Docker Image Deployment and Kubernetes Configuration

Your Name

November 2, 2024

Instructions

Prerequisites: Docker, Kubernetes, BusyBox, and OpenSSL installed, along with a working directory containing a Dockerfile.

1. **Run a Local Docker Registry (without authentication):**

```
docker run -d -p 5000:5000 --name registry
    registry:2
```

   This command starts a registry container on `localhost:5000`.

2. **Optional: Enable Authentication for the Local Registry**

   If you want to secure your local registry with a username and password, follow these steps:

   (a) **Create a User and Password Hash with BusyBox:** Generate a hashed password for ‘passpass‘ and store it in the ‘htpasswd‘ file:

```
mkdir -p /home/user/auth
echo "user:$(busybox␣mkpasswd␣-m␣sha256␣-S␣$(
    openssl␣rand␣-hex␣6)␣passpass)" > /home/user
    /auth/htpasswd
```

   This command generates a hashed password ‘passpass‘ for the username ‘user‘.

   (b) **Run the Registry with Authentication:** Restart the registry container to require authentication.

```
docker stop registry
docker rm registry
docker run -d -p 5000:5000 --name registry \
  -v /home/user/auth:/auth \
  -e "REGISTRY_AUTH=htpasswd" \
```

```
  -e "REGISTRY_AUTH_HTPASSWD_REALM=Registry␣
     Realm" \
  -e "REGISTRY_AUTH_HTPASSWD_PATH=/auth/
     htpasswd" \
  registry:2
```

3. **Build and Push Docker Image to Local Registry:**

```
docker build -f Dockerfile -t localhost:5000/my-
   app:latest ..
docker push localhost:5000/my-app:latest
```

4. **Configure Kubernetes to Use the Local Registry (if using authentication):** Create a Kubernetes secret with credentials to access the registry:

```
kubectl create secret docker-registry regcred \
  --docker-server=localhost:5000 \
  --docker-username=user \
  --docker-password=passpass \
  --docker-email=user@example.com
```

   This secret can be referenced by Kubernetes to pull the image from the secured local registry.

5. **Deploy a Pod Using the Image from the Local Registry:**

```
kubectl run my-app --image=localhost:5000/my-app:
   latest --image-pull-policy=IfNotPresent
```

1 Adding a Docker Image to Minikube and Exposing It

1.1 Load Docker Image into Minikube

1. **Build and Load Docker Image Directly into Minikube:** If the image is built locally, you can load it directly into Minikube:

```
   minikube image load my-app:latest
```

Alternatively, if you are using a local registry, ensure Minikube can access it by setting the registry address in the image tag.

1.2 Create and Run a Pod with the Loaded Image

1. **Create a Deployment in Minikube:** Create a Kubernetes Deployment to manage the pod using the loaded image:

```
kubectl create deployment my-app --image=my-app:
    latest
```

2. **Expose the Deployment as a Service:** Expose the Deployment to make the application accessible on an external port:

```
kubectl expose deployment my-app --type=NodePort --
    port=80 --target-port=8080
```

In this example, Kubernetes maps port 80 of the service to port 8080 of the container. Modify 'target-port' if your application uses a different port.

1.3 Access the Application Externally

1. **Retrieve the Minikube Service URL:** Minikube provides a command to get the external URL for accessing services. Run the following command to get the URL:

```
minikube service my-app --url
```

This command will output a URL that you can use to access the service from outside of Minikube.