Open in app ↗ Sign up Sign in

Medium Search

How to Install Kubernetes Cluster on Ubuntu 24.04 LTS (Step-by-Step Guide)

Introduction

Hakan Bayraktar · Follow

7 min read · Nov 2, 2023

Listen Share

Kubernetes is a powerful container orchestration platform used for automating the deployment, scaling, and management of containerized applications. In this guide, we will walk you through the step-by-step process of installing Kubernetes on Ubuntu 22.04. This cluster configuration includes a master node and worker nodes, allowing you to harness the full power of Kubernetes.

Kubernetes Nodes

In a Kubernetes cluster, you will encounter two distinct categories of nodes:

Master Nodes: These nodes play a crucial role in managing the control API calls for various components within the Kubernetes cluster. This includes overseeing pods, replication controllers, services, nodes, and more.

Worker Nodes: Worker nodes are responsible for providing runtime environments for containers. It's worth noting that a group of container pods can extend across multiple worker nodes, ensuring optimal resource allocation and management.

Prerequisites

Before diving into the installation, ensure that your environment meets the following prerequisites:

- An Ubuntu 24.04 LTS system.
- Privileged access to the system (root or sudo user).
- Active internet connection.

- Minimum 2GB RAM or more.
- Minimum 2 CPU cores (or 2 vCPUs).
- 20 GB of free disk space on /var (or more).

Step 1: Update and Upgrade Ubuntu (all nodes)

Begin by ensuring that your system is up to date. Open a terminal and execute the following commands:

```
sudo apt update && sudo apt upgrade -y
```

Step 2: Disable Swap (all nodes)

To enhance Kubernetes performance, disable swap and set essential kernel parameters. Run the following commands on all nodes to disable all swaps:

```
sudo swapoff -a
sudo sed -i '/ swap / s/^\(.*\)$/#\1/g' /etc/fstab
```

Step 3: Add Kernel Parameters (all nodes)

Load the required kernel modules on all nodes:

```
sudo tee /etc/modules-load.d/containerd.conf <<EOF
overlay
br_netfilter
EOF
sudo modprobe overlay
sudo modprobe br_netfilter
```

Configure the critical kernel parameters for Kubernetes using the following:

```
sudo tee /etc/sysctl.d/kubernetes.conf <<EOF
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1
EOF
```

Then, reload the changes:

```
sudo sysctl --system
```

Step 4: Install Containerd Runtime (all nodes)

We are using the containerd runtime. Install containerd and its dependencies with the following commands:

```
sudo apt install -y curl gnupg2 software-properties-common apt-transport-https c
```

Enable the Docker repository:

```
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmc
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubur
```

Update the package list and install containerd:

```
sudo apt update
sudo apt install -y containerd.io
```

Configure containerd to start using systemd as cgroup:

```
containerd config default | sudo tee /etc/containerd/config.toml >/dev/null 2>&1
sudo sed -i 's/SystemdCgroup \= false/SystemdCgroup \= true/g' /etc/containerd/c
```

Restart and enable the containerd service:

```
sudo systemctl restart containerd
sudo systemctl enable containerd
```

Step 5: Add Apt Repository for Kubernetes (all nodes)

Kubernetes packages are not available in the default Ubuntu 22.04 repositories. Add the Kubernetes repositories with the following commands:

```
echo "deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.30/deb/Release.key | sudo gpg --
```

Step 6: Install Kubectl, Kubeadm, and Kubelet (all nodes)

After adding the repositories, install essential Kubernetes components, including kubectl, kubelet, and kubeadm, on all nodes with the following commands:

```
sudo apt update
sudo apt install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl
```

Step 7: Initialize Kubernetes Cluster with Kubeadm (master node)

With all the prerequisites in place, initialize the Kubernetes cluster on the master node using the following Kubeadm command:

```
sudo kubeadm init
```

```
root@master:~# sudo kubeadm init
[init] Using Kubernetes version: v1.30.3
[preflight] Running pre-flight checks
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your inte
[preflight] You can also perform this action in beforehand using 'kubeadm config
W0810 16:57:59.292992   15792 checks.go:844] detected that the sandbox image "re
[certs] Using certificateDir folder "/etc/kubernetes/pki"
[certs] Generating "ca" certificate and key
[certs] Generating "apiserver" certificate and key
[certs] apiserver serving cert is signed for DNS names [kubernetes kubernetes.de
[certs] Generating "apiserver-kubelet-client" certificate and key
[certs] Generating "front-proxy-ca" certificate and key
[certs] Generating "front-proxy-client" certificate and key
[certs] Generating "etcd/ca" certificate and key
[certs] Generating "etcd/server" certificate and key
[certs] etcd/server serving cert is signed for DNS names [localhost master] and
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [localhost master] and IP
[certs] Generating "etcd/healthcheck-client" certificate and key
[certs] Generating "apiserver-etcd-client" certificate and key
[certs] Generating "sa" key and public key
[kubeconfig] Using kubeconfig folder "/etc/kubernetes"
[kubeconfig] Writing "admin.conf" kubeconfig file
[kubeconfig] Writing "super-admin.conf" kubeconfig file
[kubeconfig] Writing "kubelet.conf" kubeconfig file
[kubeconfig] Writing "controller-manager.conf" kubeconfig file
[kubeconfig] Writing "scheduler.conf" kubeconfig file
[etcd] Creating static Pod manifest for local etcd in "/etc/kubernetes/manifests
[control-plane] Using manifest folder "/etc/kubernetes/manifests"
[control-plane] Creating static Pod manifest for "kube-apiserver"
[control-plane] Creating static Pod manifest for "kube-controller-manager"
[control-plane] Creating static Pod manifest for "kube-scheduler"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/ku
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.y
[kubelet-start] Starting the kubelet
[wait-control-plane] Waiting for the kubelet to boot up the control plane as sta
[kubelet-check] Waiting for a healthy kubelet. This can take up to 4m0s
[kubelet-check] The kubelet is healthy after 1.004099735s
[api-check] Waiting for a healthy API server. This can take up to 4m0s
[api-check] The API server is healthy after 6.502017982s
[upload-config] Storing the configuration used in ConfigMap "kubeadm-config" in
[kubelet] Creating a ConfigMap "kubelet-config" in namespace kube-system with th
[upload-certs] Skipping phase. Please see --upload-certs
[mark-control-plane] Marking the node master as control-plane by adding the labe
[mark-control-plane] Marking the node master as control-plane by adding the tair
[bootstrap-token] Using token: 72ww2b.6orffywqcf5s4p2z
[bootstrap-token] Configuring bootstrap tokens, cluster-info ConfigMap, RBAC Rol
[bootstrap-token] Configured RBAC rules to allow Node Bootstrap tokens to get nc
[bootstrap-token] Configured RBAC rules to allow Node Bootstrap tokens to post C
[bootstrap-token] Configured RBAC rules to allow the csrapprover controller autc
[bootstrap-token] Configured RBAC rules to allow certificate rotation for all nc
```

```
[bootstrap-token] Creating the "cluster-info" ConfigMap in the "kube-public" nam
[kubelet-finalize] Updating "/etc/kubernetes/kubelet.conf" to point to a rotatab
[addons] Applied essential addon: CoreDNS
[addons] Applied essential addon: kube-proxy

Your Kubernetes control-plane has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config

Alternatively, if you are the root user, you can run:

  export KUBECONFIG=/etc/kubernetes/admin.conf

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
  https://kubernetes.io/docs/concepts/cluster-administration/addons/

Then you can join any number of worker nodes by running the following on each as

kubeadm join 138.197.184.45:6443 --token 72ww2b.6orffywqcf5s4p2z \
        --discovery-token-ca-cert-hash sha256:aafb79cdd45a6e3b3fac01fb3efba08173
root@master:~#  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

After the initialization is complete make a note of the `kubeadm join` command for future reference.

Run the following commands on the master node:

```
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

Next, use `kubectl` commands to check the cluster and node status:

```
kubectl get nodes
```

```
root@master:~# kubectl get nodes
NAME     STATUS   ROLES           AGE     VERSION
master   Ready    control-plane   2m47s   v1.30.3
```

Step 8: Add Worker Nodes to the Cluster (worker nodes)

On each worker node, use the `kubeadm join` command you noted down earlier:

```
kubeadm join 138.197.184.45:6443 --token 72ww2b.6orffywqcf5s4p2z \
        --discovery-token-ca-cert-hash sha256:aafb79cdd45a6e3b3fac01fb3efba08173
```

```
root@worker:~# kubeadm join 146.190.135.86:6443 --token f1h95l.u4nkex9cw8d0g63w        --discovery-token-ca-cert-has
h sha256:6d15f2a79bdb38d1666af50c85f060b9fadc73f13c932e0e2a9eeef08f51f91a
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.
```

Step :9 Install Kubernetes Network Plugin (master node)

To enable communication between pods in the cluster, you need a network plugin. Install the Calico network plugin with the following command from the master node:

```
kubectl apply -f https://raw.githubusercontent.com/projectcalico/calico/v3.25.0/
```

Step 10: Verify the cluster and test (master node)

Finally, we want to verify whether our cluster is successfully created.

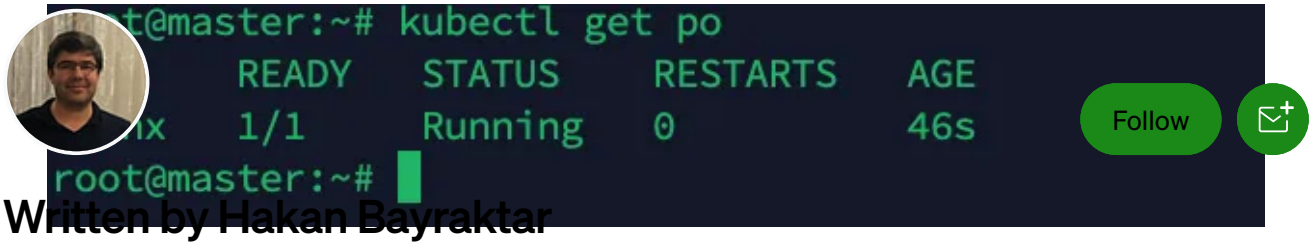
```
kubectl get pods -n kube-system
kubectl get nodes
```

```
root@master:~# kubectl get nodes
NAME     STATUS   ROLES           AGE   VERSION
master   Ready    control-plane   17m   v1.30.3
worker   Ready    <none>          65s   v1.30.3
root@master:~# kubectl get pods -n kube-system
NAME                                       READY   STATUS    RESTARTS   AGE
calico-kube-controllers-5b9b456c66-dkvq5   1/1     Running   0          15m
calico-node-q8x86                          1/1     Running   0          68s
calico-node-rtllk                          1/1     Running   0          15m
coredns-7db6d8ff4d-ctdfh                   1/1     Running   0          17m
coredns-7db6d8ff4d-m6wxt                   1/1     Running   0          17m
etcd-master                                1/1     Running   0          17m
kube-apiserver-master                      1/1     Running   0          17m
kube-controller-manager-master             1/1     Running   0          17m
kube-proxy-756dt                           1/1     Running   0          17m
kube-proxy-qih4n                           1/1     Running   0          68s
```

Kubernetes Cluster | Ubuntu | Kubernetes Installation | Kubeadm | Containers

Step 11: Deploy test application on cluster (master node)

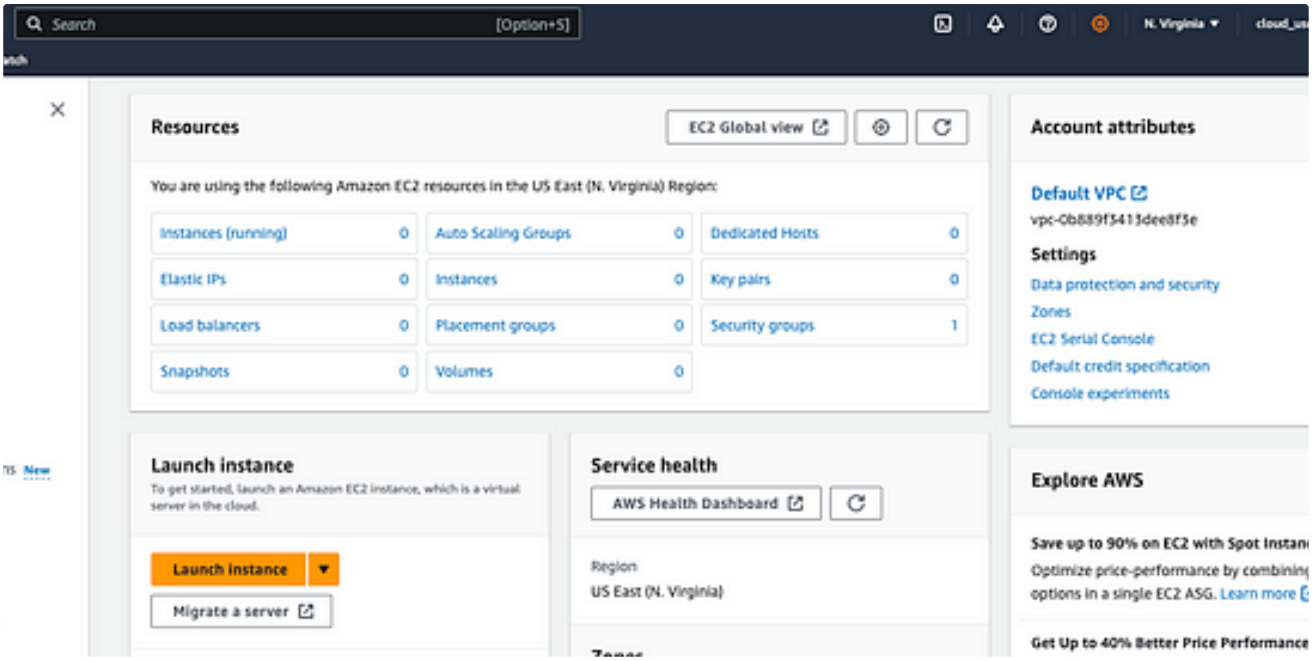
```
kubectl run nginx --image=nginx
```



Written by Hakan Bayraktar

288 Followers

More from Hakan Bayraktar

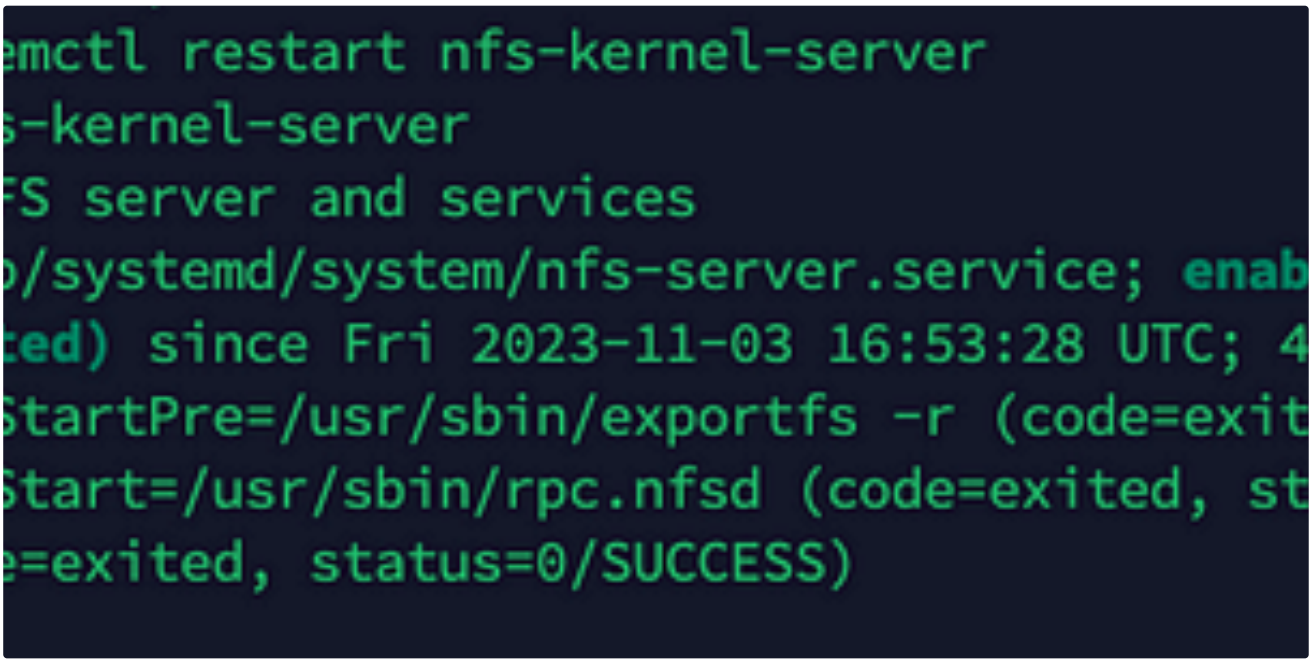


Hakan Bayraktar

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

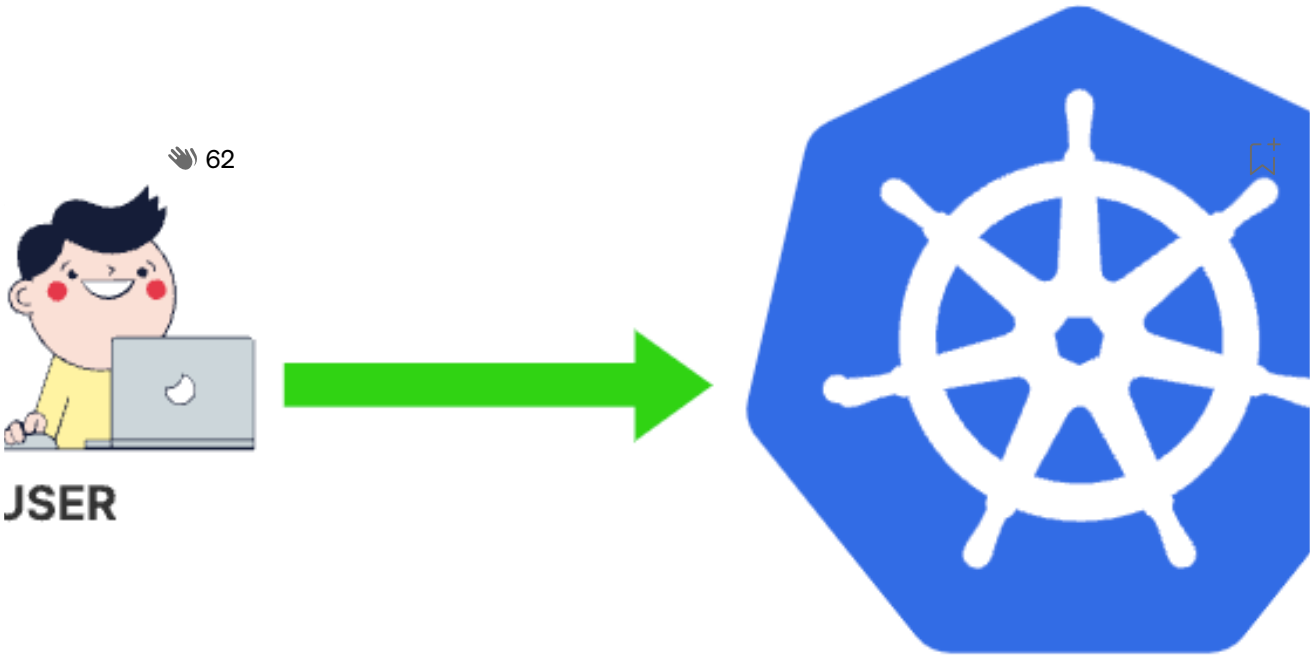
Introduction

Nov 9, 2023 85 4



Hakan Bayraktar

How to Setup Dynamic NFS Provisioning in a Kubernetes Cluster



Hakan Bayraktar

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

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

Nov 17, 2023 20

Hakan Bayraktar

Merging Multiple kubeconfig Files into One: A Comprehensive Guide

Introduction

Recommended from Medium

Nov 6, 2023 4

Amazon.com Seattle, WA
Software Development Engineer Mar. 2020 – May 2021

- Developed Amazon checkout and payment services to handle traffic of 10 Million daily global transactions
- Integrated Iframes for credit cards and bank accounts to secure 80% of all consumer traffic and prevent CSRF, cross-site scripting, and cookie-jacking
- Led Your Transactions implementation for JavaScript front-end framework to showcase consumer transactions and reduce call center costs by $25 Million
- Recovered Saudi Arabia checkout failure impacting 4000+ customers due to incorrect GET form redirection

Projects

NinjaPrep.io (React)
- Platform to offer coding problem practice with built in code editor and written + video solutions in React
- Utilized Nginx to reverse proxy IP address on Digital Ocean hosts
- Developed using Styled-Components for 95% CSS styling to ensure proper CSS scoping
- Implemented Docker with Seccomp to safely run user submitted code with < 2.2s runtime

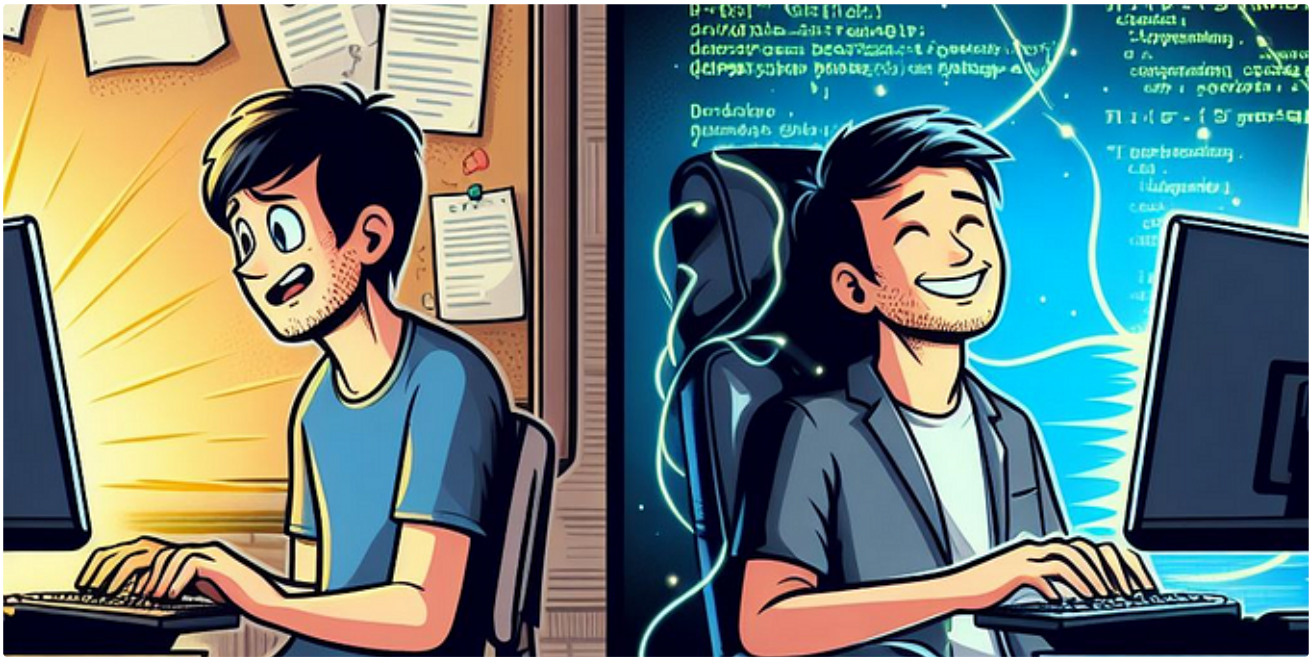
HeatMap (JavaScript)
- Visualized Google Takeout location data of location history using Google Maps API and Google Maps heatmap code with React
- Included local file system storage to reliably handle 5mb of location history data
- Implemented Express to include routing between pages and jQuery to parse Google Map and implement heatmap overlay

Alexander Nguyen in Level Up Coding

The resume that got a software engineer a $300,000 job at Google.

1-page. Well-formatted.

Jun 1 17.4K 274



Abhay Parashar in The Pythoneers

17 Mindblowing Python Automation Scripts I Use Everyday

Scripts That Increased My Productivity and Performance

Jul 29 6.3K 51

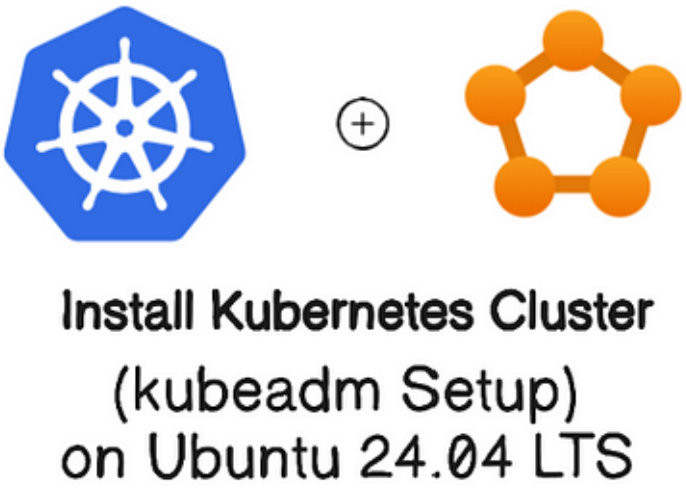
Lists

Staff Picks
713 stories · 1221 saves

Stories to Help You Level-Up at Work
19 stories · 739 saves

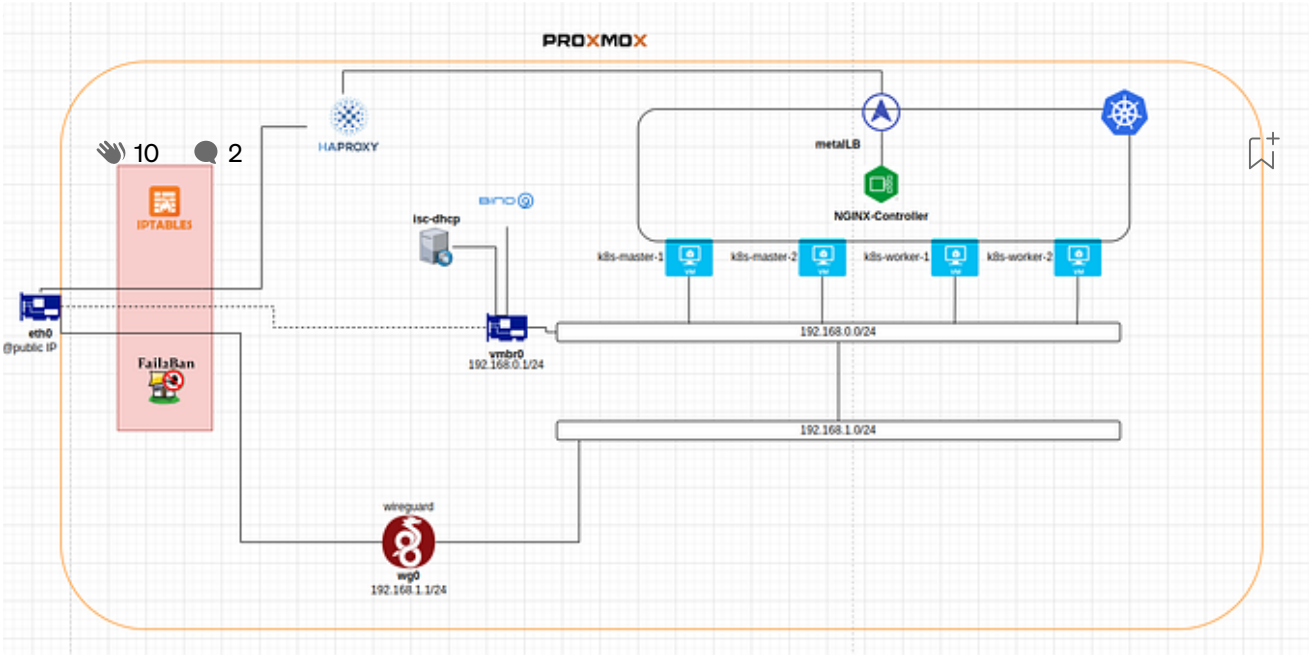
Self-Improvement 101
20 stories · 2529 saves

Productivity 101
20 stories · 2195 saves



Subham Pradhan

How to Install Kubernetes Cluster (kubeadm Setup) on Ubuntu 24.04 LTS (Step-by-Step Guide)



Genesta Sebastien

Kubernetes — Kubernetes cluster deployment on Proxmox 8 — Part 1 — Proxmox deployment on Debian 12...

we'll talked about Proxmox deployment and configuration on a Debian 12 dedicated server (iptables, fail2ban, isc-dhcp, bind9, wireguard)

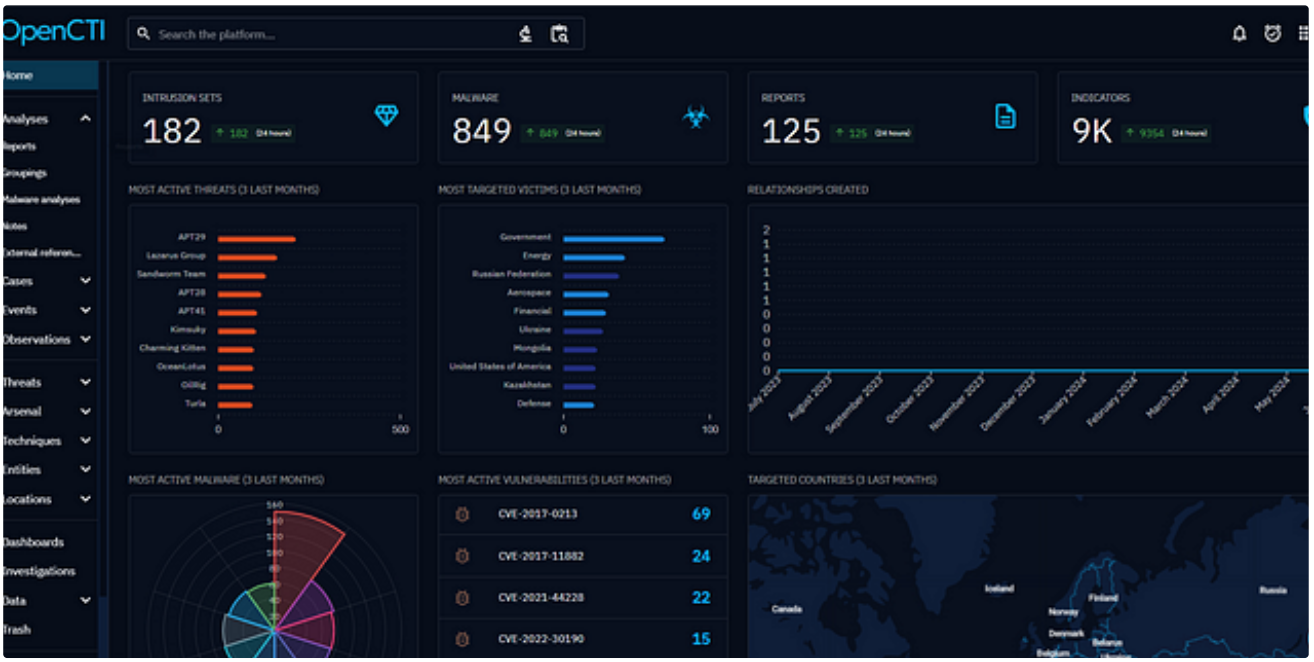
May 9 21

Amir Ad

How to join master node or control plane to Kubernetes cluster

You might want to add a new master node to your new or existing Kubernetes cluster or change a worker node role to master but do not know...

May 21 9



Yogasatriautama

SOC: Install OpenCTI

OpenCTI (Open Cyber Threat Intelligence) is an open-source platform designed to collect, store, and utilize cyber threat data. It helps...

Jul 22 80 1

See more recommendations