

Cloud Engineering Principles and Practice Shengqing Pei

by Shengqing Pei

FILE	25511_SHENGQING_PEI_CLOUD_ENGINEERING_PRINCIPLES_AND_PRACTICE_SHENGQING_PEI_343871_33226042.DOCX (1.9M)		
TIME SUBMITTED	09-MAY-2021 12:55PM (UTC+0100)	WORD COUNT	5412
SUBMISSION ID	152325846	CHARACTER COUNT	29554

Cloud Engineering Principles and Practice

Shengqing Pei

Abstract

Cloud computing is characterized by convenient access to the shared resource pool on demand, anytime and anywhere. It can access the computing resources in the unified shared resource pool without changing the existing hardware infrastructure of the enterprise. A complete cloud computing environment includes three parts: cloud, network and terminal, namely cloud and management. The cloud is responsible for calling the hardware resources in the platform to complete the tasks submitted by users; The terminal is a device used by users to input and receive output results; Computer network realizes the connectivity between users and the cloud, and is responsible for scheduling various computing resources in the distributed cluster. Cloud computing concept provides three service modes, corresponding to the software layer, operating system layer and hardware layer. The three service modes are infrastructure-as-a-service that provides hardware resources directly, platform-as-a-service that provides platform function services, and software as a service that provides application software as a service.

As the most important technical feature of cloud computing, virtualization is an important technology to help cloud platform realize resource pooling and provide resource sharing, resource management and other functions on demand, so as to improve the utilization rate of idle resources. Virtualization technology makes up for the limitation that x86 servers can only run one operating system. At the same time, the utilization efficiency of computing resources of each server is less than 30%, and precious computing resources are wasted. If a single server fails, the normal operation of the enterprise will be affected immeasurably in a short time. The emergence of virtualization technology just makes up for the vulnerability of the server. Virtualization is generally divided into two types, which are based on the host computer with an operating system. This kind of virtualization is called sojourn virtualization; If the host does not have an independent operating system, it is called bare metal virtualization. Hypervisor establishes and runs several different sub

operating systems on the host host host operating system. If users need to run multiple applications which are isolated from each other, they need to use virtual machine management system to establish multiple virtual machines

The host computer runs several sub operating systems separately, which will cause huge CPU and memory consumption to the host computer. At the same time,

The operating system of each virtual machine needs to install many dependencies to ensure the normal operation of the software. Redundant and repetitive operating system files will occupy a lot of disk space of the host computer. Docker container technology promotes the development of PAS platform

In the vigorous development of management and control platform business, the number of virtual machines in the main data center has doubled, and the technology scheme based on hardware virtualization cannot keep pace with the rapid growth of business in flexibility. How to change the relative traditional production environment deployment mode and application operation and maintenance mode through modern technology, improve the utilization rate of hardware equipment resources, software deployment efficiency and application maintenance mode are the problems that the main data center needs to solve. Based on docker container technology, this paper adopts kubernetes distributed architecture and relies on the main data center to virtualize computing, storage and network resources, and then ensures the efficient and sustainable operation of the control platform through the kubernetes high availability infrastructure. In order to meet the normal operation of the control platform and background service, the container automatic deployment, health status monitoring and monitoring alarm service are set up.

1. Technical principle of dockers

As a lightweight container engine, docker technology is open source and isolated. Using this technology, the independent virtual network can be realized in the form of container. The network space is mainly composed of processor, memory and hard disk,

At this time, users can use the packaged migration method by using the docker server image. The application program is stored in the system image to form a new and advanced capacity. By installing and deploying the container to any machine with docker server, it can realize the effective sharing and utilization of network protocol, and improve the interaction efficiency of network protocol

It lays a solid foundation for efficiency and effect. In recent years, with the rapid development of micro service application system. In recent years, docker technology has been widely used in the deployment of micro service application system to ensure the quality of micro service. Service application system can run reliably, stably and safely.

The bottom core of docker container is to use LxC (Linux containers) to realize the basic function of container

Function. LxC is very lightweight. Its function is to disguise the virtual machine process as a host process

On the other hand, it can quickly create and destroy tasks. Its core technology is composed of two parts: namespace and groups.

Docker isolates docker processes by using a namespace, which ensures that each docker task has its own independent running space and does not interfere with each other. Its namespace includes PID, net, MNT, etc. by using an independent namespace, it ensures that docker and docker will not interfere with each other. The namespace is like an isolation layer, which ensures that every docker application runs in its own independent namespace and cannot access other docker applications

Docker achieves resource allocation and restriction through group control. Group control can dynamically allocate CPU, memory and other resources for each docker task, so that many docker tasks can coexist friendly on the host. Group control can

monitor the resource consumption status of each docker task, and ensure the fair allocation of resources. In addition, group control also has the functions of priority control and process group control, which can deal with the situation of multi docker task preempting resources and docker task deadlock.

2.The benefits of containerisation.

Because the docker container has a different operating principle from the virtual machine, compared with the traditional virtual machine,

Docker has the following advantages

Standardized application Publishing: the docker task is published using the docker image. This publishing scheme enables users to complete the publishing of the docker task with just a few lines of commands after successfully building the docker image. 2. The services saved in the docker image can be run, saving the tedious steps of virtual machine installation and configuration.

Save time and cost: docker container applications can directly access the host's operating system kernel, unlike traditional virtual machines, which virtualize the operating system kernel. Therefore, the storage space occupied by docker tasks is much less than that of virtual machines, and the disk space occupied by deploying virtual machines often needs hundreds or even gigabytes, However, it only takes a few megabytes to deploy docker. At the same time, tens of thousands of docker tasks can be easily deployed on the host in tens of seconds, no matter the deployment time or quantity is far beyond the virtual machine.

Excellent isolation scheme: using the docker technology of LxC can complete the access isolation between tasks very well. The collapse of any docker process will not affect the use and allocation of CPU, memory, hard disk and other resources of the host or other docker processes. Excellent isolation makes docker technology have high fault tolerance in practical applications

Simple management scheme: docker tasks can be viewed at a glance through the API interface provided by docker. 2. When users encounter server failure, compared with tasks deployed through virtual machine and physical machine, tasks deployed through docker can clearly observe which applications are in failure state, so that users can easily and quickly repair the task process.

In this paper, the cloud platform, distributed database and database management

system will be deployed with docker technology. Compared with the virtual machine, it not only has faster startup speed and takes less resources, but also facilitates the migration and backup of data and applications and the management and isolation of multi-user tasks.

3.How to construct docker file to build image

Dockerfile example

```
# Base images 基础镜像
FROM centos

#MAINTAINER 维护者信息
MAINTAINER lorenwe

#ENV 设置环境变量
ENV PATH /usr/local/nginx/sbin:$PATH

#ADD 文件放在当前目录下，拷过去会自动解压
ADD nginx-1.13.7.tar.gz /tmp/

#RUN 执行以下命令
RUN rpm --import /etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7 \
    && yum update -y \
    && yum install -y vim less wget curl gcc automake autoconf libtool make gcc-c++ z1 \
    && yum clean all \
    && rm -rf /usr/local/src/*
RUN useradd -s /sbin/nologin -M www

#WORKDIR 相当于cd
WORKDIR /tmp/nginx-1.13.7

RUN ./configure --prefix=/usr/local/nginx --user=www --group=www --with-http_ssl_modul

RUN cd / && rm -rf /tmp/
```

```

RUN ./configure --prefix=/usr/local/nginx --user=www --group=www --with-http_ssl_module

RUN cd / && rm -rf /tmp/

COPY nginx.conf /usr/local/nginx/conf/

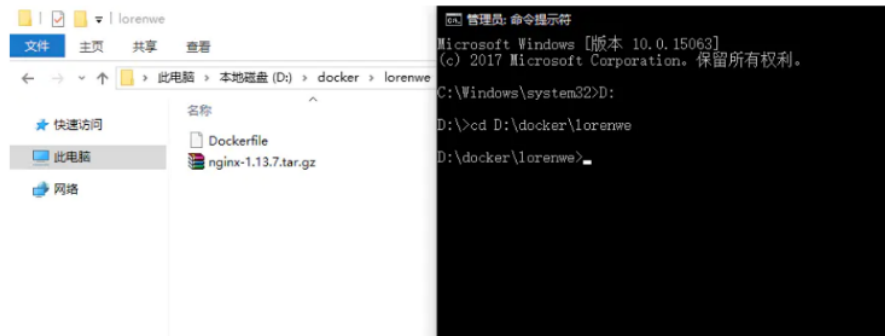
#EXPOSE 映射端口
EXPOSE 80 443

#ENTRYPOINT 运行以下命令
ENTRYPOINT ["nginx"]

#CMD 运行以下命令
CMD ["-h"]复制代码

```

The above code example is a dockerfile file that I think is very representative. It does not involve much content, but it basically uses all the dockerfile instructions and also contains some details. In order to achieve the effect of the example, it is not the most concise dockerfile. Create a folder and put the above dockerfile in the file, Then go to the nginx official website and put the nginx source code package in the folder. .



dockerfile catalog

The command for dockerfile to build the image is very simple. In my example, my command is "docker build -t lorenwe / CentOS"_"Nginx." note that the following points cannot be omitted. It means that you can look for dockerfile from the current directory to build the image.

```
D:\docker\lorenwe>docker build -t lorenwe/centos_nginx .
Sending build context to Docker daemon 995.8kB
Step 1/13 : FROM centos
----> d123f4e55e12
Step 2/13 : MAINTAINER lorenwe
----> Running in e5c7274f50e8
----> 606f7222e69a
Removing intermediate container e5c7274f50e8
Step 3/13 : ENV PATH /usr/local/nginx/sbin:$PATH
----> Running in 23716b428809
----> 5d8ee1b5a899
....
Successfully built eaae6b40b151
Successfully tagged lorenwe/centos_nginx:latest复制代码
```

```
D:\docker\lorenwe>docker images
REPOSITORY          TAG          IMAGE ID        CREATED         SIZE
lorenwe/centos_nginx latest       eaae6b40b151   7 minutes ago   427MB
lorenwe/centos_net_tools latest       35f8073cede1   6 days ago      277MB
centos               latest       d123f4e55e12   3 weeks ago     197MB
d4w/nsenter          latest       9e4f13a0901e   14 months ago   83.8kB

D:\docker\lorenwe>docker run -itd --name nginx1 lorenwe/centos_nginx
15d4f108dab7c2f276209eb501cac0d3be828e1e81bae22d3fd97c617439eb

D:\docker\lorenwe>docker ps
CONTAINER ID    IMAGE          COMMAND          CREATED    STATUS    PORTS    NAMES

D:\docker\lorenwe>docker ps -a
CONTAINER ID    IMAGE          COMMAND          CREATED    STATUS    PORTS    NAMES
15d4f108dab7    lorenwe/centos_nginx "nginx -h"          7 minutes ago    Up 7 minutes    80/tcp    nginx1

D:\docker\lorenwe>docker run -itd --name nginx2 lorenwe/centos_nginx -c /usr/local/nginx
b6b0e962ca3056d67c24145b08975ffddb9cc050fce5f09f65310fb323ffc1c3

D:\docker\lorenwe>docker ps
CONTAINER ID    IMAGE          COMMAND          CREATED    STATUS    PORTS    NAMES
b6b0e962ca30    lorenwe/centos_nginx "nginx -c /usr/loc..." 7 minutes ago    Up 7 minutes    80/tcp    nginx2

D:\docker\lorenwe>docker run -itd -p 8080:80 --name nginx3 lorenwe/centos_nginx -c /usr/local/nginx
2f6997745641e3e3edbbfe5213e6235cab3b5a929f116a2c132df504156090c6
```

```
D:\docker\lorenwe>docker ps
CONTAINER ID    IMAGE          COMMAND          CREATED    STATUS    PORTS    NAMES
2f6997745641    lorenwe/centos_nginx "nginx -c /usr/loc..." 7 minutes ago    Up 7 minutes    0.0.0.0:8080->80/tcp    nginx3
b6b0e962ca30    lorenwe/centos_nginx "nginx -c /usr/loc..." 7 minutes ago    Up 7 minutes    80/tcp    nginx2

D:\docker\lorenwe>docker stop nginx2
nginx2复制代码
```

< >

Among them, "docker run - ITD - P 8080:80 -- name nginx3 lorenwe / CentOS" The - P 8080:80 in "nginx - C / usr / local / nginx / conf / nginx. Conf" means to map the host's 8080 port to the container's 80 port, because we exposed the 80 port in the docker file before. Now we can open the browser in the host and visit 127.0.0.1:8080 to see the welcome page of nginx.

4. Push images to dockerhub

After the image is built, we can also upload it to docker hub, so that other people can use the image. For example, we can share the image within the organization, or completely expose the image.

Now let's try how to push the root image

```
1 $ sudo docker push static_web
2 2013/07/01 18:34:47 Impossible to push a "root" repository.
3 Please rename your repository in <user>/<repo> (ex: jamtur01/
4 static_web)
```

We try to push the image to the remote warehouse static_ Web, but docker thinks it's a root repository. The root repository is managed by the docker team, so our push request will be rejected. Let's try another way to push the docker image

```
1 $ sudo docker push jamtur01/static_web
2 The push refers to a repository [jamtur01/static_web] (len: 1)
3 Processing checksums
4 Sending image list
5 Pushing repository jamtur01/static_web to registry-1.docker.io (1 tags)
6 ...
```

This time, we used a program called jamtur01 / static_ The user warehouse of Web successfully pushes the image to docker hub. We will use our own user ID, which we created earlier, and choose a legal image name (such as youruser / yourimage).

We can see our uploaded image in docker hub,

5. Create docker volume

5.1 Docker create container

A container is a running instance of an image, It's a lightweight environment based on mirror operation, It's an application or a group of applications. How to create a container? Pass in the image name on which the container is based, Docker will look for the image from the local warehouse, If not in the local warehouse, Will be automatically pulled from the remote warehouse. docker pull nginx Pull the image of the latest version of nginx

```
[root@localhost ~]# docker pull nginx
Using default tag: latest
latest: Pulling from library/nginx
f2aa67a397c4: Pull complete
3c091c23e29d: Pull complete
4a99993b8636: Pull complete
Digest: sha256:0fb320e2a1b1620b4905facb3447e3d84ad36da0b2c8aa8fe3a5a81d1187b884
Status: Downloaded newer image for nginx:latest
```

docker images

View image

```
[root@localhost ~]# docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
nginx	latest	ae513a47849c	11 days ago	109MB
registry	latest	d1fd7d86a825	4 months ago	33.3MB

docker create nginx

Create nginx container

```
[root@localhost ~]# docker create nginx
8cf85f83522c5569517164adb0f369e3880909d50f30e3b8dda7b0368589d5d3
```

docker run nginx

Start the nginx container

At this time, xshell is in the foreground interaction mode,

Press Ctrl + C to exit,

Then the container stops running

docker run -i -t nginx /bin/bash

Run the nginx container and assign it a pseudo terminal

```
[root@localhost ~]# docker run -i -t nginx /bin/bash
root@e9dcf58fa575:/#
```

uname -a .Check system version .You can see the container ID

```
root@89fa54161b87:/# uname -a
Linux 89fa54161b87 3.10.0-693.el7.x86_64 #1 SMP Tue Aug 22 21:09:27 UTC 2017 x86_64 GNU/Linux
```

ls -a.Show all current directories

```
root@e9dcf58fa575:/# ls -a
.  .dockerenv  boot  etc  lib  media  opt  root  sbin  sys  usr
.. bin  dev  home  lib64  mnt  proc  run  srv  tmp  var
```

exit

sign out

The container also stops running

docker run -d nginx

Let the nginx container run in the background

```
[root@localhost ~]# docker run -d nginx
805e7cc5f10f4fc9cbc2ec9344203769bbf6e9dbeab5847471f661db816549fa
```

docker ps

View running containers

```
[root@localhost ~]# docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS
PORTS         NAMES
805e7cc5f10f   nginx     "nginx -g 'daemon of..." About a minute ago Up About a minute
80/tcp        friendly_hamilton
```

docker stop 805

Stop nginx container

(container ID can be fuzzy matched)

```
[root@localhost ~]# docker stop 805
805
```

5.2 Docker create data volume

When the program is running in a container,

When you need to read and write files with programs in other containers or programs outside the container,

Data exchange is needed;

File system inside the container,

It's a temporary floor,
When the container stops running and is removed,
This temporary layer will be discarded together;
To obtain files from outside and persist storage,
Docker provides data volumes;
A data volume is a directory that is mounted externally on a file system within a container,
Independent of the container's life cycle,
When the container stops running and is removed,
Data volumes are not affected,
It will still exist in docker host!
docker run -d --name test_1 -v /yjl nginx

Run test_1 container and create a data volume with path /yjl in the container

```
[root@localhost ~]# docker run -d --name test_1 -v /yjl nginx  
f660705ce5c31f1ff6e511215e0f8b8212b53ea1166daabb01a545438eb42532
```

docker run -d --name test_2 -v /root/yjl:/data_2 nginx

```
[root@localhost ~]# docker run -d --name test_2 -v /root/yjl:/data_2 nginx  
f5c8cee12d606ed0c84b5257d43ff75bc40842c17c8ee9efce9f312de882fcd2
```

Create an alias named data using the docker volume command_ Data volume for 3.

```
[root@localhost ~]# docker volume create --name data_3  
data_3
```

docker volume inspect data_ three

View data volume information

```
[root@localhost ~]# docker volume inspect data_3  
[  
  {  
    "CreatedAt": "2018-05-13T19:02:23+08:00",  
    "Driver": "local",  
    "Labels": {},  
    "Mountpoint": "/var/lib/docker/volumes/data_3/_data",  
    "Name": "data_3",  
    "Options": {},  
    "Scope": "local"  
  }  
]
```

Note that this directory is the data volume of the host,

Instead of the container's data volume.

```
docker run -d --name test_3 -v data_3:/data_3 nginx
```

Run test_3 containers, mounting data_3 data volume to container /data_3 directory as data volume

```
[root@localhost ~]# docker run -d --name test_3 -v data_3:/data_3 nginx
1c2d55286579a56edee0155dab1d6b3a25e9f1980b9cd6801e65a46db8de60aa
```

docker volume list

View data volumes in host

```
[root@localhost ~]# docker volume list
DRIVER      VOLUME NAME
local       15c073cf85b910de2c9e632c9ed8401426098cb68135b64fef0830fa9dbe622c
local       8f61116434604a7c0d9a43e89c828938f12ac69d55c6c303c0aa3a58fdc9b5a6
local       91e5ae10ba0b88390e084aea393aa3f517a13006e672db3f761e009b54cdf3f0
local       data_3
```

Docker volume RM < parameters: volume name > of the data volume

Delete data volume

Note that if the data volume is not aliased,

The name is 64 bit full name,

Fuzzy matching is not supported,

So it is best to take an alias when creating a data volume.

Docker rm-v < parameters: container name >

Delete data volumes with container,

Only for data volumes created with the creation of containers.

5.3 Docker create data volume container

```
docker create --name test_4 -v /data_4 nginx
```

Create a test_4 data volume container, in the container's /data_4 directory mount data volume

```
[root@localhost ~]# docker create --name test_4 -v /data_4 nginx
fdea17972bdd76cf231d4bfff3ce4e390cd1a259de58181ad22faacab2a86ef63
```

When using a data volume container,

There is no need to ensure that the data volume container is in operation,

So use the docker create command.

```
docker inspect test_4
```

View test_4 data volume container information

```

    "Mounts": [
      {
        "Type": "volume",
        "Name": "58a87ba89aacb141ebdb03ff617f232ef9a25b4ef8a21c5d06d76a9b9346a375",
        "Source": "/var/lib/docker/volumes/58a87ba89aacb141ebdb03ff617f232ef9a25b4ef8a21c5d06d76a9b9346a375/_data",
        "Destination": "/data_4",
        "Driver": "local",
        "Mode": "",
        "RW": true,
        "Propagation": ""
      }
    ],

```

docker run -d --name test_5 --volumes-from test_4 nginx

docker run -d --name test_6 --volumes-from test_4 nginx

Create test_5 and test_6 two containers,

And mount test_4 data volume container

```

[root@localhost ~]# docker run -d --name test_5 --volumes-from test_4 nginx
93862b6711ecedbab7cd1005e86db4e7f9b1b56d43578ea3c0bf126cd76e1929
[root@localhost ~]# docker run -d --name test_6 --volumes-from test_4 nginx
c3a47aeaad6f7871074c4e0f3dbeaae02e028e49645c272aed8c53a8b73f6162

```

Any one of the three containers is in /data_4 directory,

You can see all the other containers.

You can use the --volumes-from parameter more than once,

Realize the simultaneous use of multiple different data volume containers,

The data volumes in these data volume containers,

Will be mounted in a new container.

docker run --name test_7 --volumes-from test_4 -v \$(pwd):/folder nginx tar cvfP /folder/folder.tar /data_four.

```

[root@localhost ~]# docker run --name test_7 --volumes-from test_4 -v $(pwd):/folder nginx tar cvfP /folder/folder.tar /data_4
/data_4/

```

export

Backup command:

1. Using nginx image to create test_7 container;
2. Mount test_4 data in data volume container_4 data volume;
3. Mount the current directory of the host to test_7 / folder directory of the container;
4. test_ After the container is started, the /data_4. Compress the data volume to folder.tar

||

View the current directory of host

```
[root@localhost ~]# ll
总用量 16
-rw-----. 1 root root 1259 4月 21 20:38 anaconda-ks.cfg
-rw-r--r--. 1 root root 10240 5月 14 00:11 folder.tar
drwxr-xr-x. 2 root root 6 5月 13 17:47 yjl
```

6 Creating docker network

```
1 $ docker run -itd --name=container1 busybox
2
3 18c062ef45ac0c026ee48a83afa39d25635ee5f02b58de4abc8f467bcaa28731
4
5 $ docker run -itd --name=container2 busybox
6
7 498eaaaf328e1018042c04b2de04036fc04719a6e39a097a4f4866043a2c2152
```

```
1 $ docker network create -d bridge --subnet 172.25.0.0/16 isolated_nw
2
3 06a62f1c73c4e3107c0f555b7a5f163309827bfbbf999840166065a8f35455a8
```

```
1 $ docker network connect isolated_nw container2
2
3 $ docker network inspect isolated_nw
4
5 [
6   {
7     "Name": "isolated_nw",
8     "Id": "06a62f1c73c4e3107c0f555b7a5f163309827bfbbf999840166065a8f35455a8",
9     "Scope": "local",
10    "Driver": "bridge",
11    "IPAM": {
12      "Driver": "default",
13      "Config": [
14        {
15          "Subnet": "172.25.0.0/16",
16          "Gateway": "172.25.0.1/16"
17        }
18      ]
19    },
```

```
1 $ docker run --network=isolated_nw --ip=172.25.3.3 -itd --name=container3 busybox
2
3 467a7863c3f0277ef8e661b38427737f28099b61fa55622d6c30fb288d88c551
```

```
1 $ docker inspect --format='{{.NetworkSettings.IPAddress}}' container3
2
3 {"isolated_nw":
4   {"IPAMConfig":
5     {
6       "IPv4Address": "172.25.3.3"},
7       "NetworkID": "1196a4c5af43a21ae38ef34515b6af19236a3fc48122cf585e3f3054d509679b",
8       "EndpointID": "dffc7ec2915af58cc827d995e6ebdc897342be0420123277103c40ae35579103",
9       "Gateway": "172.25.0.1",
10      "IPAddress": "172.25.3.3",
11      "IPPrefixLen": 16,
12      "IPv6Gateway": "",
13      "GlobalIPv6Address": "",
14      "GlobalIPv6PrefixLen": 0,
15      "MacAddress": "02:42:ac:19:03:03"}
16    }
17  }
18 }
```

```
1 $ docker inspect --format='{{.NetworkSettings.IPAddress}}' container2 | python -m json.tool
2
3 {
4   "bridge": {
5     "NetworkID": "7ea29fc1412292a2d7bba362f9253545fecdfa8ce9a6e37dd10ba8bee7129812",
6     "EndpointID": "0099f9efb5a3727f6a554f176b1e96fca34cae773da68b3b6a26d046c12cb365",
7     "Gateway": "172.17.0.1",
8     "GlobalIPv6Address": "",
9     "GlobalIPv6PrefixLen": 0,
10    "IPAMConfig": null,
11    "IPAddress": "172.17.0.3",
12    "IPPrefixLen": 16,
13    "IPv6Gateway": "",
14    "MacAddress": "02:42:ac:11:00:03"
15  },
16  "isolated_nw": {
17    "NetworkID": "1196a4c5af43a21ae38ef34515b6af19236a3fc48122cf585e3f3054d509679b",
18    "EndpointID": "11cedac1810e864d6b1589d92da12af66203879ab89f4ccd8c8fdaa9b1c48b1d",
19    "Gateway": "172.25.0.1",
20    "GlobalIPv6Address": "",
21    "GlobalIPv6PrefixLen": 0,
22    "IPAMConfig": null,
23    "IPAddress": "172.25.0.2",
24    "IPPrefixLen": 16,
25    "IPv6Gateway": "",
26    "MacAddress": "02:42:ac:19:00:02"
27  }
28 }
```

```

1 $ sudo ifconfig -a
2
3 eth0      Link encap:Ethernet  HWaddr 02:42:AC:11:00:03
4           inet addr:172.17.0.3  Bcast:0.0.0.0  Mask:255.255.0.0
5           inet6 addr: fe80::42:acff:fe11:3/64 Scope:Link
6           UP BROADCAST RUNNING MULTICAST  MTU:9001  Metric:1
7           RX packets:8 errors:0 dropped:0 overruns:0 frame:0
8           TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
9           collisions:0 txqueuelen:0
10          RX bytes:648 (648.0 B)  TX bytes:648 (648.0 B)
11
12 eth1      Link encap:Ethernet  HWaddr 02:42:AC:15:00:02
13           inet addr:172.25.0.2  Bcast:0.0.0.0  Mask:255.255.0.0
14           inet6 addr: fe80::42:acff:fe19:2/64 Scope:Link
15           UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
16           RX packets:8 errors:0 dropped:0 overruns:0 frame:0
17           TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
18           collisions:0 txqueuelen:0
19          RX bytes:648 (648.0 B)  TX bytes:648 (648.0 B)
20
21 lo        Link encap:Local Loopback
22           inet addr:127.0.0.1  Mask:255.0.0.0
23           inet6 addr: ::1/128 Scope:Host
24           UP LOOPBACK RUNNING  MTU:65536  Metric:1
25           RX packets:0 errors:0 dropped:0 overruns:0 frame:0
26           TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
27          collisions:0 txqueuelen:0

```

```

1 / # ping -w 4 container1
2 ping: bad address 'container1'

```

```

1 / # ping -w 4 172.17.0.2
2 PING 172.17.0.2 (172.17.0.2): 56 data bytes
3 64 bytes from 172.17.0.2: seq=0 ttl=64 time=0.095 ms
4 64 bytes from 172.17.0.2: seq=1 ttl=64 time=0.075 ms
5 64 bytes from 172.17.0.2: seq=2 ttl=64 time=0.072 ms
6 64 bytes from 172.17.0.2: seq=3 ttl=64 time=0.101 ms
7
8 --- 172.17.0.2 ping statistics ---
9 4 packets transmitted, 4 packets received, 0% packet loss
10 round-trip min/avg/max = 0.072/0.085/0.101 ms

```

6. Making a simple web server to display HTML pages with docker

On the basis of mastering the installation of docker, the basic use of docker warehouse, the basic operation and mutual transformation of docker image and container, we can try to build a web server through docker, which is convenient for distributed deployment or rapid migration of web server.

With the development of industry 4.0 and the large-scale introduction of intelligent equipment in industrial enterprises in recent years, enterprises put forward higher requirements for the function and performance of the equipment management system in the factory. Traditional enterprise equipment management system relies on traditional workers to input equipment information manually, equipment exception handling is not timely, equipment maintenance intelligent management is not high, equipment management system operation and maintenance deployment means are too backward and so on. The establishment of enterprise private cloud platform has become the focus of the development of industrial information system. Based on docker container technology, this paper constructs a PAAS cloud platform for device management, which provides a new way of micro service, lightweight, efficient and agile application operation and maintenance deployment. Combined with the business characteristics and functional requirements of enterprise equipment management, the cloud platform of equipment management is constructed from the practical application to realize the unified monitoring and scheduling of system resources, faster development and deployment, which is convenient for the iterative upgrade of system version, saves the cost of R & D and operation and maintenance, brings huge economic benefits for enterprises, and improves the competitiveness of enterprises. In order to solve the problem of low deployment efficiency of existing device management system, this paper proposes a cloud platform of device management system based on docker container technology and kubernetes technology, and analyzes the advantages of 20 docker container technology. Based on docker to establish PAAS cloud service, there are many disadvantages in traditional device management system deployment. Combined with the future development direction of industrial intelligent system cloud platform, this paper designs micro service device management application deployment architecture, and analyzes the advantages of container technology in device management system cloud platform deployment application. Through the specific implementation of image code warehouse, image warehouse, deployment instance and mobile application deployment scheme, the feasibility of the design scheme of device management cloud platform is verified, the deployment process of device

management application is simplified, and the efficiency of system development and deployment is improved. Finally, through the functional test of each module of the platform, the results show that the platform can better meet the application needs of industrial enterprises for equipment management business, and verify the effectiveness of related technologies. The cloud platform provides resource monitoring, resource scheduling, convenient development, continuous delivery, operation and maintenance deployment, version update, resource management and service As a good guarantee for enterprise device management application, the functions of automatically packaging and generating mobile app are also provided.

Install nginx

apt-get install nginx

Modify nginx configuration file#

```
1 # vi /etc/rc.local
2
3 ####!/bin/sh -e
4 #
5 # rc.local
6 #
7 # This script is executed at the end of each multiuser runlevel.
8 # Make sure that the script will "exit 0" on success or any other
9 # value on error.
10 #
11 # In order to enable or disable this script just change the execution
12 # bits.
13 #
14 # By default this script does nothing.
15
16 service ssh start
17 service nginx start
18
19 /bin/bash
20
21 exit 0
```

Create an image to start the container again

Creating docker image file by commit operation

```

1 kevin@apple:~/docker$ docker ps -a
2 CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS
3 568e5204fff3        ubuntu             "/bin/sh -c 'while..." 40 hours ago        Exited (137) 38 hours ago
4 00f561d97811        ubuntu             "/bin/echo hello w..." 40 hours ago        Exited (0) 40 hours ago
5 93a1b9d39683        ubuntu             "bash"              40 hours ago        Exited (0) 5 seconds ago
6 abdc084f9821        hello-world        "/hello"             41 hours ago        Exited (0) 18 hours ago
7 kevin@apple:~/docker$ docker commit 93a1b9d39683 learn/nginx:v2
8 sha256:ab92edd21696770f1eb37e9229b6834094a8d3381e5b4e9edc620b7927004bb4
9 kevin@apple:~/docker$ docker images
10 REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
11 learn/nginx          v2                 ab92edd21696        5 seconds ago      370MB
12 learn/visual_init   v1                 56a4eab7dc5b        37 hours ago       321MB
13 registry.cn-beijing.aliyuncs.com/zhangsp/ai visual_init         56a4eab7dc5b        37 hours ago       321MB
14 ubuntu              latest              14f60031763d        5 days ago         120MB
15 hello-world          latest              1815c82652c0        5 weeks ago        1.84kB

```

Start new container

Use the newly created mirror learn/ nginx:v2 , start the new container

```

1 # docker run -it --name nginx_test -h docker-nginx -p 8001:81 -p 8000:80 -v /home/kevin/docker/nginx_web:/home/visu

```

Parameter introduction of start container

-It, start interactively

- name nginx_ Test, which specifies that the name of the new container is nginx_ test
- H docker nginx, which specifies that the host name of the new container is specify the port mapping between the host and the docker container. 8001 of the host corresponds to 81 of the docker container, and 8000 of the host corresponds to 80 of the docker container
- /, specifies the file mapping between the host and the docker container, and / home / Kevin / docker / nginx of the host_ Web / corresponding to / home / visual / nginx of docker container_ web/
- learn/ nginx:v2 , which specifies that the image corresponding to the startup container is learn/ nginx:v2 , which can be the image ID
- /Bin / SH / etc / rc.local, specifies that after the container is started, the shell script to be executed is / etc / rc.local

View docker container, container nginx_ Test is in the up state, indicating that the startup is normal。

```

1 kevin@apple:~/docker/nginx_web$ docker ps -a
2 CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
3 cbbbe7a5d47a        learn/nginx:v2     "/bin/sh /etc/rc.l..." 25 minutes ago     Up 24 minutes
4 966bd52b72da        ubuntu             "/bin/sh -c 'while..." 42 hours ago       Exited (137) 40 hours ago
5 00f561d97811        ubuntu             "/bin/echo hello w..." 42 hours ago       Exited (0) 42 hours ago
6 93a1b9d39683        ubuntu             "bash"              43 hours ago       Exited (0) About an hour ago
7 abdc084f9821        hello-world        "/hello"             43 hours ago       Exited (0) 20 hours ago

```

Test docker_ Is nginx normal

Test port 8001 through browser

Test 8001 port-1.png through browser

Test 8000 port through browser

Test 8000 port.png through browser

Modify the host's file and test port 8001

Modify the host's / home / Kevin / docker / nginx_ Web / index.html file

```

1 kevin@apple:~/docker/nginx_web$ vi index.html
2 <!DOCTYPE html>
3 <html>
4 <head>
5 <title>Welcome to nginx! I am in docker!</title>
6 <style>
7     body {
8         width: 35em;
9         margin: 0 auto;
10        font-family: Tahoma, Verdana, Arial, sans-serif;
11    }
12 </style>
13 </head>
14 <body>
15 <h1>Welcome to nginx! I am in Docker!</h1>
16 <p>If you see this page, the nginx web server is successfully installed and
17 working. Further configuration is required.</p>
18
19 <p>For online documentation and support please refer to
20 <a href="http://nginx.org/">nginx.org</a>.<br/>
21 Commercial support is available at
22 <a href="http://nginx.com/">nginx.com</a>.</p>
23
24 <p><em>Thank you for using nginx.</em></p>
25 </body>
26 </html>

```

Through the browser test 8001 port, found that "welcome to nginx! I am in docker!" , The description has been modified, and the function of using docker as web server is OK.

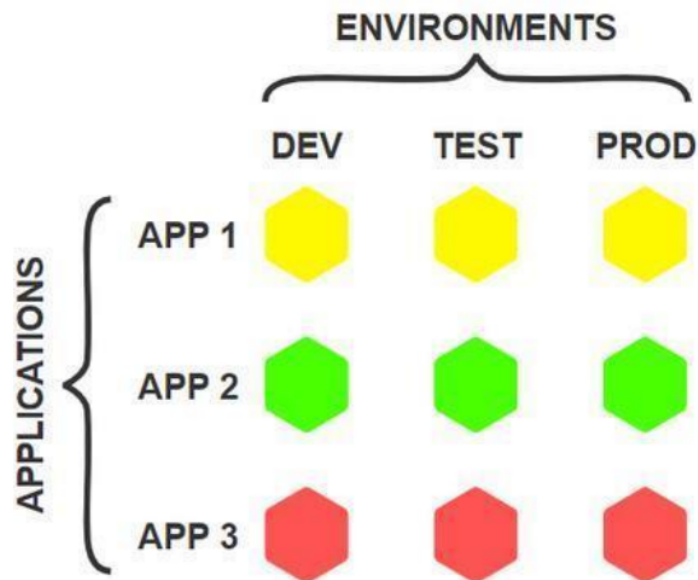
7.The use of kubernetes controllers and services in managing containers to generate web applications

Cluster analysis

As a result, the application and the environment form a different intersection. In the following example, there are three applications and three environments, resulting in nine application instances. Each application instance is an independent deployment unit that can run independently.

With the increasing of mobile services, traditional architecture is no longer suitable for new mobile services, and cloud computing has become the evolution direction of core network architecture. However, with the advent of 5g era, the high bandwidth, super density and low delay will become the mainstream characteristics of business. If the original cloud computing mode is still used, the speed, delay and stability can not be guaranteed at all. Edge computing is a computing mode which provides services to users on the edge of the network, which has the characteristics of low delay and high bandwidth. This mode can enable users to enjoy stable high-quality network experience. At the same time, the emergence of virtualization technology makes the traditional server resources more fully utilized. Platform and application virtualization is moving rapidly to the edge of the cloud, paving the way for the core multi-core processors of Internet of things, mobile and embedded application devices. This paper builds an edge computing platform by integrating docker container technology and kubernetes container management technology. In order to make up for the shortage of edge elastic algorithm, this paper improves the original elastic algorithm, and proposes a 20cpu memory crossover maximum algorithm. The algorithm takes into account the application requirements for memory and CPU, and makes decisions on the number of target copies by memory and CPU parameters. This paper verifies the algorithm on kubernetes platform, and compares it with the original algorithm. The results show that the algorithm can provide more accurate target copies while ensuring the stability of expansion / shrinking. Based on the improved elastic algorithm, this paper conducts network service experiments on edge computing platform. By comparing the indexes of edge side and cloud side, the stability delay advantage brought by edge side compared with cloud side is verified. In addition, the design and construction process of edge computing platform are described in detail. In order to make full use of the resources of cloud side and edge side of the edge computing platform, this paper also proposes a machine learning idea of training

and edge prediction on the cloud side. The feasibility of the idea is verified by the successful implementation of image recognition service on the edge side.

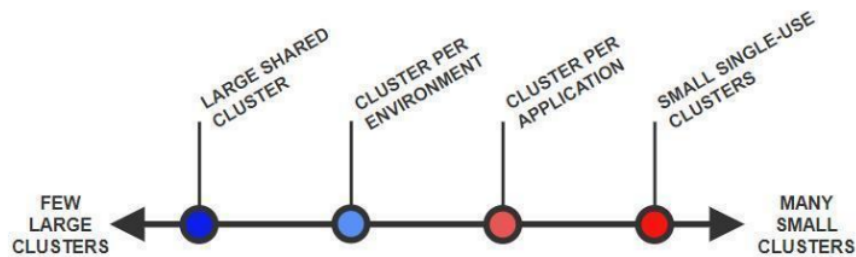


An application instance may consist of multiple components, such as front end, back end, database, etc. In a microservice application, an application instance will consist of all microservice components. Here are some options you can choose from:

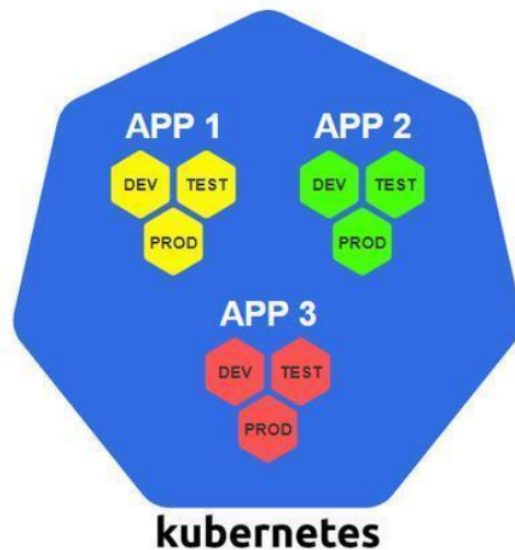
Virtualization technology makes up for the limitation that x86 servers can only run one operating system. At the same time, the utilization efficiency of computing resources of each server is less than 30%, and precious computing resources are wasted. If a single server fails, the normal operation of the enterprise will be affected immeasurably in a short time. The emergence of virtualization technology just makes up for the vulnerability of the server. Virtualization is generally divided into two types, which are based on the host computer with an operating system. This kind of virtualization is called sojourn virtualization; If the host does not have an independent operating system, it is called bare metal virtualization. Hypervisor establishes and runs several different sub operating systems on the host host host operating system. If users need to run multiple applications which are isolated from each other, they need to use virtual machine management system to establish multiple virtual machines

The first two methods are the scale limit from several large clusters to multiple small clusters, as

shown in the figure below:



In general, if a cluster contains a larger sum of nodes and pods, it can be defined as "larger". For example, a cluster with 10 nodes and 100 pods is larger than a cluster with 1 node and 10 pods.

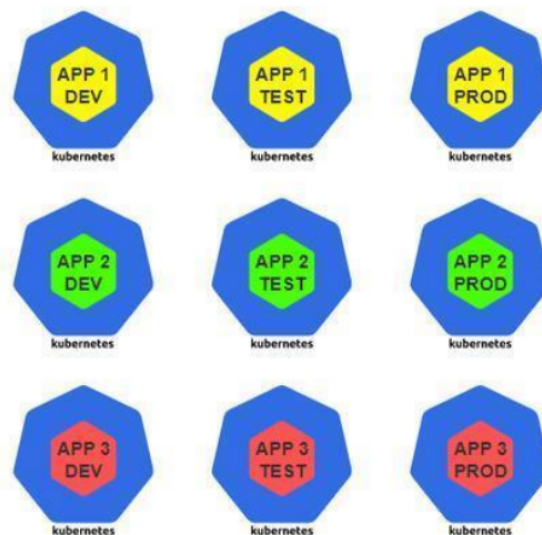


Using this approach, you can use a separate kubernetes cluster for each deployment unit (in this article, the deployment unit is an application instance, such as the development version of a single application). With this strategy, kubernetes is dedicated to each application instance.

Underlying facilities: relying on the main data center of Wuqing, the computing, storage and network resources are virtualized through virtualization technology, and

the kubernetes high availability infrastructure is used to ensure the efficient and stable operation of the management and control platform. Background services: the software background design adopts container architecture, and the background settings of the management and control platform are considered from the business function module, with clear responsibilities at each level, The overall system structure is clear, which is helpful for software development, integration, delivery and deployment. At the same time, in order to meet the normal operation of the background services of the management and control platform, the container automatic deployment, health monitoring and monitoring alarm services are set up.

Foreground application: the foreground application adopts B / s and CS hybrid architecture model to support passenger service, passenger transport management



Kubernetes has many excellent cluster management capabilities, including various security protection and access mechanisms, stable service registration and discovery, self owned intelligent load balancer, strong fault discovery and automatic recovery capability, service rolling upgrade and online expansion capability, and scalable automatic resource scheduling mechanism. At the same time, kubernetes also includes software development, continuous integration, continuous delivery, gray release, monitoring alarm and other management tools. Therefore, kubernetes is based on Docker is not only a distributed architecture, but also a powerful distributed system support platform

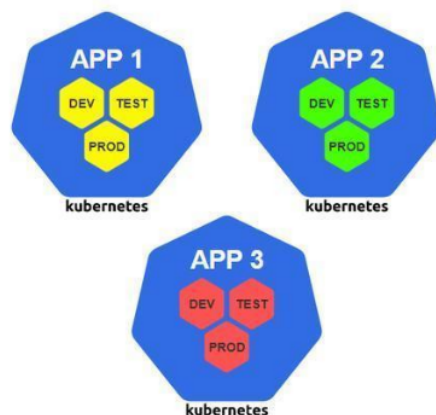
Kubernetes architecture based on docker container is the infrastructure for the management and control platform to achieve high availability. It will provide rapid deployment, elastic scaling, load balancing, high availability and other features for the application services of the management and control platform

Kubernetes, as the management platform of container application, monitors the running status of pod 24 hours a day and automatically detects the running status of host or container. If the host or container is not healthy, then

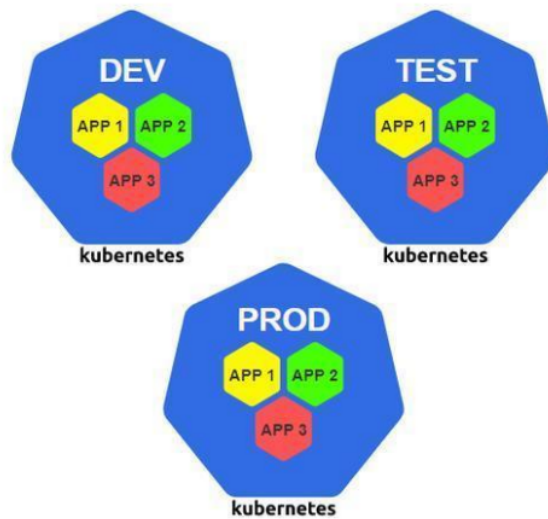
Kubernetes schedules the new pod to other normal nodes to achieve the high availability of the application layer. about

The high availability of kubernetes infrastructure is mainly reflected in the high availability of etcd2 data storage and kubernetes masrer components. The high availability infrastructure of kubernetes infrastructure is shown in Figure

Each component of kubernetes system uses TLS certificate for encrypted communication. This document uses cloudflare's tool set CFSL to generate certificate and secret key file. Certificate authority (CA) certificate is self signed, which is used to sign other TLS certificates created later. The following steps are operated on master01. The certificate only needs to be created once. When adding a new node to the cluster in the future, you only need to copy the certificate in the directory of / LETC / kubernetes / cert to the new node.



Kubernetes is not only a distributed architecture based on docker, but also a powerful support platform for distributed system. Kubernetes can quickly and efficiently meet the requirements of rapid and accurate deployment of the management and control platform, real-time scaling of applications, and limiting the amount of hardware to only the required resources. This will make the control platform easier to use, more efficient and more robust,



8.Using kubernetes, build the web service application of your choice using the appropriate controller and yaml services.

Kubernetes cluster is composed of master and node, on which several kubernetes

services are running.

(1) . master node:

The master node is the brain of kubernetes cluster, running the daemon services: Kube apiserver, Kube scheduler, Kube controller manager, etcd and pod network (such as calico)

API Server (kube-apiserver)

API server provides http / HTTPS restful API, namely kubernetes API. API server is the front-end interface of kubernetes cluster and various client tools (CLI or UI)

Kubernetes and other components can manage various resources of the cluster through it.

scheduler (kube-scheduler)

The scheduler is responsible for deciding which node to run pod on. When scheduling, the scheduler will fully consider the topology of the cluster, the current load of each node, and the application requirements for high availability, performance, and data affinity.

controller manager (kube-controller-manager)

The controller manager is responsible for managing all kinds of resources in the cluster to ensure that the resources are in the expected state. The controller manager consists of a variety of controllers, including replication controller, endpoints controller, namespace controller, service accounts controller, etc

Different controllers manage different resources. For example, the replication controller manages the lifecycle of deployment, statefulset and daemonset, and the namespace controller manages the namespace resources

etcd

Etcd is responsible for storing the configuration information of kubernetes cluster and the status information of various resources. When the data changes, etcd will quickly inform kubernetes related components

Pod network

In order to communicate with each other, kubernetes cluster must deploy pod network, among which flannel and calico are optional.

(2) .node

Node is where pod runs. Kubernetes supports container runtime such as docker and RKT. Kubernetes components running on node include kubelet, kubelet proxy and

pod network (such as calico, flannel)

kubelet

Kubelet is the agent of a node. When the scheduler determines to run a pod on a node, it will send the specific configuration information (image, volume, etc.) of the pod to the node kubelet. Kubelet creates and runs the container according to the information, and reports the running status to the master.

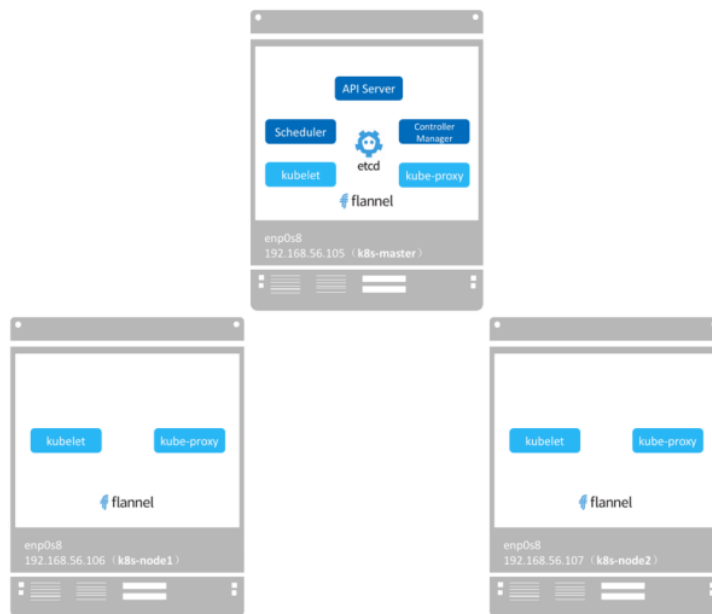
kube-proxy

Logically, service represents multiple pods on the back end, and the outside world accesses the pod through service. How is the request received by service forwarded to pod? Is that what Kube proxy does?

Each node will run the Kube proxy service, which is responsible for forwarding the TCP / UDP data flow accessing the service to the back-end container. If there are multiple copies, Kube proxy will achieve load balancing

Pod network

Pods can communicate with each other, and kubernetes cluster must deploy pod network.



(3) . instance deployment

The deployment application shows how the components work together.

Execute command:

```
kubectl run httpd-app --image=httpd --replicas=2
```

```
[root@master ~]# kubectl run nginx-app --image=nginx --replicas=2
deployment "nginx-app" created
```

Wait for a period of time, and the application deployment is completed.

```
[root@master ~]# kubectl get po
NAME                                READY    STATUS    RESTARTS   AGE
httpd-8469c85f9d-jwdqt             1/1     Running   0           3h
httpd-8469c85f9d-ntxkk             1/1     Running   0           3h
nginx-app-76cc5c87fd-9q7mx         1/1     Running   0           14s
nginx-app-76cc5c87fd-rc9gm         1/1     Running   0           14s
[root@master ~]# kubectl get deployment
NAME          DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
httpd         2         2         2             2           5h
nginx-app     2         2         2             2           34s
```

Kubernetes deployed deployment httpd app, which has two copies of pod, running on k8s-node1 and k8s-node2 respectively.

The whole deployment process is discussed in detail.



- ① Kubectl sends deployment request to API server.
- ② The API server notifies the controller manager to create a deployment resource.
- ③ The scheduler performs the scheduling task and distributes the two copies of pod to k8s-node1 and k8s-node2.
- ④ Kubectl on k8s-node1 and k8s-node2 create and run pods on their respective nodes.

Reference

- [1] Research and application of combination forecasting model based on sub proximity [J]. Guo Haiyan, Li Yanfei, Wu Tao. Journal of Jiamusi University (NATURAL SCIENCE EDITION). 2019 (04)
- [2] An elastic scaling strategy for container cloud based on load feature prediction [J]. Ma Xiaolin. Research on information security. 2019 (03)
- [3] Research on resource prediction modeubernetecharacteristics [J]. Xie Wenzhou, sun Yanxia. Network chnology and tion. 2018 (04)
- [4] The relationship between consistency and availability in distributed database [J]. Zhu Tao, Guo Jinwei, Zhou Huan, Zhou Ji, Zhou Aoying. Acta Sinica Sinica. 2018 (01)
- [5] Research on expansion and reduction mechanism of elastic web cluster based on cloud platform [J]. Wang Xiaoyu, Wu Weiming, Gu Yonghao. Software. 2017 (11)

- [6]Research on key value NoSQL local storage system [J]. Ma Wenlong, Zhu Yuqing, Jiang Dejun, Xiong Jin, Zhang Lixin, Meng Xiao, Bao Yungang. Acta computer Sinica. 2018 (08)
- [6]Dynamic migration decision algorithm of virtual machine based on load characteristics [J]. Liu Jian, Wang Malong. Network security technology and application. 2017 (03)
- [7]Improvement and application of grey prediction model based on GM (1,1) [J]. Li Mengwan, Sha Xiuyan. Computer engineering and application. 2016 (04)
- [8]Application of technology in operation system [J]. Sha ua, Chen Yong, Zhang ng. Computerring and application. 2015 (12)
- [9]A business elastic scaling algorithm based on short-term prediction [J]. Zheng Pengfei, you Jiali, Wang Jinlin, Zeng Xuewen. Computer engineering. 2014 (12)
- [11]Development mode and practice of front end and back end separation for enterprise web application [J]. Du Yanmei, Huang Xiaofang. Journal of Southwest University of science and technology. 2018 (02)
- [12]Design of operation and maintenance platform based on docker [J]. Zhang Yandong, Xing Yanfang. Computer era. 2018 (04)
- [13]"Cloud enlightenment" of Alibaba technology platform [J]. Gao Guowei. Project management review. 2018 (02)
- [14]A method of building highly available and extensible web service framework based on docker, swarm, consul and nginx [J]. Zhang Ningxi, Zhu Xiaomin. Telecom technology. 2016 (11)
- [15]Discussion on Java database connection pool [J]. Liu Huanjie, Wei Jingmin. Information system engineering. 2012 (05)

[16]Research on online migration of GUI application based on docker technology [j]. Fangjinzhang, wuter system application. 2016 (10)

[17]Research on Hadoop platform architecture based on dlele, Huang Gang, a Yue. Computer technology and development. 201

[18]aling, lichunyang, Cui Wei, Zhang Jing. Computer system application. 2016 (0

[19]The application of virtualization technology invices [j]. Bai Guojing. Electronic world. 2011 (09)

Cloud Engineering Principles and Practice Shengqing Pei

GRADEMARK REPORT

FINAL GRADE

62/100

GENERAL COMMENTS

Instructor

A good report interspersing the academic and technical issues, well written and structured.

Some of the sources trend a little bit towards the non-academic and fall out of IEEE format, but this is a minor issue.

A reasonably good report, with a good implementation. You've provided a number of sources, but this acts more as a bibliography - it's important to include inline citations in your work.

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

PAGE 14

PAGE 15

PAGE 16

PAGE 17

PAGE 18

PAGE 19

PAGE 20

PAGE 21

PAGE 22

PAGE 23

PAGE 24

PAGE 25

PAGE 26

PAGE 27

PAGE 28

PAGE 29

PAGE 30

PAGE 31

PAGE 32
