

Creating a Service

Method 1

Operation 1: Write a yaml file for the Deployment that you are creating

apiVersion: v1

kind: ReplicationController

metadata:

name: rcsise

spec:

replicas: 1

selector:

app: sise

template:

metadata:

name: somename

labels:

app: sise

spec:

containers:

- name: sise

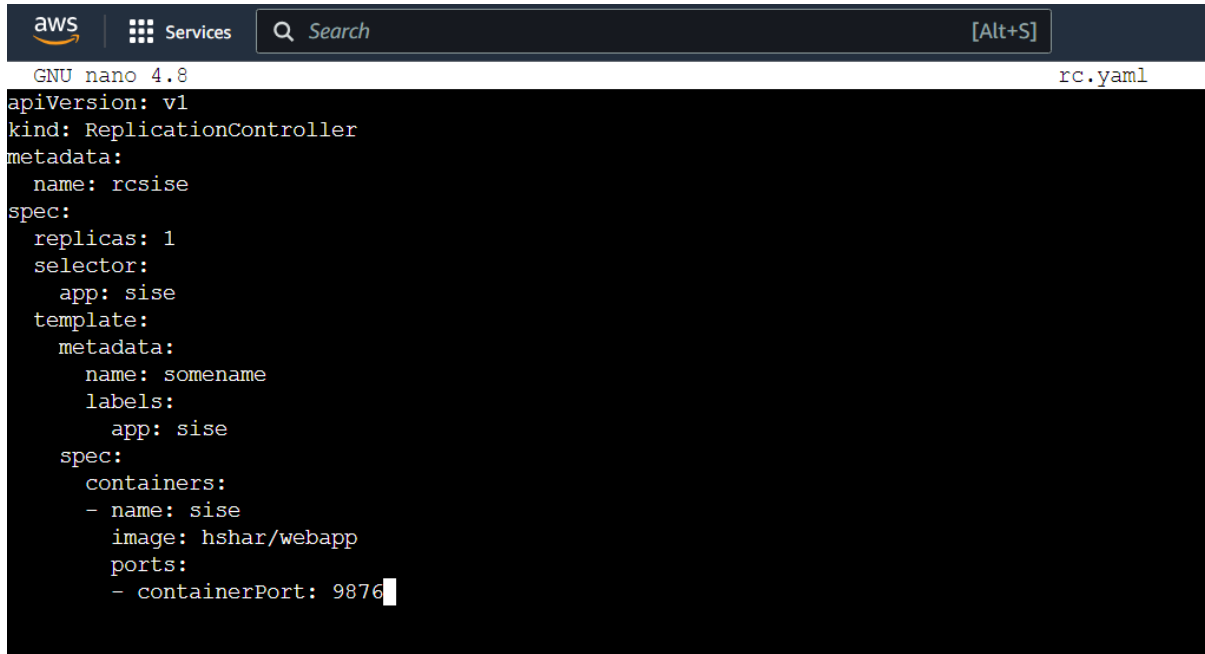
image: hshar/webapp

ports:

- containerPort: 9876

Go to command line and create a yaml file and paste the above created specs.

nano <filename>.yaml

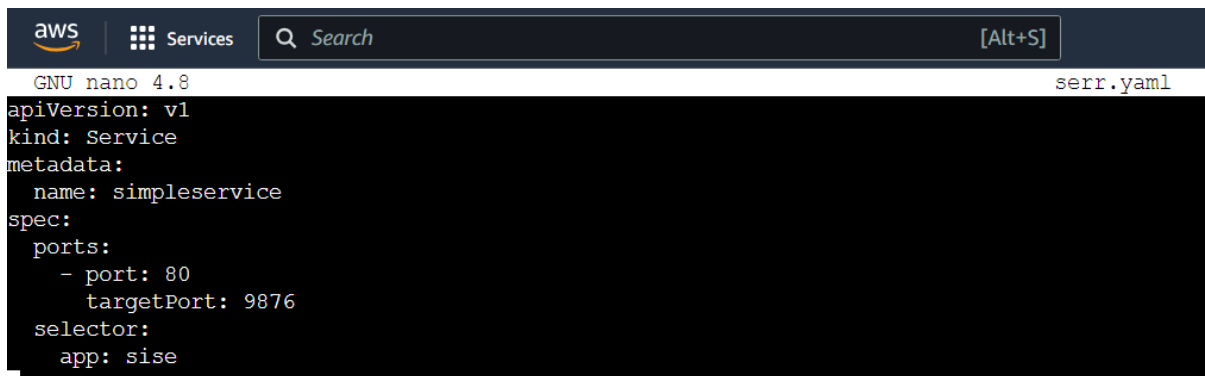


```
aws | Services | Search [Alt+S]
GNU nano 4.8 rc.yaml
apiVersion: v1
kind: ReplicationController
metadata:
  name: rcsise
spec:
  replicas: 1
  selector:
    app: sise
  template:
    metadata:
      name: somename
      labels:
        app: sise
    spec:
      containers:
      - name: sise
        image: hshar/webapp
        ports:
        - containerPort: 9876
```

Once done hit Ctrl+s and then Ctrl+x to save & exit

Do the same for service file

```
apiVersion: v1
kind: Service
metadata:
  name: simpleservice
spec:
  ports:
    - port: 80
      targetPort: 9876
  selector:
    app: sise
```



```
aws | Services | Search [Alt+S]
GNU nano 4.8 serr.yaml
apiVersion: v1
kind: Service
metadata:
  name: simpleservice
spec:
  ports:
    - port: 80
      targetPort: 9876
  selector:
    app: sise
```

Operation 2: next thing to do is to create the yaml file.

kubectl create -f <file name>

```
ubuntu@ip-172-31-90-123:~$ nano rep.yaml
ubuntu@ip-172-31-90-123:~$ nano rc.yaml
ubuntu@ip-172-31-90-123:~$ nano serr.yaml
ubuntu@ip-172-31-90-123:~$ kubectl create -f rc.yaml
replicationcontroller/rcsise created
ubuntu@ip-172-31-90-123:~$
```

Before creating services let's check the service we have, you can have a look at the services by using the following command

kubectl get svc

```
ubuntu@ip-172-31-90-123:~$ kubectl get svc
NAME                TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)    AGE
kubernetes          ClusterIP     10.96.0.1     <none>         443/TCP    2d
ubuntu@ip-172-31-90-123:~$
```

As of now we have default service as kubernetes

Operation 3: now let's create service

kubectl create -f <name of the file>

```
ubuntu@ip-172-31-90-123:~$ kubectl create -f serr.yaml
service/simpleservice created
ubuntu@ip-172-31-90-123:~$
```

And now if will check services, one service name as simpleservices should be created

```
ubuntu@ip-172-31-90-123:~$ kubectl get svc
NAME                TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)    AGE
kubernetes          ClusterIP     10.96.0.1     <none>         443/TCP    2d
simpleservice        ClusterIP     10.99.22.153  <none>         80/TCP     87s
ubuntu@ip-172-31-90-123:~$
```

Method 2 to create service

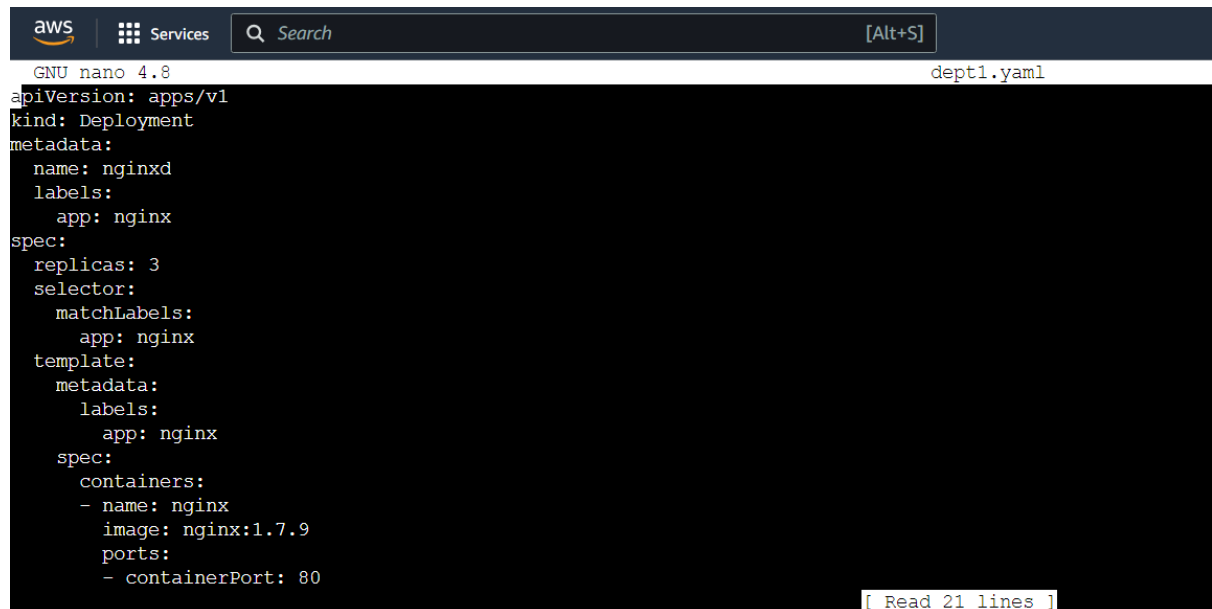
Now if I want to expose my deployment as a service will do the following operations

Operation 4: first we need to create one yaml file with following commands:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginxd
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:1.7.9
          ports:
            - containerPort: 80
```

Go to command line and create a yaml file and paste the above created specs.

nano <filename>.yaml



```
aws | Services | Search [Alt+S]
GNU nano 4.8 dept1.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginxd
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:1.7.9
          ports:
            - containerPort: 80
[ Read 21 lines ]
```

Once done hit Ctrl+s and then Ctrl+x to save & exit

next thing to do is create the deployment yaml file.

`kubectl create -f <file name>`

I have already created, so I used `kubectl get pods` command to see the following pods

```
ubuntu@ip-172-31-90-123:~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-86dcfdf4c6-vqldx   1/1     Terminating    0          2d
nginxd-9d6cbcc65-2zkvs              1/1     Running        0          5m35s
nginxd-9d6cbcc65-6dwdv              1/1     Running        0          5m35s
nginxd-9d6cbcc65-kfhd9              1/1     Running        0          5m35s
rcsise-9xz8v                        1/1     Running        0          21m
ubuntu@ip-172-31-90-123:~$
```

And now let's check the deployments-

`kubectl get deployment` or `kubectl get deploy`

```
ubuntu@ip-172-31-90-123:~$ kubectl get deployment
NAME      READY   UP-TO-DATE   AVAILABLE   AGE
nginxd    3/3     3            3           7m56s
ubuntu@ip-172-31-90-123:~$ kubectl get deploy
NAME      READY   UP-TO-DATE   AVAILABLE   AGE
nginxd    3/3     3            3           8m5s
ubuntu@ip-172-31-90-123:~$
```

Operation 5: now to expose our deployment write-

`kubectl expose deployment/nginxd --type="NodePort" --port 8080`

```
ubuntu@ip-172-31-90-123:~$ kubectl expose deployment/nginxd --type="NodePort" --port 8080
service/nginxd exposed
ubuntu@ip-172-31-90-123:~$
```

And now we will do

`kubectl get svc`

```
ubuntu@ip-172-31-90-123:~$ kubectl get svc
NAME            TYPE          CLUSTER-IP    EXTERNAL-IP   PORT(S)          AGE
kubernetes      ClusterIP     10.96.0.1     <none>        443/TCP          2d1h
nginxd          NodePort      10.104.156.49 <none>        8080:30470/TCP   2m9s
simpleservice   ClusterIP     10.99.22.153  <none>        80/TCP           26m
ubuntu@ip-172-31-90-123:~$
```

Method 3 to create service

Operation 6: using command line

kubectl create svc nodeport nginx --tcp=80:80

```
ubuntu@ip-172-31-90-123:~$ kubectl create svc nodeport nginx --tcp=80:80
service/nginx created
ubuntu@ip-172-31-90-123:~$
```

And if I do **kubectl get svc** then it should show me this new service named nginx as well

```
ubuntu@ip-172-31-90-123:~$ kubectl get svc
NAME                TYPE          CLUSTER-IP      EXTERNAL-IP      PORT(S)          AGE
kubernetes           ClusterIP      10.96.0.1        <none>           443/TCP          2d1h
nginx                 NodePort       10.97.197.131    <none>           80:32477/TCP     69s
nginxd                NodePort       10.104.156.49    <none>           8080:30470/TCP   9m30s
simpleservice         ClusterIP      10.99.22.153     <none>           80/TCP           34m
ubuntu@ip-172-31-90-123:~$
```

Now let's check the nginx port is exposed or not, for that will copy Master public IP and paste it in web browser followed by port

Publicip:port no

The screenshot shows the AWS Management Console interface. The top navigation bar includes the AWS logo, a search bar, and user information (N. Virginia, Rahul Verma). The left sidebar contains navigation links for EC2 Dashboard, EC2 Global View, Events, Console-to-Code, and various instance types and templates. The main content area displays a table of EC2 instances under the heading 'Instances (1/3) Info'. The table has columns for Name, Instance ID, Instance state, Instance type, Status check, Alarm status, and Availability Zone. Three instances are listed: Master (Running, t2.medium, 2/2 checks passed), Slave (Running, t2.medium, 2/2 checks passed), and Slave 2 (Stopped, t2.medium, -). Below the table, the details for the Master instance (i-0c20b31056ef412f5) are shown. The 'Details' tab is active, displaying the Instance summary, Instance ID, IPv6 address, Hostname type, and IP name. A tooltip indicates that the Public IPv4 address (44.202.114.29) has been copied. The Private IPv4 addresses are listed as 172.31.90.123. The Public IPv4 DNS is also shown. The bottom of the screen shows the Windows taskbar with the date and time (18:17, 19-04-2024).

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
Master	i-0c20b31056ef412f5	Running	t2.medium	2/2 checks passed	View alarms +	us-east-1c
Slave	i-0b021a3ea8850317f	Running	t2.medium	2/2 checks passed	View alarms +	us-east-1c
Slave 2	i-0da2f06f713200c03	Stopped	t2.medium	-	View alarms +	us-east-1c

Instance: i-0c20b31056ef412f5 (Master)

Details | Status and alarms New | Monitoring | Security | Networking | Storage | Tags

Instance summary Info

Instance ID: i-0c20b31056ef412f5 (Master)

IPv6 address: -

Hostname type: -

IP name: ip-172-31-90-123.ec2.internal

Public IPv4 address copied: 44.202.114.29 [Open address](#)

Instance state: Running

Private IPv4 DNS name (IPv4 only): ip-172-31-90-123.ec2.internal

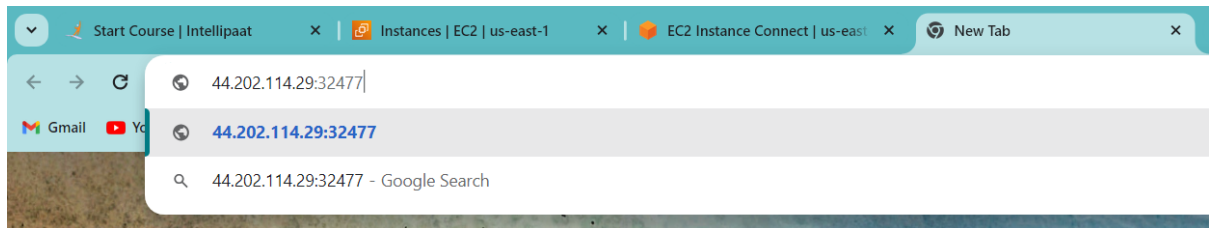
Private IPv4 addresses: 172.31.90.123

Public IPv4 DNS: ec2-44-202-114-29.compute-1.amazonaws.com [Open address](#)

Activate Windows
Go to Settings to activate Windows.



Copy paste **publicip:portno** in your web browser



And it's showing nginx page so it is exposed to external world

