

Capstone Project Submission

Title: Effect of Orchestration and Microsegmentation on Web Application Security **Student Name:** Oluwaseun Osunsola **Student ID:** IDEAS/24/63183 **Program:** Professional Diploma in Cybersecurity – Baze University

Abstract

This project investigates how orchestration and microsegmentation influence web application security. By deploying a WordPress web application on AWS using CloudFormation templates, the project demonstrates automated provisioning, autoscaling, and the enforcement of microsegmentation principles through security groups. The experiment verifies that these strategies improve web application availability, resilience against misconfigurations, and isolation of critical resources, thereby reducing the attack surface.

1. Introduction

Web applications are often targeted due to their exposure to the internet. Security risks arise from misconfigured servers, unpatched software, and lack of network isolation. Modern deployment strategies such as **orchestration** (automated provisioning and scaling) and **microsegmentation** (fine-grained network access control) are increasingly adopted to improve security posture.

This project focuses on implementing a fully orchestrated WordPress web application with microsegmented network components on AWS and testing its security, availability, and resilience under load.

2. Objectives

1. Deploy a fully functional WordPress web application using AWS CloudFormation orchestration.
 2. Implement microsegmentation via AWS Security Groups to isolate the database and web servers.
 3. Test autoscaling behavior under high CPU load to assess resilience.
 4. Verify that security and connectivity remain intact after termination or recreation of resources.
-

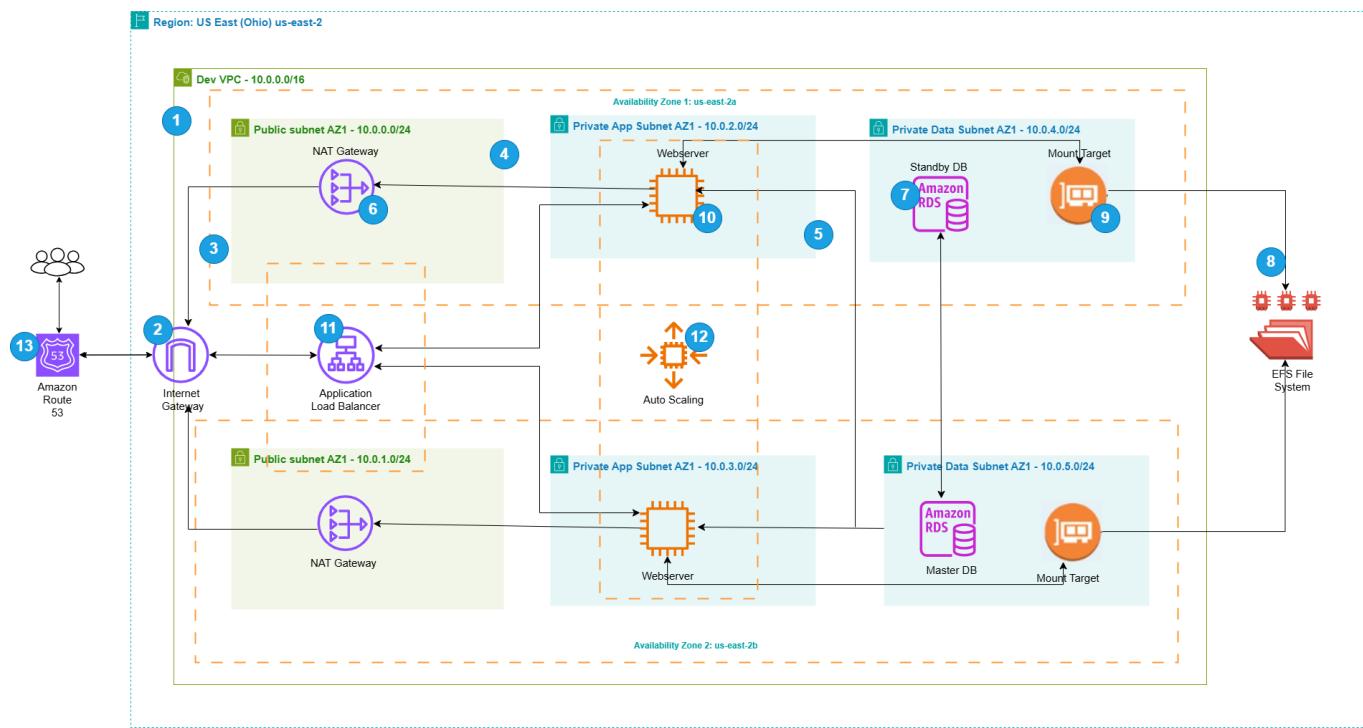
3. Methodology

3.1 Architecture Design

The architecture consists of:

- **Frontend:** EC2 instances hosting WordPress.
- **Database:** RDS instance.
- **Load Balancer:** AWS Application Load Balancer (ALB).
- **Orchestration:** AWS CloudFormation for provisioning and management.
- **Microsegmentation:** Security groups controlling inbound/outbound traffic.

Screenshot Reference:

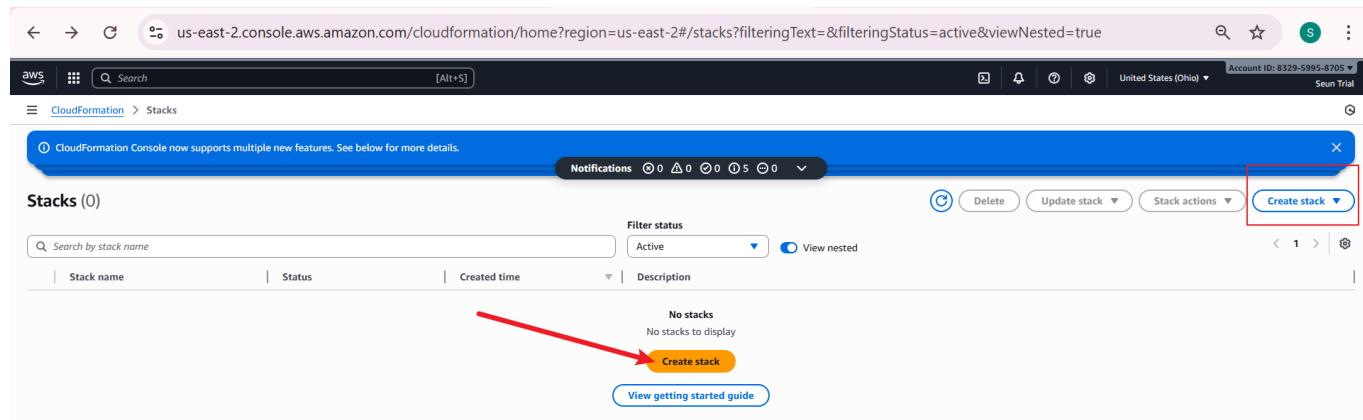


3.2 Deployment Steps

Step 1: Navigate to CloudFormation Dashboard

- Access AWS CloudFormation console.
- Click *Create Stack*.

Screenshot Reference:



Step 2: Upload Template

- Choose existing template (YAML/JSON) for WordPress deployment.
- Click *Next*.

Screenshot Reference:

The screenshot shows the 'Create stack' wizard in the AWS CloudFormation console. The current step is 'Step 2: Specify stack details'. The 'Prerequisite - Prepare template' section is visible, showing options to choose an existing template or build from Infrastructure Composer. A red box highlights the 'Choose an existing template' option. Below it, the 'Template source' section shows 'Amazon S3 URL' selected, with a red box around the 'Upload a template file' button. A file named 'scalable-wordpress-app.yaml' is listed in the dropdown. At the bottom right, a red box surrounds the 'Next' button.

Step 3: Configure Stack

- Enter stack name.
- Enter database credentials.
- Scroll to next steps.

Screenshot Reference:

The screenshot shows the 'Specify stack details' step of the CloudFormation wizard. The 'Provide a stack name' section has a red box around the 'Stack name' input field, which contains 'scalable-wordpress-app'. The 'Parameters' section is expanded, showing fields for 'DBName' (wordpress), 'DBPassword' (redacted with '*****'), and 'DBUsername' (admin). A red box highlights the 'DBName' field. The 'ACMCertificateArn' section is also partially visible.

Step 4: Key Pair & IAM Acknowledgment

- Select key pair. Leave defaults.
- Acknowledge IAM resource creation warning.

Screenshot References:

DBPassword
WordPress database password

DBUsername
WordPress database username
admin

DesiredCapacity
Desired number of instances
2

KeyName
EC2 Key Pair for SSH access
MyKeyPair1

LatestAmiId
Use SSM parameter to resolve the latest Amazon Linux 2 AMI in the region
/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2

MaxCapacity
Maximum number of instances
4

MinCapacity
Minimum number of instances
2

Cancel **Previous** **Next**

ADDITIONAL SETTINGS
You can set additional options for your stack, like notification options and a stack policy. [Learn more](#)

▶ Stack policy - optional
Defines the resources that you want to protect from unintentional updates during a stack update.

▶ Rollback configuration - optional
Specify alarms for CloudFormation to monitor when creating and updating the stack. If the operation breaches an alarm threshold, CloudFormation rolls it back.

▶ Notification options - optional
Specify a new or existing Amazon Simple Notification Service topic where notifications about stack events are sent.

▶ Stack creation options - optional
Specify the timeout and termination protection options for stack creation.

Capabilities

The following resource(s) require capabilities: [AWS::IAM::Role]
This template contains Identity and Access Management (IAM) resources that might provide entities access to make changes to your AWS account. Check that you want to create each of these resources and that they have the minimum required permissions. [Learn more](#)

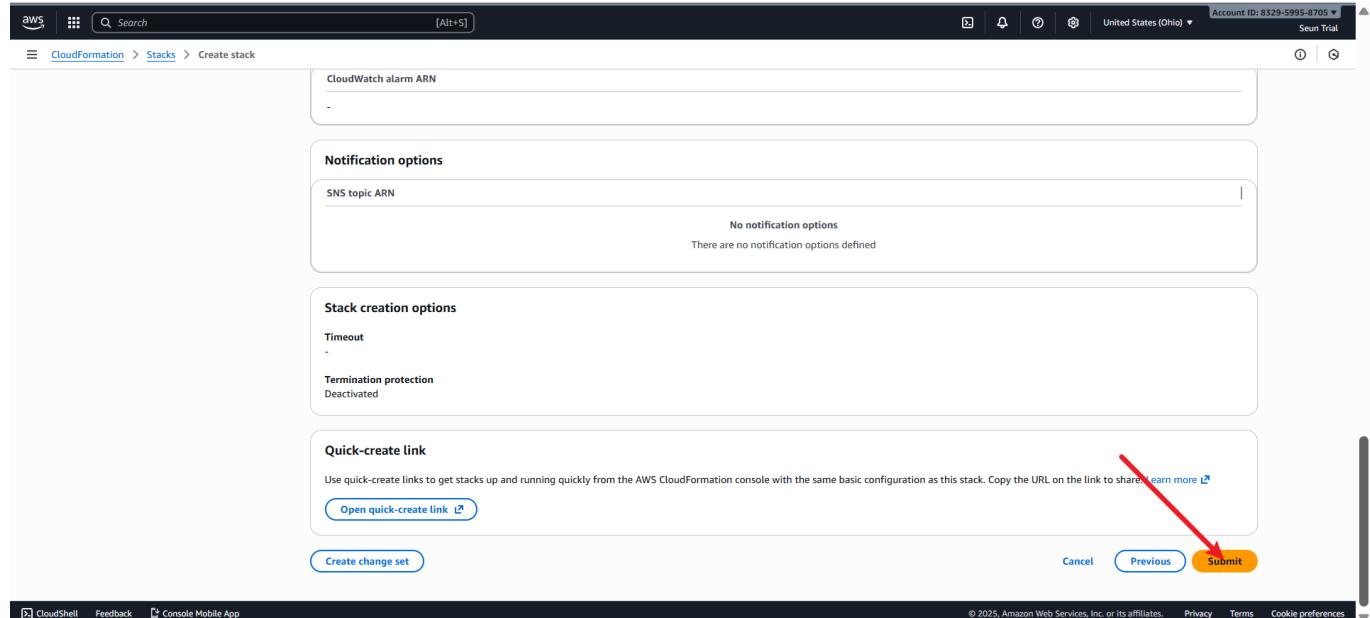
I acknowledge that AWS CloudFormation might create IAM resources.

Cancel **Previous** **Next**

Step 5: Submit Stack

- Leave other options at default.
- Submit.

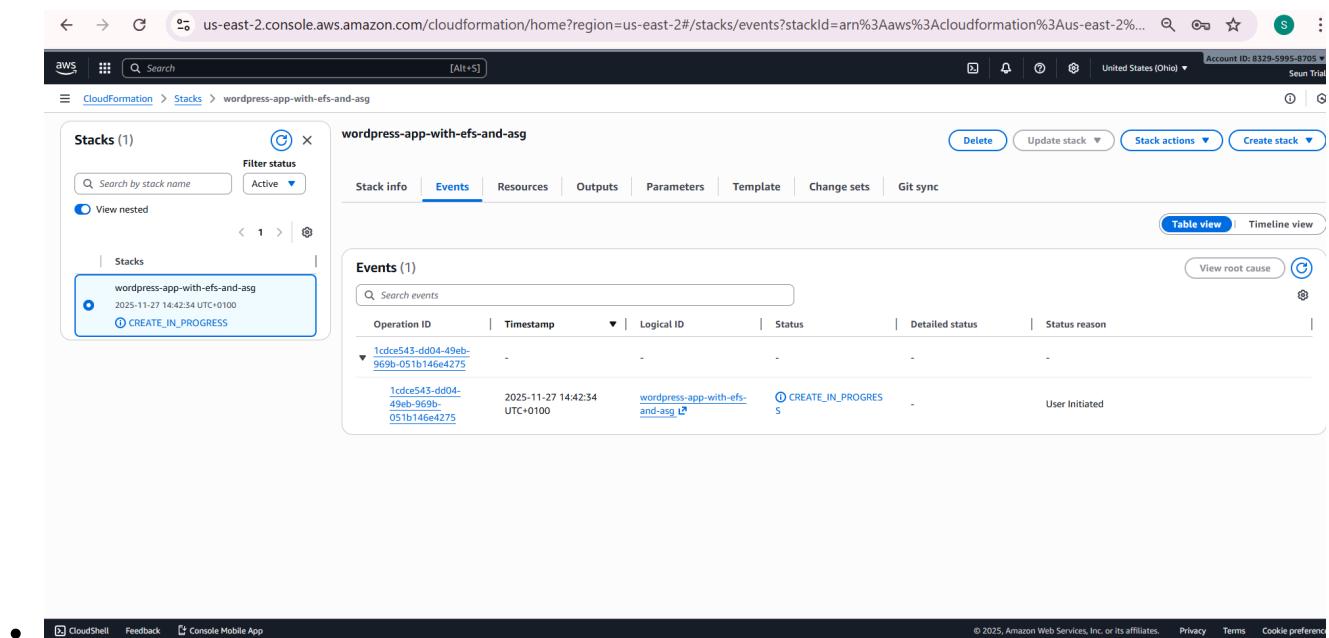
Screenshot Reference:



Step 6: Monitor Stack Creation

- Wait until stack creation completes.

Screenshot References:



The screenshot shows the AWS CloudFormation console with the 'scalable-wordpress-app' stack selected. The 'Outputs' tab is active, displaying a table of outputs. The 'WebsiteURL' output, which contains the value <http://wordpress-alb-295077538.us-east-2.elb.amazonaws.com>, is highlighted with a red box.

Key	Value	Description
DatabaseEndpoint	scalable-wordpress-app-wordpressdb-fcfd2u0vhcm.c1gkoy0jick.us-east-2.rds.amazonaws.com	RDS MySQL Endpoint
EFSFileSystemId	fs-0f1c23f61388aaa57	EFS File System ID
LoadBalancerDNS	wordpress-alb-295077538.us-east-2.elb.amazonaws.com	Application Load Balancer DNS Name
VPCId	vpc-0ea90fdf581dbb6ee0	VPC ID
WebsiteURL	http://wordpress-alb-295077538.us-east-2.elb.amazonaws.com	WordPress Website URL

3.3 Connect to EC2 Instance

- Navigate to running EC2 instances.
- Connect using **Session Manager**.

Screenshot References:

The screenshot shows the AWS EC2 Instances page. The left sidebar shows navigation options like Dashboard, EC2 Global View, Instances, Images, Elastic Block Store, Network & Security, and more. The main area displays a table of instances. The first instance, with the ID [i-0e47468a6710c5311](#), is highlighted with a red box. The table columns include Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, Public IPv4 DNS, Public IPv4, and Elastic IP.

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4	Elastic IP
wordpress-we...	i-0e47468a6710c5311	Running	t3.micro	5/3 checks passed	View alarms +	us-east-2b	-	-	-
wordpress-we...	i-0ccdf5b54fa6d6b2ef	Running	t3.micro	Initializing	View alarms +	us-east-2a	-	-	-
wordpress-we...	i-0efaa4632ac7bd055	Running	t3.micro	5/3 checks passed	View alarms +	us-east-2a	-	-	-

Instance summary for i-0c47468a6710c5311 (wordpress-web-server) [Info](#)

Updated less than a minute ago

Instance ID	i-0c47468a6710c5311	Public IPv4 address	–
IPv6 address	–	Instance state	Running
Hostname type	IP name: ip-10-0-4-63.us-east-2.compute.internal	Private IP DNS name (IPv4 only)	ip-10-0-4-63.us-east-2.compute.internal
Answer private resource DNS name	–	Instance type	t3.micro
Auto-assigned IP address	–	VPC ID	vpc-dea90fd581dbb6e0 (wordpress-vpc) View
IAM Role	scalable-wordpress-app-WordPressEC2Role-tDrrtfvst95 View	Subnet ID	subnet-0fc5310e042c8861c (wordpress-private-app-az2) View
IMDSv2	Optional	Instance ARN	arn:aws:ec2:us-east-2:832959958705:instance/i-0c47468a6710c5311
Operator	–	Platform details	Linux/UNIX

Details **Status and alarms** **Monitoring** **Security** **Networking** **Storage** **Tags**

Instance details [Info](#)

AMI ID: ami-0979de60fed46582f Monitoring: disabled

Session Manager usage:

- Connect to your instance without SSH keys, a bastion host, or opening any inbound ports.
- Sessions are secured using an AWS Key Management Service key.
- You can log session commands and details in an Amazon S3 bucket or CloudWatch Logs log group.
- Configure sessions on the Session Manager [Preferences](#) page.

[Try for free](#) [X](#)

[Cancel](#) [Connect](#)

3.4 Verify WordPress Deployment

- Navigate to WordPress folder.
- List contents to verify `wp-config.php`.
- Check database connection and HTTPS settings.

Screenshot References:

The image contains three vertically stacked screenshots of the AWS Systems Manager Session Manager terminal interface. Each screenshot shows a terminal window with a session ID, instance ID, and various AWS navigation icons.

Screenshot 1: Shows the command `ls -la wp-config.php` being run, displaying the contents of the file which includes database connection settings.

```
Session ID: root-qyp7ksx8qfjlrq89c0kz3q58 Instance ID: i-0958ccdf67b78a4e6
sh-4.2$ cd /var/www/html
sh-4.2$ ls -la wp-config.php
-rw-r--r-- 1 apache apache 3577 Dec 12 14:23 wp-config.php
sh-4.2$
```

Screenshot 2: Shows the command `grep -A6 -B2 "DB_NAME|DB_USER|DB_PASSWORD|DB_HOST|DB_CHARSET|DB_COLLATE" wp-config.php` being run, highlighting the database connection settings in the output.

```
Session ID: root-qyp7ksx8qfjlrq89c0kz3q58 Instance ID: i-0958ccdf67b78a4e6
sh-4.2$ cd /var/www/html
sh-4.2$ ls -la wp-config.php
-rw-r--r-- 1 apache apache 3577 Dec 12 14:23 wp-config.php
sh-4.2$ # ... Only the exact lines you need for the documentation
sh-4.2$ echo "==== DATABASE CONNECTION SETTINGS ===="
==== DATABASE CONNECTION SETTINGS ====
sh-4.2$ grep -A6 -B2 "DB_NAME|DB_USER|DB_PASSWORD|DB_HOST|DB_CHARSET|DB_COLLATE" wp-config.php
// ...
/* Database settings - You can get this info from your web host */
/* The name of the database for WordPress */
define('DB_NAME', 'wordpress');

/* Database username */
define('DB_USER', 'admin');

/* Database password */
define('DB_PASSWORD', 'Password1');

/* Database hostname */
define('DB_HOST', 'scalable-wordpress-app-wordpressdb-5nrkv7efzboz.cob6k6osq7iv.us-east-1.rds.amazonaws.com');

/* Database charset to use in creating database tables. */
define('DB_CHARSET', 'utf8mb4');

/* The database collate type. Don't change this if in doubt. */
define('DB_COLLATE', '');

/*#*/
/* Authentication unique keys and salts.
 * Change these to different unique phrases! You can generate these using
 * the (#link https://api.wordpress.org/secret-key/1.1/salt/) WordPress.org secret-key service).
sh-4.2$ echo ""
sh-4.2$ echo "==== SSL / HTTPS SETTINGS ===="
```

Screenshot 3: Shows the command `df -h | grep efs` being run, displaying disk usage information for the EFS volume.

```
Session ID: root-euan9ompbe0lkv08x4ku Instance ID: i-06872a7d77908dcba
sh-4.2$ df -h | grep efs
Filesystem      Size  Used  Avail Capacity  Mounted on
efs-045acd1433eacd7e.efs.us-east-1.amazonaws.com:  8.0E    0  8.0E  0% /var/www/html
sh-4.2$
```

3.5 Test Orchestration (Autoscaling)

Step 1: Check Load Balancer DNS output.

Screenshot References:

CloudFormation Console now supports multiple new features. See below for more details.

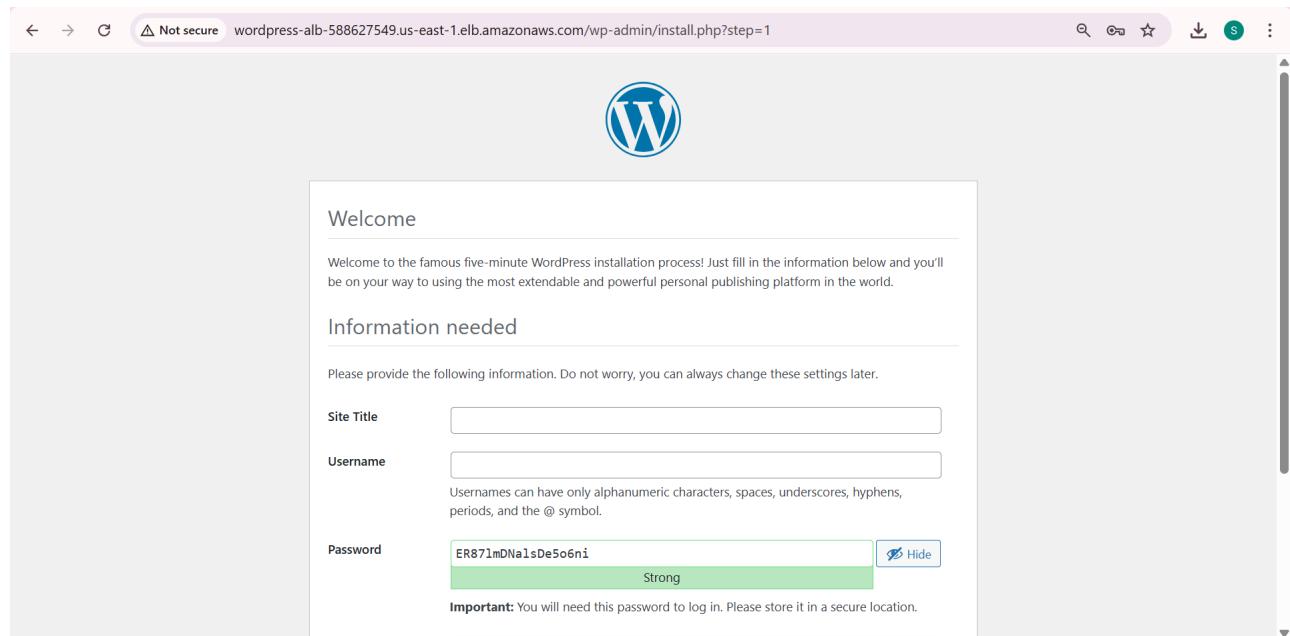
scalable-wordpress-app

Outputs (5)

Key	Value	Description	Export name
DatabaseEndpoint	scalable-wordpress-app-wordpressdb-5nrx7efbox.cob6k6oq7iv.us-east-1.rds.amazonaws.com	RDS MySQL Endpoint	-
EFSFileSystemId	fs-045dc1433eacd7e	EFS File System ID	-
LoadBalancerDNS	wordpress-alb-588627549.us-east-1.elb.amazonaws.com	Application Load Balancer DNS Name	-
VPCId	vpc-0bb1f2cd5e0ca478d	VPC ID	-
WebsiteURL	http://wordpress-alb-588627549.us-east-1.elb.amazonaws.com	WordPress Website URL	-

Step 2: Confirm configuration files.

Screenshot Reference:



Step 3: Terminate an EC2 instance to simulate failure.

- Verify website availability via Load Balancer.

Screenshot References:

Instances (1/4) Info

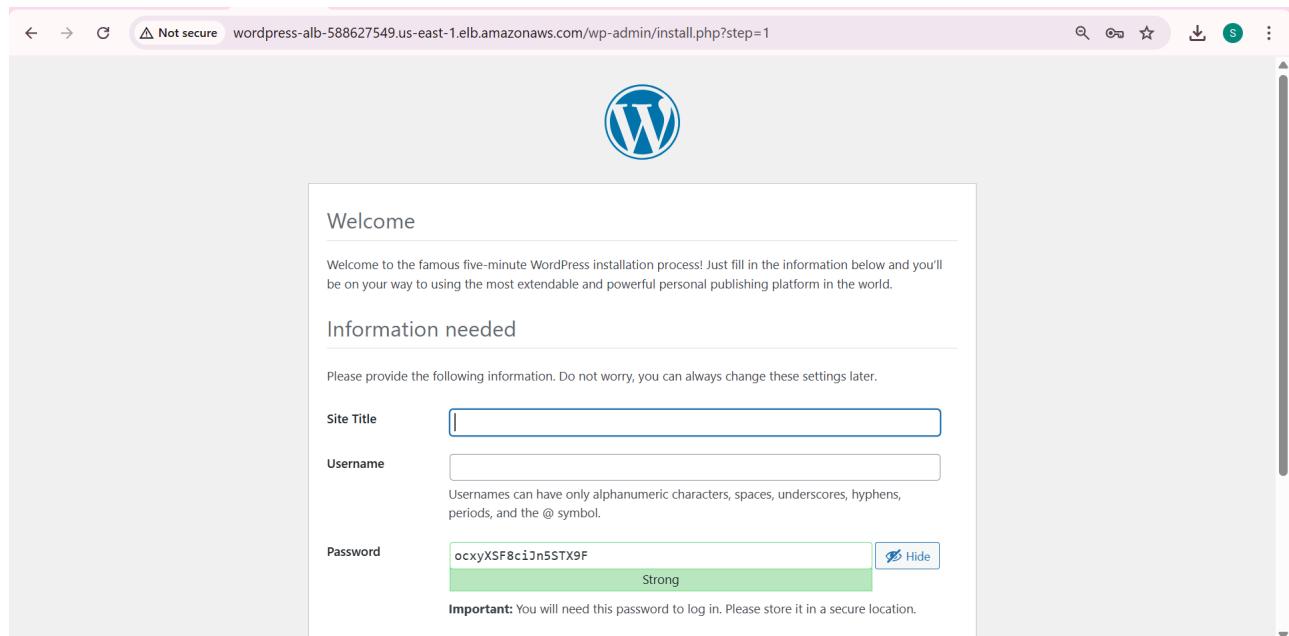
Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP
wordpress-we...	i-06872a7d77908dcba	Running	t3.micro	-	View alarms +	us-east-1a	-	-	-
wordpress-we...	i-063fd5d55d87ab1	Running	t3.micro	3/3 checks passed	View alarms +	us-east-1a	-	-	-
wordpress-we...	i-0958ccdf7b78a4e6	Terminated	t3.micro	-	View alarms +	us-east-1b	-	-	-
wordpress-we...	i-05efd3198e1c8fb6	Running	t3.micro	-	View alarms +	us-east-1b	-	-	-

i-06872a7d77908dcba (wordpress-web-server)

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

Instance summary Info

Instance ID	i-06872a7d77908dcba	Public IPv4 address	Private IPv4 addresses
IPv6 address	-	Instance state	Public DNS
Hostname type	-	Instance type	Elastic IP addresses
Answer private resource DNS name		t3.micro	



Step 4: Generate high CPU load to test autoscaling.

- Install stress package.
- Load CPU for 10 mins.

Screenshot References:

```
sh-4.2$ sudo amazon-linux-extras install epel -y
Popularity check has ended - support date of 2024-06-30
Installing epel-release
Loaded plugins: extras suggestions, langpacks, priorities, update-motd
Cleaning repos: amzn2-core amzn2extra-docker amzn2extra-epel amzn2extra-php7.4
17 metadata files removed
6 sqlite files removed
9 rpmdb files removed
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core
amzn2extra-docker
amzn2extra-epel
amzn2extra-php7.4
(1/8): amzn2-core/2/x86_64/group_gz
(2/8): amzn2-core/2/x86_64/updateinfo
(3/8): amzn2extra-php7.4/2/x86_64/updateinfo
(4/8): amzn2extra-docker/2/x86_64/primary_db
(5/8): amzn2extra-epel/2/x86_64/primary_db
(6/8): amzn2extra-php7.4/2/x86_64/primary_db
(7/8): amzn2extra-php7.4/2/x86_64/primary_db
(8/8): amzn2-core/2/x86_64/primary_db
Resolving Dependencies
--> Running transaction check
-->> Package epel-release.noarch 0:7-11 will be installed
-->> Finished Dependency Resolution
Dependencies Resolved

Transaction Summary
  Package           Arch      Version            Repository        Size
Install  1 Package

Total download size: 15 k
Installed size: 24 k
Downloading packages:
epel-release-7-11.noarch.rpm                                         | 15 kB  00:00:00
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : epel-release-7-11.noarch                                         1/1
  Verifying  : epel-release-7-11.noarch                                         1/1
```

```
Session ID: root-8ea8jf2v5hz7vq3vahp2bra [Shortcuts] Instance ID: i-059873f33e58e635f [Terminate]

State : Running, pid: 6645
Another app is currently holding the yum lock; waiting for it to exit...
The other application is yum
Memory usage: 1003 MB (1.03 GB)
Started: Fri Dec 14 14:57:23 2025 - 00:11 ago
State : Running, pid: 6645
244 packages excluded due to repository priority protections
Resolving Dependencies
--> Running transaction check
--> Package stress.x86_64 0:1.0.4-16.el7 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

Package           Arch      Version            Repository  Size
installing:      stress    x86_64   1.0.4-16.el7    epel       39 k
Transaction Summary
Install 1 Package

Total download size: 39 k
Installed size: 94 k
Proceeding to install...
warning: /var/tmp/yum-root/um/86_64/2/epel/packages/stress-1.0.4-16.el7.x86_64.rpm: Header V3 RSA/SHA256 Signature, key ID 352c64e5: NOKEY
Public key for stress-1.0.4-16.el7.x86_64.rpm is not installed
stress-1.0.4-16.el7.x86_64.rpm
Retrieving key from file:///etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-7
Importing GPG key 0x352c64e5
[verified] https://dl.fedoraproject.org/pub/epel/7/ | 39 kB 00:00:00
Fileingerprint: 91a9 7c74 45a9 6f5f 7f30 0a8f 6a2f ae02 352c 64e5
Package : epel-release-7-11.noarch (@amzn2extra-epel)
From     : /etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-7
Running transaction check
Running transaction test
transaction test succeeded
Running transaction
  Installing : stress-1.0.4-16.el7.x86_64
  Verifying  : stress-1.0.4-16.el7.x86_64
Installed:
  stress.x86_64 0:1.0.4-16.el7

Complete!
sh-4.2$
```



```
Session ID: root-8ea8jf2v5hz7vq3vahp2bra [Shortcuts] Instance ID: i-059873f33e58e635f [Terminate]

sh-4.2$ stress --cpu 8 --timeout 600
stress: info: [6709] dispatching hogs: 8 cpu, 0 io, 0 vm, 0 hdd
```

Step 5: Verify new EC2 instance spins up under load.

Screenshot Reference:

The screenshot shows the AWS EC2 Instances page. The left sidebar includes sections for Dashboard, EC2 Global View, Events, Instances (selected), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Capacity Manager, Images, AMIs, AMI Catalog, Elastic Block Store, Volumes, Snapshots, Lifecycle Manager, and Network & Security. The main content area displays a table of 9 instances, each with columns for Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, Public IPv4 DNS, and Public IPv4. One instance, named "wordpress-web-server" with Instance ID i-013361a6007222a5d, is highlighted with a red box. This instance is currently "Running" and has a status check of "3/3 checks passed".

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4
wordpress-web-server	i-002d4016d46cb235a	Terminated	t3.micro	-	View alarms +	us-east-1b	-	-
wordpress-web-server	i-068722a7d77908c6ba	Terminated	t3.micro	-	View alarms +	us-east-1a	-	-
wordpress-web-server	i-063f2d5a5f5d87ab1	Terminated	t3.micro	-	View alarms +	us-east-1a	-	-
wordpress-web-server	i-059873f3e58e635f	Running	t3.micro	3/3 checks passed	View alarms +	us-east-1b	-	-
wordpress-web-server	i-095ccdcfe7b78a4e6	Terminated	t3.micro	-	View alarms +	us-east-1b	-	-
wordpress-web-server	i-05efd3198e1c8fb6	Terminated	t3.micro	-	View alarms +	us-east-1b	-	-
wordpress-web-server	i-092038185ca965151	Running	t3.micro	Initializing	View alarms +	us-east-1b	-	-
wordpress-web-server	i-01728214b72506a3c	Terminated	t3.micro	-	View alarms +	us-east-1a	-	-
wordpress-web-server	i-013361a6007222a5d	Running	t3.micro	3/3 checks passed	View alarms +	us-east-1a	-	-

3.6 Verify Database Microsegmentation

- Copy RDS Database Endpoint.
 - Test connection using credentials.

Screenshot References:

The screenshot shows the AWS CloudFormation console with the stack named "scalable-wordpress-app". The "Outputs" tab is selected, displaying a table with five outputs. One output, "DatabaseEndpoint", is highlighted with a red border. Its value is "scalable-wordpress-app-wordpressdb-5nrkv7efzboz.cob6k6sq7iv.us-east-1.rds.amazonaws.com".

The screenshot shows a terminal session in the AWS CloudShell. The user is connected to a MySQL database on the RDS instance. The command entered was "mysql -h scalable-wordpress-app-wordpressdb-5nrkv7efzboz.cob6k6sq7iv.us-east-1.rds.amazonaws.com -u admin \t". The password was entered, and the user is now connected to the MySQL prompt.

3.7 Verify Security Groups (Microsegmentation)

- Check security groups for EC2 and RDS instances.
- Confirm that only necessary ports are open (HTTP/HTTPS for web servers, MySQL for database).

Screenshot Reference:

The screenshot shows the AWS EC2 console under the "Security Groups" section. A table lists six security groups. The first group, "scalable-wordpress-app-WebServerSec...", has its "sg-015bd4de1de2e6fb26" ID highlighted with a red border. The table includes columns for Name, Security group ID, Security group name, VPC ID, Description, and Owner.

4. Results

1. CloudFormation successfully orchestrated the WordPress deployment.
2. Autoscaling worked as expected: terminating an instance did not disrupt service.
3. High CPU load triggered the creation of additional EC2 instances, maintaining availability.
4. Microsegmentation ensured database and web servers were isolated, reducing exposure.

Conclusion: Orchestration combined with microsegmentation significantly improved both security and resilience of the web application.

5. Discussion

- **Orchestration Benefits:** Automated provisioning reduces human errors, ensures consistent configurations, and supports rapid scaling.
 - **Microsegmentation Benefits:** Limits lateral movement within the network; if one instance is compromised, others remain protected.
 - **Security Posture:** The combination provides strong protection against common threats, misconfigurations, and downtime.
-

6. References

1. AWS Documentation: Amazon Web Services. (n.d.). AWS CloudFormation User Guide. Retrieved from <https://docs.aws.amazon.com/cloudformation/>. (Note: This also covers EC2, RDS, and Security Groups within the AWS documentation suite used in your methodology). +2
2. OWASP Top 10: OWASP Foundation. (2021). OWASP Top 10:2021 - The Ten Most Critical Web Application Security Risks. <https://owasp.org/www-project-top-ten/>.
3. NIST SP 800-125: Scarfone, K., Souppaya, M., & Hoffman, P. (2011). Guide to Security for Full Virtualization Technologies (NIST Special Publication 800-125). National Institute of Standards and Technology. <https://doi.org/10.6028/NIST.SP.800-125>.