

ADVANCE DEVOPS EXP 4

Sanket More

D15A 30

Aim: To install Kubectl and execute Kubectl commands to manage the Kubernetes cluster and deploy Your First Kubernetes Application.

Step 1: Install Kubectl on Ubuntu

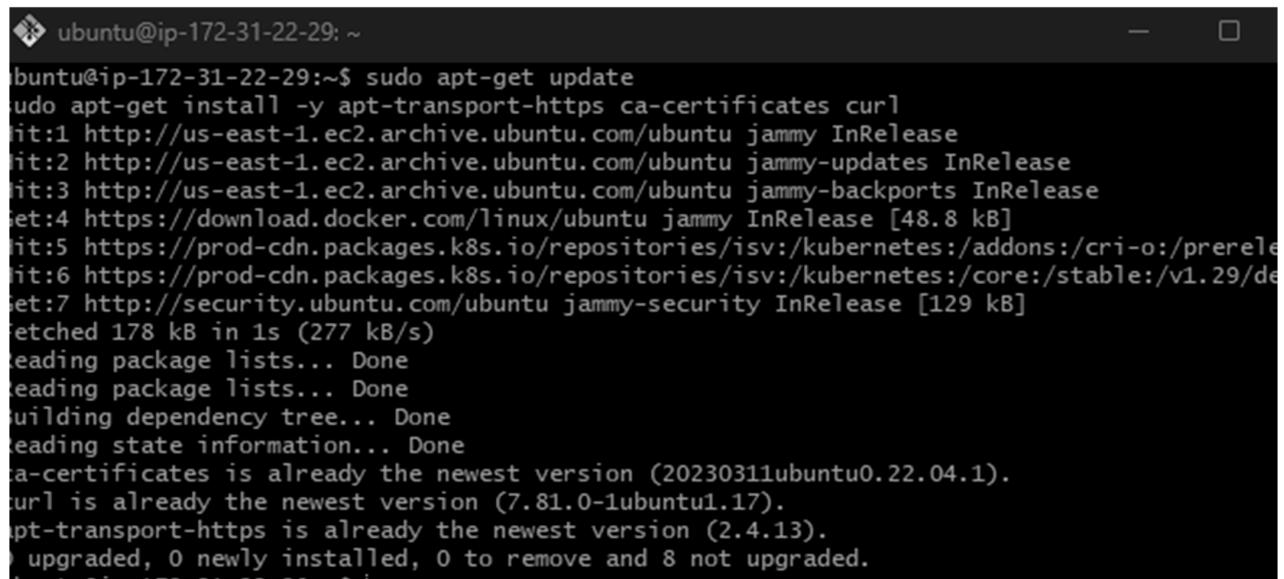
1.1 Add Kubernetes APT repository

First, add the Kubernetes repository to your system.

1. Install prerequisites:

```
sudo apt-get update
```

```
sudo apt-get install -y apt-transport-https ca-certificates curl
```



```
ubuntu@ip-172-31-22-29:~$ sudo apt-get update
[sudo] password for ubuntu:
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease
Get:4 https://download.docker.com/linux/ubuntu jammy InRelease [48.8 kB]
Get:5 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/addons:/cri-o:/prerelease
Get:6 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.29/debian
Get:7 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Fetched 178 kB in 1s (277 kB/s)
Reading package lists... Done
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ca-certificates is already the newest version (20230311ubuntu0.22.04.1).
curl is already the newest version (7.81.0-1ubuntu1.17).
apt-transport-https is already the newest version (2.4.13).
0 upgraded, 0 newly installed, 0 to remove and 8 not upgraded.
[11:51:17 21-03-2023]
```

2. Add the GPG key for Kubernetes:

```
sudo curl -fsSLo  
/usr/share/keyrings/kubernetes-archive-keyring.gpg  
https://packages.cloud.google.com/apt/doc/apt-key.gpg
```

```
ubuntu@ip-172-31-22-29:~$ sudo curl -fsSLo /usr/share/keyrings/kubernetes-archive-keyring.gpg  
https://packages.cloud.google.com/apt/doc/apt-key.gpg
```

3. Add the Kubernetes repository:

```
echo "deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring.gpg]
https://apt.kubernetes.io/ kubernetes-focal main" | sudo tee
/etc/apt/sources.list.d/kubernetes.list
```

```
ubuntu@ip-172-31-22-29:~$ echo "deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring.gpg]
https://apt.kubernetes.io/ kubernetes-focal main" | sudo tee /etc/apt/sources.list.d/kubernetes.list
deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring.gpg] https://apt.kubernetes.io/
kubernetes-focal main
```

1.2 Install

kubectl Now install

kubectl: sudo

apt-get update

sudo apt-get install -y kubectl

```
ubuntu@ip-172-31-22-29:~$ sudo apt-get update
sudo apt-get install -y kubectl
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu jammy-security InRelease
Hit:5 https://download.docker.com/linux/ubuntu jammy InRelease
Hit:6 https://prod-cdn.packages.k8s.io/repositories/istv:/kubernetes:/addons:/cri-o:/prerelease:/main/deb InRelease
Ign:7 https://packages.cloud.google.com/apt kubernetes-focal InRelease
Err:8 https://packages.cloud.google.com/apt kubernetes-focal Release
  404 Not Found [IP: 172.253.62.138 443]
Reading package lists... Done
E: The repository 'https://apt.kubernetes.io kubernetes-focal Release' does not have a Release file.
N: Updating from such a repository can't be done securely, and is therefore disabled by default.
N: See apt-secure(8) manpage for repository creation and user configuration details.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
kubectl is already the newest version (1.29.0-1.1).
0 upgraded, 0 newly installed, 0 to remove and 5 not upgraded.
```

Verify the installation(extra):

kubectl version --client

```
ubuntu@ip-172-31-22-29:~$ kubectl version --client
Client Version: v1.29.0
Kustomize Version: v5.0.4-0.20230601165947-6ce0bf390ce3
```

Step 2: Deploying Your Application on Kubernetes

2.1 Set up Kubernetes Cluster

1. If you haven't already set up a Kubernetes cluster (e.g., with kubeadm), use minikube or any managed Kubernetes service (like EKS, GKE, etc.) to get a cluster running.
2. Once your cluster is ready, verify the nodes: `kubectl get nodes`

```
ubuntu@ip-172-31-45-227:~$ kubectl get nodes
NAME           STATUS   ROLES      AGE    VERSION
ip-172-31-43-211   Ready    <none>    50s   v1.29.0
ip-172-31-45-13   Ready    <none>    34s   v1.29.0
ip-172-31-45-227   Ready    control-plane  5m17s  v1.29.0
ubuntu@ip-172-31-45-227:~$ |
```

Step 3: Create the Deployment YAML file

- a) Create the YAML file: Use a text editor to create a file named `nginx-deployment.yaml`

```
ubuntu@ip-172-31-45-227:~$ nano nginx-deployment.yaml
```

- b) Add the Deployment Configuration: Copy and paste the following YAML content into the file. Save and exit the editor (Press Ctrl+X, then Y, and Enter).

```
ubuntu@ip-172-31-45-227: ~
GNU nano 6.2                                     nginx-deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:1.21.3
          ports:
            - containerPort: 80
```

Step 4:Create the Service YAML File

- a) Create the YAML File: Create another file named nginx-service.yaml

```
ubuntu@ip-172-31-45-227:~$ nano nginx-service.yaml
```

- b) Add the Service Configuration: Copy and paste the following YAML content into the file given below

```
ubuntu@ip-172-31-45-227:~$ GNU nano 6.2                                     nginx-service.yaml *
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
  type: LoadBalancer
```

Step 5: Apply the YAML Files

- a) Deploy the Application: Use kubectl to create the Deployment and Service from the YAML files.

```
ubuntu@ip-172-31-45-227:~$ kubectl apply -f nginx-deployment.yaml
kubectl apply -f nginx-service.yaml
deployment.apps/nginx-deployment created
service/nginx-service created
```

- b) Verify the Deployment: Check the status of your Deployment, Pods and Services.

```
ubuntu@ip-172-31-45-227:~$ kubectl get deployments
kubectl get pods
kubectl get services
NAME           READY   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment   2/2     2           2          40s
NAME                  READY   STATUS    RESTARTS   AGE
nginx-deployment-6b4d6fdbf-6k84m   1/1     Running   0          40s
nginx-deployment-6b4d6fdbf-9d8j6   1/1     Running   0          40s
NAME            TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
kubernetes      ClusterIP   10.96.0.1    <none>        443/TCP      40m
nginx-service   LoadBalancer  10.106.182.152  <pending>      80:32317/TCP  40s
```

Describe the deployment(Extra)

```
ubuntu@ip-172-31-45-227:~$ kubectl get deployments
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment  1/1     1           1           14h
ubuntu@ip-172-31-45-227:~$ kubectl describe deployment
Name:            nginx-deployment
Namespace:       default
CreationTimestamp:  Wed, 11 Sep 2024 17:16:17 +0000
Labels:          <none>
Annotations:    deployment.kubernetes.io/revision: 2
Selector:        app=nginx
Replicas:        1 desired | 1 updated | 1 total | 1 available | 0 unavailable
StrategyType:   RollingUpdate
MinReadySeconds: 0
RollingUpdateStrategy: 25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=nginx
  Containers:
    nginx:
      Image:      nginx:latest
      Port:       80/TCP
      Host Port:  0/TCP
      Environment: <none>
      Mounts:
        /usr/share/nginx/html from website-volume (rw)
  Volumes:
    website-volume:
      Type:      ConfigMap (a volume populated by a ConfigMap)
      Name:      nginx-website
      Optional:  false
Conditions:
  Type    Status  Reason
  ----  -----
  Available  True    MinimumReplicasAvailable
  Progressing  True    NewReplicaSetAvailable
OldReplicaSets:  nginx-deployment-6b4d6fdbf (0/0 replicas created)
NewReplicaSet:   nginx-deployment-776b8fd845 (1/1 replicas created)
Events:          <none>
```

Step 6:Ensure Service is Running

6.1 **Verify Service:** Run the following command to check the services running in your cluster:

```
kubectl get service
```

```
ubuntu@ip-172-31-45-227:~$ kubectl get service
NAME      TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
kubernetes  ClusterIP  10.96.0.1      <none>        443/TCP      16h
nginx     NodePort   10.106.0.176    <none>        80:32618/TCP  76m
nginx-service  NodePort   10.106.182.152    <none>        80:30007/TCP  15h
nginx2     NodePort   10.99.32.156    <none>        80:31421/TCP  8s
```

Step 7:Forward the Service Port to Your Local Machine

kubectl port-forward allows you to forward a port from your local machine to a port on a service running in the Kubernetes cluster.

1. **Forward the Service Port:** Use the following command to forward a local port to the service's target port.

```
kubectl port-forward service/<service-name> <local-port>:<service-port>
```

```
ubuntu@ip-172-31-45-227:~$ kubectl port-forward service/nginx-service 8080:80
Forwarding from 127.0.0.1:8080 -> 80
Forwarding from [::1]:8080 -> 80
```

This command will forward local port 8080 on your machine to port 80 of the service nginx-service running inside the cluster.

2. This means port forwarding is now active, and any traffic to localhost:8080 will be routed to the nginx-service on port 80.

```
ubuntu@ip-172-31-45-227:~$ kubectl port-forward service/nginx-service 8080:80
Forwarding from 127.0.0.1:8080 -> 80
Forwarding from [::1]:8080 -> 80
^Cubuntu@ip-172-31-45-227:~$ kubectl port-forward service/nginx-service 8081:8080
Forwarding from 127.0.0.1:8081 -> 80
Forwarding from [::1]:8081 -> 80
^Cubuntu@ip-172-31-45-227:~$ kubectl get pods
NAME           READY   STATUS    RESTARTS   AGE
nginx-deployment-776b8fd845-k9cx4   1/1     Running   0          113m
ubuntu@ip-172-31-45-227:~$ kubectl logs nginx-deployment-776b8fd845-k9cx4
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2024/09/12 06:35:51 [notice] 1#1: using the "epoll" event method
2024/09/12 06:35:51 [notice] 1#1: nginx/1.27.1
2024/09/12 06:35:51 [notice] 1#1: built by gcc 12.2.0 (Debian 12.2.0-14)
2024/09/12 06:35:51 [notice] 1#1: OS: Linux 6.5.0-1022-aws
2024/09/12 06:35:51 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2024/09/12 06:35:51 [notice] 1#1: start worker processes
2024/09/12 06:35:51 [notice] 1#1: start worker process 24
2024/09/12 06:35:51 [notice] 1#1: start worker process 25
^C
```

Step 8: Access the Application Locally

1. **Open a Web Browser:** Now open your web browser and go to the following URL:

http://localhost:8080

You should see the application (in this case, Nginx) that you have deployed running in the Kubernetes cluster, served locally via port 8080.

In case the port 8080 is unavailable, try using a different port like 8081

