Amazon Web Services (AWS)

**BY**

**G. Ranjith Kumar**

**Services:**

* **EC2 (Elastic Compute Cloud)**
* **VPC (Virtual Private Cloud)**
* **RDS (Ralational Data Services)**
* **IAM (Identity And Access Management)**
* **Lambda**
* **CloudWatch**
* **S3 (Simple Storage Service)**

**Getting Started with Amazon EC2:**

**Prerequisites:**

* [**Sign Up for AWS**](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/get-set-up-for-amazon-ec2.html#sign-up-for-aws)
* [**Create an IAM User**](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/get-set-up-for-amazon-ec2.html#create-an-iam-user)
* [**Create a Key Pair**](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/get-set-up-for-amazon-ec2.html#create-a-key-pair)
* [**Create a Virtual Private Cloud (VPC)**](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/get-set-up-for-amazon-ec2.html#create-a-vpc)
* [**Create a Security Group**](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/get-set-up-for-amazon-ec2.html#create-a-base-security-group)
* **Launch EC2 Instance**

**To create an AWS account**

1. Open **https://portal.aws.amazon.com/billing/signup**.
2. Follow the online instructions.

Part of the sign-up procedure involves receiving a phone call and entering a verification code on the phone keypad.

**Note** your **AWS account number**, because you'll **need** it for the **next task**.

**IAM**

**Create an IAM User**

**To create an administrator user for yourself and add the user to an administrators group (console)**

1. Use your AWS account email address and password to sign in as the ***AWS account root user***to the **IAM** console at **https://console.aws.amazon.com/iam/.**

**Note**

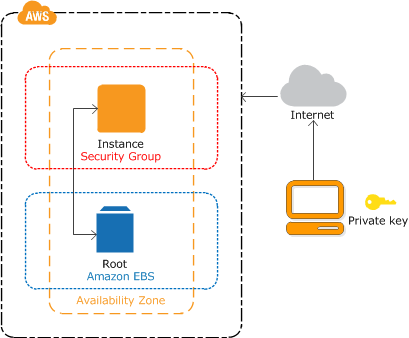
We strongly recommend that you adhere to the best practice of using the **Administrator** IAM user below and securely lock away the root user credentials. Sign in as the root user only to perform a few account and service management tasks.

1. In the navigation pane, choose **Users** and then choose **Add user**.
2. For **User name**, enter **Administrator**.
3. Select the check box next to **AWS Management Console access**. Then select **Custom password**, and then enter your new password in the text box.
4. (Optional) By default, AWS requires the new user to create a new password when first signing in. You can clear the check box next to **User must create a new password at next sign-in** to allow the new user to reset their password after they sign in.
5. Choose **Next: Permissions**.
6. Under **Set permissions**, choose **Add user to group**.
7. Choose **Create group**.
8. In the **Create group** dialog box, for **Group name** enter **Administrators**.
9. Choose **Filter policies**, and then select **AWS managed -job function** to filter the table contents.
10. In the policy list, select the check box for **AdministratorAccess**. Then choose **Create group**.

**Note**

You must activate **IAM user** and **role** access to Billing before you can use the **AdministratorAccess** permissions to access the **AWS Billing** and Cost Management console. To do this, follow the instructions in **step 1** of the tutorial about delegating access to the billing console.

1. Back in the list of groups, select the check box for your new group. Choose **Refresh** if necessary to see the group in the list.
2. Choose **Next: Tags**.
3. (Optional) Add metadata to the user by attaching tags as key-value pairs.
4. Choose **Next: Review** to see the list of group memberships to be added to the new user. When you are ready to proceed, choose **Create user**.

****

**Basic Overview**

**Key Pair**

**Creating an SSH Key Pair**

To interact with the cluster launcher and other instances, you must create an SSH key pair or use an existing EC2 key pair. If you do not have a key pair, follow these steps:

When you launch an instance, you specify the key pair. You must create a SSH key pair that you create at launch or use an existing key pair .  At boot time, the public key content is placed on the instance in an entry within **~/.ssh/authorized\_keys**.

**To create your key pair using the Amazon EC2 console**

1. Open the **Amazon EC2** console at **https://console.aws.amazon.com/ec2/.**
2. In the navigation pane, under **NETWORK & SECURITY**, choose **Key Pairs**.

**Note**

The navigation pane is on the left side of the Amazon EC2 console. If you do not see the pane, it might be minimized; choose the arrow to expand the pane.

1. Choose **Create Key Pair**.
2. For **Key pair name**, enter a name for the new key pair, and then choose **Create**.
3. The **private key** file is automatically downloaded by your browser. The base file name is the name you specified as the name of your **key pair**, and the file name extension is **.pem**. Save the **private key file** in a safe place.

**Important**

This is the only chance for you to **save** the **private key** file. You'll need to provide the name of your **key pair** when you launch an instance and the corresponding private key (**.ppk file**) each time you **connect** to the **instance**.

1. If you will use an **SSH client** on a **Mac** or **Linux** computer to **connect** to your **Linux instance**, use the following command to set the permissions of your **private key** file so that only you can read it.

**Chmod 400 smartcreed-key.pem**

If you do not set these permissions, then you cannot connect to your instance using this key pair. For more information, see **Error**: Unprotected Private Key File.

## VPC

## Create a Virtual Private Cloud (VPC)

1. Open the **Amazon EC2** console at **https://console.aws.amazon.com/vpc/.**
2. From the navigation bar, select a **region** for the **VPC**. VPCs are specific to a region, so you should select the same region in which you created your **key pair**.
3. In the **VPC Dashboard**, choose **VPC**, and then **create VPC.**
4. On the configuration page, enter a **name** for your **VPC** in the **VPC name field**; for example, my-vpc, and enter a name for your subnet in the Subnet name field. This helps you to identify the **VPC** and **subnet** in the Amazon VPC console after you've created them. For this exercise, you can leave the rest of the configuration settings on the page, and choose **Create VPC**.
5. After completing the **VPC** creation, select **SAM-VPC** and go to Actions, enable DNS.
6. Next Create **Subnet** and name it as **SAM-Subnet,** Select **VPC, Availability Zone** and **CIDR Range**
7. Select **Actions** and go to **Modify auto-assign IP-settings**
8. Next Create **Internet Gateway**
9. And then **IGW** attached to **VPC**
10. Next Create **Route Table** and name it as **SAM-Route**
11. Select **SAM-Route** and go to **edit subnet association**
12. Go to **edit subnet association** and add it
13. Next go to **edit routes** and add **all traffic** to **IGW** like below
14. Create **Security Group**, name it as **SCC-SG**.
15. And add **rules** to it like shown in below :
16. After completing the **VPC** creation and **Security Group** next, go to **launch EC2 instance** and click the launch instance.

**Security Group**

**Creating Your SC-SG Security Group**

1. You can create your **security group** using the **Amazon VPC** console.
2. To create the **SC-SG** security group and **add rules**
3. Open the **Amazon VPC** console at **https://console.aws.amazon.com/vpc/**.
4. In the navigation pane, choose Security Groups.
5. Choose Create Security Group.
6. In the Group name field, enter **SC-SG** as the name of the security group, and provide a

description. You can optionally use the **Name tag** field to **create** a tag for the security group with a **key of Name** and a value that you specify.

1. Select the **ID** of your **VPC** from the VPC menu, and then choose **Yes**, **Create**.
2. Select the **SC-SG** security group that you just created (you can view its name in the Group

Name column).

1. On the **Inbound** Rules tab, choose **Edit** and **add rules** for inbound traffic as follows:

**a**. Select **HTTP** from the Type list, and enter **0.0.0.0/0** in the Source field.

**b**. Choose **Add** another rule, then select **HTTPS** from the Type list, and enter **0.0.0.0/0** in the

Source field.

**c**. Choose **Add** another rule. If you're launching a Linux instance, select **SSH** from the Type list,

or if you're launching a Windows instance, select **RDP** from the Type list. Enter your **network's**

**public IP** address range in the Source field. If you don't know this address range, you can use

**0.0.0.0/0** for this exercise.

**Important :**

If you use **0.0.0.0/0**, you enable all IP addresses to access your instance using

**SSH or RDP**. This is acceptable for the short exercise, but it's unsafe for production

environments. In production, you'll authorize only a specific IP address or range of

addresses to access your instance.

**d**. Choose **Save.**

**EC2**

**Launch an Instance into Your VPC**

1. Open the **Amazon EC2** console at **https://console.aws.amazon.com/ec2/.**
2. In the **navigation bar**, on the **top-right**, ensure that you select the **same region** in which you created your **VPC** and **security group**.
3. From the **dashboard**, **choose Launch Instance.**
4. On the first page of the wizard, choose the AMI that you want to use. Select an Amazon Linux AMI(**Ubuntu server**) : **Ubuntu Server 18.04 LTS (HVM)**
5. After selecting **Ubuntu server**, go to **instance type : t2.micro (free tier).**

**1 vcpu – 1gb memory**

1. Click **Next: Configure Instance Details**.
   1. Select the **VPC** and **subnet**, what you have created earlier.
   2. The cluster launcher requires Internet access; from the **Auto-assign Public IP** list box, select **Enable**.
   3. Use the default shutdown behavior, **Stop**.
   4. Click the **Protect against accidental termination** checkbox.
   5. (Optional) Click the IAM role drop-down list and select an IAM role.
   6. In Advanced Details (Optional) :
2. Click **Next: Add Storage**. It requires a minimum of 8 GB.
3. Click **Next: Add Tags**. On the Add Tags page, click the **Add Tag** button. For the **Name** key, enter a name for the instance in the **Value** field. Optionally, click **Add Tag** again to create an additional tags for the instance (up to a maximum of 50 tags).
4. Click **Next: Configure Security Group**.
5. On the **Configure Security Group** page, create a new security group or add ports to an existing group. (If you have already created a security group with the required ports for SC, as described on the previous steps 17-25, you can skip this step.)
   1. Select either **Create a new security group** or **Select an existing security group**. If you create a new group, enter a **Security group name** and **Description**. To edit an existing group, select the group you want to edit.
   2. Click the **Type** drop-down list, and select a protocol type. Type the port number in the **Port Range** field.
   3. For each additional port needed, click the **Add Rule** button. Then click the **Type** drop-down list, select a protocol type, and type the port number in the **Port Range** field.
6. Click **Review and Launch**. Scroll down to review the AMI details, instance type, and security group information, and then click **Launch**

**At the prompt for a key pair:**

Select **Choose an existing key pair** and select the key pair you created in page no. 4 “**Creating an SSH Key Pair” .**

1. Click the check box to acknowledge that you have access to the private key.
2. Click **Launch Instances**.

**Elastic IP**

**Assign an Elastic IP Address to Your Instance**

In the **previous step**, you launched your **instance** into a **public subnet** — a subnet that has a route to an **Internet gateway**. However, the instance in your subnet also needs a **public IPv4 address** to be able to communicate with the Internet. By **default**, an instance in a **nondefault VPC** is not assigned a **public IPv4** address. **In this step**, you'll **allocate** an **Elastic IP** address to your account, and then **associate** it with your **instance**. For more information about **Elastic IP addresses**, see **Elastic IP Addresses**.

**To allocate and assign an Elastic IP address**

1. Open the **Amazon VPC console** at **https://console.aws.amazon.com/vpc/ (or)** Open the **Amazon EC2 Console** at **https://console.aws.amazon.com/ec2/**
2. In the navigation pane, **choose Elastic IPs**.
3. Choose **Allocate new address**, and then **Allocate**.
4. Select the **Elastic IP** address from the list, **choose Actions**, and then choose **Associate Address**.
5. Choose your **instance** from the Instance list. Choose **Associate** when you're done. Your **instance** is now **accessible** from the Internet. You can **connect** to your **instance** through its **Elastic IP** address **using SSH** or **Remote Desktop** from your **home network**.

**Relational Database Services (RDS)**

Amazon **Relational Database Service** (**Amazon RDS**) is a **web service** that makes it easier to set up, operate, and scale a relational database in the cloud. It provides cost-efficient, resizable capacity for an industry-standard relational database and manages common database administration tasks.

**To create a MySQL DB instance with Easy Create enabled**

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the upper-right corner of the Amazon RDS console, choose the AWS Region in which you want to create the DB instance.
3. In the navigation pane, choose **Databases**.
4. Choose **Create database** and make sure that **Easy Create** is chosen.
5. In **Configuration**, choose **MySQL**.
6. For **DB instance size**, choose **Free tier**.
7. For **DB instance identifier**, enter a name for the DB instance, or leave the default name.
8. For **Master username**, enter a name for the master user, or leave the default name.
9. To use an automatically generated master password for the DB instance, enable **Auto generate a password**.

To enter your master password, disable **Auto generate a password**, and then enter the same password in **Master password** and **Confirm password**.

1. (Optional) Open **View default settings for Easy create**.
2. You can examine the default settings used when **Easy Create** is enabled. If you want to change one or more settings during database creation, choose **Standard Create** to set them. The **Editable after database creation** column shows which options you can change after database creation. To change a setting with **No** in that column, use **Standard Create**. For settings with **Yes** in that column, you can either use **Standard Create** or modify the DB instance after it is created to change the setting.
3. Choose **Create database**.

If you chose to use an automatically generated password, the **View credential details** button appears on the **Databases** page.

To view the master username and password for the DB instance, choose **View credential details**.

**To connect to a database on a DB instance using MySQL monitor**

1. Find the endpoint (DNS name) and port number for your DB Instance.
   1. Open the RDS console and then choose **Databases** to display a list of your DB instances.
   2. Choose the MySQL DB instance name to display its details.
   3. On the **Connectivity & security** tab, copy the endpoint. Also, note the port number. You need both the endpoint and the port number to connect to the DB instance.
   4. **Connect** to the a **database** on a **MySQL DB instance** from **command prompt** :

**mysql -h *<endpoint>* -P 3306 -u *<mymasteruser>* -p**

**To create a MySQL DB instance with Standard Create**

1. In the navigation pane, choose **Databases**.
2. Choose **Create database**.
3. On the **Create database** page, shown following, make sure that the **Standard Create** option is chosen, and then choose **MySQL**.
4. the **Templates** section, choose **Dev/Test**.
5. In the **Settings** section, set these values:
   1. **DB instance identifier** – **tutorial-db-instance**
   2. **Master username** – **tutorial\_user**
   3. **Auto generate a password** – Disable the option
   4. **Master password** – Choose a password.
   5. **Confirm password** – Retype the password.
6. In the **DB instance size** section, set these values
   1. **DB instance performance type** – **Burstable**
   2. **DB instance class** – **db.t2.small**
7. In the **Storage** and **Availability & durability** sections, use the default values.
8. In the **Connectivity** section, open **Additional connectivity configuration** and set these values:
   1. **Virtual Private Cloud (VPC)** – Choose an existing VPC with both public and private subnets, such as the tutorial-vpc (vpc-*identifier*) created in Create a VPC with Private and Public Subnets

**Note**

The VPC must have subnets in different Availability Zones.

* 1. **Subnet group** – The DB subnet group for the VPC, such as the tutorial-db-subnet-group created in Create a DB Subnet Group
  2. **Publicly accessible** – **No**
  3. **VPC security groups** – Choose an existing VPC security group that is configured for private access, such as the tutorial-db-securitygroup created in Create a VPC Security Group for a Private DB Instance.

Remove other security groups, such as the default security group, by choosing the **X** associated with each.

* 1. **Availability zone** – **No Preference**
  2. **Database port** – **3306**

1. Open the **Additional configuration** section, and enter **sample** for **Initial database name**. Keep the default settings for the other options.
2. To create your Amazon RDS MySQL DB instance, choose **Create database**.

Your new DB instance appears in the **Databases** list with the status **Creating**.

1. Wait for the **Status** of your new DB instance to show as **Available**. Then choose the DB instance name to show its details.
2. In the **Connectivity & security** section, view the **Endpoint** and **Port** of the DB instance.
3. **Note** : The **endpoint** and **port** for your **DB instance**. You use this information to **connect** your **web server** to your **RDS DB instance**.

**LAMBDA**

**To create the Lambda function (console)**

Scheduled Start/Stop of EC2 Instances using Lambda and Cloud Watch Events

1. Sign in to the AWS Management Console and Go to **IAM Console**
2. In this **IAM** go to **Policies** and select **create policy.**
3. Next go to **visual editor** seclect **Service** : **EC2**

**Actions** : **Manual Actions --- Access level :**

**List,**

**Read,**

**Tagging,**

**Write.**

**Write  stopinstances** & **startinstances** (select both).

1. **Resources : Specific**

* **All Resources**

**OR** copy the code into **JSON** direct

**Lambda start/stop policy :**

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Action": [

"logs:CreateLogGroup",

"logs:CreateLogStream",

"logs:PutLogEvents"

],

"Resource": "arn:aws:logs:\*:\*:\*"

},

{

"Effect": "Allow",

"Action": [

"ec2:Start\*",

"ec2:Stop\*"

],

"Resource": "\*"

}

]

}

1. **Review policy :**

**Name** : **StartStopInstancesPolicy** -----> **Create policy**

1. Next go to **create role** in **IAM :**

**Select type of trusted entity** : **AWS Service**  ---->

1. **Choose the service that will use this role** : **Lambda ----->** Next **Permissions .**
2. **Attach permissions policy : Select**  **StartStopInstancesPolicy (Already Created : Step 2-6) and go to tags.**
3. Next go to **Review** enter role **Name** : **Lambda \_Ec2\_Start\_Stop -----> Create Role.**
4. **Go to Lambda select Create Function -----> Author From Scratch**

**Function Name: StartEC2Instance**

**Runtime : Node.js 10.x/Python 3.6**

**Permissions:**

**Execution Role**: Choose an existing role (**Lambda \_Ec2\_Start\_Stop**)

**Hit enter Create function**.

1. Go to the **StartEC2Instance function configuration**
2. **Go to function code section (index.js/index.py) : delete the existing code and replace below code :**

**StartEC2Instance: Node.js code**

const AWS = require('aws-sdk');

exports.handler = (event, context, callback) => {

const ec2 = new AWS.EC2({ region: 'ap-south-1' });

ec2.startInstances ({ InstanceIds: ['i-0fc442cc86d2ac356', 'i-03b3ce95558561e2b'] }).promise()

.then (() => callback(null, `Successfully started ${event.instanceId}`))

.catch (err => callback(err));

};

**Python code for start instance:**

Import boto3

# Enter the region your instances are in. Include only the region without specifying Availability Zone; e.g.; 'us-east-1'

region = 'XX-XXXXX-X'

# Enter your instances here: ex. ['X-XXXXXXXX', 'X-XXXXXXXX']

instances = ['X-XXXXXXXX']

def lambda\_handler(event, context):

ec2 = boto3.client('ec2', region\_name=region)

ec2.start\_instances(InstanceIds=instances)

print 'started your instances: ' + str(instances)

**StopEC2Instance : Node.js code**

const AWS = require('aws-sdk');

exports.handler = (event, context, callback) => {

const ec2 = new AWS.EC2({ region: 'ap-south-1' });

ec2.stopInstances({ InstanceIds: ['i-0fc442cc86d2ac356', 'i-03b3ce95558561e2b'] }).promise()

.then(() => callback(null, `Successfully stopped ${event.instanceId}`))

.catch(err => callback(err));

};

**Python code for stop instance:**

import boto3

# Enter the region your instances are in. Include only the region without specifying Availability Zone; e.g., 'us-east-1'

region = 'XX-XXXXX-X'

# Enter your instances here: ex. ['X-XXXXXXXX', 'X-XXXXXXXX']

instances = ['X-XXXXXXXX']

def lambda\_handler(event, context):

ec2 = boto3.client('ec2', region\_name=region)

ec2.stop\_instances(InstanceIds=instances)

print 'stopped your instances: ' + str(instances)

1. Change **Timeout** 3sec to 10 sec.
2. And Next **Configure Test** event ----> enter the event name and save Test. Properly test it working or not.
3. Repeat steps **11 to 15** **for StopEC2Instance** also.

**CloudWatch**

1. Go to **Cloudwatch console and**  event setup for **cron schedule** ----> **Rules**
2. In the navigation pane, under **Events**, choose **Rules.**
3. Select **create rule** ----> **Event Source ----->**

**Event Pattern**

* **Schedule ----> Select Cron Expression :**

|  |  |
| --- | --- |
| **Schedule** | Cron expression  30 2 \* \* ? \* |

1. **If this is correct expression showing metrics like this below :**

|  |  |
| --- | --- |
| **Next 10 Trigger Date(s)** | Tue, 11 Jun 2019 02:30:00 GMT  Wed, 12 Jun 2019 02:30:00 GMT  Thu, 13 Jun 2019 02:30:00 GMT  Fri, 14 Jun 2019 02:30:00 GMT  Sat, 15 Jun 2019 02:30:00 GMT  Sun, 16 Jun 2019 02:30:00 GMT  Mon, 17 Jun 2019 02:30:00 GMT  Tue, 18 Jun 2019 02:30:00 GMT  Wed, 19 Jun 2019 02:30:00 GMT  Thu, 20 Jun 2019 02:30:00 GMT |

1. Now go to **Target** ----> **add target** ----> select **topic** **: Lambda function**

**Function** : **StartEc2Instance** ---> **configure details**

1. Repeat steps **18 & 19** **for StopEC2Instance** also.

## Hosting a PHP Web Portal with AWS Ubuntu Server

**Prerequisites:**

* **LAMP Server**
* **Rewrite Engine**
* **PhpMyadmin**
* **SSL Certificate**

**Installation Process:**

#### 1. Install Apache web server:

**Update:**

* **sudo apt update**

**Next, install Apache web server:**

* **sudo apt install apache2**

**Check if Apache web server is running or not:**

* **sudo systemctl status apache2**

**Adjust firewall to allow apache web server:**

By default, the apache web browser can’t be accessed from remote systems if you have enabled the UFW firewall in Ubuntu 18.04 LTS. You must allow the http and https ports by following the below steps.

First, list out the application profiles available on your Ubuntu system using command:

* **sudo ufw app list**

If you look at the Apache Full profile, it should show that it enables traffic to ports 80 and 443:

* **sudo ufw allow in "Apache Full"**

Now Test your Apache Web Server: **http://your\_server\_ip**

If you see apache web page, then your web server is now correctly installed and accessible through your firewall.

#### Install MySQL:

**install Mysql server:**

**sudo apt install mysql-server**

**Verify if MySQL service is running or not using command:**

**sudo systemctl status mysql**

During the installation, you will be prompted to set the root user password. Otherwise follow the below steps.

**Setup Database root password:**

By default, MySQL **root** user password is blank. You need to secure your MySQL server by running the following script:

**sudo mysql\_secure\_installation**

This will ask if you want to configure the **VALIDATE PASSWORD PLUGIN**.

If your answer is **Yes**, you will be asked to choose the level of password validation.

The available password validations are **low**, **medium** and **strong**. Just enter the appropriate number (0 for low, 1 for medium and 2 for strong password) and hit ENTER key.

Output will be shown in below. Now enter the password for **MySQL root** user.

Please set the password for root here.

New password: **@sample123**

Re-enter new password: **@sample123**

Estimated strength of the password: **50**

Do you wish to continue with the password provided?(Press y|Y for Yes, any other key for No) : **y**

For the rest of questions, just type **y** and hit **ENTER**. This will remove anonymous user, disallow root user login remotely and remove test database.

That’s it. Password for MySQL root user has been set.

**Change Authentication method for Mysql root user:**

By default, MySQL root user is set to authenticate using the **auth\_socket** plugin in MySQL 5.7 and newer versions on Ubuntu.

If you prefer to use a password when connecting to MySQL as **root**, you will need to switch its authentication method from **auth\_socket**to **mysql\_native\_password**. To do this, open up the MySQL prompt from your terminal:

**sudo mysql**

Next, check which authentication method each of your MySQL user accounts use with the following command:

**SELECT user,authentication\_string,plugin,host FROM mysql.user;**

As you see, mysql root user uses **auth\_socket** plugin for authentication.

To change this authentication to **mysql\_native\_password** method, run the following command

**ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql\_native\_password BY 'password';**

Update the changes using command:

**FLUSH PRIVILEGES;**

Now check again if the authentication method is changed or not using command:

**SELECT user,authentication\_string,plugin,host FROM mysql.user;**

Once you confirm this on your own server, you can exit the MySQL shell:

**exit**

#### Install PHP:

**sudo apt install php libapache2-mod-php php-mysql**

After installing PHP, create **info.php** file in the Apache root document folder. Usually, the apache root document folder will be **/var/www/html/** or **/var/www/** in most Debian based Linux distributions. In Ubuntu 18.04 LTS, it is **/var/www/html/**.

Let us create **info.php** file in the apache root folder:

**sudo nano /var/www/html/info.php**

Add the following lines:

**<?php**

**phpinfo();**

**?>**

**save** and **exit** the file. **Restart apache** service to take effect the changes.

**sudo systemctl restart apache2**

**Test PHP:**

Open up your web browser and navigate to **http://IP-address/info.php** URL.

Usually, when a user requests a directory from the web server, Apache will first look for a file named **index.html**. If you want to change Apache to serve php files rather than others, move **index.php** to first position in the **dir.conf** file.

**sudo nano /etc/apache2/mods-enabled/dir.conf**

Here is the contents of the above file.

**<IfModule mod\_dir.c>**

**DirectoryIndex index.html index.cgi index.pl index.php index.xhtml index.htm**

**</IfModule>**

**# vim: syntax=apache ts=4 sw=4 sts=4 sr noet**

Move the “**index.php**” file to first. Once you made the changes, your **dir.conf** file will look like below.

**<IfModule mod\_dir.c>**

**DirectoryIndex index.php index.html index.cgi index.pl index.xhtml index.htm**

**</IfModule>**

**# vim: syntax=apache ts=4 sw=4 sts=4 sr noet**

Press ESC key and type :wq to save and close the file. Restart Apache service to take effect the changes.

**sudo systemctl restart apache2**

**To install all modules (not necessary though), run:**

**sudo apt-get install php\***

Do not forget to **restart Apache** service after installing any php module. To check if the module is loaded or not, open **info.php** file in your browser and check if it is present.

**Next, you might want to install any database management tools to easily manage databases via a web browser.**

#### Install phpMyAdmin:

sudo nano /etc/apache2/mods-enabled/dir.conf

To install all modules (not necessary though), run:

sudo apt-get install php\*

Do not forget to restart Apache service after installing any php module. To check if the module is loaded or not, open info.php file in your browser and check if it is present.

sudo add-apt-repository universe

Now is the time to install phpMyAdmin.

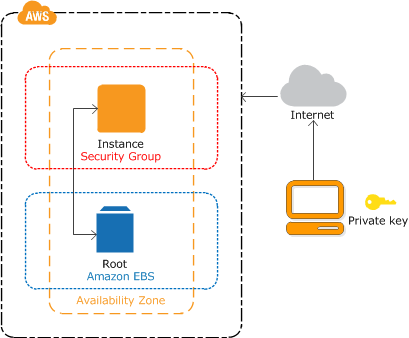
To install phpMyAdmin on Ubuntu 18.04, run:

sudo apt update

sudo apt install phpmyadmin php-mbstring php-gettext

Now, select the web server that should be automatically configured to run phpMyAdmin. Choose the web server with UP/DOWN arrows and hit the SPACEBAR key. Once you chose the web server, you will see a \* (star) symbol in-front of it. Hit the TAB key to choose OK and again hit ENTER key to continue.

If you've already signed up for Amazon Web Services (AWS), you can start using Amazon EC2 immediately. You can open the Amazon EC2 console, choose Launch Instance, and follow the steps in the launch wizard to launch your first instance.



**Step 1** − Sign-in to AWS account and open IAM console by using the following link [https://console.aws.amazon.com/iam/.](https://console.aws.amazon.com/iam/)

**Step 2** − In the navigation Panel, create/view groups and follow the instructions.

**Step 3** − Create IAM user. Choose users in the navigation pane. Then create new users and add users to the groups.

**Step 4** − Create a Virtual Private Cloud using the following instructions.

Open the Amazon VPC console by using the following link − <https://console.aws.amazon.com/vpc/>

Select VPC from the navigation panel. Then select the same region in which we have created key-pair.

Select start VPC wizard on VPC dashboard.

Select VPC configuration page and make sure that VPC with single subnet is selected. The choose Select.

VPC with a single public subnet page will open. Enter the VPC name in the name field and leave other configurations as default.

Select create VPC, then select Ok.

**Step 5** – Create security groups and add rules using the following instructions.

* On the VPC console, select Security groups in the navigation panel.
* Select create security group and fill the required details like group name, name tag, etc.
* Select your VPC ID from the menu. Then select yes, create button.
* Now a group is created. Select the edit option in the inbound rules tab to create rules.

**Step 6** − Launch EC2 instance into VPC using the following instructions.

* Open EC2 console by using the following link − <https://console.aws.amazon.com/ec2/>
* Select launch instance option in the dashboard.
* A new page will open. Choose Instance Type and provide the configuration. Then select Next: Configure Instance Details.
* A new page will open. Select VPC from the network list. Select subnet from the subnet list and leave the other settings as default.
* Click Next until the Tag Instances page appears.

**Step 7** − On the Tag Instances page, provide a tag with a name to the instances. Select Next: Configure Security Group.

**Step 8** − On the Configure Security Group page, choose the Select an existing security group option. Select the WebServerSG group that we created previously, and then choose Review and Launch.

**Step 9** − Check Instance details on Review Instance Launch page then click the Launch button.

**Step 10** − A pop up dialog box will open. Select an existing key pair or create a new key pair. Then select the acknowledgement check box and click the Launch Instances button.