harbor安装

- 添加源

```
yum-config-manager \
    --add-repo \
    https://download.docker.com/linux/centos/docker-ce.repo
```

- 更新源

```
yum update
yum install docker-ce
```

- 设置开机启动

```
systemctl enable docker && systemctl start docker
```

- 安装compose

```
curl -L https://github.com/docker/compose/releases/download/1.18.0/docker-compose-`uname -s`-`uname -m` -o /usr/local/bin/docker-compose
chmod +x /usr/local/bin/docker-compose   #设置执行权限
docker-compose --version  #查看安装是否程成功
```

- 修改hosts文件

```
## 备份
cat /etc/hosts >> /etc/hosts.back
##添加映射
cat >> /etc/hosts << EOF
10.2.40.210 registry
EOF
#修改docker请求方式为http
cat >> /etc/docker/daemon.json << EOF
{
   "insecure-registries" :["registry:5000"]
}
EOF
# 加载修改内容
systemctl daemon-reload
stsremctl restart docker
```

- 下载离线包

```
##离线包下载地址
https://github.com/goharbor/harbor/releases
## 上传到服务
scp d:\harbor.tar.gz root@10.2.40.210:/home
#解压
tar xvf harbor.tar.gz
#修改harbor.yaml
hostname
port
password
```

```
# Configuration file of Harbor

# The IP address or hostname to access admin UI and registry service.
# DO NOT use localhost or 127.0.0.1, because Harbor needs to be accessed by
external clients.
hostname: 10.2.40.210 ## docker tag SOURCE_IMAGE[:TAG]
10.2.40.210:5000/gams2/IMAGE[:TAG]

# http related config
http:
  # port for http, default is 80. If https enabled, this port will redirect to
https port
  port: 5000 #web端访问的端口号

# 登陆密码
harbor_admin_password: 123456

# Harbor DB configuration
database:
  # The password for the root user of Harbor DB. Change this before any
production use.
  # 数据库密码
  password: 123456

# The default data volume
# 仓库数据存储路径
data_volume: /home/data

# Clair configuration
clair:
  # The interval of clair updaters, the unit is hour, set to 0 to disable the
updaters.
  updaters_interval: 12

  # Config http proxy for Clair, e.g. http://my.proxy.com:3128
  # Clair doesn't need to connect to harbor internal components via http proxy.
  http_proxy:
  https_proxy:
  no_proxy: 127.0.0.1,localhost,core,registry

jobservice:
  # Maximum number of job workers in job service
  max_job_workers: 10
```

```
chart:
  # Change the value of absolute_url to enabled can enable absolute url in chart
  absolute_url: disabled

# Log configurations
log:
  # options are debug, info, warning, error, fatal
  level: info
  # Log files are rotated log_rotate_count times before being removed. If count
is 0, old versions are removed rather than rotated.
  rotate_count: 50
  # Log files are rotated only if they grow bigger than log_rotate_size bytes.
If size is followed by k, the size is assumed to be in kilobytes.
  # If the M is used, the size is in megabytes, and if G is used, the size is in
gigabytes. So size 100, size 100k, size 100M and size 100G
  # are all valid.
  rotate_size: 200M
  # The directory on your host that store log
  location: /var/log/harbor
```

- 安装成功后

```
Creating network "harbor_harbor" with the default driver
Creating harbor-log ... done
Creating harbor-db   ... done
Creating registryctl ... done
Creating registry    ... done
Creating redis       ... done
Creating harbor-core ... done
Creating harbor-portal     ... done
Creating harbor-jobservice ... done
Creating nginx             ... done

✓ ----Harbor has been installed and started successfully.----

Now you should be able to visit the admin portal at http://10.2.7.180.
For more details, please visit https://github.com/goharbor/harbor .
```

- 仓库启动停止

```
#由于harbor是由docker-compose管理，所以启动停止也交由docker-compse管理
#进入docker-compose.yml 所在目录执行以下命令
启动Harbor
# docker-compose start
停止Harbor
# docker-compose stop
重启Harbor
# docker-compose restart
```

- 登陆容器

```
[root@registry home]# docker login registry:5000 -u admin -p Harbor12345
WARNING! Using --password via the CLI is insecure. Use --password-stdin.
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
```

- 推出容器

```
docker logout
```

- 删除所有容器

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

- 重启所有镜像

```
docker container restart  $(docker container ps -q -a)
```

- 将本地镜像上传至仓库

```
portainer                           latest                    19d07168491a
   5 months ago        74.1MB
registry:5000/portainer             latest                    19d07168491a
   5 months ago        74.1MB
centos-jdk8-u181                    latest                    afce4ff77c13
   12 months ago       581MB
registry:5000/centos-jdk8-u181      latest                    afce4ff77c13
   12 months ago       581MB
alpine                              latest                    dc98aa467aa0
   21 months ago       4.81MB
```

- 修改tag
- docker tag SOURCE_IMAGE[:TAG] 10.2.7.180:5000/library/IMAGE[:TAG]

```
[root@registry home]# docker tag sjdy521/mojo-weixin:latest
registry:5000/library/mojo-weixin:latest &&\
> docker tag registry:5000/portainer:latest
registry:5000/library/portainer:latest  &&\
> docker tag registry:5000/centos-jdk8-u181:latest
registry:5000/library/centos-jdk8-u181:latest &&\
> docker tag centos-jdk8-u181:latest   registry:5000/library/alpine:latest
```

- push
- docker push 10.2.7.180:5000/library/IMAGE[:TAG]

```
docker push  registry:5000/library/mojo-weixin:latest &&\
docker push  registry:5000/library/portainer:latest  &&\
docker push  registry:5000/library/centos-jdk8-u181:latest &&\
docker push  registry:5000/library/alpine:latest
```

- 修改构建脚本

```
cat > ./push_image.sh << EOF
#!/bin/bash

# ensure this module just could be included only once

if [ "$push" ]; then
  return
fi

export push="push.sh"


# push_image(name,version)
# return __result_pushed_version
push_image(){
  name=""
  version=""
  project=""

  if [ "$1" = "" ]; then
    echo "error: no name" 1>&2
    return 1
  else
    name=$1
  fi

  if [ "$2" = "" ];then
    version=`date +%y%m%d-%H%M%S`
  else
    version=$2
  fi

  if [ "$3" = "" ]; then
    echo "error: no project" 1>&2
    return 1
  else
    project=$3
  fi

  docker login registry:5000 -u builder -p 123456
  docker tag $name registry:5000/$project/$name
  #docker push registry:5000/$project/$name
  docker push registry:5000/$project/$name:latest

  #add version tag and push to registry
  docker tag registry:5000/$project/$name registry:5000/$project/$name:$version
  docker push registry:5000/$project/$name:$version
  docker logout
  __result_pushed_version=$version
```

```
}

EOF
```

```
cat > ./push.sh << EOF
#!/bin/bash

realpath=$(readlink -f "$0")
export basedir=$(dirname "$realpath")
export filename=$(basename "$realpath")

export PATH=$PATH:$basedir/../modules
. push_image.sh

image_name=`head -1 $basedir/_image_name`
projec_tname=`head -1 $basedir/_project_name`
#version=$1-`date +%y%m%d-%H%M%S`
version=`date +%y%m%d-%H%M%S`
push_image $image_name $version $projec_tname
result=$?
#version=$1-$__result_pushed_version

if [ "$result" -eq "0" ]; then
  echo $version > $basedir/_version
  #echo $version > $basedir/_version-$1
  echo "push $image_name:$version completed."
else
 exit
fi
EOF
```

- 删除镜像

```
#!/bin/bash
function rmi_images(){
   docker rmi registry:5000/$2/$1:`cat _version`
   docker rmi registry:5000/$2/$1:latest
   docker rmi $1:latest
}
project=`head -1 _project_name`
IMAGE_NAME=`cat _image_name`
echo $IMAGE_NAME
rmi_images $IMAGE_NAME $project
```