**Step-by-Step: What Is a Kubernetes Cluster (Single Master + Multi Node)?**

**🧊 First — What is Kubernetes?**

Think of **Kubernetes** like a **smart robot manager** that takes care of your **apps**, like websites or services. It **runs, heals, scales, and updates** them across many computers.

**What is a Kubernetes Cluster?**

A **cluster** is just a group of computers (real or virtual) working together in Kubernetes.

There are two types of computers in a cluster:

| **Type** | **What It Does** |
| --- | --- |
| **Master Node** (Control Plane) | The **brain**. Makes decisions like what app runs where. |
| **Worker Nodes** | The **hands**. Actually **run the apps** and do the work. |

So a **Single Master + Multi Node cluster** =  
🔹 1 boss (master)  
🔹 Many workers (nodes)

**🌩️ In AWS — What Does This Look Like?**

We use **EC2 instances** (virtual computers in AWS) to create:

* ✅ 1 EC2 for the **master**
* ✅ 2 or more EC2s for the **workers**

We connect them using **Kubernetes tools** like kubeadm, or use **Kops** (Kubernetes Operations), which sets this up automatically.

**🛠️ Tools You'll Use**

| **Tool** | **What it Does** |
| --- | --- |
| **AWS EC2** | Virtual machines (servers) |
| **Kops** | A tool that builds the cluster for you on AWS (easy setup) |
| **kubectl** | The command-line tool to **talk to Kubernetes** and give orders |
| **Route53** | AWS DNS service — Kops uses it to help set up the master node |
| **S3** | AWS storage bucket — used by Kops to store the cluster state |

**🔧 What You Will Build**

A **cluster** that looks like this:

pgsql

CopyEdit

+---------------------+

| Master Node | (k8s "control plane")

| Runs: API Server, Scheduler, etc.

+----------+----------+

|

-----------------------------------------

| | |

+----------------+ +----------------+ +----------------+

| Worker Node 1 | | Worker Node 2 | | Worker Node 3 |

| Runs: Your App | | Runs: Your App | | Runs: Your App |

+----------------+ +----------------+ +----------------+

**🪄 How Kops Sets This Up (Overview)**

Kops does everything for you. The steps look like this:

**✅ Step 1: Install Kops and kubectl**

We'll do this on your local computer or a VM.

bash

CopyEdit

# Install kops

curl -LO https://github.com/kubernetes/kops/releases/download/v1.28.0/kops-linux-amd64

chmod +x kops-linux-amd64

sudo mv kops-linux-amd64 /usr/local/bin/kops

# Install kubectl (to control Kubernetes)

curl -LO "https://dl.k8s.io/release/v1.28.0/bin/linux/amd64/kubectl"

chmod +x kubectl

sudo mv kubectl /usr/local/bin/kubectl

💡 kops = creates the cluster  
💡 kubectl = talks to the cluster

**✅ Step 2: Create an S3 Bucket (holds cluster config)**

bash

CopyEdit

aws s3api create-bucket \

--bucket my-k8s-state-store \

--region us-east-1

Then export the name for Kops to use:

bash

CopyEdit

export KOPS\_STATE\_STORE=s3://my-k8s-state-store

**✅ Step 3: Create a Domain (using Route 53)**

Kops needs a DNS name like cluster.example.com.

You can:

1. Buy a domain (like from AWS Route53 or GoDaddy)
2. Create a hosted zone in Route53
3. Then use that domain like:

bash

CopyEdit

export NAME=cluster.example.com

**✅ Step 4: Create the Cluster**

bash

CopyEdit

kops create cluster \

--name=$NAME \

--zones=us-east-1a \

--master-count=1 \

--node-count=2 \

--node-size=t3.medium \

--master-size=t3.medium \

--dns-zone=example.com

This creates:

* 1 Master EC2
* 2 Node EC2s
* Sets up all networking, Kubernetes, etc

**✅ Step 5: Actually Build It**

bash

CopyEdit

kops update cluster --name $NAME --yes

**✅ Step 6: Check the Cluster**

bash

CopyEdit

kubectl get nodes

You should see 1 master + 2 workers!

**📦 Summary of What You Learn**

| **Concept** | **Easy Explanation** |
| --- | --- |
| **Cluster** | Group of machines working together |
| **Master Node** | Boss that gives orders |
| **Worker Node** | Runs your app |
| **Kops** | Tool to create Kubernetes on AWS |
| **kubectl** | Remote control for Kubernetes |
| **EC2** | AWS virtual computer |
| **S3** | Storage for cluster configs |
| **Route53** | DNS system for your cluster |
| **Pods** | Small units that hold your app |

Would you like me to:

✅ Help you do this **live step-by-step on your Ubuntu machine**?  
✅ Build this in a **simulator or diagram** first?  
✅ Write you a **ready-made script** to launch the cluster?

Let me know how you'd like to learn — slowly with practice or fast with scripts!

**🧠 Pro Tip**

To make this easier:

# Set a unique bucket name

export BUCKET\_NAME="my-k8s-state-store-said-$(date +%s)"

# Create the bucket

aws s3api create-bucket --bucket $BUCKET\_NAME --region us-east-1

# Export it for Kops to use

export KOPS\_STATE\_STORE=s3://$BUCKET\_NAME

**✅ Step-by-Step: Log in to AWS from Ubuntu VM**

**🧰 You need:**

* An **AWS account**
* Your **Access Key ID** and **Secret Access Key**

**🔑 Step 1: Get Your AWS Access Keys**

1. Log in to [AWS Console](https://console.aws.amazon.com/)
2. Go to **"My Security Credentials"**  
   (Click your name at the top → Security Credentials)
3. Under **Access Keys**, create a new one.
4. You will get:
   * Access Key ID: AKIA...
   * Secret Access Key: wJalrXUtn...

📌 **Keep these safe.** Don’t share them.

**💻 Step 2: Configure AWS CLI on Ubuntu**

Run this in your Ubuntu terminal:

bash

CopyEdit

aws configure

Then enter the details:

pgsql

CopyEdit

AWS Access Key ID [None]: YOUR\_KEY\_ID

AWS Secret Access Key [None]: YOUR\_SECRET\_KEY

Default region name [None]: us-east-1

Default output format [None]: json

Now your VM is **logged into AWS**, and commands like aws s3api will work.

**✅ Step 3: Test It**

Try this:

bash

CopyEdit

aws sts get-caller-identity

If it returns your AWS account ID and user info, you’re **successfully logged in** 🎉

Would you like help getting your Access Key or running aws configure now?

**You said:**

aid@said-VirtualBox:~$ aws s3api create-bucket \ --bucket Sabir-k8s-state-store \ --region us-east-1 An error occurred (InvalidBucketName) when calling the CreateBucket operation: The specified bucket is not valid.

**ChatGPT said:**

Thanks for sharing the exact error — let’s fix it step by step! 😊

**❗ Problem: InvalidBucketName**

**✅ Cause:**

Your bucket name Sabir-k8s-state-store has **uppercase letters** (S), and **S3 bucket names must follow strict rules**.

**✅ ✅ S3 Bucket Naming Rules (Simplified)**

1. **Only lowercase letters** (no uppercase!)
2. **Only use**:  
   a-z, 0-9, - (hyphen)
3. Must start and end with a **letter or number**
4. Length: 3–63 characters
5. **No special characters**, no spaces, no underscores

**🛠️ Solution**

Just change the name to all **lowercase**, like this:

bash

CopyEdit

aws s3api create-bucket \

--bucket sabir-k8s-state-store \

--region us-east-1

OR if it still says "already exists", add a unique number or date:

bash

CopyEdit

aws s3api create-bucket \

--bucket sabir-k8s-state-store-240609 \

--region us-east-1

Then:

bash

CopyEdit

export KOPS\_STATE\_STORE=s3://sabir-k8s-state-store-240609

✅ Now you're ready to go forward and create the Kubernetes cluster with Kops!

Let me know once this works and I’ll guide you to the next step.

**You said:**

ok now I am ready to go forward and create the Kubernetes cluster with Kops!

**ChatGPT said:**

Awesome! You're doing great — now let's move forward and **create your Kubernetes cluster with Kops** step-by-step on AWS.

**🧱 What You’re Building**

You’re about to create:

* ✅ 1 **Master node** (the brain of the cluster)
* ✅ 2 **Worker nodes** (machines that run your apps)
* All managed by **Kops**, running in AWS EC2
* Cluster DNS like: mycluster.k8s.local

**✅ STEP 1: Choose a DNS Name**

If you don’t have a real domain yet, we can use a **fake internal domain** like mycluster.k8s.local.

Let’s define the cluster name:

bash

CopyEdit

export CLUSTER\_NAME=mycluster.k8s.local

**✅ STEP 2: Create the Cluster Configuration**

Now we tell Kops how many machines we want and what size.

bash

CopyEdit

kops create cluster \

--name=$CLUSTER\_NAME \

--zones=us-east-1a \

--node-count=2 \

--node-size=t3.medium \

--master-size=t3.medium \

--dns=none

**🔍 Explanation:**

| **Option** | **Meaning** |
| --- | --- |
| --name | The cluster DNS name (we're using .k8s.local) |
| --zones | The AWS availability zone for EC2 (like us-east-1a) |
| --node-count | Number of worker nodes |
| --node-size | Type of EC2 for workers (t3.medium = good balance) |
| --master-size | Type of EC2 for master node |
| --dns=none | We’re not using Route53 now, just internal DNS for test cluster |

**✅ STEP 3: Build the Cluster on AWS**

This command actually **starts building EC2 instances** and all the network stuff:

bash

CopyEdit

kops update cluster --name=$CLUSTER\_NAME --yes

**⚠️ Wait 5–10 minutes. It takes time to:**

* Create VPC, networking
* Create EC2 instances
* Install Kubernetes

You can monitor with:

bash

CopyEdit

kops validate cluster --name=$CLUSTER\_NAME

When it's **"ready"**, you'll see that nodes are connected and healthy.

**✅ STEP 4: Talk to Your Cluster with kubectl**

Run:

bash

CopyEdit

kubectl get nodes

You should see:

* 1 master node
* 2 worker nodes (in "Ready" state)

🎉 Congrats — You now have a working **Kubernetes cluster on AWS**!

**✅ What's Next?**

Would you like to:

1. 🚀 Deploy a **simple app** on this cluster (like Nginx or a Python app)?
2. 🧠 Understand the components of Kubernetes (Pods, Services, Deployments)?
3. 💾 Add Persistent Volumes (storage)?
4. 🔐 Secure access with authentication (RBAC)?

Tell me your next goal — I’ll walk you through like a pro teaching a beginner.

**You said:**

I need to understand does KOPS is installed in my vm ubuntu machine or in EC2

**ChatGPT said:**

That’s an **excellent question**, and I’ll explain it very clearly. 🤓

**🔍 Where Is Kops Installed?**

✅ **Kops is installed and runs on your local machine** — in your case, **your Ubuntu VM**.

**🧠 Think of it like this:**

| **Tool** | **Where it runs** | **What it does** |
| --- | --- | --- |
| **Kops** | ✅ Your Ubuntu VM | 💡 *Generates and manages* Kubernetes cluster on AWS |
| **EC2 instances** | AWS Cloud | 🖥️ Run the actual Kubernetes Master and Worker nodes |
| **kubectl** | ✅ Your Ubuntu VM | 🕹️ *Talks to* the cluster (after it's created by Kops) |

**📦 So the flow is like this:**

pgsql

CopyEdit

+-----------------+ +--------------------------+

| Your Ubuntu VM | ==> | AWS Cloud Infrastructure |

| (Kops + kubectl)| | (EC2, VPC, S3, etc) |

+-----------------+ +--------------------------+

|

| You run "kops create cluster"

|----------------------------->

| Kops uses your AWS credentials (from `aws configure`)

| to create the EC2 instances and network

**🛠️ Recap**

* You **install Kops on your Ubuntu VM**
* Kops uses the AWS CLI and your ~/.kube/config to manage the cluster **remotely**
* It never runs inside the cluster — it’s an **external admin tool**

Would you like to now check if Kops is already installed correctly on your Ubuntu VM and if it’s working?

Just run:

bash

CopyEdit

kops version

Let me know what you see!

**You said:**

I dont understand what you mean by domain , but I do have a domain with godaddy

**ChatGPT said:**

Perfect! You having a domain with **GoDaddy** is **very helpful** — let me explain everything in **simple words** so you understand clearly what a **domain** is, and **why Kubernetes (Kops) needs it**.

**🧠 What Is a Domain?**

A **domain** is a human-friendly name for computers on the internet.

Examples of domains:

* google.com
* facebook.com
* mybusiness.net
* sabirtech.com (just an example if you bought it)

When you buy a domain from **GoDaddy**, you're buying a **unique name** on the internet that you can use for your:

* website,
* email,
* or in our case: **Kubernetes cluster**.

**🌐 Why Does Kops Need a Domain?**

When Kops builds a cluster, it uses a **domain name** like:

CopyEdit

mycluster.sabirtech.com

To:  
✅ Identify the cluster  
✅ Let you connect to it using kubectl  
✅ Make the **master node reachable** using DNS

So if you use your GoDaddy domain, Kops can make something like:

CopyEdit

api.mycluster.sabirtech.com

which lets kubectl talk to the cluster master node.

**🔄 How Kops Uses the Domain**

It needs to:

1. Create a **"hosted zone"** in DNS (this is like the phonebook where DNS names are stored)
2. Add records like:

css

CopyEdit

api.mycluster.sabirtech.com --> [master node IP]

🟢 For this to work, your **GoDaddy domain must point to AWS Route53** (Amazon’s DNS service), because Kops automatically updates Route53.

**🛠️ What You Can Do (2 Options)**

**✅ Option 1: Use Your GoDaddy Domain with Route53 (Recommended for Real Projects)**

Steps:

1. Go to **GoDaddy → DNS Settings**
2. Change **Nameservers** to AWS Route53
3. In AWS → Route53, create a **"Hosted Zone"** for your domain (e.g. sabirtech.com)
4. Use that in Kops like:

bash

CopyEdit

export CLUSTER\_NAME=mycluster.sabirtech.com

kops create cluster \

--name=$CLUSTER\_NAME \

--zones=us-east-1a \

--node-count=2 \

--node-size=t3.medium \

--master-size=t3.medium \

--dns-zone=sabirtech.com

Kops will then create:

CopyEdit

api.mycluster.sabirtech.com → master node IP

And you’ll be able to run your cluster!

**✅ Option 2: Use Fake Domain for Testing (k8s.local)**

If you’re just **testing or learning**, you can skip real domain and DNS and use:

bash

CopyEdit

export CLUSTER\_NAME=mycluster.k8s.local

kops create cluster \

--name=$CLUSTER\_NAME \

--zones=us-east-1a \

--node-count=2 \

--dns=none

⚠️ But this won't work for **production**, only local VMs.

**🎯 Summary**

| **Term** | **Meaning** |
| --- | --- |
| **Domain** | A name like sabirtech.com you own via GoDaddy |
| **DNS** | System that turns domain name into IP |
| **Route53** | AWS DNS service that Kops uses |
| **Hosted Zone** | Like a control panel for managing records under your domain |
| **api.cluster.yourdomain.com** | Name Kops uses to connect to your master node |

Would you like me to walk you through:

* 🔄 How to connect your **GoDaddy domain to Route53**?
* Or use a **temporary .k8s.local domain** for testing?

Let me know which way you want to go!

**You said:**

said@said-VirtualBox:~$ kops create cluster \ --name=$CLUSTER\_NAME \ --zones=us-east-1a \ --node-count=2 \ --node-size=t3.medium \ --master-size=t3.medium \ --dns=none Flag --master-size has been deprecated, use --control-plane-size instead Error: error reading cluster configuration "myk8scluster.xyz": error reading s3://myk8scluster.xyz/myk8scluster.xyz/config: Could not retrieve location for AWS bucket myk8scluster.xyz

**ChatGPT said:**

You're doing a great job so far — and this error is **very common** when using Kops for the first time. Let's break it down in **dummy-friendly, step-by-step way**.

**❌ Error Explained**

❗ Could not retrieve location for AWS bucket myk8scluster.xyz  
This means: Kops is trying to **load the cluster config from an S3 bucket** called myk8scluster.xyz, but:

* That **bucket doesn't exist**, or
* You **didn’t tell Kops** where to look for it

**✅ Fix: Set the Correct S3 Bucket Using Environment Variable**

**Step 1: You need to tell Kops where to store the cluster info.**

Let’s say earlier you successfully created a bucket like:

perl

CopyEdit

sabir-k8s-state-store-240609

You need to tell Kops to use this bucket as your **state store**:

bash

CopyEdit

export KOPS\_STATE\_STORE=s3://sabir-k8s-state-store-240609

✅ You must run this in **the same terminal** before using any kops command.

**🛠️ Correct Full Flow (Fixed Version)**

**Step 1: Set the environment variables**

bash

CopyEdit

export KOPS\_STATE\_STORE=s3://sabir-k8s-state-store-240609

export CLUSTER\_NAME=myk8scluster.xyz

✅ KOPS\_STATE\_STORE: where your cluster config is saved  
✅ CLUSTER\_NAME: the domain name (real or fake)

**Step 2: Create the cluster configuration**

bash

CopyEdit

kops create cluster \

--name=$CLUSTER\_NAME \

--zones=us-east-1a \

--node-count=2 \

--node-size=t3.medium \

--control-plane-size=t3.medium \

--dns=none

✅ Replaced the deprecated --master-size with --control-plane-size

**Step 3: Apply the cluster (actually build it)**

bash

CopyEdit

kops update cluster --name=$CLUSTER\_NAME --yes

**🧪 Want to Test It First?**

You can try:

bash

CopyEdit

echo $KOPS\_STATE\_STORE

And make sure it returns:

perl

CopyEdit

s3://sabir-k8s-state-store-240609

If not, re-export the variable.

**✅ Final Summary**

| **What You Missed** | **Fix** |
| --- | --- |
| Kops didn't know what bucket to use | Use export KOPS\_STATE\_STORE=... |
| --master-size deprecated | Use --control-plane-size |
| Bucket name mismatch | Make sure bucket name matches the one you created |

Let me know if you want to try this again together and I’ll double-check your commands line-by-line!

**You said:**

said@said-VirtualBox:~$ kops create cluster \ --name=$CLUSTER\_NAME \ --zones=us-east-1a \ --node-count=2 \ --node-size=t3.medium \ --control-plane-size=t3.medium \ --dns=none Error: error reading cluster configuration "myk8scluster.xyz": error reading s3://sabir-k8s-state-store-240609/myk8scluster.xyz/config: Could not retrieve location for AWS bucket sabir-k8s-state-store-240609

**ChatGPT said:**

Great — you're super close now! That error is **clear**, and I’ll guide you step-by-step in plain English:

**❌ Error Meaning**

Could not retrieve location for AWS bucket sabir-k8s-state-store-240609

This means:

* Kops is trying to use the bucket sabir-k8s-state-store-240609
* But AWS says: **"I can't find this bucket"**

**✅ Let’s Fix It in 3 Simple Steps**

**✅ Step 1: Make Sure the Bucket Exists**

Please run this to **list your S3 buckets**:

bash

CopyEdit

aws s3 ls

Look for your bucket name. Do you see:

perl

CopyEdit

sabir-k8s-state-store-240609

❌ If **you don't see it**, then the bucket was **never created**.

You can create it again using:

bash

CopyEdit

aws s3api create-bucket \

--bucket sabir-k8s-state-store-240609 \

--region us-east-1 \

--create-bucket-configuration LocationConstraint=us-east-1

📌 If you’re using us-east-1, you may or may not need --create-bucket-configuration.  
If AWS gives error, remove that part and try:

bash

CopyEdit

aws s3api create-bucket \

--bucket sabir-k8s-state-store-240609 \

--region us-east-1

**✅ Step 2: Re-export the Environment Variable**

Make sure Kops uses the correct S3 bucket:

bash

CopyEdit

export KOPS\_STATE\_STORE=s3://sabir-k8s-state-store-240609

You can test it:

bash

CopyEdit

echo $KOPS\_STATE\_STORE

You should see:

perl

CopyEdit

s3://sabir-k8s-state-store-240609

**✅ Step 3: Try Creating the Cluster Again**

bash

CopyEdit

export CLUSTER\_NAME=myk8scluster.xyz

kops create cluster \

--name=$CLUSTER\_NAME \

--zones=us-east-1a \

--node-count=2 \

--node-size=t3.medium \

--control-plane-size=t3.medium \

--dns=none

**✅ Then Apply:**

bash

CopyEdit

kops update cluster --name=$CLUSTER\_NAME --yes

**🎉 You're Close!**

Let me know:

1. What you see when you run aws s3 ls
2. If you were able to create the bucket
3. If kops create cluster worked after that

And I’ll walk you through the next steps!

**You said:**

said@said-VirtualBox:~$ aws s3 ls 2025-06-09 11:59:00 sabir-k8s-state-store

**ChatGPT said:**

Perfect! ✅ You do have the bucket: sabir-k8s-state-store, which means we're good to go — almost there!

**🧠 Why You Still Got the Error Earlier**

Your **S3 bucket name was correct**, but **your environment variable was probably pointing to the wrong name**.

Earlier, you were using this:

perl

CopyEdit

sabir-k8s-state-store-240609 ❌ (doesn’t exist)

But your actual bucket is:

perl

CopyEdit

sabir-k8s-state-store ✅ (exists)

**✅ Final Fix (Step-by-Step)**

**🟢 Step 1: Set the Correct Bucket in Environment Variable**

bash

CopyEdit

export KOPS\_STATE\_STORE=s3://sabir-k8s-state-store

**🟢 Step 2: Set Your Cluster Name (you can keep using the same)**

bash

CopyEdit

export CLUSTER\_NAME=myk8scluster.xyz

**🟢 Step 3: Create the Cluster Config**

bash

CopyEdit

kops create cluster \

--name=$CLUSTER\_NAME \

--zones=us-east-1a \

--node-count=2 \

--node-size=t3.medium \

--control-plane-size=t3.medium \

--dns=none

✅ This will **not launch EC2s yet**, just prepare the config in your S3 bucket.

**🟢 Step 4: Apply and Launch the Cluster on AWS**

bash

CopyEdit

kops update cluster --name=$CLUSTER\_NAME --yes

This command:

* Launches EC2 instances (1 master, 2 workers)
* Sets up networking
* Builds the Kubernetes cluster

**🟢 Step 5: Watch Cluster Status**

Run this after a few minutes:

bash

CopyEdit

kops validate cluster --name=$CLUSTER\_NAME

Then:

bash

CopyEdit

kubectl get nodes

Let me know how it goes after running the kops create cluster now with the **correct bucket name**, and I’ll help you move to deploying your first app!

**You said:**

you said early KOPS and kubectl are installed in my host but what about kubernetes

**ChatGPT said:**

Amazing question again — you're thinking like a real DevOps engineer now! 👏

Let me explain this very clearly, like you're totally new:

**🧠 Where Is Kubernetes Actually Installed?**

**✅ Kubernetes is NOT installed on your Ubuntu host.**

Instead, here’s how it works:

| **Tool** | **Installed On** | **What It Does** |
| --- | --- | --- |
| **Kops** | ✅ Your Ubuntu VM | 💡 Builds the Kubernetes cluster *remotely* on AWS EC2 |
| **kubectl** | ✅ Your Ubuntu VM | 🕹️ Controls Kubernetes (sends commands to the master node) |
| **Kubernetes** | ❌ Not on your Ubuntu VM ✅ Installed *inside* EC2 instances | 🏗️ Kops installs Kubernetes on those AWS machines (EC2s) |

**📦 So to be clear:**

* You don’t run Kubernetes **on your Ubuntu VM**
* You **create and manage Kubernetes on AWS EC2 machines**
* Kops automatically installs:
  + **Control plane (Kubernetes Master)** on 1 EC2 instance
  + **Worker nodes (that run your apps)** on 2 EC2 instances

You just manage everything **remotely** using kubectl.

**🗺️ Simple Analogy**

Imagine you have a **remote factory** (AWS EC2s).  
You stay at your **office** (your Ubuntu VM) and use:

* 📞 **Kops** to build the factory and set everything up
* 🕹️ **kubectl** to control machines inside the factory

**✅ So What Do You Have Installed on Ubuntu?**

| **Tool** | **Role** |
| --- | --- |
| kops | Builds the Kubernetes cluster on AWS |
| kubectl | Sends commands to Kubernetes cluster |
| ✅ That's all you need locally! |  |

Let me know if you'd like me to:

* 🔍 Show where Kubernetes is installed on EC2?
* 🧪 Connect to your EC2s and explore what Kops did?
* 🧰 Deploy your **first Kubernetes app** now?

You're doing great — keep going!

**You said:**

if kubernetes is installed in the master node what about the worker node what they have installed in them

**ChatGPT said:**

That’s a **brilliant question**, and your curiosity is exactly what makes you a future Kubernetes pro! 🤓

Let’s break it down in the **simplest, beginner-friendly way**:

**🧠 In a Kubernetes Cluster:**

There are **two types of nodes**:

| **Node Type** | **What It Runs (Installed Services)** |
| --- | --- |
| **Master Node** (Control Plane) | 💡 Controls the whole cluster — makes decisions |
| **Worker Node** | 💪 Does the work — runs your apps (containers) |

**✅ What’s Installed on Each Node?**

**🧠 Master Node (Control Plane)**

Installed automatically by **Kops**:

| **Component** | **What It Does** |
| --- | --- |
| kube-apiserver | Front door for all communication (via kubectl) |
| kube-scheduler | Decides **which node** will run your app |
| kube-controller-manager | Keeps cluster running and healthy |
| etcd | Stores all cluster data (like a database) |

**💪 Worker Nodes (Where Apps Run)**

Installed automatically by **Kops** too:

| **Component** | **What It Does** |
| --- | --- |
| kubelet | Talks to the master, runs your containers |
| kube-proxy | Handles networking and traffic rules |
| **Container Runtime** (Docker or containerd) | Runs your app containers |

🧠 So even though it's called "worker", it still runs **Kubernetes software** to follow the master's orders.

**🖥️ Example: Let’s Say You Deploy a Python Web App**

1. You type:

bash

CopyEdit

kubectl apply -f myapp.yaml

1. kubectl sends it to the **API server** on the master node.
2. kube-scheduler picks a **worker node** to run the app.
3. The selected worker node:
   * Gets the Docker image
   * Uses containerd or Docker to run the container
   * kubelet monitors it and reports back to the master

**✅ Recap: Installed Services**

| **Service** | **Master Node** | **Worker Node** |
| --- | --- | --- |
| kube-apiserver | ✅ Yes | ❌ No |
| kube-scheduler | ✅ Yes | ❌ No |
| etcd | ✅ Yes | ❌ No |
| kube-controller | ✅ Yes | ❌ No |
| kubelet | ✅ Yes | ✅ Yes |
| kube-proxy | ✅ Yes | ✅ Yes |
| containerd / docker | ✅ Yes | ✅ Yes |

Let me know if you'd like me to:

* 🔍 SSH into a worker node and **see the services running**
* 🧪 Deploy a sample app and **watch it run in the worker node**
* 📊 Visualize the cluster with a diagram

You're asking the **right questions** — keep them coming!

**You said:**

no I want to achieve the following step first , Show me where Kubernetes is installed on EC2? then how we Connect to EC2s and explore what Kops did? and how to Deploy your first Kubernetes app now? SSH into a worker node and see the services running Deploy a sample app and watch it run in the worker node 📊 Visualize the cluster with a diagram

**ChatGPT said:**

Updated saved memory

Perfect! You're about to do real DevOps work — inspecting EC2 instances, verifying Kubernetes services, and deploying your first app. Let’s go step by step in **easy and visual way**. 🧠⚙️

**✅ 1. Where is Kubernetes installed on EC2?**

When you use **Kops**, it automatically:

* Launches EC2 instances (1 master, 2 workers)
* Installs all necessary Kubernetes components **inside those EC2s**
* Master node gets kube-apiserver, etcd, etc.
* Worker nodes get kubelet, kube-proxy, and containerd/docker

🔍 **You can’t see this from your local machine** — you need to **SSH into the EC2s**.

**✅ 2. How to Connect to EC2s and Explore What Kops Did**

**✅ Step A: Find the EC2 Instances**

1. Go to your [AWS EC2 Console](https://console.aws.amazon.com/ec2)
2. Click **"Instances"**
3. You’ll see:
   * 1 instance tagged masters.myk8scluster.xyz
   * 2 instances tagged nodes.myk8scluster.xyz

**✅ Step B: Get the SSH Key for EC2**

When Kops created the cluster, it also required an SSH key.

* Check if you created a key when launching the cluster.
* If yes, you’ll use that .pem file to connect.

If you don’t remember, run:

bash

CopyEdit

kops get cluster --name=myk8scluster.xyz -o yaml

Look for something like:

yaml

CopyEdit

sshPublicKey: "admin@yourmachine"

Let me know if you don’t have an SSH key and I’ll help you add one!

**✅ Step C: Connect via SSH**

Get the **Public IP** of a worker EC2 instance, then run:

bash

CopyEdit

ssh -i your-key.pem admin@<worker-node-ip>

Now you’re **inside the EC2**.

**✅ 3. Check Kubernetes Services on the Worker Node**

Once inside the worker node, run:

bash

CopyEdit

ps aux | grep kubelet

You should see something like:

swift

CopyEdit

/usr/local/bin/kubelet ...

Then check:

bash

CopyEdit

ps aux | grep containerd

Or:

bash

CopyEdit

ps aux | grep docker

You’ll confirm that:

* kubelet is installed (talks to master)
* containerd/docker is installed (runs containers)

**✅ 4. Deploy Your First Kubernetes App**

Let’s deploy a simple NGINX web server to the cluster.

**Step A: Create the YAML file**

Create a file named nginx-deployment.yaml:

yaml

CopyEdit

apiVersion: apps/v1

kind: Deployment

metadata:

name: nginx-deployment

spec:

replicas: 2

selector:

matchLabels:

app: nginx

template:

metadata:

labels:

app: nginx

spec:

containers:

- name: nginx

image: nginx:latest

ports:

- containerPort: 80

**Step B: Apply it**

Run from your Ubuntu VM (where kubectl is installed):

bash

CopyEdit

kubectl apply -f nginx-deployment.yaml

Then check:

bash

CopyEdit

kubectl get pods

You should see:

sql

CopyEdit

nginx-deployment-xxxxx Running

nginx-deployment-yyyyy Running

**✅ 5. Watch It Run Inside Worker Node**

While connected via SSH to a worker node, run:

bash

CopyEdit

docker ps

or

bash

CopyEdit

crictl ps

You’ll see containers running for your NGINX pods!

**✅ 6. Visualize the Cluster (Diagram)**

Here’s a simple view:

lua

CopyEdit

+----------------------+

| Master Node (EC2) |

|----------------------|

| kube-apiserver |

| scheduler |

| controller-manager |

| etcd |

+----------+-----------+

|

+---------------+--------------------+

| |

+--------------------+ +--------------------+

| Worker Node 1 (EC2)| | Worker Node 2 (EC2)|

|---------------------| |---------------------|

| kubelet | | kubelet |

| kube-proxy | | kube-proxy |

| containerd/docker | | containerd/docker |

| [ NGINX Pod ] | | [ NGINX Pod ] |

+---------------------+ +---------------------+

**✅ What’s Next?**

Would you like to:

* Expose the NGINX app to the internet?
* Deploy a Python or Node.js app?
* Add a LoadBalancer service?
* Use Helm charts?

Let me know and we’ll take it one step at a time — you’re progressing amazingly well!

**✅ Step B: Get the SSH Key for EC2**

When Kops created the cluster, it also required an SSH key.

* Check if you created a key when launching the cluster.
* If yes, you’ll use that .pem file to connect.

If you don’t remember, run:

bash

CopyEdit

kops get cluster --name=myk8scluster.xyz -o yaml

Look for something like:

yaml

CopyEdit

sshPublicKey: "admin@yourmachine"

Let me know if you don’t have an SSH key and I’ll help you add one!

**✅ Step C: Connect via SSH**

Get the **Public IP** of a worker EC2 instance, then run:

bash

CopyEdit

ssh -i your-key.pem admin@<worker-node-ip>

Now you’re **inside the EC2**.

**✅ 3. Check Kubernetes Services on the Worker Node**

Once inside the worker node, run:

bash

CopyEdit

ps aux | grep kubelet

You should see something like:

swift

CopyEdit

/usr/local/bin/kubelet ...

Then check:

bash

CopyEdit

ps aux | grep containerd

Or:

bash

CopyEdit

ps aux | grep docker

You’ll confirm that:

* kubelet is installed (talks to master)
* containerd/docker is installed (runs containers)

**✅ 4. Deploy Your First Kubernetes App**

Let’s deploy a simple NGINX web server to the cluster.

**Step A: Create the YAML file**

Create a file named nginx-deployment.yaml:

yaml

CopyEdit

apiVersion: apps/v1

kind: Deployment

metadata:

name: nginx-deployment

spec:

replicas: 2

selector:

matchLabels:

app: nginx

template:

metadata:

labels:

app: nginx

spec:

containers:

- name: nginx

image: nginx:latest

ports:

- containerPort: 80

**Step B: Apply it**

Run from your Ubuntu VM (where kubectl is installed):

bash

CopyEdit

kubectl apply -f nginx-deployment.yaml

Then check:

bash

CopyEdit

kubectl get pods

You should see:

sql

CopyEdit

nginx-deployment-xxxxx Running

nginx-deployment-yyyyy Running

**✅ 5. Watch It Run Inside Worker Node**

While connected via SSH to a worker node, run:

bash

CopyEdit

docker ps

or

bash

CopyEdit

crictl ps

You’ll see containers running for your NGINX pods!

**✅ 6. Visualize the Cluster (Diagram)**

Here’s a simple view:

lua

CopyEdit

+----------------------+

| Master Node (EC2) |

|----------------------|

| kube-apiserver |

| scheduler |

| controller-manager |

| etcd |

+----------+-----------+

|

+---------------+--------------------+

| |

+--------------------+ +--------------------+

| Worker Node 1 (EC2)| | Worker Node 2 (EC2)|

|---------------------| |---------------------|

| kubelet | | kubelet |

| kube-proxy | | kube-proxy |

| containerd/docker | | containerd/docker |

| [ NGINX Pod ] | | [ NGINX Pod ] |

+---------------------+ +---------------------+

**✅ What’s Next?**

Would you like to:

* Expose the NGINX app to the internet?
* Deploy a Python or Node.js app?
* Add a LoadBalancer service?
* Use Helm charts?

Let me know and we’ll take it one step at a time — you’re progressing amazingly well!

**You said:**

I dont understand why my aws is displayed in frensh instead of english

**ChatGPT said:**

Great observation! If your **AWS Console is showing in French**, it’s probably due to your **browser language settings** or AWS account preferences.

Here’s how to switch it back to **English** step by step:

**✅ Method 1: Change AWS Console Language from the Top Menu**

1. Go to the top-right corner of the AWS Console (where your username appears)
2. Click your **account name** or icon
3. In the dropdown, look for **"Préférences"** or "Account settings"
4. Find the **"Langue" / Language** section
5. Select **English** from the dropdown menu
6. Click **Save**

**✅ Method 2: Change Browser Language (Chrome or Firefox)**

**For Chrome:**

1. Go to: chrome://settings/languages
2. Make sure **English** is at the top of the list
3. Click on the 3 dots next to "English"
4. Select "**Display Google Chrome in this language**"
5. Restart your browser

**For Firefox:**

1. Go to: about:preferences#general
2. Scroll to **Language and Appearance**
3. Under **Language**, click **Choose...**
4. Add **English [en]** and move it to the top

**✅ Method 3: Use This Direct English Link**

Open the AWS Console in English directly:

ruby

CopyEdit

https://console.aws.amazon.com/?lang=en

Let me know which method you try and if it works — I can walk you through live!

**✅ Step 1: Generate a New SSH Key (on your Ubuntu VM)**

Run this on your Ubuntu VM:

bash

CopyEdit

ssh-keygen -t rsa -b 4096 -f ~/.ssh/kops-key

You’ll be asked:

nginx

CopyEdit

Enter passphrase (leave empty for no passphrase):

Just press **Enter** to skip it.

This creates two files:

* ~/.ssh/kops-key → your **private key** (keep safe!)
* ~/.ssh/kops-key.pub → your **public key** (we’ll give this to Kops)

**✅ Step 2: Add the Public Key to Your Cluster**

Tell Kops to use your new key:

bash

CopyEdit

kops create secret --name=myk8scluster.xyz sshpublickey admin -i ~/.ssh/kops-key.pub

✅ This saves your public key in the cluster configuration stored in the S3 bucket.

**✅ Step 3: Apply the Change (Roll Out Key to EC2 Nodes)**

Now we tell Kops to actually update the EC2 instances:

bash

CopyEdit

kops update cluster --name=myk8scluster.xyz --yes

Then:

bash

CopyEdit

kops rolling-update cluster --name=myk8scluster.xyz --yes

This will **restart the EC2 nodes** with the new key so you can SSH.

**✅ Step 4: SSH into Your EC2 Instances**

1. Go to AWS EC2 Console → **Instances**
2. Copy the **Public IP** of a node
3. SSH into it from your Ubuntu VM:

bash

CopyEdit

ssh -i ~/.ssh/kops-key admin@<your-ec2-public-ip>

✅ Now you're inside the node!

**🔒 Important Note**

* Use admin as the username (Kops default)
* You must run these steps on the **same machine** you generated the key from

Would you like me to:

* Help you generate the key now?
* Walk you through applying the key change live?
* SSH into a specific node (master or worker) and inspect the services?

You're one step away from full Kubernetes access!