Deploy Your First Kubernetes Cluster:

Overview

1. KIND set-up
    1. Docker
    2. Kubectl
    3. KIND
2. Creating your cluster
3. Deploy an application
4. Expose the service
5. Set your own text

This tutorial shows you a step-by-step guide and will walk you through setting up a Kubernetes cluster on your own computer and deploy a simple application into the cluster. Throughout this tutorial we'll use kind because it's the fastest to set up with minimal dependencies, as long as you are able to run Docker on your machine.

Prerequisites:

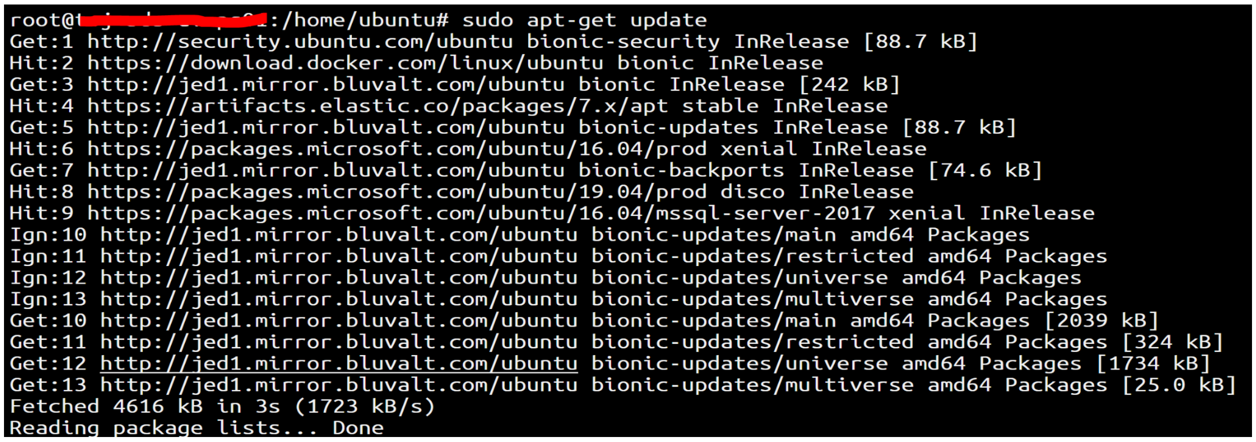
- 2 or more Linux servers running Ubuntu 18.04
- Access to a user account on each system with sudo or root privileges
- The apt package manager, included by default
- Command-line/terminal window (Ctrl-Alt-T)

1. Update Ubuntu dependencies

Update your system's dependencies to get ready for the Kubernetes installation:

```
# sudo apt-get update
```

Output

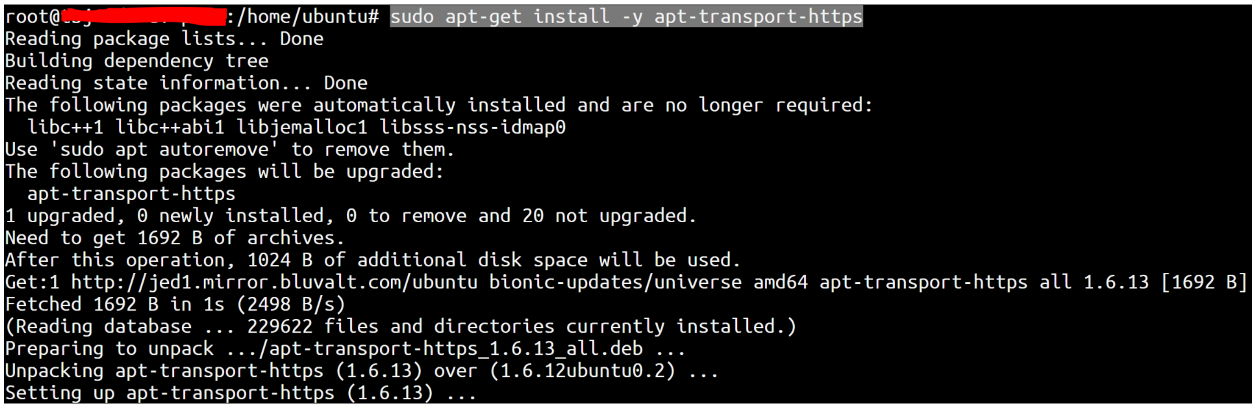


```
root@[redacted]:/home/ubuntu# sudo apt-get update
Get:1 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Hit:2 https://download.docker.com/linux/ubuntu bionic InRelease
Get:3 http://jed1.mirror.bluvalt.com/ubuntu bionic InRelease [242 kB]
Hit:4 https://artifacts.elastic.co/packages/7.x/apt stable InRelease
Get:5 http://jed1.mirror.bluvalt.com/ubuntu bionic-updates InRelease [88.7 kB]
Hit:6 https://packages.microsoft.com/ubuntu/16.04/prod xenial InRelease
Get:7 http://jed1.mirror.bluvalt.com/ubuntu bionic-backports InRelease [74.6 kB]
Hit:8 https://packages.microsoft.com/ubuntu/19.04/prod disco InRelease
Hit:9 https://packages.microsoft.com/ubuntu/16.04/mssql-server-2017 xenial InRelease
Ign:10 http://jed1.mirror.bluvalt.com/ubuntu bionic-updates/main amd64 Packages
Ign:11 http://jed1.mirror.bluvalt.com/ubuntu bionic-updates/restricted amd64 Packages
Ign:12 http://jed1.mirror.bluvalt.com/ubuntu bionic-updates/universe amd64 Packages
Ign:13 http://jed1.mirror.bluvalt.com/ubuntu bionic-updates/multiverse amd64 Packages
Get:10 http://jed1.mirror.bluvalt.com/ubuntu bionic-updates/main amd64 Packages [2039 kB]
Get:11 http://jed1.mirror.bluvalt.com/ubuntu bionic-updates/restricted amd64 Packages [324 kB]
Get:12 http://jed1.mirror.bluvalt.com/ubuntu bionic-updates/universe amd64 Packages [1734 kB]
Get:13 http://jed1.mirror.bluvalt.com/ubuntu bionic-updates/multiverse amd64 Packages [25.0 kB]
Fetched 4616 kB in 3s (1723 kB/s)
Reading package lists... Done
```

Deploy Your First Kubernetes Cluster:

# sudo apt-get install -y apt-transport-https

Output

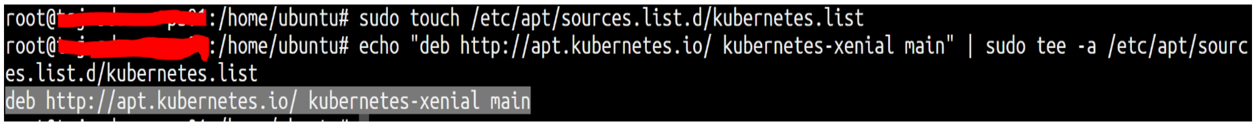


```
root@[redacted]:/home/ubuntu# sudo apt-get install -y apt-transport-https
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libc++1 libc++abi1 libjemalloc1 libsss-nss-idmap0
Use 'sudo apt autoremove' to remove them.
The following packages will be upgraded:
  apt-transport-https
1 upgraded, 0 newly installed, 0 to remove and 20 not upgraded.
Need to get 1692 B of archives.
After this operation, 1024 B of additional disk space will be used.
Get:1 http://jed1.mirror.bluvalt.com/ubuntu bionic-updates/universe amd64 apt-transport-https all 1.6.13 [1692 B]
Fetched 1692 B in 1s (2498 B/s)
(Reading database ... 229622 files and directories currently installed.)
Preparing to unpack .../apt-transport-https_1.6.13_all.deb ...
Unpacking apt-transport-https (1.6.13) over (1.6.12ubuntu0.2) ...
Setting up apt-transport-https (1.6.13) ...
```

2. Install kubectl, which you use to interact with the Kubernetes cluster:

# echo "deb http://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee -a /etc/apt/sources.list.d/kubernetes.list

Output:



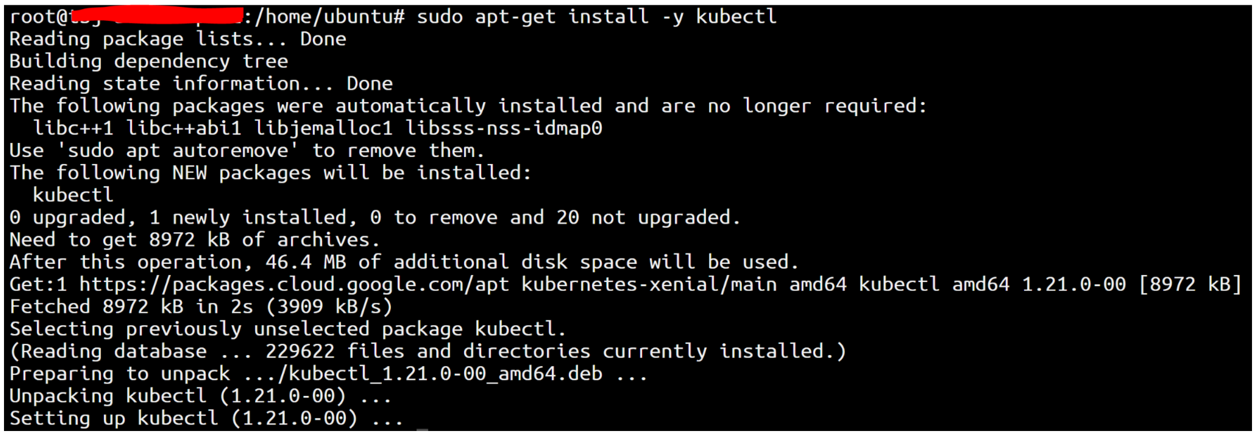
```
root@[redacted]:/home/ubuntu# sudo touch /etc/apt/sources.list.d/kubernetes.list
root@[redacted]:/home/ubuntu# echo "deb http://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee -a /etc/apt/sourc
es.list.d/kubernetes.list
deb http://apt.kubernetes.io/ kubernetes-xenial main
```

3. Update again and install Kubectl:

# sudo apt-get update

# sudo apt-get install -y kubectl

Output



```
root@[redacted]:/home/ubuntu# sudo apt-get install -y kubectl
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libc++1 libc++abi1 libjemalloc1 libsss-nss-idmap0
Use 'sudo apt autoremove' to remove them.
The following NEW packages will be installed:
  kubectl
0 upgraded, 1 newly installed, 0 to remove and 20 not upgraded.
Need to get 8972 kB of archives.
After this operation, 46.4 MB of additional disk space will be used.
Get:1 https://packages.cloud.google.com/apt kubernetes-xenial/main amd64 kubectl amd64 1.21.0-00 [8972 kB]
Fetched 8972 kB in 2s (3909 kB/s)
Selecting previously unselected package kubectl.
(Reading database ... 229622 files and directories currently installed.)
Preparing to unpack .../kubectl_1.21.0-00_amd64.deb ...
Unpacking kubectl (1.21.0-00) ...
Setting up kubectl (1.21.0-00) ...
```

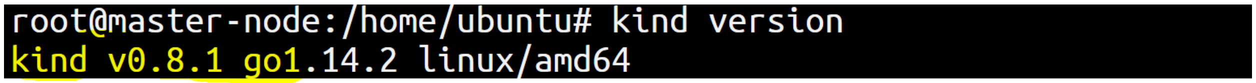
Deploy Your First Kubernetes Cluster:

4. Kind-setup:

To get kind working, you need to have Docker installed. To install **kind** command:

```
# curl -L
https://github.com/kubernetes-sigs/kind/releases/download/v0.8.1/kind-linux-amd64 -o
kind

# chmod +x ./kind
# sudo mv ./kind /usr/local/bin/kind
```

As you can see, I already have kind installed:

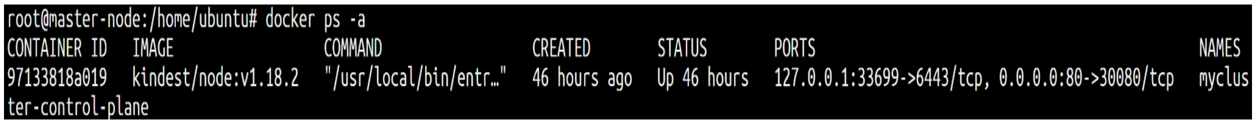


```
root@master-node:/home/ubuntu# kind version
kind v0.8.1 go1.14.2 linux/amd64
```

4.Creating your cluster:

Once all these components are installed, you're ready to create your local Kubernetes cluster. docker ps will show you any containers that are running already. If you have any, you can stop them all at once using: **docker ps -a -q**

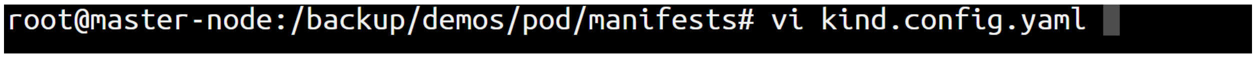
As you can see I already have my container already created, but will show you how to create it.



```
root@master-node:/home/ubuntu# docker ps -a
CONTAINER ID   IMAGE                  COMMAND                  CREATED        STATUS        PORTS                                           NAMES
97133818a019   kindest/node:v1.18.2   "/usr/local/bin/entr…"   46 hours ago   Up 46 hours   127.0.0.1:33699->6443/tcp, 0.0.0.0:80->30080/tcp   myclus
ter-control-plane
```

Note: I'm using PyCharm editor from Jetbrains as it's very excellent when it comes to indentations when creating .yaml files. You can download it from the below link:
https://www.jetbrains.com/pycharm/download/#section=windows

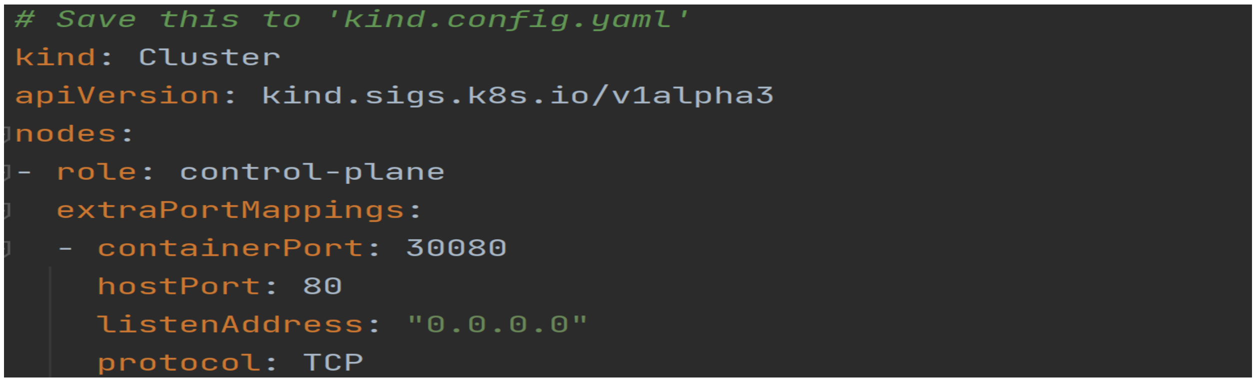
Now, we need a little configuration to prepare our new Kubernetes node. Make a file as below:



```
root@master-node:/backup/demos/pod/manifests# vi kind.config.yaml
```

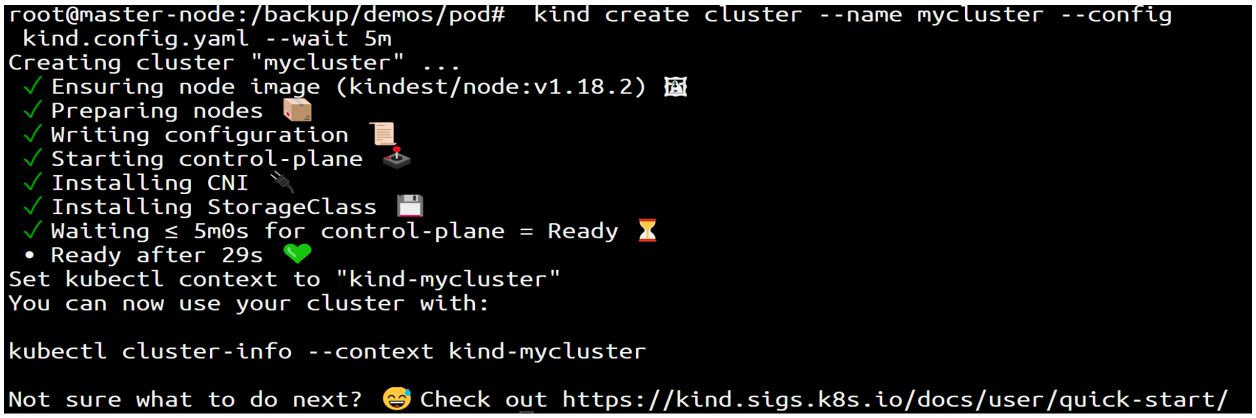
Deploy Your First Kubernetes Cluster:

See below file how it's indented perfectly:



```yaml
# Save this to 'kind.config.yaml'
kind: Cluster
apiVersion: kind.sigs.k8s.io/v1alpha3
nodes:
- role: control-plane
  extraPortMappings:
  - containerPort: 30080
    hostPort: 80
    listenAddress: "0.0.0.0"
    protocol: TCP
```

The extra port mapping is required to allow us to talk to the web-server we will run later on.

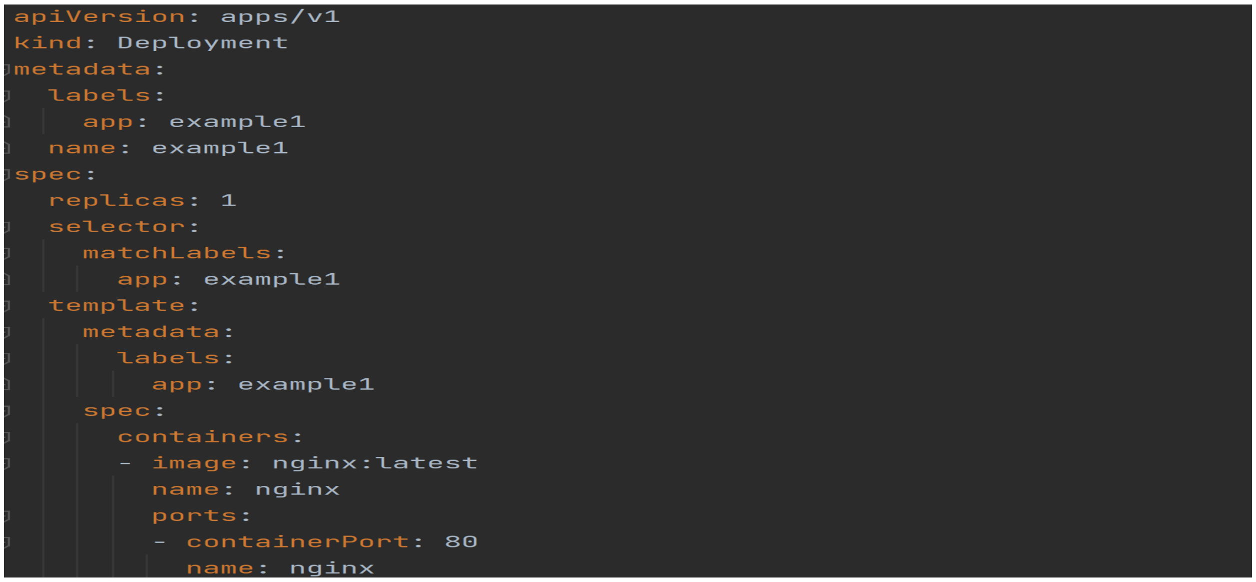


```
root@master-node:/backup/demos/pod#  kind create cluster --name mycluster --config
 kind.config.yaml --wait 5m
Creating cluster "mycluster" ...
 ✓ Ensuring node image (kindest/node:v1.18.2) 🖼
 ✓ Preparing nodes 📦
 ✓ Writing configuration 📜
 ✓ Starting control-plane 🕹️
 ✓ Installing CNI 🔌
 ✓ Installing StorageClass 💾
 ✓ Waiting ≤ 5m0s for control-plane = Ready ⏳
 • Ready after 29s 💚
Set kubectl context to "kind-mycluster"
You can now use your cluster with:

kubectl cluster-info --context kind-mycluster

Not sure what to do next? 😅 Check out https://kind.sigs.k8s.io/docs/user/quick-start/
```

It only takes a few minutes, and after this runs you should see a friendly message telling you your cluster is ready as shown above.

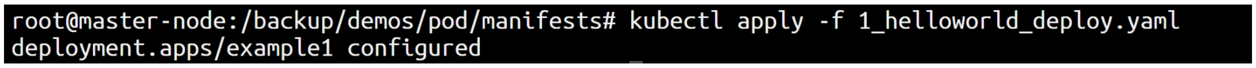
5. Deploy an application:

First, let's describe a workload deployment:



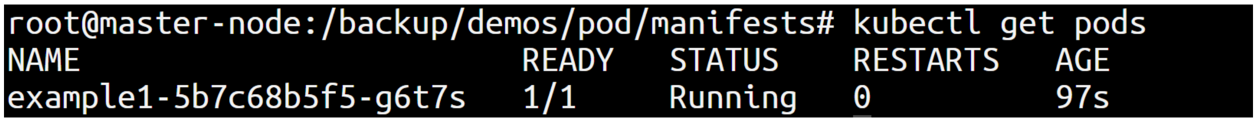
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: example1
  name: example1
spec:
  replicas: 1
  selector:
    matchLabels:
      app: example1
  template:
    metadata:
      labels:
        app: example1
    spec:
      containers:
      - image: nginx:latest
        name: nginx
        ports:
        - containerPort: 80
          name: nginx
```

To apply the workload definition:



```
root@master-node:/backup/demos/pod/manifests# kubectl apply -f 1_helloworld_deploy.yaml
deployment.apps/example1 configured
```

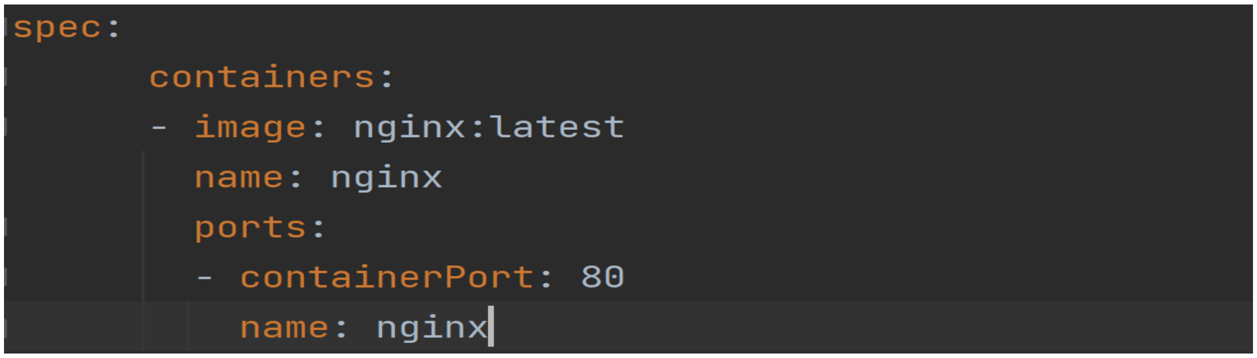
This will deploy the nginx docker container and run it as a process on the cluster. To confirm it's running:



```
root@master-node:/backup/demos/pod/manifests# kubectl get pods
NAME                        READY   STATUS    RESTARTS   AGE
example1-5b7c68b5f5-g6t7s   1/1     Running   0          97s
```

Expose the service:

Replace the spec block from your yaml file with the lines below:



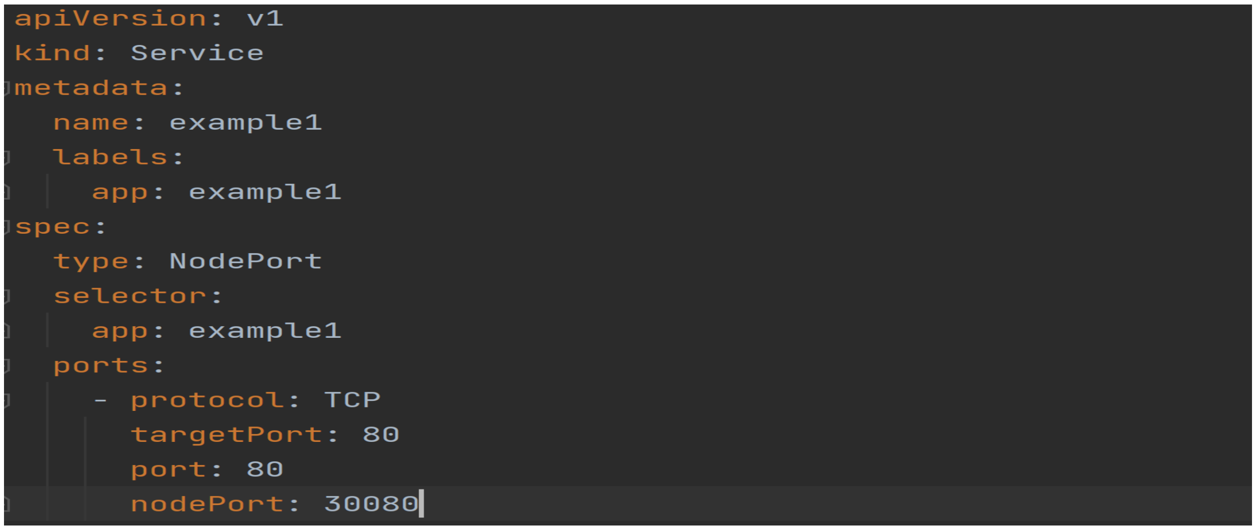
```yaml
spec:
      containers:
      - image: nginx:latest
        name: nginx
        ports:
        - containerPort: 80
          name: nginx
```

Deploy Your First Kubernetes Cluster:

Then, apply the file again:

```
# kubectl apply -f manifests/2_helloworld_deploy_ports.yaml
```

Create a service defined in a yaml file, which should look like this:

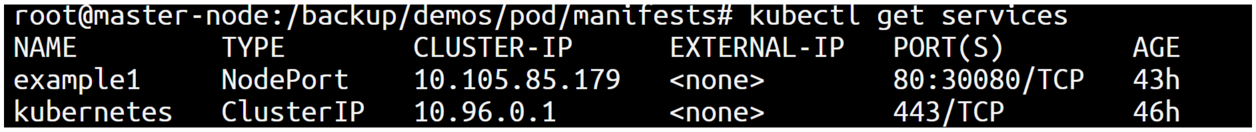


```
apiVersion: v1
kind: Service
metadata:
  name: example1
  labels:
    app: example1
spec:
  type: NodePort
  selector:
    app: example1
  ports:
    - protocol: TCP
      targetPort: 80
      port: 80
      nodePort: 30080
```

Then, apply the file again:

```
# kubectl apply -f  3_helloworld_service.yaml
```

Once you've done this you should see the Service (if you get Services):

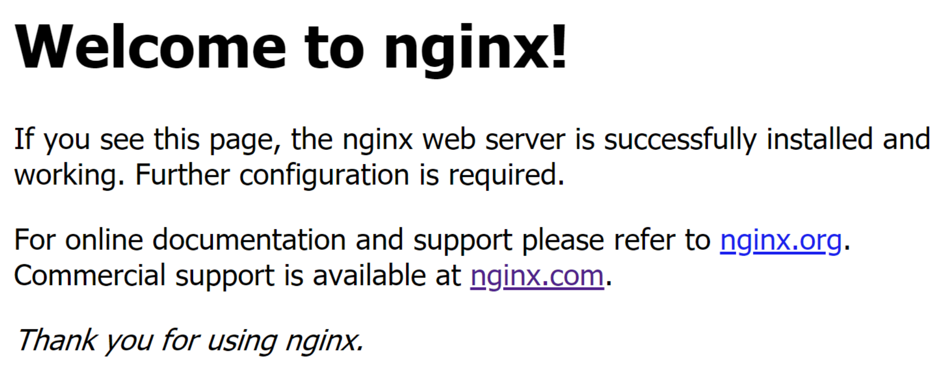


```
root@master-node:/backup/demos/pod/manifests# kubectl get services
NAME          TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
example1      NodePort    10.105.85.179   <none>        80:30080/TCP   43h
kubernetes    ClusterIP   10.96.0.1       <none>        443/TCP        46h
```

You will notice two Services: 1. your example. 2. the Kubernetes Service. This is normal, kind exposes the Service you are using to communicate with the cluster in this way.

Now, navigate to your browser and paste the following http://localhost and you should see the below:

Deploy Your First Kubernetes Cluster:



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

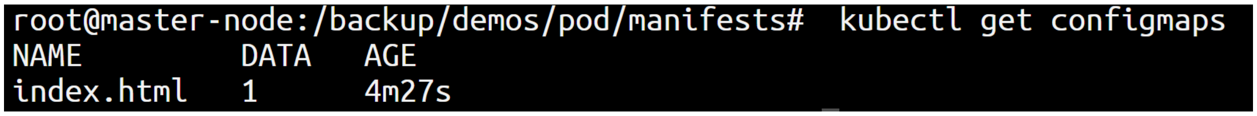
For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

If you want a custom page of your own design:
we're going to use one to supply a replacement index.html file to the nginx default /usr/share/nginx/html web server file path.

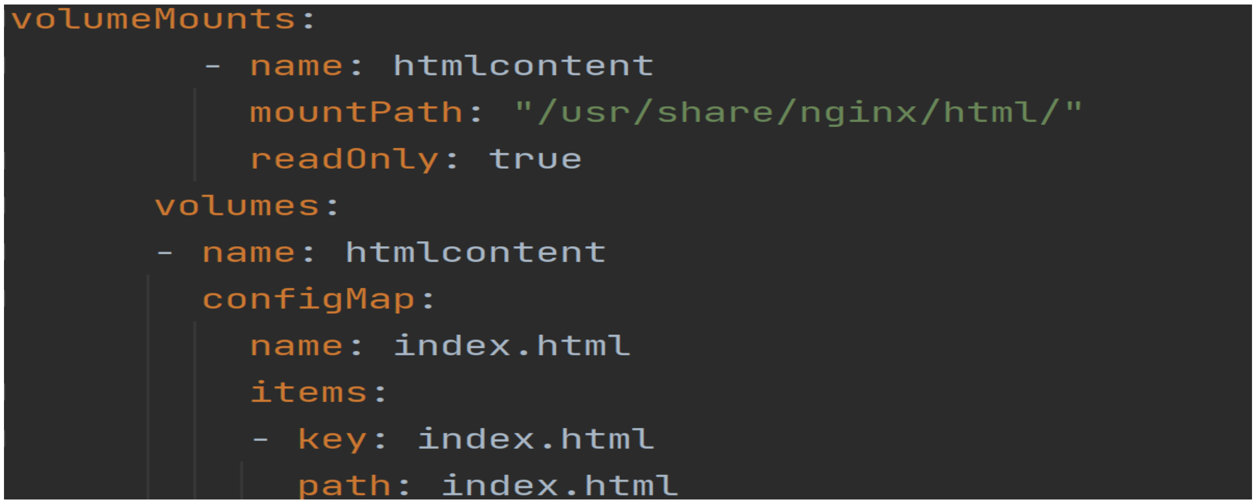
```
# kubectl create configmap index.html --from-file html/index.html
```

Make sure you specify the path correctly.
To make sure it's created issue the following:



```
root@master-node:/backup/demos/pod/manifests#  kubectl get configmaps
NAME          DATA   AGE
index.html    1      4m27s
```

Update your yaml file with the new content below at the end of the file:



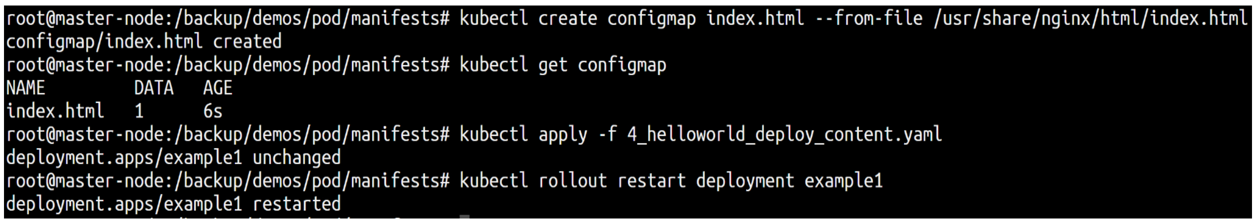
```
volumeMounts:
        - name: htmlcontent
          mountPath: "/usr/share/nginx/html/"
          readOnly: true
      volumes:
      - name: htmlcontent
        configMap:
          name: index.html
          items:
          - key: index.html
            path: index.html
```

Apply it as explained previously :

```
kubectl apply -f manifests/4_helloworld_deploy_content.yaml
```

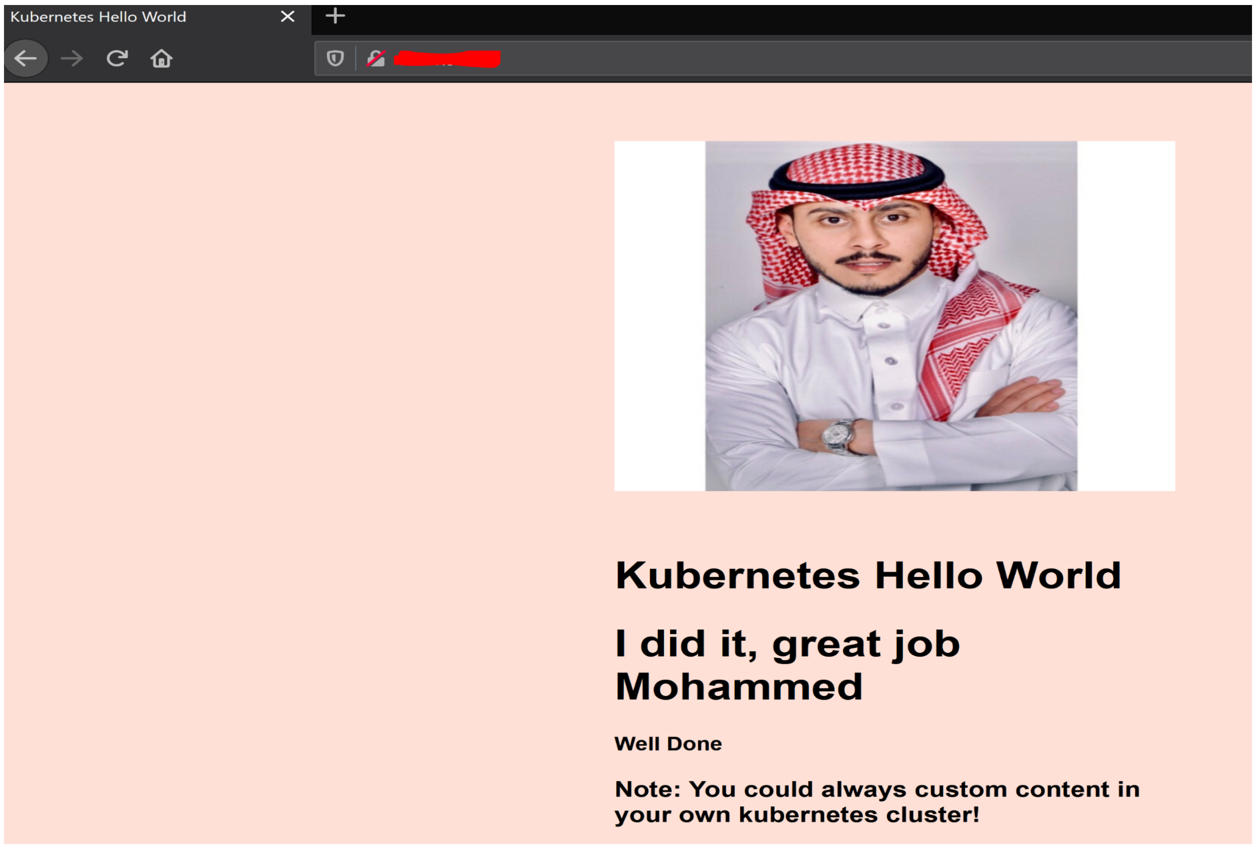
Deploy Your First Kubernetes Cluster:

Note that I created my own design index.html (attached in the repo).



```
root@master-node:/backup/demos/pod/manifests# kubectl create configmap index.html --from-file /usr/share/nginx/html/index.html
configmap/index.html created
root@master-node:/backup/demos/pod/manifests# kubectl get configmap
NAME         DATA   AGE
index.html   1      6s
root@master-node:/backup/demos/pod/manifests# kubectl apply -f 4_helloworld_deploy_content.yaml
deployment.apps/example1 unchanged
root@master-node:/backup/demos/pod/manifests# kubectl rollout restart deployment example1
deployment.apps/example1 restarted
```

Finally, navigate again to your browser and paste the following http://localhost and you should see the below:



Kubernetes Hello World

I did it, great job Mohammed

Well Done

Note: You could always custom content in your own kubernetes cluster!

Hope this helps you get started!

Reference:

1. https://matthewpalmer.net/kubernetes-app-developer/articles/install-kubernetes-ubuntu-tutorial.html
2. https://www.appvia.io/blog/tutorial-deploy-kubernetes-clus