Private Registry

Basic Registry - No SSL & Authentication

내부에서 통신구간에 암호화를 하지 않을 경우 비보안 통신을 허용하도록 insecure registry에 해당 레지스트리를 추가할 수 있습니다. 다만 이는 보안적으로 취약할 수 있으니 권장하는 사례는 아니지만 성능을 위해서 때로는 암호화/복호화를 수행하지 않도록 구성할 수 있습니다. registry 서버를 구성할 때는 다음과 같이 구성합니다.

```
# On registry node
docker run -d -p 5000:5000 \
-v /data:/var/lib/registry \
--name registry-server \
--restart always registry:latest
```

insecure registry 등록

client 에서는 반드시 비보안 통신을 허용하도록 insecure registries 에 사용하는 registry를 등록해야 합니다. 각 node마다 /etc/docker/daemon.json 파일을 새로 생성해서 다음 내용을 추가합니다.

```
# edit /etc/docker/daemon.json on each node
{
 "insecure-registries" : ["myregistry.com:5000"]
}
```

파일을 등록한 뒤 docker 데몬을 재기동해서 설정을 적용하도록 합니다. 이때 노드에서 동작하는 컨테이너들은 중지 될 수 있습니다.

```
systemctl restart docker
```

kubernetes에서 테스트

```
docker pull docker.io/nginx
docker tag docker.io/nginx myregistry.com:5000/nginx
```

```
docker push myregistry.com:5000/nginx
kubectl create deployment mynginx --image=myregistry.com:5000/nginx
kubectl get pod -w
```

Clean up

```
kubectl delete deploy mynginx
docker rmi myregistry.com:5000/nginx
docker rm -f registry-server
```

Registry With SSL & Authentication

인증서를 통해 암호화 통신을 구성하고 사용자 계정과 암호를 통해 인증하는 registry를 구성하면 좀더 안전한 시스템을 구성할 수 있습니다. 다음은 openssl모듈로 인증서를 생성하고 htpasswd 모듈을 통해 인증 프로바이더를 구성하는 registry server의 예시입니다.

```
# deploy test registry
mkdir ~/auth ; cd ~/auth

docker run --rm -v /root/auth:/auth alpine/openssl \
req -newkey rsa:4096 -nodes -sha256 -x509 \
-days 365 -keyout /auth/myregistry.com.key -out /auth/myregistry.com.crt \
-subj '/CN=myregistry.com' \
-addext "subjectAltName = DNS:myregistry.com"

mkdir -p /etc/docker/certs.d/myregistry.com/

cp myregistry.com.crt /etc/docker/certs.d/myregistry.com/ca.crt

docker run --rm --entrypoint htpasswd httpd:2.4 -Bbn demouser password> htpasswd

docker run -d -p 443:443 \
 --restart=always --name registry \
 -v /image-data:/var/lib/registry \
 -v /root/auth:/certs \
 -e REGISTRY_HTTP_TLS_CERTIFICATE=/certs/myregistry.com.crt \
 -e REGISTRY_HTTP_TLS_KEY=/certs/myregistry.com.key \
 -e REGISTRY_HTTP_ADDR=0.0.0.0:443 \
-e "REGISTRY_AUTH=htpasswd" \
 -e "REGISTRY_AUTH_HTPASSWD_REALM=Registry Realm" \
 -e REGISTRY_AUTH_HTPASSWD_PATH=/certs/htpasswd \
 registry
```

사용자는 client(master, node1, node2) 에 인증서를 배포하고, 사용자 계정과 암호 정보를 통해 인증하는 자격증명을 kubernetes에서 공유해서 사용할수 있도록 secret을 생성합니다.

```
# deploy certification to kubernetes
scp -r /etc/docker/certs.d master:/etc/docker/
scp -r /etc/docker/certs.d node1:/etc/docker/
scp -r /etc/docker/certs.d node2:/etc/docker/

# create secret
kubectl create secret docker-registry registry-secret \
--docker-server=myregistry.com \
--docker-username=demouser \
--docker-password=password

# verify Secret data
kubectl get secret registry-secret\
--output="jsonpath={.data.\.dockerconfigjson}" | base64 --decode
```

registry (server) 와 kubernetes (client) 가 모두 준비 되었으면 테스트를 진행합니다. 먼저 image를 registry에 게시합니다.

```
# Image push
docker pull nginx
docker tag nginx myregistry.com/mynginx:custom
docker push myregistry.com/mynginx:custom
```

테스트용 pod인 nginx-pod.yaml 파일을 생성합니다.

```
#nginx-pod.yaml
apiVersion: v1
kind: Pod
metadata:
  name: mynginx
spec:
  containers:
  - name: mynginx
    image: myregistry.com/mynginx:custom
  imagePullSecrets:
  - name: registry-secret
```

```
kubectl create -f mynginx.yaml
kubectl get pod
```

올바르게 동작이 된다면 running 상태로 출력됩니다. 만약 error가 발생된다면 다음 명령어로 해당 내용을 확인 할수 있습니다.

```
kubectl describe pod mynginx
```