**Installation guide for a Kubernetes**

**Cluster on deep learning machines**

This document has been elaborated on the basis of the requirements and information you have communicated to us, with reference to your context, and taking into account the current legal and economic environment.

The findings presented, were elaborated on the basis of our methods, processes, techniques and know-how. Accordingly, they, together with the support medium, are our property. The decision as to whether or not to implement these findings, as well as the methods of such implementation, is your responsibility.

This document, the form and content of which are reserved for your internal use only, is confidential. It may only be disclosed to third parties with our agreement; however, EY expressly authorizes the communication, to any person, of the detailed advice pertaining to the tax areas of this document; in any case, we do not assume any liability with respect to third parties.

This document is issued in accordance with the contract entered into between us.

**Sommaire**

[**1.** **Installation guide** 4](#_Toc517269205)

1. [Quick introduction of kubernetes 4](#_Toc517269206)
2. [Requirements 5](#_Toc517269207)

[2. Installation of docker on each node 5](#_Toc517269208)

[3. Installation of different Kubernetes package on each node 6](#_Toc517269209)

[4. Configuration of the docker driver on the master node. 6](#_Toc517269210)

[5. Selection of the containers network interface for kubernetes. 7](#_Toc517269211)

[6. Installation of kubeadm on the master node. 9](#_Toc517269212)

[7. Installation of the CNI 11](#_Toc517269213)

[8. Joint of the slave node to the master node 12](#_Toc517269214)

[9. Final check in the master node 13](#_Toc517269215)

# **Installation guide**

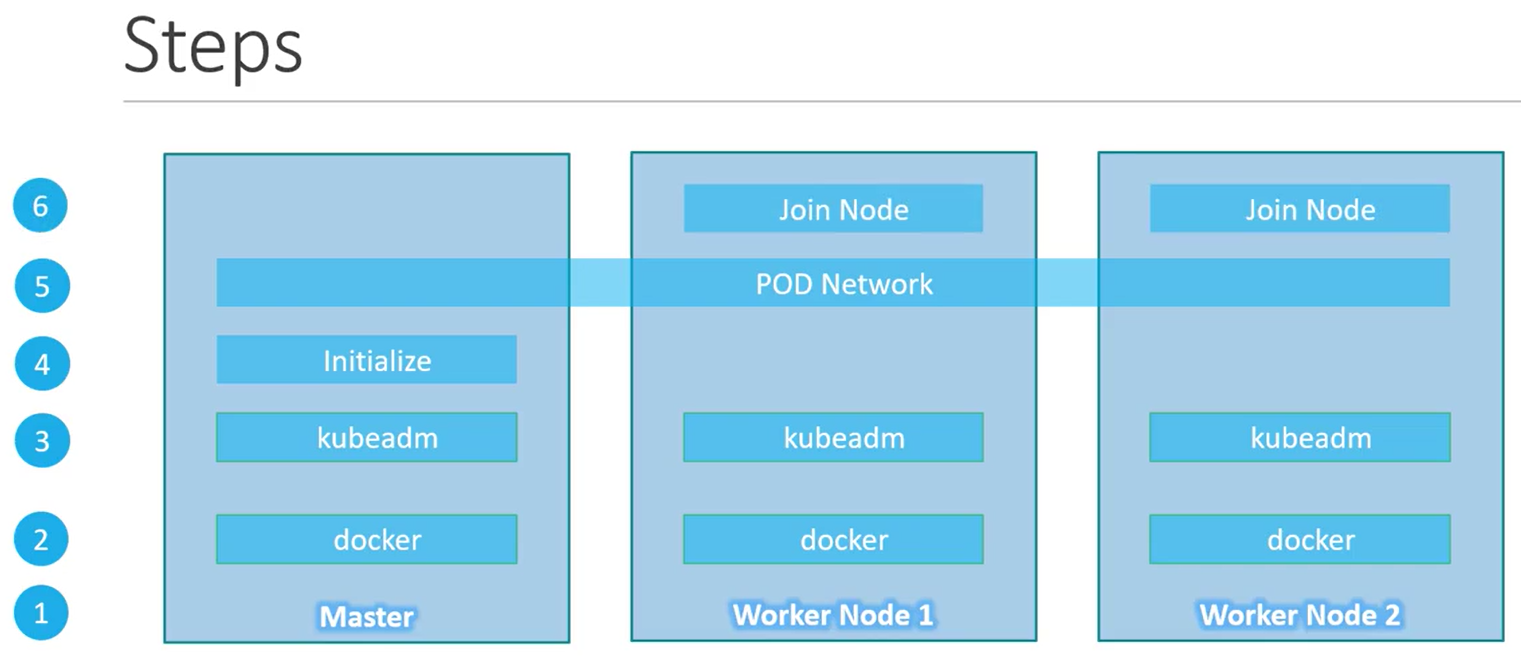
## Quick introduction of kubernetes

Kubernetes is an orchestrator of containers.

It is a portable, extensible open-source platform use for managing containerized workloads and services. Here are some main features:

* **Automatic binpacking:** Automatically places containers based on their resource requirements and other constraints, while not sacrificing availability. Mix critical and best-effort workloads in order to drive up utilization and save even more resources.
* **Self-healing:** Restarts containers that fail, replaces and reschedules containers when nodes die, kills containers that don't respond to your user-defined health check, and doesn't advertise them to clients until they are ready to serve.
* **Horizontal scaling:** Scale your application up and down with a simple command, with a UI, or automatically based on CPU usage.
* **Service discovery and load balancing:** No need to modify your application to use an unfamiliar service discovery mechanism. Kubernetes gives containers their own IP addresses and a single DNS name for a set of containers, and can load-balance across them.
* **Automated rollouts and rollbacks:** Kubernetes progressively rolls out changes to your application or its configuration, while monitoring application health to ensure it doesn't kill all your instances at the same time. If something goes wrong, Kubernetes will rollback the change for you. Take advantage of a growing ecosystem of deployment solutions.
* **Secret and configuration management:** Deploy and update secrets and application configuration without rebuilding your image and without exposing secrets in your stack configuration.
* **Storage orchestration:** Automatically mount the storage system of your choice, whether from local storage, a public cloud provider.
* **Batch execution**: In addition to services, Kubernetes can manage your batch and CI workloads, replacing containers that fail, if desired.

here is the diagram summarizing the installation performed



There are five steps:

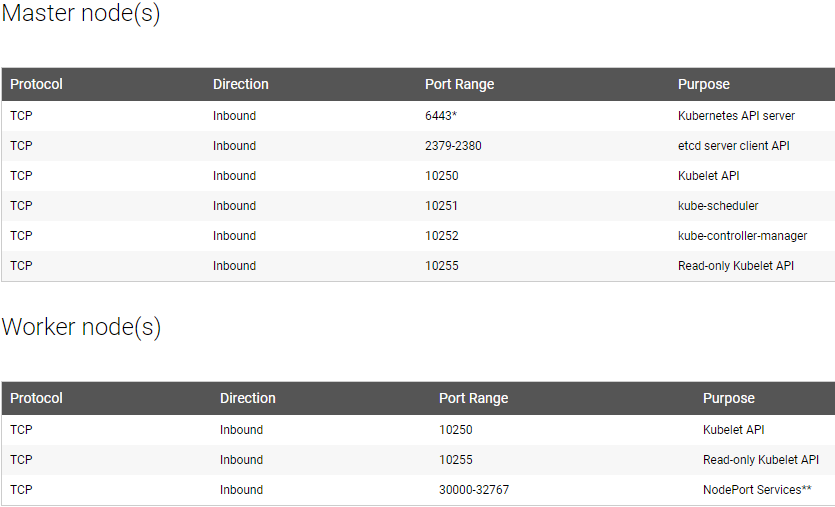
1. Installation of the same docker version on all the nodes
2. Install of the kubernetes packages (kubeadmin, kubelet, kubectl) on all the nodes
3. Initialization of the cluster on the master node
4. Selection and installation of the container network interface which permits to communicate between the containers and kubernetes
5. Join from the slaves to the cluster

## Requirements

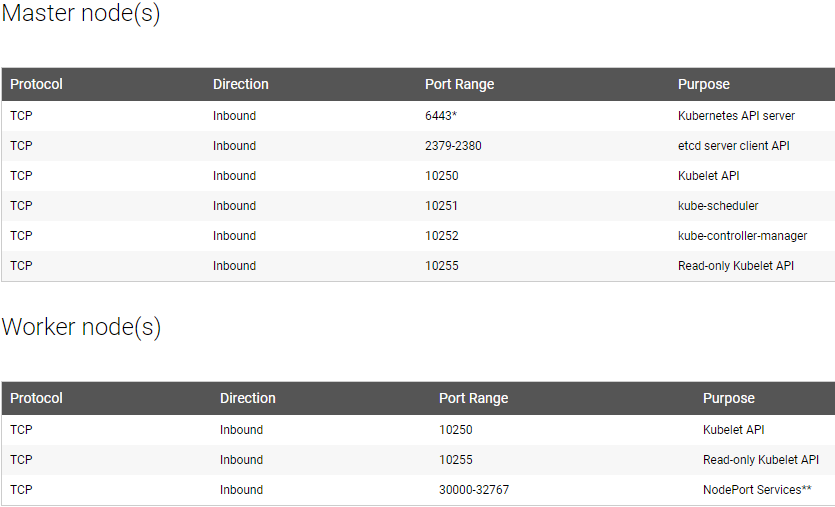
Before beginning the installation, check each of your machine have:

* Ubuntu 16.04+
* Full network connectivity between all machines in the cluster ( public and private network)
* Unique hostname, MAC address and product\_uuid for every node.
* Check that some ports are open on your machines.
* Swap disabled.

Ports required for the master node\*



Ports required for the worker node\*



\*\*\* Tables coming from the “Check required ports” part from: [https://kubernetes.io/docs/tasks/tools/install-kubeadm/#check-required-ports](https://kubernetes.io/docs/tasks/tools/install-kubeadm/%23check-required-ports%20)

## **Installation of docker on each node**

Each installation must be executed by the root user: **sudo -i**

We start by installing Docker on each machine. ( in our case they are the Titan V and Jedi Order machines) . Note that Kubernetes has installed version 17.03 of docker on its demo and that we have installed the latest versions on our machines: version (18.03).

Lien d’installation d’Ubuntu: <https://docs.docker.com/install/linux/docker-ce/ubuntu/>

The command line to install the 17.03 docker version:

**apt-get update**

**apt-get install -y apt-transport-https ca-certificates curl software-properties-common**

**curl -fsSL https://download.docker.com/linux/ubuntu/gpg | apt-key add -**

**add-apt-repository "deb https://download.docker.com/linux/$(. /etc/os-release; echo "$ID") $(lsb\_release -cs) stable"**

**apt-get update && apt-get install -y docker-ce=$(apt-cache madison docker-ce | grep 17.03 | head -1 | awk '{print $3}')**

## **Installation of different Kubernetes package on each node**

The different package are:

* Kubeadm: the command to bootstrap the cluster
* Kubelet: this component allows you to communicate on all the machines of your cluster and allows you to launch/deploy containers.
* Kubectl: This command line used to talk to your cluster

**apt-get update && apt-get install -y apt-transport-https curl**

**curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-key add -**

**cat <<EOF >/etc/apt/sources.list.d/kubernetes.list**

**deb http://apt.kubernetes.io/ kubernetes-xenial main**

**EOF**

**apt-get update**

**apt-get install -y kubelet kubeadm kubectl**

Command line coming from: <https://kubernetes.io/docs/tasks/tools/install-kubeadm/>

## **Configuration of the docker driver on the master node.**

Check if your Docker cgroup driver matches with the kubelet config:

**docker info | grep -i cgroup**

**cat /etc/systemd/system/kubelet.service.d/10-kubeadm.conf**

If there is no match:

**sed -i "s/cgroup-driver=systemd/cgroup-driver=cgroupfs/g" /etc/systemd/system/kubelet.service.d/10-kubeadm.conf**

Then you can restart kubelet:

**systemctl daemon-reload**

**systemctl restart kubelet**

At this point, step 2 and 3 are finished.

## **Selection of the containers network interface for kubernetes.**

At this stage, we must create a cluster on the master node then we must go to the slave nodes to join the cluster. The joint is secured by a SHA 256 key created during cluster initialization.

In order to create a cluster, a Container Network Interface need to be plug-in. A CNI is a simple interface between container runtime and network implementation.

Coordinating ports across multiple developers is very difficult to do at scale and exposes users to cluster-level issues outside of their control. Dynamic port allocation brings a lot of complications to the system - every application has to take ports as flags, the API servers have to know how to insert dynamic port numbers into configuration blocks, services have to know how to find each other, etc. Rather than deal with this, Kubernetes takes a different approach.

Kubernetes imposes the following fundamental requirements on any networking implementation (barring any intentional network segmentation policies):

* all containers can communicate with all other containers without NAT (Network Address Translation)
* all nodes can communicate with all containers (and vice-versa) without NAT
* the IP that a container sees itself as is the same IP that others see it as

CNI has three main components:

1. CNI Specification: defines an API between runtimes and network plugins for container network setup.
2. Plugins: provide network setup for a variety of use-cases and serve as reference examples of plugins conforming to the CNI specification
3. Library: provides a Go implementation of the CNI specification that runtimes can use to more easily consume CNI

CNI specification and libraries exist to write plugins to configure network interfaces in Linux containers. The plugins support the addition and removal of container network interfaces to and from networks.

 These CNI can use different interface (explanations of these different interface in the annexe):

* Bridge
* Ipvlan
* Loopback
* Macvlan
* Ptp
* Vlan

There are several Containers Network Interface for Kubernetes.

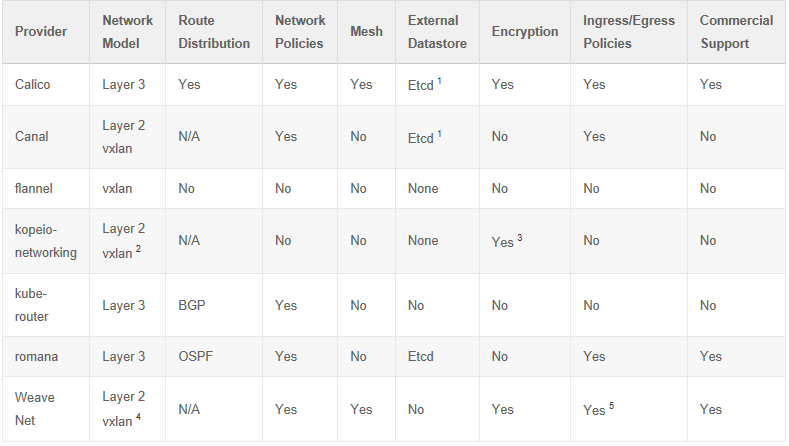


Table dated from November 2017 coming from: <https://chrislovecnm.com/kubernetes/cni/choosing-a-cni-provider/>

**Network Model**

The Network Model with providers is either encapsulated networking such as VXLAN or unencapsulated layer 2 networking. Encapsulating network traffic requires compute to process, so theoretically is slower.

\*VXLAN: Virtual Extensible LAN: is a network virtualization technology that attempts the scalability problems with large cloud computing deployments.

**Route Distribution**

For layer 3 CNI providers, route distribution is necessary. Route distribution is typically via BGP. Route distribution is a nice to have a feature with CNI, if you plan to build clusters split across network segments. It is an exterior gateway protocol designed to exchange routing and reachability information on the Internet. BGP can assist with pod to pod networking between clusters.

\*BGP: Border Gateway Protocol is a standardized exterior gateway protocol designed to exchange routing and reachability information among autonomous systems (AS) on the Internet

**Network Policies:** offer functionality to enforce rules about which pods can communicate with each other using network policies. The feature has become K8S 1.7 and it is ready to use with supported networks plugins. The release 1.8 has added better capabilities to this feature. ***Important***

**Mesh Networking:** This feature allows for “pod to pod” networking between Kubernetes clusters. This technology is not Kubernetes federation, but it pure networking between pods.

**Encryption*:*** Encrypting the network control plane, so all TCP and UDP traffic is encrypted.

**Ingress / Egress Policies :** it is a reverse proxy for kubernetes.

So we can keep Canal, Calico and Weave Net for encrypting part and the network polices.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Stars | Contributors | Forks |
| Calico | 39 | 139 | 98 |
| Canal | 496 | 19 | 64 |
| Flannel | 2962 | 102 | 718 |
| Kube-router | 624 | 24 | 99 |
| Romana | 5 | 5 | 1 |
| Weave | 5037 | 54 | 436 |

Table dated On May 2018 by Loïc de Jouëtte

Data coming from :

Calico: <https://github.com/projectcalico/cni-plugin>

Canal: <https://github.com/projectcalico/canal>

Flannel: <https://github.com/coreos/flannel>

Kube-router: <https://github.com/cloudnativelabs/kube-router>

Romana: <https://github.com/romana/kube>

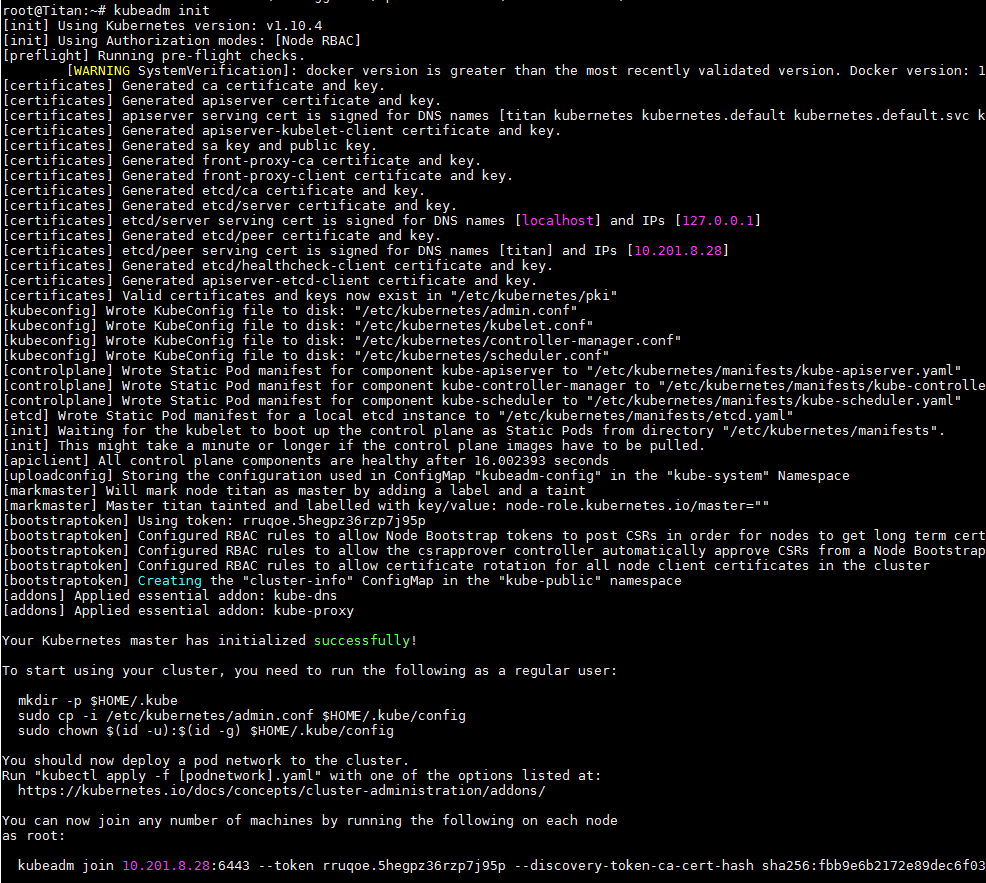
Weave: <https://github.com/weaveworks/weave>

## **Installation of kubeadm on the master node.**

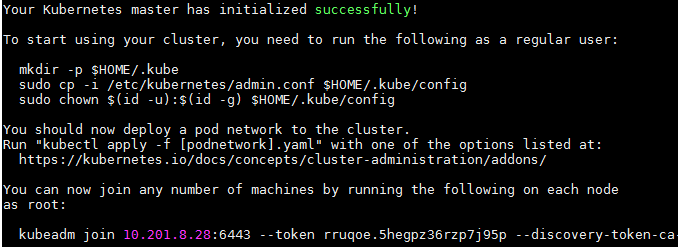
Finally Weave Net for the kubernetes CNI is a good solution due to its utilization and the contributions and supports. The installation of *kubeadm int* is different indeed the CNI chosen.

Link for the installation of others CNI and for the installation of the kubeadm: <https://kubernetes.io/docs/setup/independent/create-cluster-kubeadm/>

**kubeadm init**



So the kubernetes cluster installation on the master is a success. We can see that Kubernetes gives a message to start the cluster.



Kubernetes indicates also the follow process to complete the installation for the non-root user.

**mkdir -p $HOME/.kube**

**sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config**

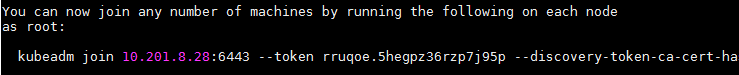
**sudo chown $(id -u):$(id -g) $HOME/.kube/config**

If you are the root user, you can run:

**export KUBECONFIG=/etc/kubernetes/admin.conf**

**More important, it gives the line to copy then past in the slave node in order to realize the joint.** **So don’t forget to copy this line somewhere**. The link lasts 24h. It compose of master-ip:master-port with a generated token and a key sha256 to ensure the connection.

So copy the line in the red frame:



Example of a typical link:

**kubeadm join <master-ip>:<master-port> --token <token> --discovery-token-ca-cert-hash sha256:<hash>**

## **Installation of the CNI**

A pod network add-on (CNI) must be installed so that your pods can communicate with each other. For Weave Net, a parameter **bridge-nf-call-iptables** set to 1.

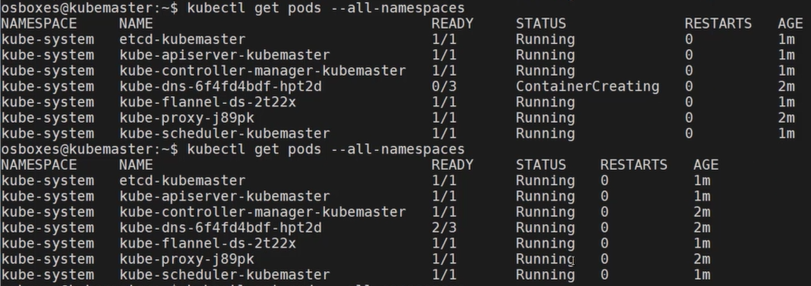
**sysctl net.bridge.bridge-nf-call-iptables=1**

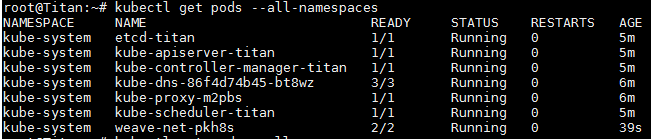
To install a pod network add-on with the following command:

**kubectl apply -f "https://cloud.weave.works/k8s/net?k8s-version=$(kubectl version | base64 | tr -d '\n')"**

Check with the command line that all your components are running before the next step. Especially the kube-dns component. It may take few minutes.

**kubectl get pods --all-namespaces**





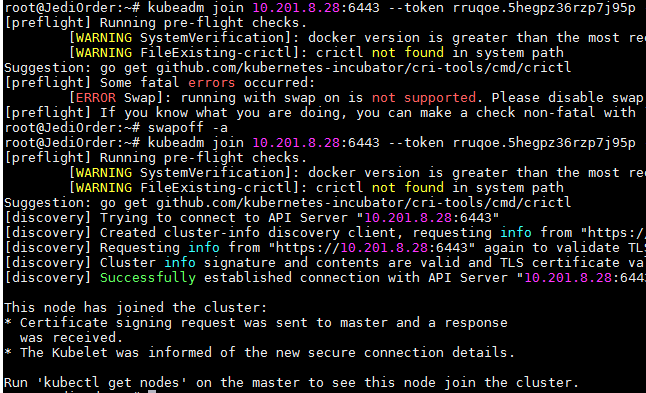
## **Joint of the slave node to the master node**

Check if the swap disabled.

**sudo -i**

So you have to copy the line at the end of the *kubeadm init* and launch it in the slave nodes.

**kubeadm join 10.201.8.28:6443 --token rruqoe.5hegpz36rzp7j95p --discovery-token-ca-cert-hash sha256:fbb9e6b2172e89dec6f0392526ce02d9b4c3995734d8634e59a326929f786678**



A final message appears:

**Node join complete:**

**\* Certificate signing request sent to master and response**

**received.**

**\* Kubelet informed of new secure connection details.**

**Run 'kubectl get nodes' on the master to see this machine join.**

## **Final check in the master node**

Check on the master node if the pods is on the cluster.

**kubectl get nodes**

