Using Auto Scaling in AWS (Linux)

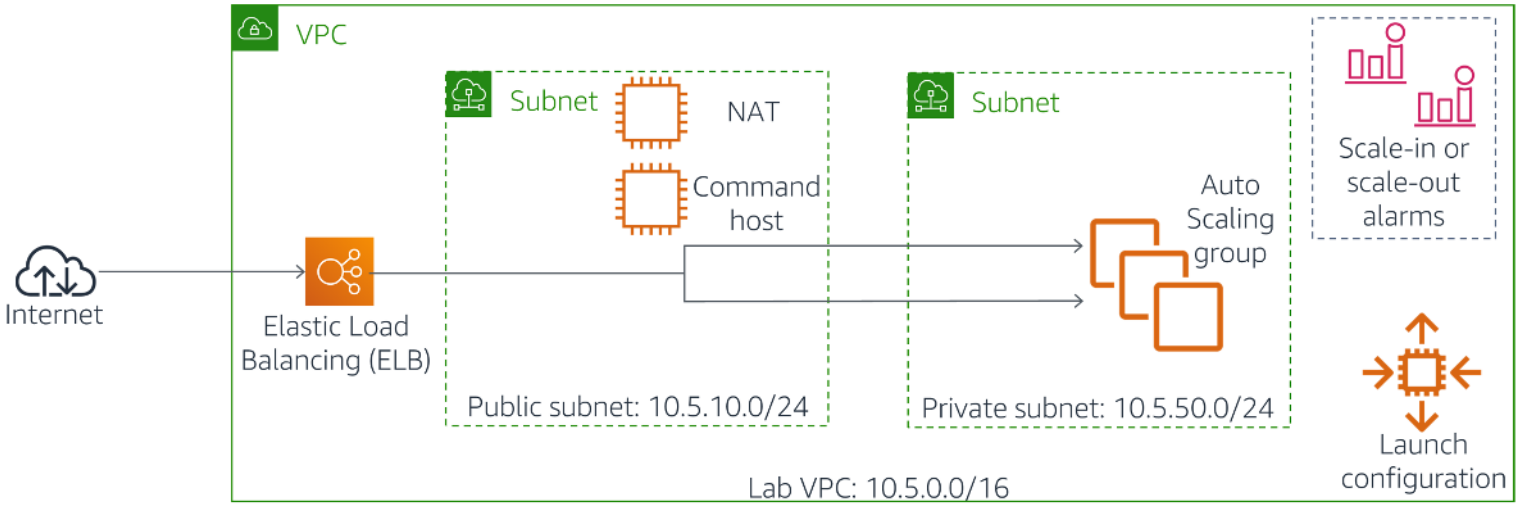
In this lab, you will be tasked with creating a new Amazon Machine Image (AMI) from an existing EC2 instance, and use that machine as the basis for defining a system that will scale automatically under increasing loads.

Duration

This lab will require approximately **45 minutes** to complete.

Scenario

In this lab, you will create the scalable web server system shown in the following diagram:



Objectives

After completing this lab, you will be able to:

- Create a new Amazon Machine Image (AMI) by using the Amazon Command Line Interface (CLI).
- Use Auto Scaling to scale up the number of servers available for a specific task when other servers are experiencing heavy load.

The following components are created for you as a part of the lab environment:

- Amazon VPC
- Public Subnets
- Private Subnets
- Amazon EC2 - Command Host (*in the public subnet*), you will log in to this instance to create a few of your AWS assets.

You will create the following components for this lab:

- Amazon EC2 - Web Server
- Amazon Machine Image (AMI)
- Auto Scaling Launch Configuration
- Auto Scaling Group
- Auto Scaling Policies
- ELB

Accessing the AWS Management Console

1. At the top of these instructions, click ▸ **Start Lab** to launch your lab.

Tip: If you need more time to complete the lab, then restart the timer for the environment by choosing the ▸ **Start Lab** button again.

2. Lab resources will be displayed on the top left corner.

   Example:

   - **AWS** 🟡 indicates that AWS lab resources are currently getting created.
   - **AWS** 🟢 indicates that AWS lab resources are ready.

   Please wait for the lab to be ready, before proceeding.

3. At the top of these instructions, click **AWS** 🟢

   This will open the AWS Management Console in a new browser tab. The system will automatically log you in.

   Tip: If a new browser tab does not open, there will typically be a banner or icon at the top of your browser indicating that your browser is preventing the site from opening pop-up windows. Click on the banner or icon and choose "Allow pop ups."

4. Arrange the AWS Management Console tab so that it displays along side these instructions. Ideally, you will be able to see both browser tabs at the same time, to make it easier to follow the lab steps.

   ⚠ **Do not change the lab region unless specifically instructed to do so**.

Task 1: Create a New Amazon Machine Image for Amazon EC2 Auto Scaling

In this task, you will launch a new EC2 instance and then create a new AMI based on that running instance. You will use the AWS CLI tools on the Command Host to perform all of these operations.

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

Windows Users: Using SSH to Connect

💬 These instructions are specifically for Windows users. If you are using macOS or Linux, skip to the next section.

5. Select the iDetails menu above these instructions you are currently reading..
6. Select the **Download PPK** button and save the **labsuser.ppk** file.
   Typically your browser will save it to the Downloads directory.
7. Make a note of the **Command Host** address.
8. Then exit the Details panel by selecting the **X**.
9. Download **PuTTY** to SSH into the Amazon EC2 instance. If you do not have PuTTY installed on your computer, download it here.
10. Open **putty.exe**
11. Configure PuTTY timeout to keep the PuTTY session open for a longer period of time.:
    - Select **Connection**
    - Set **Seconds between keepalives** to `30`
12. Configure your PuTTY session:
    - Select **Session**
    - **Host Name (or IP address):** Paste the **Public DNS or IPv4 address** of the **CommandHostIP**.
    - Back in PuTTY, in the **Connection** list, expand ⊞ **SSH**
    - Select **Auth** *(don't expand it)*
    - Select **Browse**
    - Browse to and select the lab#.ppk file that you downloaded

- Select **Open** to select it
- Select **Open** again.

13. Select **Yes**, to trust and connect to the host.
14. When prompted **login as**, enter: `ec2-user`
    This will connect you to the EC2 instance.
15. Windows Users: Select here to skip ahead to the next task.

macOS and Linux Users

These instructions are specifically for Mac/Linux users. If you are a Windows user, skip ahead to the next task.

16. Select the iDetails menu above these instructions you are currently reading. A Credentials window will be presented.
17. Select the **Download PEM** button and save the **labsuser.pem** file.
18. Make a note of the **Command Host \ Bastion Host** address, if it is displayed.
19. Then exit the Details panel by selecting the **X**.
20. Open a terminal window, and change directory `cd` to the directory where the *labsuser.pem* file was downloaded.
    For example, if the *labuser.pem* file was saved to your Downloads directory, run this command:

```
cd ~/Downloads
```

21. Change the permissions on the key to be read-only, by running this command:

```
chmod 400 labsuser.pem
```

22. Run the below command *(replace **<public-ip>** with the **Command Host \ Bastion Host** address you copied earlier)*.
    Alternatively, return to the EC2 Console and select **Instances**. Check the box next to the instance you want to connect to and in the *Description* tab copy the **IPv4 Public IP** value.:

```
ssh -i labsuser.pem ec2-user@<public-ip>
```

23. Type `yes` when prompted to allow the first connection to this remote SSH server.
    Because you are using a key pair for authentication, you will not be prompted for a password.

Configure the AWS CLI

24. Discover the region in which the Command Host instance is running:

```
curl http://169.254.169.254/latest/dynamic/instance-identity/document | grep region
```

    You will use this region information in the next step.
25. Update the AWS CLI software with the credentials.

```
aws configure
```

26. At the prompts, enter the following information:

- **AWS Access Key ID**: Press enter.
- **AWS Secret Access Key**: Press enter.
- **Default region name**: Type in the name of the region, which you just discovered a moment ago. For example, `us-east-1` or `eu-west-2`.
- **Default output format**: `json`

27. Now you are ready to access and run the scripts detailed in the exercises below. To access these scripts you will first need to navigate to their directory by issuing the following command.

```
cd /home/ec2-user/
```

Create A New EC2 Instance

Now that you are logged in to CommandHost, you will use the AWS CLI to create a new instance that hosts a web server.

28. Inspect the script UserData.txt that was installed for you as part of the CommandHost.

```
more UserData.txt
```

This script performs a number of initialization tasks, including updating all installed software on the box and installing a small PHP web application that you can use to simulate a high CPU load on the instance. Near the bottom of the script, you will see the following lines:

```
find -wholename /root/.*history -wholename /home/*/.*history -exec rm -f {} \;
find / -name 'authorized_keys' -exec rm -f {} \;
rm -rf /var/lib/cloud/data/scripts/*
```

These lines erase any history or security information that may have accidentally been left on the instance when the image was taken.

29. Use the **KEYNAME**, **AMIID**, **SUBNETID** and **HTTPACCESS** values you made a note of in the *Accessing the AWS Management Console* section and paste it into relevant sections of the below script.
Note: You can query these values by clicking the Details drop-down menu above these instructions, followed by the Show button.

```
aws ec2 run-instances --key-name KEYNAME --instance-type t3.micro --image-id AMIID --user-data file:///home
/ec2-user/UserData.txt --security-group-ids HTTPACCESS --subnet-id SUBNETID --associate-public-ip-address
--tag-specifications 'ResourceType=instance,Tags=[{Key=Name,Value=WebServerBaseImage}]' --output text
--query 'Instances[*].InstanceId'
```

The output of this command will provide you with an **InstanceId**. This value is referred to as **new-instance-id** in subsequent steps and should be replaced appropriately.

30. Use the **aws ec2 wait instance-running** command to monitor this instance's status. Replace *NEW-INSTANCE-ID* with the value you copied previously.

```
aws ec2 wait instance-running --instance-ids NEW-INSTANCE-ID
```

Wait for the command to return to a prompt, before proceeding to the next step.

31. Your instance should have started a new web server. To test that the web server was installed properly, obtain the public DNS name. Copy the output of this value (minus quotation marks). This value is referred to as **public-dns-address** in the next procedure. Replace *NEW-INSTANCE-ID* with the value you copied previously.

```
aws ec2 describe-instances --instance-id NEW-INSTANCE-ID --query
'Reservations[0].Instances[0].NetworkInterfaces[0].Association.PublicDnsName'
```

32. Use a Web browser to navigate to the address returned by the command above.
    - It could take a few minutes for the web server to be installed. Please wait for five minutes before trying other steps.
    - Do not click on *Start Stress* at this stage. Replace *PUBLIC-DNS-ADDRESS* with the value you copied in the last step.

      ```
      http://PUBLIC-DNS-ADDRESS/index.php
      ```

      If your web server does not appear to be running, check with your instructor.

Create a Custom AMI

In this procedure, you will create a new AMI based on that instance you just created.

33. Use the **aws ec2 create-image** command to create a new AMI based on this instance. Replace *NEW-INSTANCE-ID* with the value you copied previously.

```
aws ec2 create-image --name WebServer --instance-id NEW-INSTANCE-ID
```

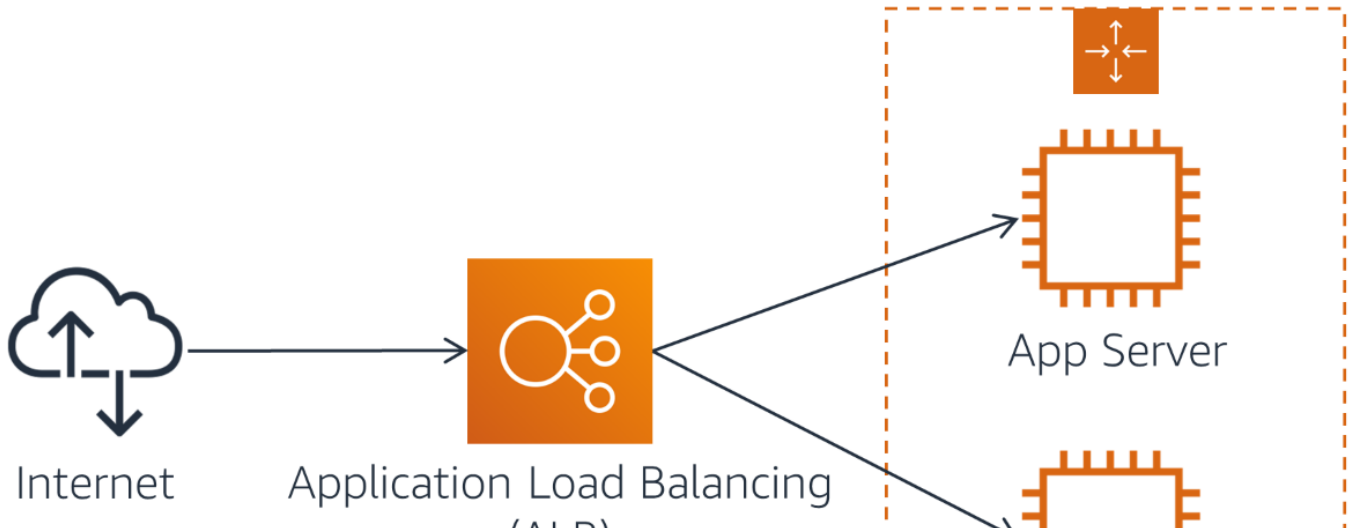
💬 By default, **create-image** will restart the current instance before creating the AMI, in order to ensure the integrity of the image on the file system. While your AMI is being created, proceed to the next section.

Task 2: Create an Auto Scaling Environment

In this section, you will create a load balancer that pools a group of EC2 instances under a single DNS address. You will use Auto Scaling to create a dynamically scalable pool of EC2 instances based on the image that you created in the previous section. Finally, you will create a set of alarms that will scale out or scale in the number of instances in your load balancer group whenever the CPU performance of any machine within the group exceeds or falls below a set of specified thresholds.

The following task can be performed using either the AWS CLI or the Management Console. For purposes of simplicity, you will implement this task using the Management Console.

Create an Application Load Balancer



34. On the **Services** menu, click **EC2**.
35. In the left navigation pane, click **Load Balancers** (you might need to scroll down to find it).
36. Click **Create Load Balancer**
37. Under **Application Load Balancer**, click **Create**
38. For **Name**, enter: `webserverloadbalancer`
39. Scroll down to the **Availability Zones** section, for **VPC**, select: **Lab VPC**

    You will now specify which *subnets* the Load Balancer should use. It will be an Internet-facing load balancer, so you will select both Public Subnets.
40. Click the **first** displayed Availability Zone, then click the **Public Subnet 1** displayed underneath.
41. Click the **second** displayed Availability Zone, then click the **Public Subnet 2** displayed underneath.

    You should now have two subnets selected: **Public Subnet 1** and **Public Subnet 2**. (If not, go back and try the configuration again.)
42. Click **Next: Configure Security Settings**

    A warning is displayed, which recommends using HTTPS for improved security. This is good advice but is not necessary for this lab.
43. Click **Next: Configure Security Groups**
44. Select ☑ **HTTPAccess** and ensure it is the only item selected.
45. Click **Next: Configure Routing**

    Target Groups define where to *send* traffic that comes into the Load Balancer. The Application Load Balancer can send traffic to multiple Target Groups based upon the URL of the incoming request, such as having requests from mobile apps going to a different set of servers. Your web application will use only one Target Group.
46. For **Name**, enter: `webserver-app`
47. Expand ▸ **Advanced health check settings**.

    The Application Load Balancer automatically performs *Health Checks* on all instances to ensure that they are responding to requests. The default settings are recommended, but you will make them slightly faster for use in this lab.
48. Configure these values:
    - **Path** `/index.php`
    - **Healthy threshold:** `2`
    - **Interval:** `10`

    This means that the Health Check will be performed every 10 seconds and if the instance responds correctly twice in a row, it will be considered healthy.
49. Click **Next: Register Targets**

    Targets are the individual instances that will respond to requests from the Load Balancer. You do not have any web application instances yet, so you can skip this step.

50. Click **Next: Review**
51. Review the settings and click **Create** then **Close**

Create a Launch Configuration

The launch configuration will be used by your Auto Scaling group to specify which AMI to use to create new EC2 instances. For this example, you will launch the AMI that you created previously, which automatically configures itself as a web server when it is launched.

52. In the left navigation pane, click **Launch Configurations**.
53. Click **Create launch configuration**
54. On the **Create launch configuration** page enter the following details:
    - **Name**: `WebServerLaunchConfiguration`
    - **AMI**: `WebServer`
    - **Instance Type**: Choose the **Choose instance type** and select `t3.micro`.
    - **Monitoring**: ☑ **Enable EC2 instance detailed monitoring within CloudWatch**
    - **Security groups**: Click ⦿ **Select an existing security group** and choose ☑ **HTTPAccess**.
    - **Key pair (login)**: Choose **proceed without a key pair**.
    - Select ☑ the acknowledgement check box, then click **Create launch configuration**

Create an Auto Scaling Group

Your Auto Scaling group will create a minimum number of Amazon EC2 instances that will reside behind your load balancer. In subsequent procedures, you will also add scale-out and scale-in policies that increase or decrease the number of running instances in reaction to alarms triggered by Amazon CloudWatch.

You should be on the **Launch configurations** page.

55. Select ☑ **WebServerLaunchConfiguration** and click **Actions** followed by **Create Auto Scaling group**.
56. Configure the below settings:
    - For **Auto Scaling group** name, type `WebServersASGroup` and click **Next**.
    - For **VPC**, Select **Lab VPC**
    - Subnets: **Private Subnet 1** and **Private Subnet 2**
    - Click **Next**
    - Under the **Load balancing - optional** section, select the **Attach to an existing load balancer option**.
    - Under **Select target groups**: select **webserver-app**
    - **Monitoring**: Select **Enable group metrics collection within CloudWatch**
    - Click **Next** to proceed to the **Configure group size and scaling policies** page.
    - In the **Configure group size and scaling policies** page, Configure as follows
        - **Desired capacity** : `2`
        - **Minimum capacity** : `2`
        - **Maximum capacity** : `4`
    - Under the **Target scaling policy** section, for **Target value**, enter: `40`
    - Click **Next** and then again click **Next** on the **Add notifications page**.
    - In the **Add tags** page, click **Add tag**, and then enter `Name` under **Key** and `WebApp` under **Value**.
    - Click **Next**
57. Review your configuration, and then click **Create Auto Scaling group**

Verifying the Auto Scaling configuration

In this task, you will verify that both the Auto Scaling configuration and the load balancer are working by accessing a pre-installed script on one of your servers that will consume CPU cycles, thus triggering the scale out alarm.

58. In the left navigation pane, click **Instances**.
59. Verify that two new instances labelled **WebApp** are being created as part of your Auto Scaling group.
60. Wait for the two new instances to complete initialization before you proceed to the next step. Observe the **Status Checks** field for the instances until it shows that both status checks have completed successfully.
61. In the left navigation pane, click **Target Groups**, and then select ☑ your target group (**webserver-app**).
62. On the **Targets** tab in the lower half of your screen, verify that two instances are being created. Keep refreshing this list until the **Status** of these instances changes to **healthy**.

    You can now test the web application by accessing it via the Load Balancer.
63. In the left navigation pane, click **Load Balancers** and then select ☑ **webserverloadbalancer**.
64. On the **Description** tab below, copy the **DNS name** value (which should look like **webserverloadbalancer-xxxxxxxxx.xx-xxxx-x.elb.amazonaws.com**). This value will be referred to as **load-balancer-url** in the next step.
65. Open a new web browser tab and paste the URL into the address bar, then press Enter.
66. On the web page, click **Start Stress**.

    This will call the application **stress** in the background, causing the CPU utilization on the instance that serviced this request to spike to 100%.
67. In the left navigation pane of the management console, click **Auto Scaling Groups**.
68. Select ☑ **WebServerASGroup**.
69. Select the **Activity** tab for your Auto Scaling group. After a few minutes, you should see your Auto Scaling group add a new instance.

    💬This is because CloudWatch detected that the average CPU utilization of your Auto Scaling group exceeded 45%, and your scale-up policy has been triggered in response.