

Project: Deploying WordPress Server and EC2 Instance

Project: AWS Deploy WordPress Server

Aryan Vij

Date Started: 6/10/2024

Date Completed: 6/10/2024

Table of Contents

EXECUTIVE SUMMARY	3
PROCEDURES	3
Launch an EC2 instance	4
Install Apache2, MYSQL and PHP	4
Setup MySQL Database for WordPress	5
Download and Configure WordPress	5
Configure Apache for WordPress	6
Set Up DNS with Route 53	6
Verify Domain Ownership with Google and Set Up SSL with ACM	6
Configure Health Check and Application Load Balancer	6
Finalise Deployment	7
Configure Security Groups and Routing	7
RESULTS	8
BIBLIOGRAPHY	10
APPENDICES	11
Appendix A: Visual Documentation	11

EXECUTIVE SUMMARY

This document outlines the comprehensive procedures for deploying a static WordPress website on an AWS EC2 instance using Apache2 and MySQL, ensuring a robust and scalable web presence for aryanvijcyberblog.com. The deployment process is segmented into key phases, including setting up the server environment, configuring necessary software, establishing DNS records, and implementing security measures.

First, an Amazon Linux 2 EC2 instance is launched with appropriate security groups. Apache2, MySQL, and PHP are installed and configured to create a reliable hosting environment. Following this, WordPress is downloaded and installed on the EC2 instance. The wp-config.php file is configured with the database credentials, and Apache2 settings are adjusted to support WordPress, ensuring smooth operation and management.

Next, DNS and SSL configurations are addressed. AWS Route 53 is used to create and manage DNS records for aryanvijcyberblog.com. Domain ownership is verified with Google, and AWS Certificate Manager (ACM) is utilised to request and configure an SSL certificate, securing the website.

An Application Load Balancer (ALB) is then deployed to distribute traffic efficiently across multiple instances, with health checks configured to monitor instance health and ensure high availability. Lastly, security measures and routing configurations are put in place. Security groups are configured to allow necessary inbound and outbound traffic, and routing rules are established to ensure efficient traffic management.

By following these procedures, aryanvijcyberblog.com will have a secure, scalable, and well-managed WordPress deployment on AWS.

PROCEDURES

Prerequisites

1. AWS Account: Ensure you have an AWS account.
2. Domain: Own the domain aryanvijcyberblog.com.

Launch an EC2 instance

1. Open the AWS Management Console and navigate to the EC2 Dashboard.
2. Launch Instance:
 - a. Select Amazon Linux 2 AMI (or your preferred Linux distribution).
 - b. Choose an instance type (e.g., t2.micro for free tier).
 - c. Configure instance details (e.g., default settings for a single instance).
 - d. Add storage (e.g., 8 GB of General Purpose SSD).
 - e. Add tags (optional).
 - f. Configure security group:
 - i. Allow HTTP (port 80) and SSH (port 22).
 - ii. Allow HTTPS (port 443) if planning to use SSL.
3. Launch the instance and download the key pair (.pem file).

Install Apache2, MYSQL and PHP

1. Connect to your EC2 instance using SSH: `ssh -i "your-key-pair.pem" ec2-user@your-ec2-instance-public-dns`
2. Update packages: `sudo yum update -y`
3. Install Apache2:
 - a. `sudo yum install httpd -y`
 - b. `sudo systemctl start httpd`
 - c. `sudo systemctl enable httpd`
4. Install MySQL:
 - a. `sudo yum install mysql-server -y`
 - b. `sudo systemctl start mysqld`
 - c. `sudo systemctl enable mysqld`
5. Secure MySQL Installation:
 - a. `sudo mysql_secure_installation`
6. Install PHP and required extensions
 - a. `sudo yum install php php-mysql php-fpm -y`
 - b. `sudo systemctl restart httpd`

Setup MySQL Database for WordPress

1. Login to MySQL:

```
a. sudo mysql -u root -p
```

2. Create WordPress database and user:

```
a. CREATE DATABASE wordpress;  
b. CREATE USER 'wordpressuser'@'localhost' IDENTIFIED BY  
    'yourpassword';  
c. GRANT ALL PRIVILEGES ON wordpress.* TO  
    'wordpressuser'@'localhost';  
d. FLUSH PRIVILEGES;  
e. EXIT;
```

Download and Configure WordPress

1. Download WordPress:

```
a. cd /var/www/html  
b. sudo wget https://wordpress.org/latest.tar.gz  
c. sudo tar -xzf latest.tar.gz  
d. sudo rm latest.tar.gz  
e. sudo chown -R apache:apache /var/www/html/wordpress
```

2. Configure 'wp-config.php'

```
a. sudo cp /var/www/html/wordpress/wp-config-sample.php  
    /var/www/html/wordpress/wp-config.php  
b. sudo nano /var/www/html/wordpress/wp-config.php
```

3. Update the database details:

```
a. define('DB_NAME', 'wordpress');  
b. define('DB_USER', 'wordpressuser');  
c. define('DB_PASSWORD', 'yourpassword');  
d. define('DB_HOST', 'localhost');
```

4. Restart Apache;

```
a. sudo systemctl restart httpd
```

Configure Apache for WordPress

1. Edit Apache Configuration:
 - a. `sudo nano /etc/httpd/conf/httpd.conf`
2. Add the following lines:
 - a. `<Directory "/var/www/html/wordpress">`
 - b. `AllowOverride All`
 - c. `</Directory>`
3. Enable mod_rewrite:
 - a. `sudo a2enmod rewrite`
 - b. `sudo systemctl restart httpd`

Set Up DNS with Route 53

1. Navigate to Route 53 in the AWS Management Console.
2. Create a hosted zone for aryanvijcyberblog.com.
3. Create DNS records:
 - a. A record pointing to your EC2 instance's public IP.
 - b. CNAME record for www.aryanvijcyberblog.com pointing to aryanvijcyberblog.com.

Verify Domain Ownership with Google and Set Up SSL with ACM

1. Verify domain ownership through Google Search Console.
2. Request an SSL certificate in ACM:
 - a. Navigate to ACM in the AWS Console.
 - b. Request a public certificate for aryanvijcyberblog.com and www.aryanvijcyberblog.com.
 - c. Use DNS validation and add the provided CNAME records to Route 53.
 - d. Validate and issue the certificate.

Configure Health Check and Application Load Balancer

1. Create a Health Check in Route 53:
 - a. Navigate to Route 53 > Health Checks.

Project: Deploying WordPress Server and EC2 Instance

- b. Create a health check for aryanvijcyberblog.com.
2. Set up an Application Load Balancer (ALB):
 - a. Navigate to the EC2 Dashboard > Load Balancers.
 - b. Create a new Application Load Balancer.
 - c. Configure the ALB with at least two subnets in your VPC.
 - d. Add a listener for HTTP (and HTTPS if using SSL).
 - e. Create a target group for your EC2 instances.
3. Register an EC2 instance with the target group.
4. Configure the ALB to use the SSL certificate from ACM.
5. Update security groups:
 - a. Ensure the EC2 instance's security group allows incoming traffic from the ALB.
 - b. The ALB's security group should allow incoming traffic on ports 80 and 443.

Finalise Deployment

1. Update DNS records in Route 53:
 - a. Point the A record to the ALB DNS name.
2. Verify the setup:
 - a. Visit <http://aryanvijcyberblog.com> to ensure the WordPress site loads correctly.
 - b. Ensure the SSL certificate is working (if applicable).

Configure Security Groups and Routing

1. Security Groups:
 - a. EC2 security group: Allow inbound HTTP (80), HTTPS (443), and SSH (22) from trusted sources.
 - b. ALB security group: Allow inbound traffic on ports 80 and 443.
2. Routing:
 - a. Configure the ALB listener rules to route traffic to the target group.

RESULTS

Upon completion of the deployment process, aryanvijcyberblog.com is successfully hosted on an AWS EC2 instance with a robust and scalable architecture. The implementation includes several key achievements:

Stable Hosting Environment: The EC2 instance with Amazon Linux 2, configured with Apache2, MySQL, and PHP, provides a stable and reliable hosting environment for the WordPress site. The wp-config.php and Apache2 configurations ensure seamless interaction between the web server and the database.

Efficient DNS Management: AWS Route 53 effectively manages DNS records, ensuring that aryanvijcyberblog.com is easily accessible to users. The DNS setup provides reliable name resolution and supports the website's operational needs.

Secured Website: Domain ownership verification with Google and the issuance of an SSL certificate via AWS Certificate Manager (ACM) secure the website, ensuring encrypted connections for users. The SSL setup enhances user trust and meets industry security standards.

Optimised Traffic Distribution: The Application Load Balancer (ALB) distributes incoming traffic efficiently across multiple instances, ensuring high availability and reliability. Configured health checks monitor the status of instances, allowing automatic rerouting of traffic in case of instance failure, thus maintaining consistent uptime.

Enhanced Security: Security groups are meticulously configured to allow only necessary inbound and outbound traffic, enhancing the security posture of the deployment. Proper routing and access controls protect the EC2 instance and associated resources from unauthorised access.

High Availability and Scalability: The deployment architecture supports scalability, allowing for the addition of more EC2 instances behind the ALB as traffic demands

Project: Deploying WordPress Server and EC2 Instance

increase. The configuration ensures that the website remains highly available, providing users with a reliable browsing experience.

In summary, the deployment of aryanvijcyberblog.com on AWS using EC2, Apache2, MySQL, Route 53, ACM, and an ALB results in a secure, scalable, and highly available WordPress site. The meticulous configuration of each component ensures robust performance and a positive user experience.

BIBLIOGRAPHY

DNS Service - Amazon Route 53 - AWS. (n.d.). Amazon Web Services, Inc.

<https://aws.amazon.com/route53/>

DNS validation - AWS Certificate Manager. (n.d.).

<https://docs.aws.amazon.com/acm/latest/userguide/dns-validation.html>

GeeksforGeeks. (2024, April 23). *Introduction to Amazon Route53.* GeeksforGeeks.

<https://www.geeksforgeeks.org/introduction-to-amazon-route53/>

Install and Configure Apache | Ubuntu. (n.d.). Ubuntu.

<https://ubuntu.com/tutorials/install-and-configure-apache#1-overview>

Jain, A. (2024, May 20). AWS Certificate Manager ACM: Overview, Features and How it Works? *Cloud Training Program.*

<https://k21academy.com/amazon-web-services/aws-certificate-manager-acm/>

Tutorial: Deploy WordPress to an Amazon EC2 instance (Amazon Linux or Red Hat Enterprise Linux and Linux, macOS, or Unix) - AWS CodeDeploy. (n.d.).

<https://docs.aws.amazon.com/codedeploy/latest/userguide/tutorials-wordpress.html>

Verify your domain with a TXT record - Google Workspace Admin Help. (n.d.).

<https://support.google.com/a/answer/183895?hl=en>

APPENDICES

Appendix A: Visual Documentation

These visuals offer a detailed representation of the tools used, commands executed, and findings discovered during the assessment. By including screenshots, readers can gain a clear understanding of the methodologies employed and the outcomes achieved during the testing phase.

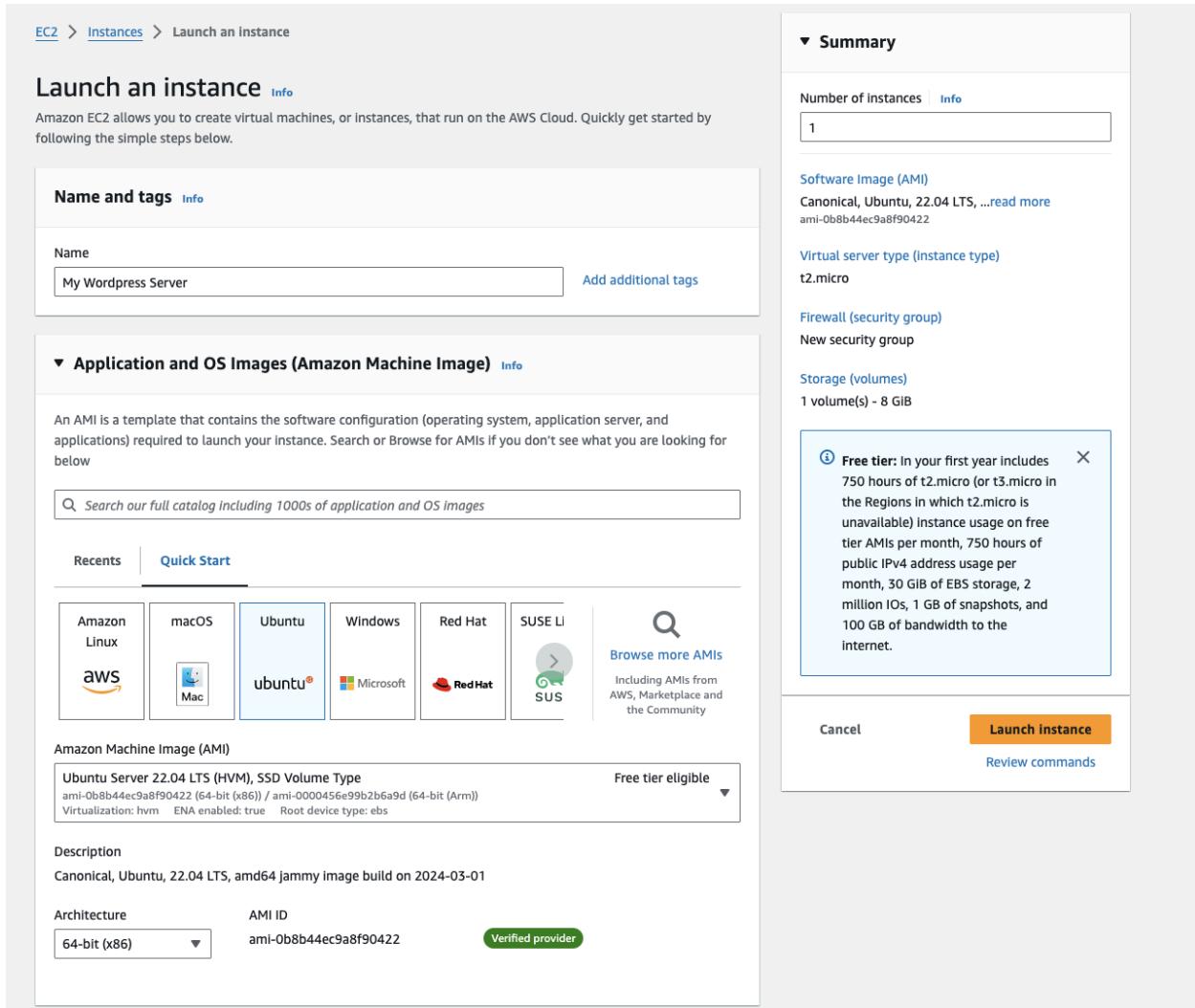


Figure 1: Launching EC2 instance

Project: Deploying WordPress Server and EC2 Instance

Instances (1/2) Info

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
<input checked="" type="checkbox"/> My Wordpress ...	i-0698ae003d93166e3	Running	t2.micro	2/2 checks passed	View alarms	us-east-2b	ec2-18-116-102-120.us.e...
<input type="checkbox"/>	server-1-us-ea...	Running	t2.micro	2/2 checks passed	View alarms	us-east-2a	ec2-3-23-101-249.us.e...

Instance: i-0698ae003d93166e3 (My Wordpress Server)

Details | [Status and alarms New](#) | [Monitoring](#) | [Security](#) | [Networking](#) | [Storage](#) | [Tags](#)

Instance summary Info

Instance ID i-0698ae003d93166e3 (My Wordpress Server)	Public IPv4 address 18.116.102.120 [open address]	Private IPv4 addresses 10.0.2.200
IPv6 address -	Instance state Running	Public IPv4 DNS ec2-18-116-102-120.us-east-2.compute.amazonaws.com [address]
Hostname type IP name: ip-10-0-2-200.us-east-2.compute.internal	Private IP DNS name (IPv4 only) ip-10-0-2-200.us-east-2.compute.internal	Elastic IP addresses 18.116.102.120 [Public IP]
Answer private resource DNS name IPv4 (A)	Instance type t2.micro	AWS Compute Optimizer finding Opt-in to AWS Compute Optimizer for recommendations. Learn more
Auto-assigned IP address -	VPC ID vpc-007ca4fccbf9a3a0 (VPC Personal Project)	Auto Scaling Group name -
IAM Role -	Subnet ID subnet-0fbeca3d7f4e1b9d7 (vpc-public-us-east-2b)	
IMDSv2 Required	Platform	AMI ID

Instance details Info

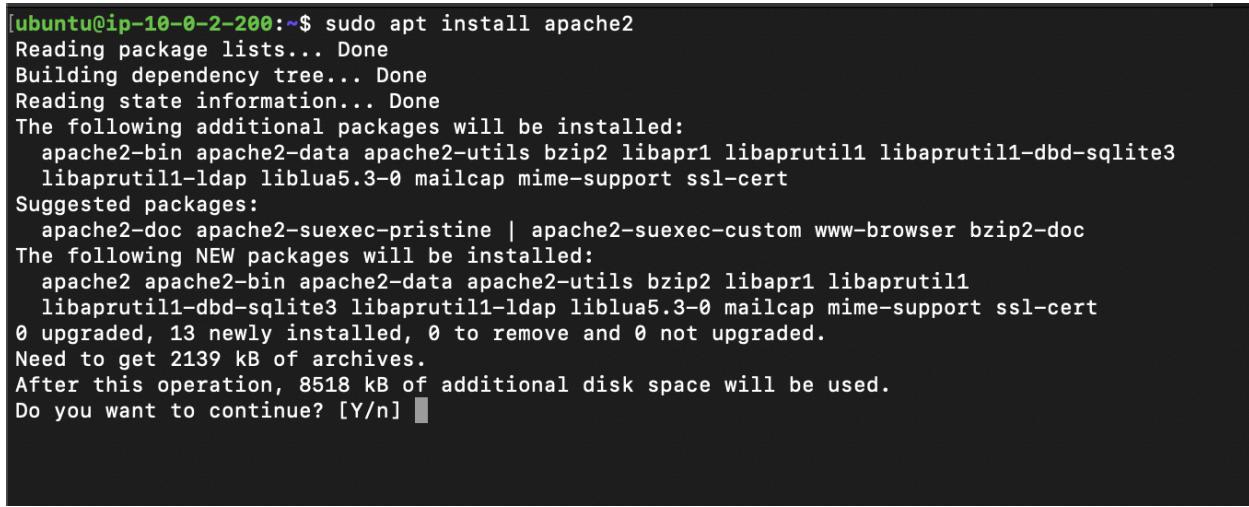
Figure 2: Wordpress instance configurations

Project: Deploying WordPress Server and EC2 Instance



The screenshot shows a terminal window titled 'Downloads — ubuntu@ip-10-0-2-200: ~ — ssh -i wordpress_server.pem ubuntu@ec2-18-1...'. The command entered was 'ssh -i wordpress_server.pem ubuntu@ec2-18-116-102-120.us-east-2.compute.amazonaws.com'. The prompt '[ubuntu@ip-10-0-2-200: ~]\$' is visible at the bottom.

Figure 3: Wordpress ssh



```
[ubuntu@ip-10-0-2-200: ~]$ sudo apt install apache2
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  apache2-bin apache2-data apache2-utils bzip2 libapr1 libaprutil1 libaprutil1-dbd-sqlite3
    libaprutil1-ldap liblua5.3-0 mailcap mime-support ssl-cert
Suggested packages:
  apache2-doc apache2-suexec-pristine | apache2-suexec-custom www-browser bzip2-doc
The following NEW packages will be installed:
  apache2 apache2-bin apache2-data apache2-utils bzip2 libapr1 libaprutil1
    libaprutil1-dbd-sqlite3 libaprutil1-ldap liblua5.3-0 mailcap mime-support ssl-cert
0 upgraded, 13 newly installed, 0 to remove and 0 not upgraded.
Need to get 2139 kB of archives.
After this operation, 8518 kB of additional disk space will be used.
Do you want to continue? [Y/n]
```

Figure 4: Apache2 Installation

Project: Deploying WordPress Server and EC2 Instance

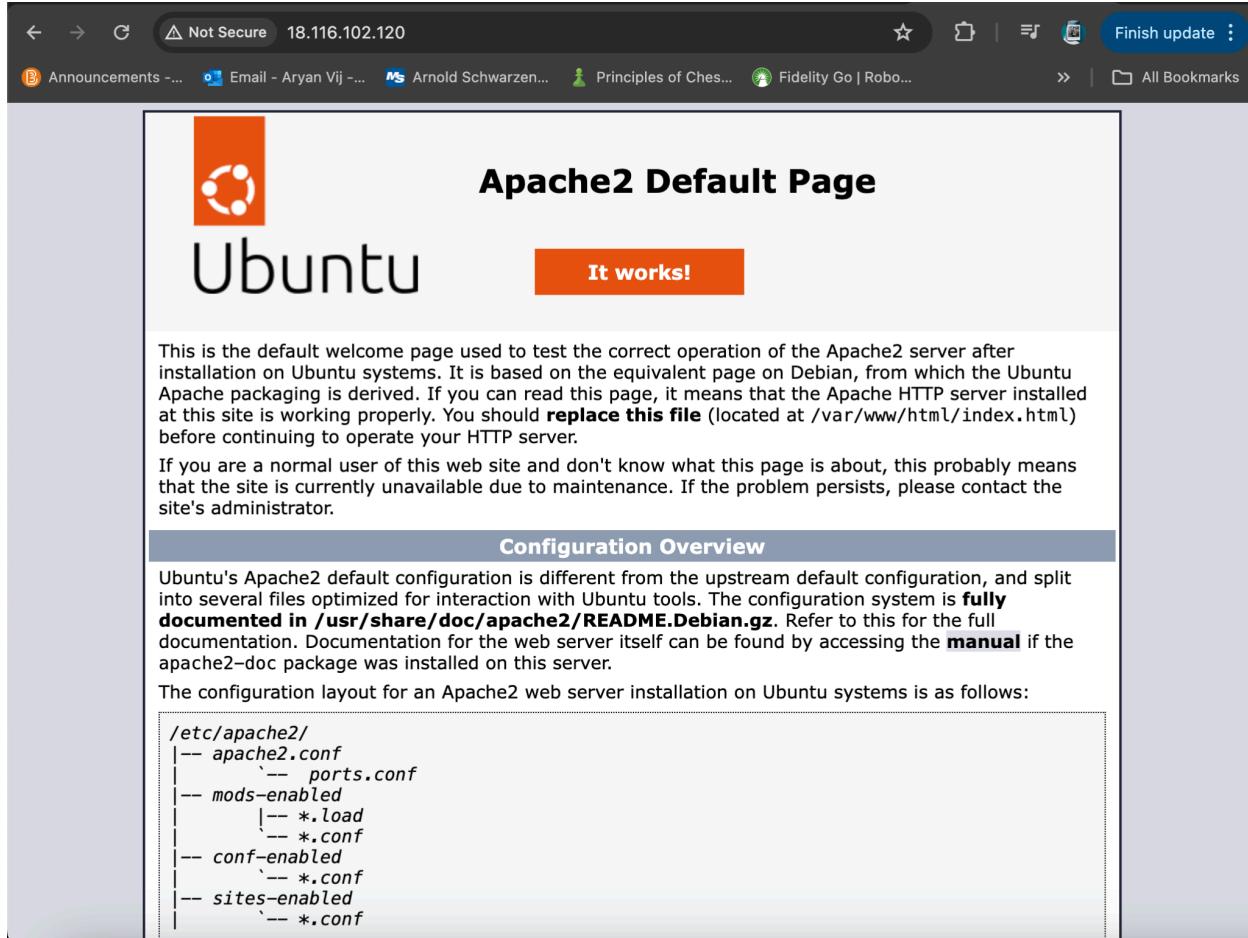


Figure 5: Successful Apache2 launch

Project: Deploying WordPress Server and EC2 Instance

```
[ubuntu@ip-10-0-2-200:~$ sudo apt install php libapache2-mod-php php-mysql
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libapache2-mod-php8.1 php-common php8.1 php8.1-cli php8.1-common php8.1-mysql php8.1-opcache
  php8.1-readline
Suggested packages:
  php-pear
The following NEW packages will be installed:
  libapache2-mod-php libapache2-mod-php8.1 php php-common php-mysql php8.1 php8.1-cli
  php8.1-common php8.1-mysql php8.1-opcache php8.1-readline
0 upgraded, 11 newly installed, 0 to remove and 52 not upgraded.
Need to get 5265 kB of archives.
After this operation, 21.8 MB of additional disk space will be used.
[Do you want to continue? [Y/n] Y
Get:1 http://us-east-2.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 php-common all 2:92ubuntu1
 [12.4 kB]
Get:2 http://us-east-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 php8.1-common amd6
4 8.1.2-1ubuntu2.15 [1127 kB]
Get:3 http://us-east-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 php8.1-opcache amd
64 8.1.2-1ubuntu2.15 [365 kB]
Get:4 http://us-east-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 php8.1-readline am
d64 8.1.2-1ubuntu2.15 [13.6 kB]
Get:5 http://us-east-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 php8.1-cli amd64 8
.1.2-1ubuntu2.15 [1833 kB]
Get:6 http://us-east-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libapache2-mod-php
8.1 amd64 8.1.2-1ubuntu2.15 [1766 kB]
Get:7 http://us-east-2.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 libapache2-mod-php all 2:8
.1+92ubuntu1 [2898 B]
Get:8 http://us-east-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 php8.1 all 8.1.2-1
ubuntu2.15 [9156 B]
Get:9 http://us-east-2.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 php all 2:8.1+92ubuntu1 [2
756 B]
Get:10 http://us-east-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 php8.1-mysql amd6
4 8.1.2-1ubuntu2.15 [130 kB]
Get:11 http://us-east-2.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 php-mysql all 2:8.1+92uba
ntu1 [1834 B]
Fetched 5265 kB in 0s (46.1 MB/s)
Selecting previously unselected package php-common.
(Reading database ... 66042 files and directories currently installed.)
Preparing to unpack .../00-php-common_2%3a92ubuntu1_all.deb ...
```

Figure 6: PHP installation

Project: Deploying WordPress Server and EC2 Instance

```
[ubuntu@ip-10-0-2-200:~$ sudo mysql -u root
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.36-0ubuntu0.22.04.1 (Ubuntu)

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement

mysql> ALTER USER 'root'@localhost IDENTIFIED WITH mysql_native_password BY
```

Figure 7: MySQL root login

```
[mysql> ALTER USER 'root'@localhost IDENTIFIED WITH my
Query OK, 0 rows affected (0.01 sec)

[mysql> CREATE USER 'wb_user'@localhost IDENTIFIED BY
Query OK, 0 rows affected (0.02 sec)

[mysql> CREATE DATABASE wp;
Query OK, 1 row affected (0.01 sec)

[mysql> GRANT ALL PRIVILEGES ON wp.* TO 'wp_user'@loca
ERROR 1410 (42000): You are not allowed to create a u
[mysql> GRANT ALL PRIVILEGES ON wp.* TO 'wb_user'@loca
Query OK, 0 rows affected (0.00 sec)

mysql> ]
```

Figure 8: MySQL database configurations

Project: Deploying WordPress Server and EC2 Instance

```
[ubuntu@ip-10-0-2-200:/tmp$ ls
latest.tar.gz
snap-private-tmp
systemd-private-632d9618099b48d799480d86c60535ac-apache2.service-TlbWVH
systemd-private-632d9618099b48d799480d86c60535ac-chrony.service-DmnmmYz
systemd-private-632d9618099b48d799480d86c60535ac-systemd-logind.service-1epy4Y
systemd-private-632d9618099b48d799480d86c60535ac-systemd-resolved.service-ixqWim
tmp.caGHHRNyZw
wordpress
ubuntu@ip-10-0-2-200:/tmp$ ]
```

Figure 9: Wordpress listed

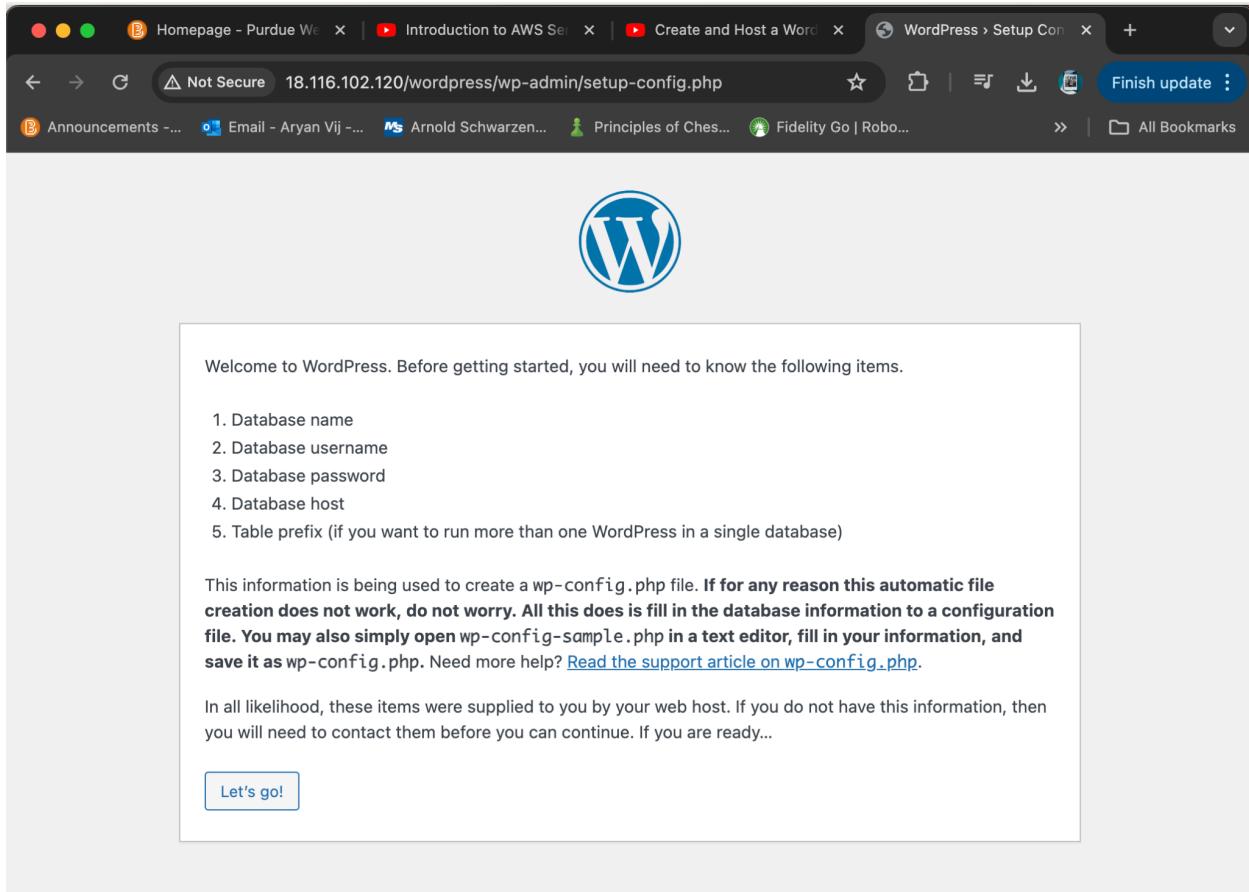


Figure 10: Wordpress wp-config finalised

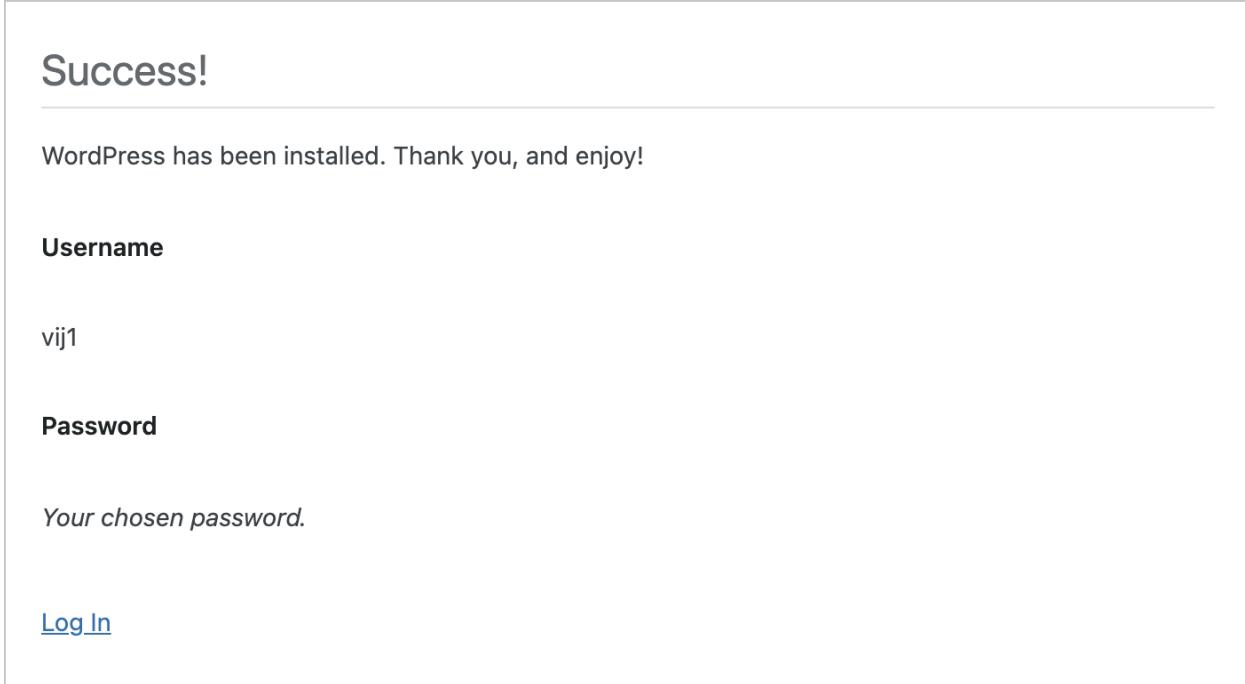


Figure 11: Wordpress successfully installed on web

Project: Deploying WordPress Server and EC2 Instance

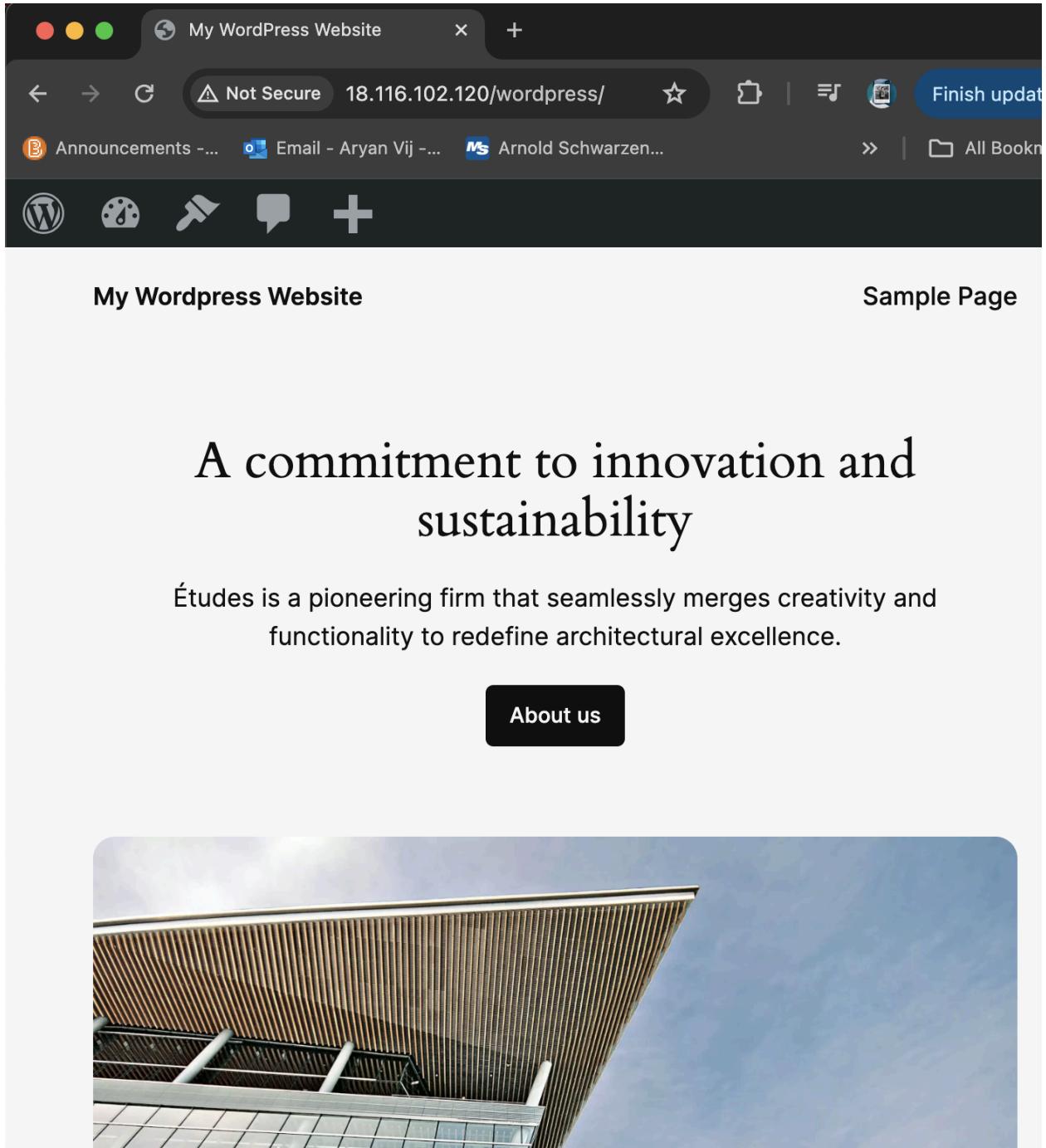
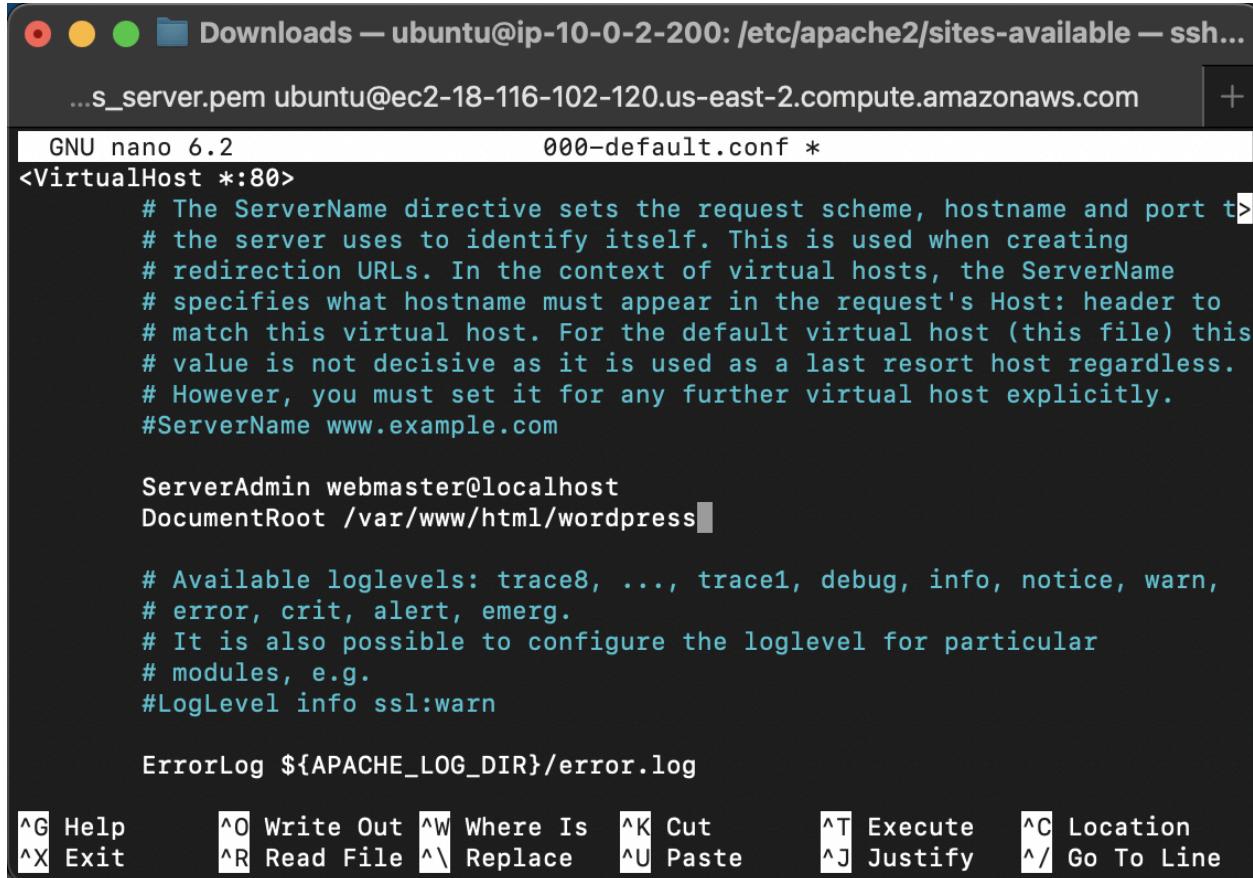


Figure 12: Site launch

Project: Deploying WordPress Server and EC2 Instance



The screenshot shows a terminal window titled "Downloads — ubuntu@ip-10-0-2-200: /etc/apache2/sites-available — ssh...". The command "...s_server.pem" is run to connect to the server. The file being edited is "000-default.conf". The content of the file is as follows:

```
GNU nano 6.2          000-default.conf *
<VirtualHost *:80>
    # The ServerName directive sets the request scheme, hostname and port t>
    # the server uses to identify itself. This is used when creating
    # redirection URLs. In the context of virtual hosts, the ServerName
    # specifies what hostname must appear in the request's Host: header to
    # match this virtual host. For the default virtual host (this file) this
    # value is not decisive as it is used as a last resort host regardless.
    # However, you must set it for any further virtual host explicitly.
    #ServerName www.example.com

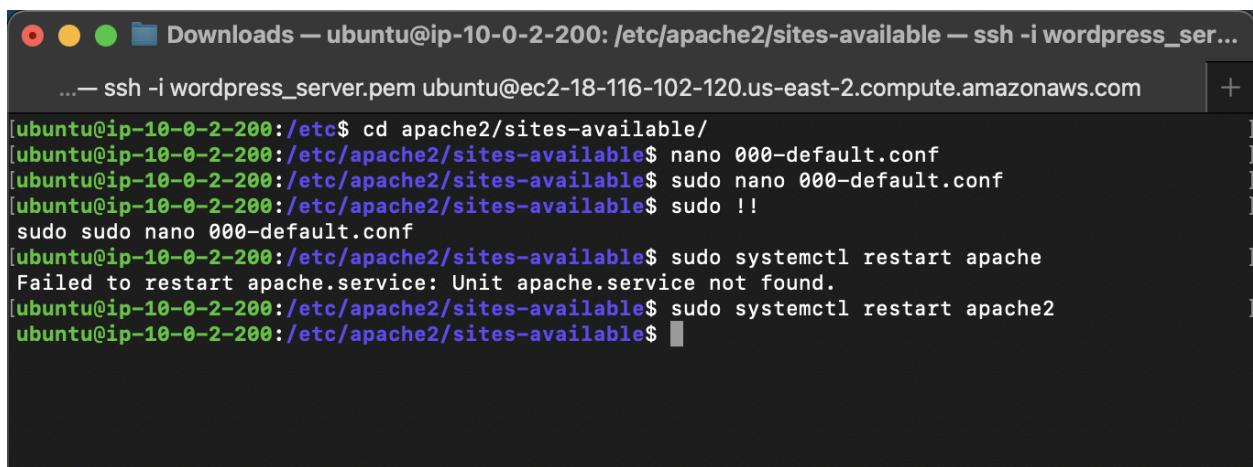
    ServerAdmin webmaster@localhost
    DocumentRoot /var/www/html/wordpress

    # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
    # error, crit, alert, emerg.
    # It is also possible to configure the loglevel for particular
    # modules, e.g.
    #LogLevel info ssl:warn

    ErrorLog ${APACHE_LOG_DIR}/error.log

^G Help      ^O Write Out  ^W Where Is  ^K Cut      ^T Execute  ^C Location
^X Exit      ^R Read File  ^\ Replace   ^U Paste    ^J Justify  ^/ Go To Line
```

Figure 13: 000-default.conf configuration

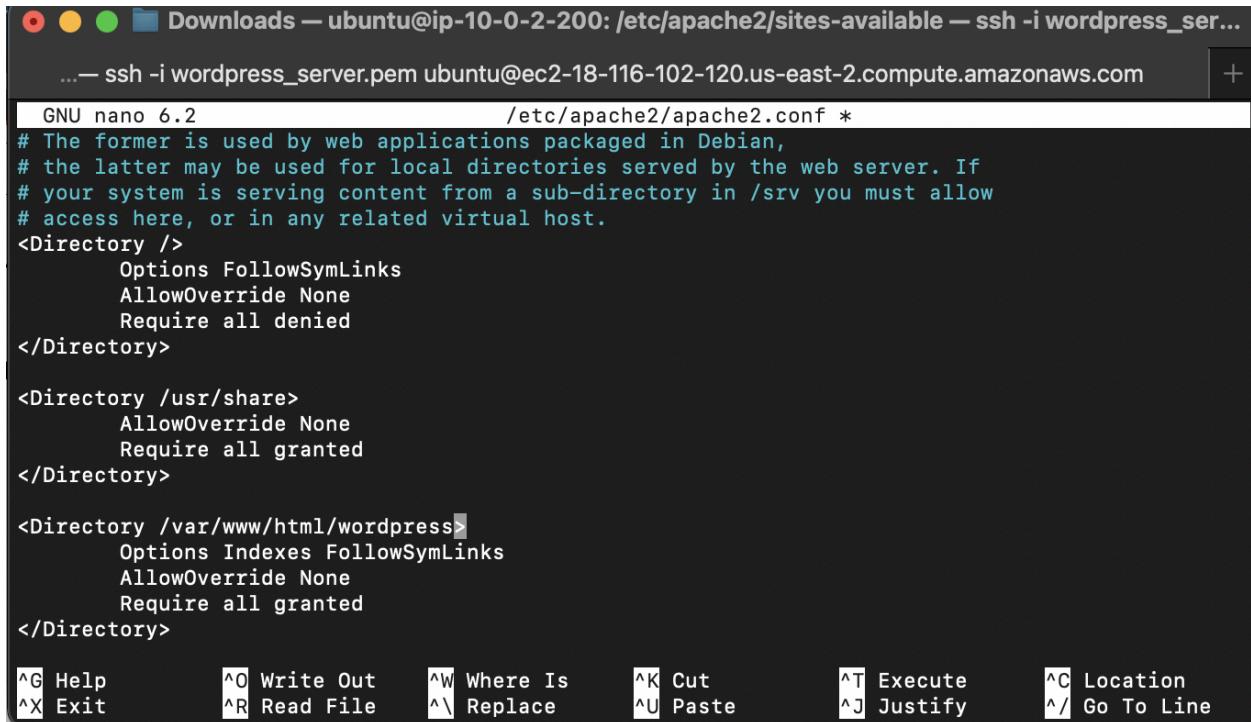


The screenshot shows a terminal window titled "Downloads — ubuntu@ip-10-0-2-200: /etc/apache2/sites-available — ssh -i wordpress_ser...". The command "... — ssh -i wordpress_server.pem" is run to connect to the server. The user then runs several commands to manage the Apache2 service:

```
[ubuntu@ip-10-0-2-200:/etc$ cd apache2/sites-available/
[ubuntu@ip-10-0-2-200:/etc/apache2/sites-available$ nano 000-default.conf
[ubuntu@ip-10-0-2-200:/etc/apache2/sites-available$ sudo nano 000-default.conf
[ubuntu@ip-10-0-2-200:/etc/apache2/sites-available$ sudo !!
sudo nano 000-default.conf
[ubuntu@ip-10-0-2-200:/etc/apache2/sites-available$ sudo systemctl restart apache
Failed to restart apache.service: Unit apache.service not found.
[ubuntu@ip-10-0-2-200:/etc/apache2/sites-available$ sudo systemctl restart apache2
ubuntu@ip-10-0-2-200:/etc/apache2/sites-available$ ]
```

Figure 14: Apache2 restart

Project: Deploying WordPress Server and EC2 Instance



```
GNU nano 6.2          /etc/apache2/apache2.conf *
# The former is used by web applications packaged in Debian,
# the latter may be used for local directories served by the web server. If
# your system is serving content from a sub-directory in /srv you must allow
# access here, or in any related virtual host.
<Directory />
    Options FollowSymLinks
    AllowOverride None
    Require all denied
</Directory>

<Directory /usr/share>
    AllowOverride None
    Require all granted
</Directory>

<Directory /var/www/html/wordpress>
    Options Indexes FollowSymLinks
    AllowOverride None
    Require all granted
</Directory>

^G Help      ^O Write Out   ^W Where Is   ^K Cut        ^T Execute   ^C Location
^X Exit      ^R Read File   ^\ Replace    ^U Paste     ^J Justify   ^/ Go To Line
```

Figure 15: Root directory modification

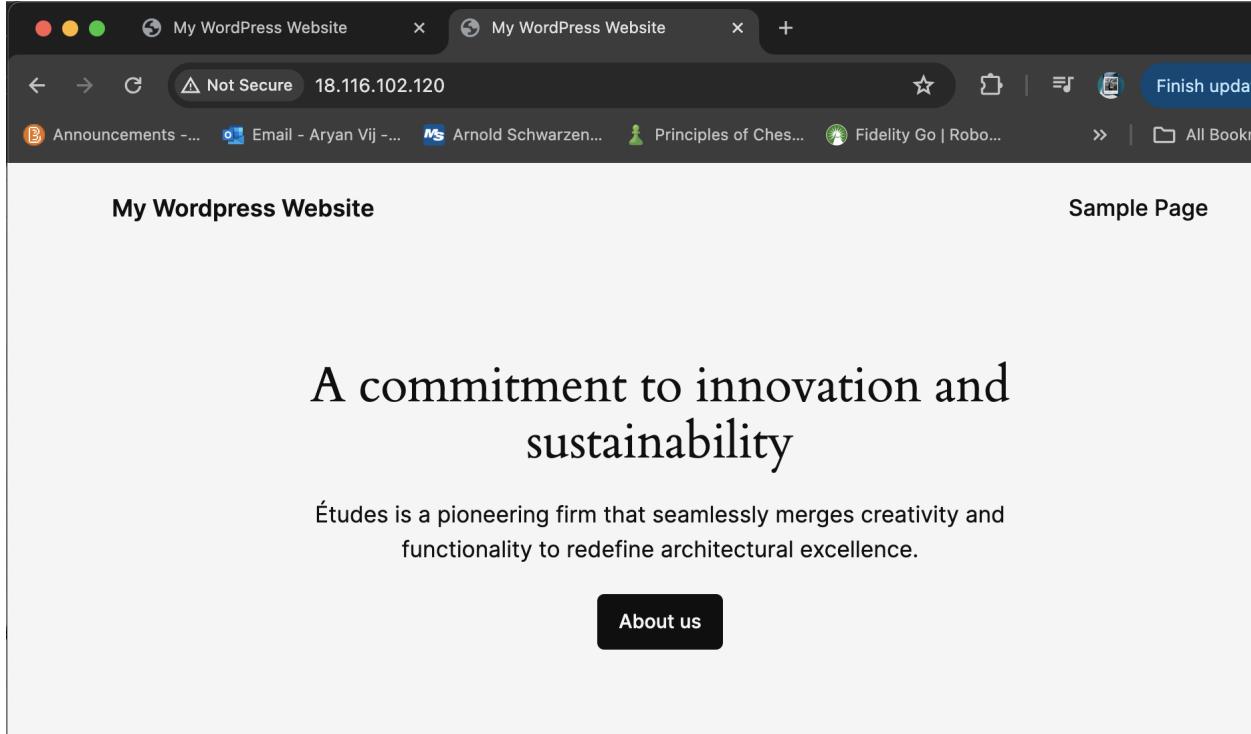


Figure 16: Default page changed

Project: Deploying WordPress Server and EC2 Instance

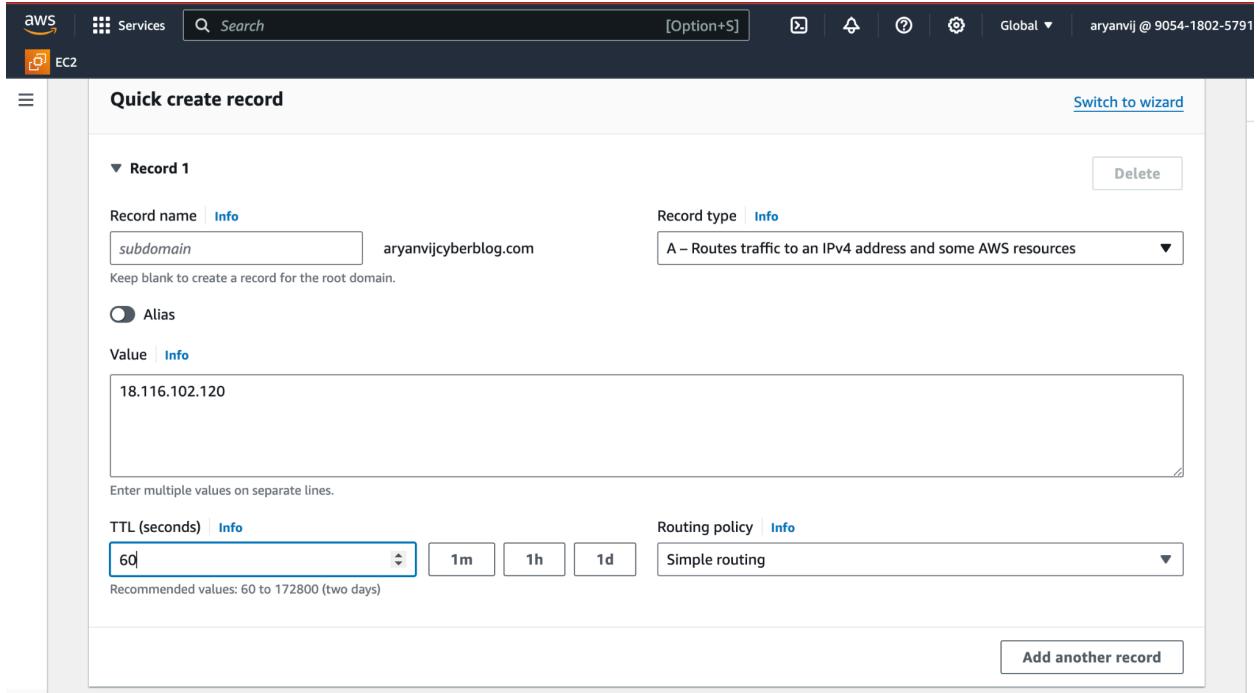


Figure 17: DNS record configuration

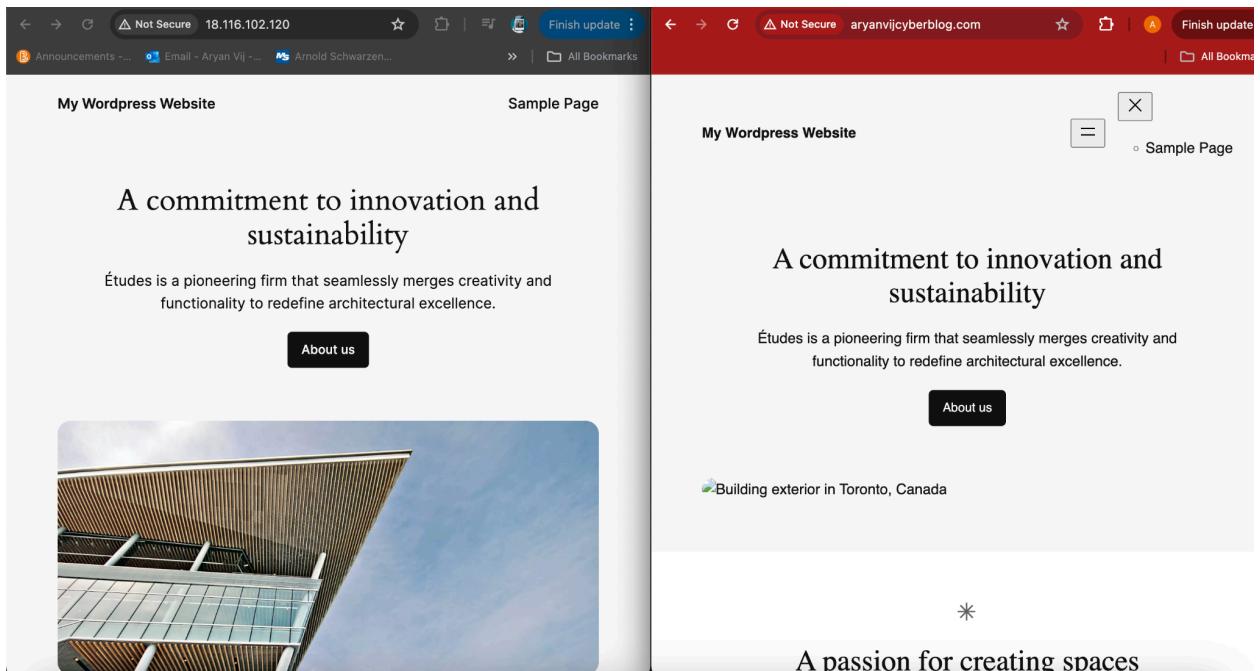


Figure 18: Accessing site via IP and domain

Project: Deploying WordPress Server and EC2 Instance



Figure 19: Domain ownership verified

A screenshot of the AWS Certificate Manager interface, specifically the "Details" section for a certificate. The certificate identifier is "2720dfe4-fabd-47fe-b7f3-ab1a5b35da77". The "Domains" table shows one domain entry: "www.aryanvijcyberblog.com" with a status of "Pending validation". The "Details" table provides more information about the certificate, including its serial number (N/A), requested date (April 22, 2024, 15:53:53 UTC-04:00), and renewal eligibility (Ineligible). The "Validating domain ownership" sidebar contains a detailed explanation of the validation process using the Domain Name System (DNS).

Figure 20: ACM configuration

Project: Deploying WordPress Server and EC2 Instance

Step 1: Configure health check

Step 2: Get notified when health check fails

Configure health check

Route 53 health checks let you track the health status of your resources, such as web servers or mail servers, and take action when an outage occurs.

Name: aryanvijcyberblog

What to monitor: Endpoint

Monitor an endpoint

Specify endpoint by: IP address

Protocol: HTTP

IP address *: 18.116.102.120

Host name: aryanvijcyberblog.com

Port *: 80

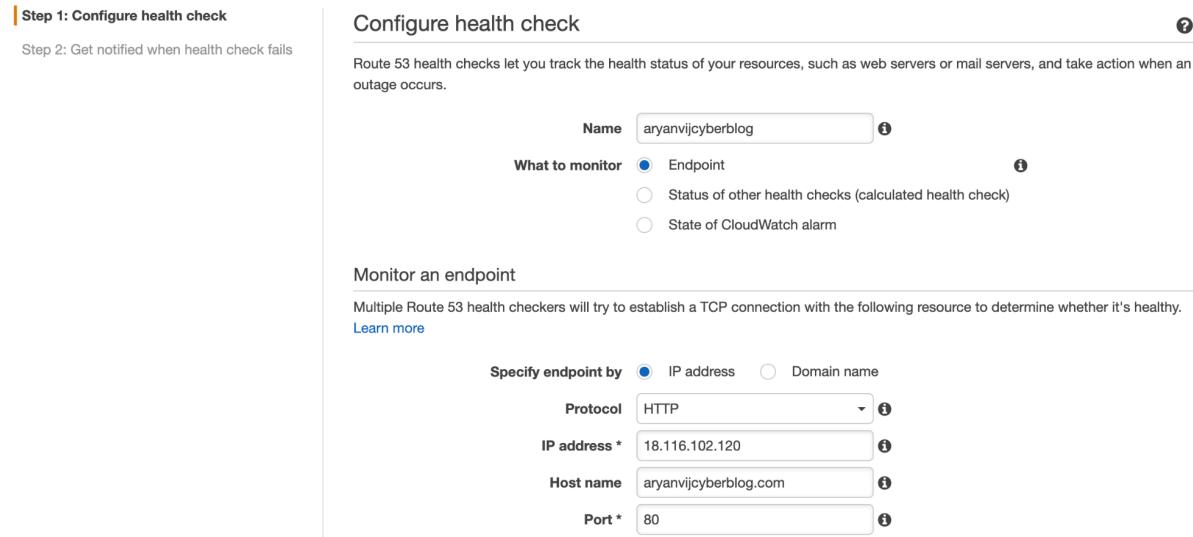


Figure 21: Health check configuration

Create health check

Step 1: Configure health check

Step 2: Get notified when health check fails

Get notified when health check fails

If you want CloudWatch to send you an Amazon SNS notification, such as an email, when the status of the health check changes to unhealthy, create an alarm and specify where to send notifications.

Create alarm: Yes

CloudWatch sends you an Amazon SNS notification whenever the status of this health check is unhealthy for at least one minute. The alarm will be located in the us-east-1 region.

Send notification to: New SNS topic

Topic name *: Alert

Recipient email addresses *: aryan.viji@gmail.com

* Required

Cancel Previous Create health check

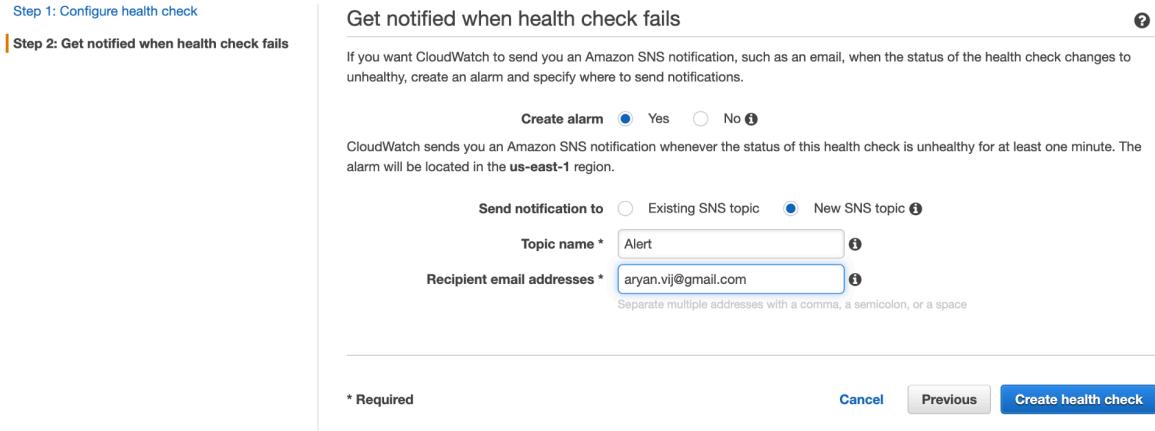


Figure 22: Health check finalisation

Project: Deploying WordPress Server and EC2 Instance

The screenshot shows the 'Create Application Load Balancer' configuration page. At the top, there is a breadcrumb navigation: 'EC2 > Load balancers > Create Application Load Balancer'. The main title is 'Create Application Load Balancer' with an 'Info' link. Below the title, a description explains that the Application Load Balancer distributes incoming HTTP and HTTPS traffic across multiple targets based on request attributes. A section titled 'How Application Load Balancers work' is shown. The 'Basic configuration' section starts with 'Load balancer name', which is set to 'Blog Site Load Balancer'. It includes a note that names must be unique and can't be changed after creation. The 'Scheme' section shows 'Internet-facing' selected, with a note that it requires a public subnet. The 'IP address type' section shows 'IPv4' selected, with a note that it includes only IPv4 addresses. The entire configuration form is contained within a light gray box.

Figure 23: Configuration Application Load Balancer

Project: Deploying WordPress Server and EC2 Instance

The screenshot shows the AWS CloudFrontListeners and routing configuration interface. It displays two listeners: Listener HTTP:80 and Listener HTTPS:443. Each listener has a 'Protocol' dropdown, a 'Port' input field, a 'Default action' section, and a 'Create target group' link. The 'Listener tags - optional' section includes an 'Add listener tag' button and a note about adding up to 50 more tags. A large 'Add listener' button is located at the bottom left.

Listeners and routing [Info](#)
A listener is a process that checks for connection requests using the port and protocol you configure. The rules that you define for a listener determine how the load balancer routes requests to its registered targets.

▼ Listener HTTP:80 [Remove](#)

Protocol	Port	Default action	Info
HTTP	: 80 1-65535	Forward to blogsitetarget Target type: Instance, IPv4	HTTP ▼ C

[Create target group](#)

Listener tags - optional
Consider adding tags to your listener. Tags enable you to categorize your AWS resources so you can more easily manage them.

[Add listener tag](#)

You can add up to 50 more tags.

▼ Listener HTTPS:443 [Remove](#)

Protocol	Port	Default action	Info
HTTPS	: 443 1-65535	Forward to blogsitesecure Target type: Instance, IPv4	HTTPS ▼ C

[Create target group](#)

Listener tags - optional
Consider adding tags to your listener. Tags enable you to categorize your AWS resources so you can more easily manage them.

[Add listener tag](#)

You can add up to 50 more tags.

[Add listener](#)

Figure 24: Listeners and routing

Project: Deploying WordPress Server and EC2 Instance

The screenshot shows the AWS VPC configuration interface. At the top, there's a navigation bar with the AWS logo, 'Services' (selected), a search bar, and '[Option+S]'. Below the navigation bar, the 'EC2' service is selected.

VPC | [Info](#)
Select the virtual private cloud (VPC) for your targets or you can create a new VPC [\[?\]](#). Only VPCs with an internet gateway are enabled for selection. The selected VPC can't be changed after the load balancer is created. To confirm the VPC for your targets, view your [target groups](#) [\[?\]](#).

VPC Personal Project
vpc-007ca4fccbf9a3a0
IPv4 VPC CIDR: 10.0.0.0/16

Mappings | [Info](#)
Select at least two Availability Zones and one subnet per zone. The load balancer routes traffic to targets in these Availability Zones only. Availability Zones that are not supported by the load balancer or the VPC are not available for selection.

us-east-2a (use2-az1)
Subnet
subnet-0f2b0e177fb8b2fba [\[?\]](#) [Edit](#)

IPv4 address
Assigned by AWS

us-east-2b (use2-az2)
Subnet
subnet-0fbeca3d7f4e1b9d7 [\[?\]](#) [Edit](#)

IPv4 address
Assigned by AWS

us-east-2c (use2-az3)

Security groups | [Info](#)
A security group is a set of firewall rules that control the traffic to your load balancer. Select an existing security group, or you can [create a new security group](#) [\[?\]](#).

Security groups
Select up to 5 security groups

default [X](#) [\[?\]](#) [Edit](#)
sg-0e63856b197556a65 VPC: vpc-007ca4fccbf9a3a0

Listeners and routing | [Info](#)
A listener is a process that checks for connection requests using the port and protocol you configure. The rules that you define for a listener determine how the load balancer routes requests to its registered targets.

▶ Listener HTTP:80 [Remove](#)

▶ Listener HTTPS:443 [Remove](#)

[Add listener](#)

Figure 25: VPC Configurations