.apache.org/dist/tomcat/tomcat-8/v$TOMCAT_VERSION/bin/apache-tomcat-$TOMCAT_VERSION.tar.gz
RUN set -x \
&& curl -fSL "$TOMCAT_TGZ_URL" -o tomcat.tar.gz \
&& tar -xvf tomcat.tar.gz --strip-components=1 \
&& rm bin/*.bat \
&& rm tomcat.tar.gz*
EXPOSE 8080
CMD ["catalina.sh", "run"]
```

EOF



通过浏览器http://localhost:8080，就可以访问吗?

docker build -t base-tomcat-maven .

DaoCloud

使用 Docker 运行 Tomcat + WAR 包 Java 应用

➢ **docker-demo-java-tomcat**

git clone https://github.com/DaoCloud/docker-demo-java-tomcat.git
cd docker-demo-java-tomcat

sed -i '/^FROM/c FROM base-tomcat-maven' Dockerfile # 修改Dockerfile中的base镜像
docker build -t docker-demo-java-tomcat .

```
[root@cosmoplat-50-1 docker-demo-java-tomcat]# docker build -t docker-demo-java-tomcat .
Sending build context to Docker daemon 73.22 kB
Step 1/5 : FROM base-tomcat-maven
 ---> 01921e9a8e34
Step 2/5 : ADD pom.xml /tmp/build/
 ---> Using cache
 ---> 4dd29f70a171
Step 3/5 : RUN cd /tmp/build && mvn -q dependency:resolve
 ---> Using cache
 ---> 1dcd9916f36a
Step 4/5 : ADD src /tmp/build/src
 ---> Using cache
 ---> 64d8a177d900
Step 5/5 : RUN cd /tmp/build && mvn -q -DskipTests=true package        && rm -rf $CATALINA_HOME/webapps/*        && mv target/*.war $CATALINA_HOME/webapps/ROOT.war        && cd / && rm -rf /tmp/build
 ---> Using cache
 ---> 74df96895606
Successfully built 74df96895606
[root@cosmoplat-50-1 docker-demo-java-tomcat]# docker images|grep docker-demo-java-tomcat
docker-demo-java-tomcat                                 latest              74df96895606        10 minutes ago      692 MB
[root@cosmoplat-50-1 docker-demo-java-tomcat]# docker run -idt --name demo -p 8001:8080 docker-demo-java-tomcat
14078dd8434a8b48c153a448b0315c25e632125c80f98d583ff7a5a2b974d079
[root@cosmoplat-50-1 docker-demo-java-tomcat]# docker ps|grep demo
14078dd8434a        docker-demo-java-tomcat                                                                 "catalina.sh run"       25 seconds ago      Up 9 seconds
    0.0.0.0:8001->8080/tcp                                  demo
[root@cosmoplat-50-1 docker-demo-java-tomcat]# curl localhost:8001
<html>
<body>
<h2>Hello World!</h2>
</body>
</html>
```

DaoCloud

Docker常用命令

DaoCloud

docker image常用命令(man docker-images)

早期命令

```
# 拉镜像
docker pull hello-world
# 列出镜像
docker images hello-world
docker images --format="{{ .ID }}”
# 查看镜像详情
docker inspect hello-world
# 导出镜像
docker save hello-world|xz - > hello-world.xz
# 删除镜像
docker rmi –f hello-world
# 导入镜像
docker load -i hello-world.xz
```

推荐命令

```
# 拉镜像
docker image pull hello-world
# 列出镜像
docker image ls hello-world
docker image ls --format="{{ .ID }}”
# 查看镜像详情
docker image inspect hello-world
# 导出镜像
docker image save hello-world|xz - > hello-world.xz
# 删除镜像
docker image rm –f hello-world
# 导入镜像
docker image load -i hello-world.xz
```

DaoCloud

docker container常用命令(man docker-container)

早期命令

```
# 运行容器
docker run -idt --name mynginx -p 8000:80 nginx
# 容器start|stop|restart
docker stop mynginx
# chroot到容器
docker attach mynginx
docker exec -it mynginx sh
# 从容器chroot环境中退出
Ctl+p+q # 不停止容器，退出到宿主机shell
exit # 停止容器，退出到宿主机shell
# 不进入容器内执行容器命令
docker exec -it mynginx uname -a
# 删除容器
docker rm -f mynginx
```

推荐命令

```
# 运行容器
docker container run -idt --name mynginx -p 8000:80 nginx
# 容器start|stop|restart
docker container stop mynginx
# chroot到容器
docker container attach mynginx
docker container exec -it mynginx sh
# 从容器chroot环境中退出
Ctl+p+q # 不停止容器，退出到宿主机shell
exit # 停止容器，退出到宿主机shell
# 不进入容器内执行容器命令
docker container exec -it mynginx uname -a
# 删除容器
docker container rm -f mynginx
```

DaoCloud

docker network常用命令(man docker-network)

```
# 创建网络
docker network create -d bridge --subnet=192.168.0.0/16 br0
# 列出网络
docker network ls
# 指定网络中运行容器
docker run -idt --name mynginx --network=br0 -p 8000:80 nginx
# 查看网络详情
docker network inspect br0
# 删除网络
docker network rm br0
```

docker volume常用命令(man docker-volume)

docker volume create --name myvol # 创建卷

docker volume ls # 列出卷

docker volume inspect myvol # 查看卷详情

启动容器时附加卷

注意: -v参数在docker中存在了非常长的时间，一直保留至今，Docker 17.06及以上版本官方鼓励使用--mount。

--mount和-v有一个区别，其它功能完全相同，当挂载主机的目录不存在时，若使用-v参数，则docker会自动在宿主机创建该目录，--mount不会自动创建，会直接报错

docker run -idt --name devtest -v myvol:/app nginx:latest

docker run -idt --name devtest --mount source=myvol,target=/app nginx:latest

docker volume rm myvol # 删除卷

容器编排

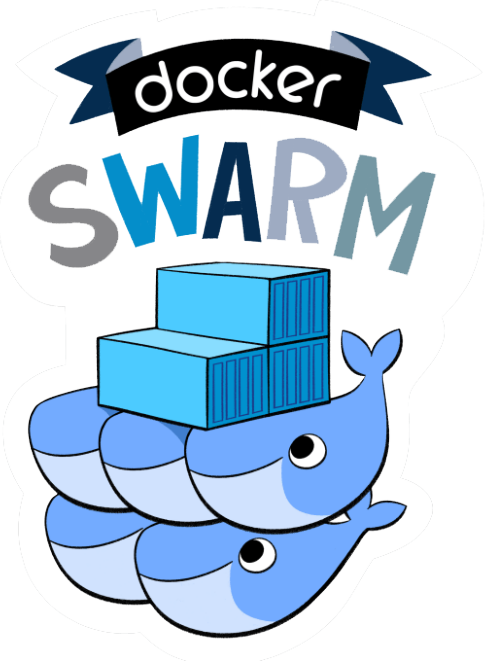
DaoCloud

容器编排引擎

- Docker https://www.docker.com
- Kubernetes https://kubernetes.io
- Mesos http://mesos.apache.org
- Rancher https://rancher.com
- Nomad https://www.nomadproject.io

kubernetes

MESOSPHERE

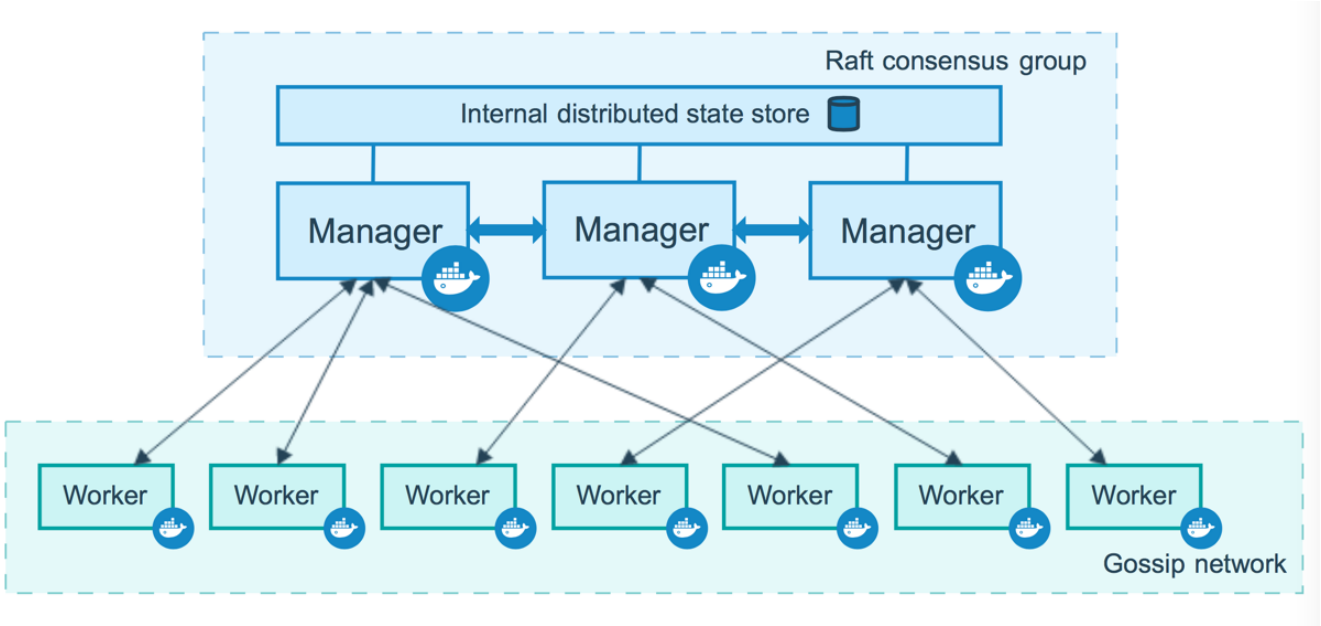


DaoCloud

Docker swarm介绍

https://docs.docker.com/swarm/overview/

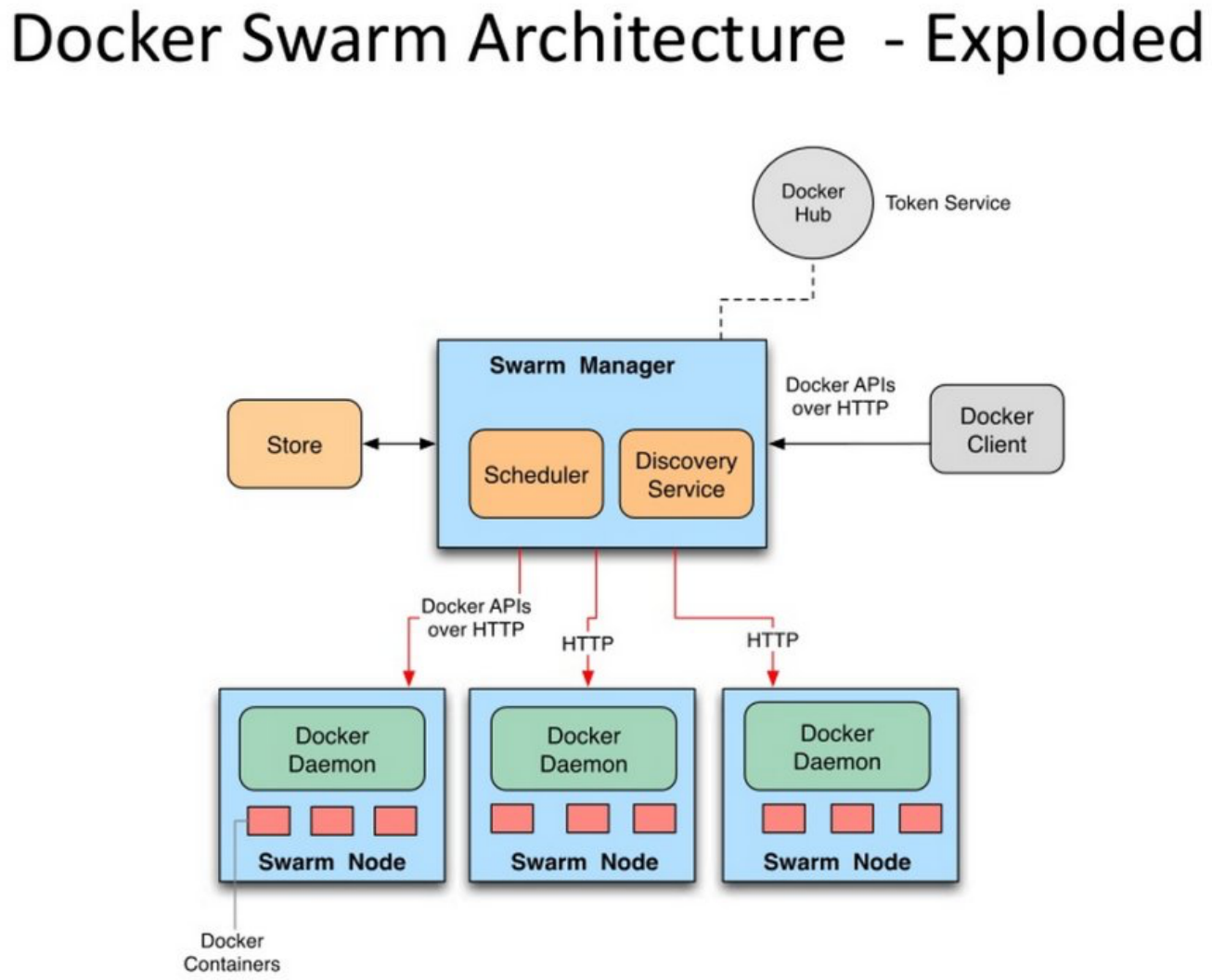
- Swarm是Docker官方提供的一款集群管理工具，Docker 1.12及以上版本直接集成swarm
- 其主要作用是把若干台Docker主机抽象为一个整体，并且通过一个入口统一管理这些Docker主机上的各种Docker资源
- Swarm相比较Kubernetes比较更轻量，具有的功能也较kubernetes更少(只有replicated service和global service)



DaoCloud

Docker swarmI架构

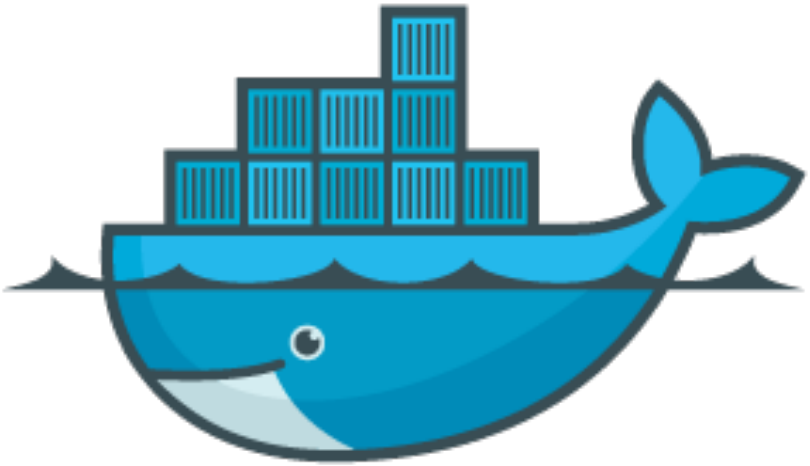
docker swarm init --advertise-addr <MANAGER-IP>
docker swarm join --token <TOKEN> <MANAGER-IP>:2377



DaoCloud

云原生操作系统Kubernetes

容器编排
领域领导者



DaoCloud

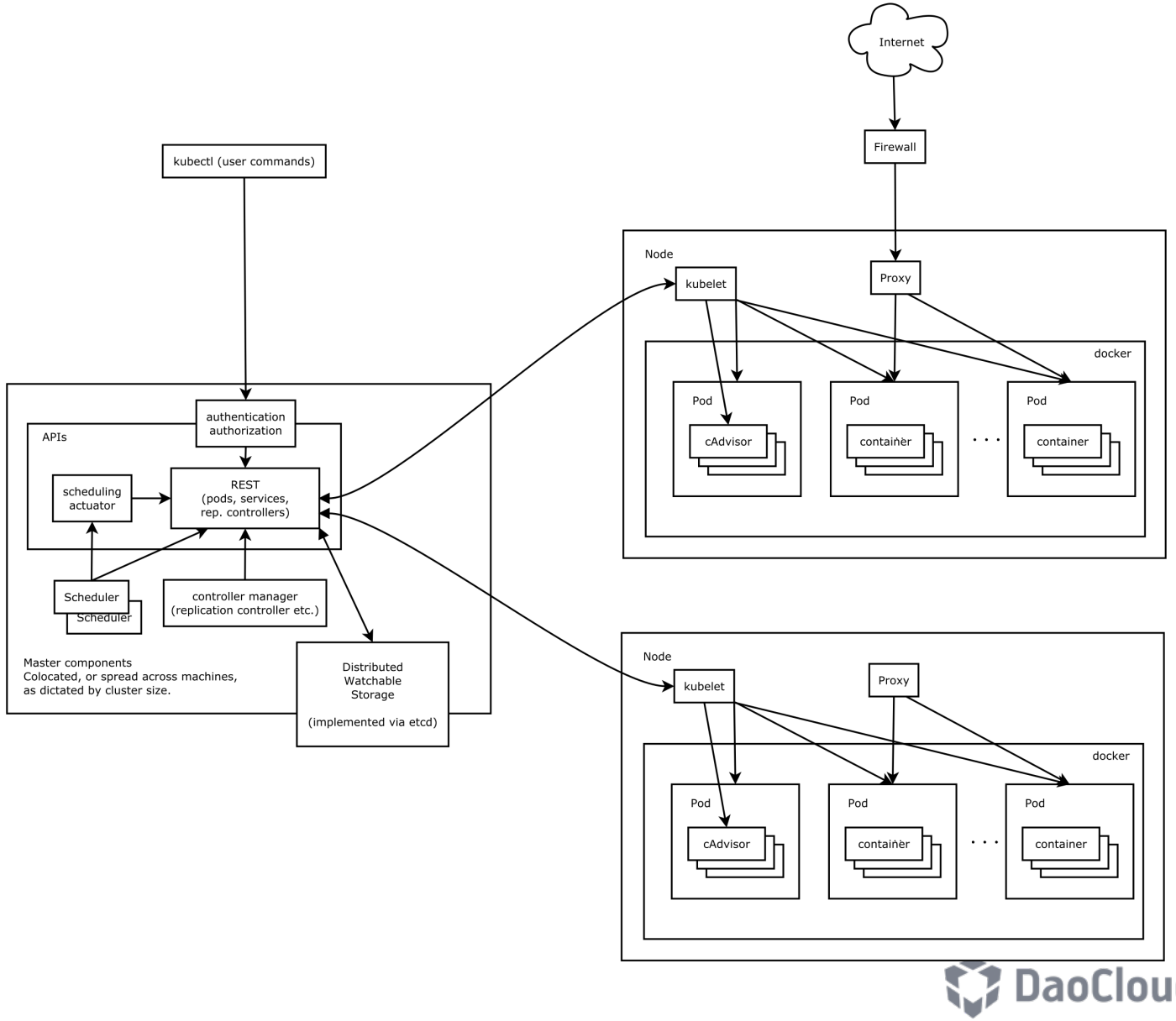
Kubernetes介绍

- Kubernetes项目是Google 2014年开源的容器集群管理系统，由Google多年大规模容器管理技术Borg演化而来并赠给云原生计算基金会，它的背后是拥有在谷歌运行生产工作负载超过十年经验的谷歌内部容器集群管理平台Borg和Omega。
- Kubernetes（CNCF），其主要功能包括
  - 基于容器的应用部署、维护和滚动升级
  - 负载均衡和服务发现
  - 跨机器和跨地区的集群调度
  - 自动伸缩
  - 无状态服务和有状态服务
  - 广泛的存储支持
  - 插件机制保证扩展性
- Kubernetes发展非常迅速，已经成为容器编排领域的领导者

DaoCloud

Kubernetes架构

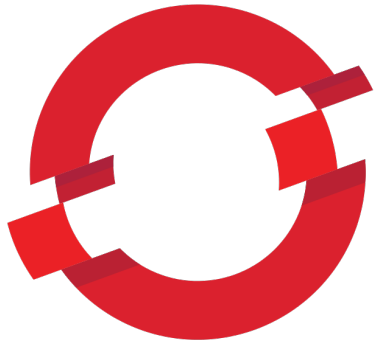
- Kubernetes借鉴了Borg的设计理念，其整体架构跟Borg非常相似
- Kubernetes主要由多个核心组件组成
- kube-apiserver提供了资源操作的唯一入口
- kube-controller-manager负责维护集群的状态
- kube-scheduler负责资源的调度
- kubelet负责维持容器的生命周期
- kube-proxy负责提供集群内部的服务发现和负载均衡
- etcd保存了整个集群的状态



DaoCloud

Kubernetes社区生态

- Kubernetes代码被托管在GitHub上，有超过28000个关注者和1400个贡献者，每周有上百个PR被提交和合并
- Kubernetes发展迅速，现已更新至1.12版本
- Google、Microsoft、Amazon等科技界巨头投入越来越多的资源在Kubernetes相关产品的研发上



OPENSHIFT

CoreOS

amazon

Microsoft

Google

ubuntu

DaoCloud

DaoCloud

谢谢

info@daocloud.io | www.daocloud.io