BLOG

Steps to Deploy Angular application on Kubernetes

Intakhab Ali

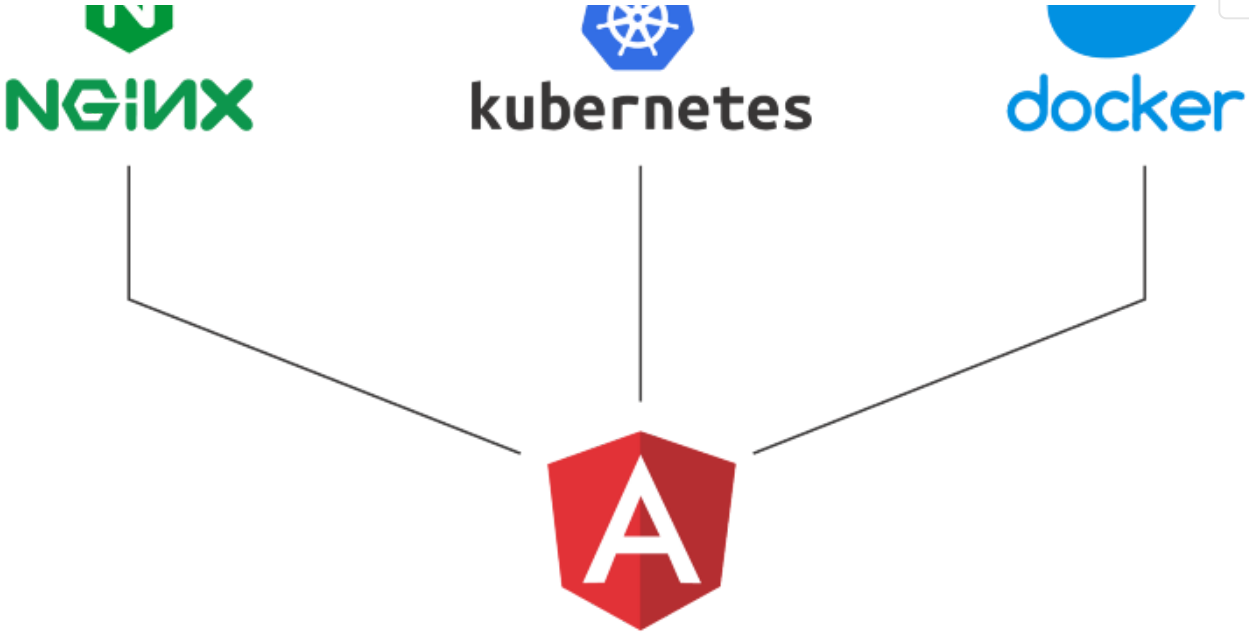
Jun 18 · 12 min read

I'm an engineer at MayaData, a company dedicated to enabling data agility through the use of Kubernetes as a data layer and the sponsor of the CNCF project OpenEBS and other open source projects. Take it for a test drive via free and easy to use management software by registering here.

Introduction

Angular is a JavaScript framework for building web applications and apps in JavaScript, HTML, and TypeScript, which is a superset of JavaScript. Angular provides built-in features for animation, HTTP service, and materials which in turn has features such as auto-complete, navigation, toolbar, menus, etc. The code is written in TypeScript, which compiles to JavaScript and displays the same in the browser.

In this tutorial, we will create a basic angular app. Write a docker file to build a compressed angular app and then create deployment manifest for angular application.



Steps to Deploy Angular application on Kubernetes

Prerequisite

Angular: A little knowledge of angular.

Nodejs: To run the application locally, we need node environment.

Docker: Docker CLI should be installed in your system to build and push the image. You can also set up a CI tool to build the docker image. I will talk about this in the next tutorial.

Nginx: Basic knowledge of Nginx configuration.

Kubernetes: Kubernetes is an orchestration tool, where we will deploy the application. For the demo sake, you can use minikube as well.

What we will do

1: Create an Angular application

2: Write custom Nginx config

3: Write a multistage docker file

3: Create a K8s deployment manifest and service manifest

Step 1: Create an Angular application

Now, let's create an Angular Application. By running the below command, angular will create and initialize a new Angular app, which we will use to deploy.

```
ng new spa-demo
```

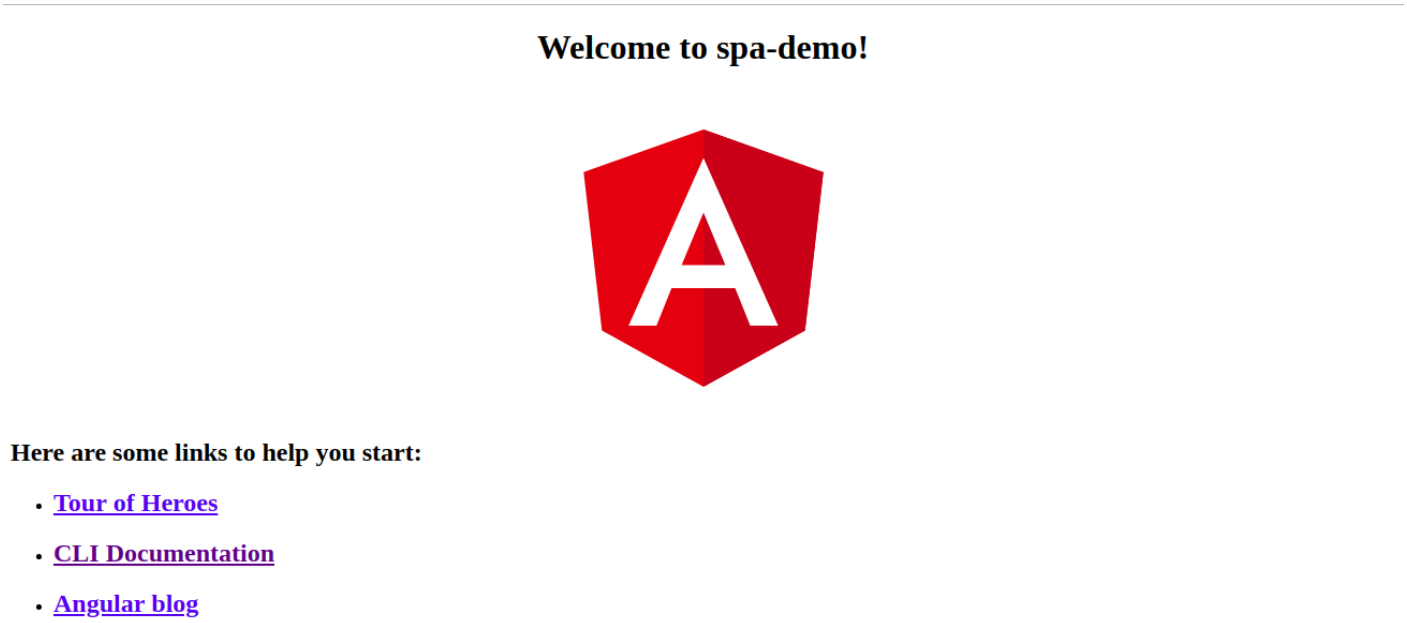
After completion of above command, go inside the directory.

```
cd spa-demo
```

Run the development server.

```
ng serve
```

Now, at visiting http://localhost:4200/, you will see the view of this Angular app.



View of Angular App

First, add an Nginx custom configuration file inside the new spa-demo directory, named nginx-custom.conf. Here is the gist link.

```
# Expires map
map $sent_http_content_type $expires {
    default                    off;
    text/html                  epoch;
    text/css                   max;
    application/json           max;
    application/javascript     max;
    ~image/                    max;
}

server {
  listen 80;
  location / {
      root /usr/share/nginx/html;
      index index.html index.htm;
      try_files $uri $uri/ /index.html =404;
    }
  expires $expires;
  gzip  on;
}
```

The above Nginx custom config contains:

- Expiration header for images and other content (CSS, HTML etc), which travels through the web to the browser for the maximum amount of time but do change it according to need.
- Every single page application uses its routing module to go to its route, but it needs to go through its home route, so we need to redirect every route to home route, then the single page application will take care of rest of the thing.

Step 3: Create a multistage docker file to build the angular application

Now, create a Dockerfile inside the spa-demo project directory, named- Dockerfile. Here is the gist link.

```
# Stage 0, "build-stage", based on Node.js, to build and compile the
FROM node:10.8.0 as build-stage
WORKDIR /app
COPY package*.json /app/
RUN npm install
COPY ./ /app/
ARG configuration=production
RUN npm run build -- --output-path=./dist/out --configuration $config

# Stage 1, based on Nginx, to have only the compiled app, ready for pr
FROM nginx:1.15
#Copy ci-dashboard-dist
COPY --from=build-stage /app/dist/out/ /usr/share/nginx/html
#Copy default nginx configuration
COPY ./nginx-custom.conf /etc/nginx/conf.d/default.conf
```

The above Dockerfile consists of two stages:

First stage: Create a node environment and build the angular application with production configuration.

Second stage: Copy the dist folder from the previous stage to Nginx container and copy nginx-custom.conf inside the nginx

Build and push the docker image

Docker build command

```
docker push inyee/spa-demo:v1
```

Docker push to docker registry.

```
docker push inyee/spa-demo:v1
```

Step 4: Create a K8s deployment manifest and service manifest

To deploy the Angular application in any of the Kubernetes environments, the deployment manifest for angular is listed below. Before deploying the application to production, make sure you modify the manifest file to best suit your needs. You can change the name of the Deployment and the labels, and change your Docker registry and image tag accordingly.

The deployment manifest gist link.

```
apiVersion: apps/v1beta1
kind: Deployment
metadata:
  name: deployment-name
spec:
  replicas: 1
  template:
    metadata:
      labels:
        label-key : label-value
    spec:
      containers:
        - name: deplоment-container-name
          image: inyee/spa-demo:v1
          imagePullPolicy: Always
          ports:
          - containerPort: 80
```

Create a normal service to access the application internally, or you can use this service in ingress to expose it to some domain, named *SPA-service.yaml*

```
apiVersion: v1
kind: Service
metadata:
  labels:
    service-label-key: service-label-value
  name: service-name
spec:
  type: ClusterIP
  ports:
  - name: service-port-name
    port: 80
    protocol: TCP
  selector:
    deployment-label-key: deployment-label-value
```

For the demo purpose, create a load balancer service file to access it outside the Kubernetes cluster. Make sure Lable selector is the same as the deployment label, named *SPA-load-balancer-service.yaml*

```
apiVersion: v1
kind: Service
metadata:
  labels:
    service-label-key: service-label-value
  name: service-name-loadbalancer
spec:
  type: LoadBalancer
  ports:
  - name: service-port-name
```

```
    protocol: TCP
  selector:
    deployment-label-key: deployment-label-value#for creating a deplo
kubectl apply -f spa-deployment.yaml
#for internal communicating to angualar application
kubeclt apply -f SPA-service.yaml
#for access the angular application outside kubernetes
kubeclt apply -f SPA-load-balancer-service.yaml
```

Run the command listed below to deploy the angular application in Kubernetes environment.

- Create a deployment in Kubernetes cluster

```
kubectl apply -f spa-deployment.yaml
```

- Create a ClusterIP service.

```
kubeclt apply -f SPA-service.yaml
```

- Create a load balancer service to access it via some External IP, provided by the service.

```
kubeclt apply -f SPA-load-balancer-service.yaml
kubectl get svc -owide
```

- Run the below command to get External IP of the of service.

```
NAME                                TYPE           CLUSTER-IP
service-name                        ClusterIP      xx.xx.xx.xx    <none>
service-name-loadbalancer                                         LoadBal
```

Go to the external IP on the browser, you will see the same angular app which we had created initially.

Angular app is ready for production

That's it! Now our Angular app is ready for production!

You can follow me at the below profiles and can ask any questions related to Angular, SPA, JavaScript, Kubernetes, etc.

- GitHub
- Linkedin
- Medium

Enter Your Email

Intakhab Ali
Software Engineer at MayaData

Managing Ephemeral Storage on Kubernetes with OpenEBS

Kiran Mova
Nov 10 · 12 min read

Why OpenEBS 3.0 for Kubernetes and Storage?

Kiran Mova
Sep 29 · 12 min read

Deploy PostgreSQL On Kubernetes Using OpenEBS LocalPV

Murat Karslioglu
Aug 20 · 12 min read

Enter Your Email

8/3/2019, 9:10:08 AM

Nice blog

Reply to *Rakesh Singh*

First Name*

Last Name

Email*

Website

Comment*

protected by **reCAPTCHA**

Privacy - Terms

Submit Comment

INTERESTED IN OUR NEWSLETTER?

Enter Your Email

Enter Your Email

PRODUCT

Kubera

OPEN SOURCE

OpenEBS

Litmus

Node Disk Manager

COMPANY

About Us

Careers

News

Partners

Blog

Contact Us

Privacy Policy

Terms & Conditions

Support