



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



Hakan Bayraktar · Follow

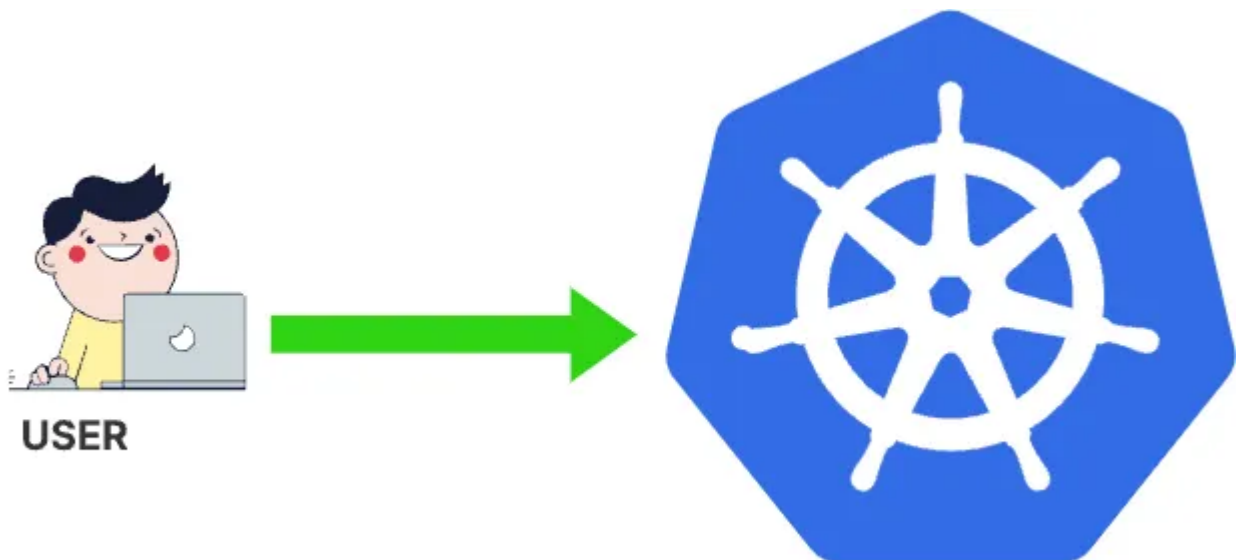
4 min read · Nov 18, 2023



Listen



Share



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 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.yaml
```

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**:

```
kubect 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:

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

```
root@hakan-yeni:~# kubect config view
No resources found
root@hakan-yeni:~# kubect config view
apiVersion: v1
clusters:
- cluster:
  certificate-authority-data: DATA+OMITTED
  server: https://104.248.28.87:6443
  name: kubernetes
contexts:
- context:
  cluster: kubernetes
  user: kubernetes-admin
  name: kubernetes-admin@kubernetes
current-context: kubernetes-admin@kubernetes
kind: Config
preferences: {}
users:
- name: kubernetes-admin
  user:
    client-certificate-data: DATA+OMITTED
    client-key-data: DATA+OMITTED
root@hakan-yeni:~# ls /etc/kubernetes/pki/
apiserver-etcd-client.crt  apiserver-kubelet-client.key  ca.crt  front-proxy-ca.crt  front-proxy-client.key
apiserver-etcd-client.key  apiserver.crt                ca.key  front-proxy-ca.key  sa.key
apiserver-kubelet-client.crt  apiserver.key                etcd    front-proxy-client.crt  sa.pub
```

I changed the command above according our cluster information.

```
# Set Cluster Configuration:
kubect 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=c
# Use Developer Context:
kubectl config use-context developer-context --kubeconfig=developer.kubeconfig
```

```
root@hakan-yeni:~/rbac# kubectl config set-cluster kubernetes --server=https://104.248.28.87:6443 --certificate-authority=/etc/kubernetes/pki/ca.crt --embed-certs=true --kubeconfig=developer.kubeconfig
Cluster "kubernetes" set.
root@hakan-yeni:~/rbac# kubectl config set-credentials developer --client-certificate=developer.crt --client-key=developer.key --embed-certs=true --kubeconfig=developer.kubeconfig
User "developer" set.
root@hakan-yeni:~/rbac# kubectl config set-context developer-context --cluster=kubernetes --namespace=default --user=developer --kubeconfig=developer.kubeconfig
Context "developer-context" created.
```

Verify the kubeconfig file's configuration:

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

```
root@hakan-yeni:~/rbac# kubectl --kubeconfig=developer.kubeconfig get pods
Error from server (Forbidden): pods is forbidden: User "developer" cannot list resource "pods" in API group "" in the namespace "default"
root@hakan-yeni:~/rbac#
```

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
```

```
root@hakan-yeni:~/rbac# kubectl --kubeconfig=developer.kubeconfig get pods -A
Error from server (Forbidden): pods is forbidden: User "developer" cannot list resource "pods" in API
group "" at the cluster scope
root@hakan-yeni:~/rbac#
```

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



```
join 146.190.135.86:6443 --token f1h95l.u4nkex9cw8d0g63w --discovery-token-ca-cert-hash sha256:8d1666af50c85f060b9fadc73f13c932e0e2a9eeef08f51f91a
# Run the following commands on the worker node to join the cluster...
# 1. Install the kubelet service
# 2. Configure the kubelet service
# 3. Start the kubelet service
# 4. Run the following command on the control-plane to see this node join the cluster.
```



Hakan Bayraktar

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

### Introduction

6 min read · Nov 3, 2023



269



13



The screenshot displays the AWS Management Console interface. At the top, there's a search bar and navigation tabs for 'N. Virginia' and 'cloud\_us'. The main content area is divided into several sections:

- Resources:** A table showing the usage of various Amazon EC2 resources in the US East (N. Virginia) Region. The resources and their counts are: Instances (running) - 0, Elastic IPs - 0, Load balancers - 0, Snapshots - 0, Auto Scaling Groups - 0, Instances - 0, Placement groups - 0, Volumes - 0, Dedicated Hosts - 0, Key pairs - 0, and Security groups - 1.
- Launch instance:** A section with a 'Launch instance' button and a 'Migrate a server' link.
- Service health:** A section with an 'AWS Health Dashboard' link.
- Account attributes:** A section showing the 'Default VPC' (vpc-0b889f3413dee8f3e) and 'Settings' for data protection and security, zones, EC2 Serial Console, default credit specification, and console experiments.
- Explore AWS:** A section with promotional text about saving up to 90% on EC2 with Spot Instances and getting up to 40% better price performance.



Hakan Bayraktar

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

## 6 min read · Nov 9, 2023

 $\omega^+$ 

```

emctl restart nfs-kernel-server
s-kernel-server
FS server and services
o/systemd/system/nfs-server.service; enabled) since Fri 2023-11-03 16:53:28 UTC; 4
StartPre=/usr/sbin/exportfs -r (code=exit
Start=/usr/sbin/rpc.nfsd (code=exited, st
e=exited, status=0/SUCCESS)

```



# How to Setup Dynamic NFS Provisioning in a Kubernetes Cluster

4 min read · Nov 3, 2023

 $\omega^+$



Hakan Bayraktar

## Merging Multiple kubeconfig Files into One: A Comprehensive Guide

Introduction

2 min read · Nov 7, 2023



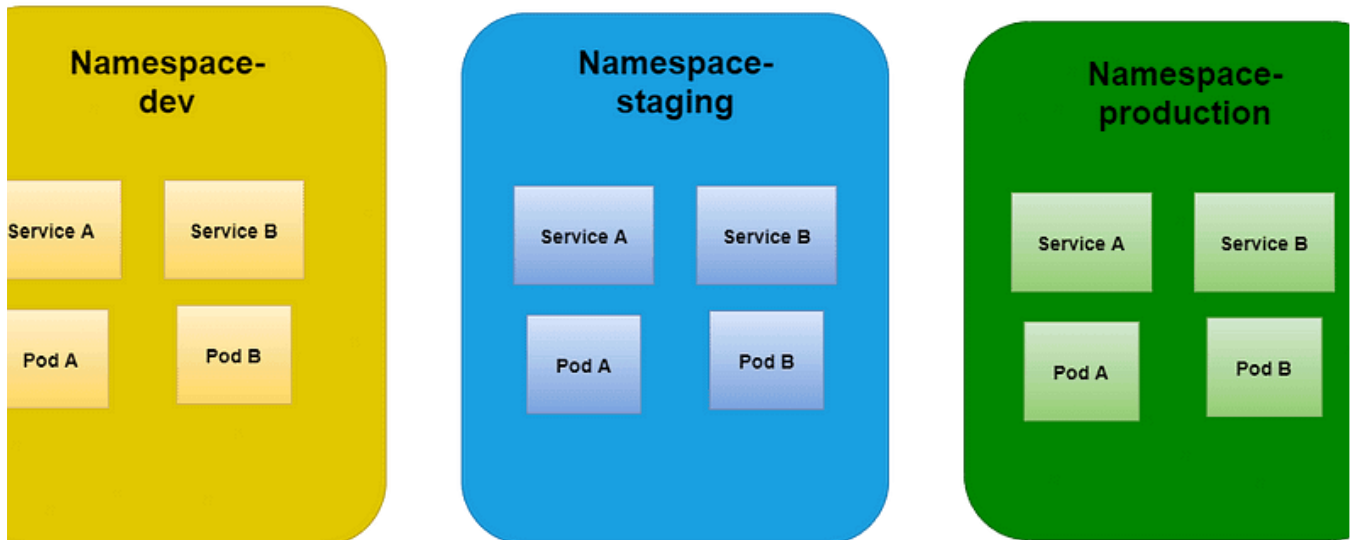
2



See all from Hakan Bayraktar

Recommended from Medium

## Kubernetes Namespaces



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



## 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





Ashish Singh in DevOps.dev

## Installing Kubernetes on Ubuntu 22.04

Introduction

4 min read · Jan 10, 2024



304



2



VAGRANT



Akriotis Kyriakos in ITNEXT

## Install Kubernetes 1.29 using Vagrant in under 10 minutes

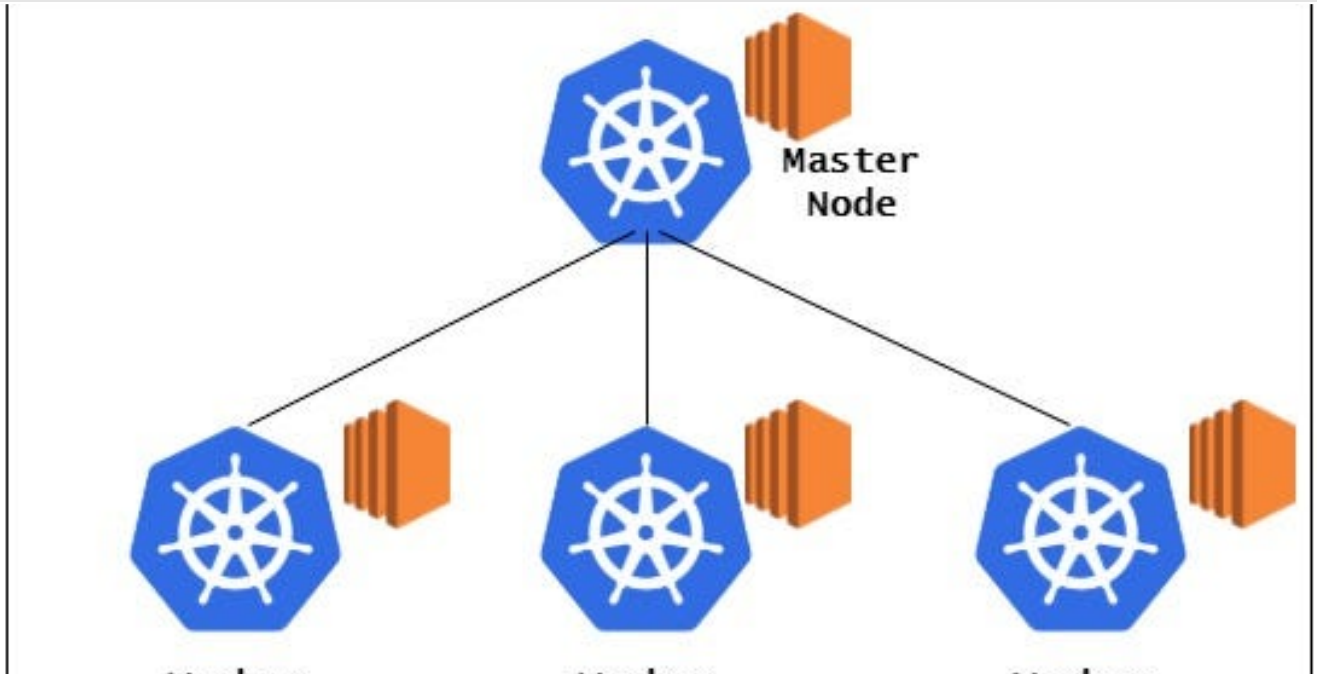


## Step by step installation of a Kubernetes 1.29 Cluster, with 1 master and 3 worker nodes, on Ubuntu virtual machines using Vagrant

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