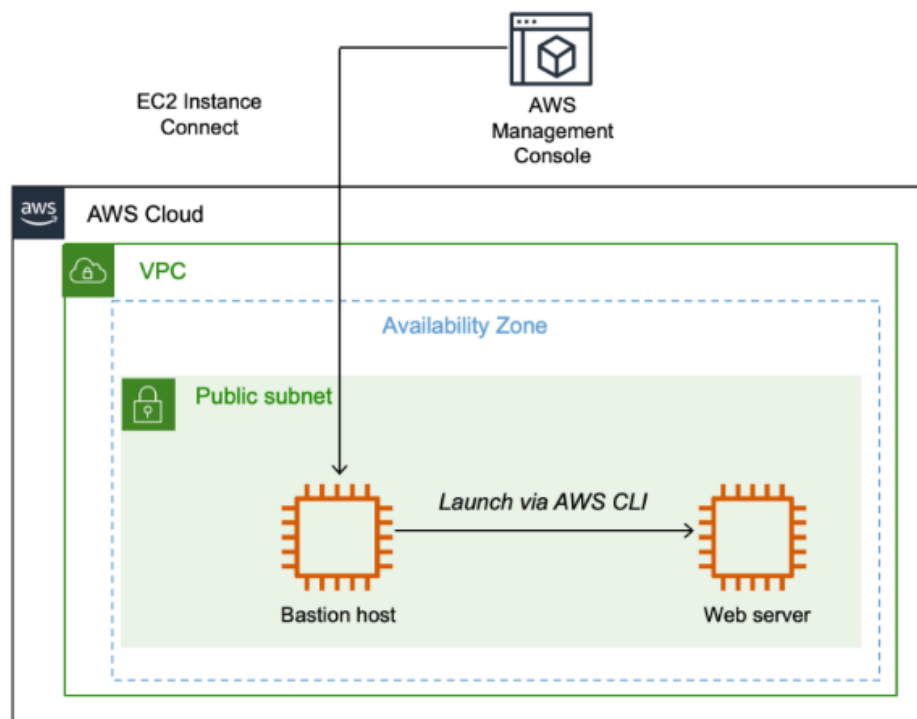


## 171-[JAWS]-Lab - Creating Amazon EC2 Instances

Umi Nur F | [nurfatih365@gmail.com](mailto:nurfatih365@gmail.com)

### A. Lab overview

AWS provides multiple ways to launch Amazon Elastic Compute Cloud (Amazon EC2) instance. In this lab, you use the AWS Management Console to launch an EC2 instance and then use it as a bastion host to launch another EC2 instance, which will be a web server. You use EC2 Instance Connect to securely connect to the bastion host and use the AWS Command Line Interface (AWS CLI) to launch a web server instance. The following diagram illustrates the final architecture that you will build:



If you have time, you can follow the steps in the optional challenge sections to troubleshoot some issues with EC2 instances.

After completing this lab, you should be able to do the following:

- Launch an EC2 instance by using the AWS Management Console.
- Connect to the EC2 instance by using EC2 Instance Connect.

- Launch an EC2 instance by using the AWS CLI.

This lab will require approximately 45 minutes to complete.

### Accessing the AWS Management Console

1. At the top of these instructions, choose **Start Lab** to launch the lab.

A **Start Lab** panel opens displaying the lab status.

2. Wait until the message "Lab status: ready" appears, and then choose **X** to close the **Start Lab** panel.

3. Next to **Start Lab**, choose **AWS** to open the AWS Management Console on a new browser tab. The system automatically signs you in.

**Tip** If a new browser tab does not open, a banner or icon at the top of your browser will indicate that your browser is preventing the site from opening pop-up windows. Choose the banner or icon, and choose **Allow pop-ups**.

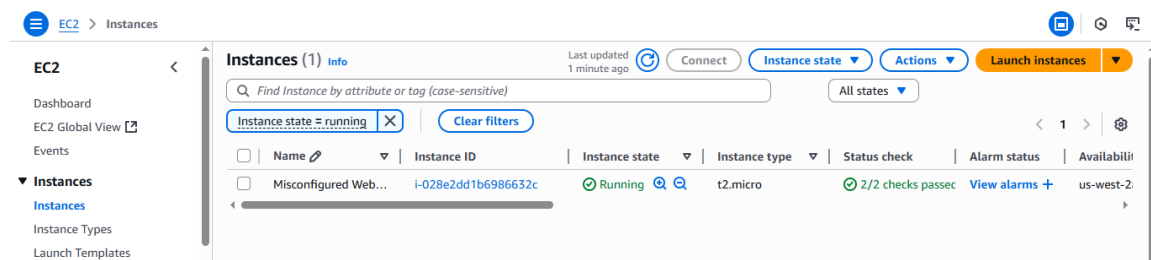
4. Arrange the AWS Management Console so that it appears alongside these instructions.

**Important:** Do not change the lab Region unless specifically instructed to do so.

### Task 1: Launching an EC2 Instance by using the AWS Management Console

In this task, you launch an EC2 instance by using the AWS Management Console. The instance will be a bastion host from which you can use the AWS CLI.

5. On the **AWS Management Console**, in the **Search** bar, enter and choose **EC2** to open the **Amazon EC2 Management Console**.
6. From the **Launch instance** dropdown list, choose **Launch instance** to open the **Launch an instance** menu.



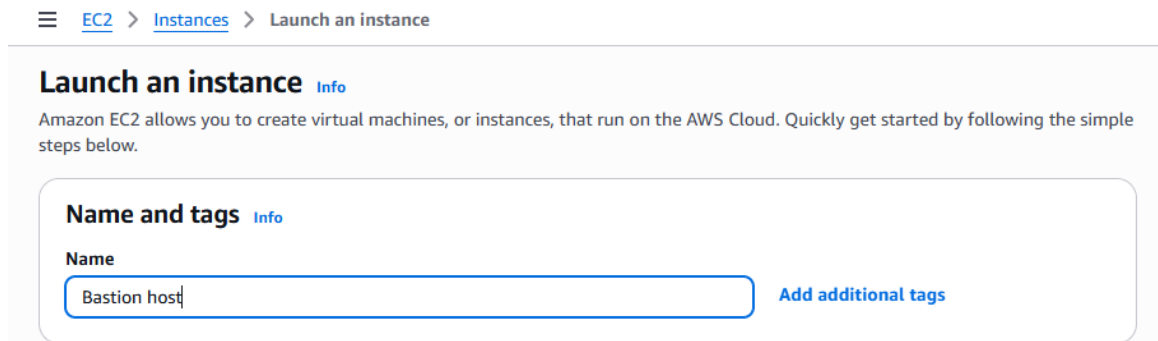
### Step 1: Choose name and tags

You use tags to categorize your AWS resources in different ways, such as by purpose, owner, or environment. This categorization is useful when you have many resources of the same

type; you can quickly identify a specific resource by its tags. Each tag consists of a *key* and a *value*, both of which you define.

When you name your instance, AWS creates a key-value pair. The key for this pair is **Name**, and the value is the name that you enter for your EC2 instance.

7. In the **Name and tags** section, for **Name**, enter `Bastion host`



☰ [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

 [Add additional tags](#)

## Step 2: Choose an AMI

In this step, you choose an Amazon Machine Image (AMI). An AMI includes the following:

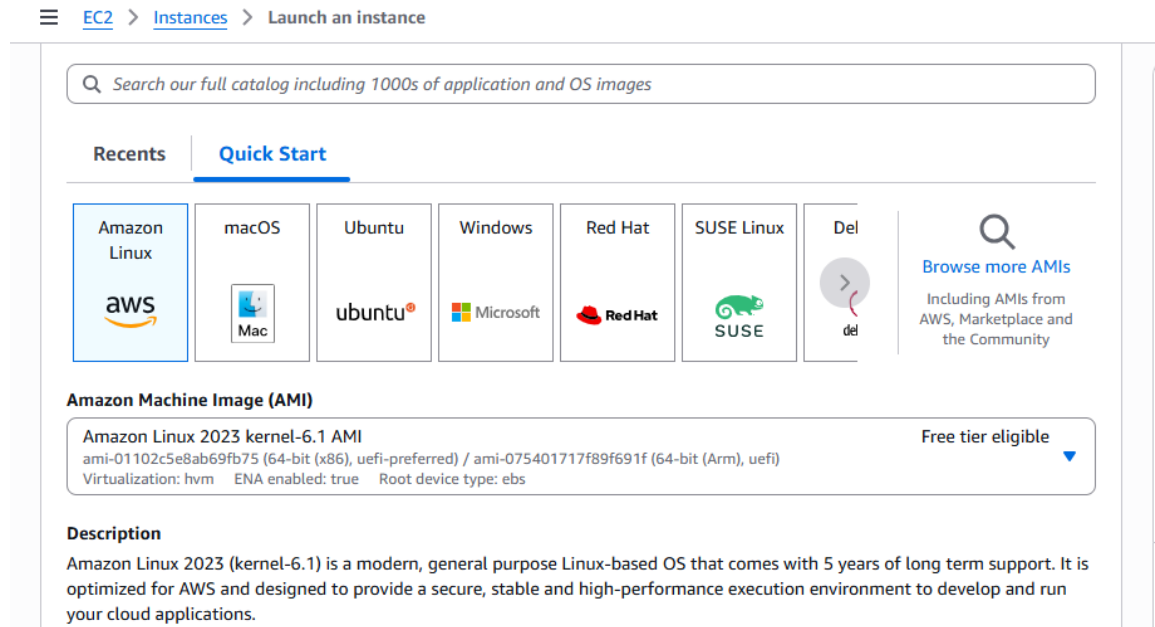
- A template for the root volume for the instance (for example, an operating system or an application server with applications)
- Launch permissions that control which AWS accounts can use the AMI to launch instances
- A block device mapping that specifies the volumes to attach to the instance when it is launched

The **Quick Start** list contains the most commonly used AMIs. You can also create your own AMI or select an AMI from the AWS Marketplace, an online store where you can sell or buy software that runs on AWS.

Your bastion host will use Amazon Linux 2.

8. In the **Application and OS Images (Amazon Machine Image)** section, for **Quick Start**, confirm that **Amazon Linux** is selected. Keep this selection.

**Note:** This option corresponds to Amazon Linux 2 AMI (HVM) as indicated in the **Description**.



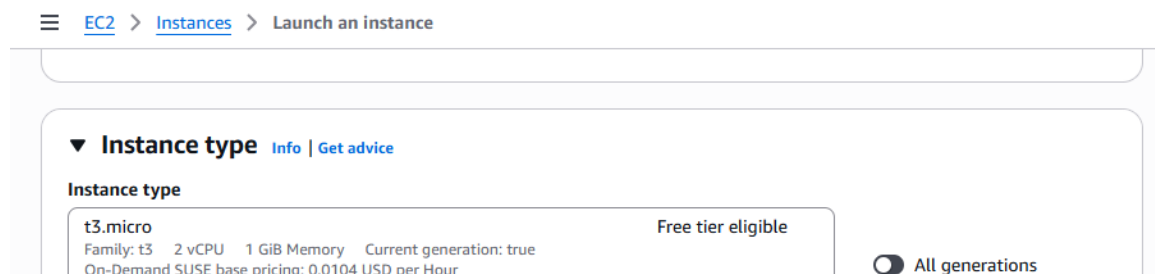
### Step 3: Choose an instance type

In this step, you choose an instance type, which determines the resources that will be allocated to your EC2 instance. Each instance type allocates a combination of virtual CPU, memory, disk storage, and network performance.

Instance types are divided into families, such as compute optimized, memory optimized, and storage optimized. The name of the instance type includes a family identifier, such as T3 and M5. The number indicates the generation of the instance, so M5 is newer than M4.

Your application uses a t3.micro instance type, which is a small instance that can burst above baseline performance when it is busy. It is suitable for development and testing purposes and for applications that have bursty workloads.

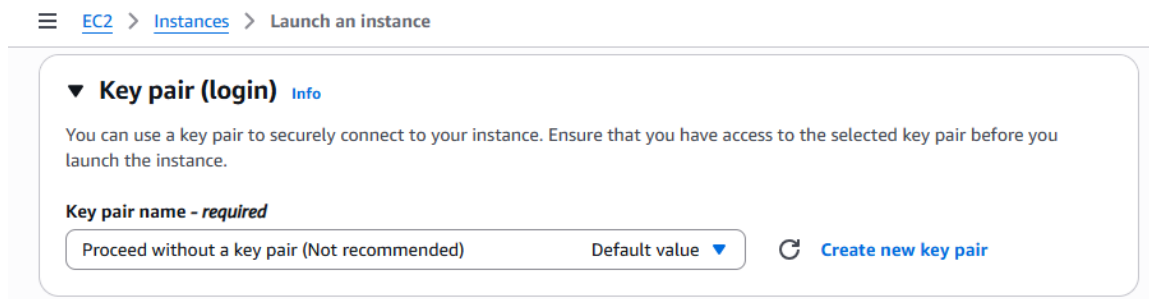
9. From the **Instance type** dropdown list, search for and choose **t3.micro**.



#### Step 4: Configure a key pair

Amazon EC2 uses public key cryptography to encrypt and decrypt login information. To log in to your instance, you must create a key pair, specify the name of the key pair when you launch the instance, and provide the private key when you connect to the instance. In this lab, you use EC2 Instance Connect to log in to your instance, so you do not require a key pair.

10. In the **Key pair (login)** section, from the **Key pair name - required** dropdown list, choose **Proceed without key pair (Not recommended)**.



#### Step 5: Configure the network settings

You use this pane to configure networking settings. The virtual private cloud (VPC) indicates which VPC you want to launch the instance into. You can have multiple VPCs, including different ones for development, testing, and production. You launch the instance in a public subnet within the Lab VPC network.

11. In the **Network settings** section, choose **Edit**.
12. From the **VPC - required** dropdown list, choose **Lab VPC**.  
The Lab VPC was created using an AWS CloudFormation template during the setup process of your lab. This VPC includes one public subnet.
13. In the **Subnet** dropdown list, notice that **Public Subnet** is selected by default. Keep this default setting.
14. In the **Auto-assign public IP** dropdown list, notice that **Enable** is selected by default. Keep this default setting.

☰ [EC2](#) > [Instances](#) > Launch an instance

▼ **Network settings** [Info](#)

**VPC - required** [Info](#)

vpc-0931400a9c5b73e97 (Lab VPC)  
10.0.0.0/16

**Subnet** [Info](#)

subnet-00d99da2d17259352  
VPC: vpc-0931400a9c5b73e97   Owner: 422127550171  
Availability Zone: us-west-2a (usw2-az1)   Zone type: Availability Zone  
IP addresses available: 250   CIDR: 10.0.0.0/24

Public Subnet

[Create new subnet](#)

**Auto-assign public IP** [Info](#)

Enable

15. In the **Firewall (security groups)** section, notice that **Create security group** is selected.

Configure the following options:

- For **Security group name - required** enter Bastion security group
- For **Description - required** enter Permit SSH connections

A security group acts as a virtual firewall that controls the traffic for one or more instances. When you launch an instance, you associate one or more security groups with the instance. You add rules to each security group that allow traffic to or from its associated instances. You can modify the rules for a security group at any time; the new rules are automatically applied to all instances that are associated with the security group.

☰ [EC2](#) > [Instances](#) > Launch an instance

☒ Create security group
☐ Select existing security group

**Security group name - required**

Bastion security group

This security group will be added to all network interfaces. The name can't be edited after the security group is created. Max length is 255 characters. Valid characters: a-z, A-Z, 0-9, spaces, and .\_-:/!@#&\*~[]+=&{}!\$\*

**Description - required** [Info](#)

Permit SSH connections

## Step 6: Add storage

You can use this step to add additional Amazon Elastic Block Store (Amazon EBS) disk volumes and configure their size and performance. You launch the EC2 instance using a default 8 gibibyte (GiB) disk volume. This is your root volume (also known as a boot volume).

16. In the **Configure storage** pane, keep the default storage configuration.

☰ [EC2](#) > [Instances](#) > Launch an instance

▶ **Advanced network configuration**

▼ **Configure storage** [Info](#) Advanced

1x  GiB  ▼ Root volume, 3000 IOPS, Not encrypted

[Add new volume](#)

---

🕒 Click refresh to view backup information  
The tags that you assign determine whether the instance will be backed up by any Data Lifecycle Manager policies. [↻](#)

---

0 x File systems Edit

## Step 7: Configure advanced details

17. Expand the **Advanced details** pane.

18. From **IAM instance profile** dropdown list, choose **Bastion-Role**.

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

19. Leave the default settings for all the other values.

☰ [EC2](#) > [Instances](#) > Launch an instance

▼ **Advanced details** [Info](#)

**Domain join directory** | [Info](#)

▼ [↻](#) [Create new directory](#) [↗](#)

**IAM instance profile** | [Info](#)

▼ [↻](#) [Create new IAM profile](#) [↗](#)

arn:aws:iam::422127550171:instance-profile/Bastion-Role

## Step 8: Launch an EC2 instance

Now that you have configured your EC2 instance settings, it is time to launch your instance.

20. In the **Summary** section, review the instance configuration details displayed, and choose **Launch instance**.

21. Choose **View all instances**.

EC2 > Instances > Launch an instance

### Advanced details

**Domain join directory** | Info

Select [Create new directory](#)

**IAM instance profile** | Info

Bastion-Role [Create new IAM profile](#)

**Hostname type** | Info

IP name

**DNS Hostname** | Info

☒ Enable IP name IPv4 (A record) DNS requests

☐ Enable resource-based IPv4 (A record) DNS requests

☐ Enable resource-based IPv6 (AAAA record) DNS requests

**Instance auto-recovery** | Info

Select

### Summary

**Number of instances** | Info

1

**Software Image (AMI)**

Amazon Linux 2023 AMI 2023.8.2...[read more](#)

ami-01102c5e8ab69fb75

**Virtual server type (instance type)**

t3.micro

**Firewall (security group)**

New security group

**Storage (volumes)**

1 volume(s) - 8 GiB

[Cancel](#) [Launch instance](#)

EC2 > Instances

Instances (2) Info

Last updated less than a minute ago [Connect](#) [Instance state](#) [Actions](#) [Launch instances](#)

Find Instance by attribute or tag (case-sensitive) All states

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability
<input type="checkbox"/>	Misconfigured Web...	i-028e2dd1b6986632c	Running	t2.micro	2/2 checks passed	<a href="#">View alarms</a>	us-west-2
<input type="checkbox"/>	Bastion host	i-04c3a2a8426f5b853	Running	t3.micro	Initializing	<a href="#">View alarms</a>	us-west-2

## Task 2: Logging in to the bastion host

In this task, you use EC2 Instance Connect to log in to the bastion host that you just created.

22. On the **EC2 Management Console**, from the list of EC2 instances displayed, choose the check box for the **bastion host** instance.

23. Choose **Connect**.

24. On the **EC2 Instance Connect** tab, choose **Connect** to connect to the bastion host.

**Note:** If you prefer to use an SSH client to connect to the EC2 instance, see the guidance to [Connect to your Linux instance](#).

Now that you are connected to the bastion host, you can use the AWS CLI to call AWS services.





You use AWS Systems Manager Parameter Store to retrieve the ID of the most recent Amazon Linux 2 AMI. AWS maintains a list of standard AMIs in Parameter Store, which makes it possible to automate this task.

25. Run the following script in your EC2 Instance Connect session:

```
#Set the Region

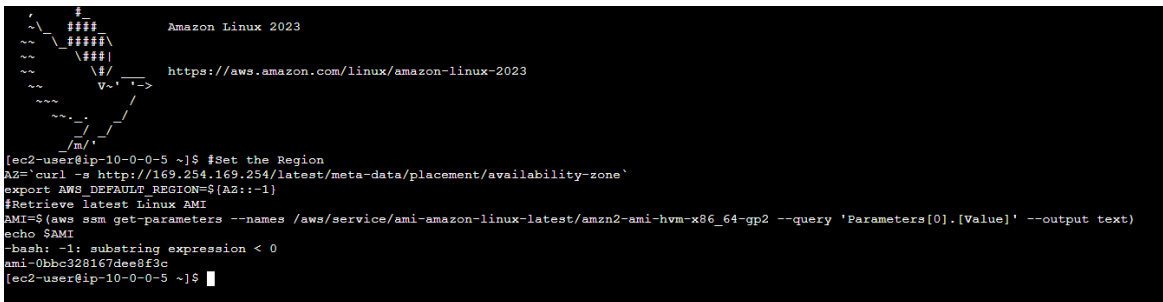
AZ=`curl -s http://169.254.169.254/latest/meta-data/placement/availability-zone`

export AWS_DEFAULT_REGION=${AZ::-1}

#Retrieve 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
```



```
Amazon Linux 2023
https://aws.amazon.com/linux/amazon-linux-2023

[ec2-user@ip-10-0-0-5 ~]$ #Set the Region
AZ=`curl -s http://169.254.169.254/latest/meta-data/placement/availability-zone`
export AWS_DEFAULT_REGION=${AZ::-1}
#Retrieve latest Linux AMI
AMI=$(aws ssm get-parameters --names /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2 --query 'Parameters[0].[Value]' --output text)
echo $AMI
-bash: -1: substring expression < 0
ami-0bdc328167dee8f3c
[ec2-user@ip-10-0-0-5 ~]$
```

### Script explanation

- The script retrieves the Availability Zone for the running instance using instance metadata.
- The script retrieves the Region from the Availability Zone and exports it into the environment for subsequent use.
- The script calls the AWS Systems Manager (indicated with the ssm command) and uses the get-parameters command to retrieve the AMI ID from Parameter Store.
- The AMI requested was for Amazon Linux 2 (which is the same as the one used for bastion host).
- The AMI ID has been stored in an environment variable called AMI.

If your EC2 Instance Connect session disconnects, it will lose the information stored in the environment variables. Refresh your browser to reconnect. If you do so, you will need to

re-run all of the steps in this task, starting with the commands in this step, to obtain the AMI ID.

### Step 2: Retrieve the subnet to use

You launch the new instance in the public subnet. When launching an instance, you can specify the subnet ID.

26. To retrieve the subnet ID for the public subnet, run the following command:

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

This script runs the `aws ec2` command with the `describe-subnets` subcommand to retrieve the subnet ID of the subnet named **Public Subnet**.

```
[ec2-user@ip-10-0-0-5 ~]$ SUBNET=$(aws ec2 describe-subnets --filters 'Name=tag:Name,Values=Public Subnet' --query Subnets[].SubnetId --output text)
echo $SUBNET
subnet-00d99da2d17259352
[ec2-user@ip-10-0-0-5 ~]$
```

```
[ec2-user@ip-10-0-0-5 ~]$ SUBNET=$(aws e
echo $SUBNET
subnet-00d99da2d17259352
[ec2-user@ip-10-0-0-5 ~]$
```

### Step 3: Retrieve the security group to use

This lab includes a web security group, which allows inbound HTTP requests.

27. Run the following command:

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

The script runs the `aws ec2` command with the `describe-security-groups` subcommand to retrieve the security group ID of the web security group.

```
[ec2-user@ip-10-0-0-5 ~]$ SG=$(aws ec2 describe-security-groups --filters Name=group-name,Values=WebSecurityGroup --query SecurityGroups[].GroupId --o
utput text)
echo $SG
sg-09d1017298fa91282
[ec2-user@ip-10-0-0-5 ~]$
```

```
[ec2-user@ip-10-0-0-5 ~]$ SG=$(aws ec2 describe-se
utput text)
echo $SG
sg-09d1017298fa91282
[ec2-user@ip-10-0-0-5 ~]$
```

#### Step 4: Download a user data script

In this step, you launch an instance that acts as a web server. To install and configure the web server, you provide a user data script that automatically runs when the instance launches.

28. To download the user data script, run the following command:

```
wget https://aws-tc-largeobjects.s3.us-west-2.amazonaws.com/CUR-TF-100-RSJAWS-1-23732/171-lab-JAWS-create-ec2/s3/UserData.txt
```

```
[ec2-user@ip-10-0-0-5 ~]$ wget https://aws-tc-largeobjects.s3.us-west-2.amazonaws.com/CUR-TF-100-RSJAWS-1-23732/171-lab-JAWS-create-ec2/s3/UserData.txt
--2025-09-02 07:17:01-- https://aws-tc-largeobjects.s3.us-west-2.amazonaws.com/CUR-TF-100-RSJAWS-1-23732/171-lab-JAWS-create-ec2/s3/UserData.txt
Resolving aws-tc-largeobjects.s3.us-west-2.amazonaws.com (aws-tc-largeobjects.s3.us-west-2.amazonaws.com)... 52.92.194.98, 3.5.83.241, 3.5.76.73, ...
Connecting to aws-tc-largeobjects.s3.us-west-2.amazonaws.com (aws-tc-largeobjects.s3.us-west-2.amazonaws.com)|52.92.194.98|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 327 [text/plain]
Saving to: 'UserData.txt'

UserData.txt          100%[=====] 327 --.-KB/s  in 0s

2025-09-02 07:17:02 (1.97 MB/s) - 'UserData.txt' saved [327/327]

[ec2-user@ip-10-0-0-5 ~]$
```

```
Length: 327 [text/plain]
Saving to: 'UserData.txt'

UserData.txt          100%[=====]

2025-09-02 07:17:02 (1.97 MB/s) - 'UserData.txt' saved [327/327]

[ec2-user@ip-10-0-0-5 ~]$
```

29. To view the contents of the script, run the following command:

```
cat UserData.txt
```

The script does the following:

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

```
[ec2-user@ip-10-0-0-5 ~]$ cat UserData.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://aws-tc-largeobjects.s3.amazonaws.com/CUR-TF-100-RESTRT-1/171-lab-$5BJAWS$5D-create-ec2/dashboard-app.zip
unzip dashboard-app.zip -d /var/www/html/
[ec2-user@ip-10-0-0-5 ~]$
```

#### Step 5: Launch the instance

You now have all the necessary information required to launch the web server instance.

30. Run the following command:

```
INSTANCE=$(  
aws ec2 run-instances \
```

```
--image-id $AMI \  
--subnet-id $SUBNET \  
--security-group-ids $SG \  
--user-data file:///home/ec2-user/UserData.txt \  
--instance-type t3.micro \  
--tag-specifications 'ResourceType=instance,Tags=[{Key=Name,Value=Web Server}]' \  
--query 'Instances[*].InstanceId' \  
--output text \  
)  
echo $INSTANCE
```

The run-instances command launches a new instance using these parameters:

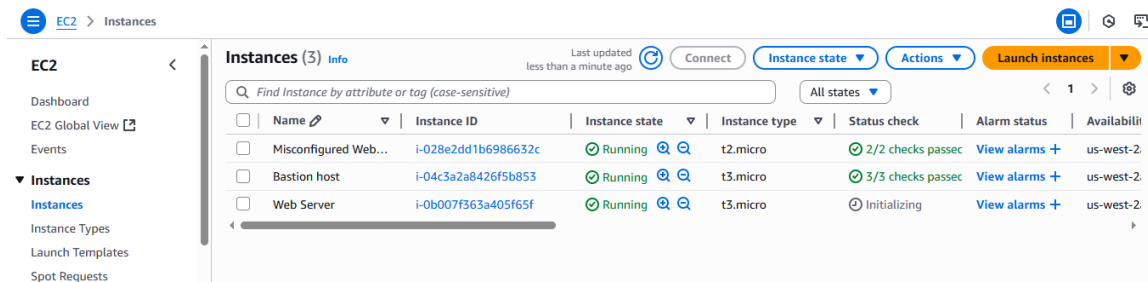
- **Image:** Uses the AMI value obtained earlier from Parameter Store
- **Subnet:** Specifies the public subnet retrieved earlier and, by association, the VPC in which to launch the instance
- **Security group:** Uses the web security group retrieved earlier, which permits HTTP access
- **User data:** References the user data script that 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.

**Note:** The ID of the new instance has been stored in the INSTANCE environment variable.

```
[ec2-user@ip-10-0-0-5 ~]$ INSTANCE=$(
aws ec2 run-instances \
--image-id $AMI \
--subnet-id $SUBNET \
--security-group-ids $SG \
--user-data file:///home/ec2-user/UserData.txt \
--instance-type t3.micro \
--tag-specifications 'ResourceType=instance,Tags=[{Key=Name,Value=Web Server}]' \
--query 'Instances[*].InstanceId' \
--output text \
)
echo $INSTANCE
i-0b007f363a405f65f
[ec2-user@ip-10-0-0-5 ~]$
```



Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability
Misconfigured Web...	i-028e2dd1b6986632c	Running	t2.micro	2/2 checks passed	View alarms +	us-west-2
Bastion host	i-04c3a2a8426f5b853	Running	t3.micro	3/3 checks passed	View alarms +	us-west-2
Web Server	i-0b007f363a405f65f	Running	t3.micro	3/3 checks passed	View alarms +	us-west-2

## Step 6: Wait for the instance to be ready

You can monitor the status of the instance by using the AWS Management Console or by querying the status by using the AWS CLI.

31. Run the following command:

```
aws ec2 describe-instances --instance-ids $INSTANCE
```

All information related to the instance is displayed in JSON format. This information includes the instance status.

You can retrieve specific information by using the query parameter.

```
{
  "Reservations": [
    {
      "ReservationId": "r-00d4443455014c86c",
      "OwnerId": "422127550171",
      "Groups": [],
      "Instances": [
        {
          "Architecture": "x86_64",
          "BlockDeviceMappings": [
            {
              "DeviceName": "/dev/xvda",
              "Ebs": {
                "AttachTime": "2025-09-02T07:19:16+00:00",
                "DeleteOnTermination": true,
                "Status": "attached",
                "VolumeId": "vol-086481613b126078c"
              }
            }
          ],
          "ClientToken": "adfa893e-0943-4929-b3eb-e9ca5a87ec51",
          "EbsOptimized": false,
          "EnaSupport": true,

```

32. Run the following command:

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

This command is the same as the command in the previous step, but rather than displaying all information about the instance, this command displays only the name of the instance state. This command displays a status of *pending* or *running*. Run this command again until it returns a status of *running*.

```
[ec2-user@ip-10-0-0-5 ~]$ aws ec2 describe-instances --instance-ids $INSTANCE --query 'Reservations[].Instances[].State.Name' --output text  
running  
[ec2-user@ip-10-0-0-5 ~]$
```

### Step 7: Test the web server

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

33. Run the following command:

```
aws    ec2    describe-instances    --instance-ids    $INSTANCE    --query  
Reservations[].Instances[].PublicDnsName --output text
```

This command returns the public IPv4 Domain Name System (DNS) name of the instance.

```
[ec2-user@ip-10-0-0-5 ~]$  
[ec2-user@ip-10-0-0-5 ~]$ aws ec2 describe-instances --instance-ids $INSTANCE --query 'Reservations[].Instances[].State.Name' --output text  
running  
[ec2-user@ip-10-0-0-5 ~]$ aws ec2 describe-instances --instance-ids $INSTANCE --query Reservations[].Instances[].PublicDnsName --output text  
ec2-35-91-117-160.us-west-2.compute.amazonaws.com  
[ec2-user@ip-10-0-0-5 ~]$
```

34. Copy the DNS name that is displayed. It should be similar to the following:

```
ec2-35-11-22-33.us-west-2.compute.amazonaws.com  
ec2-35-91-117-160.us-west-2.compute.amazonaws.com
```

35. Paste the DNS name into a new web browser tab, and then press Enter.

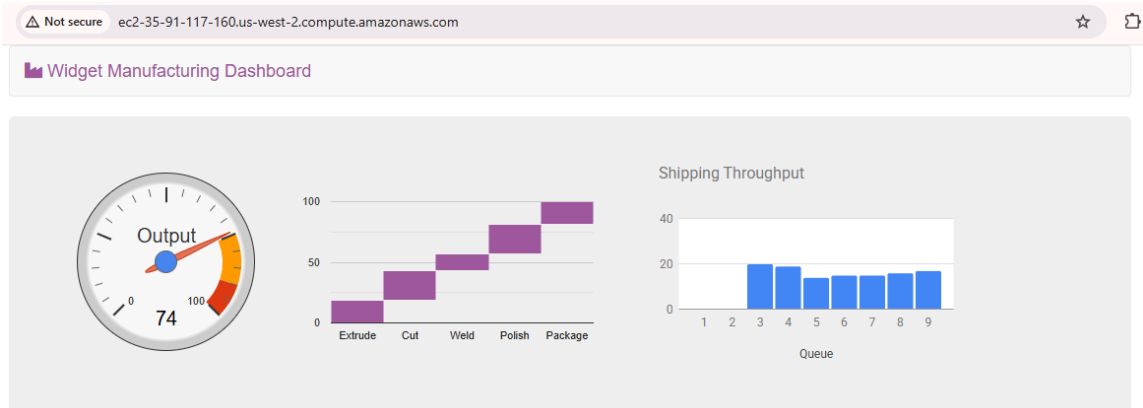
A web page should be displayed, which demonstrates that the web server was successfully launched and configured.

You can also see the instance on the Amazon EC2 management console.

36. Return to the web browser tab containing the Amazon EC2 management console. In the left navigation pane, choose **Instances**, and choose refresh.

The list should now include the **Web Server** instance that you launched by using the CLI command.

As you see in this task, the AWS CLI makes it possible to programmatically access and control AWS services. You can place these commands in a script and run them as a standard process to deploy consistent, reliable infrastructure with minimal scope for human error.



EC2 > Instances

Instances (3) Info

Last updated less than a minute ago

Find Instance by attribute or tag (case-sensitive)

All states

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability
<input type="checkbox"/>	Misconfigured Web...	i-028e2dd1b6986632c	Running	t2.micro	2/2 checks passed	View alarms +	us-west-2
<input type="checkbox"/>	Bastion host	i-04c3a2a8426f5b853	Running	t3.micro	3/3 checks passed	View alarms +	us-west-2
<input type="checkbox"/>	Web Server	i-0b007f363a405f65f	Running	t3.micro	3/3 checks passed	View alarms +	us-west-2

### Which method should you use?

- Launch from the management console when you quickly need to launch a one-off or temporary instance.
- Launch by using a script when you need to automate the creation of an instance in a repeatable, reliable manner.
- Launch by using CloudFormation when you want to launch related resources together.

### Optional challenge 1: Connect to an EC2 instance

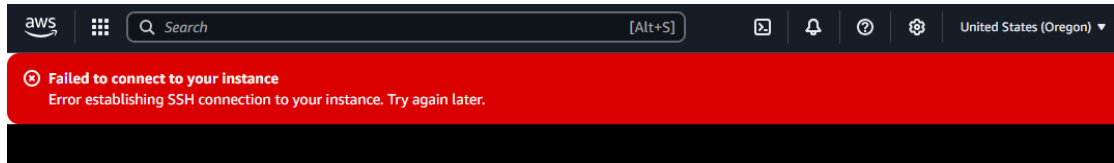
This challenge is optional and is provided in case you have lab time remaining.

In this challenge, you troubleshoot the security configuration of an instance called **Misconfigured Web Server**.

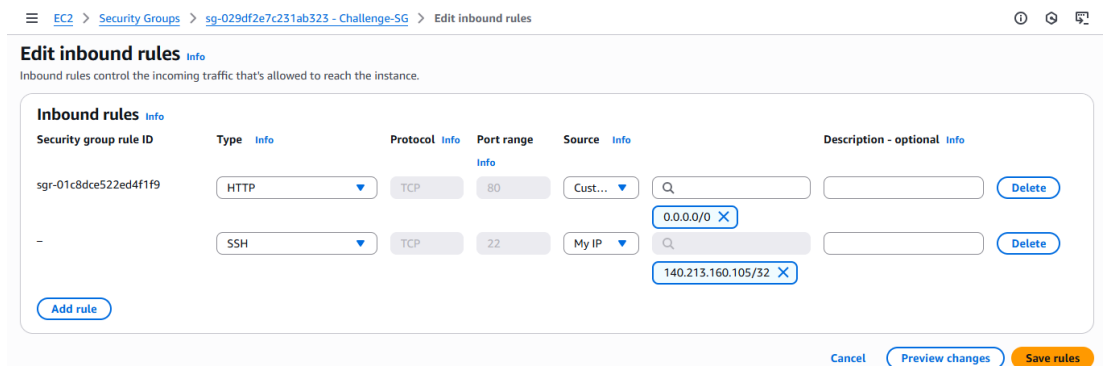
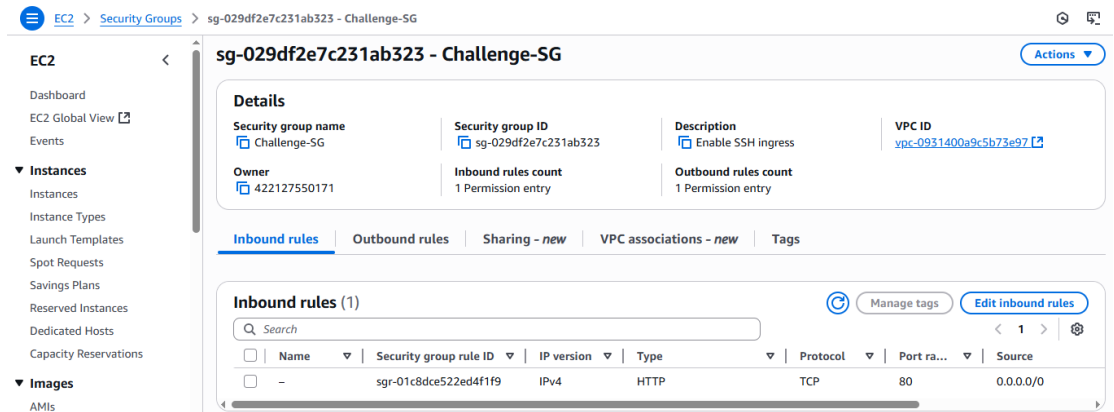
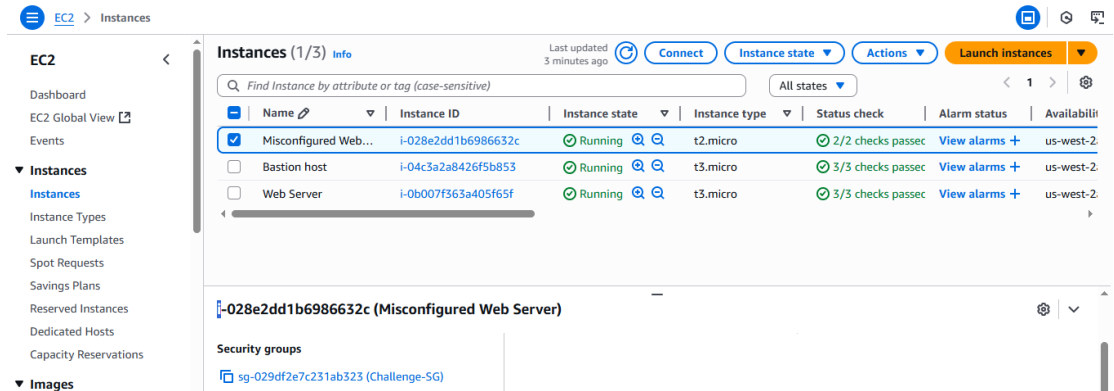
37. The following are your tasks:

- Try to connect to the **Misconfigured Web Server** instance by using EC2 Instance Connect.





- Diagnose why this does not work, and fix the misconfiguration.



EC2

Dashboard

EC2 Global View

Events

Instances

Instance Types

Launch Templates

Spot Requests

Savings Plans

Reserved Instances

Dedicated Hosts

Capacity Reservations

Images

AMIs

AMI Catalog

EC2 > Security Groups > sg-029df2e7c231ab323 - Challenge-SG

Inbound security group rules successfully modified on security group (sg-029df2e7c231ab323 | Challenge-SG)

Details

Security group name

Challenge-SG

Security group ID

sg-029df2e7c231ab323

Description

Enable SSH ingress

VPC ID

vpc-0931400a9c5b73e97

Owner

422127550171

Inbound rules count

2 Permission entries

Outbound rules count

1 Permission entry

Inbound rules

Outbound rules

Sharing - new

VPC associations - new

Tags

Inbound rules (2)

Manage tags

Edit inbound rules

Search

	Name	Security group rule ID	IP version	Type	Protocol	Port range	Source
<input type="checkbox"/>	-	sgr-01c8dce522ed4f1f9	IPv4	HTTP	TCP	80	0.0.0.0/0
<input type="checkbox"/>	-	sgr-03eee672a8452aa6e	IPv4	SSH	TCP	22	140.213.160.105/...

aws

Search

[Alt+S]

United States (Oregon)

Failed to connect to your instance

Error establishing SSH connection to your instance. Try again later.

EC2 > Security Groups > sg-029df2e7c231ab323 - Challenge-SG > Edit inbound rules

Edit inbound rules

Inbound rules control the incoming traffic that's allowed to reach the instance.

Inbound rules

Security group rule ID

Type

Protocol

Port range

Source

Description - optional

sgr-01c8dce522ed4f1f9

HTTP

TCP

80

Cust...

0.0.0.0/0

Delete

sgr-03eee672a8452aa6e

SSH

TCP

22

Any...

0.0.0.0/0

Delete

Add rule

Cancel

Preview changes

Save rules

Inbound rules

Outbound rules

Sharing - new

VPC associations - new

Tags

Inbound rules (2)

Manage tags

Edit inbound rules

Search

	Name	Security group rule ID	IP version	Type	Protocol	Port range	Source
<input type="checkbox"/>	-	sgr-03eee672a8452aa6e	IPv4	SSH	TCP	22	0.0.0.0/0
<input type="checkbox"/>	-	sgr-01c8dce522ed4f1f9	IPv4	HTTP	TCP	80	0.0.0.0/0

```
aws [Search] [Alt+S] United States (Oregon)

Amazon Linux 2
AL2 End of Life is 2026-06-30.
A newer version of Amazon Linux is available!
Amazon Linux 2023, GA and supported until 2028-03-15.
https://aws.amazon.com/linux/amazon-linux-2023/

[ec2-user@ip-10-0-0-62 ~]$
```

At the end of the lab, your instructor will ask you the following questions:

- What was the problem?

The Network ACL was too restrictive and blocked port 22. The security group of the Misconfigured Web Server did not have an inbound rule for SSH (port 22), so I could not connect to the instance.

- What did you do to fix the problem?

I edited the security group and added an inbound rule to allow SSH (port 22) from my IP address.

## Optional challenge 2: Fix the web server installation

In this challenge, you troubleshoot the web server installation on the **Misconfigured Web Server** instance:

38. The following are your tasks:

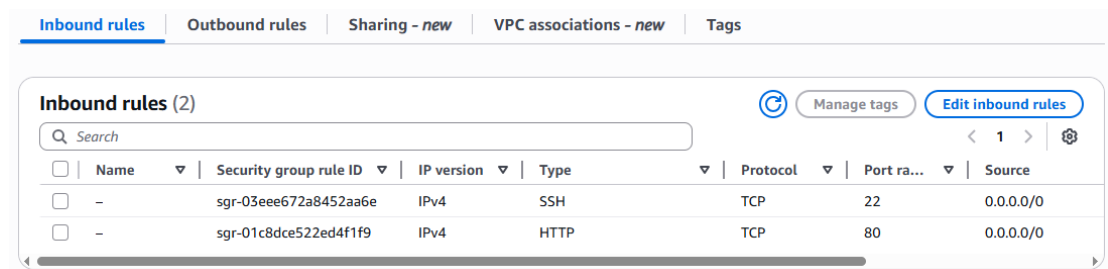
- Retrieve the public IPv4 DNS name of the **Misconfigured Web Server** instance.
- In a new browser window, try to open the public IPv4 DNS name that you retrieved.
- Diagnose why this does not work, and fix the misconfiguration.

The screenshot shows the AWS Management Console for EC2 Instances. The left sidebar shows the navigation menu with 'Instances' selected. The main panel displays a table of instances. The 'Misconfigured Web Server' instance is selected, and its details are shown below the table.

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability
Misconfigured Web...	i-028e2dd1b6986632c	Running	t2.micro	2/2 checks passed	View alarms +	us-west-2
Bastion host	i-04c3a2a8426f5b853	Running	t3.micro	3/3 checks passed	View alarms +	us-west-2
Web Server	i-0b007f363a405f65f	Running	t3.micro	3/3 checks passed	View alarms +	us-west-2

**i-028e2dd1b6986632c (Misconfigured Web Server)**

IPv6 address	Instance state	Public DNS
-	Running	ec2-54-188-78-88.us-west-2.compute.amazonaws.com   <a href="#">open address</a>



```
#
~\##### Amazon Linux 2
~~\#####\
~~\####| AL2 End of Life is 2026-06-30.
~~\#/
~~V~' '->
~~~~
~~_.. /
_/m/' -/ A newer version of Amazon Linux is available!

Amazon Linux 2023, GA and supported until 2028-03-15.
https://aws.amazon.com/linux/amazon-linux-2023/

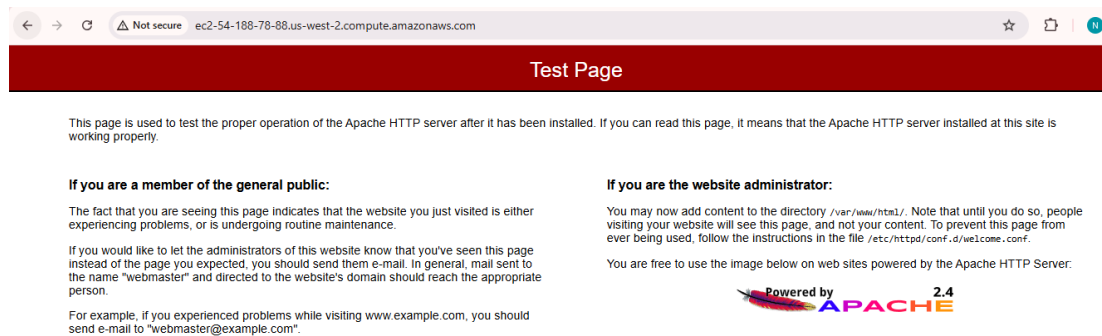
[ec2-user@ip-10-0-0-62 ~]$ sudo systemctl status httpd
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; vendor preset: disabled)
   Active: inactive (dead)
     Docs: man:httpd.service(8)

[ec2-user@ip-10-0-0-62 ~]$
```

## PublicIPs: 54.188.78.88 PrivateIPs: 10.0.0.62

```
[ec2-user@ip-10-0-0-62 ~]$ sudo systemctl start httpd
[ec2-user@ip-10-0-0-62 ~]$ sudo systemctl enable httpd
[ec2-user@ip-10-0-0-62 ~]$ sudo systemctl status httpd
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; vendor preset: disabled)
   Active: active (running) since Tue 2025-09-02 07:44:05 UTC; 18s ago
     Docs: man:httpd.service(8)
  Main PID: 32668 (httpd)
    Status: "Total requests: 0; Idle/Busy workers 100/0; Requests/sec: 0; Bytes served/sec: 0 B/sec"
    CGroup: /system.slice/httpd.service
            └─32668 /usr/sbin/httpd -DFOREGROUND
              └─32669 /usr/sbin/httpd -DFOREGROUND
                └─32670 /usr/sbin/httpd -DFOREGROUND
                  └─32671 /usr/sbin/httpd -DFOREGROUND
                    └─32672 /usr/sbin/httpd -DFOREGROUND
                      └─32673 /usr/sbin/httpd -DFOREGROUND

Sep 02 07:44:05 ip-10-0-0-62.us-west-2.compute.internal systemd[1]: Starting The Apache HTTP Server...
Sep 02 07:44:05 ip-10-0-0-62.us-west-2.compute.internal systemd[1]: Started The Apache HTTP Server.
[ec2-user@ip-10-0-0-62 ~]$
```



- At the end of the lab, your instructor will ask you the following questions:
  - What was the problem?
  - What did you do to fix the problem?

I started the Apache web server service on the instance.

## Conclusion

Congratulations! You now have successfully done the following:

- Launched an EC2 instance by using AWS Management Console
- Connected to the instance by using EC2 Instance Connect
- Launched an EC2 instance by using the AWS CLI

## Lab complete

Congratulations! You have completed the lab.