

# Installation k3s ec2 ubuntu.20.04

Tuesday, November 29, 2022 12:02 AM

Launch 2 instances with Amazon Ubuntu 20.04 image, t3.medium (2CPU and 4GI) one will be master, one will be the worker

Open port TCP 6443 to anywhere and port TCP TCP 22 to MyIP in the security group

Update both the instances with "sudo apt update -y" and "sudo apt upgrade -y" commands

Reboot the instances with "sudo systemctl reboot"

Change the master node's hostname with "sudo hostname master&&bash"

Change the worker node's hostname with "sudo hostname worker&&bash"

Only after performing the steps mentioned above proceed with the following:

## MASTER NODE K3S SERVER INSTALLATION

Connect via ssh to the master node with "ssh -i <pem-key> ubuntu@master-ip

Establish in which mode to write k3s configuration when not running as root with ; export K3S\_KUBECONFIG\_MODE="644"

Run installer with " curl -sfl <https://get.k3s.io> | sh -

Verify the status ; sudo systemctl status k3s

Get details on the " nodes with kubectl get nodes -o wide "

Save access token (we need it in the nex steps when setting worker node) with: sudo cat /var/lib/rancher/k3s/server/node-token

## Worker NODE k3s INSTALLATION

Connect via ssh to the worker node with "ssh -i <pem-key> ubuntu@worker-ip

Configure environment variables with : export K3S\_KUBECONFIG\_MODE="644"; export

K3S\_URL="https:master\_private\_ip:6443", export K3S\_TOKEN="....."

Or You can use this command only; run the installer with "curl -sfl <https://get.k3s.io> |

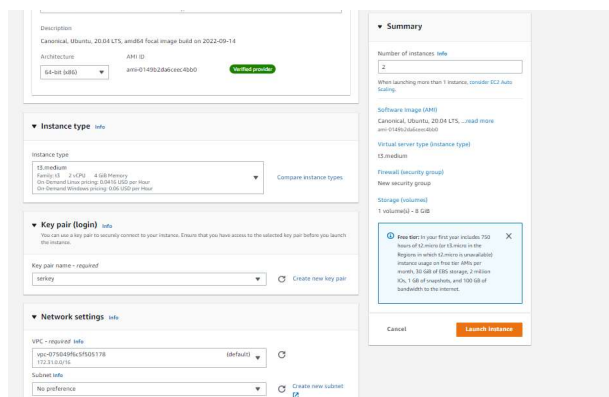
K3S\_URL=https://<master\_private\_ip>:6443 K3S\_TOKEN="....." sh -

Verify the status with sudo systemctl status k3s-agent

Go to master node and make sure the node has been added : kubectl get nodes -o wide

Lets GO

Launch 2 instances with Amazon Ubuntu 20.04 image, t3.medium (2CPU and 4GI) one will be master, one will be the worker



Open port TCP 6443 to anywhere and port TCP TCP 22 to MyIP in the security group

Security group name - required

k3s-ec2-sg

This security group will be added to all server instances. The name can't be added after the security group is created. Max length is 255 characters. Valid characters are a-z 0-9 hyphen and . (eg sg-12345678)

Description - optional info

k3s

Inbound security group rules

Security group rule 1 (TCP: 22, 0.0.0.0/0)

Type info Protocol info Port range info

ssh TCP 22

Source type info Source info Description - optional info

Anywhere Anywhere Add CIDR prefix list or security e.g. 10.0.0.0/16 for admin desktop

Security group rule 2 (TCP: 6443, 0.0.0.0/0)

Type info Protocol info Port range info

Custom TCP 6443

Source type info Source info Description - optional info

Custom Anywhere Add CIDR prefix list or security e.g. 10.0.0.0/16 for admin desktop

Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

## MASTER NODE K3S SERVER INSTALLATION

Connect via ssh to the master node with "ssh -i <pem-key> ubuntu@master-ip

Update both the instances with "sudo apt update -y" and "sudo apt upgrade -y " commands

```
ubuntu@ip-172-31-12-168:~$ sudo apt update -y | sudo apt upgrade -y
WARNING: apt does not have a stable CLI interface. Use with caution in scripts.

Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
```

Reboot the instances with "sudo systemctl reboot"

Change the master node's hostname with "sudo hostname master&&bash"

Establish in which mode to write k3s configuration when not running as root with ; export K3S\_KUBECONFIG\_MODE="644"

Run installer with " curl -sfl <https://get.k3s.io> | sh -

Verify the status ; sudo systemctl status k3s

```
ubuntu@ip-172-31-12-168:~$ sudo hostname master
ubuntu@ip-172-31-12-168:~$ bash
ubuntu@master:~$ curl -sfl https://get.k3s.io | sh -
[INFO] Finding release for channel stable
[INFO] Using v1.25.4+k3s1 as release
[INFO] Downloading hash https://github.com/k3s-io/k3s/releases/download/v1.25.4+k3s1/sha256sum-amd64.txt
[INFO] Downloading binary https://github.com/k3s-io/k3s/releases/download/v1.25.4+k3s1/k3s
[INFO] Verifying binary download
[INFO] Installing k3s to /usr/local/bin/k3s
[INFO] Skipping installation of SELinux RPM
[INFO] Creating /usr/local/bin/kubectll symlink to k3s
[INFO] Creating /usr/local/bin/crictl symlink to k3s
[INFO] Creating /usr/local/bin/ctr symlink to k3s
[INFO] Creating killall script /usr/local/bin/k3s-killall.sh
[INFO] Creating uninstall script /usr/local/bin/k3s-uninstall.sh
[INFO] env: Creating environment file /etc/systemd/system/k3s.service.env
[INFO] systemd: Creating service file /etc/systemd/system/k3s.service
[INFO] systemd: Enabling k3s unit
Created symlink /etc/systemd/system/multi-user.target.wants/k3s.service → /etc/systemd/system/k3s.service.
[INFO] systemd: Starting k3s
ubuntu@master:~$ sudo systemctl status k3s
● k3s.service - Lightweight Kubernetes
   Loaded: loaded (/etc/systemd/system/k3s.service; enabled; vendor preset: enabled)
   Active: active (running) since Tue 2022-11-29 19:00:18 UTC; 11s ago
```

```
Docs: https://k3s.io
Process: 19090 ExecStartPre=/bin/sh -xc ! /usr/bin/systemctl is-enabled --quiet nm-cloud-setup.service (code=exited, status=0/SUCCESS)
Process: 19097 ExecStartPre=/sbin/modprobe br_netfilter (code=exited, status=0/SUCCESS)
Process: 19099 ExecStartPre=/sbin/modprobe overlay (code=exited, status=0/SUCCESS)
Main PID: 19144 (k3s-server)
Tasks: 36
Memory: 573.1M
CGroup: /system.slice/k3s.service
└─19144 /usr/local/bin/k3s server
   └─19934 containerd -c /var/lib/rancher/k3s/agent/etc/containerd/config.toml --root /run/k3s/containerd --state /var/lib/rancher/k3s/agent/containerd

Nov 29 19:00:25 master k3s[19144]: 11129 19:00:25.284593 19144 configmap.cattle.io:202] "Starting controller" name="client-ca:kube-system:extension-apiserver-authentication:client-ca-file"
Nov 29 19:00:25 master k3s[19144]: 11129 19:00:25.284579 19144 shared_informer.go:255] Waiting for caches to sync for client-ca:kube-system:extension-apiserver-authentication:client-ca-file
Nov 29 19:00:25 master k3s[19144]: 11129 19:00:25.284656 19144 configmap.cattle.io:202] "Starting controller" name="client-ca:kube-system:extension-apiserver-authentication:requestheader-client-ca-file"
Nov 29 19:00:25 master k3s[19144]: 11129 19:00:25.284740 19144 shared_informer.go:255] Waiting for caches to sync for client-ca:kube-system:extension-apiserver-authentication:requestheader-client-ca-file
Nov 29 19:00:25 master k3s[19144]: 11129 19:00:25.285558 19144 secure_serving.go:210] Serving securely on 127.0.0.1:8029
Nov 29 19:00:25 master k3s[19144]: 11129 19:00:25.285628 19144 tlsconfig.go:240] Starting DynamicServingCertificateController"
Nov 29 19:00:25 master k3s[19144]: 11129 19:00:25.385118 19144 shared_informer.go:262] Caches are synced for client-ca:kube-system:extension-apiserver-authentication:requestheader-client-ca-file
Nov 29 19:00:25 master k3s[19144]: 11129 19:00:25.385118 19144 shared_informer.go:262] Caches are synced for RequestHeaderAuthRequestController
Nov 29 19:00:25 master k3s[19144]: 11129 19:00:25.385141 19144 shared_informer.go:262] Caches are synced for client-ca:kube-system:extension-apiserver-authentication:client-ca-file
Nov 29 19:00:28 master k3s[19144]: 11129 19:00:28.439884 19144 controller.go:436] quota admission added evaluator For: leases.coordination.k8s.io
lines 1-24]
```

Get details on the " nodes with kubectll get nodes -o wide "

```
ubuntu@master:~$ kubectll get nodes -o wide
Warning: Unable to read /etc/rancher/k3s/k3s.yaml, please start server with --write-kubeconfig-mode to modify kube config permissions
error: error loading config file "/etc/rancher/k3s/k3s.yaml": open /etc/rancher/k3s/k3s.yaml: permission denied
ubuntu@master:~$ sudo chown 777 /etc/rancher/k3s/k3s.yaml
ubuntu@master:~$ kubectll get nodes -o wide
NAME STATUS ROLES AGE VERSION INTERNAL-IP EXTERNAL-IP OS-IMAGE KERNEL-VERSION CONTAINER-RUNTIME
master Ready control-plane,master 110s v1.25.4+k3s1 172.31.12.168 <none> Ubuntu 20.04.5 LTS 5.15.0-1019-aws containerd://1.6.8-k3s1
```

Save access token (we need it in the nex steps when setting worker node) with: sudo cat /var/lib/rancher/k3s/server/node-token

```
ubuntu@master:~$ sudo cat /var/lib/rancher/k3s/server/node-token
K10394fdb5abfdb46c803e75fdcd08da861beb601e98ec7aa4285bc1f5c963491f2::server:5f8cd029e125204617b7f6f61fbc2e87
```

## Worker

Connect via ssh to the worker node with "ssh -i <pem-key> ubuntu@worker-ip

```
$ ssh -i "key.pem" ubuntu@ec2-54-91-38-250.compute-1.amazonaws.com
The authenticity of host 'ec2-54-91-38-250.compute-1.amazonaws.com (54.91.38.250)' can't be established.
ED25519 key fingerprint is SHA256:1AnhejkK19Mnvc3ryG1DUq8q07Lagbq3Fv4.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-54-91-38-250.compute-1.amazonaws.com' (ED25519) to the list of known hosts.
Welcome to Ubuntu 20.04.5 LTS (GNU/Linux 5.15.0-1019-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

System information as of Tue Nov 29 19:03:45 UTC 2022

System load: 0.0          Processes:            181
Usage of /: 19.6% of 7.57GB Users logged in:      0
Memory usage: 5%          IPv4 address for ens5: 172.31.1.170
Swap usage: 0%

0 updates can be applied immediately.
```

```
ubuntu@ip-172-31-1-170:~$ sudo hostname worker
ubuntu@ip-172-31-1-170:~$ bash
ubuntu@worker:~$
```

run the installer with "curl -sfl <https://get.k3s.io> | K3S\_URL=https://<master\_private\_ip>:6443 K3S\_TOKEN="...." sh -

Verify the status with sudo systemctl status k3s-agent

```
ubuntu@worker:~$ curl -sfl http://get.k3s.io | K3S_URL=https://172.31.12.108:6443 K3S_TOKEN="K10394fdb5abfdb46c803e75fdcd08da861beb601e98ec7aa4285bc1f5c963491f2::server:5f8cd029e125204617b7f6f61fbc2e87" sh -
[INFO] Finding release for channel stable
[INFO] Using v1.25.4+k3s1 as release
[INFO] Downloading bash https://github.com/k3s-io/k3s/releases/download/v1.25.4+k3s1/sha256sum-amd64.txt
[INFO] Downloading binary https://github.com/k3s-io/k3s/releases/download/v1.25.4+k3s1/k3s
[INFO] Verifying binary download
[INFO] Installing k3s to /usr/local/bin/k3s
[INFO] Skipping installation of SELinux RPM
[INFO] Creating /usr/local/bin/kubect1 symlink to k3s
[INFO] Creating /usr/local/bin/crictl symlink to k3s
[INFO] Creating /usr/local/bin/ctr symlink to k3s
[INFO] Creating killall script /usr/local/bin/k3s-killall.sh
[INFO] Creating uninstall script /usr/local/bin/k3s-agent-uninstall.sh
[INFO] env: Creating environment file /etc/systemd/system/k3s-agent.service.env
[INFO] system: Creating service file /etc/systemd/system/k3s-agent.service
[INFO] system: Enabling k3s-agent unit
Created symlink /etc/systemd/system/multi-user.target.wants/k3s-agent.service → /etc/systemd/system/k3s-agent.service.
[INFO] system: Starting k3s-agent
ubuntu@worker:~$ sudo systemctl status k3s-agent
● k3s-agent.service - Lightweight Kubernetes
   Loaded: loaded (/etc/systemd/system/k3s-agent.service; enabled; vendor preset: enabled)
   Active: active (running) since Tue 2022-11-29 19:07:46 UTC; 11s ago
   Docs: https://k3s.io
```

Go to master node and make sure the node has been added : kubectl get nodes -o wide

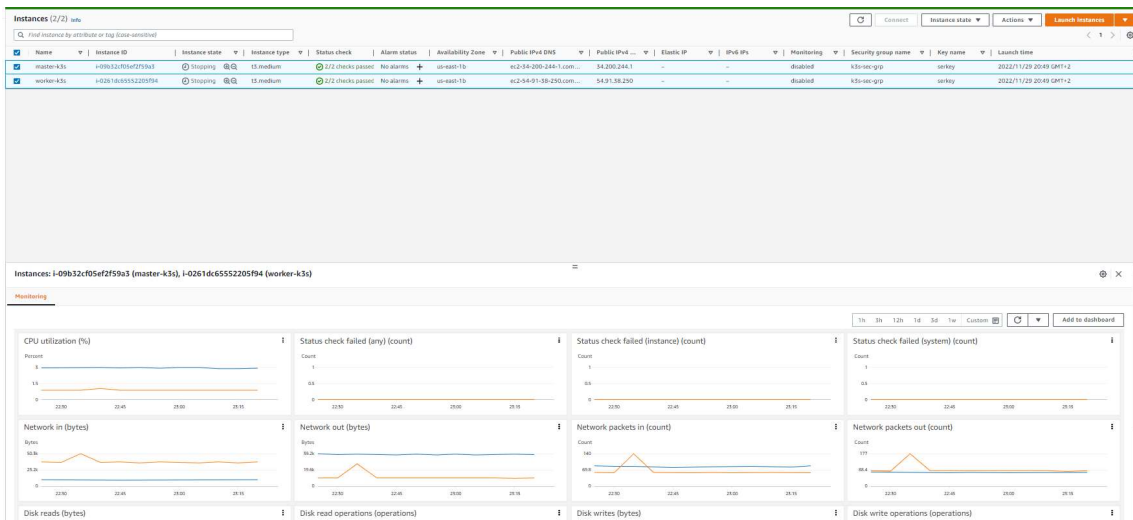
```
ubuntu@master:~$ kubectl get nodes -o wide
NAME        STATUS    ROLES    AGE   VERSION   INTERNAL-IP   EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION   CONTAINER-RUNTIME
master      Ready     control-plane,master   79s   v1.25.4+k3s1   172.31.12.108   <none>         Ubuntu 20.04.5 LTS   5.15.0-1019-aws   containerd://1.6.8-k3s1
worker      Ready     <none>    79s   v1.25.4+k3s1   172.31.1.170   <none>         Ubuntu 20.04.5 LTS   5.15.0-1019-aws   containerd://1.6.8-k3s1
```

## Master node: TOP command

```
top - 19:09:40 up 19 min, 1 user, load average: 0.02, 0.11, 0.14
Tasks: 122 total, 1 running, 121 sleeping, 0 stopped, 0 zombie
Cpu(s): 1.5 us, 0.8 sy, 0.0 ni, 95.9 id, 0.2 wa, 0.0 hi, 0.0 si, 1.7 st
Mem Mem : 3864.0 total, 590.4 free, 728.3 used, 2945.1 buff/cache
Mem Swap: 0.0 total, 0.0 free, 0.0 used, 2842.8 avail Mem

  PID USER      PR  NI    TST      RES     SHR   S%CPU   %MEM    TIME+  COMMAND
19914 root        20   0 1332484 480256 102992  5  4.7  12.1  0:39.90 k3s-server
19933 root        20   0 770480 63512 39088  5  1.0  1.6  0:09.75 containerd
21854 ubuntu     20   0 721956 47460 34208  5  0.7  1.2  0:04.18 metrics-server
20640 root        20   0 720748 10416 8168  5  0.3  0.3  0:00.22 containerd-shim
20664 root        20   0 720748 10004 7848  5  0.3  0.3  0:00.26 containerd-shim
21792 root        20   0 720692 10348 7336  5  0.3  0.3  0:00.45 containerd-shim
1 root        20   0 171344 13016 8916  5  0.0  0.3  0:00.00 systemd
2 root        20   0 0 0 0 0 0.5 0.0 0.0 0:00.00 kthreadd
3 root        20   0 0 0 0 0 0.1 0.0 0.0 0:00.00 rcu_gp
4 root        20   0 0 0 0 0 0.1 0.0 0.0 0:00.00 rcu_pae_gp
5 root        20   0 0 0 0 0 0.1 0.0 0.0 0:00.00 netns
7 root        20   0 0 0 0 0 0.1 0.0 0.0 0:00.00 kworker/0:0-events_highpri
8 root        20   0 0 0 0 0 0.0 0.0 0:00.43 kworker/0:0-events_power_efficient
9 root        20   0 0 0 0 0 0.1 0.0 0.0 0:00.00 mm_percpu_wq
10 root        20   0 0 0 0 0 0.5 0.0 0.0 0:00.00 rcu_tasks_rude
11 root        20   0 0 0 0 0 0.5 0.0 0.0 0:00.00 rcu_tasks_trace
12 root        20   0 0 0 0 0 0.5 0.0 0.0 0:00.20 ksoftirqd/0
13 root        20   0 0 0 0 0 0.1 0.0 0.0 0:00.32 rcu_sched
```

## EC2- monitor



### Deploy Addons to K3s

- K3s is a lightweight kubernetes tool that doesn't come packaged with all the tools but you can install them separately.

#### # Install Helm Commandline tool on k3s

- Download the latest version of Helm commandline tool from this page.
- Extract the tar file using tar -xvzf <downloaded-file>
- Move the binary file to /usr/local/bin/helm
- sudo mv linux-amd64/helm /usr/local/bin/helm

#### # Check version

```
helm version
```

- Add the helm chart repository to allow installation of applications using helm:

```
helm repo add stable https://charts.helm.sh/stable
```

```
helm repo update
```

#### ### Demo Application Deploy

##### # Deploy Nginx Web-proxy on K3s

- Nginx can be used as a web proxy to expose ingress web traffic routes in and out of the cluster.
- We can install nginx web-proxy using Helm:

```
helm install nginx-ingress stable/nginx-ingress --namespace kube-system \
--set defaultBackend.enabled=false
```

- We can test if the application has been installed by:

```
kubectl get pods -n kube-system -l app=nginx-ingress -o wide
```

#### ### Step 9: Removing k3s

- After all the steps if you completed your task, you can remove k3s on the worker nodes

```
sudo /usr/local/bin/k3s-agent-uninstall.sh
```

```
sudo rm -rf /var/lib/rancher
```

- and master node:

```
sudo /usr/local/bin/k3s-uninstall.sh
```

```
sudo rm -rf /var/lib/rancher
```

### Adding addon

#### Helm installation to the k3s master

### Deploy Addons to K3s

- K3s is a lightweight kubernetes tool that doesn't come packaged with all the tools but you can install them separately.

#### # Install Helm Commandline tool on k3s

- Download the latest version of Helm commandline tool from this page.
- Extract the tar file using tar -xvzf <downloaded-file>
- Move the binary file to /usr/local/bin/helm
- sudo mv linux-amd64/helm /usr/local/bin/helm

#### # Check version

```
helm version
```

- Add the helm chart repository to allow installation of applications using helm:

```
helm repo add stable https://charts.helm.sh/stable
```

```
helm repo update
```

```
ubuntu@master:~$ curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3
ubuntu@master:~$ chmod 700 get_helm.sh
ubuntu@master:~$ ./get_helm.sh
Downloading https://get.helm.sh/helm-v3.10.2-linux-amd64.tar.gz
Verifying checksum... Done.
Preparing to install helm into /usr/local/bin
helm installed into /usr/local/bin/helm
ubuntu@master:~$ helm repo add stable https://charts.helm.sh/stable
"stable" has been added to your repositories
ubuntu@master:~$ helm repo update
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "stable" chart repository
Update Complete. Happy Helming!
```

```
ubuntu@master:~$ helm version
WARNING: Kubernetes configuration file is group-readable. This is insecure. Location: /etc/rancher/k3s/k3s.yaml
WARNING: Kubernetes configuration file is world-readable. This is insecure. Location: /etc/rancher/k3s/k3s.yaml
version.BuildInfo{Version:"v3.10.2", GitCommit:"50f003e5ee8704ec937a756c646870227d7c8b58", GitTreeState:"clean", GoVersion:"go1.18.8"}
```

Install nginx with helm

### ### Demo Application Deploy

#### # Deploy Nginx Web-proxy on K3s

- Nginx can be used as a web proxy to expose ingress web traffic routes in and out of the cluster.
- We can install nginx web-proxy using Helm:

```
helm install nginx-ingress stable/nginx-ingress --namespace kube-system \
--set defaultBackend.enabled=false
```

- We can test if the application has been installed by:

```
kubectl get pods -n kube-system -l app=nginx-ingress -o wide
```

```
ubuntu@master:~$ helm install nginx-ingress stable/nginx-ingress --namespace kube-system \
> --set defaultBackend.enabled=false
WARNING: This chart is deprecated
Error: INSTALLATION FAILED: Kubernetes cluster unreachable: Get "http://localhost:8080/version": dial tcp 127.0.0.1:8080: connect: connection refused
ubuntu@master:~$ helm repo add stable https://charts.helm.sh/stable --force-update
"stable" has been added to your repositories
ubuntu@master:~$ export KUBECONFIG=/etc/rancher/k3s/k3s.yaml
ubuntu@master:~$ sudo chmod 777 /etc/rancher/k3s/k3s.yaml
ubuntu@master:~$ helm install nginx-ingress stable/nginx-ingress --namespace kube-system --set defaultBackend.enabled=false
WARNING: Kubernetes configuration file is group-readable. This is insecure. Location: /etc/rancher/k3s/k3s.yaml
WARNING: Kubernetes configuration file is world-readable. This is insecure. Location: /etc/rancher/k3s/k3s.yaml
NAME: nginx-ingress
LAST DEPLOYED: Wed Nov 30 12:58:59 2022
NAMESPACE: kube-system
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
*****
* DEPRECATED, please use https://github.com/kubernetes/ingress-nginx/tree/master/charts/ingress-nginx *
*****

The nginx-ingress controller has been installed.
```

The nginx-ingress controller has been installed.

It may take a few minutes for the LoadBalancer IP to be available.

You can watch the status by running 'kubectl --namespace kube-system get services -o wide -w nginx-ingress-controller'

An example Ingress that makes use of the controller:

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  annotations:
    kubernetes.io/ingress.class: nginx
  name: example
  namespace: foo
spec:
  rules:
  - host: www.example.com
    http:
      paths:
      - backend:
          serviceName: exampleService
          servicePort: 80
        path: /
  # This section is only required if TLS is to be enabled for the Ingress
  tls:
  - hosts:
    - www.example.com
    secretName: example-tls
```

If TLS is enabled for the Ingress, a Secret containing the certificate and key must also be provided:

```
apiVersion: v1
kind: Secret
metadata:
  name: example-tls
  namespace: foo
data:
  tls.crt: <base64 encoded cert>
  tls.key: <base64 encoded key>
type: kubernetes.io/tls
```

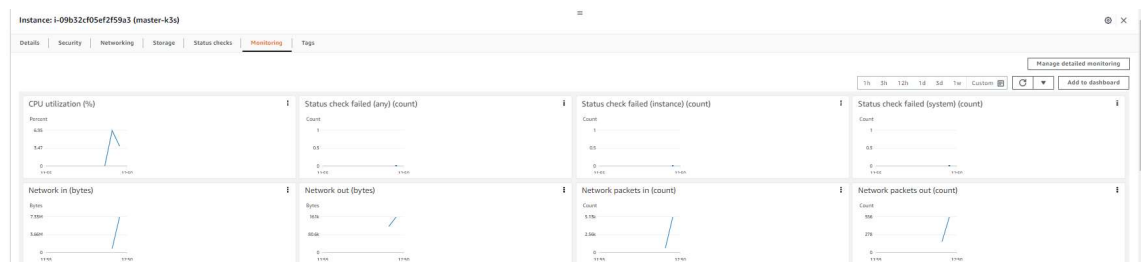
```
ubuntu@master:~$ kubectl get pods -n kube-system -l app=nginx-ingress -o wide
NAME                                READY   STATUS    RESTARTS   AGE   IP              NODE               NOMINATED NODE   READINESS GATES
nginx-ingress-controller-65786b7746-brz9p    0/1     Running   1 (41s ago)    95s   10.42.2.5       ip-172-31-12-168   <none>            <none>
```

After installing nginx check the CPU

```
top - 12:54:09 up 14 min, 1 user, load average: 0.01, 0.10, 0.09
Tasks: 121 total, 1 running, 120 sleeping, 0 stopped, 0 zombie
Rcp(s): 0.0 us, 0.3 sy, 0.0 ni, 98.3 id, 0.2 wa, 0.0 hi, 0.0 si, 0.2 st
Mem Mem : 3864.0 total, 1105.7 free, 247.2 used, 2011.1 buff/cache
Mem Swap: 0.0 total, 0.0 free, 0.0 used, 2876.5 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
568	root	20	0	1265388	519640	101840	S	2.0	13.1	0:35.12	k3s-server
789	root	20	0	769948	61048	38716	S	0.3	1.5	0:06.24	containerd
3068	root	20	0	728748	11028	7084	S	0.3	0.3	0:00.58	containerd-shim
3996	root	20	0	728748	10224	7416	S	0.3	0.3	0:00.26	containerd-shim
4186	ubuntu	20	0	751956	47172	33676	S	0.3	1.2	0:03.00	metrics-server
4795	root	20	0	728748	11028	7712	S	0.3	0.3	0:00.11	containerd-shim
1	root	20	0	104628	12792	8176	S	0.0	0.3	0:05.16	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kthreadd
3	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	rcu_gp
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	rcu_par_gp
5	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	netns
7	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kuorker/0:04-events_highpri
8	root	20	0	0	0	0	I	0.0	0.0	0:00.15	kuorker/u4:0-events_unbound
9	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	mp_percpu_irq
10	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_irqs_rude
11	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_tasks_trace
12	root	20	0	0	0	0	S	0.0	0.0	0:00.11	ksoftirqd/0
13	root	0	0	0	0	0	I	0.0	0.0	0:00.24	rcu_sched
14	root	rt	0	0	0	0	S	0.0	0.0	0:00.00	migration/0
15	root	-51	0	0	0	0	S	0.0	0.0	0:00.00	idle_inject/0
17	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpup/0
18	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpup/1
19	root	-51	0	0	0	0	S	0.0	0.0	0:00.00	idle_inject/1
20	root	rt	0	0	0	0	S	0.0	0.0	0:00.00	migration/1
21	root	20	0	0	0	0	S	0.0	0.0	0:00.11	ksoftirqd/1

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
568	root	20	0	1265772	515332	101840	S	2.7	13.0	0:35.65	k3s-server
1	root	20	0	104628	12800	8176	S	0.7	0.3	0:05.18	systemd
459	message+	20	0	7700	4752	3880	S	0.7	0.1	0:00.61	dbus-daemon
789	root	20	0	769948	61048	38716	S	0.7	1.5	0:06.29	containerd
2068	root	20	0	728748	11292	7084	S	0.7	0.3	0:00.60	containerd-shim
3317	ubuntu	20	0	15172	9768	8164	S	0.7	0.2	0:00.26	systemd
4795	root	20	0	728748	10912	7732	S	0.7	0.3	0:00.15	containerd-shim
4186	ubuntu	20	0	751956	47172	33676	S	0.3	1.2	0:03.12	metrics-server
5322	ubuntu	20	0	11264	2088	3144	R	0.3	0.1	0:00.03	top
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kthreadd
3	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	rcu_gp
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	rcu_par_gp
5	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	netns
7	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kuorker/0:04-events_highpri
8	root	20	0	0	0	0	I	0.0	0.0	0:00.15	kuorker/u4:0-events_unbound



If you want you could install the prometheus with helm to the node

install Prometheus-operator

add repos

helm repo add prometheus-community <https://prometheus-community.github.io/helm-charts>

repo add stable <https://kubernetes-charts.storage.googleapis.com/helm> repo update

install chart

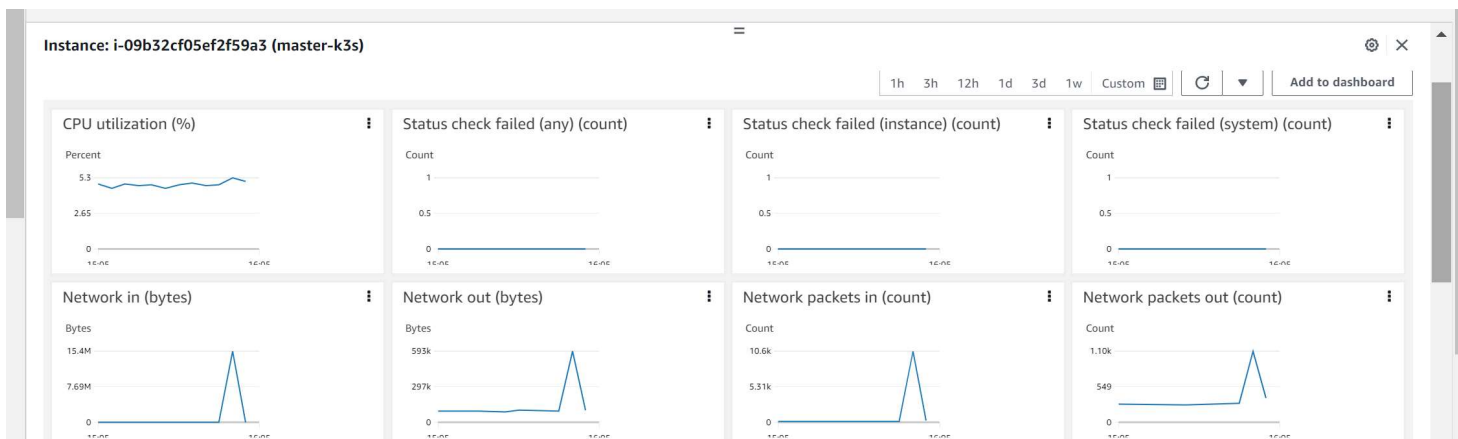
helm install prometheus prometheus-community/kube-prometheus-stack

install chart with fixed version

helm install prometheus prometheus-community/kube-prometheus-stack --version "9.4.1"

```
STATUS: deployed
REVISION: 1
NOTES:
kubernetes-prometheus-stack has been installed. Check its status by running:
kubectl --namespace default get pods -l "release=prometheus"

Visit https://github.com/prometheus-operator/kube-prometheus for instructions on how to create & configure Alertmanager and Prometheus instances using the Operator.
ubuntu@master:~$ kubectl get nodes -o wide
NAME          STATUS    ROLES    AGE   VERSION   INTERNAL-IP   EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION   CONTAINER-RUNTIME
master        NotReady  control-plane,master   20h   v1.25.4+k3s1   172.31.12.168   <none>         Ubuntu 20.04.5 LTS   5.15.0-1019-aws   containerd://1.6.8-k3s1
worker        NotReady  <none>    20h   v1.25.4+k3s1   172.31.1.170   <none>         Ubuntu 20.04.5 LTS   5.15.0-1019-aws   containerd://1.6.8-k3s1
ip-172-31-12-168    Ready    control-plane,master   179m   v1.25.4+k3s1   172.31.12.168   <none>         Ubuntu 20.04.5 LTS   5.15.0-1019-aws   containerd://1.6.8-k3s1
ip-172-31-1-170     Ready    <none>    179m   v1.25.4+k3s1   172.31.1.170   <none>         Ubuntu 20.04.5 LTS   5.15.0-1019-aws   containerd://1.6.8-k3s1
ubuntu@master:~$
```

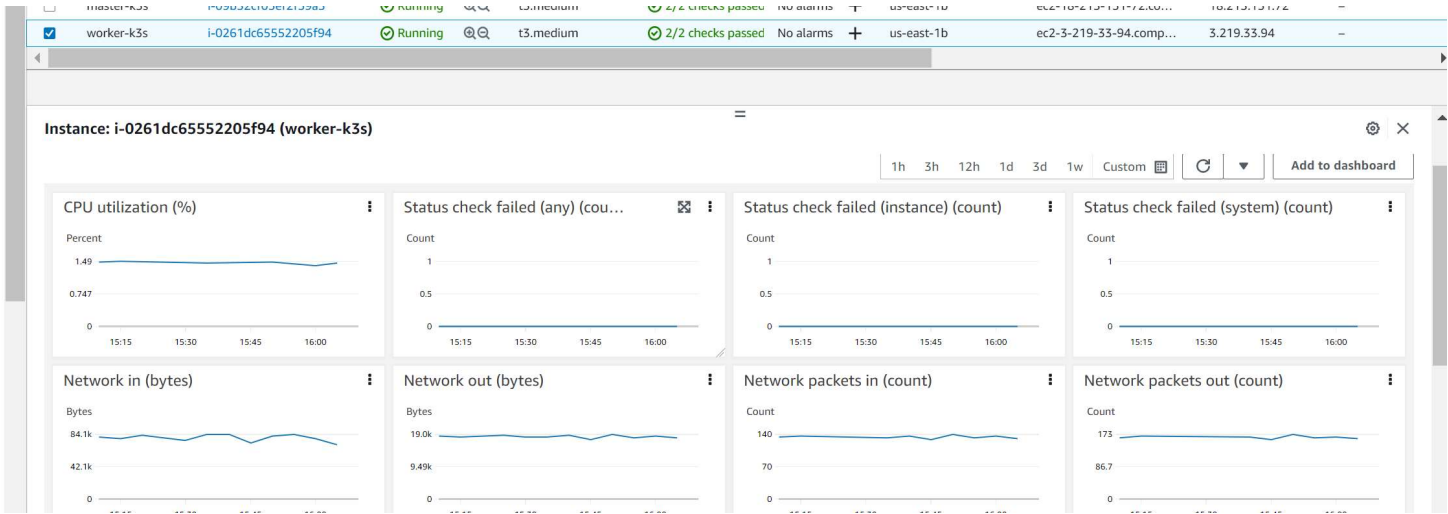




```
top - 16:08:36 up 3:29, 1 user, load average: 0.36, 0.17, 0.12
Tasks: 133 total, 1 running, 132 sleeping, 0 stopped, 0 zombie
%Cpu(s): 5.9 us, 1.0 sy, 0.0 ni, 92.9 id, 0.0 wa, 0.0 hi, 0.0 si, 0.2 st
MiB Mem : 3864.0 total, 105.1 free, 1619.8 used, 2139.1 buff/cache
MiB Swap: 0.0 total, 0.0 free, 0.0 used. 1968.1 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
560	root	20	0	1748388	688436	96096	S	7.3	17.4	8:28.08	k3s-server
7319	ubuntu	20	0	1429400	560452	65976	S	4.3	14.2	2:46.29	prometheus
6499	nobody	20	0	717892	19284	11564	S	1.3	0.5	0:11.36	node_exporter
6323	root	20	0	720748	11476	8116	S	1.0	0.3	0:22.66	containerd-shim
789	root	20	0	771848	66984	39728	S	0.3	1.7	1:04.91	containerd
2608	root	20	0	720748	11524	7872	S	0.3	0.3	0:16.51	containerd-shim
3982	root	20	0	721004	11496	7916	S	0.3	0.3	0:08.50	containerd-shim
4104	root	20	0	733460	26004	20060	S	0.3	0.7	0:02.70	local-path-prov
6811	root	20	0	720748	10772	7364	S	0.3	0.3	0:15.71	containerd-shim
1	root	20	0	104028	12808	8176	S	0.0	0.3	0:07.45	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kthreadd
3	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	rcu_gp

Worker node



```
top - 16:13:08 up 3:34, 1 user, load average: 0.06, 0.04, 0.01
Tasks: 129 total, 1 running, 128 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.8 us, 0.3 sy, 0.0 ni, 98.5 id, 0.0 wa, 0.0 hi, 0.0 si, 0.3 st
MiB Mem : 3864.0 total, 744.3 free, 460.7 used, 2659.0 buff/cache
MiB Swap: 0.0 total, 0.0 free, 0.0 used. 3118.8 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
555	root	20	0	845916	132668	88804	S	1.3	3.4	3:06.95	k3s-agent
35780	root	20	0	720492	10872	8212	S	1.0	0.3	0:03.64	containerd-shim
6360	root	20	0	770992	64616	38084	S	0.3	1.6	1:01.57	containerd
42563	ubuntu	20	0	11260	4124	3464	R	0.3	0.1	0:00.01	top
1	root	20	0	104072	12964	8340	S	0.0	0.3	0:07.94	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kthreadd
3	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	rcu_gp
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	rcu_par_gp
5	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	netns
7	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/0:0H-events_highpri
9	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	mm_percpu_wq
10	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_tasks_rude_
11	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_tasks_trace