

Setup AWS EKS Cluster With eksctl Tool

Introduction

Elastic Kubernetes Service (EKS) is a fully managed Kubernetes service from AWS. In this lab, you will work with the AWS command line interface (**aws-cli**) and console, using command line utilities like **eksctl** to launch an EKS cluster and **kubect**l to provision a Kubernetes deployment and pod running instances of nginx, and create a LoadBalancer service to expose your application over the internet.

We need to install first the **aws-cli, **eksctl** & **kubect**l tools on a server**

aws-cli: The AWS Command Line Interface (AWS CLI) is a unified tool to manage your AWS services. With just one tool to download and configure, you can control multiple AWS services from the command line and automate them through scripts.

eksctl: is a simple CLI tool for creating and managing clusters on EKS - Amazon's managed Kubernetes service for EC2. It is written in Go, uses CloudFormation.

Kubectl: allows you to run commands against Kubernetes clusters. You can use kubectl to deploy applications, inspect and manage cluster resources, and view logs etc..

To setup the above tools we may use

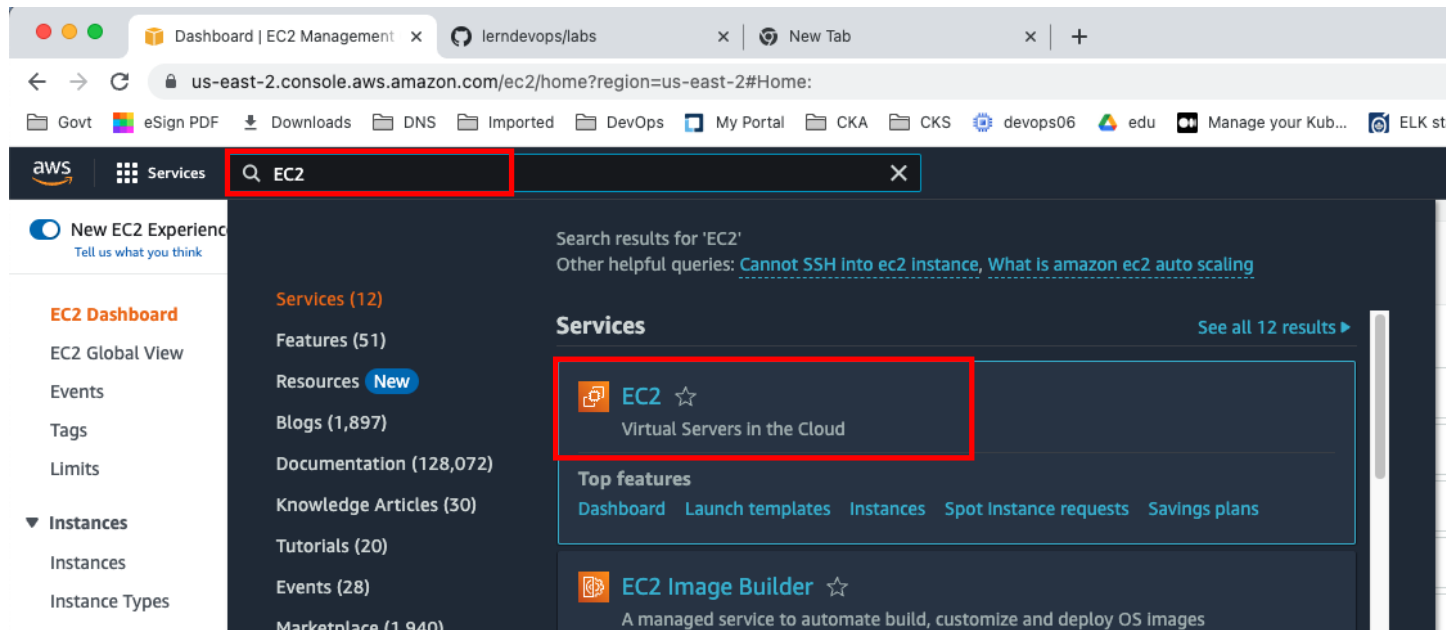
- **Use AWS Cloud Shell**
 - here you will see both aws cli & kubectl already installed
 - you will need to install eksctl tool
- **Create a EC2 VM & install**
- **install on your own laptops/desktops(windows or mac)**

go to: <https://console.aws.amazon.com/>

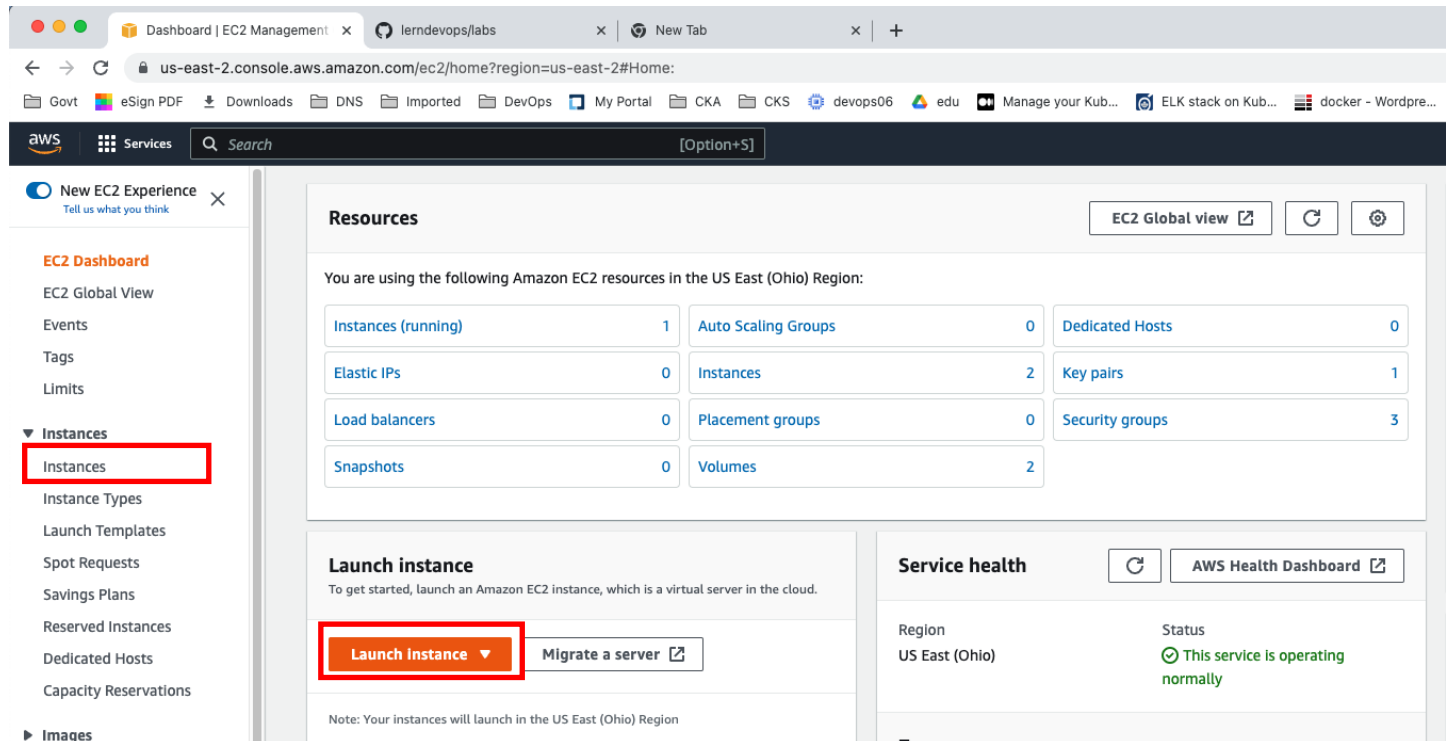
Login to your AWS account

Launch an EC2 Instance to Configure the Command Line Tools

Navigate to EC2 > Instances.

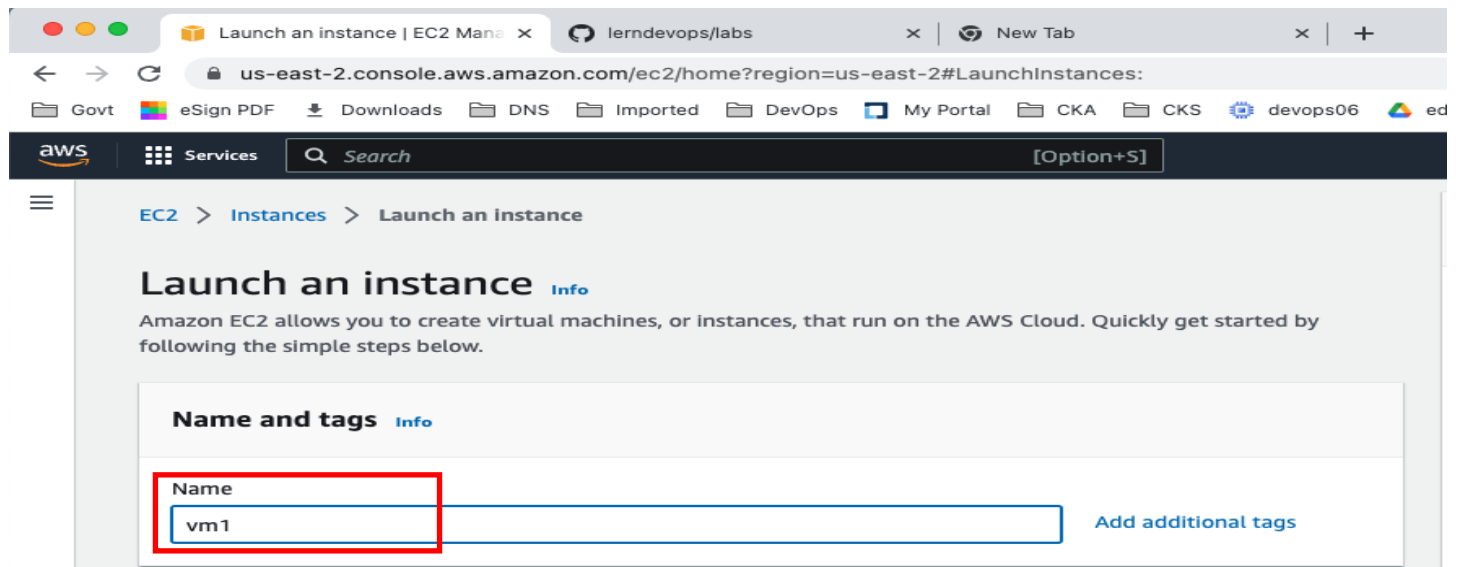


Click Launch Instance.



On the Launch an instance Page

Enter Name and tags: any value then scroll down



Launch an instance | EC2 Manager | lerndevops/labs | New Tab

us-east-2.console.aws.amazon.com/ec2/home?region=us-east-2#LaunchInstances:

Govt eSign PDF Downloads DNS Imported DevOps My Portal CKA CKS devops06 ed

aws Services Search [Option+S]

EC2 > Instances > Launch an instance

Launch an instance [Info](#)

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

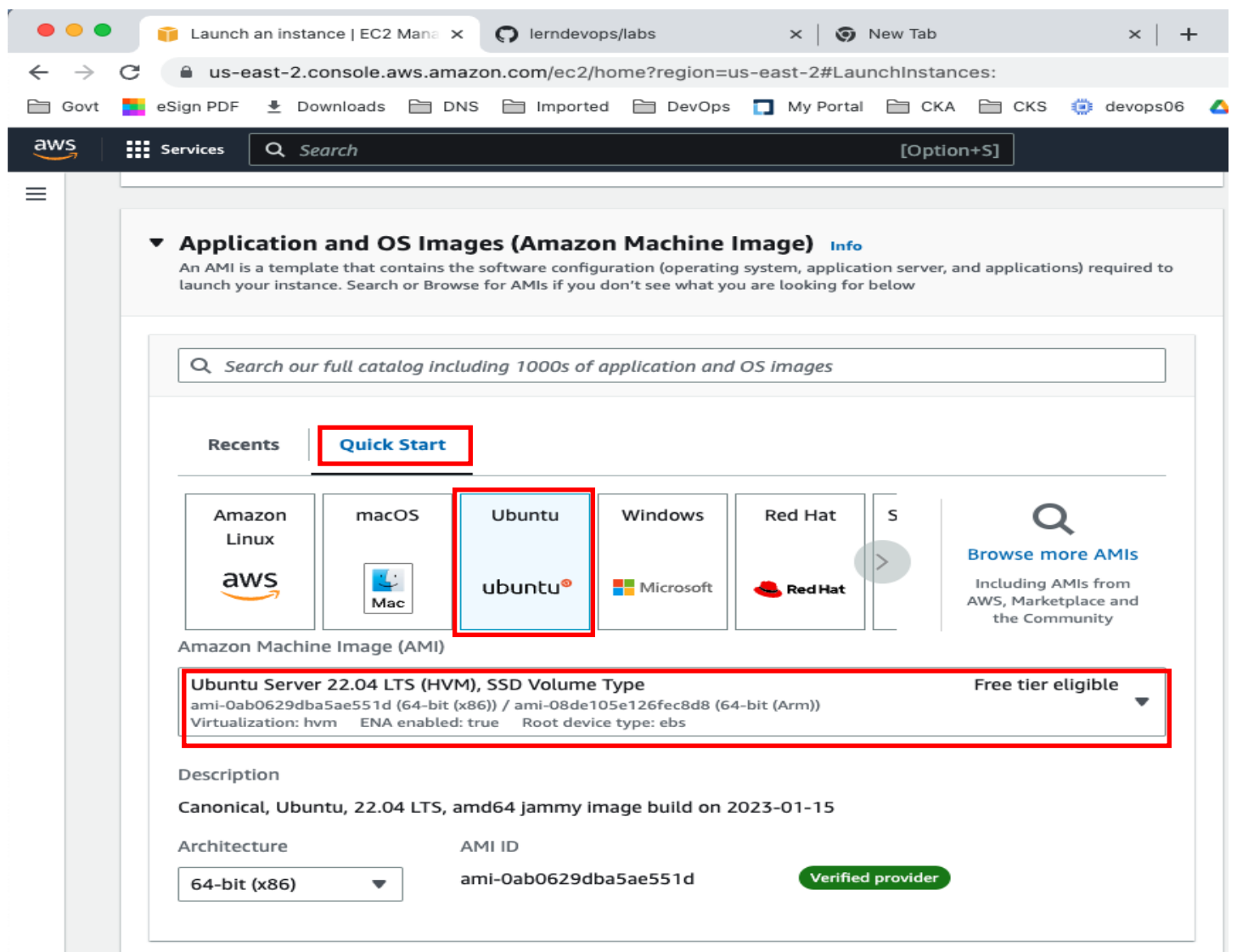
Name and tags [Info](#)

Name

vm1

Add additional tags

Choose an OS Image & Scroll down



Launch an instance | EC2 Manager | lerndevops/labs | New Tab

us-east-2.console.aws.amazon.com/ec2/home?region=us-east-2#LaunchInstances:

Govt eSign PDF Downloads DNS Imported DevOps My Portal CKA CKS devops06 ed

aws Services Search [Option+S]

EC2 > Instances > Launch an instance

Application and OS Images (Amazon Machine Image) [Info](#)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Search our full catalog including 1000s of application and OS images

Recents

Quick Start

Amazon Linux

aws

macOS

Mac

Ubuntu

ubuntu

Windows

Microsoft

Red Hat

Red Hat

S

Browse more AMIs

Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

Ubuntu Server 22.04 LTS (HVM), SSD Volume Type

Free tier eligible

ami-0ab0629dba5ae551d (64-bit (x86)) / ami-08de105e126fec8d8 (64-bit (Arm))

Virtualization: hvm ENA enabled: true Root device type: ebs

Description

Canonical, Ubuntu, 22.04 LTS, amd64 jammy image build on 2023-01-15

Architecture

64-bit (x86)

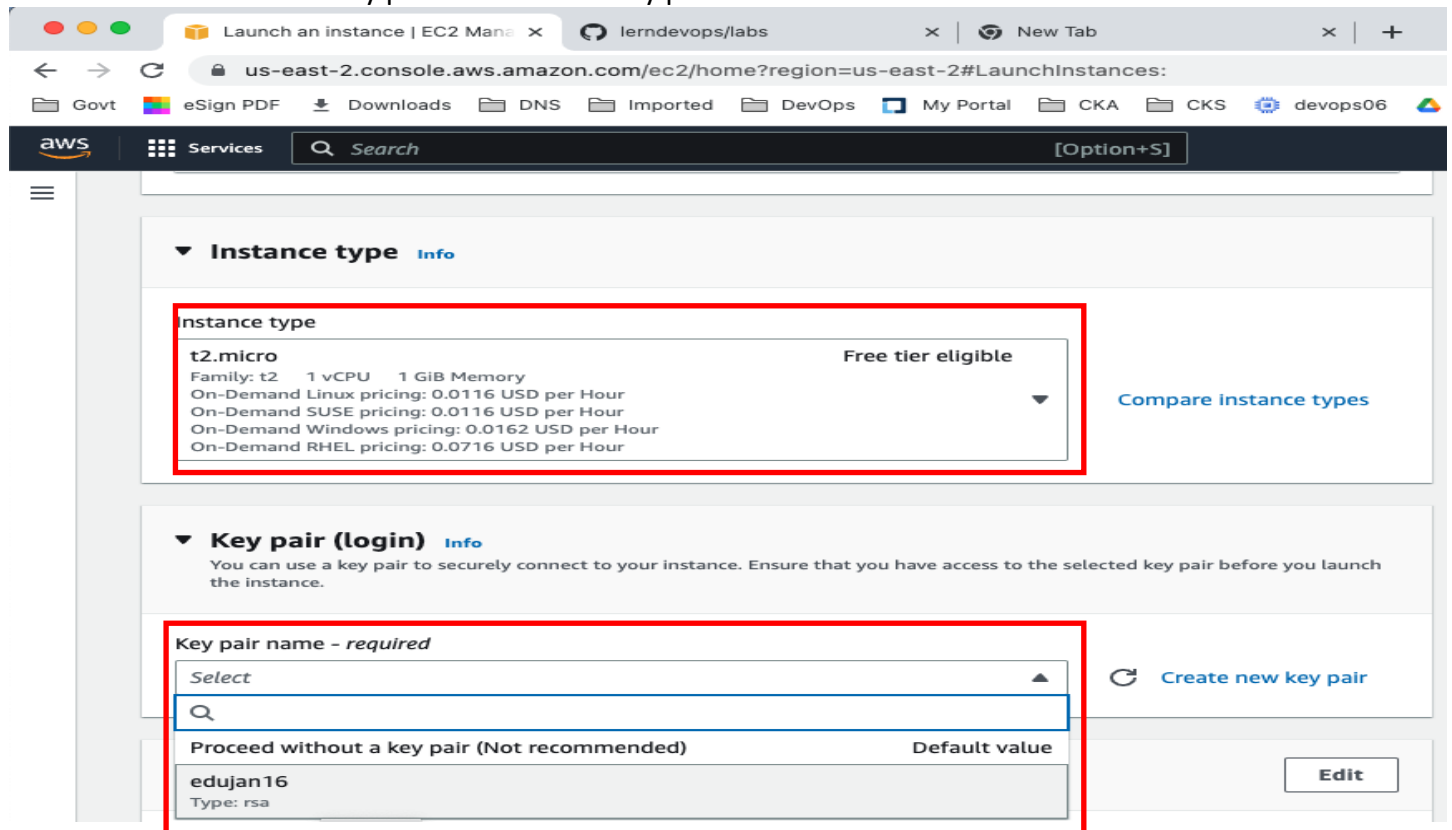
AMI ID

ami-0ab0629dba5ae551d

Verified provider

Choose Instance type & Key pair

Note: ensure to create a key pair if there is no key pair found



Instance type [Info](#)

Instance type

t2.micro **Free tier eligible**

Family: t2 1 vCPU 1 GiB Memory

On-Demand Linux pricing: 0.0116 USD per Hour

On-Demand SUSE pricing: 0.0116 USD per Hour

On-Demand Windows pricing: 0.0162 USD per Hour

On-Demand RHEL pricing: 0.0716 USD per Hour

[Compare instance types](#)

Key pair (login) [Info](#)

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - required

Select

[Create new key pair](#)

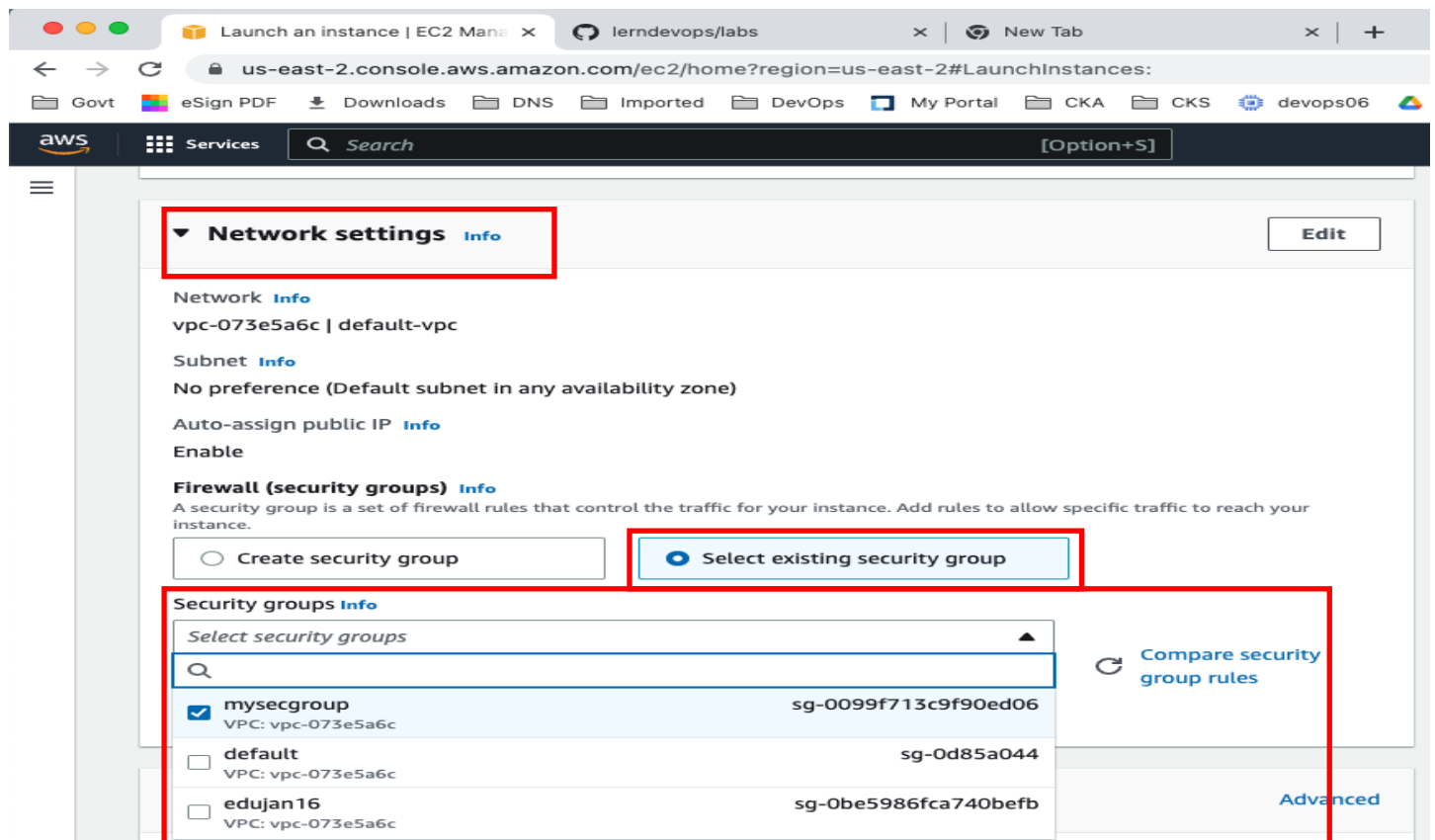
Proceed without a key pair (Not recommended) **Default value**

edujan16

Type: rsa

[Edit](#)

Network Settings: Choose an **existing Security Group** or **Create one** & Select as needed & Scroll Down



Network settings [Info](#) [Edit](#)

Network [Info](#)

vpc-073e5a6c | default-vpc

Subnet [Info](#)

No preference (Default subnet in any availability zone)

Auto-assign public IP [Info](#)

Enable

Firewall (security groups) [Info](#)

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

☐ Create security group ☒ **Select existing security group**

Security groups [Info](#)

Select security groups

☒ **mysecgroup** VPC: vpc-073e5a6c sg-0099f713c9f90ed06

☐ **default** VPC: vpc-073e5a6c sg-0d85a044

☐ **edujan16** VPC: vpc-073e5a6c sg-0be5986fca740befb

[Compare security group rules](#)

[Advanced](#)

Configure Storage & Advanced details

YOU DO NOT NEED TO CHANGE ANYTHING on Below Sections – leave the defaults values as they are.

► Configure storage Info	Advanced
► Advanced details Info	

Review the Summary & Click on Launch Instance

▼ Summary

Number of instances [Info](#)

Software Image (AMI)
Canonical, Ubuntu, 22.04 LTS, ...[read more](#)
ami-0ab0629dba5ae551d

Virtual server type (instance type)
t2.micro

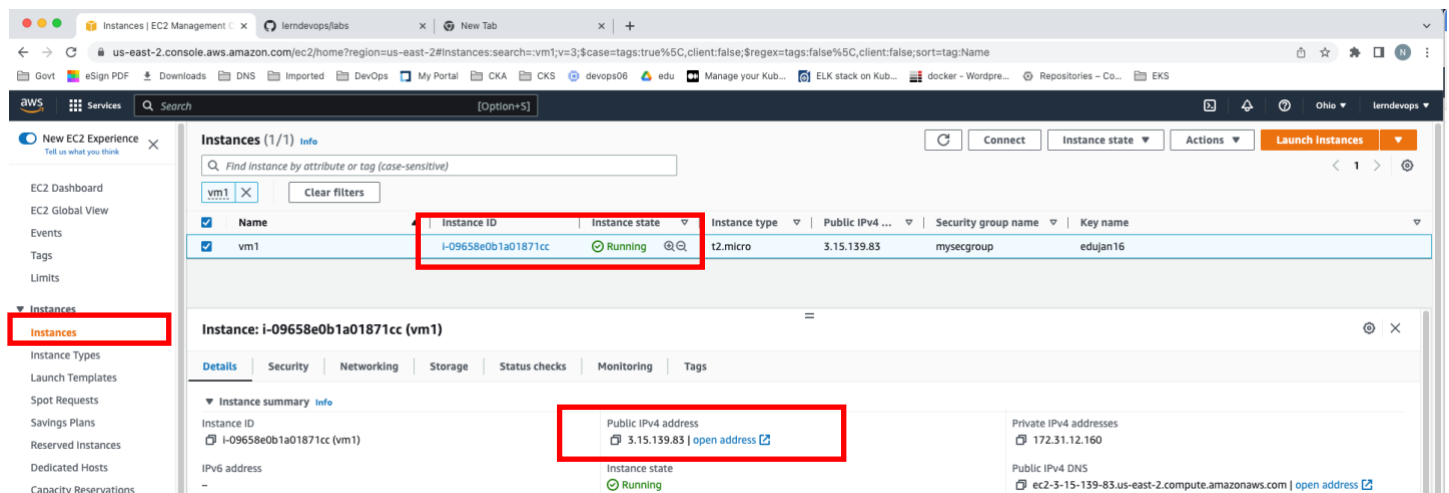
Firewall (security group)
mysecgroup

Storage (volumes)
1 volume(s) - 8 GiB

Cancel

Launch instance

Then Go Back to the Instances Page -- You Should See the VM Running in couple of min



Login to VM/EC2 Created – Follow the below to Login

There are many ways to login to remote linux servers, some recommended options as below

Using AWS SSM: (common for windows or mac users)

<https://github.com/lerndevops/labs/blob/master/cloud/aws/connect-to-aws-ec2-using-aws-ssm.pdf>

MAC Users:

<https://github.com/lerndevops/labs/blob/master/cloud/aws/connect-to-EC2-with-MAC-terminal.pdf>

Windows Users:

<https://github.com/lerndevops/labs/blob/master/cloud/aws/connect-to-EC2-with-mobaXterm.pdf>

Once Logged In

Follow the installation Instruction from below to install the tools required

install AWS CLI:

<https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html>

install kubectl CLI:

<https://kubernetes.io/docs/tasks/tools/>

install eksctl CLI:

<https://docs.aws.amazon.com/eks/latest/userguide/eksctl.html>

Validate versions after install:

- aws --version
- kubectl version --short --client
- eksctl version

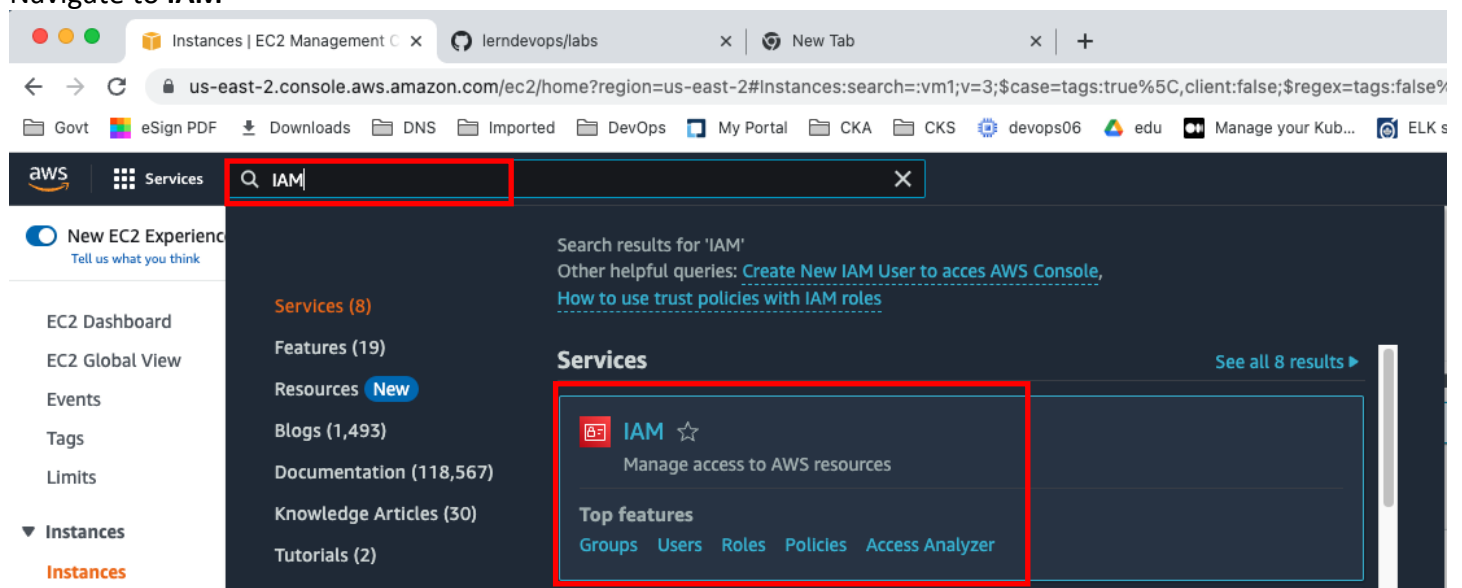
```
nareshwar@mbpro ~ %  
nareshwar@mbpro ~ %  
nareshwar@mbpro ~ % aws --version  
aws-cli/2.9.19 Python/3.11.1 Darwin/22.3.0 source/arm64 prompt/off  
nareshwar@mbpro ~ %  
nareshwar@mbpro ~ % kubectl version --short --client  
Flag --short has been deprecated, and will be removed in the future. The --short output will become the default.  
Client Version: v1.26.1  
Kustomize Version: v4.5.7  
nareshwar@mbpro ~ %  
nareshwar@mbpro ~ % eksctl version  
0.127.0  
nareshwar@mbpro ~ %  
nareshwar@mbpro ~ %
```

Now To Authenticate to your AWS Account using aws command line tool we need to

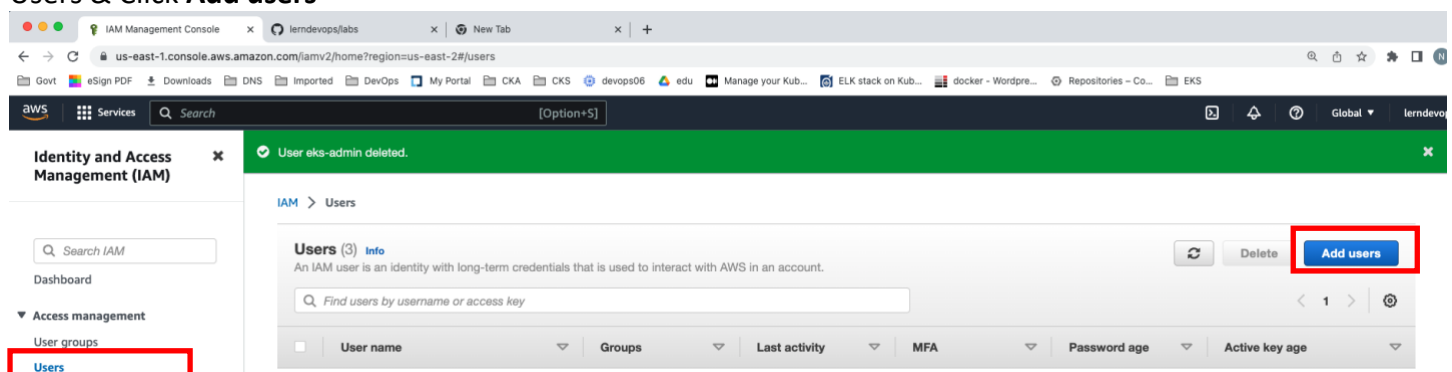
Create an IAM User with **Admin Permissions**

Note: you can fine grain your access level to IAM user accordingly

Navigate to IAM



Users & Click Add users



Specify user details & Click Next

Step 1
Specify user details

Step 2
Set permissions

Step 3
Review and create

Specify user details

User details

User name

☐ Enable console access - optional
Enables a password that allows users to sign in to the AWS Management Console.

For programmatic access, you can generate access keys after you create the user. [Learn more](#)

Cancel **Next**

Set Permissions → Select Attach policies directly → Select AdministratorAccess → Click Next

Step 1
Specify user details

Step 2
Set permissions

Step 3
Review and create

Set permissions

Add user to an existing group or create a new one. Using groups is a best-practice way to manage user's permissions by job functions. [Learn more](#)

Permissions options

☐ Add user to group
Add user to an existing group, or create a new group. We recommend using groups to manage user permissions by job function.

☐ Copy permissions
Copy all group memberships, attached managed policies, and inline policies from an existing user.

☒ **Attach policies directly**
Attach a managed policy directly to a user. As a best practice, we recommend attaching policies to a group instead. Then, add the user to the appropriate group.

Permissions policies (1/1040)
Choose one or more policies to attach to your new user.

☒ **AdministratorAccess**
AWS managed - job function 3

Policy name	Type	Attached entities
AccessAnalyzerServiceRolePolicy	AWS managed	0
AdministratorAccess	AWS managed - job function	3
AdministratorAccess-Amplify	AWS managed	0

Review and Create → Click Create User

Step 1
Specify user details

Step 2
Set permissions

Step 3
Review and create

Review and create

Review your choices. After you create the user, you can view and download the autogenerated password, if enabled.

User details

User name	Console password type	Require password reset
eks-admin	None	No

Permissions summary

Name	Type	Used as
AdministratorAccess	AWS managed - job function	Permissions policy

Tags - optional
Tags are key-value pairs you can add to AWS resources to help identify, organize, or search for resources. Choose any tags you want to associate with this user.

No tags associated with the resource.

[Add new tag](#)

Cancel Previous **Create user**

Once the User is created Click on the User created

Identity and Access Management (IAM)

Search IAM

Dashboard

Access management

User groups

Users

Roles

Policies

User created successfully
You can view and download the user's password and email instructions for signing in to the AWS Management Console.

IAM > Users

Users (4) Info
An IAM user is an identity with long-term credentials that is used to interact with AWS in an account.

Search: eks- 1 match

	User name	Groups	Last activity	MFA
<input type="checkbox"/>	eks-admin	None	Never	None

On the user page → Click on **Security Credential** Tab on user page

Identity and Access Management (IAM)

Search IAM

Dashboard

Access management

User groups

Users

Roles

Policies

Identity providers

Account settings

Access reports

Access analyzer

Archive rules

Analysts

Settings

IAM > Users > eks-admin

eks-admin Delete

Summary

ARN am:aws:iam:682530473276:user/eks-admin	Console access Disabled	Access key 1 Not enabled
Created February 02, 2023, 08:34 (UTC+05:30)	Last console sign-in -	Access key 2 Not enabled

Permissions Groups Tags **Security credentials** Access Advisor

Console sign-in Enable console access

Console sign-in link https://682530473276.signin.aws.amazon.com/console	Console password Not enabled
--	---------------------------------

Scroll down to “Access Keys” Section & Click on “Create access key”

Identity and Access Management (IAM)

Search IAM

Dashboard

Access management

User groups

Access keys (0)
Use access keys to send programmatic calls to AWS from the AWS CLI, AWS Tools for PowerShell, AWS SDKs, or direct AWS API calls. You can have a maximum of two access keys (active or inactive) at a time. [Learn more](#)

Create access key

No access keys
As a best practice, avoid using long-term credentials like access keys. Instead, use tools which provide short term credentials. [Learn more](#)

Create access key

Access Key best Practices & alternatives → Choose “**Command Line Interface (CLI)**” option & Click **Next**

Step 1

Access key best practices & alternatives

Avoid using long-term credentials like access keys to improve your security. Consider the following use cases and alternatives.

☒ **Command Line Interface (CLI)**
You plan to use this access key to enable the AWS CLI to access your AWS account.

☐ **Local code**
You plan to use this access key to enable application code in a local development environment to access your AWS account.

☐ **Application running on an AWS compute service**
You plan to use this access key to enable application code running on an AWS compute service like Amazon EC2, Amazon ECS, or AWS Lambda to access your AWS account.

☐ **Third-party service**
You plan to use this access key to enable access for a third-party application or service that monitors or manages your AWS resources.

☐ **Application running outside AWS**
You plan to use this access key to enable an application running on an on-premises host, or to use a local AWS client or third-party AWS plugin.

☐ **Other**
Your use case is not listed here.

Alternatives recommended

- Use [AWS CloudShell](#), a browser-based CLI, to run commands. [Learn more](#)
- Use the [AWS CLI V2](#) and enable authentication through a user in IAM Identity Center. [Learn more](#)

☒ I understand the above recommendation and want to proceed to create an access key.

Cancel **Next**

Enter a “**Description tag value**” & Click on **Create access key**

IAM Management Console x | lerndevops/labs x | New Tab x | +

us-east-1.console.aws.amazon.com/iamv2/home?region=us-east-2#/users/details/eks-admin/create-access-key

Govt eSign PDF Downloads DNS Imported DevOps My Portal CKA CKS devops06 edu Manage your Kub... ELK stack on Kub... docker - Wordpre... Repositories - Co... EKS

aws Services Search [Option+S]

IAM > Users > eks-admin > Create access key

Step 1
[Access key best practices & alternatives](#)

Step 2 - optional
Set description tag

Step 3
Retrieve access keys

Set description tag - optional

The description for this access key will be attached to this user as a tag and shown alongside the access key.

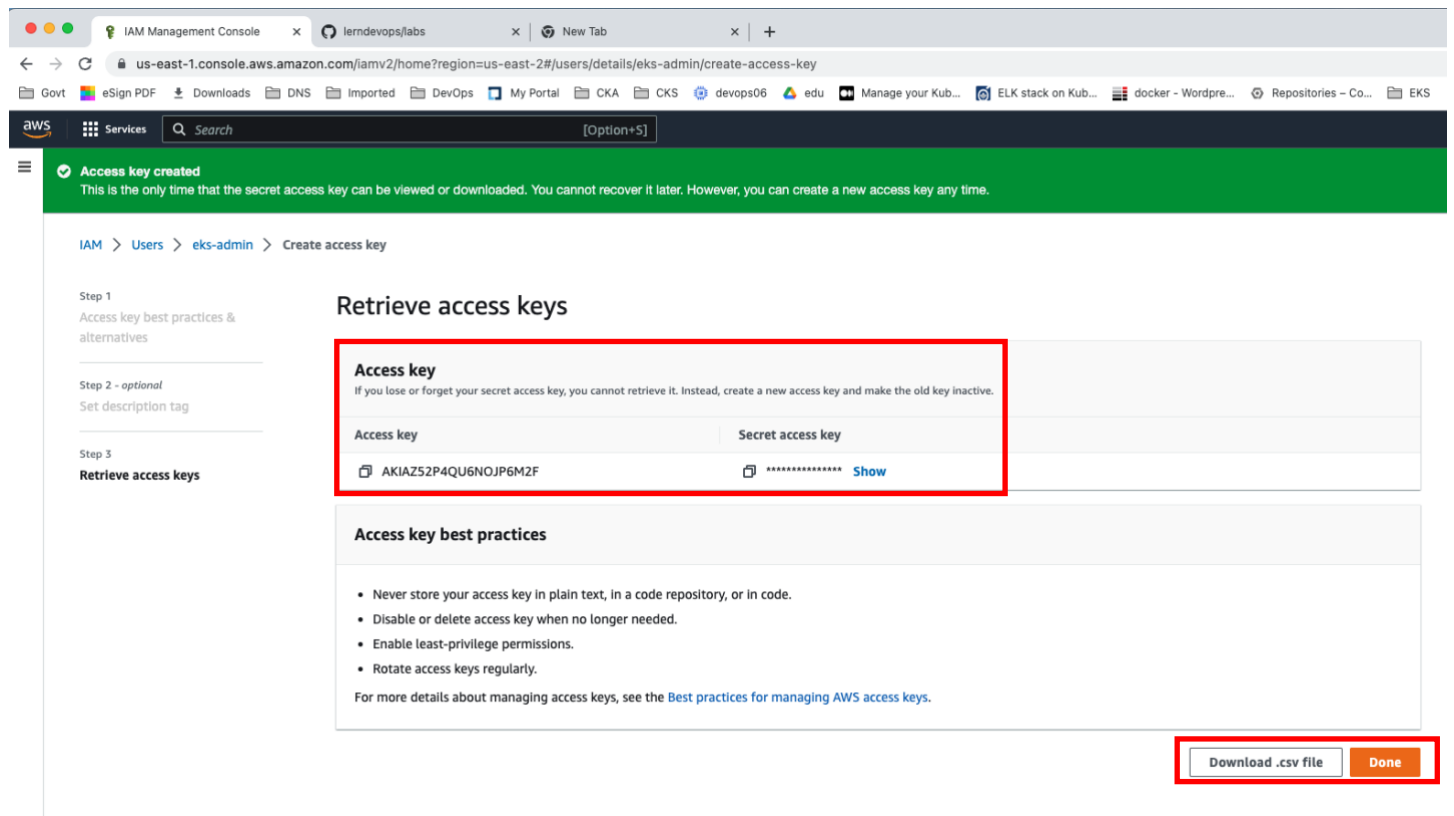
Description tag value
Describe the purpose of this access key and where it will be used. A good description will help you rotate this access key confidently later.

eks-admin

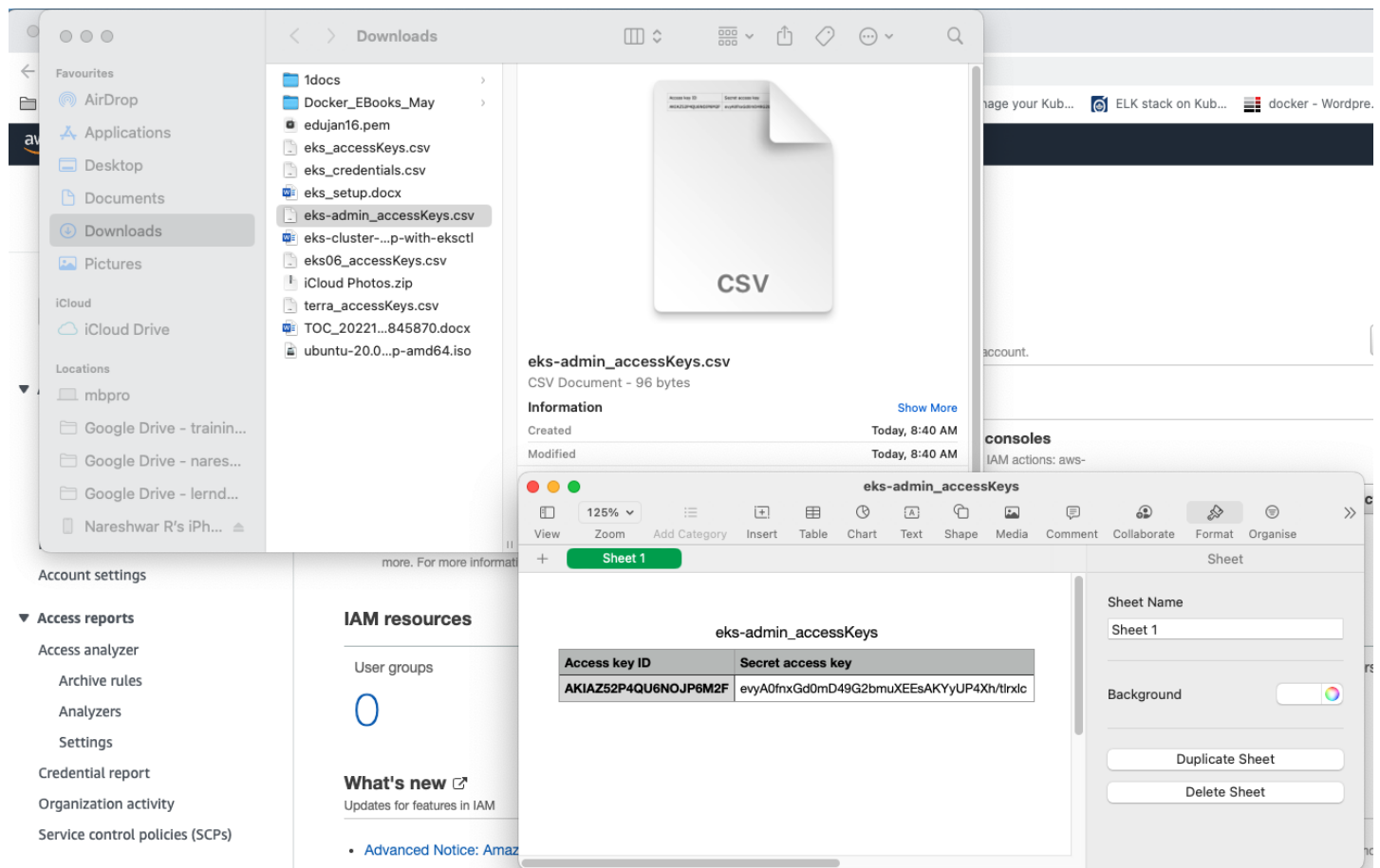
Maximum 256 characters. Allowed characters are letters, numbers, spaces representable in UTF-8, and: _ . : / = + - @

Cancel Previous **Create access key**

Save the “Access Key” & “Secret Access Key” generated on the page **OR**
Click on “Download .csv file”
then click on Done



Ensure the keys are downloaded or saved on your computer successfully

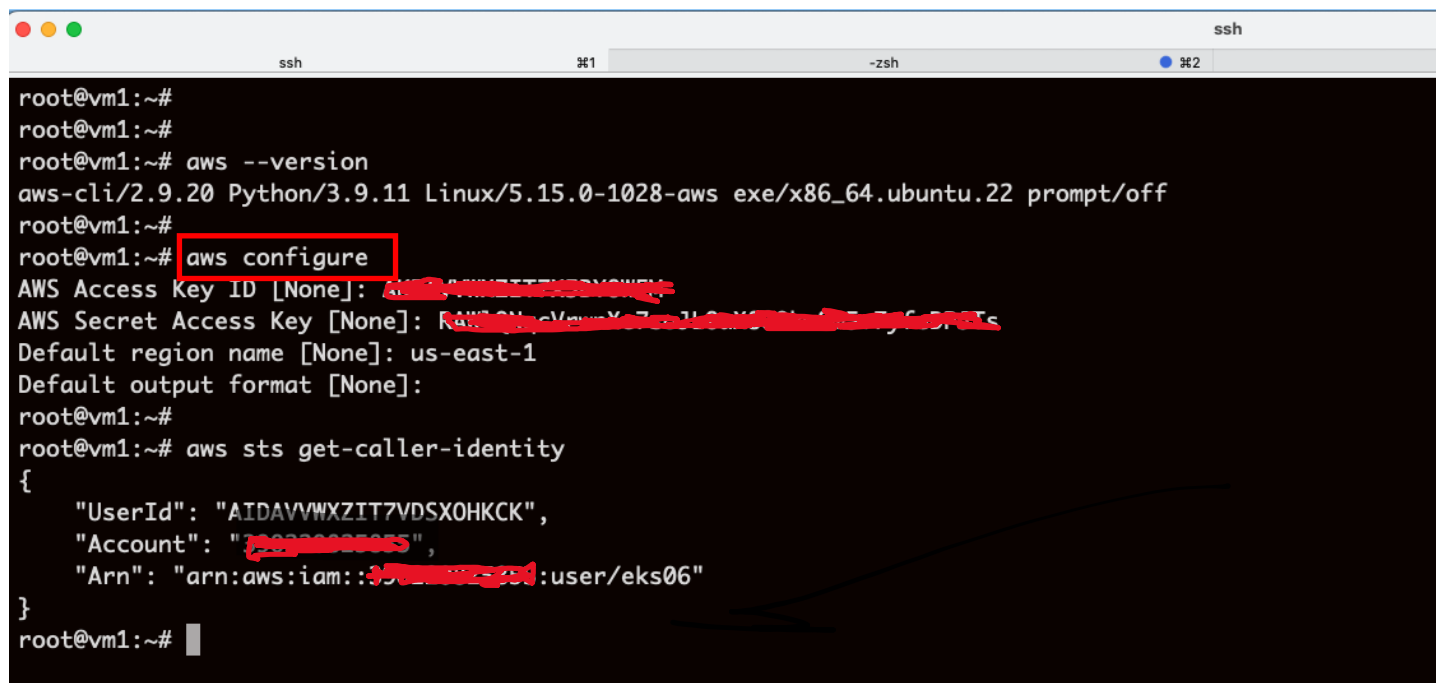


Now Let's Configure the keys created above for AWS CLI tool

Login to the server/vm where the aws cli tool is installed & run

aws configure

→ it will prompt for keys accordingly as below enter the values & validate



```
root@vm1:~#  
root@vm1:~#  
root@vm1:~# aws --version  
aws-cli/2.9.20 Python/3.9.11 Linux/5.15.0-1028-aws exe/x86_64.ubuntu.22 prompt/off  
root@vm1:~#  
root@vm1:~# aws configure  
AWS Access Key ID [None]: AKIAVWVXZIT7VDSXOHHKCK  
AWS Secret Access Key [None]: AKIAVWVXZIT7VDSXOHHKCK  
Default region name [None]: us-east-1  
Default output format [None]:  
root@vm1:~#  
root@vm1:~# aws sts get-caller-identity  
{  
  "UserId": "AIDAVVWXZIT7VDSXOHHKCK",  
  "Account": "123456789012",  
  "Arn": "arn:aws:iam::123456789012:user/eks06"  
}  
root@vm1:~#
```

Let's Create a EKS Cluster with eksctl now

By default eksctl command line tool uses aws Cloud Formation template to create EKS cluster with default configurations as below

Ex: with command **"eksctl create cluster"**

A cluster will be created with default parameters:

- exciting auto-generated name, e.g., fabulous-mushroom-1527688624
- two m5.large worker nodes—this instance type suits most common use-cases, and is good value for money
- use the official AWS [EKS AMI](#)
- us-west-2 region
- a dedicated VPC (check your quotas)
- configures kubeconfig for kubectl tool to access the cluster

```
root@vm1:~# eksctl create cluster
2023-02-02 04:36:29 [i] eksctl version 0.127.0
2023-02-02 04:36:29 [i] using region us-east-2
2023-02-02 04:36:29 [i] setting availability zones to [us-east-2b us-east-2c us-east-2a]
2023-02-02 04:36:29 [i] subnets for us-east-2b - public:192.168.0.0/19 private:192.168.96.0/19
2023-02-02 04:36:29 [i] subnets for us-east-2c - public:192.168.32.0/19 private:192.168.128.0/19
2023-02-02 04:36:29 [i] subnets for us-east-2a - public:192.168.64.0/19 private:192.168.160.0/19
2023-02-02 04:36:29 [i] nodegroup "ng-be340c9e" will use "" [AmazonLinux2/1.24]
2023-02-02 04:36:29 [i] using Kubernetes version 1.24
2023-02-02 04:36:29 [i] creating EKS cluster "attractive-party-1675312589" in "us-east-2" region with managed nodes
2023-02-02 04:36:29 [i] will create 2 separate CloudFormation stacks for cluster itself and the initial managed nodegroup
2 sequential tasks: { create cluster control plane "attractive-party-1675312589",
2 sequential sub-tasks: {
    wait for control plane to become ready,
    create managed nodegroup "ng-be340c9e",
}
}
2023-02-02 04:47:30 [i] waiting for CloudFormation stack "eksctl-attractive-party-1675312589-cluster"
2023-02-02 04:48:30 [i] waiting for CloudFormation stack "eksctl-attractive-party-1675312589-cluster"
2023-02-02 04:50:30 [i] building managed nodegroup stack "eksctl-attractive-party-1675312589-nodegroup-ng-be340c9e"
2023-02-02 04:50:31 [i] deploying stack "eksctl-attractive-party-1675312589-nodegroup-ng-be340c9e"
2023-02-02 04:50:31 [i] waiting for CloudFormation stack "eksctl-attractive-party-1675312589-nodegroup-ng-be340c9e"
2023-02-02 04:51:01 [i] waiting for CloudFormation stack "eksctl-attractive-party-1675312589-nodegroup-ng-be340c9e"
2023-02-02 04:51:40 [i] waiting for CloudFormation stack "eksctl-attractive-party-1675312589-nodegroup-ng-be340c9e"
2023-02-02 04:52:51 [i] waiting for CloudFormation stack "eksctl-attractive-party-1675312589-nodegroup-ng-be340c9e"
2023-02-02 04:54:09 [i] waiting for CloudFormation stack "eksctl-attractive-party-1675312589-nodegroup-ng-be340c9e"
2023-02-02 04:54:09 [i] waiting for the control plane to become ready
2023-02-02 04:54:09 [i] saved kubeconfig as "/root/.kube/config"
2023-02-02 04:54:09 [i] no tasks
2023-02-02 04:54:09 [i] all EKS cluster resources for "attractive-party-1675312589" have been created
2023-02-02 04:54:09 [i] nodegroup "ng-be340c9e" has 2 node(s)
2023-02-02 04:54:09 [i] node "ip-192-168-34-170.us-east-2.compute.internal" is ready
2023-02-02 04:54:09 [i] node "ip-192-168-93-19.us-east-2.compute.internal" is ready
2023-02-02 04:54:09 [i] waiting for at least 2 node(s) to become ready in "ng-be340c9e"
2023-02-02 04:54:09 [i] nodegroup "ng-be340c9e" has 2 node(s)
2023-02-02 04:54:09 [i] node "ip-192-168-34-170.us-east-2.compute.internal" is ready
2023-02-02 04:54:09 [i] node "ip-192-168-93-19.us-east-2.compute.internal" is ready
2023-02-02 04:54:11 [i] kubectl command should work with "/root/.kube/config", try 'kubectl get nodes'
2023-02-02 04:54:11 [i] EKS cluster "attractive-party-1675312589" in "us-east-2" region is ready
root@vm1:~#

root@vm1:~# eksctl get clusters
NAME          REGION    EKSCTL CREATED
attractive-party-1675312589  us-east-2  True
```

Validate the cluster

```
root@vm1:~# kubectl get nodes -o wide
NAME                                STATUS    ROLES    AGE    VERSION    INTERNAL-IP    EXTERNAL-IP    OS-IMAGE    KERNEL-VERSION    CONTAINER-RUNTIME
ip-192-168-34-170.us-east-2.compute.internal Ready    <none>    2m58s    v1.24.9-eks-49d8fe8    192.168.34.170    18.118.148.199    Amazon Linux 2    5.4.228-131.415.amzn2.x86_64    containerd://1.6.6
ip-192-168-93-19.us-east-2.compute.internal Ready    <none>    2m59s    v1.24.9-eks-49d8fe8    192.168.93.19    3.17.173.40    Amazon Linux 2    5.4.228-131.415.amzn2.x86_64    containerd://1.6.6

root@vm1:~# kubectl get pods --all-namespaces -o wide
NAMESPACE    NAME                READY    STATUS    RESTARTS    AGE    IP                NODE                                NOMINATED NODE    READINESS GATES
kube-system  aws-node-mshq4      1/1      Running    0            3m17s    192.168.93.19    ip-192-168-93-19.us-east-2.compute.internal    <none>            <none>
kube-system  aws-node-r9mmz      1/1      Running    0            3m16s    192.168.34.170    ip-192-168-34-170.us-east-2.compute.internal    <none>            <none>
kube-system  coredns-5c5677bc78-f8lqw    1/1      Running    0            11m     192.168.69.15    ip-192-168-93-19.us-east-2.compute.internal    <none>            <none>
kube-system  coredns-5c5677bc78-g4nhh    1/1      Running    0            11m     192.168.70.252    ip-192-168-93-19.us-east-2.compute.internal    <none>            <none>
kube-system  kube-proxy-2428j      1/1      Running    0            3m16s    192.168.34.170    ip-192-168-34-170.us-east-2.compute.internal    <none>            <none>
kube-system  kube-proxy-84mtn      1/1      Running    0            3m17s    192.168.93.19    ip-192-168-93-19.us-east-2.compute.internal    <none>            <none>
```

We can customize the all the configurations as required using cli options or using yaml as below

Ex: cli command

eksctl create cluster --name myekscluster --node-type t2.micro --nodegroup-name ng1

Like above we can put all parameters as required in cli but no so comfortable

instead we can write a simple yaml config file as below

vi cluster.yaml

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
metadata:
  name: basic-cluster
  region: eu-north-1
nodeGroups:
  - name: ng-1
    instanceType: m5.large
    desiredCapacity: 10
  - name: ng-2
    instanceType: m5.xlarge
    desiredCapacity: 2
```

then apply file as – **eksctl create cluster -f cluster.yaml**

we can also do a Dry Run to generate the cluster config yaml with default parameters used

root@vm1:~# **eksctl create cluster --dry-run**

```
apiVersion: eksctl.io/v1alpha5
availabilityZones:
  - us-east-2b
  - us-east-2c
  - us-east-2a
cloudWatch:
  clusterLogging: {}
iam:
  vpcResourceControllerPolicy: true
  withOIDC: false
kind: ClusterConfig
kubernetesNetworkConfig:
  ipFamily: IPv4
managedNodeGroups:
  - amiFamily: AmazonLinux2
    desiredCapacity: 2
    disableIMDSv1: false
    disablePodIMDS: false
  iam:
    withAddonPolicies:
      albIngress: false
      appMesh: false
      appMeshPreview: false
```

autoScaler: false
awsLoadBalancerController: false
certManager: false
cloudWatch: false
ebs: false
efs: false
externalDNS: false
fsx: false
imageBuilder: false
xRay: false
instanceSelector: {}
labels:
 alpha.eksctl.io/cluster-name: exciting-gopher-1675314821
 alpha.eksctl.io/nodegroup-name: ng-aa9510ed
maxSize: 2
minSize: 2
name: ng-aa9510ed
privateNetworking: false
releaseVersion: ""
securityGroups:
 withLocal: null
 withShared: null
ssh:
 allow: false
 publicKeyPath: ""
tags:
 alpha.eksctl.io/nodegroup-name: ng-aa9510ed
 alpha.eksctl.io/nodegroup-type: managed
volumeIOPS: 3000
volumeSize: 80
volumeThroughput: 125
volumeType: gp3
metadata:
 name: exciting-gopher-1675314821
 region: us-east-2
 version: "1.24"
privateCluster:
 enabled: false
 skipEndpointCreation: false
vpc:
 autoAllocateIPv6: false
 cidr: 192.168.0.0/16
 clusterEndpoints:
 privateAccess: false
 publicAccess: true
 manageSharedNodeSecurityGroupRules: true
nat:
 gateway: Single

We can also delete the cluster as below

eksctl delete cluster --name <cluster-name>

```
root@vm1:~# eksctl get clusters
NAME                REGION    EKSCTL CREATED
attractive-party-1675312589  us-east-2  True
root@vm1:~#
root@vm1:~# eksctl delete cluster --name attractive-party-1675312589
2023-02-02 05:16:15 [i] deleting EKS cluster "attractive-party-1675312589"
2023-02-02 05:16:15 [i] will drain 0 unmanaged nodegroup(s) in cluster "attractive-party-1675312589"
2023-02-02 05:16:15 [i] starting parallel draining, max in-flight of 1
2023-02-02 05:16:15 [i] deleted 0 Fargate profile(s)
2023-02-02 05:16:16 [✓] kubeconfig has been updated
2023-02-02 05:16:16 [i] cleaning up AWS load balancers created by Kubernetes objects of Kind Service or Ingress
2023-02-02 05:16:16 [i]
2 sequential tasks: { delete nodegroup "ng-be340c9e", delete cluster control plane "attractive-party-1675312589" [async]
}
2023-02-02 05:16:16 [i] will delete stack "eksctl-attractive-party-1675312589-nodegroup-ng-be340c9e"
2023-02-02 05:16:16 [i] waiting for stack "eksctl-attractive-party-1675312589-nodegroup-ng-be340c9e" to get deleted
2023-02-02 05:16:16 [i] waiting for CloudFormation stack "eksctl-attractive-party-1675312589-nodegroup-ng-be340c9e"
2023-02-02 05:16:46 [i] waiting for CloudFormation stack "eksctl-attractive-party-1675312589-nodegroup-ng-be340c9e"
2023-02-02 05:17:17 [i] waiting for CloudFormation stack "eksctl-attractive-party-1675312589-nodegroup-ng-be340c9e"
2023-02-02 05:19:00 [i] waiting for CloudFormation stack "eksctl-attractive-party-1675312589-nodegroup-ng-be340c9e"
2023-02-02 05:19:46 [i] waiting for CloudFormation stack "eksctl-attractive-party-1675312589-nodegroup-ng-be340c9e"
2023-02-02 05:21:23 [i] waiting for CloudFormation stack "eksctl-attractive-party-1675312589-nodegroup-ng-be340c9e"
2023-02-02 05:22:59 [i] waiting for CloudFormation stack "eksctl-attractive-party-1675312589-nodegroup-ng-be340c9e"
2023-02-02 05:22:59 [i] will delete stack "eksctl-attractive-party-1675312589-cluster"
2023-02-02 05:22:59 [✓] all cluster resources were deleted
root@vm1:~#
```