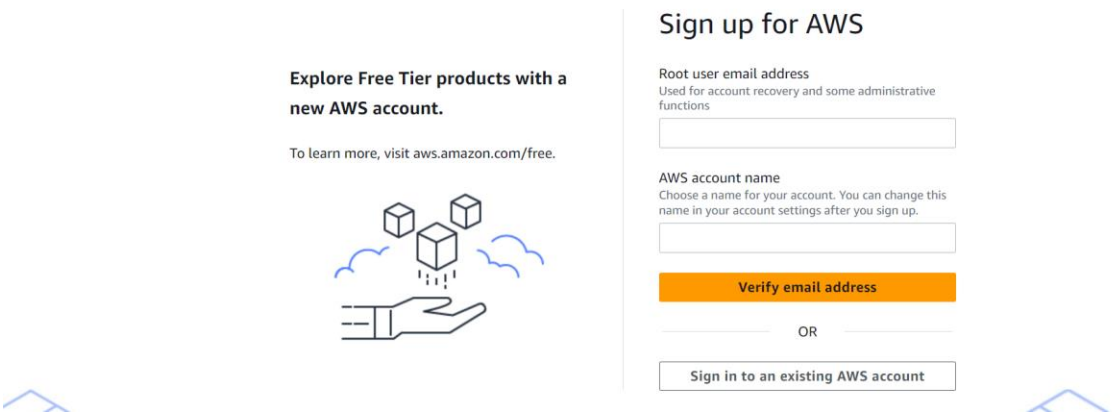
# Setting Up a LAMP Stack on an AWS Instance

By,
Neil Duraiswami

## Table of Contents

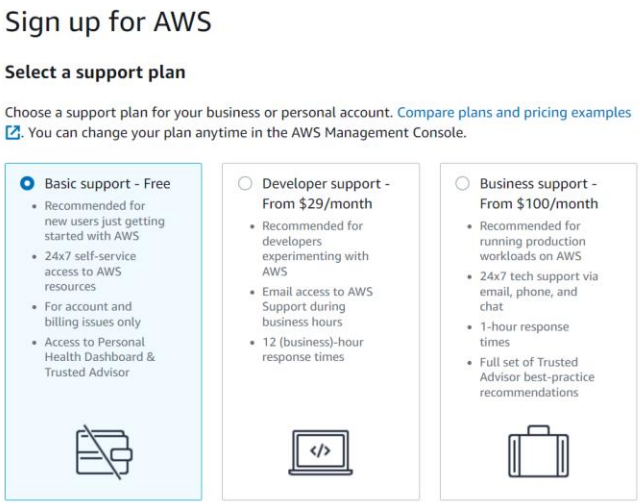
# 1. AWS Account Setup

**Step 1:** Visit the AWS website (https://aws.amazon.com/) and click "Sign in to the Console".



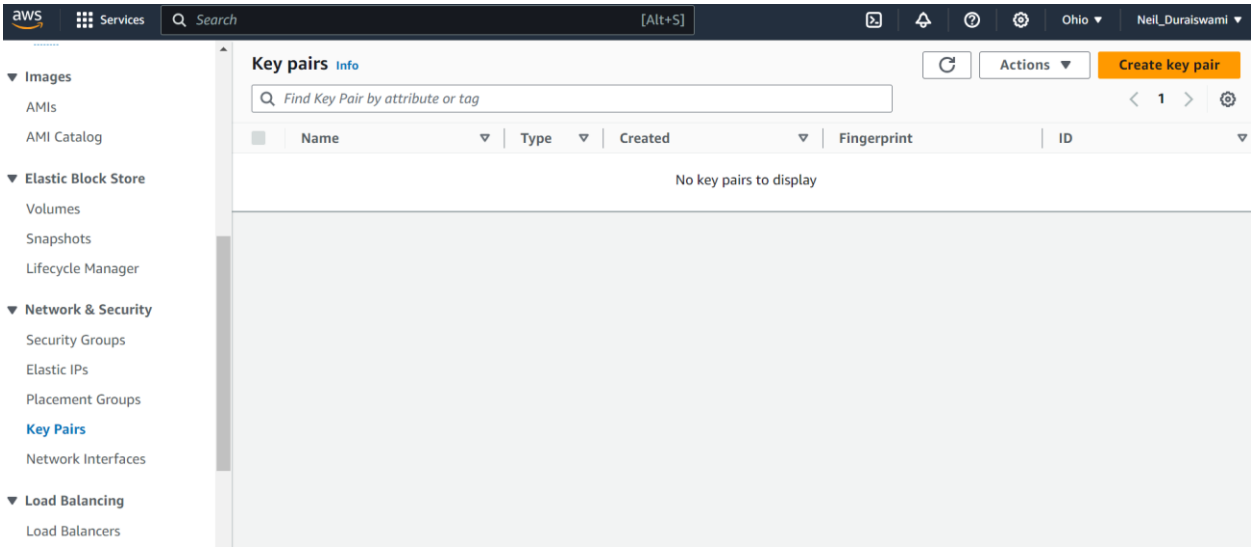

**Step 2:** Choose "Create a new AWS account" and follow the on-screen instructions to complete the account creation process, including providing personal information, payment details, and verifying identity.

**Step 3:** Once the account is set up, sign in to the AWS Management Console.



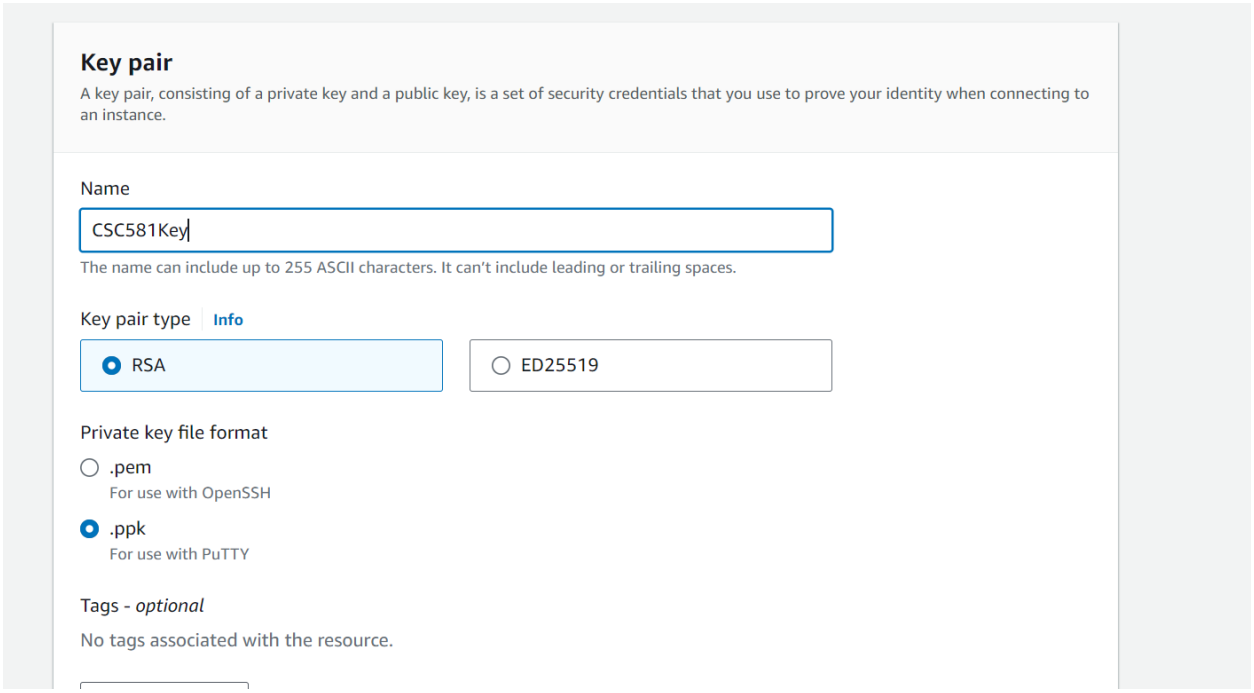

# 2. Key Pair and Security Group Creation

**Step 1:** Navigate to the EC2 Dashboard in the AWS Management Console.




**Step 2:** Create a new key pair for SSH access to the EC2 instance.

**Step 3:** Configure a security group to allow inbound traffic on port 22 (SSH) and port 80 (HTTP).




## 3. EC2 Instance Launch

**Step 1:** From the EC2 Dashboard, click on "Launch Instance".

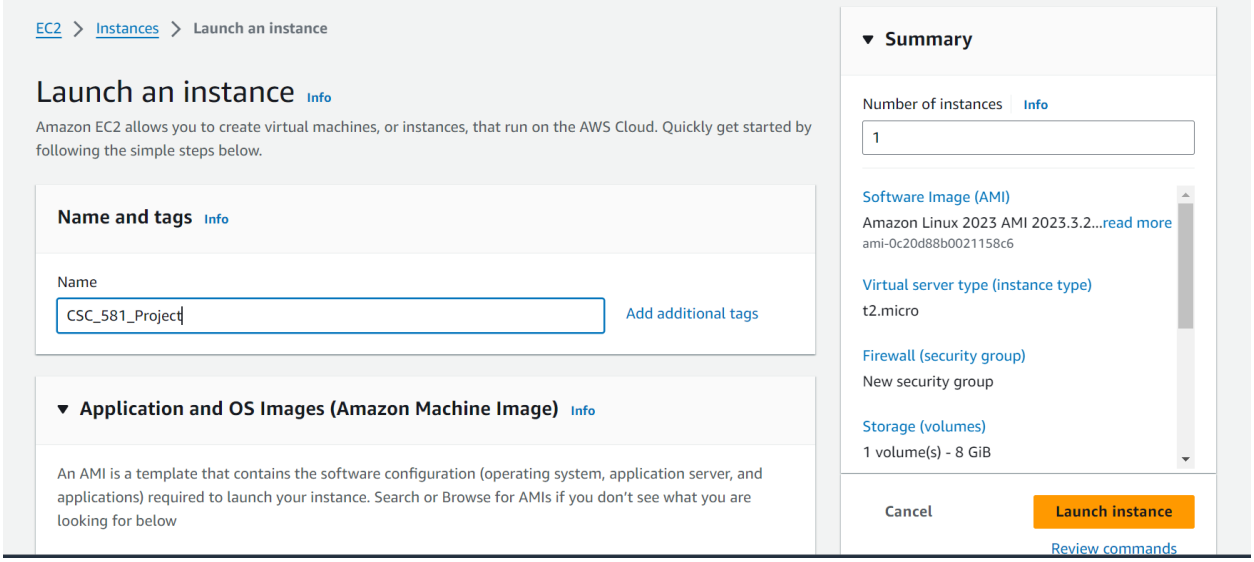
**Step 2:** Choose the "Amazon Linux 2 AMI" operating system.

**Step 3:** Select an instance type, such as t2.micro, eligible for the free tier.

**Step 4:** Configure instance details as needed and add storage.

**Step 5:** Configure the security group to use the one created in step 2.

**Step 6:** Review and launch the instance.




## 4. SSH Access via Putty

**Step 1:** Download and install Putty, a popular SSH client for Windows.

**Download PuTTY: latest release (0.80)**

Home | FAQ | Feedback | Licence | Updates | Mirrors | Keys | Links | Team
Download: **Stable** · Snapshot | Docs | Changes | Wishlist

This page contains download links for the latest released version of PuTTY. Currently this is 0.80, released on 2023-12-18.

When new releases come out, this page will update to contain the latest, so this is a good page to bookmark or link to. Alternatively, here is a permanent link to the 0.80 release.

Release versions of PuTTY are versions we think are reasonably likely to work well. However, they are often not the most up-to-date version of the code available. If you have a problem with this release, then it might be worth trying out the development snapshots, to see if the problem has already been fixed in those versions.

**Package files**

You probably want one of these. They include versions of all the PuTTY utilities (except the new and slightly experimental Windows pterm).

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

We also publish the latest PuTTY installers for all Windows architectures as a free-of-charge download at the Microsoft Store; they usually take a few days to appear there after we release them.

**MSI ('Windows Installer')**

| | | |
|---|---|---|
| 64-bit x86: | putty-64bit-0.80-installer.msi | (signature) |
| 64-bit Arm: | putty-arm64-0.80-installer.msi | (signature) |
| 32-bit x86: | putty-0.80-installer.msi | (signature) |

**Unix source archive**

| | | |
|---|---|---|
| .tar.gz: | putty-0.80.tar.gz | (signature) |

**Step 2:** Convert the downloaded key pair file (.pem) to a format compatible with Putty using PuttyGen.

**Step 3:** Open Putty and configure the session by providing the public IP address of the EC2 instance and the private key file.

**Step 4:** Connect to the EC2 instance using Putty.

## 5. LAMP Stack Installation

**Step 1:** Once connected to the EC2 instance via SSH, update the system packages with the package manager (e.g., sudo yum update).

**Step 2:** Install Apache web server: sudo yum install httpd.

**Step 3:** Start the Apache service: sudo systemctl start httpd.

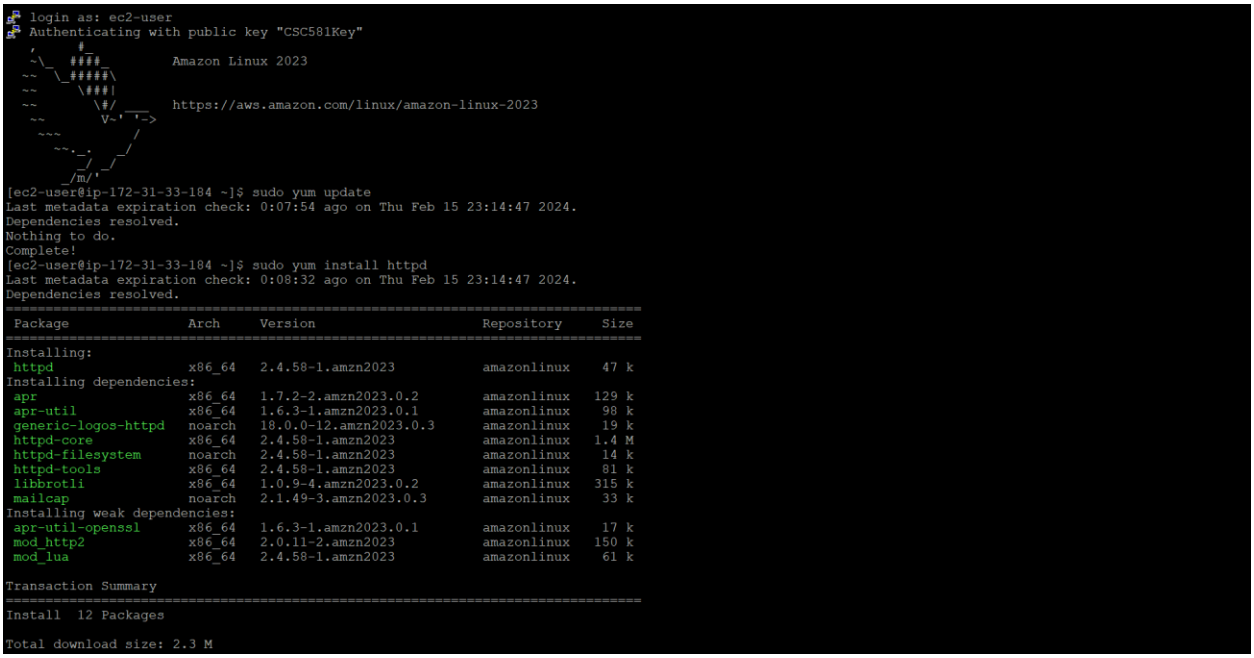
**Step 4:** Enable Apache to start on boot: sudo systemctl enable httpd.

**Step 5:** Install MySQL database server: sudo yum install mysql-server.

**Step 6:** Start the MySQL service: sudo systemctl start mysqld.

**Step 7:** Secure the MySQL installation: sudo mysql_secure_installation.

**Step 8:** Install PHP and necessary extensions: sudo yum install php php-mysql.




**Step 9:** Restart Apache to apply changes: sudo systemctl restart httpd.

## 6. Writing the Index.php File

**Step 1:** Create a simple PHP file named index.php in the Apache document root directory (/var/www/html/).

**Step 2:** Use a text editor such as nano to write the PHP script. Example: sudo nano /var/www/html/index.php.

**Step 3:** Write a basic PHP script that displays "Hello, World!".

```
<?php

echo "Hello, World!";

?>
```

## 7. Verifying Public Address

**Step 1:** Obtain the public IP address of the EC2 instance from the AWS Management Console.

**Step 2:** Access the PHP script via a web browser using the public IP address.


## Security Considerations

It is crucial to consider security when setting up a LAMP stack on an AWS instance that can be accessed by the public. To do this, always apply patches and updates to each of the components regularly to fix known vulnerabilities. Network ACLs and AWS security groups should be employed to maintain traffic coming in and going out of the system. Moreover, you must think about segmenting your network with a virtual private cloud (VPC). Ensure secure SSH access by using key-based authentication and disallowing root login. On the other hand, as you enable HTTPS through SSL/TLS certificates, remember to set up Apache so that it does not allow for directory listing and limits the number of concurrent connections. Lastly, restrict MySQL access rights and create strong passwords for it all. It must follow the best PHP security practices such as output sanitization and input validation. This means turning on the monitoring system to help identify any suspicious activities happening within the networks while ensuring disaster recovery setups are correctly done during routine backups. So, we need a strong defense against viruses by conducting safety audits to recognize and handle possible weaknesses at times like these.

Here are three of the main practices and considerations when setting up a LAMP stack on a public-facing AWS instance:

1. **Encryption**
   Encryption is vital for securing data in transit and at rest. SSL/TLS certificates ensure secure communication between clients and the web server. AWS offers encryption options like SSE for S3 buckets and encrypted EBS volumes for data storage protection.

2. **Backup and Disaster Recovery**

A robust backup strategy and disaster recovery plan are essential. Automated backups of data and configurations, along with snapshots of EC2 instances and RDS databases, ensure quick recovery from data loss or system failures. Cross-region replication enhances data durability.

3. **Network Security**
   Effective network security controls are crucial for safeguarding public-facing AWS instances. AWS Security Groups and NACLs control inbound and outbound traffic, while VPC isolation provides network segmentation. AWS Shield protects against DDoS attacks, and CloudWatch enables real-time monitoring and logging for proactive threat detection.

## References

**AWS Documentation:** https://docs.aws.amazon.com/

**Putty Documentation:** https://www.putty.org/