목차

Openstack 16.1 offline registry 구성 3
Shell Script 9
참조링크 11

Openstack 16.1 offline registry 구성

Podman 과 httpd-tools 를 먼저 설치한다. podman 패키지는 레지스트리를 제공하고 httpd-tools 패키지는 htpasswd 인증을 위한 유틸리티를 제공한다.

```
[root@registry ~]# dnf install -y podman httpd-tools
```

레지스트리를 위한 폴더를 생성한다. 레지스트리는 /opt/registry/에 저장된다.

```
[root@registry ~]# mkdir -p /opt/registry/{auth,certs,data}
```

- Auth 하위 디렉토리는 인증에 사용되는 htpasswd 파일을 저장합니다.
- Certs 하위 디렉토리는 인증을 위해 레지스트리에서 사용하는 인증서를 저장합니다.
- Data 디렉토리는 레지스트리에 저장된 실제 이미지를 저장합니다.

레지스트리에 액세스하기 위한 자격 증명을 생성합니다. 인증은 간단한 htpasswd 파일과 SSL 키 쌍에서 제공됩니다. htpasswd 유틸리티를 사용하여 레지스트리에 액세스하기 위한 자격 증명이 포함된 파일을 생성합니다.

```
[root@registry ~]# htpasswd -bBc /opt/registry/auth/htpasswd <username>
<password>
```

레지스트리는 신뢰할 수 있는 권한(내부 또는 외부)이 서명한 키 및 인증서또는 간단한 자체 서명 인증서로 TLS로 보호됩니다. 자체 서명된 인증서를 사용하려면 다음을 수행합니다.

```
[root@registry ~]# openssl req -newkey rsa:4096 -nodes -sha256 -keyout
/opt/registry/certs/domain.key -x509 -days 365 -out
/opt/registry/certs/domain.crt
```

인증서는 호스트와 클라이언트에서 신뢰할 수 있어야 합니다.

```
[root@registry ~]# cp /opt/registry/certs/domain.crt /etc/pki/ca-
trust/source/anchors/
[root@registry ~]# update-ca-trust
[root@registry ~]# trust list | grep -i "registry"
    label: registry
```

다음 단계는 노출된 포트(5000)와 이전에 만든 각 디렉터리의 탑재된 볼륨으로 레지스트리 이미지를 실행하는 것입니다. 레지스트리를 시작하는 명령:

```
[root@registry ~]# podman run --name myregistry \
```

```
-p 5000:5000 \
-v /opt/registry/data:/var/lib/registry:z \
-v /opt/registry/auth:/auth:z \
-v /opt/registry/certs:/certs:z \
-e "REGISTRY_AUTH=htpasswd" \
-e "REGISTRY_AUTH_HTPASSWD_REALM=Registry Realm" \
-e REGISTRY_AUTH_HTPASSWD_PATH=/auth/htpasswd \
-e "REGISTRY_HTTP_TLS_CERTIFICATE=/certs/domain.crt" \
-e "REGISTRY_HTTP_TLS_KEY=/certs/domain.key" \
-e REGISTRY_COMPATIBILITY_SCHEMA1_ENABLED=true \
-d docker.io/library/registry:latest
```

호스트에서 방화벽이 실행 중인 경우 노출된 포트(5000)를 허용해야 합니다.

```
[root@registry ~]# firewall-cmd --add-port=5000/tcp --zone=internal --
permanent
[root@registry ~]# firewall-cmd --add-port=5000/tcp --zone=public --
permanent
[root@registry ~]# firewall-cmd --reload
```

레지스트리에 대한 액세스를 확인하려면 컬 명령을 사용하여 레지스트리에 액세스할 수 있습니다.

```
[root@registry ~]# curl https://registry:5000/v2/_catalog
{"repositories":[]}
```

인증서는 다음을 사용하여 확인할 수 있습니다.

```
[root@registry ~]# openssl s_client -connect registry:5000 -servername
registry
```

포드맨 컨테이너를 시스템 된 서비스로 만듭니다.

```
[root@registry ~]# podman generate systemd myregistry >
/etc/systemd/system/podman.registry.service
```

생성된 파일의 내용입니다.

```
# container-
fb88c601f4b822ea5035f4a6f5fbff492b136cf991114e473085f60ec9b722fb.service
# autogenerated by Podman 1.9.3
# Thu Jan 28 19:38:46 -03 2021

[Unit]
```

```
Description=Podman container-
fb88c601f4b822ea5035f4a6f5fbff492b136cf991114e473085f60ec9b722fb.service
Documentation=man:podman-generate-systemd(1)
Wants=network.target
After=network-online.target

[Service]
Environment=PODMAN_SYSTEMD_UNIT=%n
Restart=on-failure
ExecStart=/usr/bin/podman start
fb88c601f4b822ea5035f4a6f5fbff492b136cf991114e473085f60ec9b722fb
ExecStop=/usr/bin/podman stop -t 10
fb88c601f4b822ea5035f4a6f5fbff492b136cf991114e473085f60ec9b722fb
PIDFile=/var/run/containers/storage/overlay-
containers/fb88c601f4b822ea5035f4a6f5fbff492b136cf991114e473085f60ec9b722fb/
userdata/conmon.pid
KillMode=none
Type=forking

[Install]
WantedBy=multi-user.target default.target
```

서비스가 부팅에서 시작할 수 있도록 합니다.

```
[root@registry ~]# systemctl daemon-reload
[root@registry ~]# systemctl enable podman.registry.service
```

podman 로그인 명령을 사용하여 레지스트리에 로그인합니다. 레지스트리는 registry.access.redhat.com, registry.redhat.io, docker.io 등과 같은 다른 레지스트리와 마찬가지로 액세스하고 상호 작용할 수 있습니다.

```
[root@registry ~]# podman login registry:5000
Enter Username:xxxxxxxx
Enter Password:yyyyyyyy
Login Succeeded!
```

레지스트리에서 단일 이미지를 가져오려면 소스로 인증을 진행하십시오.

```
[root@registry ~]# podman login registry.redhat.io
Enter Username:xxxxxxxx
Enter Password:yyyyyyyy
Login Succeeded!
```

이와 같은 이미지를 당겨 레지스트리 위치를 리포지토리/이미지 이름으로 준비합니다.

```
Example: podman pull <Registry Hostname>:<Registry Port>/<Repository>/<Image
Name>

[root@registry ~]#  podman pull registry.redhat.io/rhosp-rhel8/openstack-
ceilometer-base:16.1
Trying to pull registry.redhat.io/rhosp-rhel8/openstack-ceilometer-
base:16.1...
Getting image source signatures
Copying blob 59ba4923ade0 done
Copying blob f7db571f3a05 done
Copying blob 97b855f4d380 done
Copying blob fecf0868d374 done
Copying blob 4ad79ef7dca0 done
Copying config 0bce4d1d0e done
Writing manifest to image destination
Storing signatures
0bce4d1d0e3024e0427e70d65bb14a1db0418699b8466cb1a8953e77ef5e6d1d
```

레드 햇 오픈 스택 플랫폼 16.1 레지스트리에서 모든 이미지를 당겨옵니다.

```
[root@registry ~]# for i in `podman search --limit 1000
"registry.redhat.io/rhosp" | grep rhosp-rhel8 | awk '{ print $2 }' | grep -v
beta | sed "s/registry.redhat.io\///g" | tail -n+2`; do podman pull
registry.redhat.io/$i:16.1 ; done
```

레지스트리로 푸시하려면 podman 태그를 사용하여 먼저 이미지와 레지스트리 위치에 태그를 붙인 다음 이미지를 푸시합니다.

```
Example: podman tag <image id|<repo name/image name>
registry:5000/<repo>/<image>
         podman push <image id|<repo name/image name>
registry:5000/<repo>/<image>
[root@registry ~]# podman tag registry.redhat.io/rhosp-rhel8/openstack-
ceilometer-base:16.1 registry:5000/rhosp-rhel8/openstack-ceilometer-
base:16.1
[root@registry ~]# podman push registry:5000/rhosp-rhel8/openstack-
ceilometer-base:16.1
Getting image source signatures
Copying blob 226bfaae015f done
…
Storing signatures
```

모든 이미지를 태그하고 푸시하려면 다음을 수행합니다.

```
[root@registry ~]# for i in `podman search --limit 1000
"registry.redhat.io/rhosp" | grep rhosp-rhel8 | awk '{ print $2 }' | grep -v
beta | sed "s/registry.redhat.io\///g" | tail -n+2`; do podman tag
registry.redhat.io/$i:16.1 registry:5000/$i:16.1 ; done
[root@registry ~]# for i in `podman search --limit 1000
"registry.redhat.io/rhosp" | grep rhosp-rhel8 | awk '{ print $2 }' | grep -v
beta | sed "s/registry.redhat.io\///g" | tail -n+2`; do podman push
registry:5000/$i:16.1 ; done
Getting image source signatures
Copying blob 96c54a12373b done
Copying blob a196c162c139 done
Copying blob c483fe8f3e2d done
Copying blob 909bef9af69e done
Copying blob a79745bee9c6 done
Copying blob ca4f647c4fbb done
Copying config 7a7dbd538a done
Writing manifest to image destination
Storing signatures
(...)
```

curl 명령을 통해 이미지가 레지스트리에 업로드되었는지 확인할 수 있습니다.

```
[root@registry ~]# curl -u <username>:<password>
https://registry:5000/v2/_catalog
{"repositories":["rhosp-rhel8/openstack-ceilometer-base"]}
```

인증서를 디렉터/언더클라우드로 가져옵니다.

```
[stack@director ~]$ sudo su -
[root@director ~]#  vim /etc/pki/ca-trust/source/anchors/domain-registry.crt
[root@director ~]# update-ca-trust
[root@director ~]# trust list | grep -i "registry"
    label: registry
[root@director ~]# logout
```

레지스트리에 로그인하고 컨테이너를 나열합니다.

```
[stack@director ~]$  podman login registry:5000
Username: user
Password:
Login Succeeded!
[stack@director ~]$ podman search --limit 1000 "registry:5000/rhosp" | grep
rhosp-rhel8 | awk '{ print $2 }' | grep -v beta | sed "s/registry:5000\///g"
| tail -n+2
rhosp-rhel8/openstack-aodh-base
```

```
(...)
rhosp-rhel8/openstack-panko-base
```

개인 레지스트리를 사용하도록 스택을 구성합니다.

일부 컨테이너 이미지 레지스트리는 이미지에 액세스하려면 인증이 필요합니다. 이 경우 `containers-prepare-parameter.yaml` 환경 파일에서 `ContainerImageRegistryCredentials` 매개 변수를 사용합니다.

```
parameter_defaults:
  (...)
  ContainerImageRegistryCredentials:
   registry.example.com:
   'username': "p@55w0rd!"
```

Private 레지스트리는 `ContainerImagePrepare` 에서 각 전략에 대해 `push_destination` 설정해야 합니다.

```
parameter_defaults:
  (...)
  ContainerImagePrepare:
  - push_destination: true
```

`ContainerImageRegistryLogin` 매개 변수는 시스템이 컨테이너를 가져오기 위해 원격 레지스트리에 로그인해야 하는지 여부를 제어하는 데 사용됩니다.

```
parameter_defaults:
  (...)
  ContainerImageRegistryLogin: true
```

레지스트리 시작

```
[root@registry ~]# podman run --name myregistry -p 5000:5000
-v /opt/registry/data:/var/lib/registry:z \
-v /opt/registry/auth:/auth:z \
-e "REGISTRY_AUTH=htpasswd" \
-e "REGISTRY_AUTH_HTPASSWD_REALM=Registry Realm" \
-e REGISTRY_AUTH_HTPASSWD_PATH=/auth/htpasswd \
-v /opt/registry/certs:/certs:z \
-e REGISTRY_COMPATIBILITY_SCHEMA1_ENABLED=true \
-d docker.io/library/registry:latest
```

레지스트리 중단

```
[root@registry ~]# podman stop myregistry
```

레지스트리 제거

```
[root@registry ~]# podman container rm myregistry
```

레지스트리 이미지 삭제

```
[root@registry ~]# podman image rm registry:latest
```

인증서 제거

```
[root@registry ~]# rm /etc/pki/ca-trust/source/anchors/domain.crt
[root@registry ~]# update-ca-trust
[root@registry ~]# trust list | grep "registry"
```

Shell Script

위와같은 방법으로 로컬 레지스트리 구성 후 컨테이너 이미지를 자동으로 땡겨와서 로컬 레지스트리에 탑재 하는 스크립트 이다. 이 스크립트는 스토리지 엔진 컨테이너까지 포함시키므로 필요시 수정하여 사용하면 된다.

```
#!/bin/bash
USERNAME="USERNAME"
PASSWORD="PASSWORD"
REGISTRY="registry.redhat.io"
REGISTRY_STORAGE="registry.connect.redhat.com"
NAMESPACE="rhosp-rhel8"
TAG="16.2"
LOCAL="registry:5000"
NAMEPREFIX="openstack-"
MODE=$1

if [[ -z "${MODE}" ]]; then
    echo "Usage: $0 <all|openstack|ceph|storage>";
    exit;
```

```
fi

# login
podman login --username ${USERNAME} --password ${PASSWORD} ${REGISTRY}

# openstack image
if [[ "${MODE}" == "openstack" || "${MODE}" == "all" ]]; then
    echo -e "\e[1;32m### Openstack container image\e[0m";
    LIST=`podman search --limit 1000 "${REGISTRY}/${NAMESPACE}" | awk '{
print $2 }' | grep ${NAMEPREFIX} | grep -v beta | grep -v preview | sed
"s/${REGISTRY}\///g" | grep -v NAME`

  ### 이미지 pull
    for i in ${LIST[@]}; do echo -e "\e[1;32m${i}\e[0m"; podman pull
${REGISTRY}/${i}:${TAG} ; done
  ### 이미지 로컬 레지스트리로 tag
    for i in ${LIST[@]}; do echo -e "\e[1;32m${i}\e[0m"; podman tag
${REGISTRY}/${i}:${TAG} ${LOCAL}/${i}:${TAG} ; done
  ### 이미지 로컬 레지스트리로 push
    for i in ${LIST[@]}; do echo -e "\e[1;32m${i}\e[0m"; podman push
${LOCAL}/${i}:${TAG} --remove-signatures; done
fi

### ceph image
if [[ "${MODE}" == "ceph" || "${MODE}" == "all" ]]; then
    echo -e "\e[1;32m### Ceph Storage container image\e[0m";
    podman pull ${REGISTRY}/openshift4/ose-prometheus-alertmanager:v4.6
    podman pull ${REGISTRY}/rhceph/rhceph-4-dashboard-rhel8:4
    podman pull ${REGISTRY}/rhceph/rhceph-4-rhel8
    podman pull ${REGISTRY}/openshift4/ose-prometheus-node-exporter:v4.6
    podman pull ${REGISTRY}/openshift4/ose-prometheus:v4.6

    podman tag ${REGISTRY}/openshift4/ose-prometheus-alertmanager:v4.6
${LOCAL}/openshift4/ose-prometheus-alertmanager:v4.6
    podman tag ${REGISTRY}/rhceph/rhceph-4-dashboard-rhel8:4
${LOCAL}/rhceph/rhceph-4-dashboard-rhel8:4
    podman tag ${REGISTRY}/rhceph/rhceph-4-rhel8 ${LOCAL}/rhceph/rhceph-4-
rhel8
    podman tag ${REGISTRY}/openshift4/ose-prometheus-node-exporter:v4.6
${LOCAL}/openshift4/ose-prometheus-node-exporter:v4.6
    podman tag ${REGISTRY}/openshift4/ose-prometheus:v4.6
${LOCAL}/openshift4/ose-prometheus:v4.6

    podman push ${LOCAL}/openshift4/ose-prometheus-alertmanager:v4.6 --
remove-signatures
    podman push ${LOCAL}/rhceph/rhceph-4-dashboard-rhel8:4 --remove-
signatures
    podman push ${LOCAL}/rhceph/rhceph-4-rhel8 --remove-signatures
    podman push ${LOCAL}/openshift4/ose-prometheus-node-exporter:v4.6 --
remove-signatures
```

```
    podman push ${LOCAL}/openshift4/ose-prometheus:v4.6 --remove-signatures
fi

### purestorage image
if [[ "${MODE}" == "storage" || "${MODE}" == "all" ]]; then
    echo -e "\e[1;32m### Openstack Storage container image\e[0m";
    # login
    podman login --username ${USERNAME} --password ${PASSWORD}
${REGISTRY_STORAGE}

    #podman pull registry.connect.redhat.com/purestorage/openstack-cinder-
volume-pure-rhosp-16-2:latest
    #podman tag registry.connect.redhat.com/purestorage/openstack-cinder-
volume-pure-rhosp-16-2:latest ${LOCAL}/purestorage/openstack-cinder-volume-
pure-rhosp-16-2:latest
    podman pull ${REGISTRY_STORAGE}/purestorage/openstack-cinder-volume-
pure-rhosp-16-2:latest
    podman tag ${REGISTRY_STORAGE}/purestorage/openstack-cinder-volume-pure-
rhosp-16-2:latest ${LOCAL}/purestorage/openstack-cinder-volume-pure-
rhosp-16-2:latest
    podman push ${LOCAL}/purestorage/openstack-cinder-volume-pure-
rhosp-16-2:latest --remove-signatures
fi
```

참조링크

- https://access.redhat.com/solutions/5752401

From:
https://atl.kr/dokuwiki/ - **AllThatLinux!**

Permanent link:
https://atl.kr/dokuwiki/doku.php/openstack_16.1_offline_registry_%EA%B5%AC%EC%84%B1

Last update: **2023/05/08 06:06**