tags: registry, ceph

09. 部署 docker registry

本文档介绍使用 docker 官方的 registry v2 镜像部署私有仓库的步骤，你也可以参考附件文档部署 Harbor 私有仓库（D.部署 Harbor 私有仓库）。

本文档讲解部署一个 TLS 加密、HTTP Basic 认证、用 ceph rgw 做后端存储的私有 docker registry 步骤，如果使用其它类型的后端存储，则可以从“创建 docker registry”节开始；

示例两台机器 IP 如下：

- ceph rgw: 172.27.132.66
- docker registry: 172.27.132.67

部署 ceph RGW 节点

```
$ ceph-deploy rgw create 172.27.132.66 # rgw 默认监听7480端口
$
```

创建测试账号 demo

```
$ radosgw-admin user create --uid=demo --display-name="ceph rgw demo user"
$
```

创建 demo 账号的子账号 swift

当前 registry 只支持使用 swift 协议访问 ceph rgw 存储，暂时不支持 s3 协议；

```
$ radosgw-admin subuser create --uid demo --subuser=demo:swift --access=full --
secret=secretkey --key-type=swift
$
```

创建 demo:swift 子账号的 sercret key

```
$ radosgw-admin key create --subuser=demo:swift --key-type=swift --gen-secret
{
    "user_id": "demo",
    "display_name": "ceph rgw demo user",
    "email": "",
    "suspended": 0,
    "max_buckets": 1000,
    "auid": 0,
    "subusers": [
        {
            "id": "demo:swift",
            "permissions": "full-control"
        }
    ],
    "keys": [
        {
            "user": "demo",
            "access_key": "5Y1B1SIJ2YHKEHO5U36B",
            "secret_key": "nrIvtPqUj7pUlccLYPuR3ntVzIa50DToIpe7xFjT"
        }
    ],
    "swift_keys": [
        {
            "user": "demo:swift",
            "secret_key": "ttQcU1O17DFQ4I9xzKqwgUe7WIYYX99zhcIfU9vb"
        }
    ],
    "caps": [],
    "op_mask": "read, write, delete",
```

```
    "default_placement": "",
    "placement_tags": [],
    "bucket_quota": {
        "enabled": false,
        "max_size_kb": -1,
        "max_objects": -1
    },
    "user_quota": {
        "enabled": false,
        "max_size_kb": -1,
        "max_objects": -1
    },
        "temp_url_keys": []
}
```

- `ttQcU1O17DFQ4I9xzKqwgUe7WIYYX99zhcIfU9vb` 为子账号 demo:swift 的 secret key；

创建 docker registry

创建 registry 使用的 x509 证书

```
$ mkdir -p registry/{auth,certs}
$ cat > registry-csr.json <<EOF
{
  "CN": "registry",
  "hosts": [
      "127.0.0.1",
      "172.27.132.67"
  ],
  "key": {
    "algo": "rsa",
    "size": 2048
  },
  "names": [
    {
      "C": "CN",
      "ST": "BeiJing",
      "L": "BeiJing",
      "O": "k8s",
      "OU": "opsnull"
    }
  ]
}
EOF
$ cfssl gencert -ca=/etc/kubernetes/cert/ca.pem \
    -ca-key=/etc/kubernetes/cert/ca-key.pem \
    -config=/etc/kubernetes/cert/ca-config.json \
    -profile=kubernetes registry-csr.json | cfssljson -bare registry
$ cp registry.pem registry-key.pem registry/certs
```

```
$
```

- 这里复用以前创建的 CA 证书和秘钥文件；
- hosts 字段指定 registry 的 NodeIP；

创建 HTTP Baisc 认证文件

```
$ docker run --entrypoint htpasswd registry:2 -Bbn foo foo123  >
registry/auth/htpasswd
$ cat  registry/auth/htpasswd
foo:$2y$05$iZaM45Jxlcg0DJKXZMggLOibAsHLGybyU.CgU9AHqWcVDyBjiScN.
```

配置 registry 参数

```
export RGW_AUTH_URL="http://172.27.132.66:7480/auth/v1"
export RGW_USER="demo:swift"
export RGW_SECRET_KEY="ttQcU1O17DFQ4I9xzKqwgUe7WIYYX99zhcIfU9vb"
cat > config.yml << EOF
# https://docs.docker.com/registry/configuration/#list-of-configuration-options
version: 0.1
log:
  level: info
  fromatter: text
  fields:
    service: registry

storage:
  cache:
    blobdescriptor: inmemory
  delete:
    enabled: true
  swift:
    authurl: ${RGW_AUTH_URL}
    username: ${RGW_USER}
    password: ${RGW_SECRET_KEY}
    container: registry

auth:
  htpasswd:
    realm: basic-realm
    path: /auth/htpasswd

http:
  addr: 0.0.0.0:8000
  headers:
    X-Content-Type-Options: [nosniff]
  tls:
    certificate: /certs/registry.pem
    key: /certs/registry-key.pem

health:
```

```
  storagedriver:
    enabled: true
    interval: 10s
    threshold: 3
EOF
[k8s@zhangjun-k8s-01 cert]$ cp config.yml registry
[k8s@zhangjun-k8s-01 cert]$ scp -r registry 172.27.132.67:/opt/k8s
```

- storage.swift 指定后端使用 swfit 接口协议的存储，这里配置的是 ceph rgw 存储参数；
- auth.htpasswd 指定了 HTTP Basic 认证的 token 文件路径；
- http.tls 指定了 registry http 服务器的证书和秘钥文件路径；

创建 docker registry：

```
ssh k8s@172.27.132.67
$ docker run -d -p 8000:8000 --privileged \
    -v /opt/k8s/registry/auth/:/auth \
    -v /opt/k8s/registry/certs:/certs \
    -v /opt/k8s/registry/config.yml:/etc/docker/registry/config.yml \
    --name registry registry:2
```

- 执行该 docker run 命令的机器 IP 为 172.27.132.67；

向 registry push image

将签署 registry 证书的 CA 证书拷贝到 `/etc/docker/certs.d/172.27.132.67:8000` 目录下

```
[k8s@zhangjun-k8s-01 cert]$ sudo mkdir -p /etc/docker/certs.d/172.27.132.67:8000
[k8s@zhangjun-k8s-01 cert]$ sudo cp /etc/kubernetes/cert/ca.pem
/etc/docker/certs.d/172.27.132.67:8000/ca.crt
```

登陆私有 registry：

```
$ docker login 172.27.132.67:8000
Username: foo
Password:
Login Succeeded
```

登陆信息被写入 `~/.docker/config.json` 文件：

```
$ cat ~/.docker/config.json
{
        "auths": {
                "172.27.132.67:8000": {
                        "auth": "Zm9vOmZvbzEyMw=="
                }
        }
}
```

将本地的 image 打上私有 registry 的 tag：

```
$ docker tag prom/node-exporter:v0.16.0 172.27.132.67:8000/prom/node-exporter:v0.16.0
$ docker images |grep pause
prom/node-exporter:v0.16.0                              latest
f9d5de079539        2 years ago         239.8 kB
172.27.132.67:8000/prom/node-exporter:v0.16.0                        latest
   f9d5de079539        2 years ago         239.8 kB
```

将 image push 到私有 registry：

```
$ docker push 172.27.132.67:8000/prom/node-exporter:v0.16.0
The push refers to a repository [172.27.132.67:8000/prom/node-exporter:v0.16.0]
5f70bf18a086: Pushed
e16a89738269: Pushed
latest: digest:
sha256:9a6b437e896acad3f5a2a8084625fdd4177b2e7124ee943af642259f2f283359 size: 916
```

查看 ceph 上是否已经有 push 的 pause 容器文件：

```
$ rados lspools
rbd
cephfs_data
cephfs_metadata
.rgw.root
k8s
default.rgw.control
default.rgw.meta
default.rgw.log
default.rgw.buckets.index
default.rgw.buckets.data

$  rados --pool default.rgw.buckets.data ls|grep node-exporter
1f3f02c4-fe58-4626-992b-
c6c0fe4c8acf.34107.1_files/docker/registry/v2/repositories/prom/node-
exporter/_layers/sha256/cdb7590af5f064887f3d6008d46be65e929c74250d747813d85199e04fc70
463/link
```

```
1f3f02c4-fe58-4626-992b-
c6c0fe4c8acf.34107.1_files/docker/registry/v2/repositories/prom/node-
exporter/_manifests/revisions/sha256/55302581333c43d540db0e144cf9e7735423117a733cdec2
7716d87254221086/link
1f3f02c4-fe58-4626-992b-
c6c0fe4c8acf.34107.1_files/docker/registry/v2/repositories/prom/node-
exporter/_manifests/tags/v0.16.0/current/link
1f3f02c4-fe58-4626-992b-
c6c0fe4c8acf.34107.1_files/docker/registry/v2/repositories/prom/node-
exporter/_manifests/tags/v0.16.0/index/sha256/55302581333c43d540db0e144cf9e7735423117
a733cdec27716d87254221086/link
1f3f02c4-fe58-4626-992b-
c6c0fe4c8acf.34107.1_files/docker/registry/v2/repositories/prom/node-
exporter/_layers/sha256/224a21997e8ca8514d42eb2ed98b19a7ee2537bce0b3a26b8dff510ab637f
15c/link
1f3f02c4-fe58-4626-992b-
c6c0fe4c8acf.34107.1_files/docker/registry/v2/repositories/prom/node-
exporter/_layers/sha256/528dda9cf23d0fad80347749d6d06229b9a19903e49b7177d5f4f58736538
d4e/link
1f3f02c4-fe58-4626-992b-
c6c0fe4c8acf.34107.1_files/docker/registry/v2/repositories/prom/node-
exporter/_layers/sha256/188af75e2de0203eac7c6e982feff45f9c340eaac4c7a0f59129712524fa2
984/link
```

私有 registry 的运维操作

查询私有镜像中的 images

```
$ curl  --user foo:foo123 --cacert /etc/docker/certs.d/172.27.132.67\:8000/ca.crt
https://172.27.132.67:8000/v2/_catalog
{"repositories":["prom/node-exporter"]}
```

查询某个镜像的 tags 列表

```
$  curl  --user foo:foo123 --cacert /etc/docker/certs.d/172.27.132.67\:8000/ca.crt
https://172.27.132.67:8000/v2/prom/node-exporter/tags/list
{"name":"prom/node-exporter","tags":["v0.16.0"]}
```

获取 image 或 layer 的 digest

向 v2/<repoName>/manifests/<tagName> 发 GET 请求，从响应的头部 Docker-Content-Digest 获取 image digest，从响应的 body 的 fsLayers.blobSum 中获取 layDigests;

注意，必须包含请求头：Accept: application/vnd.docker.distribution.manifest.v2+json：

```
$ curl -v -H "Accept: application/vnd.docker.distribution.manifest.v2+json" --user
foo:foo123 --cacert /etc/docker/certs.d/172.27.132.67\:8000/ca.crt
https://172.27.132.67:8000/v2/prom/node-exporter/manifests/v0.16.0
* About to connect() to 172.27.132.67 port 8000 (#0)
*   Trying 172.27.132.67...
* Connected to 172.27.132.67 (172.27.132.67) port 8000 (#0)
* Initializing NSS with certpath: sql:/etc/pki/nssdb
*   CAfile: /etc/docker/certs.d/172.27.132.67:8000/ca.crt
  CApath: none
* SSL connection using TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
* Server certificate:
*       subject: CN=registry,OU=4Paradigm,O=k8s,L=BeiJing,ST=BeiJing,C=CN
*       start date: Jul 05 12:52:00 2018 GMT
*       expire date: Jul 02 12:52:00 2028 GMT
*       common name: registry
*       issuer: CN=kubernetes,OU=4Paradigm,O=k8s,L=BeiJing,ST=BeiJing,C=CN
* Server auth using Basic with user 'foo'
> GET /v2/prom/node-exporter/manifests/v0.16.0 HTTP/1.1
> Authorization: Basic Zm9vOmZvbzEyMw==
> User-Agent: curl/7.29.0
> Host: 172.27.132.67:8000
> Accept: application/vnd.docker.distribution.manifest.v2+json
>
< HTTP/1.1 200 OK
< Content-Length: 949
< Content-Type: application/vnd.docker.distribution.manifest.v2+json
< Docker-Content-Digest:
sha256:55302581333c43d540db0e144cf9e7735423117a733cdec27716d87254221086
< Docker-Distribution-Api-Version: registry/2.0
< Etag: "sha256:55302581333c43d540db0e144cf9e7735423117a733cdec27716d87254221086"
< X-Content-Type-Options: nosniff
< Date: Fri, 06 Jul 2018 06:18:41 GMT
<
{
   "schemaVersion": 2,
   "mediaType": "application/vnd.docker.distribution.manifest.v2+json",
   "config": {
      "mediaType": "application/vnd.docker.container.image.v1+json",
      "size": 3511,
      "digest":
"sha256:188af75e2de0203eac7c6e982feff45f9c340eaac4c7a0f59129712524fa2984"
   },
   "layers": [
      {
```

```
         "mediaType": "application/vnd.docker.image.rootfs.diff.tar.gzip",
         "size": 2392417,
         "digest":
"sha256:224a21997e8ca8514d42eb2ed98b19a7ee2537bce0b3a26b8dff510ab637f15c"
      },
      {
         "mediaType": "application/vnd.docker.image.rootfs.diff.tar.gzip",
         "size": 560703,
         "digest":
"sha256:cdb7590af5f064887f3d6008d46be65e929c74250d747813d85199e04fc70463"
      },
      {
         "mediaType": "application/vnd.docker.image.rootfs.diff.tar.gzip",
         "size": 5332460,
         "digest":
"sha256:528dda9cf23d0fad80347749d6d06229b9a19903e49b7177d5f4f58736538d4e"
      }
   ]
```

删除 image

向 `/v2/<name>/manifests/<reference>` 发送 DELETE 请求，reference 为上一步返回的 Docker-Content-Digest 字段内容：

```
$ curl -X DELETE  --user foo:foo123 --cacert
/etc/docker/certs.d/172.27.132.67\:8000/ca.crt
https://172.27.132.67:8000/v2/prom/node-
exporter/manifests/sha256:68effe31a4ae8312e47f54bec52d1fc925908009ce7e6f734e1b54a4169
081c5
$
```

删除 layer

向 `/v2/<name>/blobs/<digest>`发送 DELETE 请求，其中 digest 是上一步返回的 `fsLayers.blobSum` 字段内容：

```
$ curl -X DELETE  --user foo:foo123 --cacert
/etc/docker/certs.d/172.27.132.67\:8000/ca.crt
https://172.27.132.67:8000/v2/prom/node-
exporter/blobs/sha256:a3ed95caeb02ffe68cdd9fd84406680ae93d633cb16422d00e8a7c22955b46d
4
$ curl -X DELETE  --cacert /etc/docker/certs.d/172.27.132.67\:8000/ca.crt
https://172.27.132.67:8000/v2/prom/node-
exporter/blobs/sha256:04176c8b224aa0eb9942af765f66dae866f436e75acef028fe44b8a98e04551
5
$
```

常见问题

login 失败 416

执行 http://docs.ceph.com/docs/master/install/install-ceph-gateway/ 里面的 s3 test.py 程序失败：

[k8s@zhangjun-k8s-01 cert]$ python s3test.py
Traceback (most recent call last):
  File "s3test.py", line 12, in
    bucket = conn.create_bucket('my-new-bucket')
  File "/usr/lib/python2.7/site-packages/boto/s3/connection.py", line 625, in create_bucket
    response.status, response.reason, body)
boto.exception.S3ResponseError: S3ResponseError: 416 Requested Range Not Satisfiable

解决版办法：

1. 在管理节点上修改 ceph.conf
2. ceph-deploy config push zhangjun-k8s-01 zhangjun-k8s-02 zhangjun-k8s-03
3. systemctl restart 'ceph-mds@zhangjun-k8s-03.service'
   systemctl restart ceph-osd@0
   systemctl restart 'ceph-mon@zhangjun-k8s-01.service'
   systemctl restart 'ceph-mgr@zhangjun-k8s-01.service'

For anyone who is hitting this issue
set default pg_num and pgp_num to lower value(8 for example), or set mon_max_pg_per_osd to a high value in ceph.conf
radosgw-admin doesn' throw proper error when internal pool creation fails, hence the upper level error which is very confusing.

https://tracker.ceph.com/issues/21497

login 失败 503

[root@zhangjun-k8s-01 ~]# docker login 172.27.132.67:8000
Username: foo
Password:
Error response from daemon: login attempt to https://172.27.132.67:8000/v2/ failed with status: 503 Service Unavailable

原因： docker run 缺少 --privileged 参数；