**How to Create a Kubernetes Cluster with the AWS EKS CLI**

**Table of Contents**

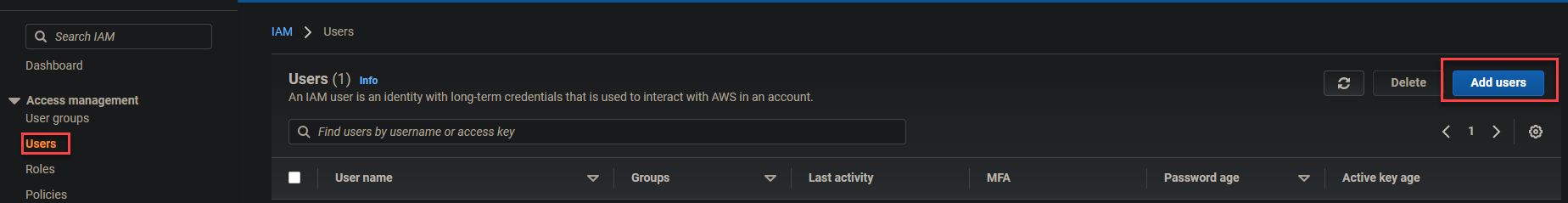
* Prerequisite
* Creating an Admin User
* Launching an EC2 Instance
* Configuring the AWS CLI Tool
* Configuring Amazon EKS Command-Line Tool (eksctl)
* Provisioning your EKS Cluster
* Deploying an Application on EKS Cluster
* Testing the Highly-available Kubernetes Control
* Conclusion

**Creating an Admin User**

Before creating a Kubernetes cluster, you’ll create an admin user. An admin user lets you log in to the AWS console to configure your cluster. Kick off this tutorial by creating a user with administrator permissions via the AWS Console.

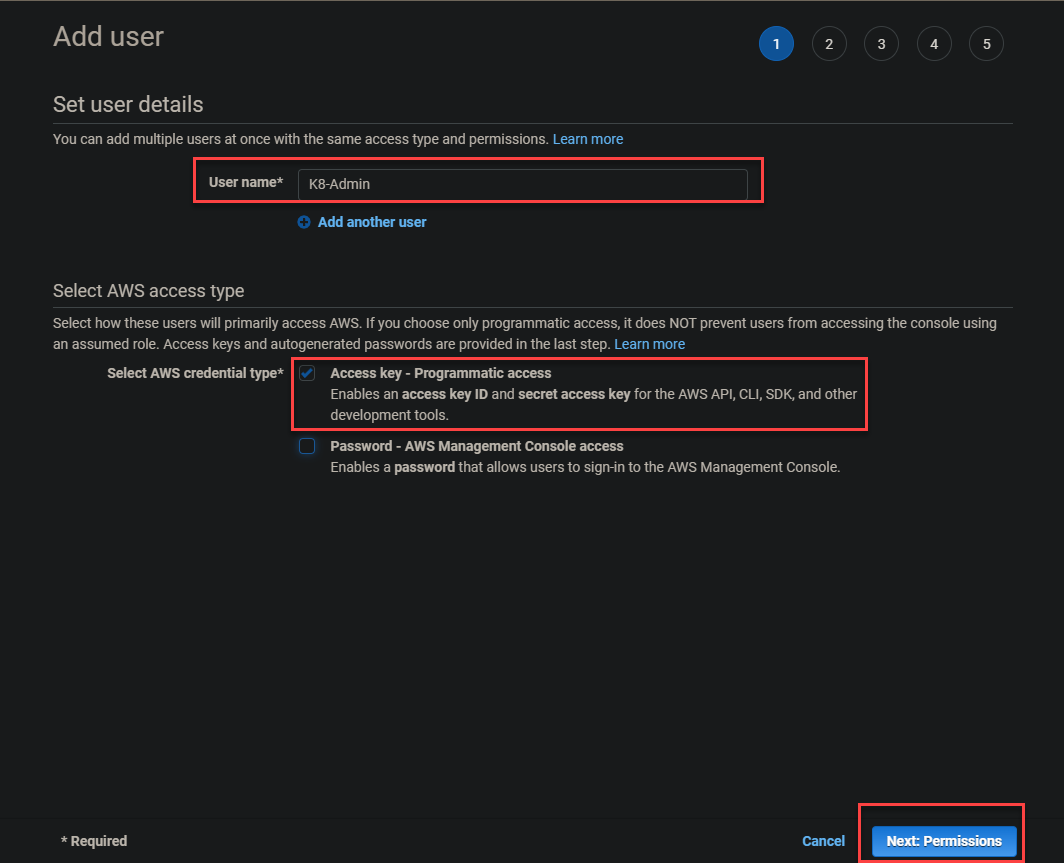
1. Log into your [AWS Console](https://aws.amazon.com/premiumsupport/knowledge-center/sign-in-console/), and navigate to your IAM dashboard.

Click on **Users** (left panel) —> **Add Users** (top-right) shown below to initialize adding users.

Initializing User Creation

2. Next, provide a username in the **User name** field, here **K8-Admin** is used, check the **Access key – Programmatic access** option, and click **Next: Permissions.**

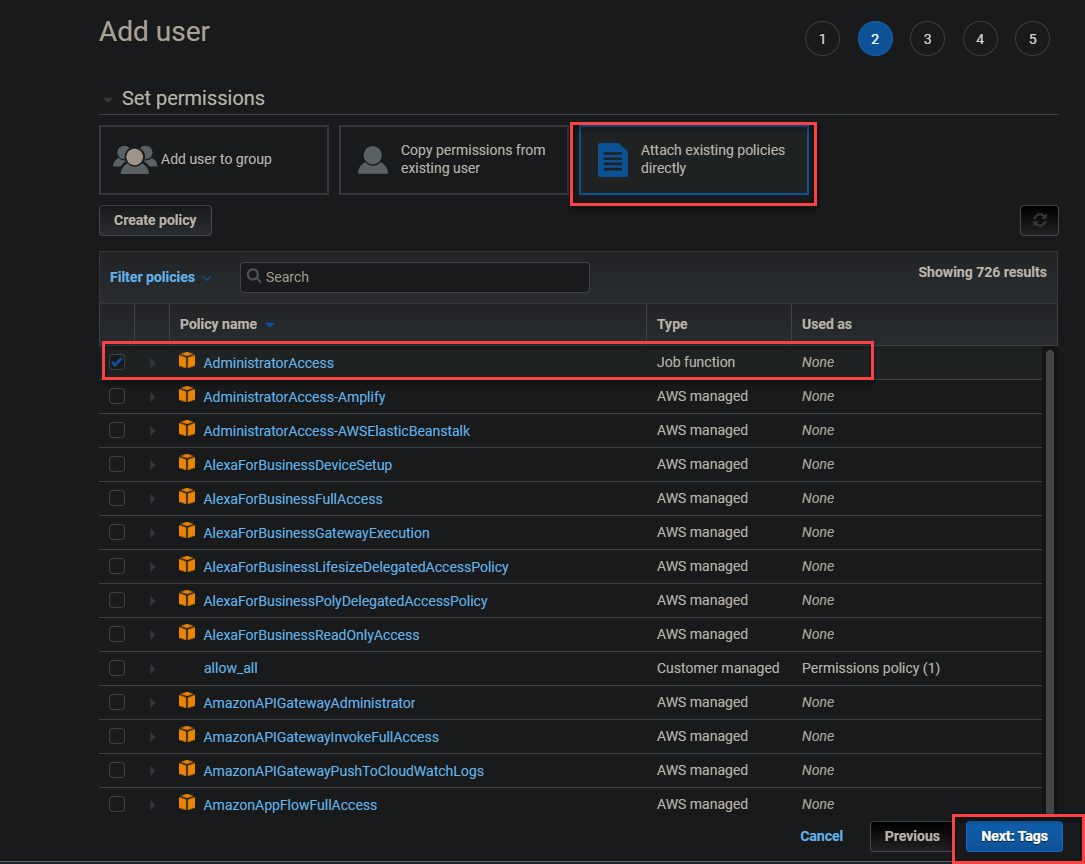
You’re selecting the **Access key – Programmatic access** option as it’s programmatically accessible. As a result, can use an application to communicate directly to AWS on what actions to take.

Configuring User Details

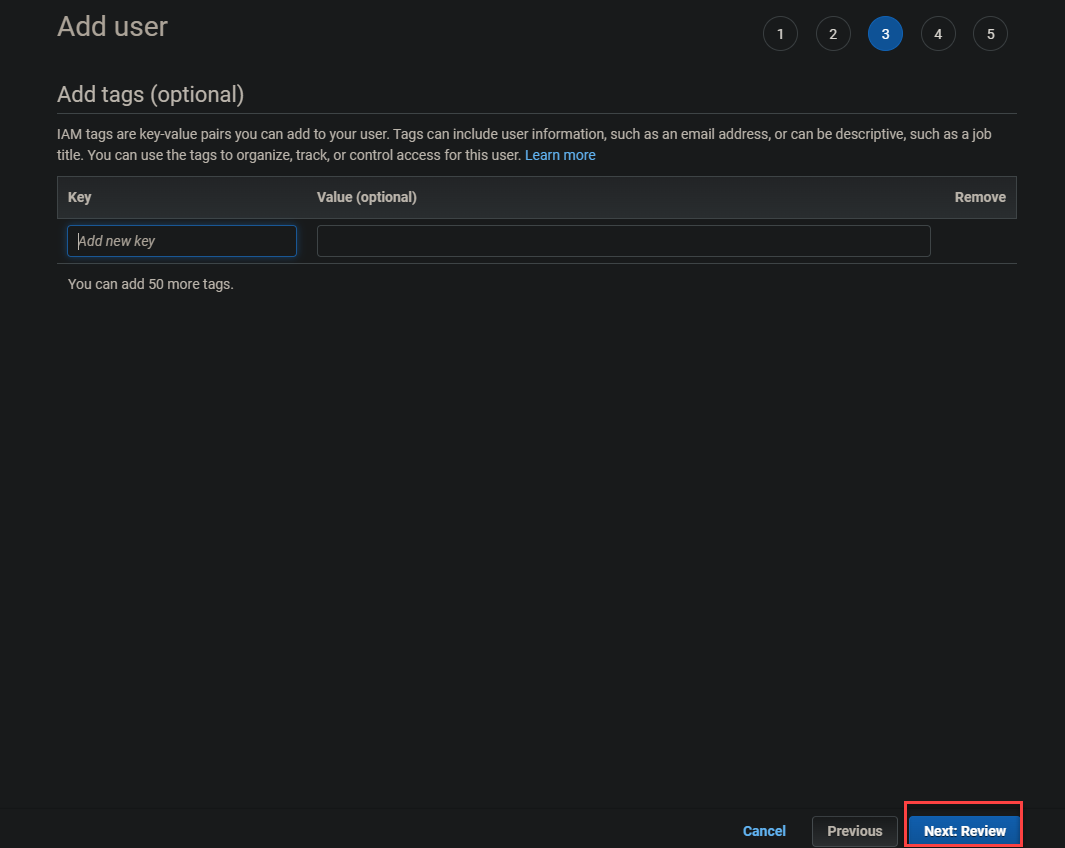
3. Click the **Attach existing policies directly** option, check the **AdministratorAccess** policy, and click **Next: Tags.**

The **AdministratorAccess** policy gives the user (K8-Admin) full access to AWS, and more as follow:

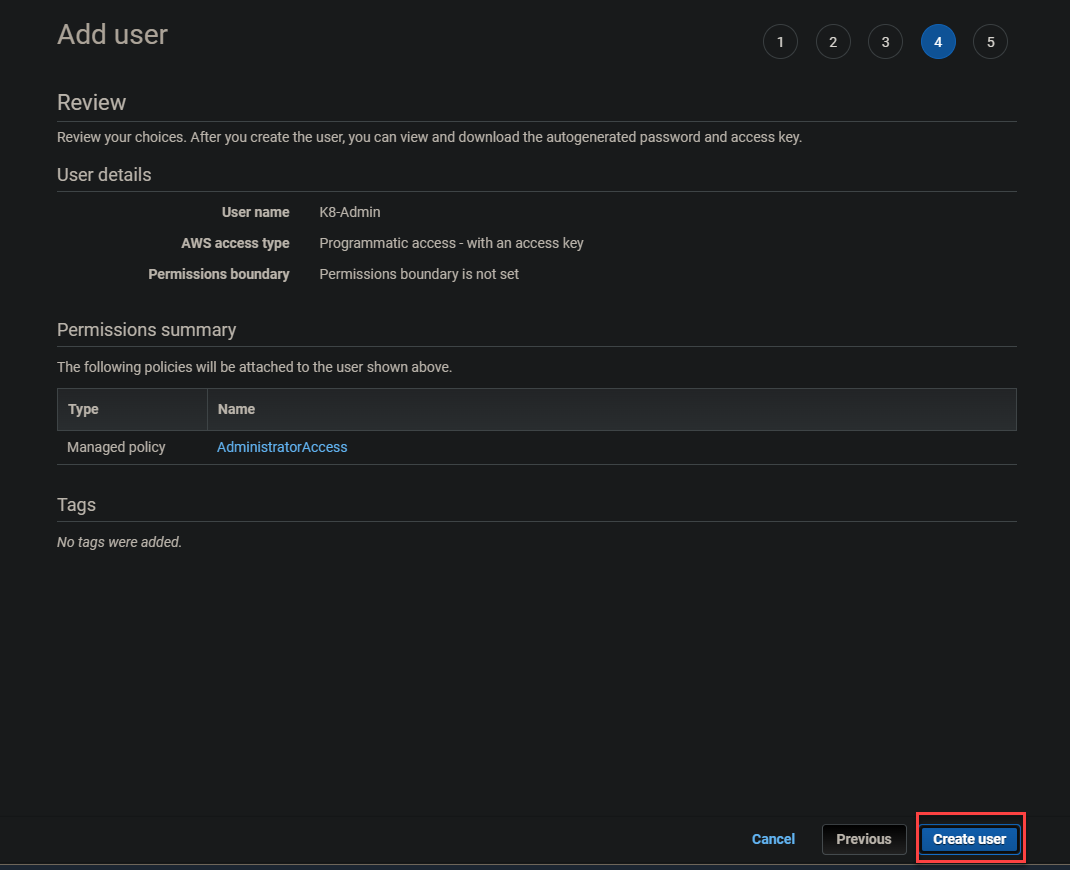
* Allows the user to use [CloudFormation](https://aws.amazon.com/cloudformation/)
* Create EC2 instances and [CloudWatch](https://aws.amazon.com/cloudwatch/) logs
* Configure [elastic load balancers](https://aws.amazon.com/elasticloadbalancing/).

Setting up AdministratorAccess policies

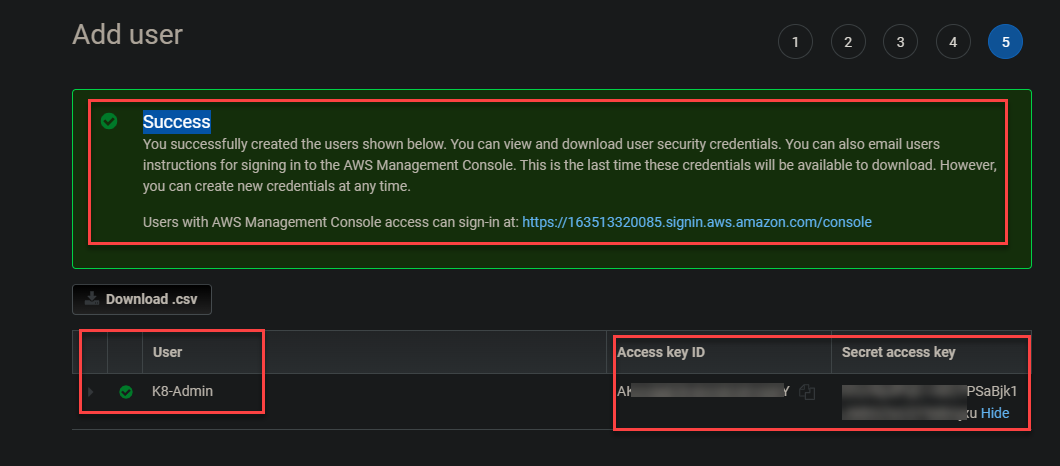
4. Click **Next: Review** to skip adding tags.

Skipping the tags screen

5. Finally, review the user details and click **Create user** to finalize creating the admin user.

Creating the admin user

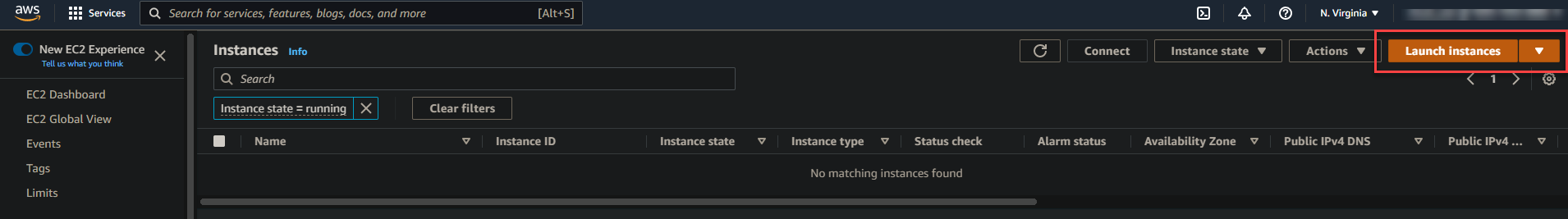
Once the admin user creation is complete, you will get a **Success** message at the top of the screen, like the one below. Note the **Access key ID** and **Secret** **access key** as you will use these keys to log in later.

Previewing the admin user keys

**Launching an EC2 Instance**

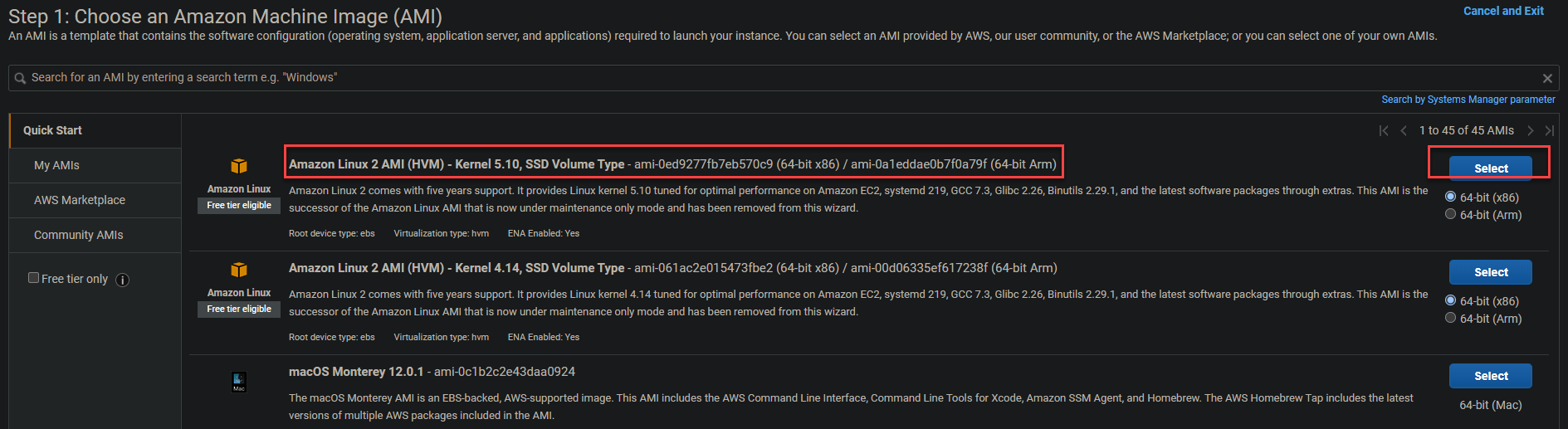
Now that you’ve created the K8-Admin, you can now create your first EC2 instance. You’ll use this instance as your master node, where you run your commands to create the cluster.

1. Navigate to your EC2 dashboard, click on **EC2,** then **Launch Instances** at the right-most part of the page. Doing so redirects your browser to a page where you can choose an [Amazon Machine Image (AMI)](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AMIs.html) (step two).

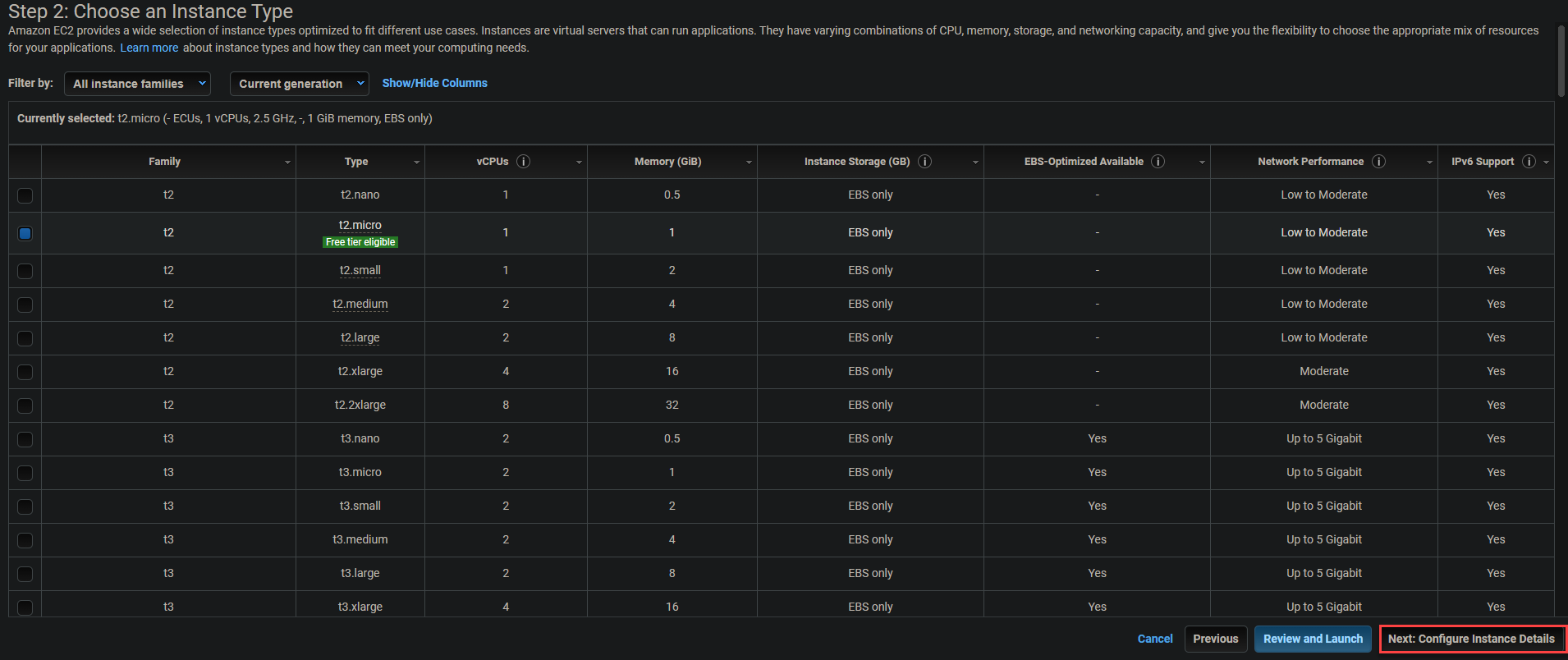
Launching an EC2 Instance

2. Next, click on **Select** beside (right-most) the **Amazon Linux 2 AMI (HVM)** from the list, as shown below.

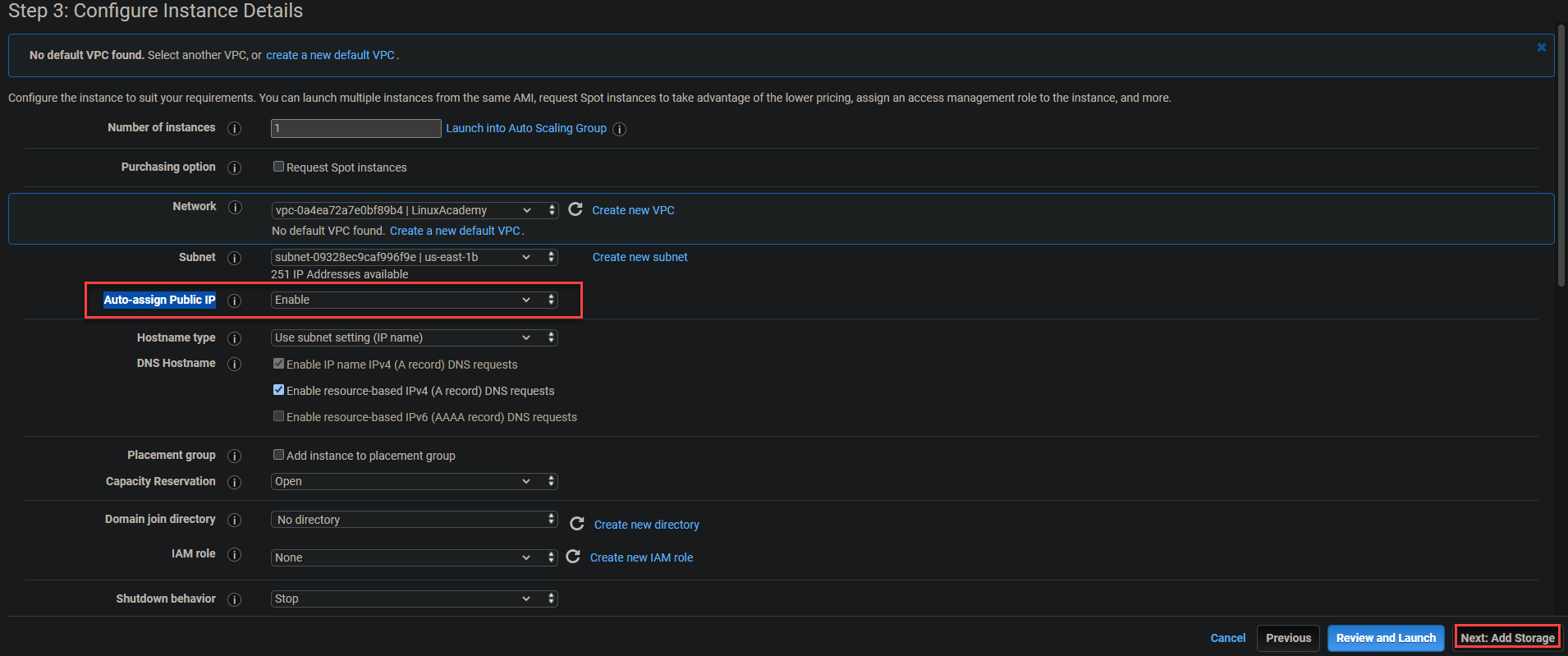
**Amazon Linux 2 AMI (HVM)** provides Linux kernel 5.10 tuned for optimal performance of the latest generation of hardware. This AMI also has many features required by production-level Kubernetes clusters.

Selecting Amazon Linux 2 AMI (HVM)

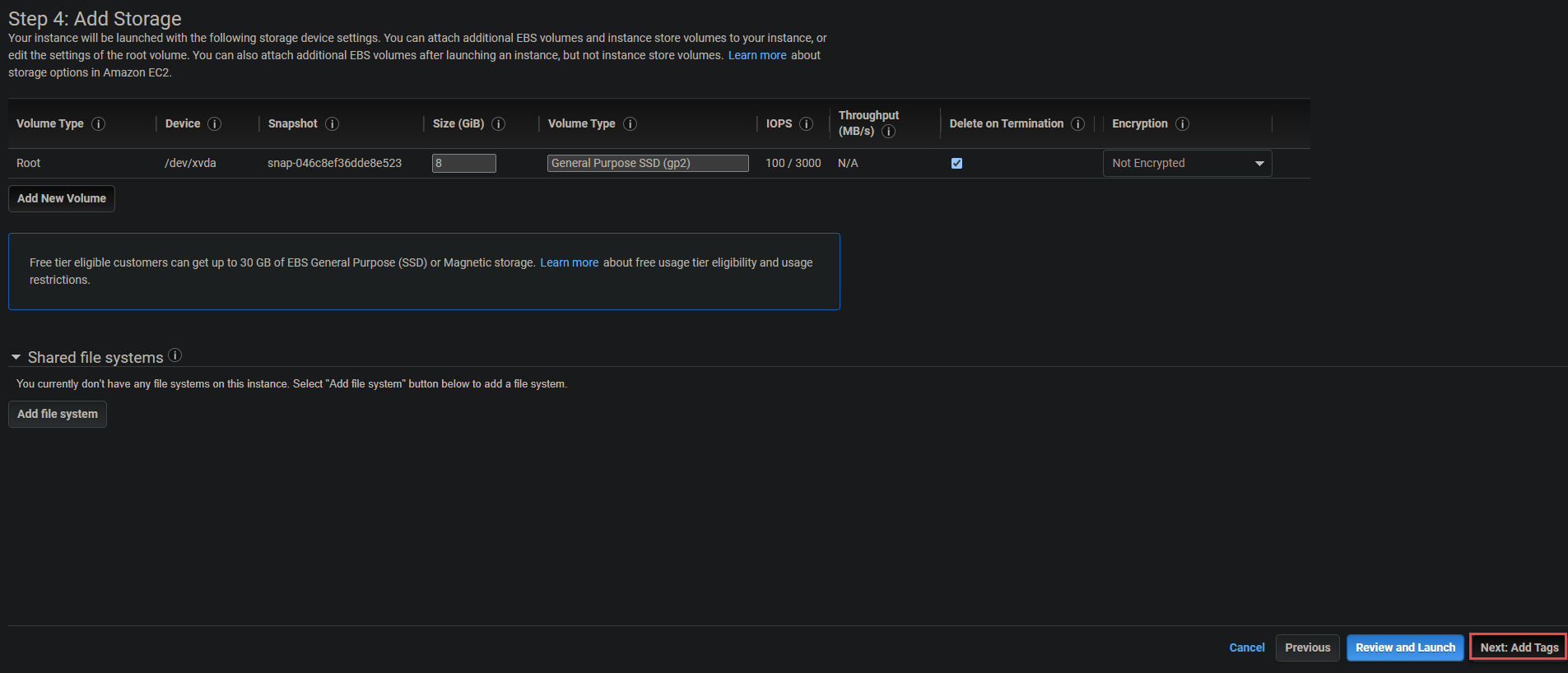
3. Keep the default ([t2.micro](https://aws.amazon.com/ec2/instance-types/t2/)) for the instance type and click **Next: Configure Instance Details** to configure the instance.

Previewing the instance type

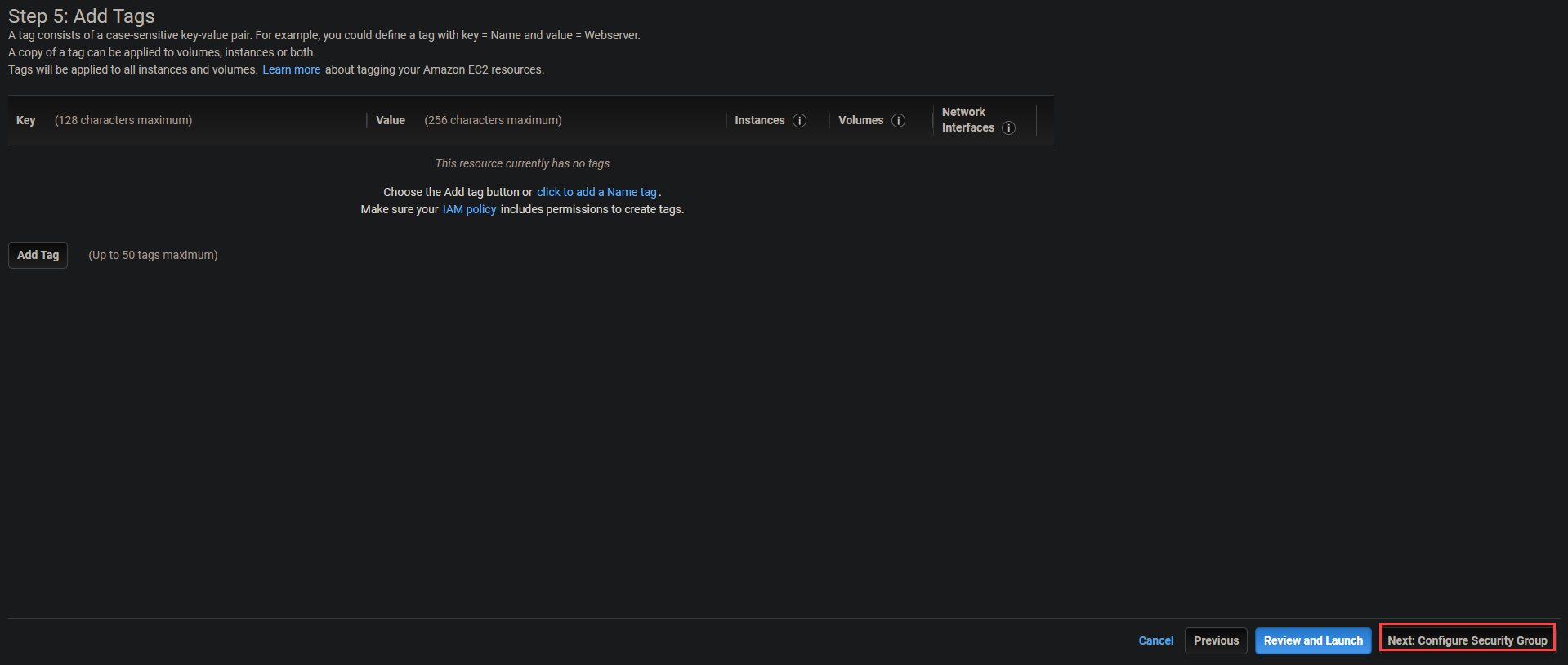
4. Enable the **Auto-assign Public IP** option and **Next**: **Add Storage**. This option ensures each of your containers can access the public IP of your Kubernetes master node and your EC2 instances.

Configuring instance details

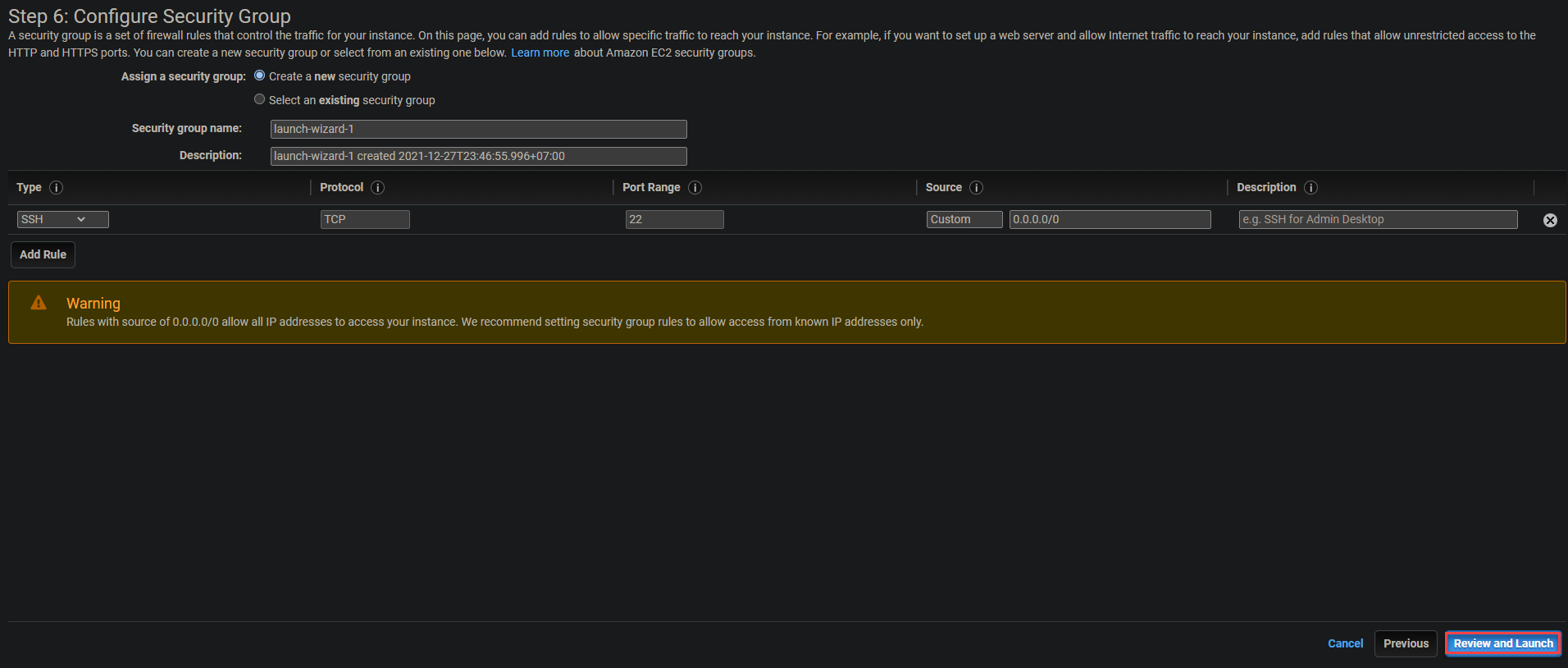
5. Keep the default (**Root**) **in the Add Storage page** and click **Next**: **Add tags.** The **Root** volume is required to read and write data from within the instance.

Configuring the storage

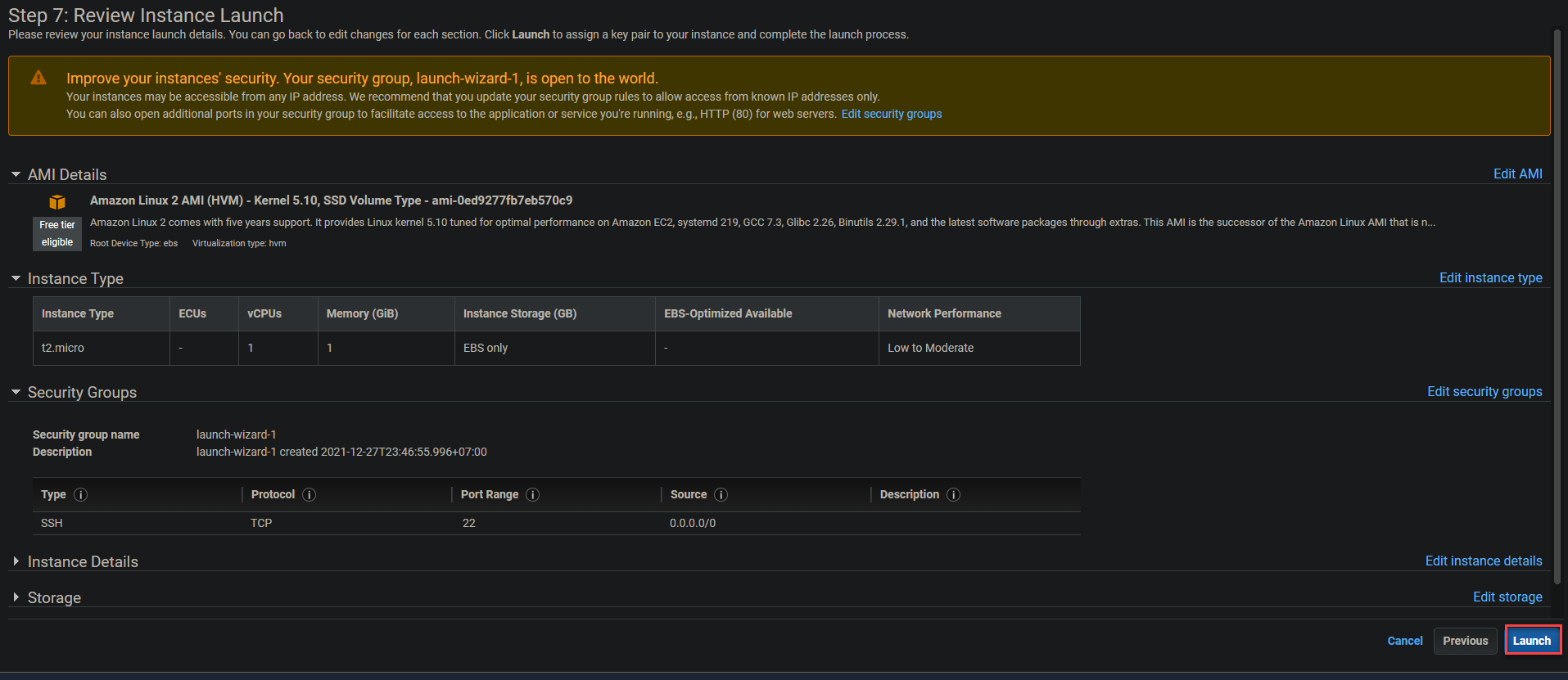
6. Skip adding tags and click on **Next**: **Configure Security Group.**

Previewing the tags

7. Keep the defaults on the security group, like shown below, and click on **Review and Launch**.

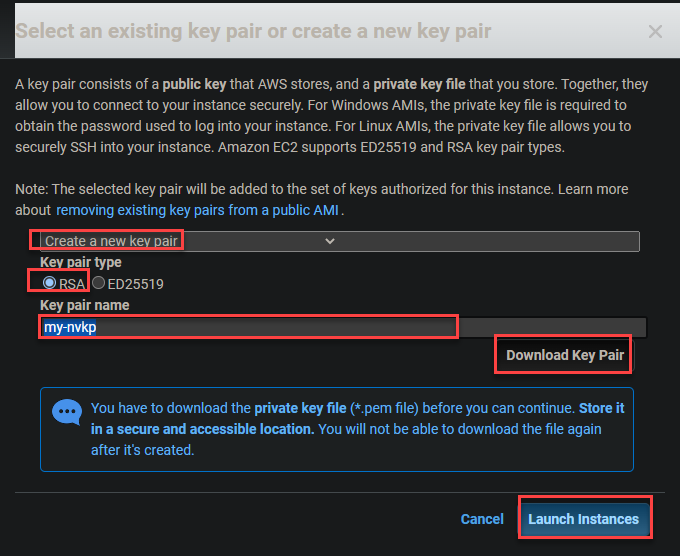
Previewing the Security Group

8. Review the instance launch details and click **Launch** to launch the instance. A pop-up will appear where you can choose to select an existing [key pair](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-key-pairs.html) or create a new one (step nine).

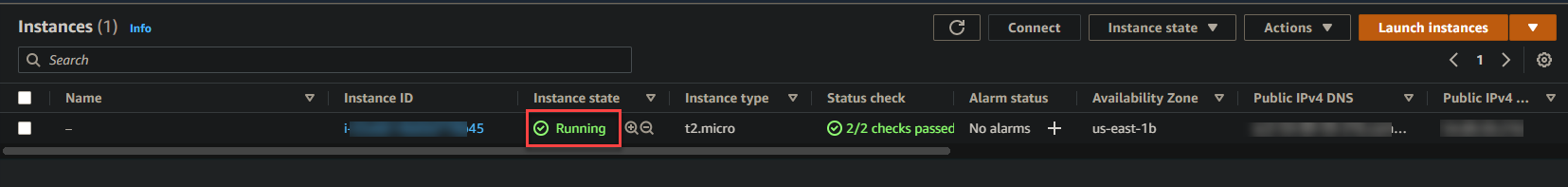
Launching an instance

9. In the dialog pop-up, configure the key pair with the following:

* Select **Create a new key pair** in the dropdown box.
* Choose [RSA](https://www.encryptionconsulting.com/education-center/what-is-rsa/) as the **Key pair type.**
* Provide your preferred **Key pair name**. But for this tutorial, the key pair name is set to **my-nvkp.**
* Click on **Download Key Pair**, then **Launch Instances.**

Creating a new key pair

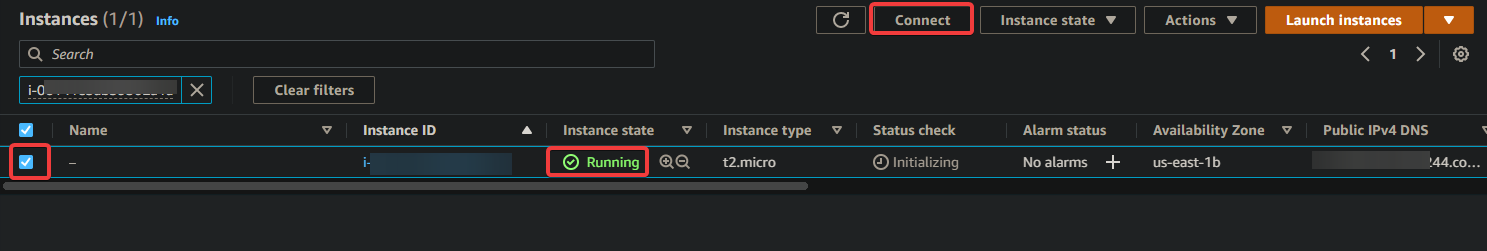
Your instance may take a minute or two to launch completely. Once your instance is running, you’ll see it listed in your EC2 dashboard, as shown below.

Previewing the newly-created instance

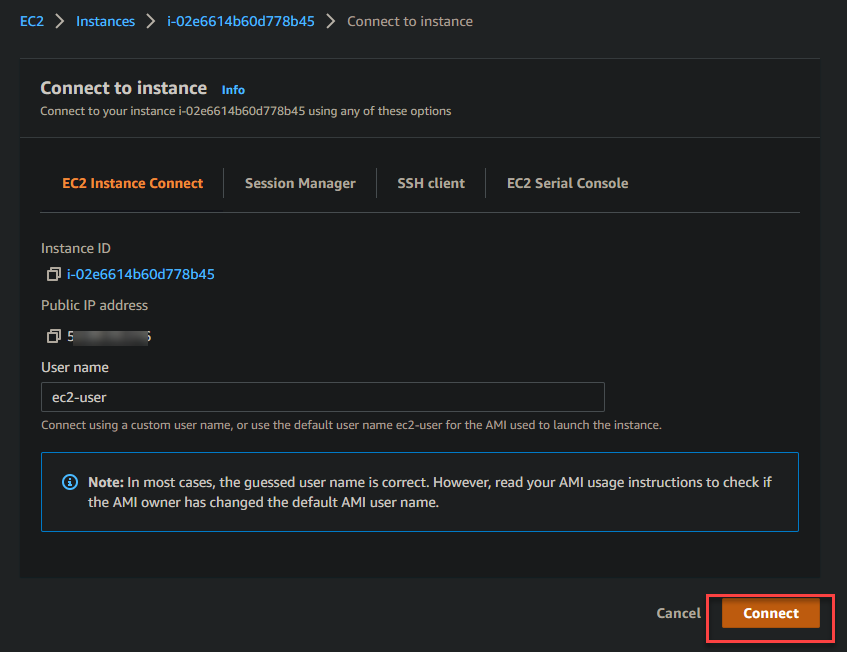
**Configuring the AWS CLI Tool**

Now that your instance is running, it’s time to configure the command line (CLI) tools. Using the CLI tools in conjunction with your AWS account is essential in creating your Kubernetes cluster.

1. From your EC2 dashboard, check the box to select the instance, as shown below. Click on **Connect** to initialize connecting to the instance.

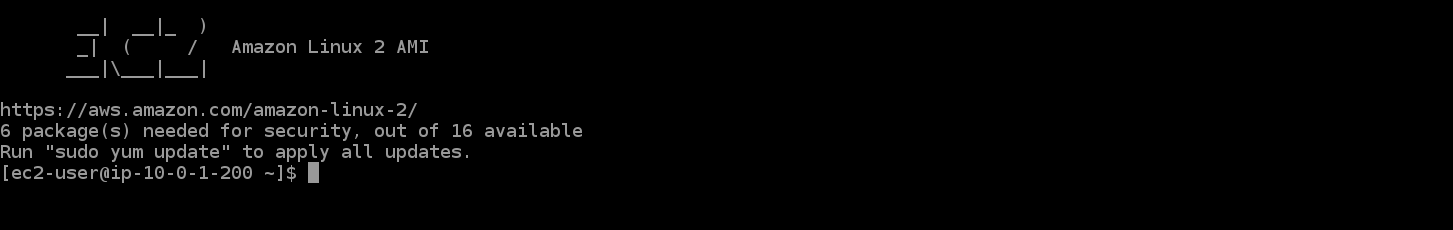
Connecting to the Ec2 instance.

2. Next, click on the **Connect** button to connect to the instance you previously selected in step one.

Connecting to the instance

Once you’ve connected to your EC2 instance, your browser redirects to the interactive terminal shown below as your temporary SSH session with your EC2 instance.

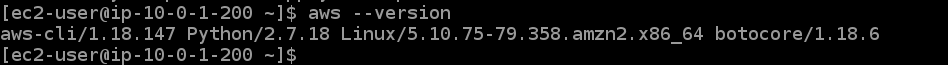
The interactive terminal lets you connect to the command line and run administrative commands to your new instance.

Previewing the interactive terminal

3. Run the **aws** command below to check your CLI version.

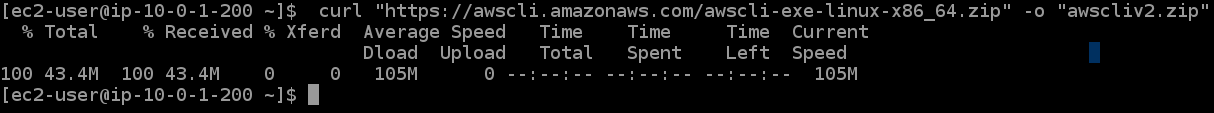
aws --version

As you can see from the output below, you are running version 1.18.147 on your Amazon Linux 2 instance, which is out of date. You need to download and install AWS CLI version 2+ to ensure you can access all of the Kubernetes features (step three).

Checking the AWS CLI version

4. Now, run the **curl** command below to download the CLI tool v2+ and save it in a zip file named **awscliv2.zip**.

curl "https://awscli.amazonaws.com/awscli-exe-linux-x86\_64.zip" -o "awscliv2.zip"

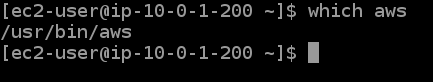
Downloading the CLI tool v2+

5. Run the following commands to unzip the downloaded file and determine where the outdated AWS CLI is installed.

unzip awscliv2.zip

which aws

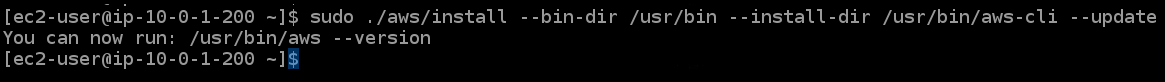
As you can see from the output below, the outdated AWS CLI is installed at **/usr/bin/aws**. You need to update this path with the updated version.

Updating outdated AWS CLI

6. Run the command below to perform the following and –update the AWS CLI’s install path on your instance:

* Install the updated AWS CLI tools on your Amazon Linux 2 instance (**sudo ./aws/install**).
* Set the directory (**--install-dir /usr/bin/aws-cli**) where to install the CLI tools. Doing so lets you transfer the updated AWS CLI to other instances without reinstalling the CLI tools.
* Update (**--update**) your current shell environment with the new path for AWS CLI tools if there’s one in your current environment.

sudo ./aws/install --bin-dir /usr/bin --install-dir /usr/bin/aws-cli --update

Installing the CLI tool v2

7. Rerun the **aws --version** command below to check that the updated AWS CLI is installed correctly.

aws --version

The AWS CLI version installed is **2.4.7**, as shown below, which is the latest AWS CLI version is 2.4.7 at the time of writing.

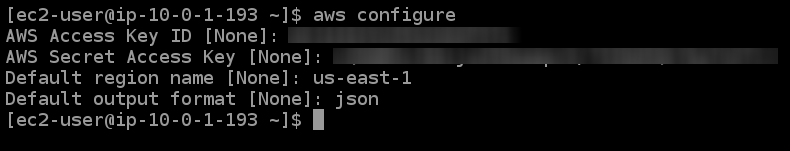
Checking the AWS CLI updated versionChecking the AWS CLI updated version

8. Next, run the **aws configure** command to configure your instance with the new AWS CLI tools.

aws configure

Enter the appropriate values in the prompts as per below:

* **AWS Access Key ID [None]** – Enter the Access Key ID you noted in the previous “Creating Your Admin User” section.
* **AWS Secret Access Key [None]** – Enter the Secret Access Key you noted in the previous “Creating Your Admin User” section.
* **Default region name [None]** – Select a supported region, like **us-east-1**.
* **Default output format [None]** – Enter **json**, since JSON format is the preferred standard for use with Kubernetes.

Configuring the AWS Environment

**Configuring Amazon EKS Command-Line Tool (eksctl)**

Since your goal is to create a Kubernetes cluster with AWS EKS CLI, you’ll also configure [Amazon EKS (eksctl)](https://docs.aws.amazon.com/eks/latest/userguide/getting-started-eksctl.html) command-line tool. This tool lets you create and manage Kubernetes clusters on Amazon EKS.

[1. Install the latest version](https://docs.aws.amazon.com/eks/latest/userguide/install-kubectl.html) of the Kubernetes command-line tool ([kubectl](https://kubernetes.io/docs/reference/kubectl/kubectl/" \t "_blank)) on your EC2 instance. This tool allows you to run commands against Kubernetes clusters.

2. Next, run the **curl** command below to retrieve the latest **eksctl** release from GitHub to your */tmp* directory as a *.tar.gz* file, then extract the archive content into the /tmp directory.

Run the below commands to perform the following:

* Retrieve the latest **eksctl** release from GitHub (**--location**) as *.tar.gz* archive (**"<https://github.com/weaveworks/eksctl/releases/latest/download/eksctl\_$>(uname -s)\_amd64.tar.gz"**)
* Extract the archive’s content to the */tmp* directory (**tar xz -C /tmp**), while the **--silent** flag suppresses the command’s progress output.
* Move (**sudo mv**) the eksctl binary (**/tmp/eksctl**) to the path where you installed the AWS CLI (**/usr/bin**)

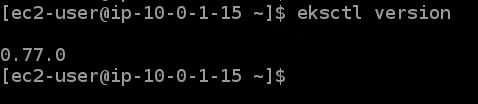
curl --silent --location "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl\_$(uname -s)\_amd64.tar.gz" | tar xz -C /tmp

sudo mv /tmp/eksctl /usr/bin

3. Finally, run the command below to confirm you’ve successfully installed **eksctl**.

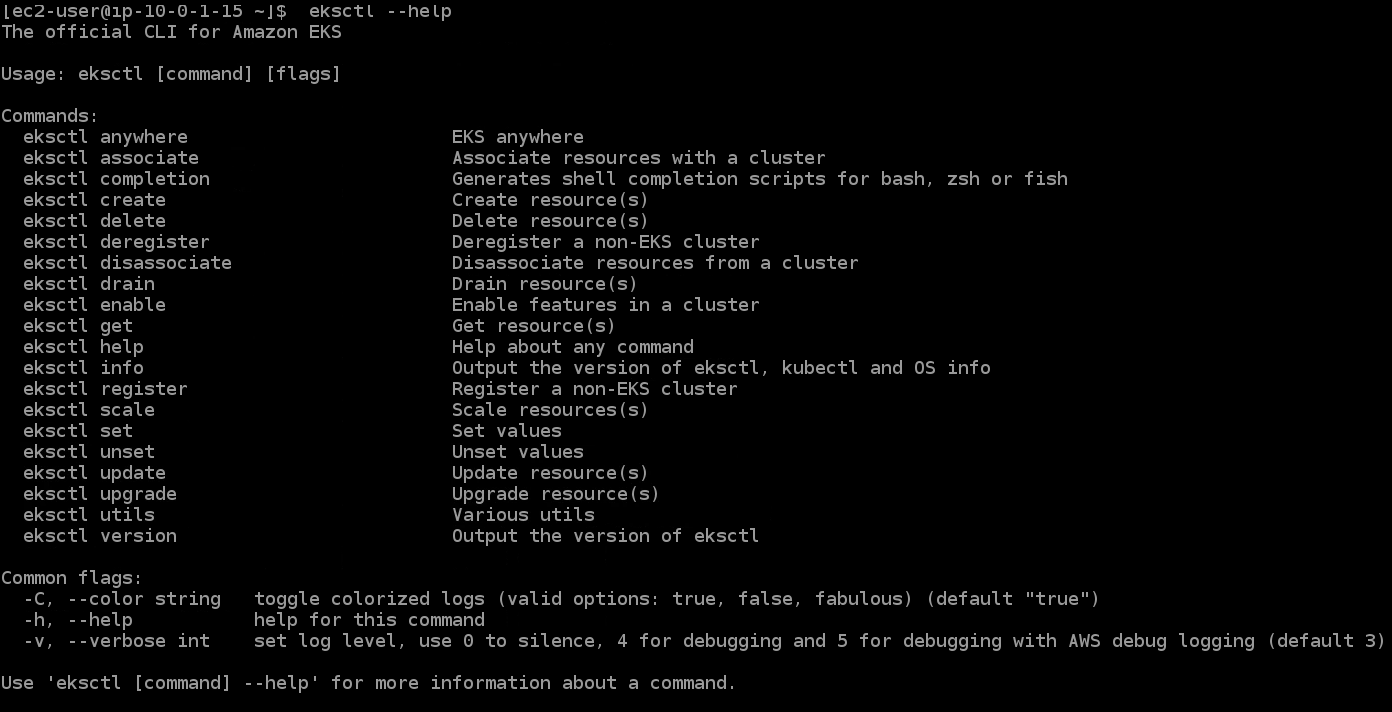
eksctl version

The output below confirms that you’ve installed **eksctl** successfully.

Checking the eksctl CLI tool version

If you’re new to **eksctl**, you can run the command below to get a list of all of the supported **eksctl** commands and their usage.

eksctl --help

Previewing the eksctl help page

**Provisioning your EKS Cluster**

Now that you have configured eksctl, you can now provision your first EKS Cluster with **eksctl** commands.

Run the **eksctl** command below to create your first cluster and perform the following:

* Create a 3-node Kubernetes cluster named **dev** with one node type as **t3.micro** and region as **us-east-1**.
* Define a minimum of one node (**--nodes-min 1**) and a maximum of four-node (**--nodes-max 4**) for this node group managed by EKS. The node group is named **standard-workers**.
* Create a node group with the name **standard-workers** and select a machine type for the **standard-workers** node group.

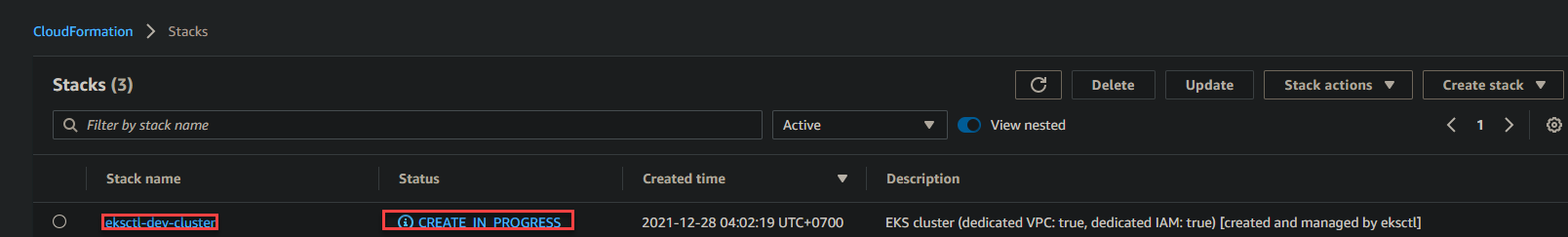
eksctl create cluster --name sai --version 1.30 --region ap-south-1 --nodegroup-name standard-workers --node-type t2.micro --nodes 3 --nodes-min 1 --nodes-max 4 --managed

Provisioning your EKS Cluster

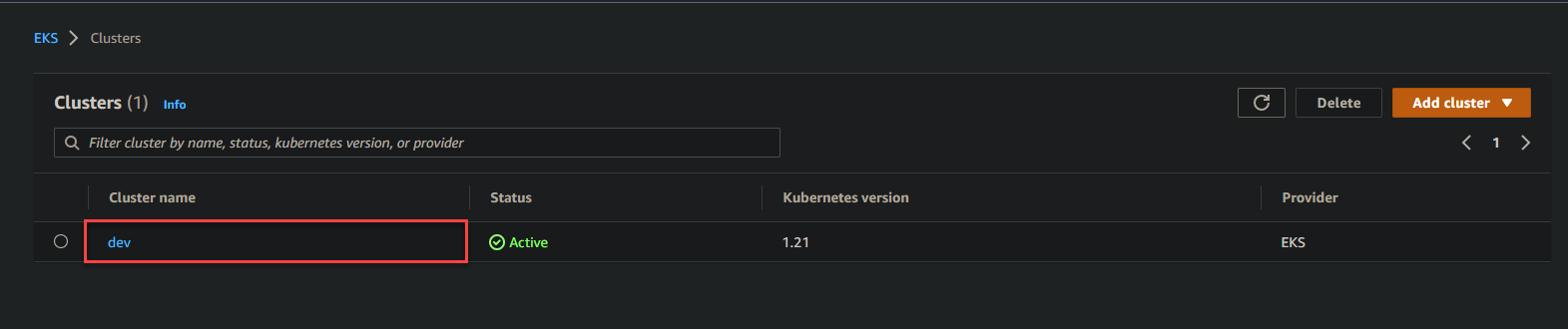
2. Navigate to your [CloudFormation](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/Welcome.html) dashboard to see the actions taken by the command. The **eksctl create cluster** command uses [CloudFormation](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/Welcome.html) to provision the infrastructure in your AWS account.

**Related:**[Deploying Infrastructure with AWS CLI and CloudFormation](https://adamtheautomator.com/aws-cli-and-cloudformation/)

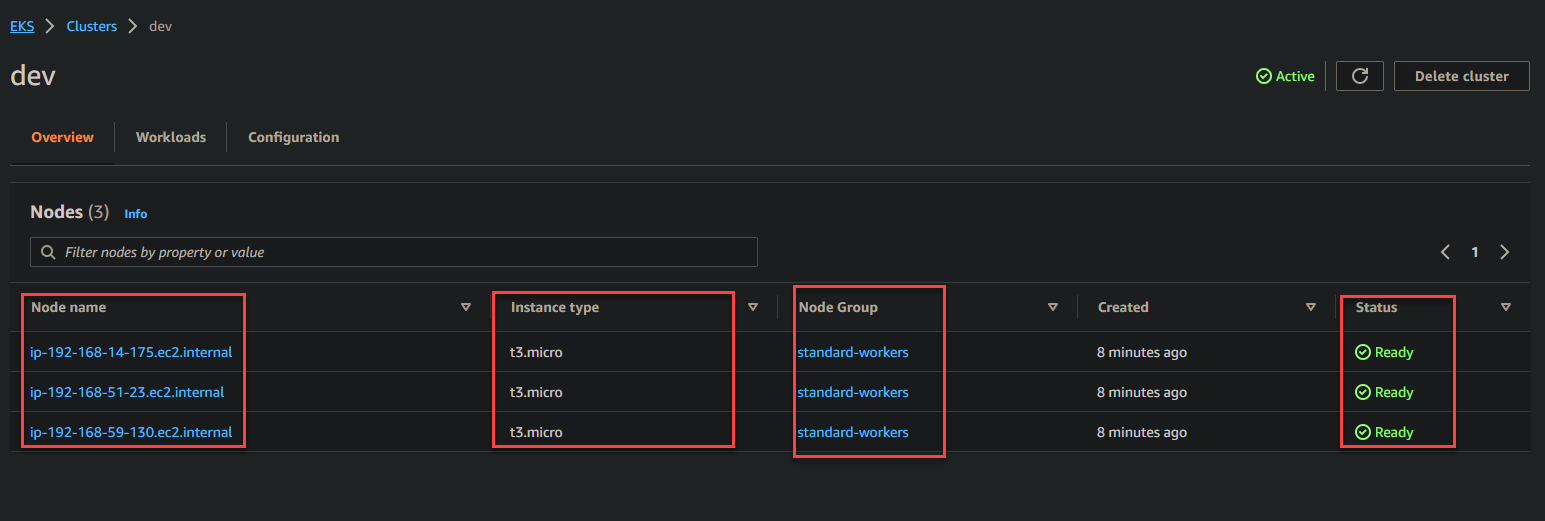
As you can see below, an **eksctl-dev-cluster** CloudFormation stack is being created. This process might take 15 minutes or more to complete.

Previewing the eksctl-dev-cluster stack.

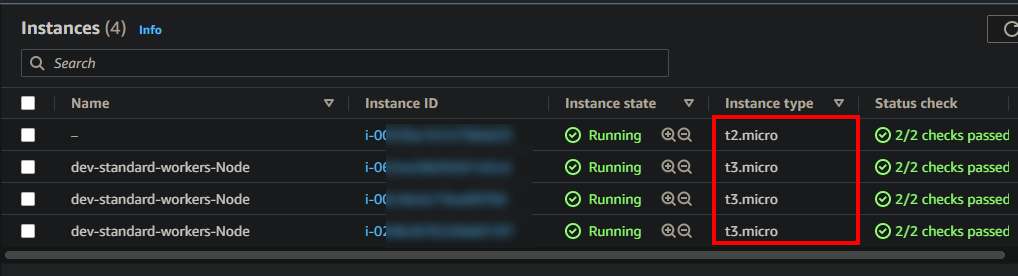
3. Now, navigate to your EKS dashboard, and you’ll see a cluster named **dev** provisioned. Click on the **dev** hyperlink to access **dev’s** EKS Cluster dashboard.

Navigating to the dev EKS Cluster dashboard.

Below, you can see the **dev’s** EKS cluster details, like **Node name**, **Instance type**, **Node Group**, and **Status**.

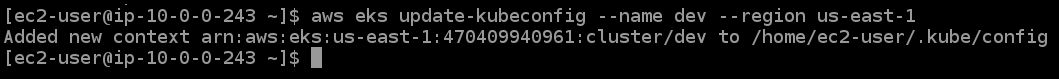
Previewing the dev EKS Cluster dashboard.

4. Switch to your EC2 dashboard, and you’ll see four nodes are running, with three having the **t3.micro** role in your AWS account (three worker nodes and one master node).

Previewing the EC2 dashboard.

5. Finally, run the command below to update your kubectl config (**update-kubeconfig**) with your cluster endpoint, certificate, and credentials.

aws eks update-kubeconfig --name sai --region ap-south-1

Updating kubectl config

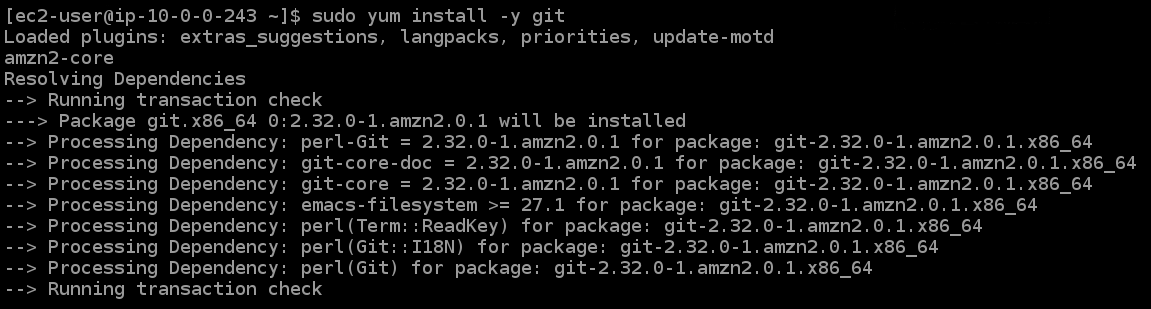
**Deploying an Application on EKS Cluster**

You’ve created your EKS cluster and confirmed it’s perfectly running. But right now, the EKS cluster is just sitting in the corner. For this demo, you’ll make use of the EKS cluster by deploying an [NGINX](https://adamtheautomator.com/nginx-test-config/) application.

**Related:**[How to Test Your NGINX Configuration Before Screwing it Up](https://adamtheautomator.com/nginx-test-config/)

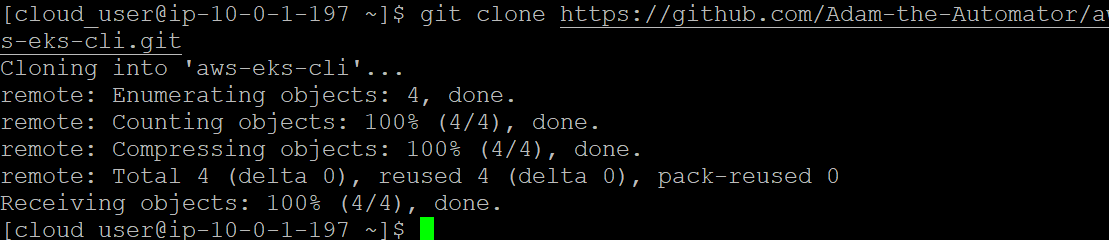
1. Run the **yum** command below to **install** **git** while accepting all prompts automatically (**-y**) during installation.

sudo yum install -y git

Installing Git

2. Next, run the **git clone** command below to clone the configuration files from the GitHub repository to your current directory. You will use these files to create an NGINX deployment on your pods and create a [load balancer (ELB)](https://aws.amazon.com/elasticloadbalancing/).

git clone https://github.com/Adam-the-Automator/aws-eks-cli.git

Cloning the configuration files

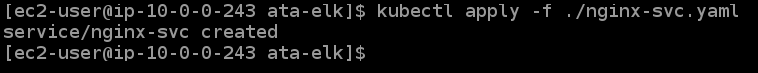
3. Run the following commands to move into the **ata-elk** directory, and create (**kubectl apply**) a service for NGINX (**./nginx-svc.yaml**).

# Change directory to ata-elk

cd ata-elk

# Apply the configuration in ./nginx-svc.yaml to a pod

kubectl apply -f ./nginx-svc.yaml

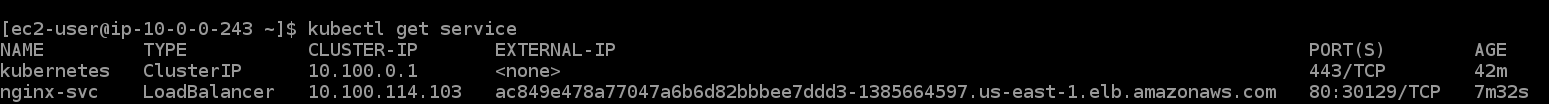
Creating a service for NGINX

4. Next, run the **kubectl get service** to check the status of your NGINX service.

kubectl get service

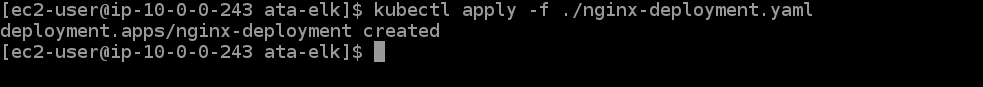
As you see below, the service type is a load balancer, and Kubernetes created a service (**nginx-svc**), which is your NGINX deployment. You can also see the external DNS hostname of the load balancer created by EKS under the **EXTERNAL IP** column.

Note down the external DNS hostname of the load balancer as you will need it later to test the load balancer.

Checking the status of your NGINX

5. Run the **kubectl** command below to deploy the NGINX pods.

kubectl apply -f ./nginx-deployment.yaml

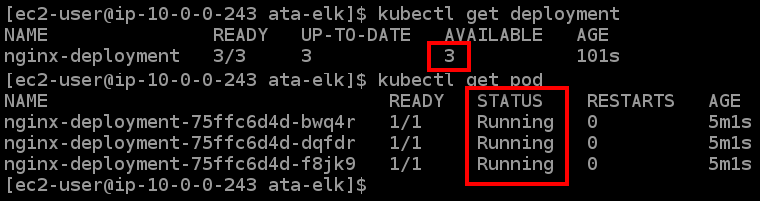
Deploying the NGINX pods

6. Run the following **kubectl get** commands to check the status of your NGINX **deployment** and your NGINX **pod**.

kubectl get deployment

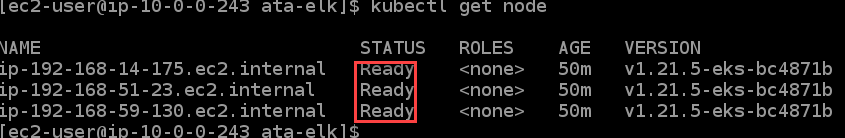
kubectl get pod

As you can see in the output below, there are three pods of your deployment, and all are running.

Checking the status of the NGINX deployment and pods

7. Next, run the **kubectl get node** command to check the status of your worker nodes.

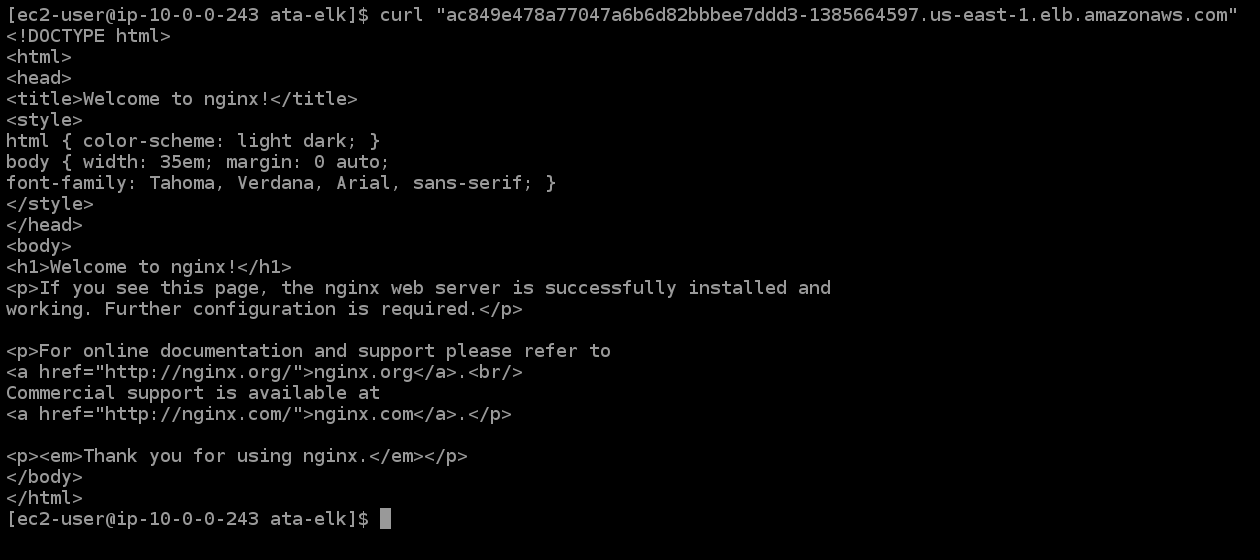
kubectl get node

Check the status of your worker nodes

8. Now, run the **curl** command below to test your load balancer. Replace **<LOAD\_BALANCER\_DNS\_HOSTNAME>** with the DNS name you previously noted (step five).

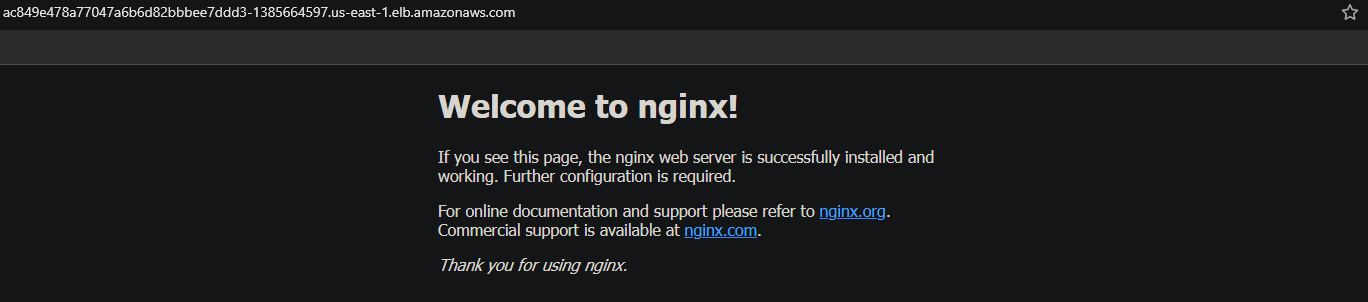
curl "<LOAD\_BALANCER\_DNS\_HOSTNAME>"

You will see the NGINX welcome page from the NGINX service created by EKS, as shown below. The below output confirms that your load balancer is working correctly and that you can access your NGINX pods.

Checking your load balancer

9. Finally, for double-checking, copy and paste the DNS name of the load balancer into a new browser tab.

You will also get a welcome page from NGINX, which indicates your application is working.

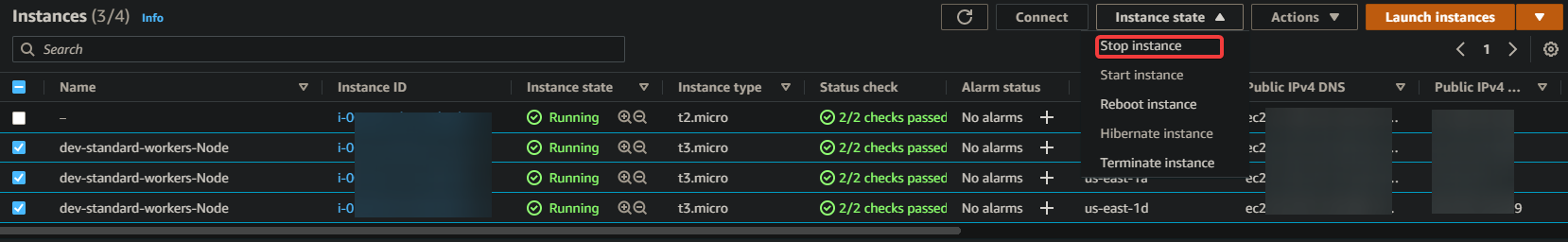
Checking your load balancer with a browser

**Testing the Highly-available Kubernetes Control**

Now that you have a cluster running, you’ll test if the Kubernetes control plane is highly available. Your application’s uptime depends on this feature. If the control plane does not work, your applications will be down and cannot serve users.

With the highly-available Kubernetes control feature, you increase the availability of your application. You’ll test this feature by stopping your EKS worker nodes and see if Kubernetes brings up new nodes to replace the failed ones.

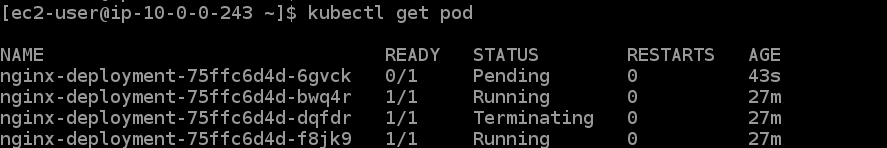
1. In your EC2 dashboard, stop all of your EKS worker node instances, as shown below.

Stopping all of your EKS worker node instances

2. Next, run the following command to check the status of the worker node.

kubectl get node

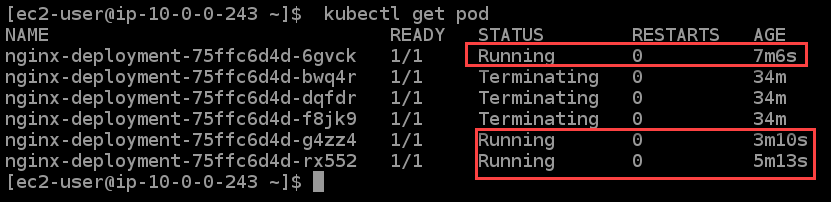
You will get a mix of statuses, like **Pending**, **Running**, and **Terminating**. Why? Because as you try to stop all the worker nodes, Kubernetes detects the failure and quickly brings up another node.

Checking the status of the worker node

3. Now run the **kubectl get pod** command to test the highly-available Kubernetes control feature.

kubectl get pod

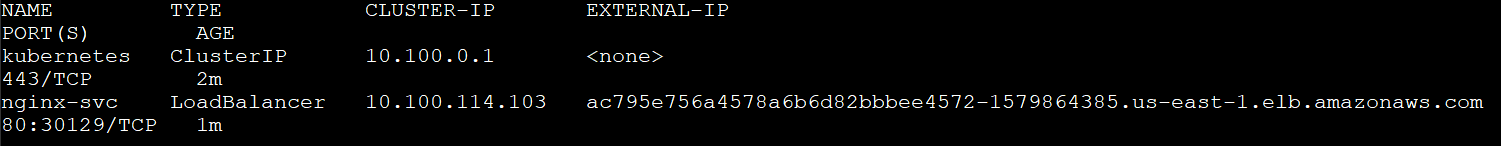
You can see in the output that there are three new pods (identified by age) in the **Running** state. These new pods indicate the highly-available Kubernetes control feature is working as intended.

Checking the status of the pods

4. Run the **kubectl get service** command below to list all available services.

You can see below that Kubernetes has created a new service, and the DNS name of the load balancer is different now. **kubectl get service**

kubectl get service

Kubernetes has created a new service

5. Finally, copy and paste the DNS name of the load balancer into your browser. You will get the welcome page from NGINX as you did in the last step of the “Deploying an Application on EKS Cluster” section.

**Conclusion**

Throughout this tutorial, you’ve learned how to create an EKS cluster, deploy an NGINX service from your container, and test the highly-available control plane functionality.

At this point, you should have a good understanding of how to create EKS clusters in your AWS environment.