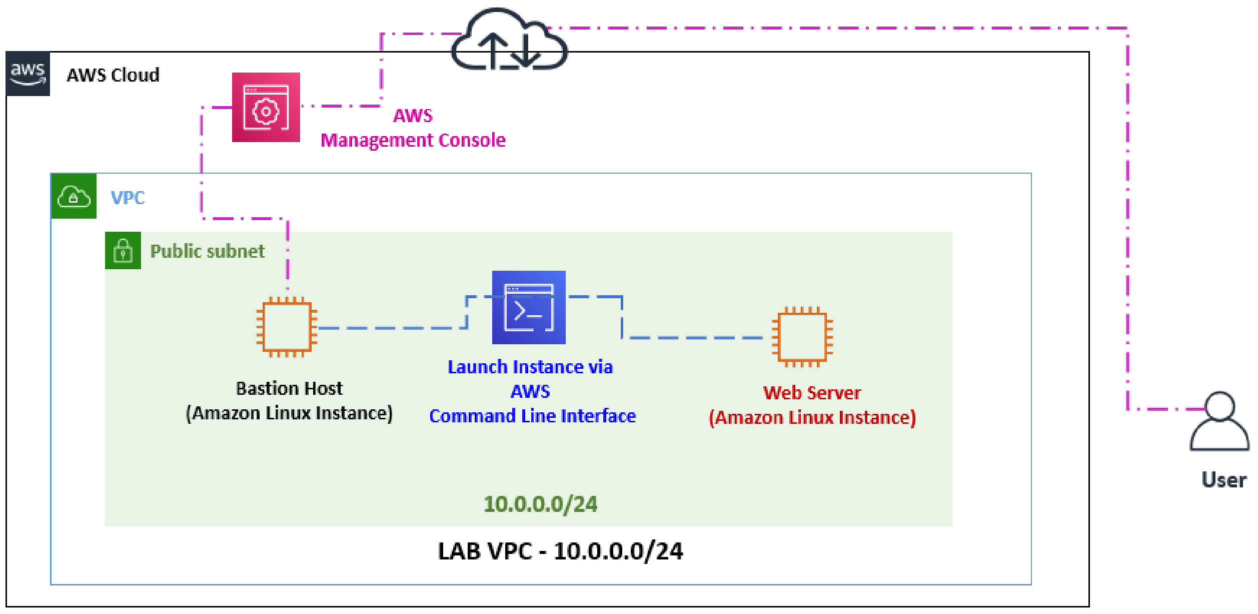
Creating Amazon EC2 Instances (Linux)

(LAB-01)

Traditional methods of deploying servers and configuring security are complex and often involve multiple teams and long delays. Fortunately, it is quick and easy to deploy secure infrastructure in the cloud. As a Systems Operator, you can automate many of these processes using the AWS Command-Line Interface.

In this lab you will:

- Launch an Amazon EC2 instance using the management console.
- Launch an Amazon EC2 instance using the AWS Command-Line Interface (AWS CLI).


AWS Cloud
AWS Management Console
VPC
Public subnet
Bastion Host (Amazon Linux Instance)
Launch Instance via AWS Command Line Interface
Web Server (Amazon Linux Instance)
10.0.0.0/24
LAB VPC - 10.0.0.0/24
User

Task 1: Create VPC

Amazon Virtual Private Cloud (Amazon VPC) lets you provision a logically isolated section of the AWS Cloud where you can launch AWS resources in a virtual network that you define.

Note: We will discuss in details about **VPC** in upcoming module.

Step 1: Choose a Region Using the Console

1. Choose the **US East (N. Virginia)** region list to the right of your account information on the navigation bar.

Step 2: Deploy a VPC

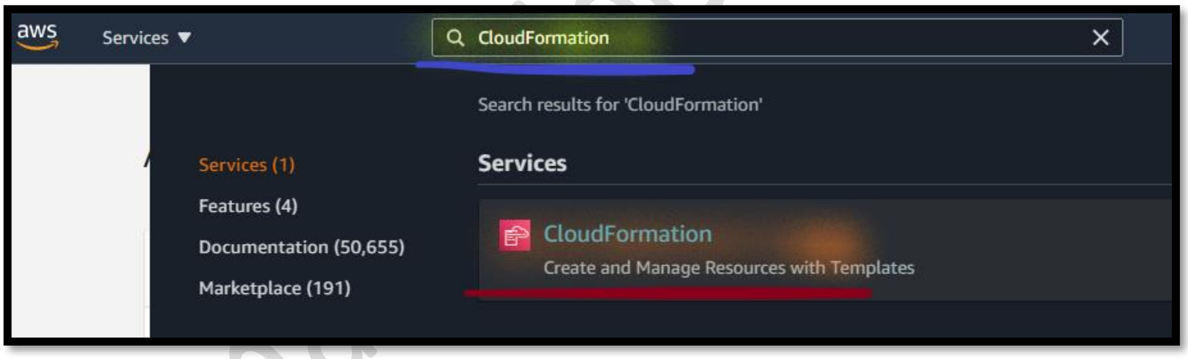
AWS CloudFormation model and provision AWS resources in your cloud environment in automated way.

Note

• In this task, CloudFormation provision Virtual Network, where you will deploy the resources.

Note: We will discuss in details about **Cloud Formation** in upcoming module.

2. In the **AWS Management Console**, on the **Services** menu Search and Select **CloudFormation**.



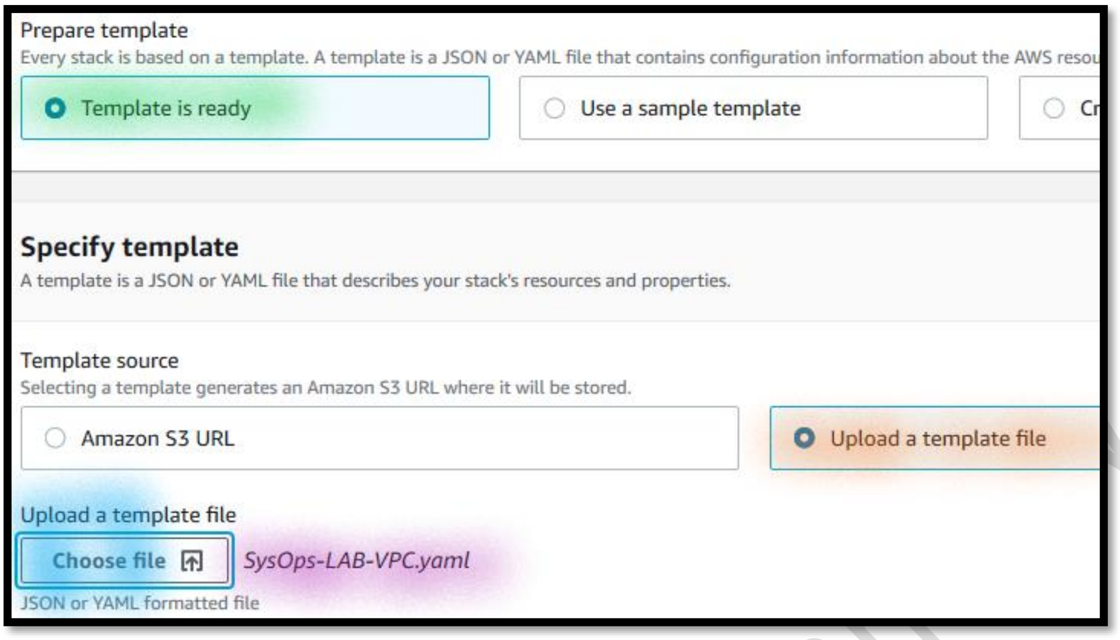
3. Click **Create stack** and **configure**:

   a. **Prepare template**: Select **Template is ready**.

   b. **Template source**: Select **Upload a template file**.

   c. **Choose file**: Click on **choose file** then select the **AWS-LAB-01-VPC**.yaml file.

Note: **AWS-LAB-01-VPC**.yaml template is provided with the lab manual.



Prepare template
Every stack is based on a template. A template is a JSON or YAML file that contains configuration information about the AWS resou

[✓ Template is ready] Use a sample template Cr

Specify template
A template is a JSON or YAML file that describes your stack's resources and properties.

Template source
Selecting a template generates an Amazon S3 URL where it will be stored.

Amazon S3 URL [✓ Upload a template file]

Upload a template file
Choose file SysOps-LAB-VPC.yaml
JSON or YAML formatted file

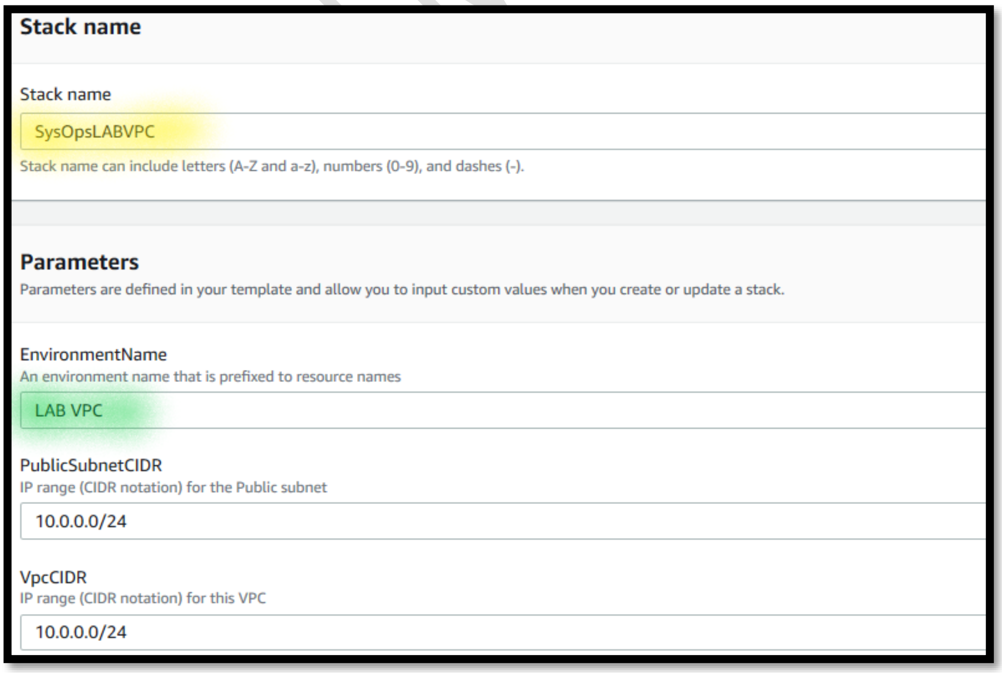
d. Click **Next**.

4. In the **Specify stack details** page:

a. **Stack name**: Write **AWSLABVPC01**.

b. **Environment name**: Write **LAB VPC-01**.

Note: Leave other details as default.



Stack name

Stack name
SysOpsLABVPC
Stack name can include letters (A-Z and a-z), numbers (0-9), and dashes (-).

Parameters
Parameters are defined in your template and allow you to input custom values when you create or update a stack.

EnvironmentName
An environment name that is prefixed to resource names
LAB VPC

PublicSubnetCIDR
IP range (CIDR notation) for the Public subnet
10.0.0.0/24

VpcCIDR
IP range (CIDR notation) for this VPC
10.0.0.0/24

c. Click **Next**.

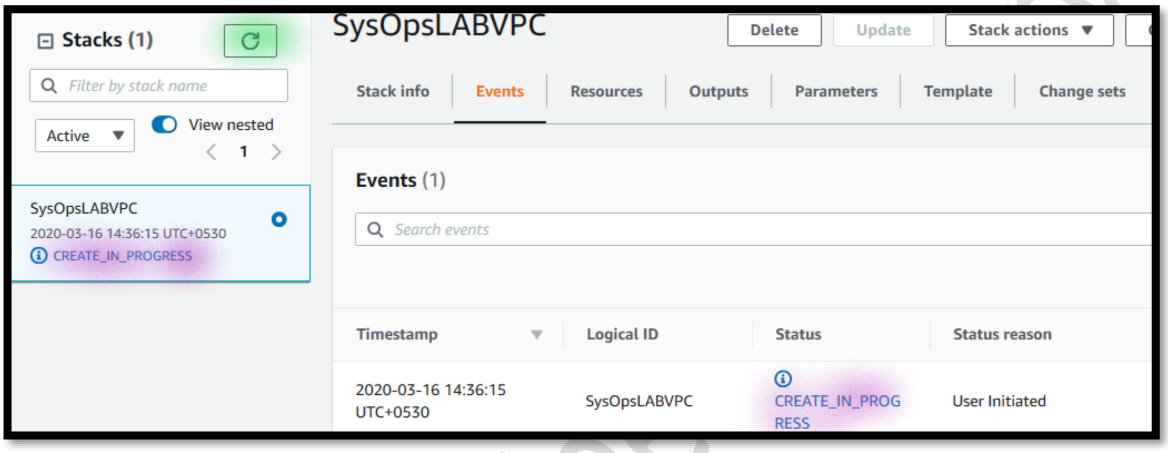
Note: Leave other details as default.

d. Click **Next**.

5. In the **Review** page:

a. Click **Create stack**.

b. The stack status is showing as **CREATE_IN_PROGRESS.**

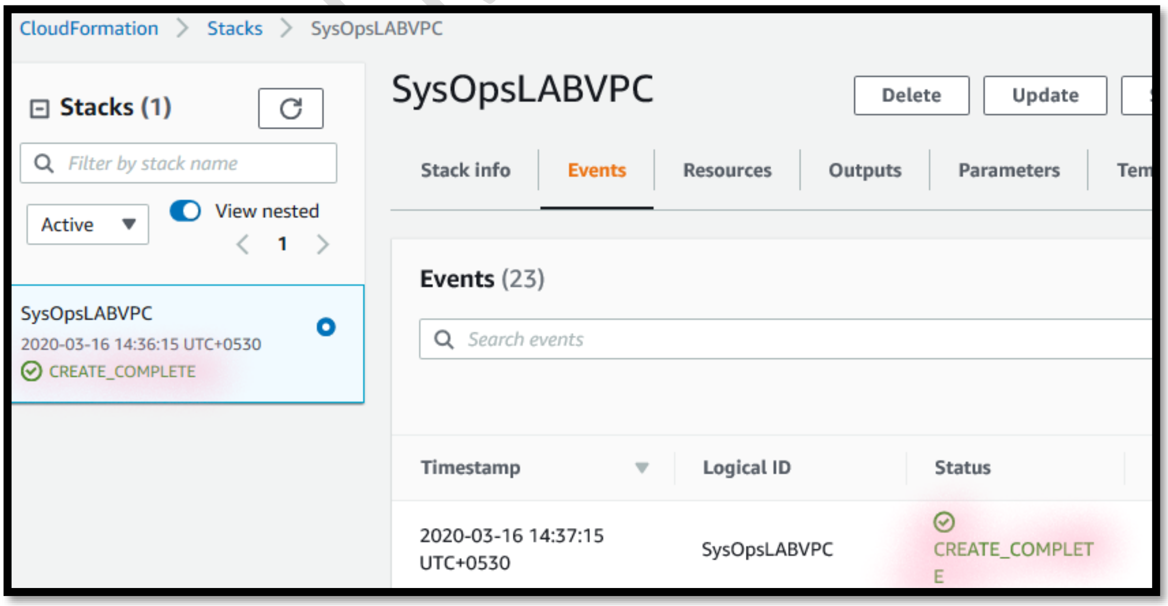


SysOpsLABVPC
2020-03-16 14:36:15 UTC+0530
CREATE_IN_PROGRESS

Events (1)

Timestamp	Logical ID	Status	Status reason
2020-03-16 14:36:15 UTC+0530	SysOpsLABVPC	CREATE_IN_PROGRESS	User Initiated

Note: Click on **Refresh** until status is turned into **CREATE_COMPLETE**.



SysOpsLABVPC
2020-03-16 14:36:15 UTC+0530
CREATE_COMPLETE

Events (23)

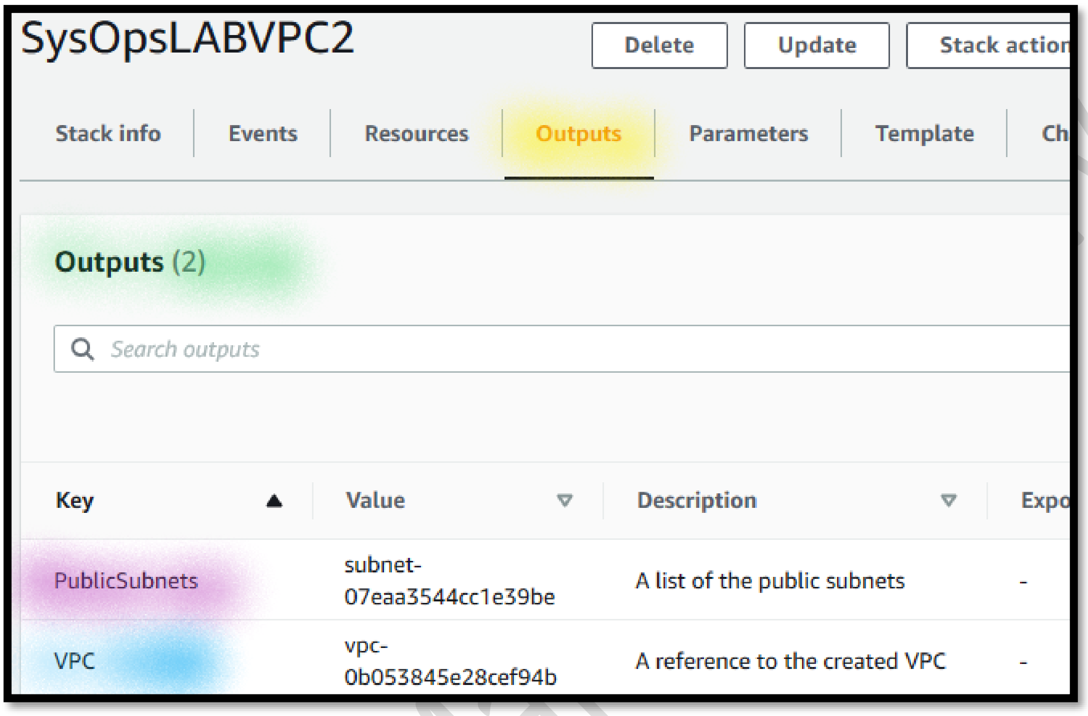
Timestamp	Logical ID	Status
2020-03-16 14:37:15 UTC+0530	SysOpsLABVPC	CREATE_COMPLETE

c. Click the **Outputs** tab.

d. A ***CloudFormation stack*** can provide **output information**, such as resources details. ***You will see two outputs***:

i. **PublicSubnets**: The Id of the Public Subnet that was created.

ii. **VPC**: The Id of the VPC that was created.



Key	Value	Description	Expo
PublicSubnets	subnet-07eaa3544cc1e39be	A list of the public subnets	-
VPC	vpc-0b053845e28cef94b	A reference to the created VPC	-

Task 2: Create IAM Role

You can use roles to delegate access to users, applications, or services that don't normally have access to your AWS resources.

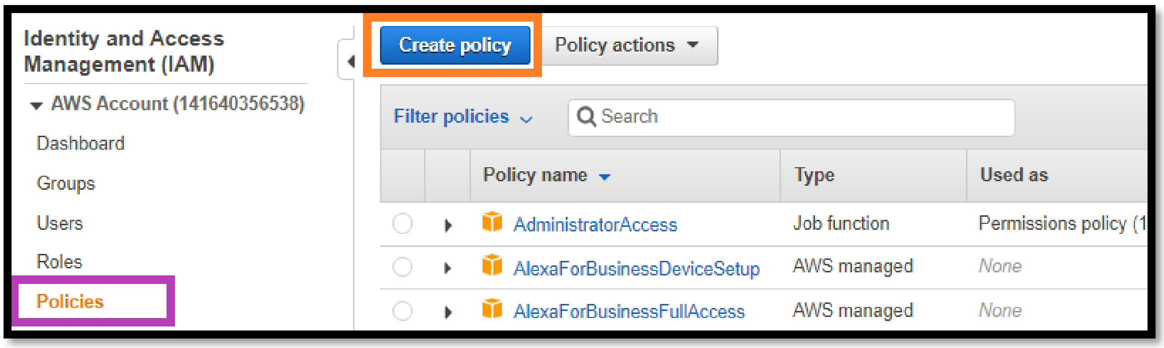
Note: We will discuss in details about **IAM** in upcoming module.

Step 1: Create IAM Policy for Amazon EC2 Server

6. In the **AWS Management Console**, on the **Services** menu Search and Select **IAM**.

7. Select **Policies**.

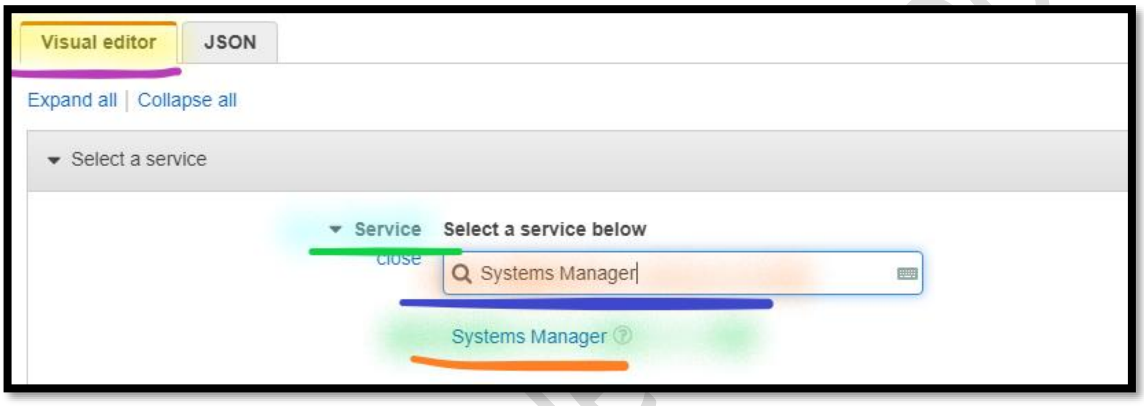
8. Click on **Create policy**.



Policy name	Type	Used as
AdministratorAccess	Job function	Permissions policy (1
AlexaForBusinessDeviceSetup	AWS managed	None
AlexaForBusinessFullAccess	AWS managed	None

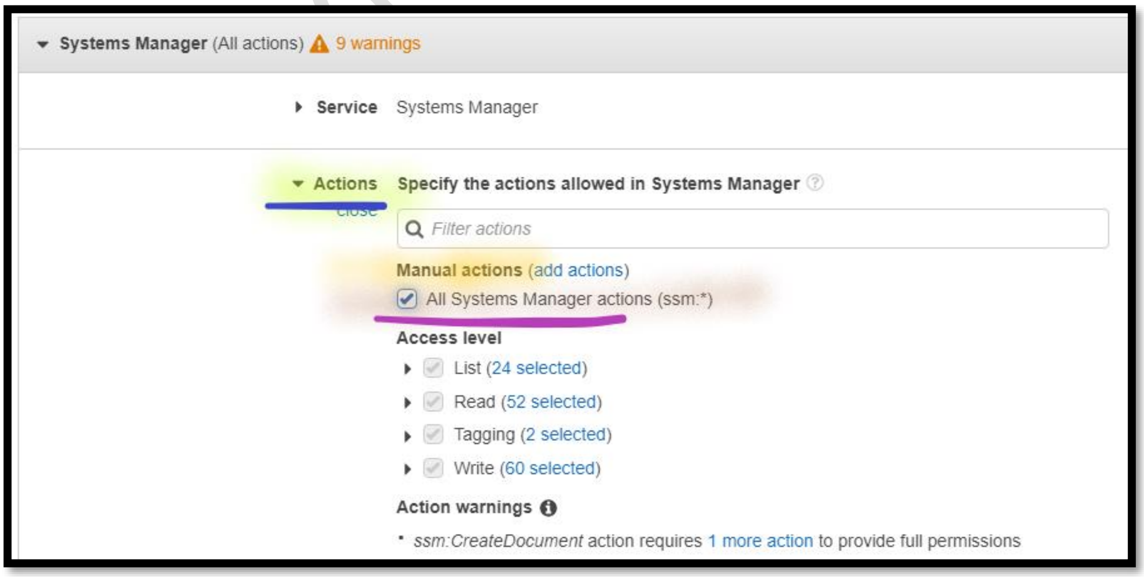
a. Select the **Visual editor**, expand the **Services**.

i. Search and Select **Systems Manager**.



b. **Expand** the **Actions**.

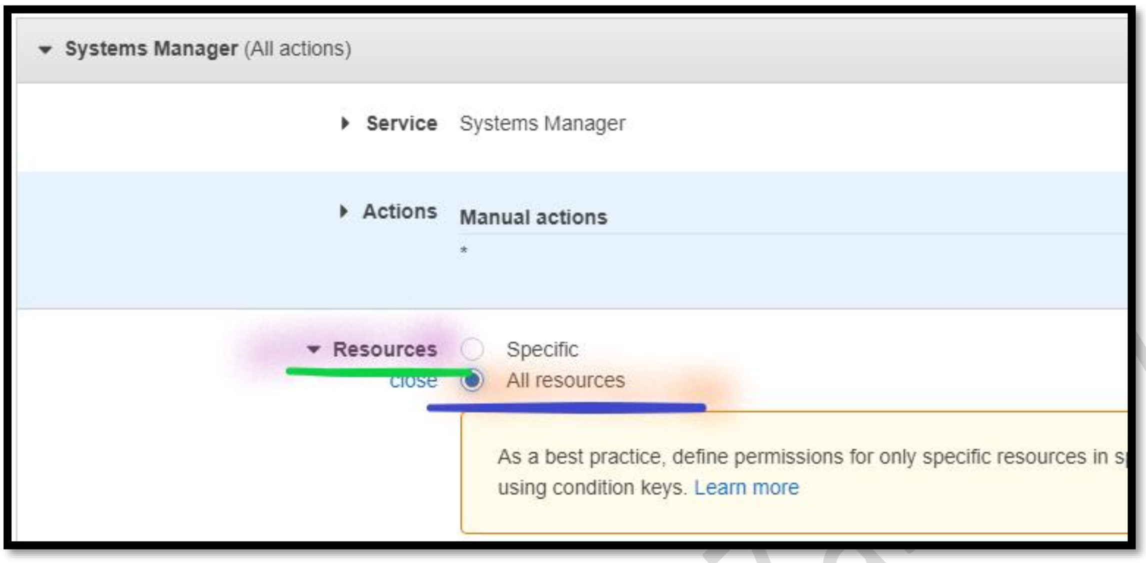
i. Select **All Systems Manager actions** under **Manual actions**.



• *ssm:CreateDocument* action requires 1 more action to provide full permissions

c. **Expand** the **Resources**.

i. Select **All Resources**.



Systems Manager (All actions)

Service: Systems Manager

Actions: Manual actions *

Resources: Specific / [✓ All resources]

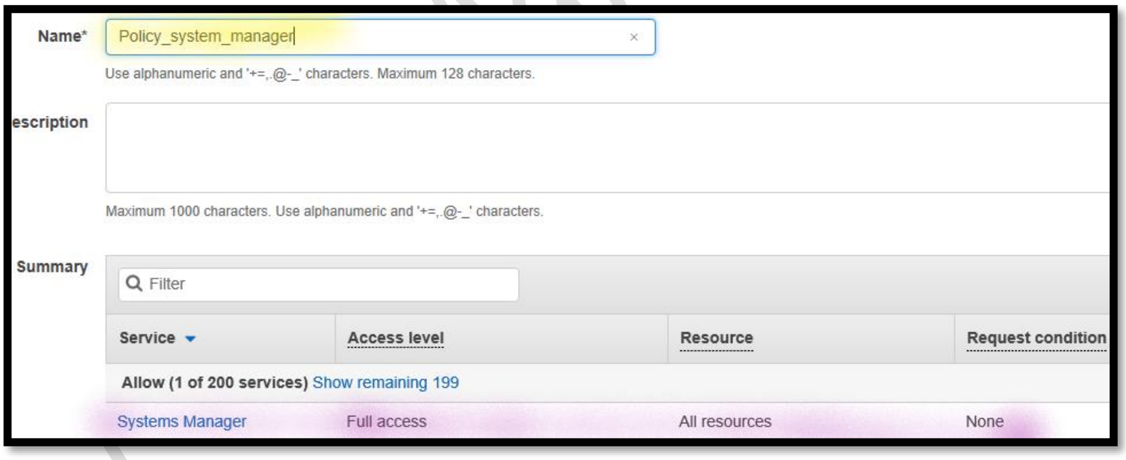
As a best practice, define permissions for only specific resources in s using condition keys. Learn more

ii. Select **Next: Tags**.

iii. Select **Next: Review**.

d. In the **Review Policy** page:

i. **Name**: Write **Policy_system_manager**.



Name*: Policy_system_manager

Use alphanumeric and '+=,.@-_' characters. Maximum 128 characters.

escription

Maximum 1000 characters. Use alphanumeric and '+=,.@-_' characters.

Summary

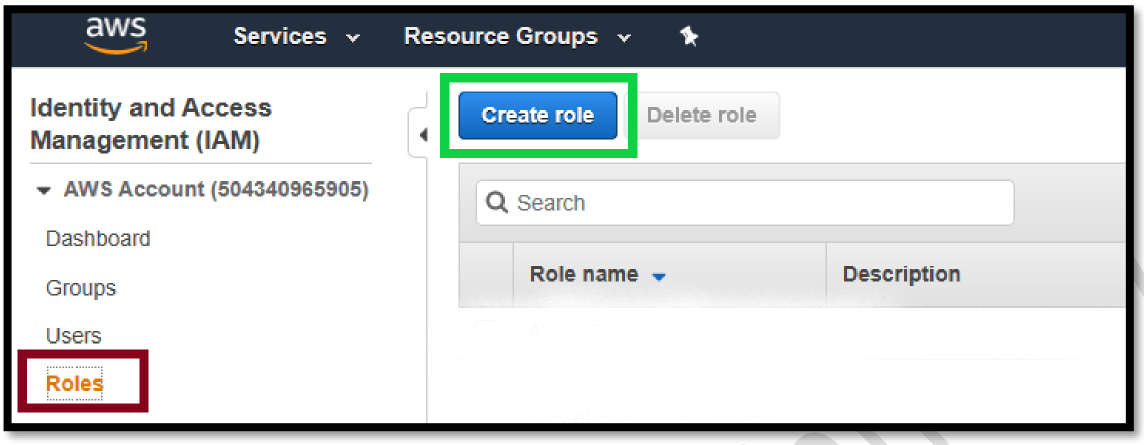
Service	Access level	Resource	Request condition
Allow (1 of 200 services) Show remaining 199			
Systems Manager	Full access	All resources	None

iv. Click on **Create Policy**.

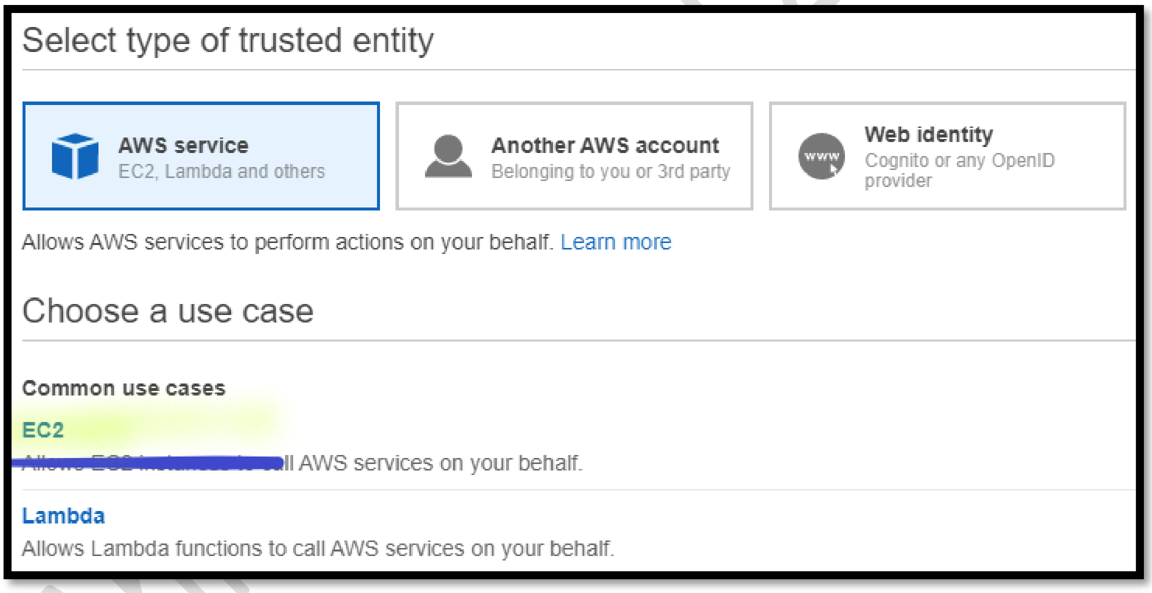
Note: You will get the message **Policy_system_manager** has been created.

Step 2: Create IAM Role for Amazon EC2 Server

9. ***Go to the left*** side, Select **Roles**.
10. Click on **Create role**.



i. Select **EC2** under **AWS service**



Select type of trusted entity

AWS service
EC2, Lambda and others

Another AWS account
Belonging to you or 3rd party

Web identity
Cognito or any OpenID provider

Allows AWS services to perform actions on your behalf. Learn more

Choose a use case

Common use cases

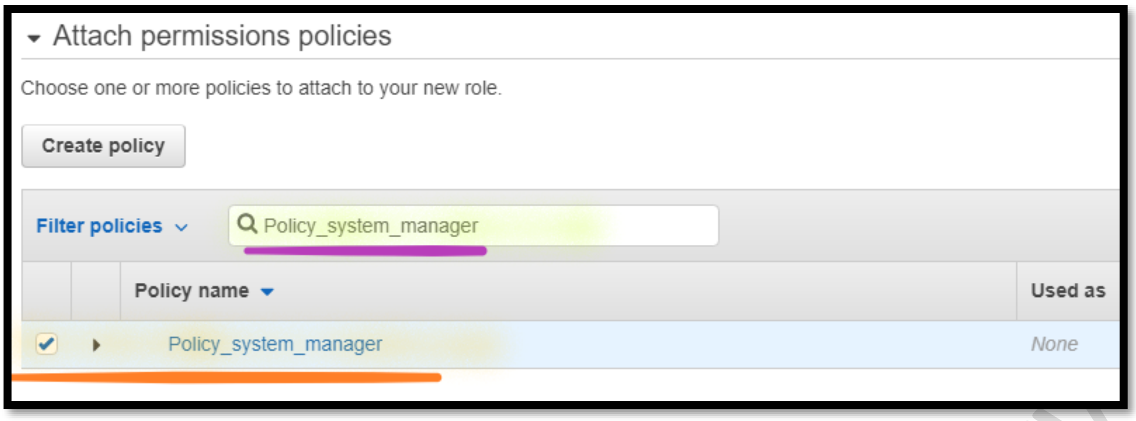
EC2
[redacted]ll AWS services on your behalf.

Lambda
Allows Lambda functions to call AWS services on your behalf.

ii. Select **Next: Permissions**.

iii. Write **Policy_system_manager** under **Search**.

i. Select **Policy_system_manager**.



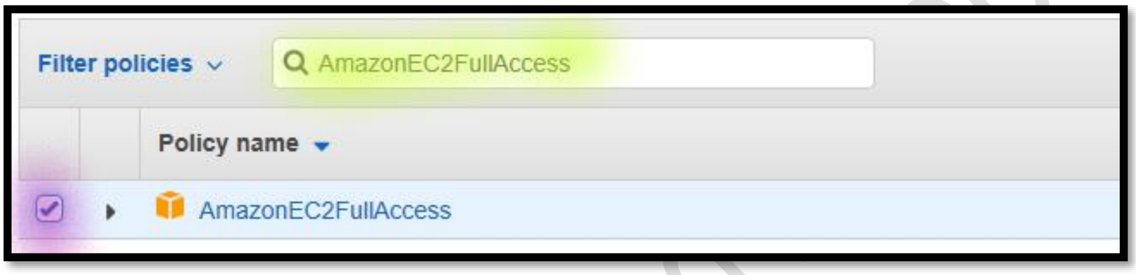
▾ Attach permissions policies

Choose one or more policies to attach to your new role.

Policy_system_manager

		Policy name ▾	Used as
✓	▸	Policy_system_manager	None

iv. In the ***same window***, Search and Select **AmazonEC2FullAccess**.



AmazonEC2FullAccess

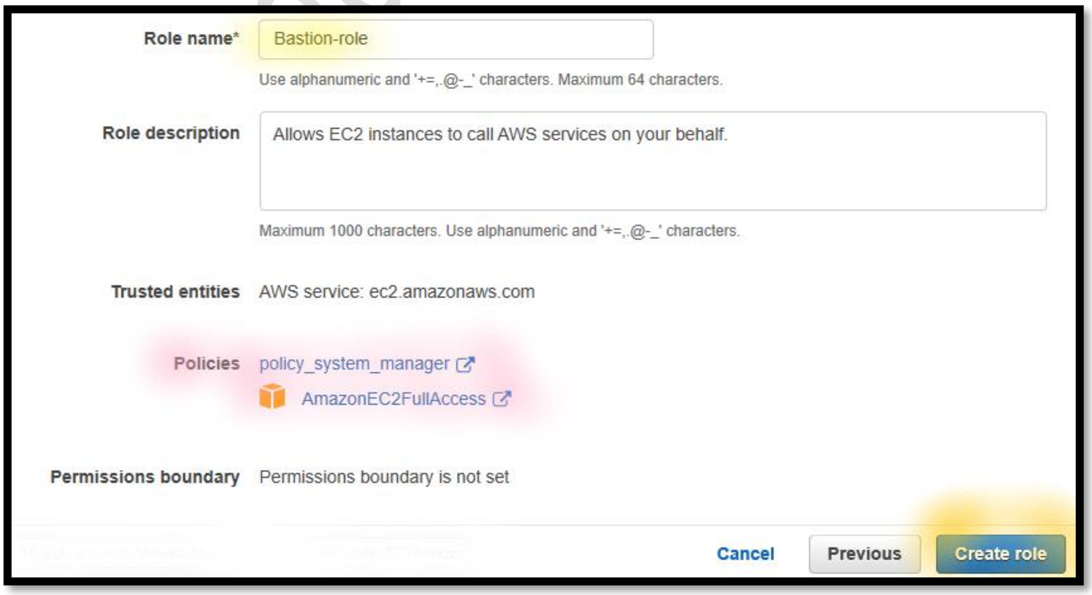
		Policy name ▾
✓	▸	AmazonEC2FullAccess

v. Select **Next: Tags**.

vi. Select **Next: Review**.

Note: Here you will see **two policies**, you have selected in previous step.

vii. **Role name**: Write **Bastion-role**.



Role name*	Bastion-role
	Use alphanumeric and '+=,.@-_' characters. Maximum 64 characters.
Role description	Allows EC2 instances to call AWS services on your behalf.
	Maximum 1000 characters. Use alphanumeric and '+=,.@-_' characters.
Trusted entities	AWS service: ec2.amazonaws.com
Policies	policy_system_manager AmazonEC2FullAccess
Permissions boundary	Permissions boundary is not set

viii. Click on **Create role**.

Note: You get the message "The role **Bastion-role** has been created".

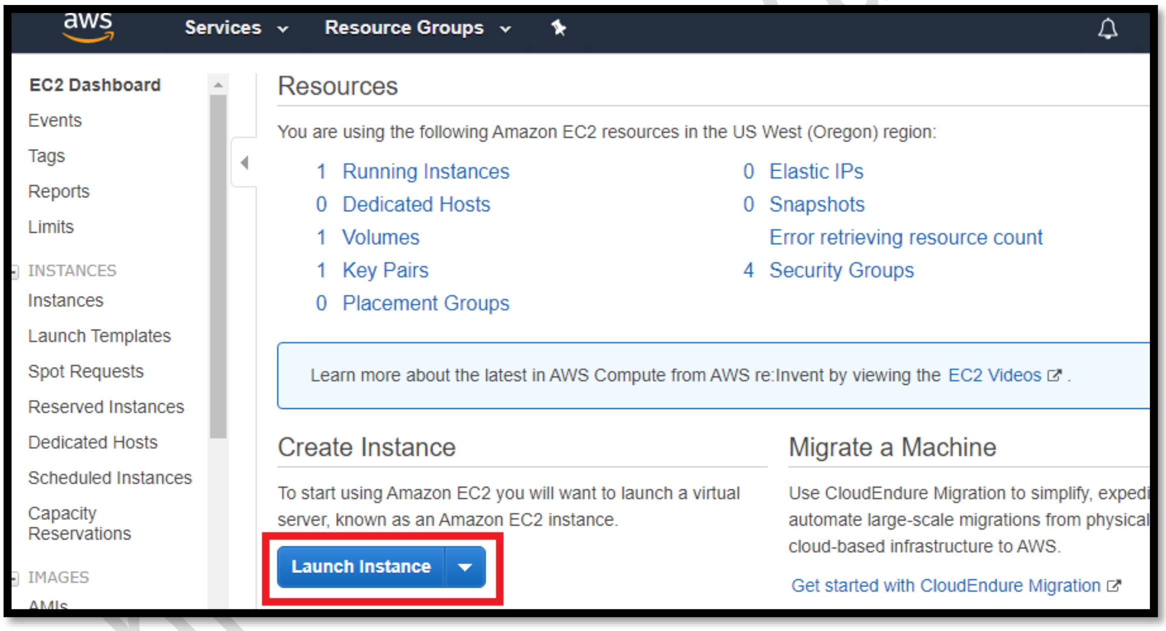
Task 3: Launch an Amazon EC2 Instance using the Management Console

In this task, you will launch an Amazon EC2 instance using the management console. The instance will be a Bastion Server, from which you can use the AWS Command-Line Interface (AWS CLI).

Step 1: Launch Amazon Instance

11. In the **AWS Management Console**, on the **Services** menu, click **EC2**.

12. Click **Launch Instance** and Click **Launch Instance**.



Resources

You are using the following Amazon EC2 resources in the US West (Oregon) region:

1 Running Instances	0 Elastic IPs
0 Dedicated Hosts	0 Snapshots
1 Volumes	Error retrieving resource count
1 Key Pairs	4 Security Groups
0 Placement Groups	

Learn more about the latest in AWS Compute from AWS re:Invent by viewing the EC2 Videos .

Create Instance

To start using Amazon EC2 you will want to launch a virtual server, known as an Amazon EC2 instance.

Launch Instance

Migrate a Machine

Use CloudEndure Migration to simplify, expedi automate large-scale migrations from physical cloud-based infrastructure to AWS.

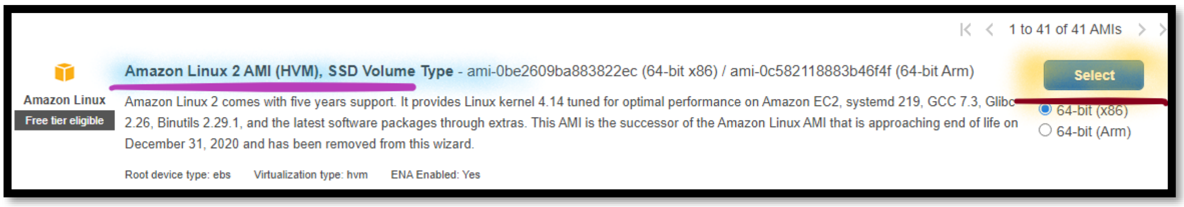
Get started with CloudEndure Migration

Step 2: Choose an Amazon Machine Image (AMI)

This step allows you to choose an AMI, which contains a copy of the disk volume that will be used to launch the instance.

Your Bastion Server will use ***Amazon Linux 2***.

13. Beside the **Amazon Linux 2 AMI** (at the top), click **Select**.



1 to 41 of 41 AMIs

Amazon Linux 2 AMI (HVM), SSD Volume Type - ami-0be2609ba883822ec (64-bit x86) / ami-0c582118883b46f4f (64-bit Arm)

Amazon Linux
Free tier eligible

Amazon Linux 2 comes with five years support. It provides Linux kernel 4.14 tuned for optimal performance on Amazon EC2, systemd 219, GCC 7.3, Glibc 2.26, Binutils 2.29.1, and the latest software packages through extras. This AMI is the successor of the Amazon Linux AMI that is approaching end of life on December 31, 2020 and has been removed from this wizard.

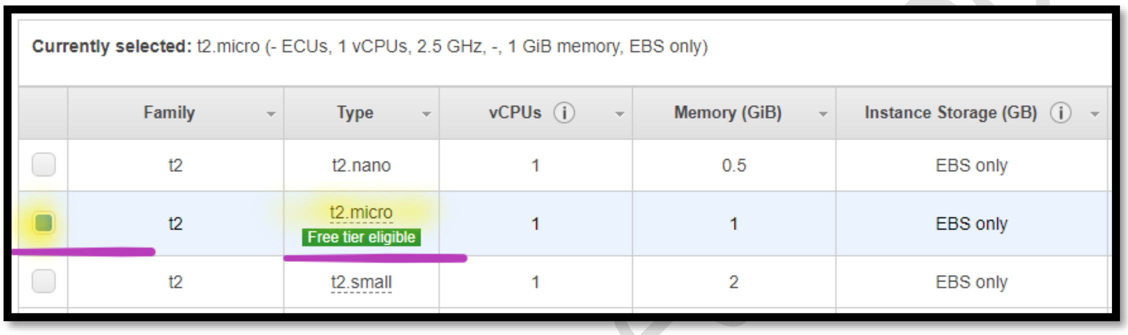
Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

[✓ 64-bit (x86)] 64-bit (Arm)

Step 3: Choose an Instance Type

Your application will use a **t2.micro**, Instance Type, which is a small instance that can burst above baseline performance when it is busy. It is ideal for development, testing and for applications that have bursty workloads.

14. Select **t2.micro**.



Currently selected: t2.micro (- ECUs, 1 vCPUs, 2.5 GHz, -, 1 GiB memory, EBS only)

	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)
☐	t2	t2.nano	1	0.5	EBS only
☑	t2	t2.micro Free tier eligible	1	1	EBS only
☐	t2	t2.small	1	2	EBS only

15. Click **Next: Configure Instance Details**.

Step 4: Configure Instance Details

This step allows you to configure instance details, such as the number of instances to launch and the network configuration.

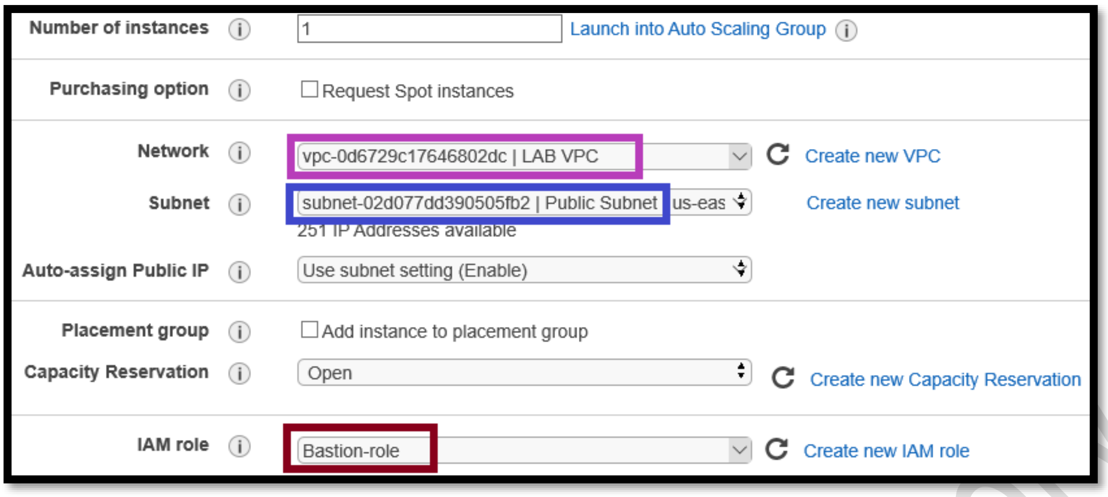
You will launch the instance in a public subnet within the ***LAB VPC*** network.

16. **Configure** these **settings**:

   a. **Network**: Dropdown and select **LAB VPC-01**.

   b. **Subnet**: Dropdown and select **Public subnet**.

   c. **IAM role**: Dropdown and **Bastion-role**.



Number of instances: 1 — Launch into Auto Scaling Group

Purchasing option: ☐ Request Spot instances

Network: vpc-0d6729c17646802dc | LAB VPC — Create new VPC

Subnet: subnet-02d077dd390505fb2 | Public Subnet | us-eas — Create new subnet

251 IP Addresses available

Auto-assign Public IP: Use subnet setting (Enable)

Placement group: ☐ Add instance to placement group

Capacity Reservation: Open — Create new Capacity Reservation

IAM role: Bastion-role — Create new IAM role

The ***Bastion-Role*** grants permission to applications running on the instance to make requests to the Amazon EC2 service. This is required for the second half of this lab, where you will use the AWS CLI to communicate with the EC2 service.

17. Click **Next: Add Storage**.

Step 5: Add Storage

This step can be used to add additional Amazon Elastic Block Store (EBS) disk volumes and configure their size and performance.

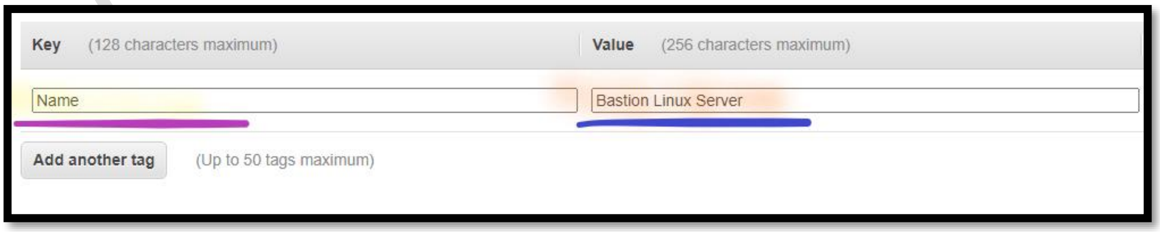
18. Click **Next: Add Tags**.

Step 6: Add Tags

Tags allow you to categorize your AWS resources in different ways, such as by purpose, owner, or environment. Each tag consists of a *Key* and a *Value*, both of which you define.

19. Click **Add Tag** then **configure**:

   a. **Key**: Write **Name**.
   b. **Value:** Write **Bastion Linux Server**.



Key (128 characters maximum)	Value (256 characters maximum)
Name	Bastion Linux Server

Add another tag (Up to 50 tags maximum)

This name will appear on the instance in the EC2 **management console**.

20. Click **Next: Configure Security Group**.

Step 7: Configure Security Group

You will create a new Security Group that permits SSH connections. This security group will allow you to log in to the Bastion Server via SSH.

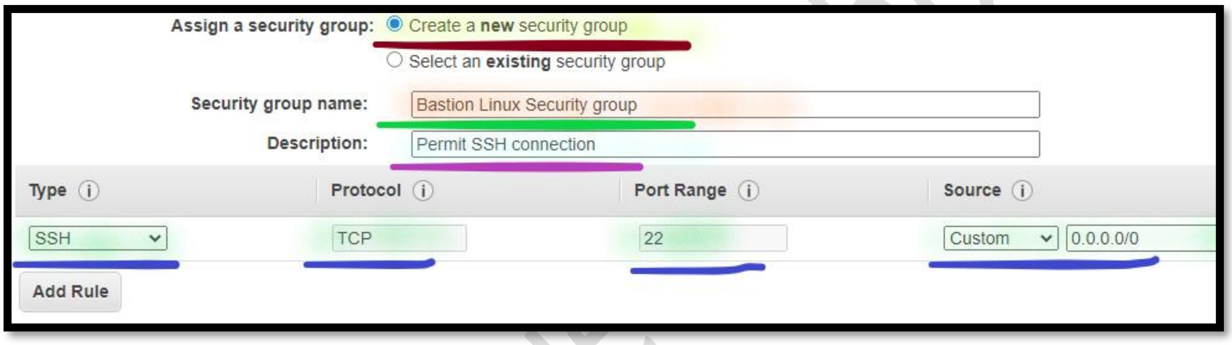
21. **Configure** these **settings**:

    a. **Assign a security group**: Select Create a new security group.

    b. **Security group name**: Write Bastion Linux Security group.

    c. **Description**: Write Permit SSH connection.

Permissions for inbound access via **SSH (port 22)** have already been configured by default.



Assign a security group: (● Create a **new** security group) ○ Select an **existing** security group

Security group name: Bastion Linux Security group

Description: Permit SSH connection

Type	Protocol	Port Range	Source
SSH	TCP	22	Custom 0.0.0.0/0

22. Click Review and Launch.

Step 8: Review Instance Launch

This step displays a summary of the configuration for the instance you are about to launch.

23. Click Launch.

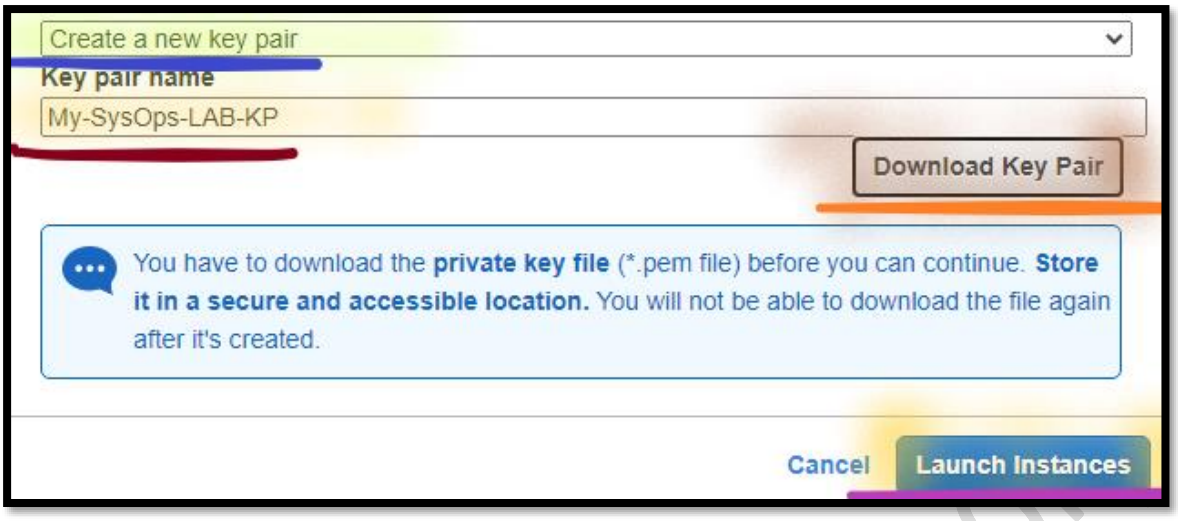
Note: A new window will popup.

24. In the **Key pair** page, Dropdown and select Create a new key pair.

    a. **Key pair name**: Write My-AWS-LAB-KP.

    b. Click on Download the Key pair file and **store** in safe place.

25. Click Launch Instances.


Create a new key pair

Key pair name

My-SysOps-LAB-KP

Download Key Pair

You have to download the **private key file** (*.pem file) before you can continue. **Store it in a secure and accessible location.** You will not be able to download the file again after it's created.

Cancel Launch Instances

Note: Your instance will now be launched.

26.Click on **View Instances**.

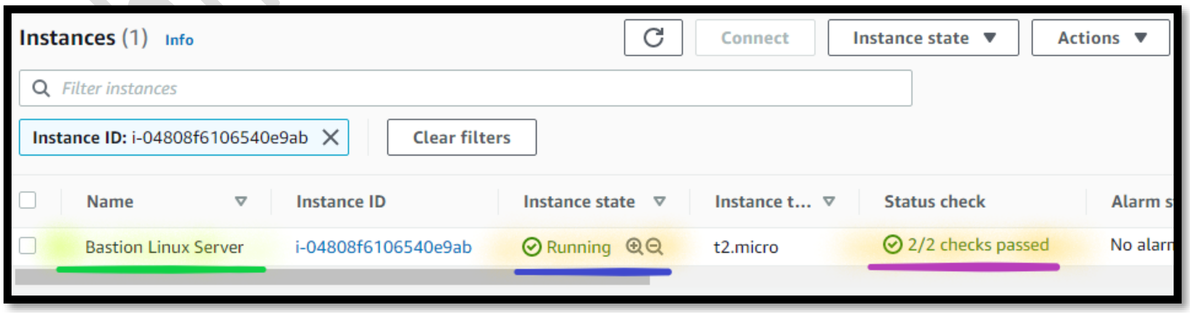
Step 9: Check the EC2 Server Status

The Bastion Server will appear in a *pending state*, which means it is being launched. It will then change to *running*, which indicates that the instance has started booting.

a. **Wait** for the **Instance State** to change to **Running** state.

b. **Wait** for the **Status check** to change to **2/2 checks passed**.

Note: **Wait** (***~5 mnts.***) for the Installation of Linux server.



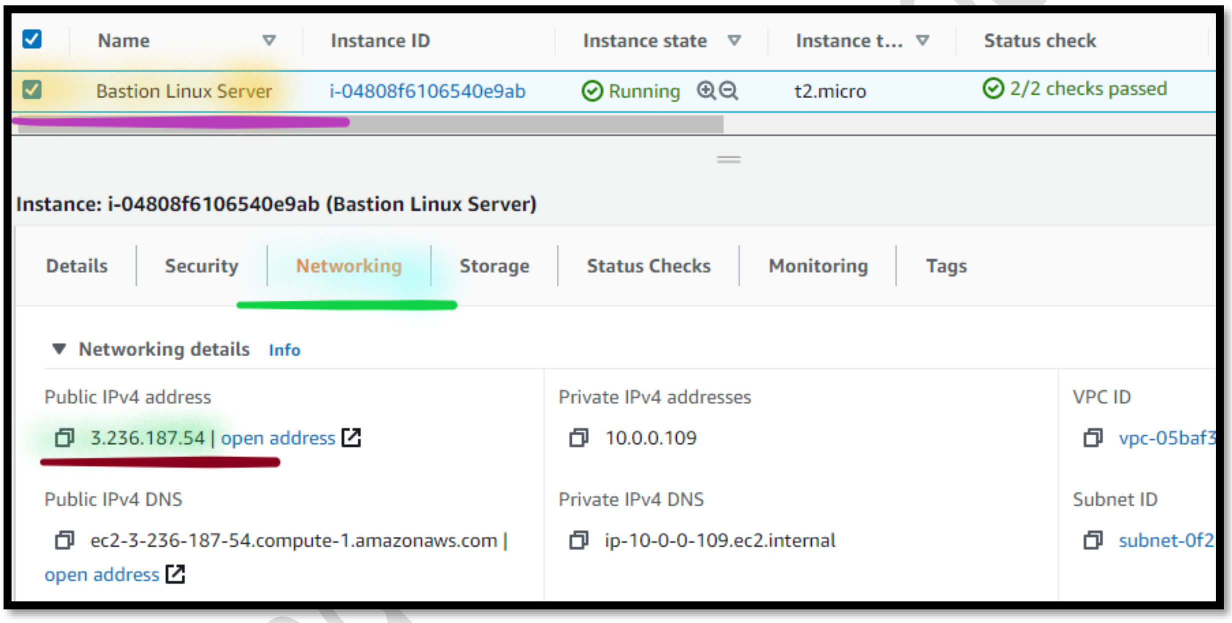
Name	Instance ID	Instance state	Instance t...	Status check	Alarm s
Bastion Linux Server	i-04808f6106540e9ab	Running	t2.micro	2/2 checks passed	No alarr

Task 4: Log into the Bastion Server

In this task, you will log into the Bastion Server that you just created.

Step 1: Copy to Linux Server Public IP address

27. In the **AWS Management Console**, on the **Services** menu, click **EC2**.
28. Click **Instances**.
29. Select **Bastion Linux Server**.
    a. Click on **Networking**.
    b. **Copy** the **Public IP address**.



	Name	Instance ID	Instance state	Instance t...	Status check
✓	Bastion Linux Server	i-04808f6106540e9ab	Running	t2.micro	2/2 checks passed

Instance: i-04808f6106540e9ab (Bastion Linux Server)

Public IPv4 address: 3.236.187.54 | open address
Private IPv4 addresses: 10.0.0.109
VPC ID: vpc-05baf3
Public IPv4 DNS: ec2-3-236-187-54.compute-1.amazonaws.com | open address
Private IPv4 DNS: ip-10-0-0-109.ec2.internal
Subnet ID: subnet-0f2

Note: The following instructions now **vary** slightly depending on whether you are using **Windows or Mac/Linux**.

Step 2: Install PuTTy

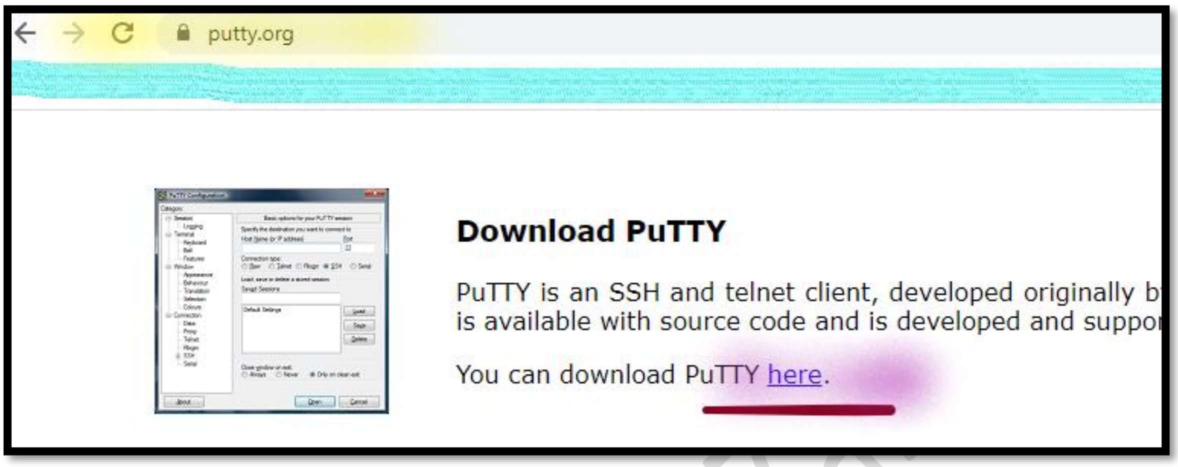
Note: If you are using **Mac Operating System**, go below to follow the **Step 5**.

PuTTY does not natively support the private key format (.pem) generated by Amazon EC2. PuTTY has a tool named PuTTYgen, which can convert keys to the required PuTTY format (.ppk). You must convert your private key into this format (.ppk) before attempting to connect to your instance using PuTTY.

30. **Install** **PuTTY** on **Windows** Operating System.

a. Browse **www.putty.org**.

b. Click on **You can download PuTTY here** under **Download PuTTY**.



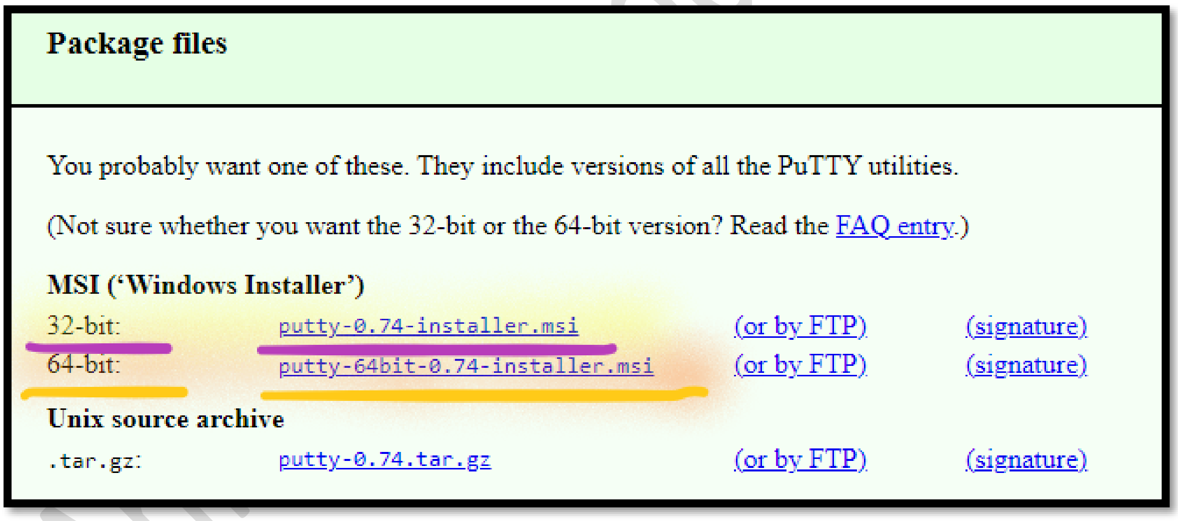
putty.org

Download PuTTY

PuTTY is an SSH and telnet client, developed originally b
is available with source code and is developed and suppo

You can download PuTTY here.

c. **Download** the **MSI Installer** (32-bit or 64-bit as per your local desktop/ laptop OS version) under **Package files**.



Package files

You probably want one of these. They include versions of all the PuTTY utilities.

(Not sure whether you want the 32-bit or the 64-bit version? Read the FAQ entry.)

MSI ('Windows Installer')

32-bit:	putty-0.74-installer.msi	(or by FTP)	(signature)
64-bit:	putty-64bit-0.74-installer.msi	(or by FTP)	(signature)

Unix source archive

.tar.gz:	putty-0.74.tar.gz	(or by FTP)	(signature)

d. **Install** the **MSI Installer**.

Note: You can also copy **PuTTY**.exe and **PuTTygen**.exe, which is also available with the Lab manual.

Step 3: Convert PEM file into PPK file

31. **Convert** Private Key using ***PuTTYgen***.

a. In the ***local Desktop/ Laptop*** (Windows 10), right click on **Start** & **Run**

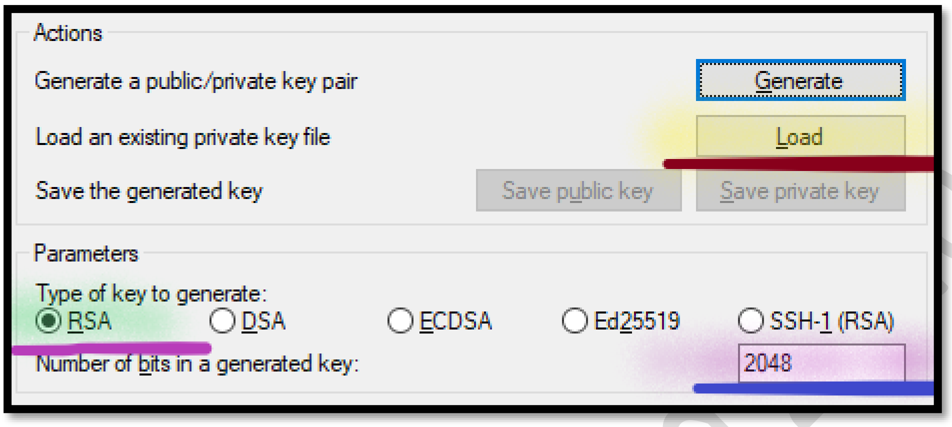
b. In the open, write **puttygen.exe** generator

c. In the **Parameters** section:

  i. Select **RSA**.

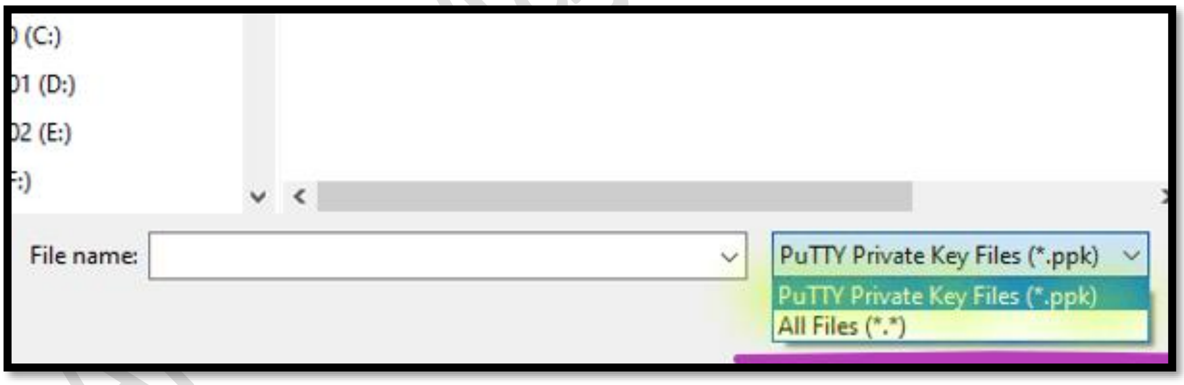
  ii. **No. of bits in generated key** should be **2048**.

d. Choose **Load**.


Actions
Generate a public/private key pair — Generate
Load an existing private key file — Load
Save the generated key — Save public key, Save private key
Parameters
Type of key to generate:
[✓ RSA] DSA ECDSA Ed25519 SSH-1 (RSA)
Number of bits in a generated key: 2048

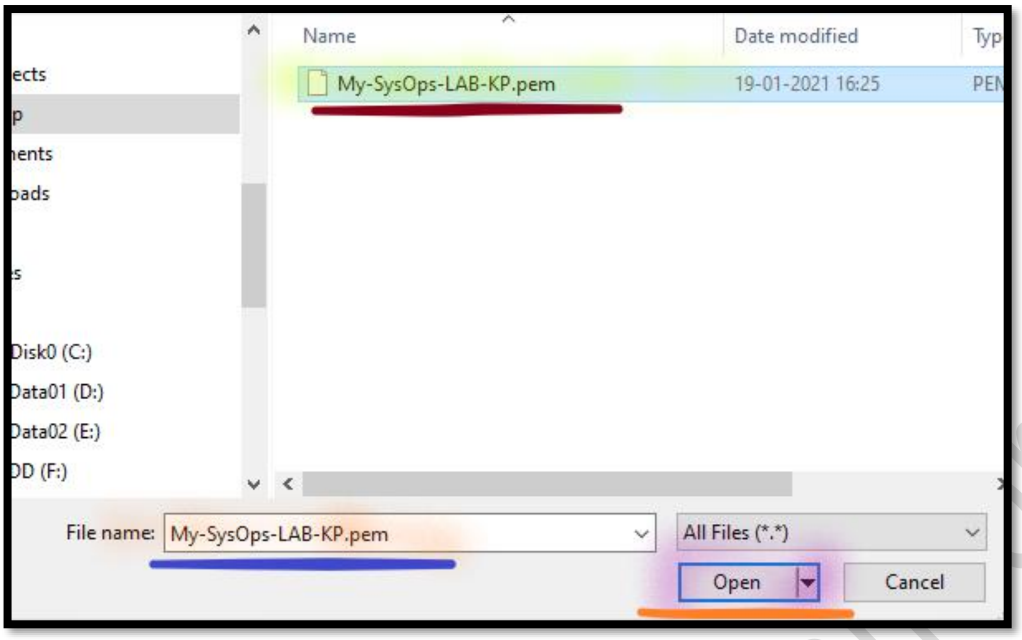
By default, PuTTYgen displays only files with the **extension .ppk**. To locate your .pem file.

  i. **File extension**: Dropdown and Select **All files**, to locate your ***.pem*** extension files.


File name:
PuTTY Private Key Files (*.ppk)
PuTTY Private Key Files (*.ppk)
All Files (*.*)

  ii. **Navigate** to the folder where you downloaded your key pair and select **My-AWS-LAB-KP**.pem file.

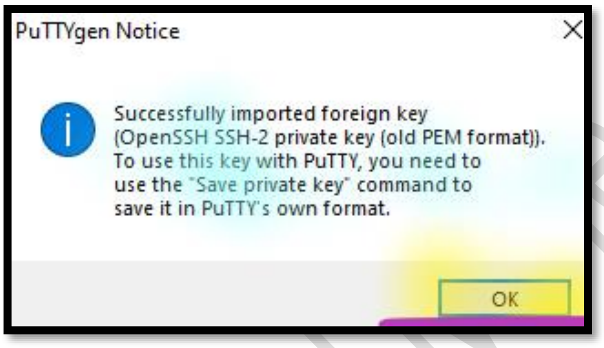
  iii. Select **Open**.



My-SysOps-LAB-KP.pem 19-01-2021 16:25 PEM

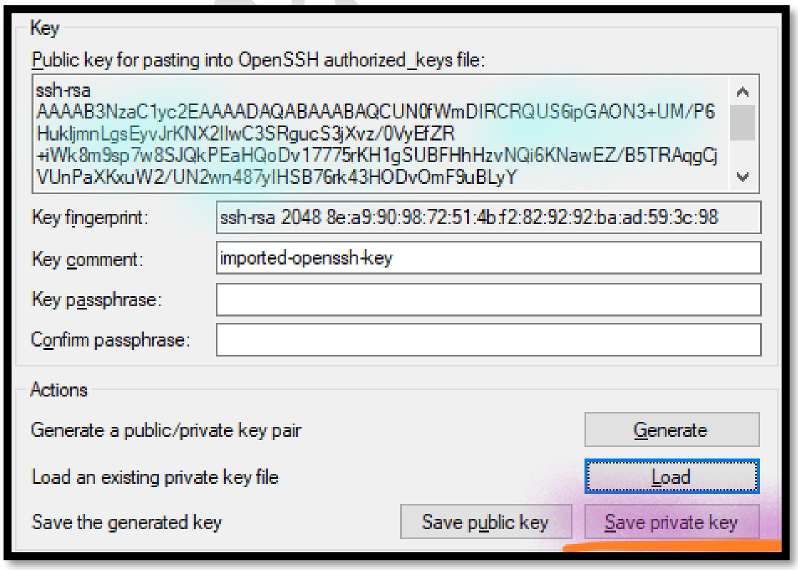
File name: My-SysOps-LAB-KP.pem

iv. Choose **OK** on the confirmation dialog box.


PuTTYgen Notice

Successfully imported foreign key (OpenSSH SSH-2 private key (old PEM format)). To use this key with PuTTY, you need to use the "Save private key" command to save it in PuTTY's own format.

e. Choose **Save private key** to save the key in the format that PuTTY can use.


Key

Public key for pasting into OpenSSH authorized_keys file:

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQCUN0fWmDIRCRQUS6ipGAON3+UM/P6
HukljmnLgsEyvJrKNX2IlwC3SRgucS3jXvz/0VyEfZR
+iWk8m9sp7w8SJQkPEaHQoDv17775rKH1gSUBFHhHzvNQi6KNawEZ/B5TRAqgCj
VUnPaXKxuW2/UN2wn487yIHSB76rk43HODvOmF9uBLyY
```

Key fingerprint: ssh-rsa 2048 8e:a9:90:98:72:51:4b:f2:82:92:92:ba:ad:59:3c:98

Key comment: imported-openssh-key

Key passphrase:

Confirm passphrase:

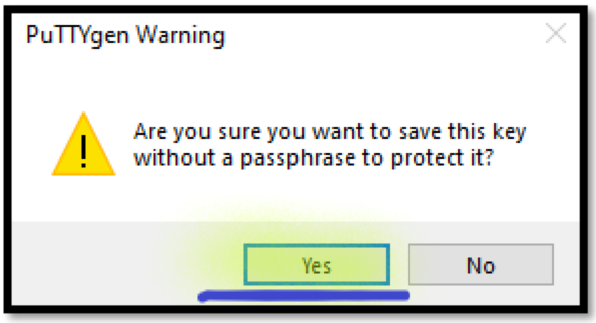
Actions

Generate a public/private key pair

Load an existing private key file

Save the generated key

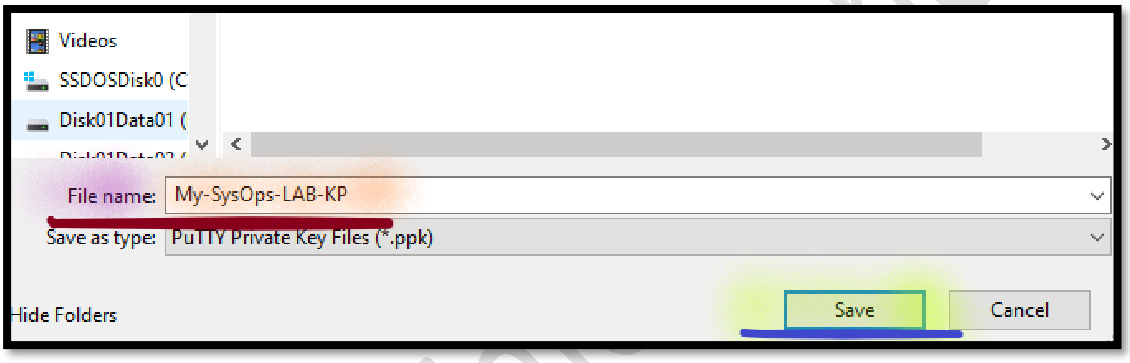
f. Choose **Yes**, when PuTTYgen ***displays a warning*** about saving the key without a passphrase.



Are you sure you want to save this key without a passphrase to protect it?

i. **File name**: Write **My-AWS-LAB-KP**.

ii. Select **Save**. PuTTY automatically adds the **.ppk** file extension.



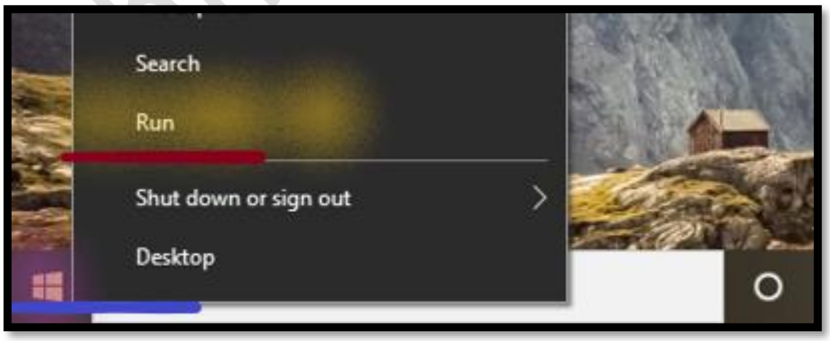
File name: My-SysOps-LAB-KP

Save as type: PuTTY Private Key Files (*.ppk)

Your private key is now in the correct format for use with PuTTY. You can now connect to your instance using PuTTY's SSH client.

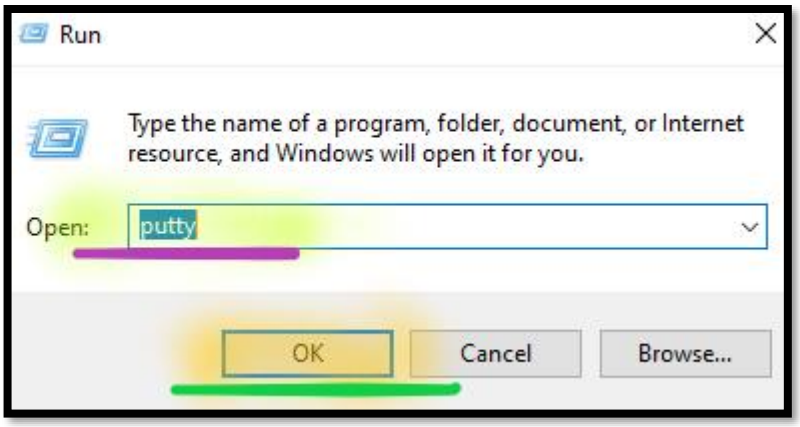
Step 4: Connect to Linux Server from Windows Operating System

32.From the **Local Desktop/ Laptop** (***Windows 10***), right click on **Start** & **Run**.



33.In the open, write **putty**.

34.Select **Ok**, it will open the PuTTY window.



Type the name of a program, folder, document, or Internet resource, and Windows will open it for you.

Open: putty

35.In the **Category** pane:

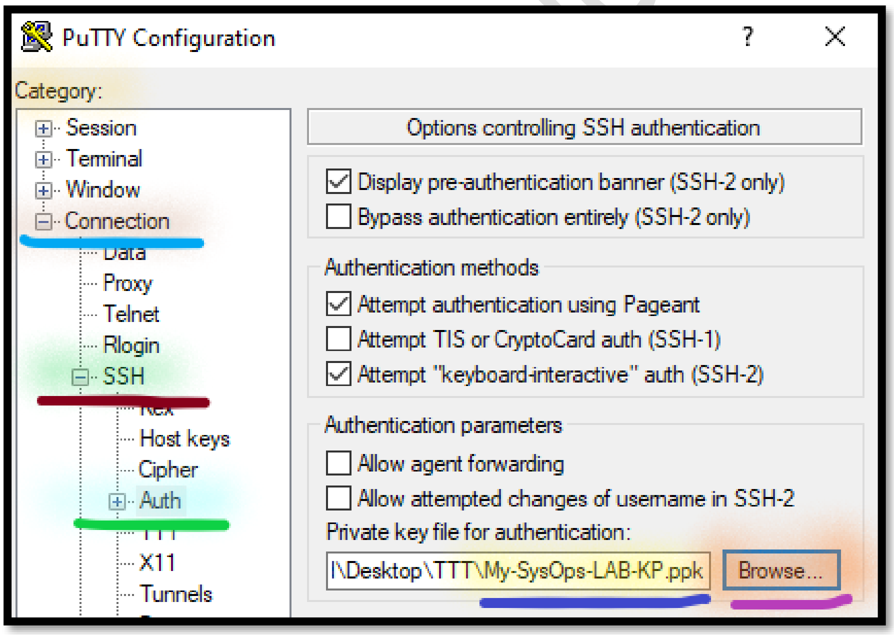
a. Expand **Connection**.

b. Expand **SSH**.

c. Select **Auth**.

i. Select **Browse**.

ii. **Navigate** and Select the **My-AWS-LAB-KP**.ppk file that you generated in previous step.



☑ Display pre-authentication banner (SSH-2 only)
☐ Bypass authentication entirely (SSH-2 only)

Authentication methods
☑ Attempt authentication using Pageant
☐ Attempt TIS or CryptoCard auth (SSH-1)
☑ Attempt "keyboard-interactive" auth (SSH-2)

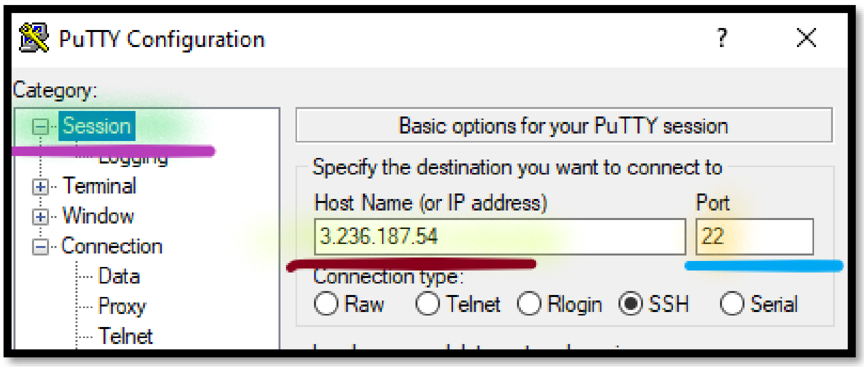
Authentication parameters
☐ Allow agent forwarding
☐ Allow attempted changes of username in SSH-2

Private key file for authentication:
|\Desktop\TTT\My-SysOps-LAB-KP.ppk

d. Select **Session**, in the **Category** pane.

i. **Host name**: Write **Public IP Address** of Linux virtual machine, which you have copied from the previous step.

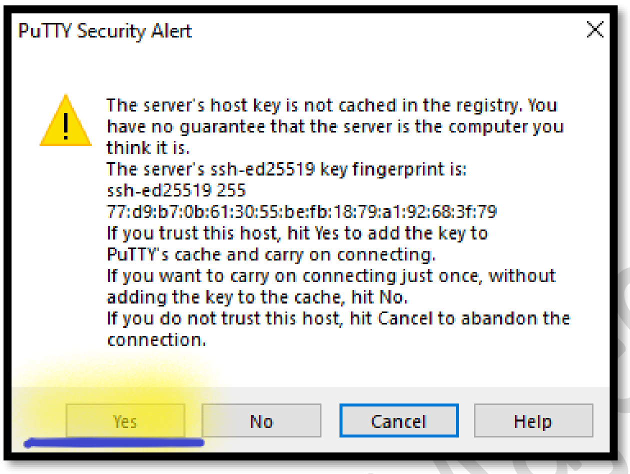
ii. **Port**: Write **22**.



Specify the destination you want to connect to
Host Name (or IP address): 3.236.187.54
Port: 22
Connection type: Raw Telnet Rlogin [✓ SSH] Serial

e. Select **Open** to start the PuTTY session.

i. Select **Yes**, once PuTTY displays a security alert dialog box.

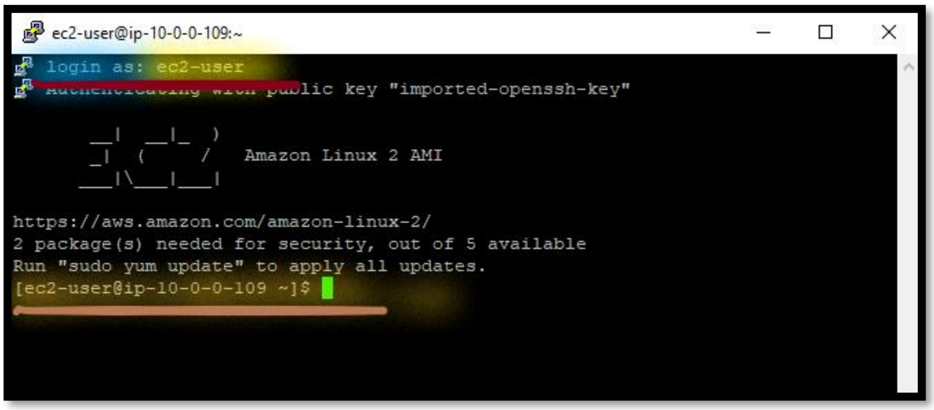

PuTTY Security Alert

The server's host key is not cached in the registry. You have no guarantee that the server is the computer you think it is.
The server's ssh-ed25519 key fingerprint is:
ssh-ed25519 255
77:d9:b7:0b:61:30:55:be:fb:18:79:a1:92:68:3f:79
If you trust this host, hit Yes to add the key to PuTTY's cache and carry on connecting.
If you want to carry on connecting just once, without adding the key to the cache, hit No.
If you do not trust this host, hit Cancel to abandon the connection.

Note: **Linux terminal** opens and you are connected to your instance.

f. **Login as**: Write **ec2-user**.

Note: Because you are using a key pair for authentication, you will not be **prompted for a password**.



```
login as: ec2-user
Authenticating with public key "imported-openssh-key"

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
2 package(s) needed for security, out of 5 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-10-0-0-109 ~]$
```

Note: Go to the next step, **But Don't close the Linux terminal**.

Step 5: Connect to Linux Server from MAC/ Linux Operating System

Note: If you are using **Windows** Operating System, go above and follow the **Step 2**.

36. **Copy** this command to a **text editor**:

```
chmod 400 My-AWS-LAB-KP.pem
ssh -i My-AWS-LAB-KP.pem ec2-user@PUBLIC-IP-ADDRESS
```

Note: **Replace** the **Public IP Address** with the **Public IP address** of the **Linux** virtual machine, which you have copied in the previous step.

37. **Paste** the command into the **MAC/ Linux** terminal window and run it.
    a. Type **Yes** when prompted to allow a first connection to this remote SSH server.

Task 5: Create Security Group

In this task, you will create Security group, which will be used in the Web Server launched via AWS CLI.

Step 1: Create Security group

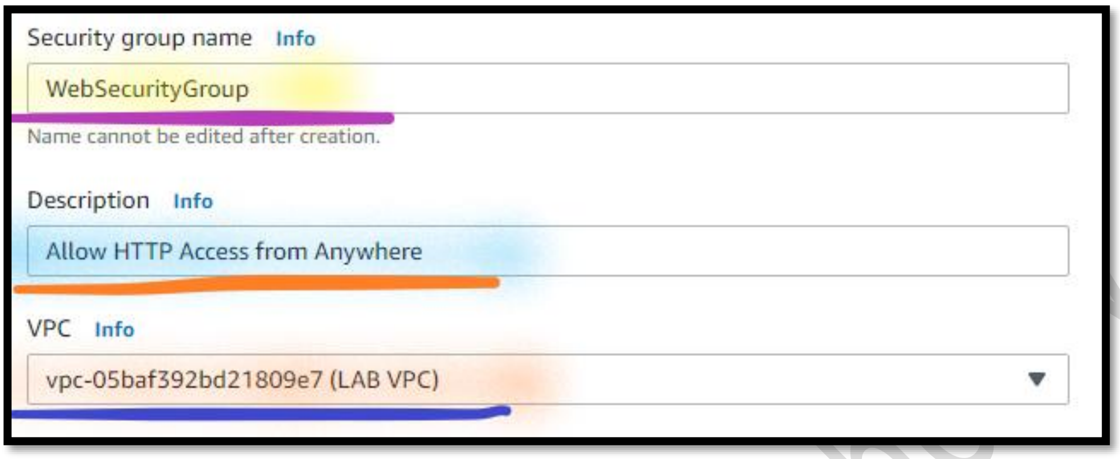
38. In the **AWS Management Console**, on the **Services** menu, click **EC2**.

39. ***Go to the left*** navigation pane, click **Security Groups** under **Network & Security**.

40. *Some **existing security groups will be displayed***. You will now create a new security group for the application servers.
    a. Click **Create security group** and **configure**:
        i. **Security group name**: Write **WebSecurityGroup**.

ii. **Description**: Write **Allow HTTP Access from Anywhere**.

iii. **VPC**: Dropdown and Select **LAB VPC**.



Security group name Info
WebSecurityGroup
Name cannot be edited after creation.

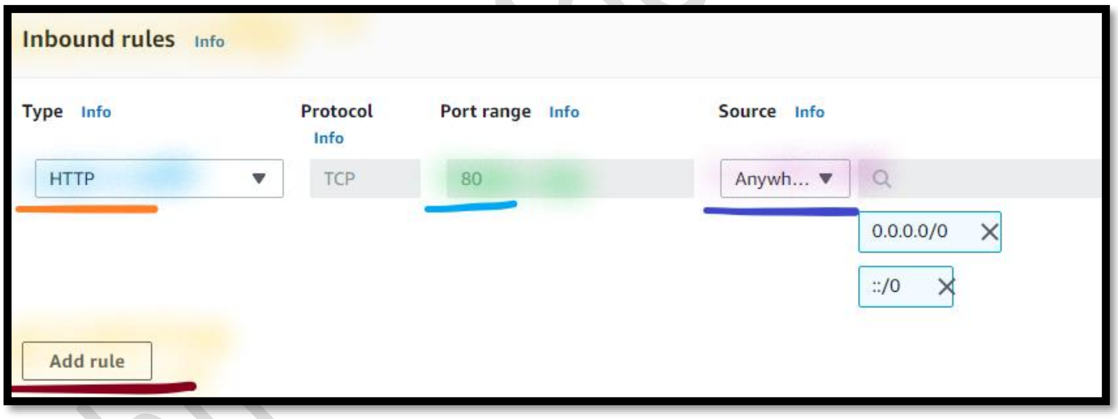
Description Info
Allow HTTP Access from Anywhere

VPC Info
vpc-05baf392bd21809e7 (LAB VPC)

b. Click **Add Rule** under **Inbound rules** then configure:

i. **Type**: Dropdown and Select **HTTP**.

ii. **Source**: Dropdown and Select **Anywhere**.



Inbound rules Info

Type	Protocol	Port range	Source
HTTP	TCP	80	Anywh... 0.0.0.0/0 ::/0

Note: **Don't do any changes** in the **Outbound Security Rule**. Keep the default rules.

c. Click **Create security group**.

Task 6: Launch a Web Server using the AWS CLI

In this task, you will launch an Amazon EC2 instance using the AWS Command-Line Interface (CLI). The AWS CLI makes it easy to automate the provision and configuration of AWS resources.

The new instance will be configured as a Web Server.

Step 1: Launch web server using AWS CLI

41. **Return** to **Bastion Linux Server**.

42. **Obtain** the **AMI Id** to use:

One of the parameters required when launching an instance is the Amazon Machine Image (AMI), which will populate the boot disk of the instance. AMIs are continually patched and updated by AWS, so it is recommended to always use the latest AMI when launching instances.

You will use the **AWS Systems Manager Parameter Store** to obtain the ID of the most recent *Amazon Linux 2* AMI. AWS maintains a list of standard AMIs in the Parameter Store, making this task easy to automate.

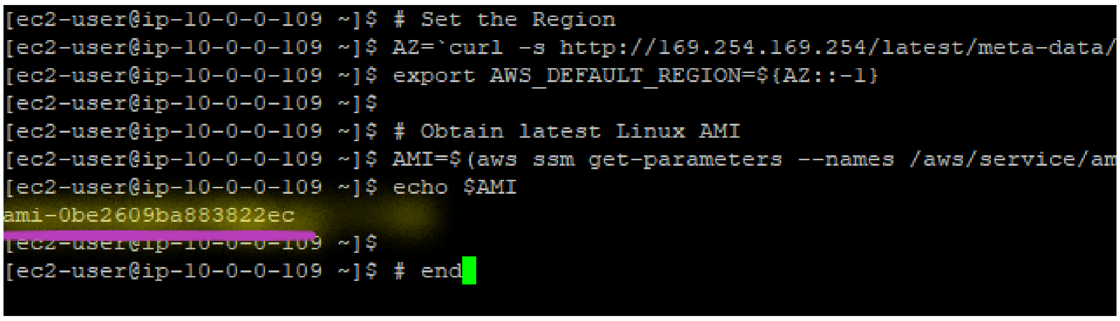
a. **Paste** the below **script** into **Linux terminal**:

```
# Set the Region
AZ=`curl -s http://169.254.169.254/latest/meta-data/placement/availability-
zone`
export AWS_DEFAULT_REGION=${AZ::-1}

# Obtain latest Linux AMI
AMI=$(aws ssm get-parameters --names /aws/service/ami-amazon-linux-
latest/amzn2-ami-hvm-x86_64-gp2 --query 'Parameters[0].[Value]' --output text)
echo $AMI
# end
```

Note: You can also copy the **Section 1** of the script from **LAB-01_Script Launch Web Server using CLI**.txt file provided with the LAB Manual.

Note: If ***command executed succesfully***, you can see the **AMI Id** in the response.



```
[ec2-user@ip-10-0-0-109 ~]$ # Set the Region
[ec2-user@ip-10-0-0-109 ~]$ AZ=`curl -s http://169.254.169.254/latest/meta-data/
[ec2-user@ip-10-0-0-109 ~]$ export AWS_DEFAULT_REGION=${AZ::-1}
[ec2-user@ip-10-0-0-109 ~]$
[ec2-user@ip-10-0-0-109 ~]$ # Obtain latest Linux AMI
[ec2-user@ip-10-0-0-109 ~]$ AMI=$(aws ssm get-parameters --names /aws/service/am
[ec2-user@ip-10-0-0-109 ~]$ echo $AMI
ami-0be2609ba883822ec
[ec2-user@ip-10-0-0-109 ~]$
[ec2-user@ip-10-0-0-109 ~]$ # end
```

This command **did the following**:

- Obtained the Region where the instance is running.
- Called the AWS Systems Manager (*ssm*) and used the **get-parameters** command to retrieve a value from the Parameter Store.
- The AMI requested was for Amazon Linux 2 (*amzn2-ami*).
- The AMI ID has been stored in an Environment Variable called *AMI.*

Note: If your **SSH session disconnects**, it will lose the information stored in environment variables. Once you reconnect, you will **need to re-run all of the steps** in this task, starting with the above commands to obtain the AMI ID..

43.**Obtain** the **Subnet Id** to use:

You will be launching the new instance in the Public Subnet. When launching an instance, the **SubnetId** can be specified.

The following command will retrieve the *SubnetId* of the PublicSubnet:

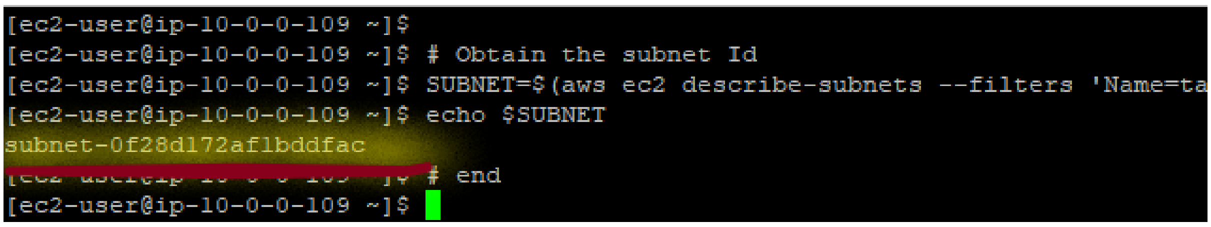
a. **Paste** the below **script** into **Linux terminal**:

```
# Obtain subnet Id
SUBNET=$(aws ec2 describe-subnets --filters 'Name=tag:Name,Values=Public
Subnet' --query Subnets[].SubnetId --output text)
echo $SUBNET
# end
```

This uses the AWS CLI to retrieve the Subnet Id of the subnet named *Public Subnet*.

Note: You can also copy the **Section 2** of the script from **LAB-01_Script Launch Web Server using CLI**.txt file provided with the LAB Manual.

Note: If ***command executed succesfully***, you can see the **Subnet Id** in the response.



```
[ec2-user@ip-10-0-0-109 ~]$
[ec2-user@ip-10-0-0-109 ~]$ # Obtain the subnet Id
[ec2-user@ip-10-0-0-109 ~]$ SUBNET=$(aws ec2 describe-subnets --filters 'Name=ta
[ec2-user@ip-10-0-0-109 ~]$ echo $SUBNET
subnet-0f28d172af1bddfac
[ec2-user@ip-10-0-0-109 ~]$ # end
[ec2-user@ip-10-0-0-109 ~]$
```

44.**Obtain** the **Security Group Id** to use:

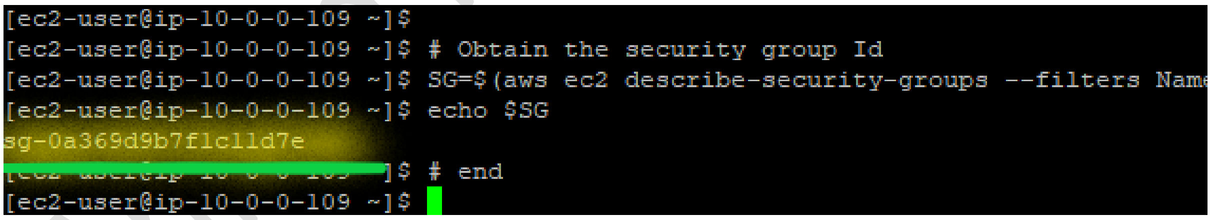
A *Web Security Group* has been created in the previous step, which allows inbound HTTP (Port 80) requests.

a. **Paste** the below **script** into **Linux terminal**:

```
# Obtain the security group Id
SG=$(aws ec2 describe-security-groups --filters Name=group-
name,Values=WebSecurityGroup --query SecurityGroups[].GroupId --output text)
echo $SG
# end
```

Note: You can also copy the **Section 3** of the script from **LAB-01_Script Launch Web Server using CLI**.txt file provided with the LAB Manual.

Note: If ***command executed succesfully***, you can see the **Security Group Id** in the response.



```
[ec2-user@ip-10-0-0-109 ~]$
[ec2-user@ip-10-0-0-109 ~]$ # Obtain the security group Id
[ec2-user@ip-10-0-0-109 ~]$ SG=$(aws ec2 describe-security-groups --filters Name
[ec2-user@ip-10-0-0-109 ~]$ echo $SG
sg-0a369d9b7f1c11d7e
[ec2-user@ip-10-0-0-109 ~]$ # end
[ec2-user@ip-10-0-0-109 ~]$
```

45.**Download** a **User Data** script:

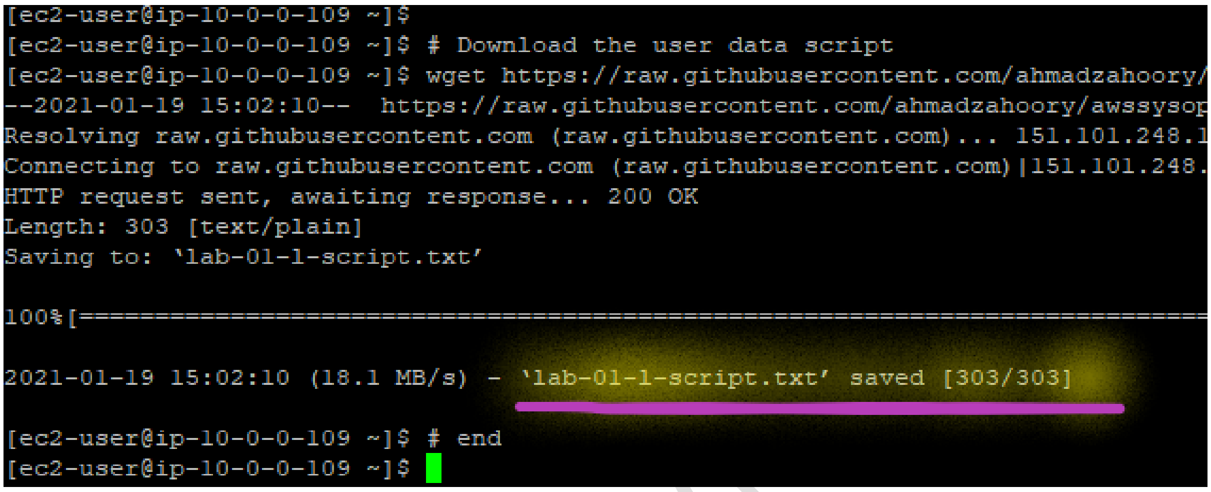
You will be launching an instance that will act as a Web Server. To install and configure the web server, you will provide a **User Data script** that will be automatically executed when the instance launches.

a. **Paste** the below **script** into **Linux terminal**:

```
# Download the user data script
wget https://raw.githubusercontent.com/ahmadzahoory/awssysops/master/lab-
01-l-script.txt
# end
```

Note: You can also copy the **Section 4** of the script from **LAB-01_Script Launch Web Server using CLI**.txt file provided with the LAB Manual.

Note: If ***command executed succesfully***, you can see the **File downloaded** in to linux server.



```
[ec2-user@ip-10-0-0-109 ~]$
[ec2-user@ip-10-0-0-109 ~]$ # Download the user data script
[ec2-user@ip-10-0-0-109 ~]$ wget https://raw.githubusercontent.com/ahmadzahoory/
--2021-01-19 15:02:10--  https://raw.githubusercontent.com/ahmadzahoory/awssysop
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 151.101.248.1
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|151.101.248.
HTTP request sent, awaiting response... 200 OK
Length: 303 [text/plain]
Saving to: 'lab-01-1-script.txt'

100%[=========================================================================

2021-01-19 15:02:10 (18.1 MB/s) - 'lab-01-1-script.txt' saved [303/303]

[ec2-user@ip-10-0-0-109 ~]$ # end
[ec2-user@ip-10-0-0-109 ~]$
```

46. **View** the **Contents** of the user data script:

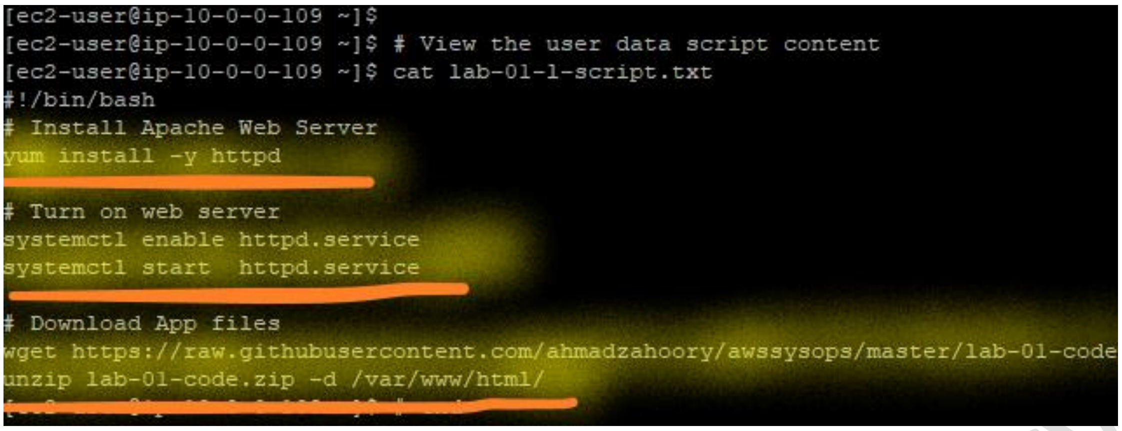
You can **view** the user data **script** contents, that you will be using to install and deploy the web server and web application.

a. **Paste** the below **script** into **Linux terminal**:

```
# View the user data script content
cat lab-01-l-script.txt
# end
```

Note: You can also copy the **Section 5** of the script from **LAB-01_Script Launch Web Server using CLI**.txt file provided with the LAB Manual.

Note: If ***command executed succesfully***, you can see the **User data script contents**.



```
[ec2-user@ip-10-0-0-109 ~]$
[ec2-user@ip-10-0-0-109 ~]$ # View the user data script content
[ec2-user@ip-10-0-0-109 ~]$ cat lab-01-l-script.txt
#!/bin/bash
# Install Apache Web Server
yum install -y httpd

# Turn on web server
systemctl enable httpd.service
systemctl start  httpd.service

# Download App files
wget https://raw.githubusercontent.com/ahmadzahoory/awssysops/master/lab-01-code
unzip lab-01-code.zip -d /var/www/html/
[illegible]
```

The script does the **following**:

- Installs a web server.
- Downloads a zip file containing the web application.
- Installs the web application.

47.**Launch** the **Instance**.

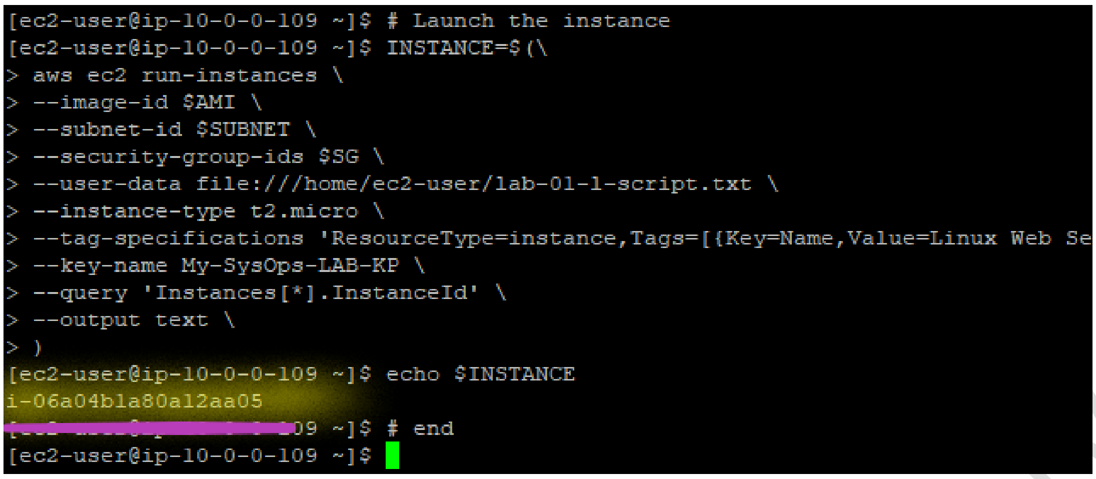
You now have all the necessary information require to launch the Web Server instance!

a. **Paste** the below **script** into **Linux terminal**:

```
# Launch the instance
INSTANCE=$(\
aws ec2 run-instances \
--image-id $AMI \
--subnet-id $SUBNET \
--security-group-ids $SG \
--user-data file:///home/ec2-user/lab-01-l-script.txt \
--instance-type t2.micro \
--tag-specifications 'ResourceType=instance,Tags=[{Key=Name,Value=Linux Web Server}]' \
--key-name My-AWS-LAB-KP \
--query 'Instances[*].InstanceId' \
--output text \
)
echo $INSTANCE
# end
```

Note: You can also copy the **Section 6** of the script from **LAB-01_Script Launch Web Server using CLI**.txt file provided with the LAB Manual.

Note: If ***command executed succesfully***, you can see the **Instance Id** in the response.



```
[ec2-user@ip-10-0-0-109 ~]$ # Launch the instance
[ec2-user@ip-10-0-0-109 ~]$ INSTANCE=$(\
> aws ec2 run-instances \
> --image-id $AMI \
> --subnet-id $SUBNET \
> --security-group-ids $SG \
> --user-data file:///home/ec2-user/lab-01-1-script.txt \
> --instance-type t2.micro \
> --tag-specifications 'ResourceType=instance,Tags=[{Key=Name,Value=Linux Web Se
> --key-name My-SysOps-LAB-KP \
> --query 'Instances[*].InstanceId' \
> --output text \
> )
[ec2-user@ip-10-0-0-109 ~]$ echo $INSTANCE
i-06a04b1a80a12aa05
[redacted]09 ~]$ # end
[ec2-user@ip-10-0-0-109 ~]$ 
```

The command launches a new instance (*run_instances*) **using** these **parameters**:

- **Image:** Uses the AMI value obtained earlier from the Parameter store.
- **Subnet:** Specifies the *Public Subnet* obtained earlier and, by association, the VPC in which to launch the instance.
- **Security Group:** Uses the *Web Security Group* obtained earlier, which permits HTTP access.
- **User Data:** References the User Data script you downloaded, which installs the web application.
- **Instance Type:** Specifies the type of instance to launch.
- **Tags:** Assigns a *Name* tag with the value of *Web Server*.

The **query** parameter specifies that the command should return the *Instance ID* once the instance is launched.

The **output** parameter specifies that the output of the command should be in *text*. Other output options are *json* and *table*.

The Id of the new instance has been stored in the *INSTANCE* environment variable.

48. **Describe** the **details** of the instance:

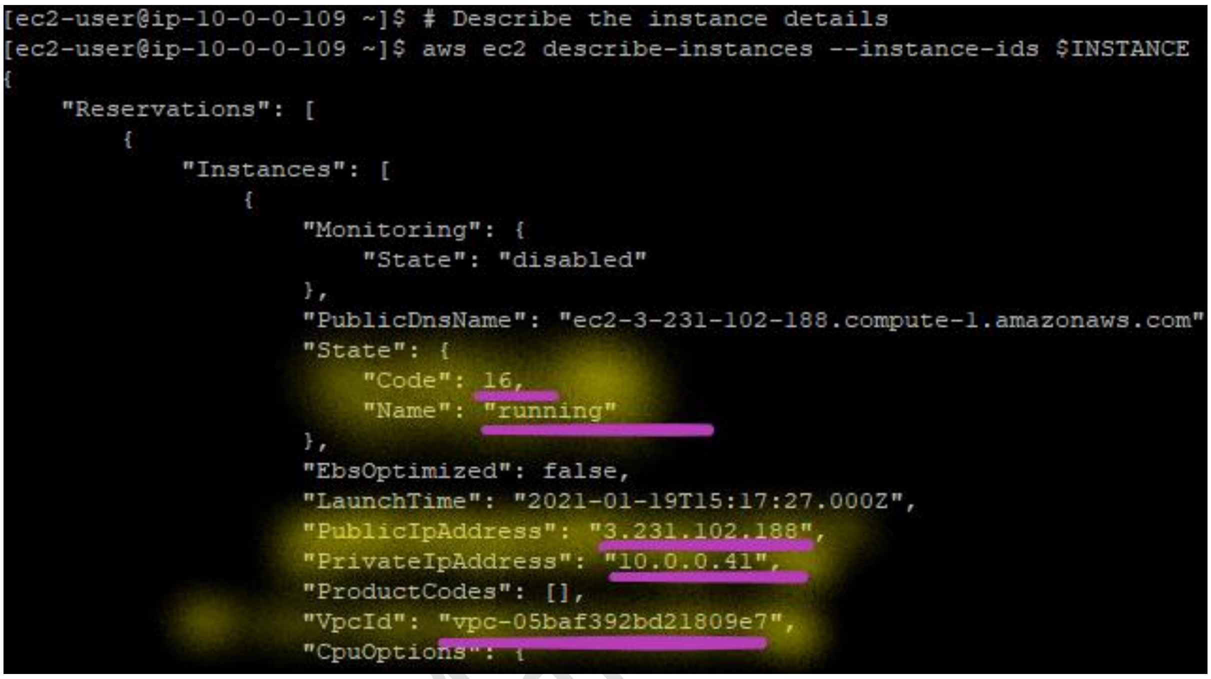
You can monitor the status of the instance via the Management Console, but you can also query the status via the AWS CLI.

a. **Paste** the below **script** into **Linux terminal**:

```
# Describe the instance details
aws ec2 describe-instances --instance-ids $INSTANCE
# end
```

Note: You can also copy the **Section 7** of the script from **LAB-01_Script Launch Web Server using CLI**.txt file provided with the LAB Manual.

Note: If ***command executed succesfully***, you can see the **Instance details** in the response.



```
[ec2-user@ip-10-0-0-109 ~]$ # Describe the instance details
[ec2-user@ip-10-0-0-109 ~]$ aws ec2 describe-instances --instance-ids $INSTANCE
{
    "Reservations": [
        {
            "Instances": [
                {
                    "Monitoring": {
                        "State": "disabled"
                    },
                    "PublicDnsName": "ec2-3-231-102-188.compute-1.amazonaws.com"
                    "State": {
                        "Code": 16,
                        "Name": "running"
                    },
                    "EbsOptimized": false,
                    "LaunchTime": "2021-01-19T15:17:27.000Z",
                    "PublicIpAddress": "3.231.102.188",
                    "PrivateIpAddress": "10.0.0.41",
                    "ProductCodes": [],
                    "VpcId": "vpc-05baf392bd21809e7",
                    "CpuOptions": {
```

All information related to the instance will be displayed in JSON format. Amongst this information is the instance status.

The values for instance-state-code will all be in the range of the:

- 0: pending
- 16: running
- 32: shutting-down
- 48: terminated
- 64: stopping
- 80: stopped

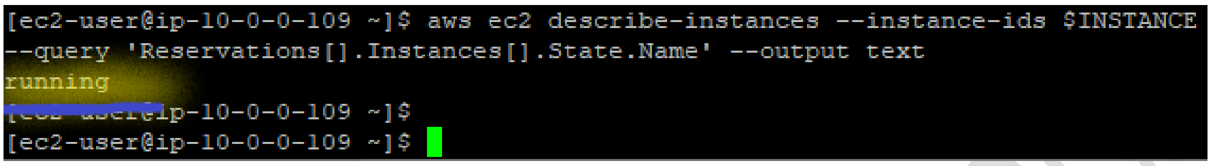
Specific information can be obtained by using the **query** parameter.

b. **Paste** the below **script** into **Linux terminal**:

```
# Describe the instance state
aws ec2 describe-instances --instance-ids $INSTANCE --query
'Reservations[].Instances[].State.Name' --output text
# end
```

Note: You can also copy the **Section 8** of the script from **LAB-01_Script Launch Web Server using CLI**.txt file provided with the LAB Manual.

Note: If ***command executed succesfully***, you can see the **Instance State** in the response.


```
[ec2-user@ip-10-0-0-109 ~]$ aws ec2 describe-instances --instance-ids $INSTANCE
--query 'Reservations[].Instances[].State.Name' --output text
running
[ec2-user@ip-10-0-0-109 ~]$
[ec2-user@ip-10-0-0-109 ~]$
```

This is the same command but, rather than displaying all information about the instance, only displays the name of the instance *State*.

This will display a status of **pending** or **running**.

Repeat the above command until it returns a status of **running**.

49. **Test** the **Web Server**

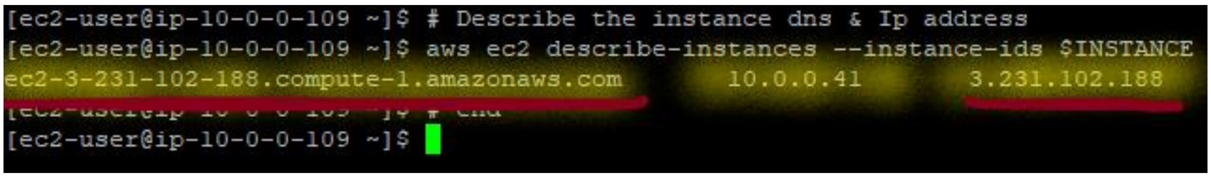
You can now test that the web server is working. You can retrieve a URL to the instance via the AWS CLI.

a. **Paste** the below **script** into **Linux terminal**:

```
# Describe the instance dns & Ip address
aws ec2 describe-instances --instance-ids $INSTANCE --query
Reservations[].Instances[].[PublicDnsName,PrivateIpAddress,PublicIpAddress] --
output text
# end
```

Note: You can also copy the **Section 9** of the script from **LAB-01_Script Launch Web Server using CLI**.txt file provided with the LAB Manual.

Note: If ***command executed succesfully***, you can see the **Instance DNS name**, **Private** and **Public IP address** in the response.


```
[ec2-user@ip-10-0-0-109 ~]$ # Describe the instance dns & Ip address
[ec2-user@ip-10-0-0-109 ~]$ aws ec2 describe-instances --instance-ids $INSTANCE
ec2-3-231-102-188.compute-1.amazonaws.com       10.0.0.41       3.231.102.188
[ec2-user@ip-10-0-0-109 ~]$ # end
[ec2-user@ip-10-0-0-109 ~]$
```

This returns the **DNS Name, Private and Public** IP Address of the instance.

50. **Copy** the **DNS name** that is displayed.

It should look similar to: *ec2-35-11-22-33.us-west-2.compute.amazonaws.com*

Task 7: Access a Web Server

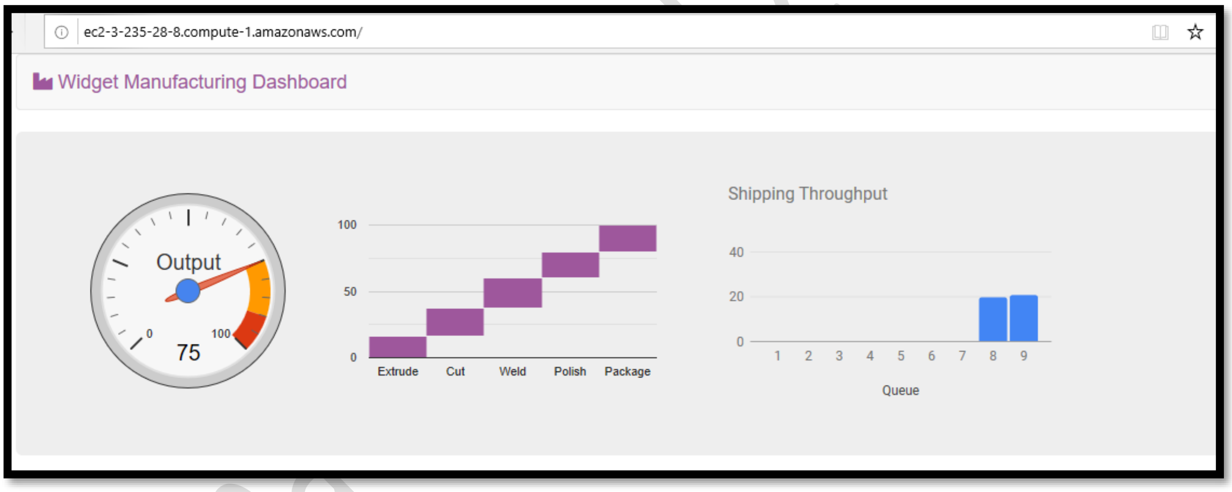
The new instance will be configured as a Web Server. In this task, you will access the web server.

Step 1: Access the Web server

51. **Paste** the **DNS name** into a new ***web browser*** tab, then press Enter.

A web page should be displayed, demonstrating that the web server was successfully launched and configured.

You can also see the instance in the EC2 management console.



Note

- Do not delete any resources you deployed in this lab.
- You will be using them in the next lab.