**方法1：利用Node上的配置訪問Private Registry**

在玩Docker時，很多朋友都[搭建過自己的Private Registry](http://tonybai.com/2016/02/26/deploy-a-private-docker-registry/)。Docker訪問那些以basic auth方式進行鑑權的Private Registry，只需在本地執行docker login，輸入用戶名、密碼後，就可以自由向Registry Push鏡像或pull鏡像到本地了：

# docker login registry.cn-hangzhou.aliyuncs.com/xxxx/rbd-rest-api Username: {UserName} Password: Login Succeeded

在這一過程結束後，Docker實際上會在~/.docker目錄下創建一個config.json文件，保存後續與Registry交互過程中所要使用的鑑權串（這個鑑權串只是一個base64編碼結果，安全性欠佳^\_^）：

# cat ~/.docker/config.json { "auths": { "registry.cn-hangzhou.aliyuncs.com/xxxx/rbd-rest-api": { "auth": "xxxxyyyyzzzz" } } }

一但Node上有了這個配置，那麼K8s就可以通過docker直接訪問Private Registry了，這是[K8s文檔中與私有鏡像倉庫交互的第一個方法](http://kubernetes.io/docs/user-guide/images/#using-a-private-registry)。考慮到Pod可以被調度到集群中的任意一個Node上，需要在每個Node上執行上述login操作，或者可以簡單地將~/.docker/config.json scp到各個node上的~/.docker目錄下。

實際效果如何呢? 我們創建了一個Pod yaml，測試一下是否能run起來：

//rbd-rest-api-using-node-config.yaml apiVersion: v1 kind: Pod metadata: name: rbd-rest-api-using-node-config spec: containers: - name: rbd-rest-api-using-node-config image: registry.cn-hangzhou.aliyuncs.com/xxxx/rbd-rest-api:latest imagePullPolicy: Always

我們來創建一下這個Pod並查看pod的創建狀態：

# kubectl create -f rbd-rest-api-using-node-config.yaml pod "rbd-rest-api-using-node-config" created # kubectl get pods NAME READY STATUS RESTARTS AGE rbd-rest-api-using-node-config 0/1 ErrImagePull 0 5s

通過describe查看Pod失敗的詳細信息：

# kubectl describe pod/rbd-rest-api-using-node-config ... ... Events: FirstSeen LastSeen Count From SubobjectPath Type Reason Message --------- -------- ----- ---- ------------- -------- ------ ------- 1m 1m 1 {default-scheduler } Normal Scheduled Successfully assigned rbd-rest-api-using-node-config to 10.66.181.146 1m 42s 3 {kubelet 10.66.181.146} spec.containers{rbd-rest-api-using-node-config} Normal Pulling pulling image "registry.cn-hangzhou.aliyuncs.com/xxxx/rbd-rest-api:latest" 1m 42s 3 {kubelet 10.66.181.146} spec.containers{rbd-rest-api-using-node-config} Warning Failed Failed to pull image "registry.cn-hangzhou.aliyuncs.com/xxxx/rbd-rest-api:latest": image pull failed for registry.cn-hangzhou.aliyuncs.com/xxxx/rbd-rest-api:latest, this may be because there are no credentials on this request. details: (Error: image xxxx/rbd-rest-api:latest not found) 1m 42s 3 {kubelet 10.66.181.146} Warning FailedSync Error syncing pod, skipping: failed to "StartContainer" for "rbd-rest-api-using-node-config" with ErrImagePull: "image pull failed for registry.cn-hangzhou.aliyuncs.com/xxxx/rbd-rest-api:latest, this may be because there are no credentials on this request. details: (Error: image xxxx/rbd-rest-api:latest not found)" ... ...

這個方法對我們的環境並不有效

#### 方法2：通過kubectl創建docker-registry的secret

K8s提供的第二種方法是通過kubectl創建一個docker-registry的secret，並在Pod描述文件中引用該secret以達到從Private Registry Pull Image的目的。

操作之前，我們先刪除掉各個Node上的~/.docker/config.json。

執行kubectl create secret docker-registry時需要提供private registry的訪問UserName和Password：

# kubectl create secret docker-registry registrykey-m2-1 --docker-server=registry.cn-hangzhou.aliyuncs.com/xxxx/rbd-rest-api --docker-username={UserName} --docker-password={Password} --docker-email=team@domain.com secret "registrykey-m2-1" created # kubectl get secret NAME TYPE DATA AGE registrykey-m2-1 kubernetes.io/dockercfg 1 29s

secret: registrykey-m2-1創建成功。我們來測試一下引用這個secret對象的Pod是否能Pull Image成功並Run起來。Pod yaml文件如下：

//rbd-rest-api-registrykey-m2-1.yaml

apiVersion: v1

kind: Pod

metadata: name: rbd-rest-api-registrykey-m2-1

spec:

containers: -

name: rbd-rest-api-registrykey-m2-1

image: registry.cn-hangzhou.aliyuncs.com/xxxx/rbd-rest-api:latest imagePullPolicy: Always

imagePullSecrets:

- name: registrykey-m2-1

創建Pod，並觀察Pod狀態

# kubectl create -f rbd-rest-api-registrykey-m2-1.yaml pod "rbd-rest-api-registrykey-m2-1" created # kubectl get pods NAME READY STATUS RESTARTS AGE rbd-rest-api-registrykey-m2-1 1/1 Running 0 7s rbd-rest-api-using-node-config 0/1 ImagePullBackOff 0 29m

通過describe pod，查看創建的event序列：

Events: FirstSeen LastSeen Count From SubobjectPath Type Reason Message --------- -------- ----- ---- ------------- -------- ------ ------- 1m 1m 1 {default-scheduler } Normal Scheduled Successfully assigned rbd-rest-api-registrykey-m2-1 to 10.57.136.60 1m 1m 1 {kubelet 10.57.136.60} spec.containers{rbd-rest-api-registrykey-m2-1} Normal Pulling pulling image "registry.cn-hangzhou.aliyuncs.com/xxxx/rbd-rest-api:latest" 1m 1m 1 {kubelet 10.57.136.60} spec.containers{rbd-rest-api-registrykey-m2-1} Normal Pulled Successfully pulled image "registry.cn-hangzhou.aliyuncs.com/xxxx/rbd-rest-api:latest" 1m 1m 1 {kubelet 10.57.136.60} spec.containers{rbd-rest-api-registrykey-m2-1} Normal Created Created container with docker id d842565e762d 1m 1m 1 {kubelet 10.57.136.60} spec.containers{rbd-rest-api-registrykey-m2-1} Normal Started Started container with docker id d842565e762d

正如我們期望的那樣，引用了secret: registrykey-m2-1的Pod成功Run起來了。

如果一個pod中有來自不同私有倉庫的不同鏡像，我們需要怎麼做呢？通過kubectl create secret docker-registry我們一次只能建立一個registrykey，如果要訪問兩個鏡像倉庫，我們就需要分別為每個倉庫創建一個registrykey。我們再來創建一個registrykey，對應的倉庫為：registry.cn-hangzhou.aliyuncs.com/xxxx/test：

# kubectl create secret docker-registry registrykey-m2-2 --docker-server=registry.cn-hangzhou.aliyuncs.com/xxxx/test --docker-username={UserName} --docker-password={Password} --docker-email=team@domain.com secret "registrykey-m2-2" created root@node1:~/pullimagetest/test# kubectl get secret NAME TYPE DATA AGE registrykey-m2-1 kubernetes.io/dockercfg 1 1h registrykey-m2-2 kubernetes.io/dockercfg 1 6s

接下來，我們來建一個包含多個container的Pod：

//rbd-rest-api-multi-registrykeys-m2-2.yaml

apiVersion: v1

kind: Pod

metadata:

name: rbd-rest-api-multi-registrykeys-m2-2

spec:

containers:

- name: rbd-rest-api-multi-registrykeys-m2-2

image: registry.cn-hangzhou.aliyuncs.com/xxxx/rbd-rest-api:latest imagePullPolicy: Always

- name: test-multi-registrykeys-m2-2

image: registry.cn-hangzhou.aliyuncs.com/xxxx/test:latest

imagePullPolicy: Always

command: - "tail" - "-f" - "/var/log/bootstrap.log"

imagePullSecrets:

- name: registrykey-m2-1

- name: registrykey-m2-2

在secret引用中，我們將兩個key都引用了進來。

創建該Pod：

# kubectl create -f rbd-rest-api-multi-registrykeys-m2-2.yaml pod "rbd-rest-api-multi-registrykeys-m2-2" created # kubectl get pod NAME READY STATUS RESTARTS AGE rbd-rest-api-multi-registrykeys-m2-2 2/2 Running 0 5s

k8s分別從兩個鏡像倉庫嘗試pull image，並且最終都成功了！

#### 方法3：通過secret yaml文件創建pull image所用的secret

除了上面通過kubectl可以快捷的創建pull image所用的secret外，我們還可以使用常規的手段-yaml描述文件來創建我們需要的secret資源。

//registrykey-m3-1.yaml

apiVersion: v1

kind: Secret

metadata:

name: registrykey-m3-1

namespace: default

data: .dockerconfigjson: {base64 -w 0 ~/.docker/config.json}

type: kubernetes.io/dockerconfigjson

前面說過docker login會在~/.docker下面創建一個config.json文件保存鑑權串，這裡secret yaml的.dockerconfigjson後面的數據就是那個json文件的base64編碼輸出（-w 0讓base64輸出在單行上，避免折行）。

# kubectl create -f registrykey-m3-1.yaml secret "registrykey-m3-1" created # kubectl get secret NAME TYPE DATA AGE

myregistrykey3 kubernetes.io/dockerconfigjson 1 3h registrykey-m2-1 kubernetes.io/dockercfg 1 1h registrykey-m2-2 kubernetes.io/dockercfg 1 23m registrykey-m3-1 kubernetes.io/dockerconfigjson 1 29s

對比後，我們發現通過kubectl和yaml創建的兩個registrykey secret的類型略有不同，前者是kubernetes.io/dockercfg，後者是kubernetes.io/dockerconfigjson

接下來，我們編寫一個引用了registrykey-m3-1的Pod：

//rbd-rest-api-registrykey-m3-1.yaml

apiVersion: v1 kind: Pod

metadata: name: rbd-rest-api-registrykey-m3-1

spec:

containers:

- name: rbd-rest-api-registrykey-m3-1

image: registry.cn-hangzhou.aliyuncs.com/xxxx/rbd-rest-api:latest imagePullPolicy: Always

imagePullSecrets:

- name: registrykey-m3-1

創建成功。

那麼這種方法如何應對含有來自多個鏡像倉庫container的Pod的呢？這裡的思路與方法2略有不同。我們不需要創建並引用兩個或多個secret，而是創建一個可以訪問多個私有鏡像倉庫的secret，我們需要將多個鏡像倉庫的訪問鑑權串都放到~/.docker/config.json中：

按照方法1的介紹，我們先login registry.cn-hangzhou.aliyuncs.com/xxxx/rbd-rest-api，得到config.json如下：

{ "auths": { "registry.cn-hangzhou.aliyuncs.com/xxxx/rbd-rest-api": { "auth": "....省略...." } } }

我們再login registry.cn-hangzhou.aliyuncs.com/xxxx/test，得到config.json如下

{ "auths": { "registry.cn-hangzhou.aliyuncs.com/xxxx/rbd-rest-api": { "auth": "....省略...." }, "registry.cn-hangzhou.aliyuncs.com/xxxx/test": { "auth": "....省略...." } } }

我們看到Docker自動將新login的private registry的鑑權串merge到了同一個config.json中了。現在我們基於該包含了兩個庫鑑權串的config.json創建一個新secret：registrykey-m3-2：

//registrykey-m3-2.yaml

apiVersion: v1

kind: Secret

metadata:

name: registrykey-m3-2

namespace: default

data: .dockerconfigjson: {base64 -w 0 ~/.docker/config.json} type: kubernetes.io/dockerconfigjson

創建secret: registrykey-m3-2

# kubectl create -f registrykey-m3-2.yaml secret "registrykey-m3-2" created # kubectl get secrets NAME TYPE DATA AGE registrykey-m2-1 kubernetes.io/dockercfg 1 1h registrykey-m2-2 kubernetes.io/dockercfg 1 42m registrykey-m3-1 kubernetes.io/dockerconfigjson 1 19m registrykey-m3-2 kubernetes.io/dockerconfigjson 1 6s

我們編輯一個包含兩個容器，引用secret “registrykey-m3-2″ 的Pod yaml：

//rbd-rest-api-multi-registrykeys-m3-2.yaml

apiVersion: v1

kind: Pod

metadata: name: rbd-rest-api-multi-registrykeys-m3-2

spec:

containers:

- name: rbd-rest-api-multi-registrykeys-m3-2

image: registry.cn-hangzhou.aliyuncs.com/xxxx/rbd-rest-api:latest imagePullPolicy: Always

- name: test-multi-registrykeys-m3-2

image: registry.cn-hangzhou.aliyuncs.com/xxxx/test:latest

imagePullPolicy: Always

command:

- "tail" - "-f" - "/var/log/bootstrap.log"

imagePullSecrets:

- name: registrykey-m3-2

創建該Pod：

# kubectl create -f rbd-rest-api-multi-registrykeys-m3-2.yaml pod "rbd-rest-api-multi-registrykeys-m3-2" created # kubectl get pod NAME READY STATUS RESTARTS AGE rbd-rest-api-multi-registrykeys-m3-2 2/2 Running 0 4s

#### 調用API創建registrykey secret

對比了方法2和方法3，方法2更簡潔，方法3更強大。但在任何一個產品中，secret都不應該是手動創建的，在這種情況下，[API](http://kubernetes.io/docs/api-reference/v1/operations/)創建[registrykey secret](http://kubernetes.io/docs/api-reference/v1/definitions/#_v1_secret)便是必經之路。一旦選擇通過API創建，我們顯然將依仗著方法2中的原理，將config.json中的內容通過API請求的Body Post給K8s api server

如何在遠端構建出config.json的內容呢繼而構建出secret yaml中.dockerconfigjson的值數據呢？我們發現config.json套路中，唯一不確定的就是每個private repository下的auth串，那麼這個串是啥呢？你大可base64 -d一下：

# echo -n "VXNlck5hbWU6UGFzc3dvcmQ="|base64 –d

UserName:Password

沒錯，實質上這個auth串就是UserName:Password的base64編碼值。因此，你首先要用某個倉庫的UserName和Password按照'UserName:Password'格式進行base64編碼，利用編碼的結果值構造json內容，比如：

{ "auths": { "registry.cn-hangzhou.aliyuncs.com/xxxx/rbd-rest-api": { "auth": "VXNlck5hbWU6UGFzc3dvcmQ=" } }

然後對這段json數據再做base64編碼，所得到的值就是secret yaml中的.dockerconfigjson的值數據。至此，我們來通過API創建一個secret：

$ curl -v -H "Content-type: application/json" -X POST -d ' { "apiVersion": "v1", "kind": "Secret", "metadata": { "name": "registrykey-m4-1", "namespace": "default" }, "data": { ".dockerconfigjson": "{cat ~/.docker/config.json |base64 -w 0}" }, "type": "kubernetes.io/dockerconfigjson" }' <http://10.57.136.60:8080/api/v1/namespaces/default/secrets>

# kubectl get secret NAME TYPE DATA AGE registrykey-m2-1 kubernetes.io/dockercfg 1 2h registrykey-m2-2 kubernetes.io/dockercfg 1 1h registrykey-m3-1 kubernetes.io/dockerconfigjson 1 43m registrykey-m3-2 kubernetes.io/dockerconfigjson 1 24m

基於registrykey-m4-1，我們啟動一個Pod：

//rbd-rest-api-registrykey-m4-1.yaml

apiVersion: v1

kind: Pod

metadata: name: rbd-rest-api-registrykey-m4-1

spec:

containers: - name: rbd-rest-api-registrykey-m4-1

image: registry.cn-hangzhou.aliyuncs.com/xxxx/rbd-rest-api:latest imagePullPolicy: Always

imagePullSecrets:

- name: registrykey-m4-1

# kubectl create -f rbd-rest-api-registrykey-m4-1.yaml pod "rbd-rest-api-registrykey-m4-1" created # kubectl get pod NAME READY STATUS RESTARTS AGE rbd-rest-api-registrykey-m4-1 1/1 Running 0 5s