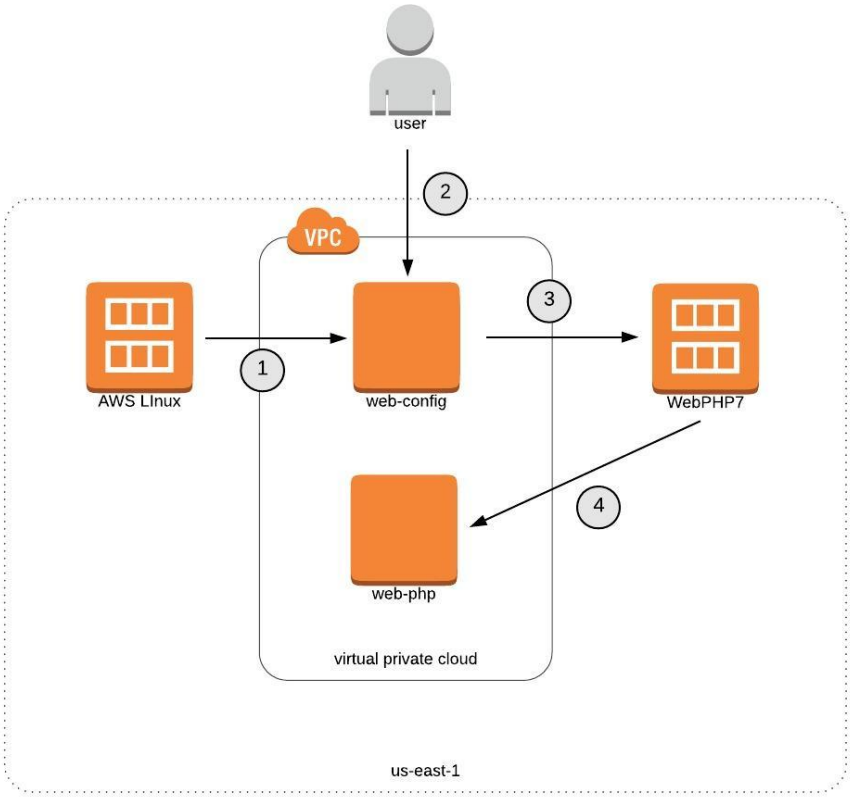
Create a Custom AMI in AWS

Introduction

AWS AMIs contain little more than just an operating system. So to run a PHP web application, for example, you would need to install Apache and PHP. This can be done automatically using user data, or with an automation tool like Chef or Puppet. However, doing it this way extends the wait time for the instance to be ready.

Many applications, such as those in an autoscaling environment, need instances ready to use immediately. By creating custom AMIs, we can launch pre-configured instances and skip the wait.



In this lab, we're going to create a custom AMI with Apache and PHP already installed. Then, we'll use that AMI to create a new instance that already has that configuration.

To do this, we'll:

1. Launch an instance from a standard AWS Linux AMI.
2. Connect to the instance via SSH and manually configure Apache and PHP.
3. Create a custom AMI.
4. Create a new instance from the custom AMI and verify that Apache and PHP are installed correctly on it.

1- Create the Custom AMI

Under *AWS services* at the top, type "EC2". You can also find it by opening the **Services** menu and clicking **EC2** under the *Compute* header.

Click **Launch Instance** and then select the **Amazon Linux 2 AMI**.

Next, select **General purpose t2.micro** from the list of instance types, and click **Next: Configure Instance Details**.

We'll leave it in the default VPC for now. Make sure *Auto-assign Public IP* is enabled, and click **Next: Add Storage**.

We can also leave the default settings here — the *Size (GiB)* should be **8GB** and the *Volume Type* should be **General Purpose SSD (gp2)**. Click **Next: Add Tags**.

Now, click **Add Tag** and add a *Key* called "Name" and a *Value* called "web-config". Click **Next: Configure Security Group**.

On the *Configure Security Group* page, we can leave the default setting under *Assign a security group* (it should be **Create a new security group**). Under *Security group name*, type "Web Security Group". For the *Description*, type "All SSH and HTTP".

There is already an SSH rule in place. Under *Source*, click the **Custom** dropdown and set it to **My IP** to restrict it to just our IP address. Then, we need to add an HTTP rule so

we can access Apache. Click **Add Rule** and select **HTTP**. This will allow port 80 from anywhere, which is fine. Click **Review and Launch**.

Everything should look good, so click **Launch**.

The final step is to create a key pair, so we can access the instance via SSH using a key pair. In the "*Select an existing key pair or create a new key pair*" pop-up, click the default **Choose an existing key pair** to open the dropdown, and select **Create a new key pair**. In *Key pair name*, enter "WebConfig" and click **Download Key Pair**. Make sure you check where it's saved (most likely to your *Downloads* folder) because we'll need it soon. Click **Launch Instances**.

On the *Launch Status* page, click **View Instances**.

Here, we'll see the status is *pending*. It may take a few minutes before it changes to *running* — after a few minutes, if it still says *pending*, you can click the *refresh* icon in the *Launch Instance* toolbar to see if it's done.

Once it changes to *running*, click the **Description** tab below the instance's public IP address and public DNS name.

Next, launch Terminal. (*Note:* If you are using Windows, you will need an SSH client such as PuTTY.)

First, we need to lock down the permissions on the SSH key, so enter:

$ chmod 400 Downloads/WebConfig.pem

Now we're ready to log in to the operating system by entering:

$ ssh -i Downloads/WebConfig.pem ec2-user@IP_ADDRESS

Make sure to replace IP_ADDRESS with your EC2 instance's public IP address. Copy it the old-fashioned way, or click the *copy* icon that appears when you hover over the IP address.

A prompt may ask, "Are you sure you want to continue connecting (yes/no)?" Type yes.

We should now be connected to the webconfig instance. Now, let's do our configuration.

First, let's make sure all of our packages are up to date.

$ sudo yum update -y

Next, enter the following bash script to install Apache and PHP:

$ sudo yum install -y httpd php

Now, let's start the Apache service.

$ sudo service httpd start

Set it to start up automatically at boot.

$ sudo chkconfig httpd on

Now, let's check that our Apache web page will come up if we put our public IP in our browser. Open a new tab and paste in your EC2 instance's public IP address. This should open up a *Test Page*.

Back in Terminal, let's set up a PHP page. For this, we're going to create a PHP page in */var/www/html*, and we're going to give our user the correct permissions.

Let's take the ec2-user and add it to the Apache group. Enter:

$ sudo usermod -a -G apache ec2-user

We'll also change the ownership of the /var/www directory:

$ sudo chown -R ec2-user:apache /var/www

Now we should have permissions to create a PHP page in var/www/html. For that, enter:

$ echo "<?php phpinfo(); ?>" > /var/www/html/phpinfo.php

Now let's see if PHP is configured correctly. In the *Test Page* window of the browser, add /phpinfo.php after your instance's IP address and press **Enter**. Our PHP page should come up, verifying that our instance is properly configured.

Next, let's create our custom AMI. Head back to your instance dashboard in AWS, and select your instance. Select **Actions** at the top, hover over **Image**, and click **Create Image**. For *Image name*, enter "WebPHP7". For *Image description*, enter "Web server with Apache and PHP7". Leave the rest of the default settings, click **Create Image**, and then close out of the *Create Image* success message pop-up.

In the sidebar under *Images*, click **AMIs** to watch our image's creation progress. We'll notice its status is *pending*. A snapshot is being made of the boot volume, and that will take a few minutes.

Once the snapshot finishes, it will be registered as an AMI — a custom AMI private to us. We can then use that to launch more instances. However, note that when we create a custom AMI, it's only usable in the region we created it in. If we needed to use it in a different region, we would have to copy it. We could do that by clicking **Actions**, selecting **Copy AMI**, and then setting a different *Destination region*.

Once our AMI's status changes to *available*, we can launch our instance using our new custom AMI, so we won't have to do any configuration to it. We could launch it here, but let's go to the EC2 service so we can see it in the custom AMIs tab. Click **Instances** in the sidebar, and then click **Launch Instance**.

On the *Choose an Amazon Machine Image (AMI)* page, click the **My AMIs** tab in the sidebar. We should see our **WebPHP7** AMI listed. Instead of choosing an AMI from the *Quick Start* tab like we did earlier, select our custom AMI. Then we'll go through the same steps as before.

Select **General purpose t2.micro** from the list of instance types, and click **Next: Configure Instance Details**.

Leave it in the default VPC, make sure *Auto-assign Public IP* is enabled, and click **Next: Add Storage**.

Leave the default settings here, and click **Next: Add Tags**.

Now click **Add Tag** and add a *Key* called "Name" and a *Value* called "web-php". Click **Next: Configure Security Group**.

For the security group, click **Select an existing security group** and select the security group we created for the previous instance (the one named *Web Security Group*). Click **Review and Launch**.

Click **Launch**.

In the "*Select an existing key pair or create a new key pair"* pop-up, leave the default **Choose an existing key pair** setting. Under *Select a key pair*, select our previously created "WebConfig" key pair. Check the acknowledgement checkbox to indicate that we do have access to the selected private key file, and then click **Launch Instances**.

On the *Launch Status* page, click **View Instances**.

Once our web-php instance gets to the *running* state, we'll make sure everything is properly installed. Select **web-php**, click the **Description** tab below, and copy its IP address. Paste it into a new browser tab, add /phpinfo.php after it, and press **Enter**. The page should pop up to show everything is working.