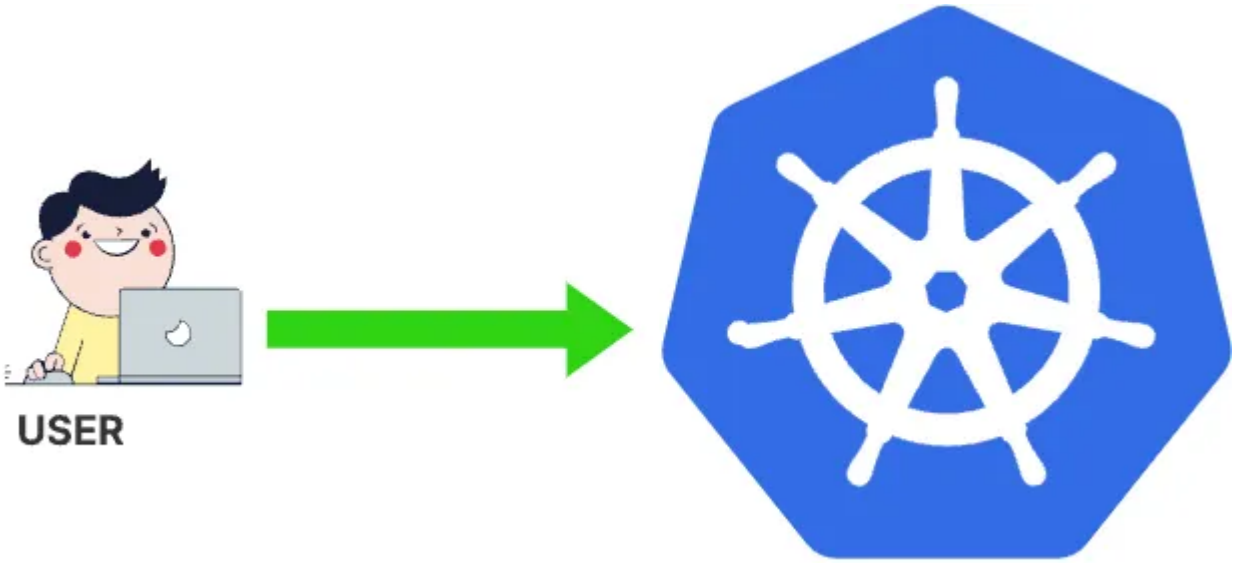
How to Create a User in a Kubernetes Cluster and Grant Access

Hakan Bayraktar · Follow

4 min read · Nov 18, 2023



In this detailed guide, we'll illustrate the steps required to create a user, generate necessary certificates, and configure access using a kubeconfig file within a Kubernetes cluster.

Step 1: Generating a Key Pair and Certificate Signing Request (CSR)

First, let's generate a key pair and a Certificate Signing Request (CSR) using OpenSSL:

```
openssl genrsa -out developer.key 2048
openssl req -new -key developer.key -out developer.csr -subj "/CN=developer"
```

```
root@hakan-yeni:~/rbac# openssl genrsa -out developer.key 2048
root@hakan-yeni:~/rbac# openssl req -new -key developer.key -out developer.csr -subj "/CN=developer"
```

Now, let's create create a CSR YAML file named " `csr_template.yaml` " to submit to Kubernetes:

csr_template.yaml

```
cat <<EOF > csr_template.yaml
apiVersion: certificates.k8s.io/v1
kind: CertificateSigningRequest
metadata:
  name: developer-csr
spec:
  request: <Base64_encoded_CSR>
  signerName: kubernetes.io/kube-apiserver-client
  usages:
  - client auth
EOF
```

Replace `<Base64_encoded_CSR>` with the Base64-encoded content of the `developer.csr` file.

Generate the CSR content in Base64 and create the YAML file:

```
CSR_CONTENT=$(cat developer.csr | base64 | tr -d '\n')
sed "s|<Base64_encoded_CSR>|$CSR_CONTENT|" csr_template.yaml > developer_csr.ya
```

Apply the CSR YAML file to Kubernetes:

```
kubectl create -f developer_csr.yaml
```

```
root@hakan-yeni:~/rbac# cat <<EOF > csr_template.yaml
apiVersion: certificates.k8s.io/v1
kind: CertificateSigningRequest
metadata:
  name: developer-csr
spec:
  request: <Base64_encoded_CSR>
  signerName: kubernetes.io/kube-apiserver-client
  usages:
  - client auth
EOF
root@hakan-yeni:~/rbac# CSR_CONTENT=$(cat developer.csr | base64 | tr -d '\n')
sed "s|<Base64_encoded_CSR>|$CSR_CONTENT|" csr_template.yaml > developer_csr.yaml
root@hakan-yeni:~/rbac# kubectl create -f developer_csr.yaml
certificatesigningrequest.certificates.k8s.io/developer-csr created
root@hakan-yeni:~/rbac#
```

Approve the CSR and retrieve the approved certificate:

```
kubectl get csr
kubectl certificate approve developer-csr
kubectl get csr developer-csr -o jsonpath='{.status.certificate}' | base64 --de
kubectl get csr
```

```
root@hakan-yeni:~/rbac# kubectl get csr
NAME            AGE   SIGNERNAME                            REQUESTOR          REQUESTEDDURATION   CONDITION
developer-csr   66s   kubernetes.io/kube-apiserver-client   kubernetes-admin   <none>              Pending
root@hakan-yeni:~/rbac# kubectl certificate approve developer-csr
certificatesigningrequest.certificates.k8s.io/developer-csr approved
root@hakan-yeni:~/rbac# kubectl get csr developer-csr -o jsonpath='{.status.certificate}' | base64 --decode > developer.crt
root@hakan-yeni:~/rbac# kubectl get csr
NAME            AGE    SIGNERNAME                            REQUESTOR          REQUESTEDDURATION   CONDITION
developer-csr   106s   kubernetes.io/kube-apiserver-client   kubernetes-admin   <none>              Approved,Issued
root@hakan-yeni:~/rbac#
```

Step 2: Generate and Configure a kubeconfig File

To access the Kubernetes cluster, it's essential to generate a configuration file tailored for the 'developer' user. This file needs to encompass critical information, including the Kubernetes API access specifics, the Cluster CA certificate, as well as the 'developer' user's certificate and context name. Initially, we'll generate the kubeconfig file specifically for the 'developer' user.

Configure the kubeconfig file:

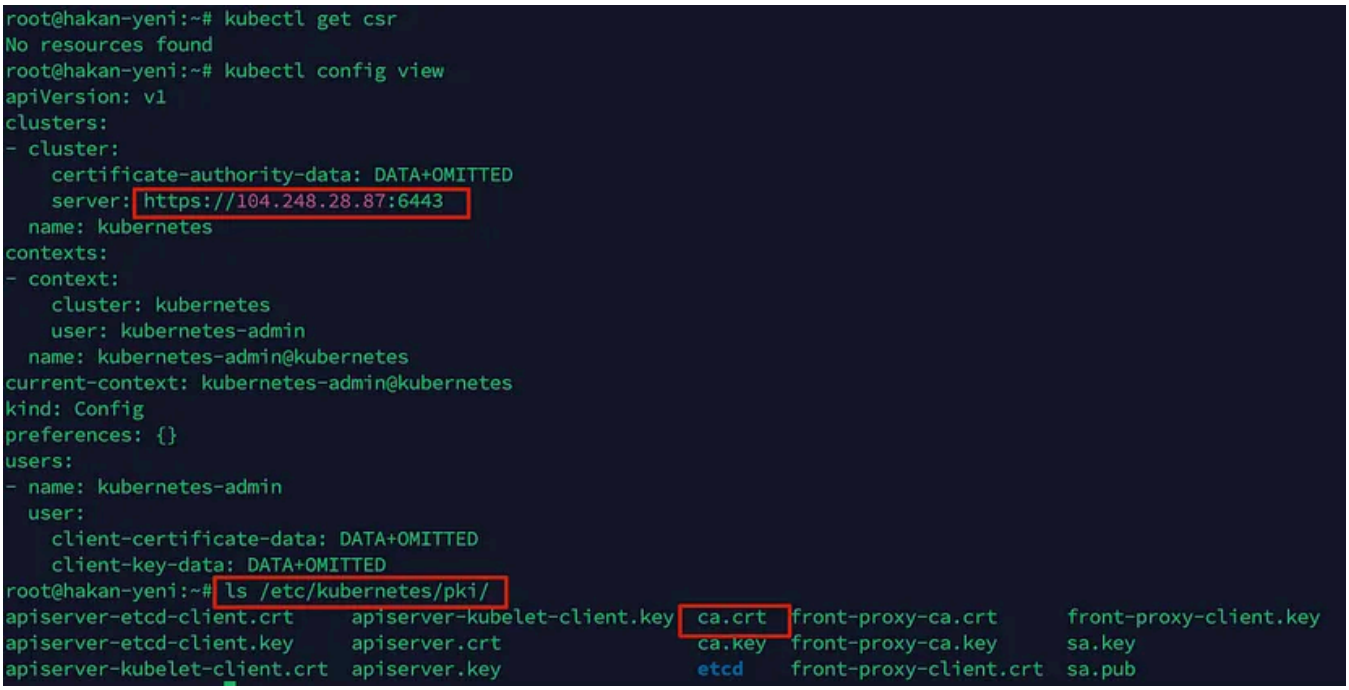
We need to modify below the command according to our cluster-specific information **to Set Cluster Configuration:**

kubectl config set-cluster kubernetes — server=**https://<Kubernetes_API_server_endpoint>:<port>** — certificate-authority=**<Base64_encoded_CA_certificate>** — embed-certs=true — kubeconfig=developer.kubeconfig

> *Replace **<Kubernetes_API_server_endpoint>** with the address of the Kubernetes API server and **<port>** with the corresponding port number. Also, replace **<Base64_encoded_CA_certificate>** with the file path of the CA certificate in Base64 encoding.*

First, we need to locate the cluster's Kubernetes API access details and the Cluster CA certificate:

```
kubectl config view
ls /etc/kubernetes/pki/
```



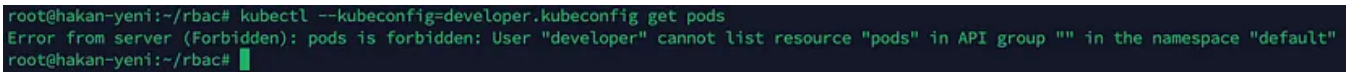
I changed the command above according our cluster information.

```
# Set Cluster Configuration:
kubectl config set-cluster kubernetes --server=https://104.248.28.87:6443 --cer
```

```
# Set Credentials for Developer:
kubectl config set-credentials developer --client-certificate=developer.crt --c
# Set Developer Context:
kubectl config set-context developer-context --cluster=kubernetes --namespace=d
# Use Developer Context:
kubectl config use-context developer-context --kubeconfig=developer.kubeconfig
```

Verify the kubeconfig file's configuration:

```
kubectl --kubeconfig=developer.kubeconfig get pods
```



We logged into the cluster with the 'Developer' user and attempted to list the pods in the 'default' namespace. However, due to the lack of necessary permissions for the 'Developer' user, we couldn't retrieve the list of pods. Below, you can find how to grant the required permissions to this user.

Step 3: Assign Roles and Bindings for the Developer User

Create and apply roles and role bindings for the developer user:

developer-cluster-role.yaml

```
cat <<EOF > developer-cluster-role.yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: developer-role
rules:
- apiGroups: ["", "extensions", "apps"]
```

```
  resources: ["*"]
  verbs: ["*"]
EOF
```

developer-role-binding.yaml

```
cat <<EOF > developer-role-binding.yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: developer-binding
  namespace: default
subjects:
- kind: User
  name: developer
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: developer-role
  apiGroup: rbac.authorization.k8s.io
EOF
```

Apply the roles and role bindings:

```
kubectl apply -f developer-cluster-role.yaml -f developer-role-binding.yaml
```

```
root@hakan-yeni:~/rbac# cat <<EOF > developer-cluster-role.yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: developer-role
rules:
- apiGroups: ["", "extensions", "apps"]
  resources: ["*"]
  verbs: ["*"]
EOF
root@hakan-yeni:~/rbac# cat <<EOF > developer-role-binding.yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: developer-binding
  namespace: default
subjects:
- kind: User
  name: developer
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: developer-role
  apiGroup: rbac.authorization.k8s.io
EOF
root@hakan-yeni:~/rbac# kubectl apply -f developer-cluster-role.yaml -f developer-role-binding.yaml
clusterrole.rbac.authorization.k8s.io/developer-role created
rolebinding.rbac.authorization.k8s.io/developer-binding created
root@hakan-yeni:~/rbac#
```

Step 4: Verify developer User Rights

You can run the following commands to check the permissions assigned to the ‘developer’ user for accessing the Kubernetes cluster resources.

```
kubectl --kubeconfig=developer.kubeconfig get pods
kubectl --kubeconfig=developer.kubeconfig run nginx --image=nginx
kubectl --kubeconfig=developer.kubeconfig get pods
```

```
root@hakan-yeni:~/rbac# kubectl --kubeconfig=developer.kubeconfig get pods
No resources found in default namespace.
root@hakan-yeni:~/rbac# kubectl --kubeconfig=developer.kubeconfig run nginx --image=nginx
pod/nginx created
root@hakan-yeni:~/rbac# kubectl --kubeconfig=developer.kubeconfig get pods
NAME    READY   STATUS    RESTARTS   AGE
nginx   1/1     Running   0          4s
```

This confirms that the developer user has appropriate access to pods in the default namespace.

```
kubectl --kubeconfig=developer.kubeconfig get pods -A
```

We couldn't retrieve the information about pods across all namespaces. This limitation occurred because the permissions granted to the developer user are only applicable to the 'default' namespace.

Note: You can find more information about RBAC Authorization and different user role-related details at his link: https://kubernetes.io/docs/reference/access-authn-authz/rbac/

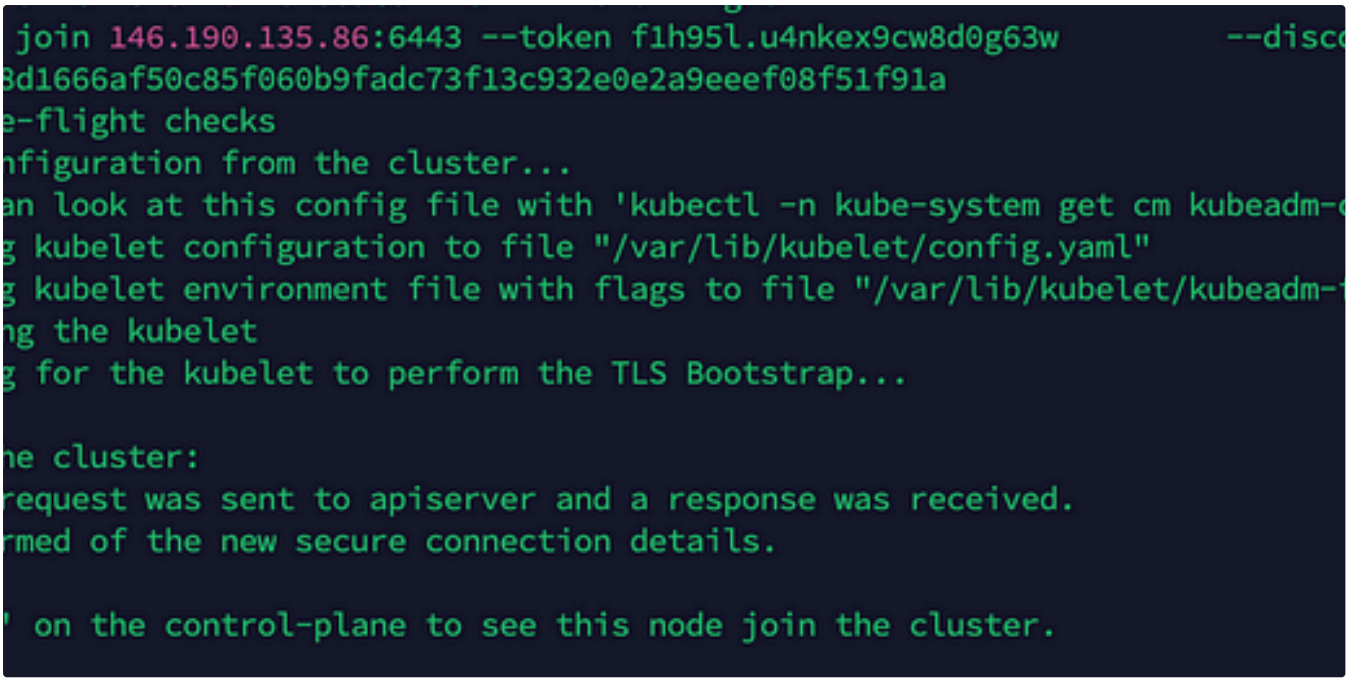
Kubernetes Cluster | Rbac Access Control | Kubeconfig | Kubernetes

Follow

Written by Hakan Bayraktar

265 Followers

More from Hakan Bayraktar



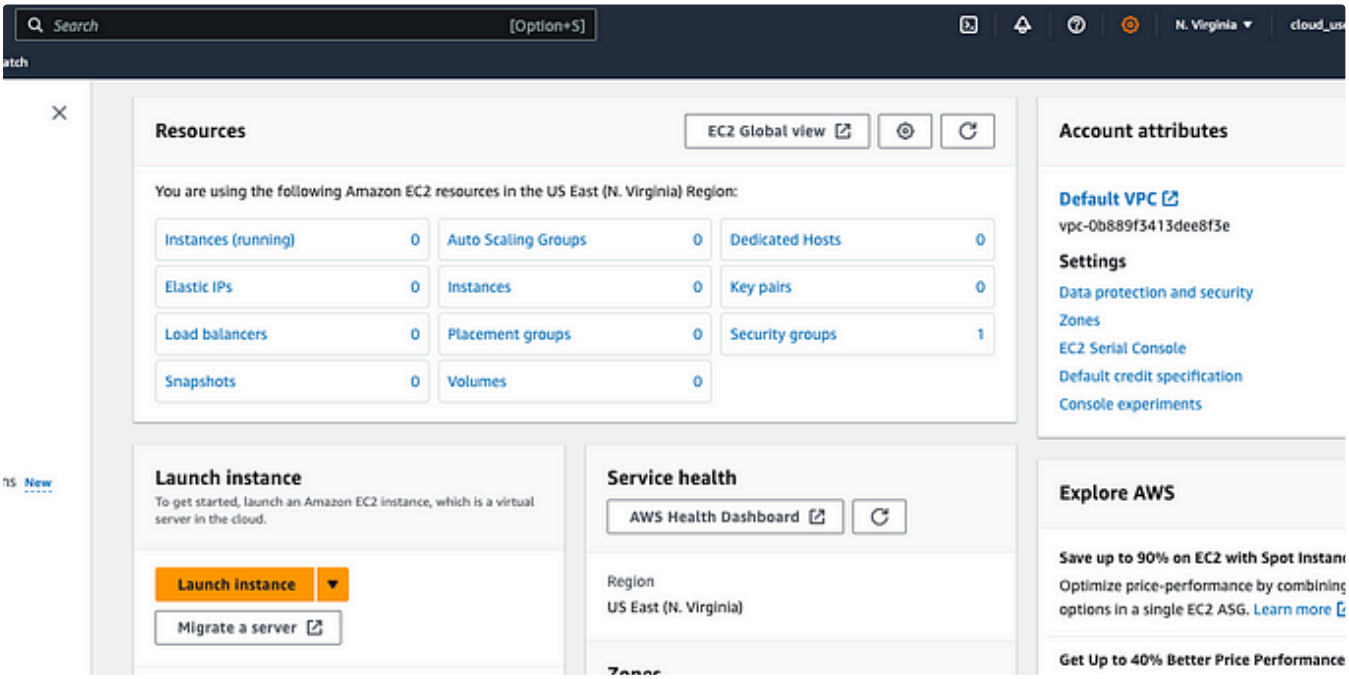
Hakan Bayraktar

How to Install Kubernetes Cluster on Ubuntu 22.04 (Step-by-Step Guide)

Introduction

6 min read · Nov 3, 2023

269 13



Hakan Bayraktar

How to Install PostgreSQL 15 on Amazon Linux 2023: A Step-by-Step Guide

Introduction

6 min read · Nov 9, 2023

80 4

Hakan Bayraktar

How to Setup Dynamic NFS Provisioning in a Kubernetes Cluster

Dynamic NFS storage provisioning in Kubernetes streamlines the creation and management of NFS volumes for your Kubernetes applications. It...

4 min read · Nov 3, 2023

60

Hakan Bayraktar

Merging Multiple kubeconfig Files into One: A Comprehensive Guide

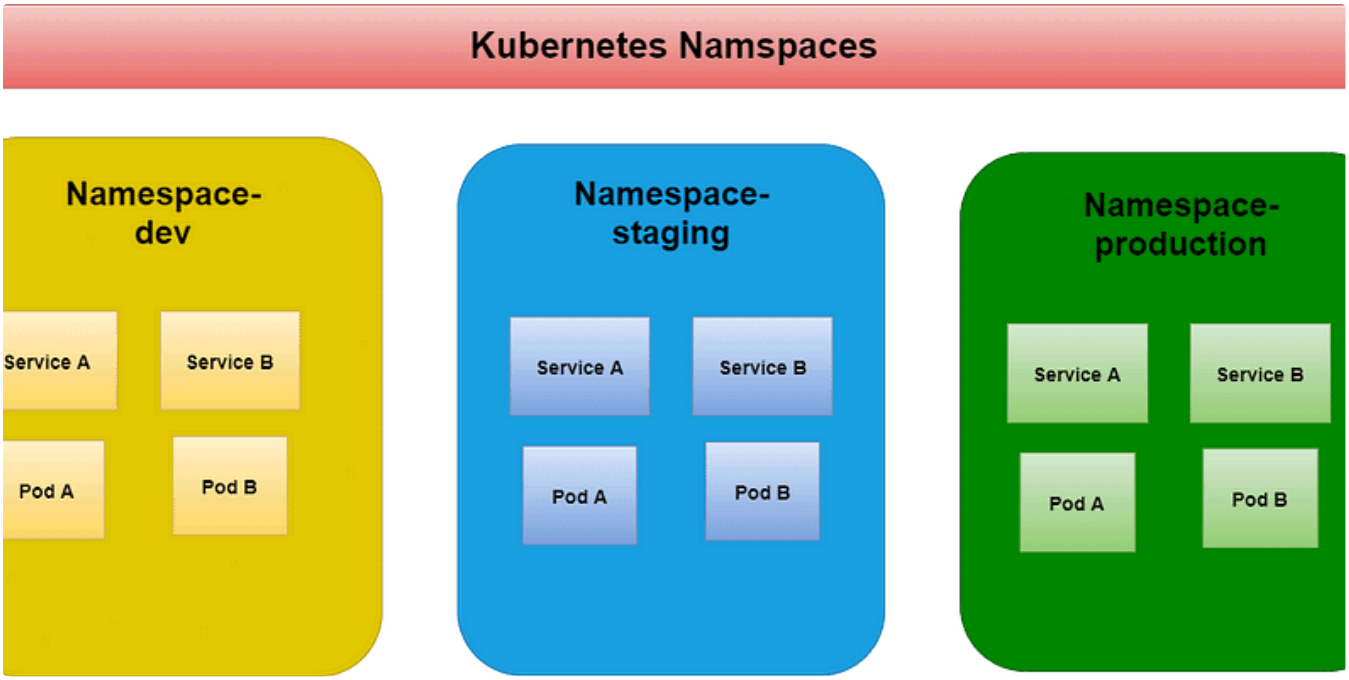
Introduction

2 min read · Nov 7, 2023

2

See all from Hakan Bayraktar

Recommended from Medium

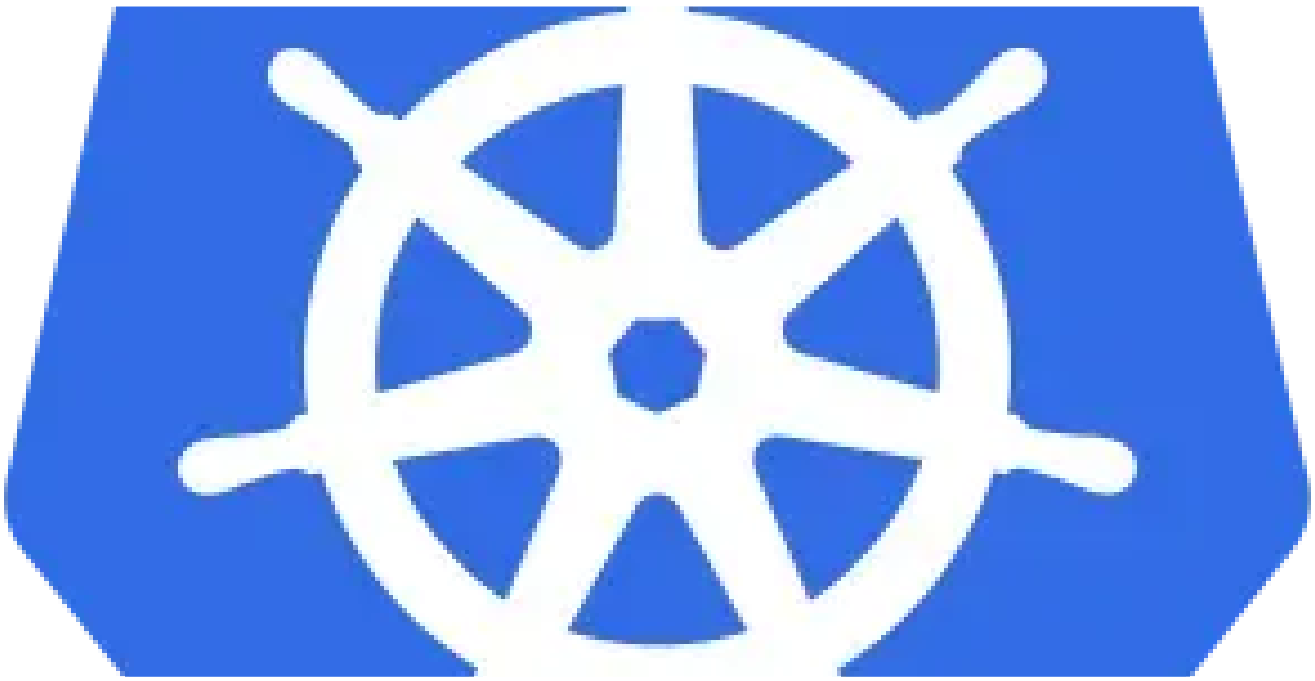


Ravi Patel

Understanding Kubernetes Namespaces

Kubernetes has revolutionized the way we deploy, scale, and manage containerized applications. Among its many features, Namespaces stand...

3 min read · Mar 28, 2024



Lubomir Tobek

Kubernetes Multi-Master Node Cluster

Creating and operating a highly available Kubernetes cluster requires multiple Kubernetes control plane nodes and “Master Nodes”. To...

10 min read · Dec 13, 2023

13

Lists

Natural Language Processing

1489 stories · 1001 saves

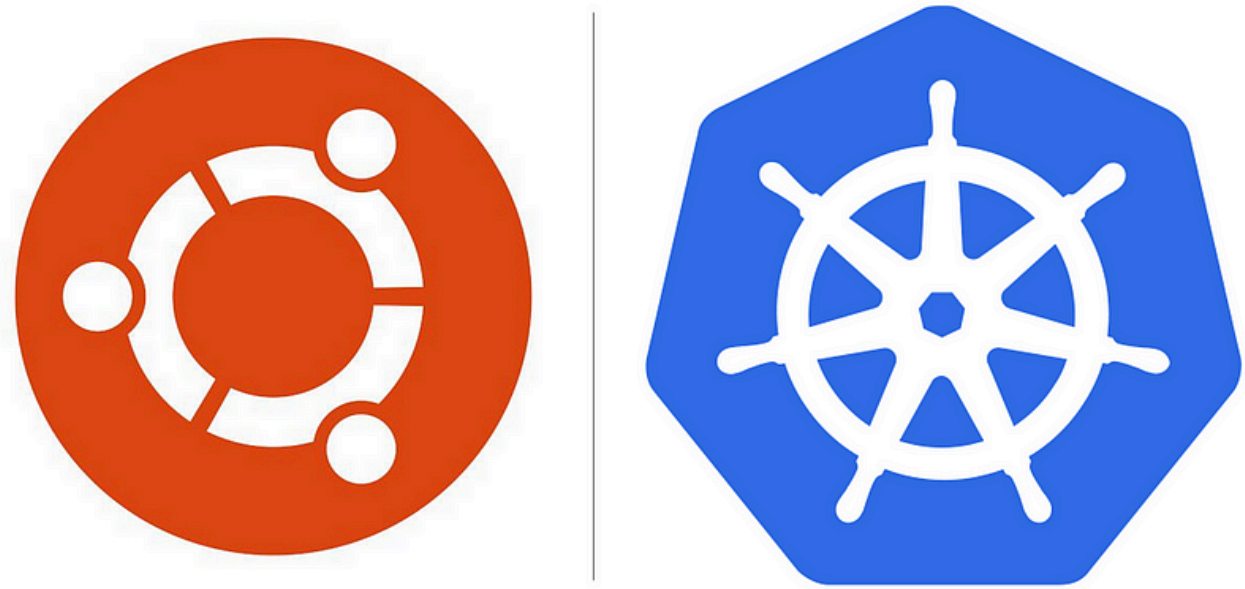
Ebube Ndubuisi

A Step-by-Step Guide to CI/CD with Jenkins, GitHub, and Kubernetes

What is CI/CD?

8 min read · Jan 16, 2024

8



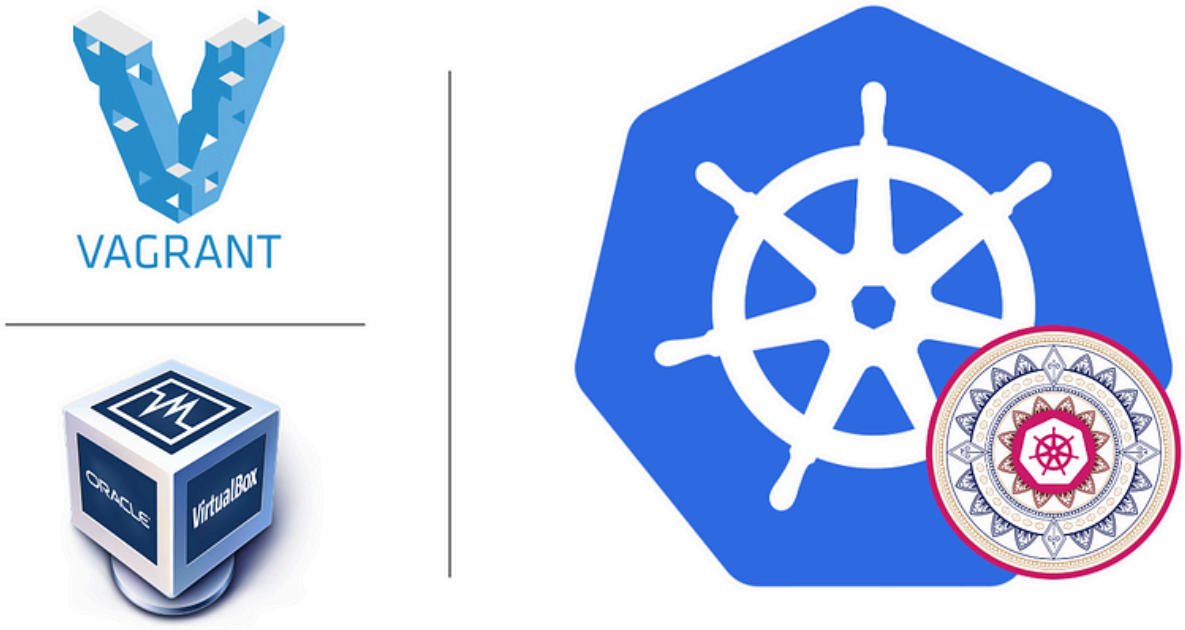
Ashish Singh in DevOps.dev

Installing Kubernetes on Ubuntu 22.04

Introduction

4 min read · Jan 10, 2024

304 2



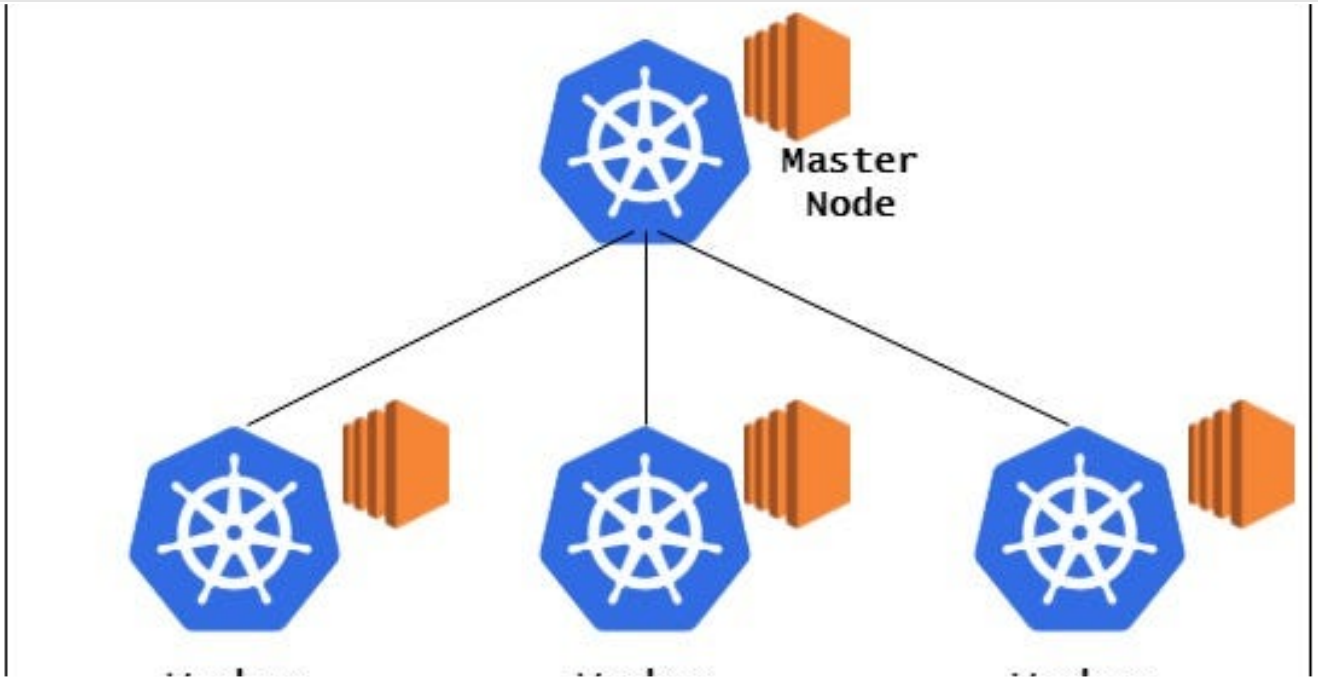
Akriotis Kyriakos in ITNEXT

Install Kubernetes 1.29 using Vagrant in under 10 minutes

Step by step installation of a Kubernetes 1.29 Cluster, with 1 master and 3 worker nodes, on Ubuntu virtual machines using Vagrant

9 min read · Feb 1, 2024

69



RAGHVENDRA TYAGI

Setup a Kubernetes Multi-Node Cluster on AWS: A Step-by-Step Guide

INTRODUCTION

7 min read · Mar 25, 2024

1

See more recommendations