Step 1) Generate self-signed certificates for private registry

Login to your control plane or master node and use openssl command to generate self-signed certificates for private docker repository.

$ cd /opt

$ sudo openssl req -newkey rsa:4096 -nodes -sha256 -keyout ./certs/registry.key -addext "subjectAltName = DNS:master-node-k8" -x509 -days 365 -out ./certs/registry.crt

Once the key and certificate file are generated, use [ls command](https://www.linuxtechi.com/linux-ls-command-examples-beginners/) to verify them,

[kadmin@k8s-master opt]$ ls -l certs/

total 8

-rw-r--r--. 1 nobody nobody 2175 Mar 21 08:06 registry.crt

-rw-------. 1 nobody nobody 3272 Mar 21 08:05 registry.key

[kadmin@k8s-master opt]$

Step 2) Deploy private registry as deployment via yaml file

On your master node, create a private-registry.yaml file with the following contents

[kadmin@k8s-master ~]$ mkdir docker-repo

[kadmin@k8s-master ~]$ cd docker-repo/

[kadmin@k8s-master docker-repo]$ vi private-registry.yaml

apiVersion: apps/v1

kind: Deployment

metadata:

  name: private-repository-k8s

  labels:

    app: private-repository-k8s

spec:

  replicas: 1

  selector:

    matchLabels:

      app: private-repository-k8s

  template:

    metadata:

      labels:

        app: private-repository-k8s

    spec:

      volumes:

      - name: certs-vol

        hostPath:

          path: /opt/certs

          type: Directory

      - name: registry-vol

        hostPath:

          path: /opt/registry

          type: Directory

      containers:

        - image: registry:2

          name: private-repository-k8s

          imagePullPolicy: IfNotPresent

          env:

          - name: REGISTRY\_HTTP\_TLS\_CERTIFICATE

            value: "/certs/registry.crt"

          - name: REGISTRY\_HTTP\_TLS\_KEY

            value: "/certs/registry.key"

          ports:

            - containerPort: 5000

          volumeMounts:

          - name: certs-vol

            mountPath: /certs

          - name: registry-vol

            mountPath: /var/lib/registry

save and close the yaml file

Run the following kubectl command deploy the private registry using above created yaml file,

[kadmin@k8s-master docker-repo]$ kubectl create -f private-registry.yaml

deployment.apps/private-repository-k8s created

[kadmin@k8s-master docker-repo]$

Execute below kubectl commands to verify status of registry deployment and its pod.

[kadmin@k8s-master ~]$ kubectl get deployments private-repository-k8s

NAME                     READY   UP-TO-DATE   AVAILABLE   AGE

private-repository-k8s   1/1     1            1           3m32s

[kadmin@k8s-master ~]$

[kadmin@k8s-master ~]$ kubectl get pods | grep -i private-repo

private-repository-k8s-85cf76b9d7-qsjxq   1/1     Running   0          5m14s

[kadmin@k8s-master ~]$

Perfect, above output confirms that registry has been deployed successfully, Now copy the registry certificate file to worker nodes and master node under the folder “/etc/pki/ca-trust/source/anchors“. Execute the following commands on master node and each worker nodes

$ sudo cp /opt/certs/registry.crt /etc/pki/ca-trust/source/anchors/

$ sudo update-ca-trust

Step 3) Expose registry deployment as a nodeport service type

To expose registry deployment as a nodeport service type, create the following yaml file with the beneath contents,

[kadmin@k8s-master ~]$ cd docker-repo/

[kadmin@k8s-master docker-repo]$ vi private-registry-svc.yaml

apiVersion: v1

kind: Service

metadata:

  labels:

    app: private-repository-k8s

  name: private-repository-k8s

spec:

  ports:

  - port: 5000

    nodePort: 31320

    protocol: TCP

    targetPort: 5000

  selector:

    app: private-repository-k8s

  type: NodePort

save and close the file.

Now deploy the service by running following kubectl command,

$ kubectl create -f private-registry-svc.yaml

service/private-repository-k8s created

$

Run below kubectl command to verify the service status,

[kadmin@k8s-master ~]$ kubectl get svc private-repository-k8s

NAME                   TYPE     CLUSTER-IP    EXTERNAL-IP PORT(S)        AGE

private-repository-k8s NodePort 10.100.113.39 <none>      5000:31320/TCP 2m1s

[kadmin@k8s-master ~]$

Step 4) Test and Use private docker registry in k8s

To test private registry, we will download nginx image locally and then will upload that image to private registry, from the master node run the following set of commands,

$ sudo docker pull nginx

$ sudo docker tag nginx:latest k8s-master:31320/nginx:1.17

$ sudo docker push k8s-master:31320/nginx:1.17

Output of above command would like below:

Run below docker command to verify whether nginx is uploaded to private repository or not.

[kadmin@k8s-master ~]$ sudo docker image ls | grep -i nginx

nginx                     latest   7e4d58f0e5f3        2 weeks ago      133MB

k8s-master:31320/nginx    1.17     7e4d58f0e5f3        2 weeks ago      133MB

[kadmin@k8s-master ~]$

Now, let’s deploy a nginx based deployment and in the yaml file specify the image’s path as our private docker registry. Example is shown below:

[kadmin@k8s-master ~]$ vi nginx-test-deployment.yaml

apiVersion: apps/v1

kind: Deployment

metadata:

  name: nginx-test-deployment

  labels:

    app: nginx

spec:

  replicas: 3

  selector:

    matchLabels:

      app: nginx

  template:

    metadata:

      labels:

        app: nginx

    spec:

      containers:

      - name: nginx-1-17

        image: k8s-master:31320/nginx:1.17

        ports:

        - containerPort: 80

Save and Close the file

Run following kubectl commands,

[kadmin@k8s-master ~]$ kubectl create -f nginx-test-deployment.yaml

deployment.apps/nginx-test-deployment created

[kadmin@k8s-master ~]$ kubectl get deployments  nginx-test-deployment

NAME                    READY   UP-TO-DATE   AVAILABLE   AGE

nginx-test-deployment   3/3     3            3           13s

[kadmin@k8s-master ~]$

[kadmin@k8s-master ~]$ kubectl get  pods | grep nginx-test-deployment

nginx-test-deployment-f488694b5-2rvmv     1/1     Running   0      80s

nginx-test-deployment-f488694b5-8kb6c     1/1     Running   0      80s

nginx-test-deployment-f488694b5-dgcxl     1/1     Running   0      80s

[kadmin@k8s-master ~]$

Try to describe any pod using ‘kubectl describe‘ command and verify image path

$ kubectl describe pod nginx-test-deployment-f488694b5-2rvmv

Output of above command would be,

Above output confirms that container’s image path is our private docker registry, so it means nginx image has been downloaded from private registry. That’s all from this article, I hope these steps help you to setup private docker registry on your Kubernetes cluster. Please do share your feedback and comments in the comments section below.

**Also Read** : [**How to Setup Kubernetes Cluster on Google Cloud Platform (GCP)**](https://www.linuxtechi.com/setup-kubernetes-cluster-google-cloud-platform-gcp/)

**Also Read** : [**How to Setup NGINX Ingress Controller in Kubernetes**](https://www.linuxtechi.com/setup-nginx-ingress-controller-in-kubernetes/)

About The Author

[Pradeep Kumar](https://www.linuxtechi.com/author/pradeep/" \t "_self)

I am a Cloud Consultant with over 15 years of experience in Linux, Kubernetes, cloud technologies (AWS, Azure, OpenStack), automation (Ansible, Terraform), and DevOps. I hold certifications like RHCA, CKA, CKAD, CKS, AWS, and Azure.