ASHOK IT

Learn Here.. Lead Anywhere..!!

DevOps Project Setup

By Mr. Ashok

(Backend)

spring boot

(Frontend)

ANGULAR

docker

(Containerization)

kubernetes

(Orchestration)

Spring Boot + Angular + Docker + Kubernetes – Project Setup

-> In this tutorial we will deploy one Fullstack Project (Spring Boot + Angular) as Docker Containers in Kubernetes Cluster

-> Here we will complete our setup in 3 steps

1) **Environment Setup**
2) **Backend Deployment (Spring Boot Application)**
3) **Frontend Deployment (Angular Application)**

Step - 1. Environment Setup

1.1) Setup Kubernetes Cluster

1.2) Launch Ubuntu VM in AWS Cloud

1.3) Connect to Ubuntu VM using MobaXterm

1.4) Install Docker in Ubuntu VM using below commands

```
$ curl -fsSL get.docker.com | /bin/bash

$ sudo usermod -aG docker $USER

$ newgrp docker

$ docker info
```

1.5) Install Maven in Ubuntu VM using below command

```
$ sudo apt install maven
```

1.6) Install Git client in Ubuntu VM using below command

```
$ sudo apt install git
```

1.7) Install Node and Angular CLI in Ubuntu VM using below commands

```
$ curl https://raw.githubusercontent.com/creationix/nvm/master/install.sh | bash

$ source ~/.bashrc

$ nvm install node

$ node version

$ npm install -g @angular/cli

$ ng v
```

Note: With this we have completed environment setup to start our Build and Deployment.

Step - 2. Backend Application Deployment

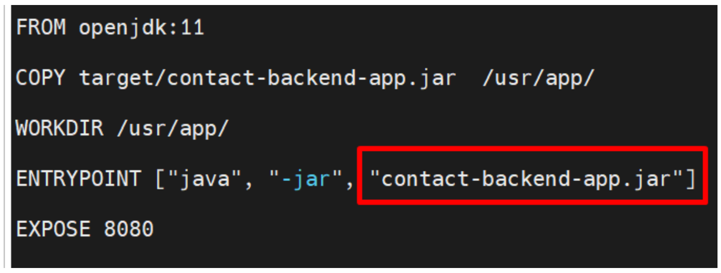
2.1) Clone Backend Application using git clone

```
$ git clone <repo-URL>
```

2.2) Perform Maven Build for backend application

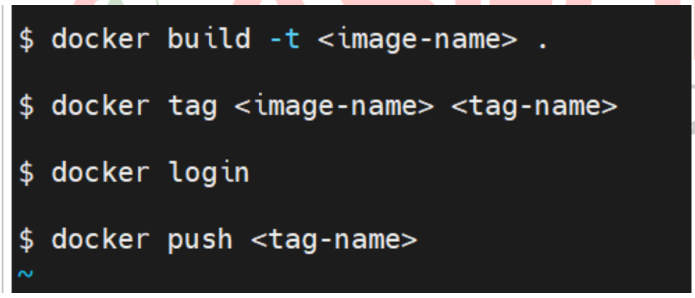
```
$ cd <project-directory>
$ mvn clean package
```

2.3) Write Dockerfile for backend application



```
FROM openjdk:11
COPY target/contact-backend-app.jar  /usr/app/
WORKDIR /usr/app/
ENTRYPOINT ["java", "-jar", "contact-backend-app.jar"]
EXPOSE 8080
```

2.4) Create Docker image for backend application using below commands



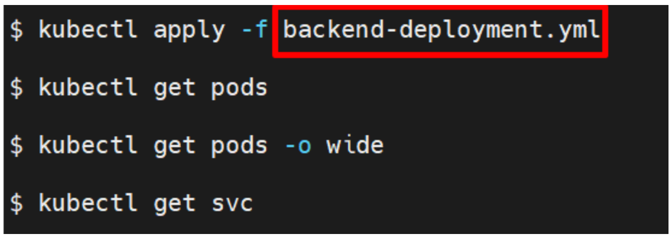
```
$ docker build -t <image-name> .
$ docker tag <image-name> <tag-name>
$ docker login
$ docker push <tag-name>
~
```

2.5) Connect to Kubernetes Cluster Control Plane

2.6) Create Deployment Manifest file for backend application like below

```
---
apiVersion: apps/v1
kind: Deployment
metadata:
 name: contactbackendappdeployment
spec:
 replicas: 2
 selector:
   matchLabels:
    app: contactbackend
 template:
  metadata:
   name: contactbackend
   labels:
    app: contactbackend
  spec:
   containers:
   - name: contactbackendcontainer
     image: <image-name>
     ports:
     - containerPort: 8080
---
apiVersion: v1
kind: Service
metadata:
 name: contactbackendsvc
spec:
  type: NodePort
  selector:
   app: contactbackend
  ports:
   - port: 80
    targetPort: 8080
    nodePort: 30001
...
```

2.7) Deploy backend application on Kubernetes Cluster



```
$ kubectl apply -f backend-deployment.yml

$ kubectl get pods

$ kubectl get pods -o wide

$ kubectl get svc
```

2.8) Access Backend application using URL

URL : http://node-port:nodeip/

Step - 3) Frontend Application Deployment

3.1) Install Node and Angular CLI in Ubuntu

$ curl https://raw.githubusercontent.com/creationix/nvm/master/install.sh | bash

$ source ~/.bashrc

$ nvm install node

$ node version

$ npm install -g @angular/cli

$ ng v

3.2) Clone Frontend application using git clone

$ git clone <repo-url>

3.3) Configure Backend Application URL in Frontend application

$ cd <project-directory>

$ cd src/app

$ vi contact.service.ts

Note: configure backend url in frontend application for integration

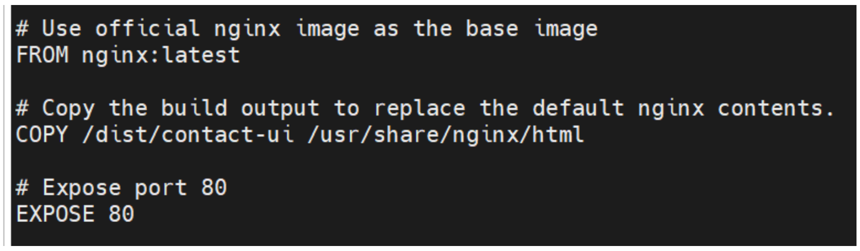
3.4) Build frontend application

```
$ ng build
```

Note: If you get a problem saying "could-not-find-the-implementation-for-builder-angular-devkit-build-angulardev" then execute below commands to fix the problem

```
$ npm install --save-dev @angular-devkit/build-angular
```

3.4) Create Dockerfile for Angular application



```
# Use official nginx image as the base image
FROM nginx:latest

# Copy the build output to replace the default nginx contents.
COPY /dist/contact-ui /usr/share/nginx/html

# Expose port 80
EXPOSE 80
```

3.5) Create docker image for frontend apliation

```
$ docker build -t contact_ui_ng_app .

$ docker tag contact_ui_ng_app ashokit/contact_ui_ng_app

$ docker login

$ docker push ashokit/contact_ui_ng_app
```

3.6) Create deployment manifest file for frontend application

```
---
apiVersion: apps/v1
kind: Deployment
metadata:
 name: contactfronendappdeployment
spec:
 replicas: 2
 selector:
  matchLabels:
   app: contactfronend
 template:
  metadata:
   name: contactfronend
  labels:
    app: contactfronend
  spec:
    containers:
    - name: contactfronendcontainer
      image: ashokit/contact_ui_ng_app
      ports:
      - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
 name: contactfronendsvc
spec:
  type: NodePort
  selector:
   app: contactfronend
  ports:
   - port: 80
     targetPort: 80
     nodePort: 30002
...
```

3.7) Deploy frontend application on Kubernetes and expose as Node Port

```
$ kubectl apply -f frontend-deployment.yml
$ kubectl get pods
$ kubectl get pods -o wide
$ kubectl get svc
```

3.8) Access Frontend Application using URL

URL : http://node-ip:nodeport/

=== Learn Here… Lead Anywhere…!!! ===