SWINBURNE VIETNAM

HO CHI MINH CAMPUS

Class: COS40006

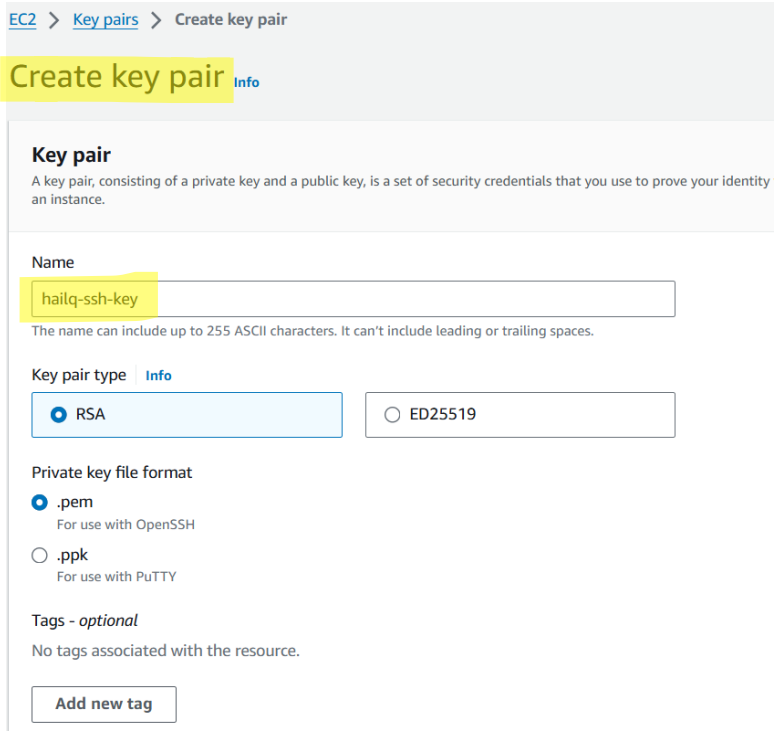
Deployment Portfolio Task 2

Instructor: Dr. Thomas Hang

Student names: Le Quang Hai

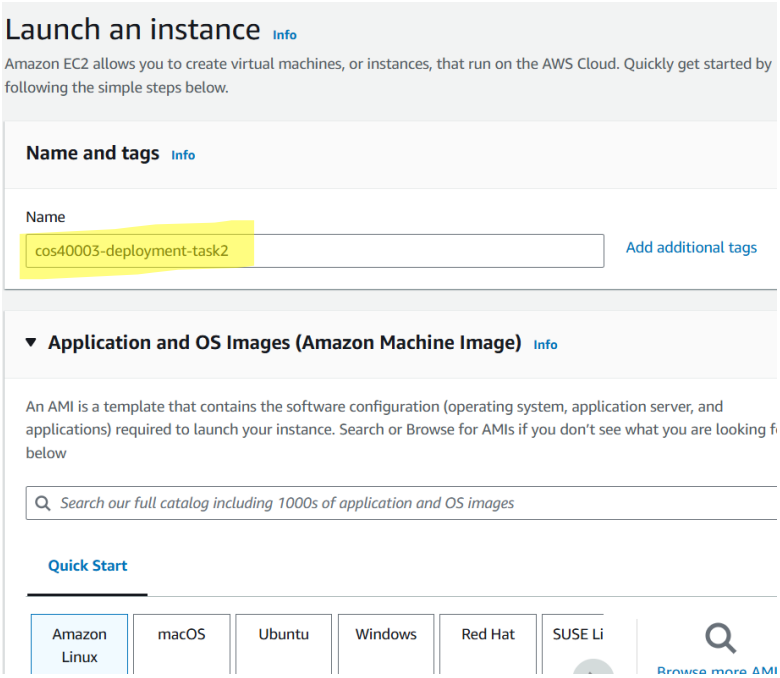
Task 2.1

Create an SSH key pair

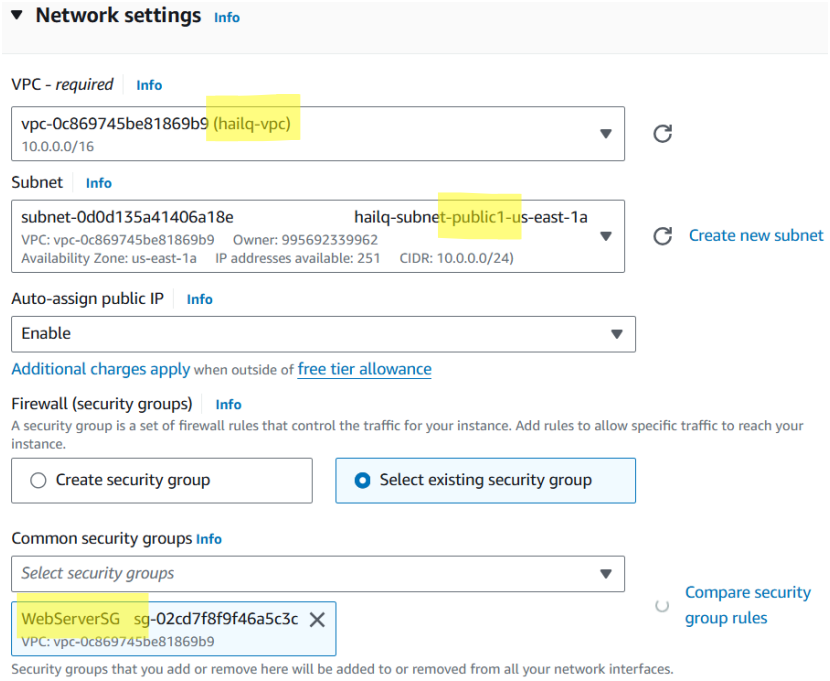


Create an EC2 instance

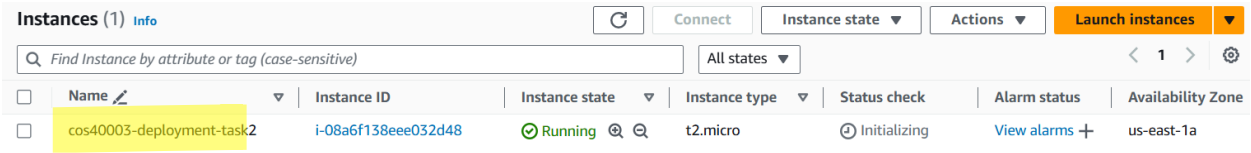
Initiate instance creation



Select VPC, located subnet, configure security group



For other settings, use default setting, then launch instance



Deploy a WordPress instance

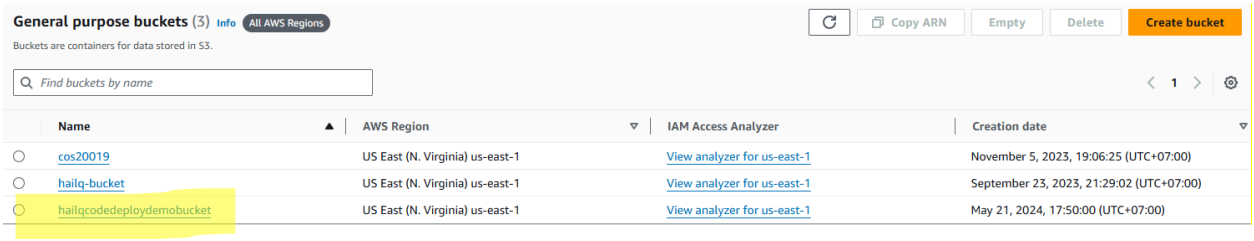
Load the source code to the EC2 instance

```
[ec2-user@ip-10-0-0-171 WordPress]$ ls
index.php    wp-activate.php     wp-comments-post.php  wp-cron.php        wp-load.php    wp-settings.php    xmlrpc.php
license.txt  wp-admin            wp-config-sample.php  wp-includes        wp-login.php   wp-signup.php
readme.html  wp-blog-header.php  wp-content            wp-links-opml.php  wp-mail.php    wp-trackback.php
```

Create scripts to run the application

```
[ec2-user@ip-10-0-0-171 scripts]$ cat start_server.sh
#!/bin/bash
systemctl start mariadb.service
systemctl start httpd.service
systemctl start php-fpm.service
[ec2-user@ip-10-0-0-171 scripts]$ cat stop_server.sh
#!/bin/bash
isExistApp=pgrep httpd
if [[ -n $isExistApp ]]; then
systemctl stop httpd.service
fi
isExistApp=pgrep mysqld
if [[ -n $isExistApp ]]; then
systemctl stop mariadb.service
fi
isExistApp=pgrep php-fpm
if [[ -n $isExistApp ]]; then
systemctl stop php-fpm.service

fi
[ec2-user@ip-10-0-0-171 scripts]$ cat create_test_db.sh
#!/bin/bash
mysql -uroot <<CREATE_TEST_DB
CREATE DATABASE IF NOT EXISTS test;
CREATE_TEST_DB
[ec2-user@ip-10-0-0-171 scripts]$ cat install_dependencies.sh
#!/bin/bash
sudo amazon-linux-extras install php7.4
sudo yum install -y httpd mariadb-server php
```

Create an S3 bucket for storing source code



Set public policy for S3 bucket

Block *all* public access

On

▶ **Individual Block Public Access settings for this bucket**

Bucket policy

The bucket policy, written in JSON, provides access to the objects stored in the bucket. Bucket policies don't apply to objects owned by other accounts. Learn more

Public access is blocked because Block Public Access settings are turned on for this bucket
To determine which settings are turned on, check your Block Public Access settings for this bucket. Learn more about using Amazon S3 Block Public Access


```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::995692339962:root"
      },
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::hailqcodedeploydemobucket/*"
    }
  ]
}
```


Bundle the application's files into a single archive file and push the archive file

```
[ec2-user@ip-10-0-0-171 WordPress]$ aws deploy create-application --application-name WordPress_App
{
    "applicationId": "2cf33b46-4223-47d1-89b4-9e90d62046f8"
}
[ec2-user@ip-10-0-0-171 WordPress]$ aws deploy push \
  --application-name WordPress_App \
  --s3-location s3://hailqcodedeploydemobucket/WordPressApp.zip \
  --ignore-hidden-files
To deploy with this revision, run:
aws deploy create-deployment --application-name WordPress_App --s3-location bucket=hailqcodedeploydemobucket,key=WordPre
ssApp.zip,bundleType=zip,eTag=76a80c1db7258b6acba20445f5f3b436-8 --deployment-group-name <deployment-group-name> --deplo
yment-config-name <deployment-config-name> --description <description>
```

hailqcodedeploydemobucket Info

Objects | Properties | Permissions | Metrics | Management | Access Points

Objects (1) Info

Copy S3 URI | Copy URL | Download | Open | Delete | Actions | Create folder

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Find objects by prefix

Name	Type	Last modified	Size	Storage class
WordPressApp.zip	zip	May 21, 2024, 18:51:09 (UTC+07:00)	44.2 MB	Standard

Create a deployment group

WordPress_App

Application details

Name	Compute platform
WordPress_App	EC2/On-premises

Deployments | Deployment groups | Revisions

Deployment groups

View details

Name	Status	Last attempted deployment	Last successful deployment
WordPress_DepGroup	In progress	-	-

Successfully deploy the app

Application	Deployment ID	Status
WordPress_App	d-XGL34QX96	Succeeded

Deployment configuration	Deployment group	Initiated by
CodeDeployDefault.OneAtATime	WordPress_DepGroup	User action

Deployment description

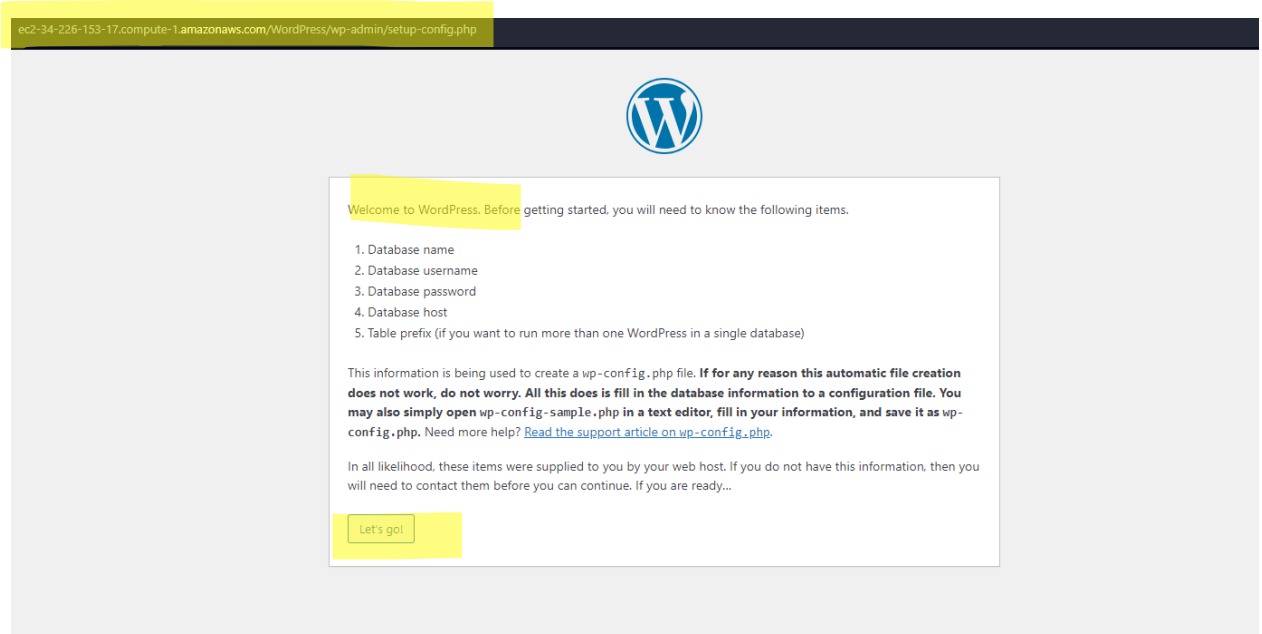
-

Revision details

Revision location	Revision created	Revision descriptic
s3://cos40006-code-deploy-bucket/WordPressApp.zip?eTag=80dba16259d3d3b251eccdb82c32b066-8	42 minutes ago	Uploaded by AWS

Event	Duration	Status	Error code	Start time
ApplicationStop	less than one second	Succeeded	-	May 23, 2024 11:13 AM (UTC+7:00)
DownloadBundle	3 seconds	Succeeded	-	May 23, 2024 11:13 AM (UTC+7:00)
BeforeInstall	24 seconds	Succeeded	-	May 23, 2024 11:14 AM (UTC+7:00)
Install	less than one second	Succeeded	-	May 23, 2024 11:14 AM (UTC+7:00)
AfterInstall	less than one second	Succeeded	-	May 23, 2024 11:14 AM (UTC+7:00)
ApplicationStart	2 seconds	Succeeded	-	May 23, 2024 11:14 AM (UTC+7:00)
ValidateService	less than one second	Succeeded	-	May 23, 2024 11:14 AM (UTC+7:00)

Access the application via http



Notes: the given script on Amazon website is compatible with Amazon Linux 2 EC2 instance; since we use the newest version Amazon Linux 2023, some script must be re-written.

- For installing PHP use: sudo dnf install php php-cli php-common php-mysqlnd php-json php-xml -y
- For installing MariaDB use: sudo dnf install mariadb105-server -y

Update the WordPress application

Set up the site

ec2-34-226-153-17.compute-1.amazonaws.com/WordPress/wp-admin/setup-config.php

Below you should enter your database connection details. If you are not sure about these, contact your host.

Database Name test

The name of the database you want to use with WordPress.

Username root

Your database username.

Password password Show

Your database password.

Database Host localhost

You should be able to get this info from your web host, if `localhost` does not work.

Table Prefix wp_

If you want to run multiple WordPress installations in a single database, change this.

Submit

ec2-34-226-153-17.compute-1.amazonaws.com/WordPress/wp-admin/install.php

Welcome

Welcome to the famous five-minute WordPress installation process! Just fill in the information below and you'll be on your way to using the most extendable and powerful personal publishing platform in the world.

Information needed

Please provide the following information. Do not worry, you can always change these settings later.

Site Title my site title

Username username

Usernames can have only alphanumeric characters, spaces, underscores, hyphens, periods, and the @ symbol.

Password password Hide

Very weak

Important: You will need this password to log in. Please store it in a secure location.

Confirm Password ☑ Confirm use of weak password

Your Email lequanghai1301@gmail.com

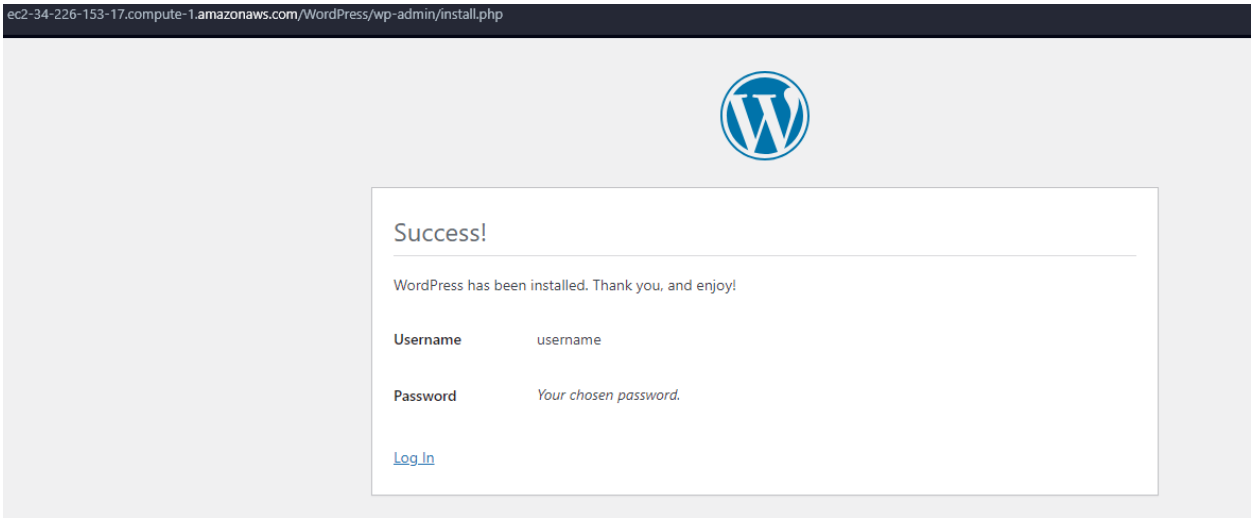
Double-check your email address before continuing.

Search engine visibility ☑ Discourage search engines from indexing this site

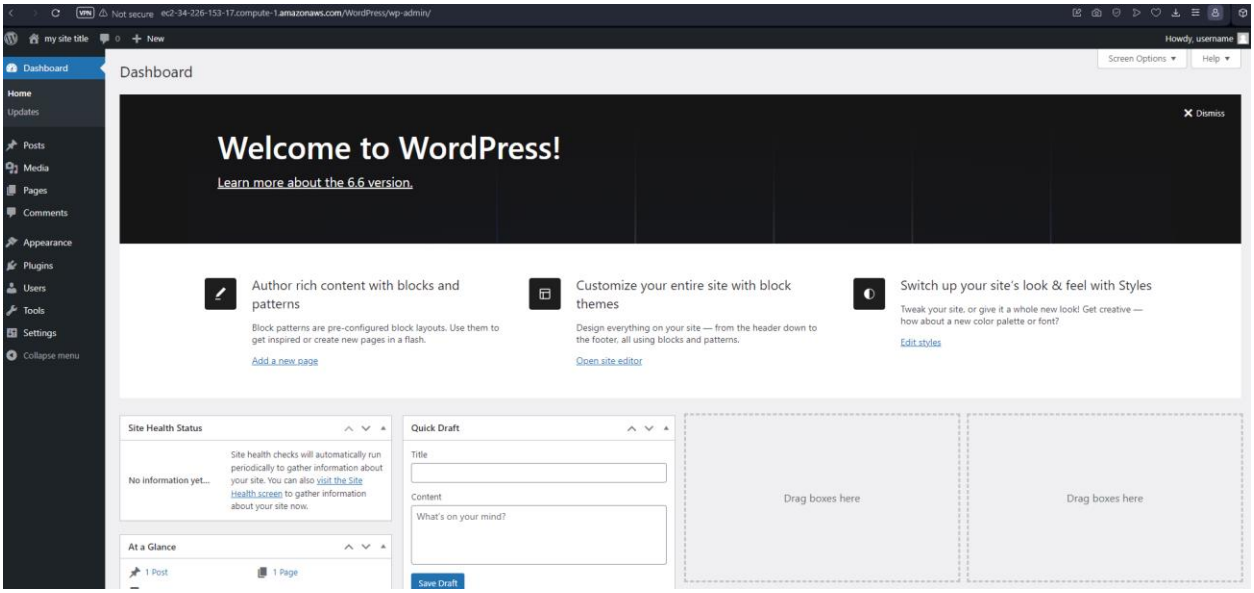
It is up to search engines to honor this request.

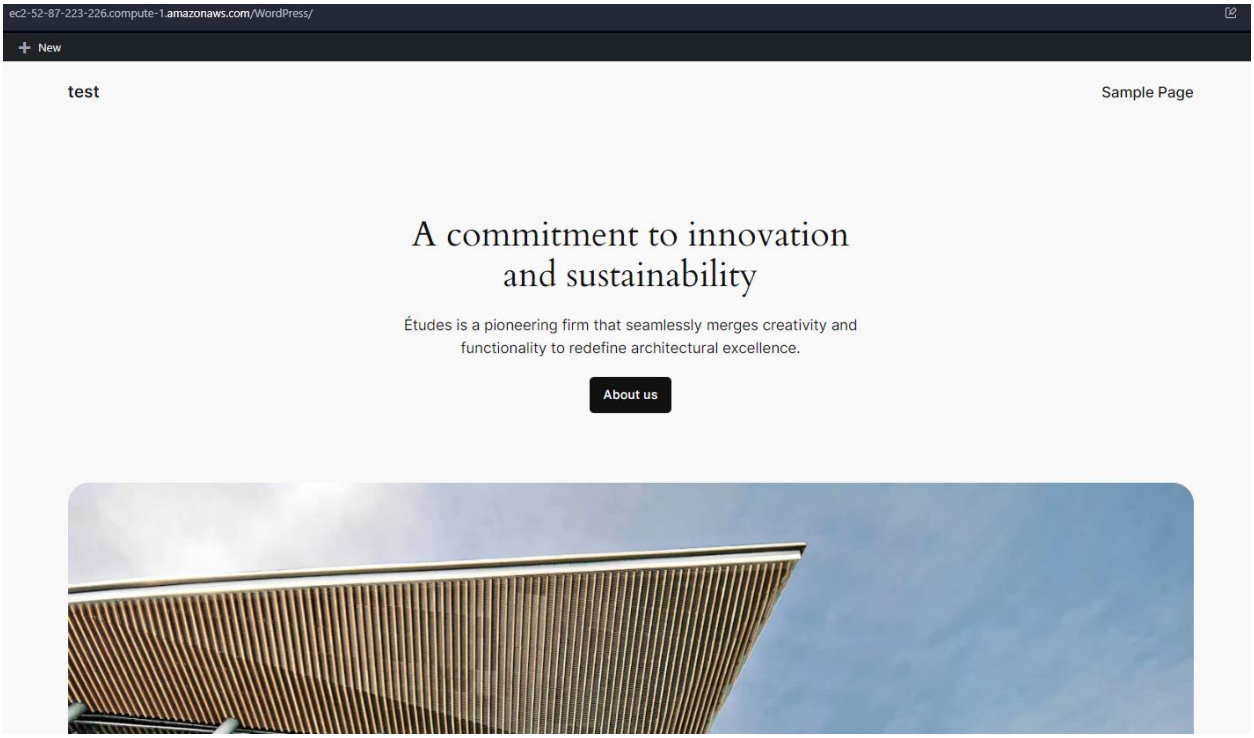
Install WordPress

Install WordPress



Successfully deploy

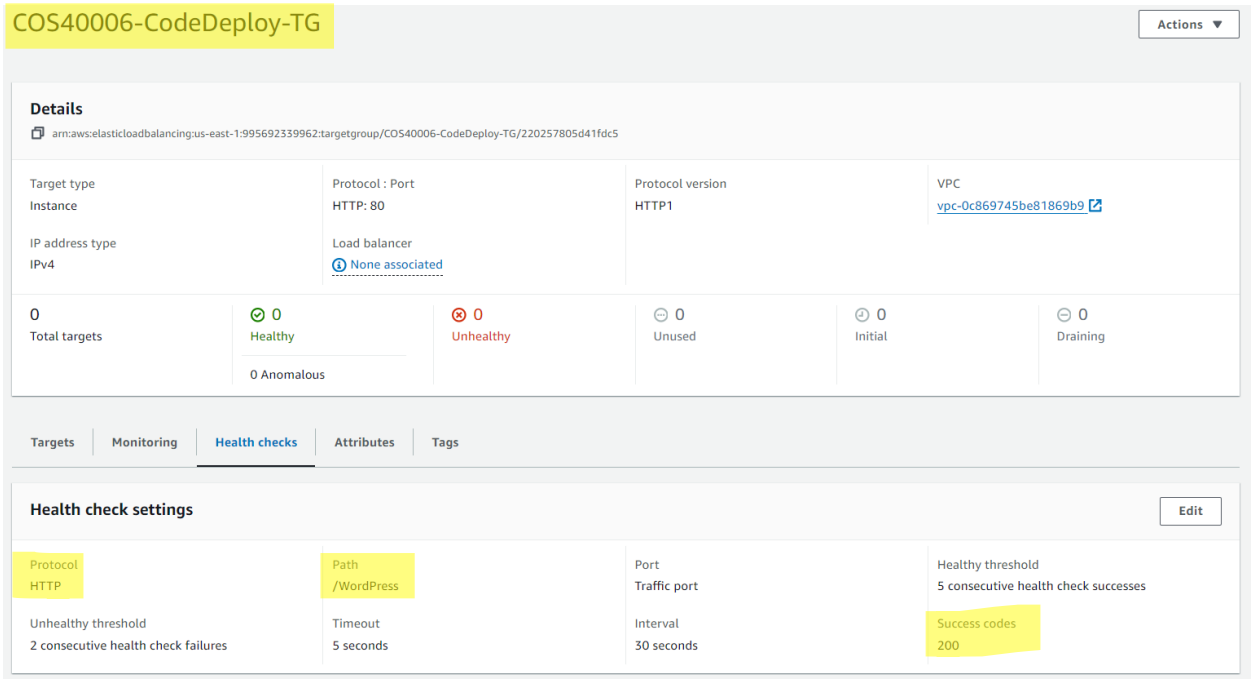




Task 2.2

Create an ELB

Create a target group, prepare for ELB creation



Create security group for ELB

sg-0fdd436de6b43382b - LoadBalanceSG

Actions

Details

Security group name	Security group ID	Description	VPC ID
LoadBalanceSG	sg-0fdd436de6b43382b	Allow SSH and HTTP from anywhere	vpc-0c869745be81869b9

Owner	Inbound rules count	Outbound rules count
995692339962	2 Permission entries	0 Permission entries

Inbound rules | Outbound rules | Tags

Inbound rules (2)

Manage tags | Edit inbound rules

Search

1

Name	Security group rule...	IP version	Type	Protocol	Port range	Source
–	sgr-011a714714fe5f4ad	IPv4	HTTP	TCP	80	0.0.0.0/0
–	sgr-0fc0793009f409b98	IPv4	SSH	TCP	22	0.0.0.0/0

Create an application ELB with the prepared resources

EC2 > Load balancers > Create Application Load Balancer

Create Application Load Balancer Info

The Application Load Balancer distributes incoming HTTP and HTTPS traffic across multiple targets such as Amazon EC2 instances, micros
on request attributes. When the load balancer receives a connection request, it evaluates the listener rules in priority order to determine w
applicable, it selects a target from the target group for the rule action.

▸ **How Application Load Balancers work**

Basic configuration

Load balancer name

Name must be unique within your AWS account and can't be changed after the load balancer is created.

COS40006-CodeDeploy-AELB

A maximum of 32 alphanumeric characters including hyphens are allowed, but the name must not begin or end with a hyphen.

Scheme Info

Scheme can't be changed after the load balancer is created.

- (●) Internet-facing
  An internet-facing load balancer routes requests from clients over the internet to targets. Requires a public subnet. Learn more
- ( ) Internal
  An internal load balancer routes requests from clients to targets using private IP addresses. Compatible with the **IPv4** and **Dualstack** IP address types.

IP address type Info

Select the type of IP addresses that your subnets use. Public IPv4 addresses have an additional cost.

- (●) IPv4
  Includes only IPv4 addresses.

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

LoadBalanceSG
sg-0fdd436de6b43382b VPC: vpc-0c869745be81869b9

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 COS40006-CodeDeploy-TG, Target type: Instance, IPv4, HTTP

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.

COS40006-CodeDeploy-AELB Actions

Details

Load balancer type	Status	VPC	IP address type
Application	Provisioning	vpc-0c869745be81869b9	IPv4
Scheme	**Hosted zone**	**Availability Zones**	**Date created**
Internet-facing	Z35SXDOTRQ7X7K	subnet-0d0d135a41406a18e us-east-1a (use1-az2) subnet-025975f8ff5fda79e us-east-1b (use1-az4)	May 23, 2024, 18:05 (UTC+07:00)

Load balancer ARN
arn:aws:elasticloadbalancing:us-east-1:995692339962:loadbalancer/app/COS40006-CodeDeploy-AELB/ec1f5ea9a40d0c07

DNS name Info
COS40006-CodeDeploy-AELB-1740144617.us-east-1.elb.amazonaws.com (A Record)

Listeners and rules | Network mapping | Resource map - *new* | Security | Monitoring | Integrations | Attributes | Tags

Network mapping Info

Edit IP address type | Edit subnets

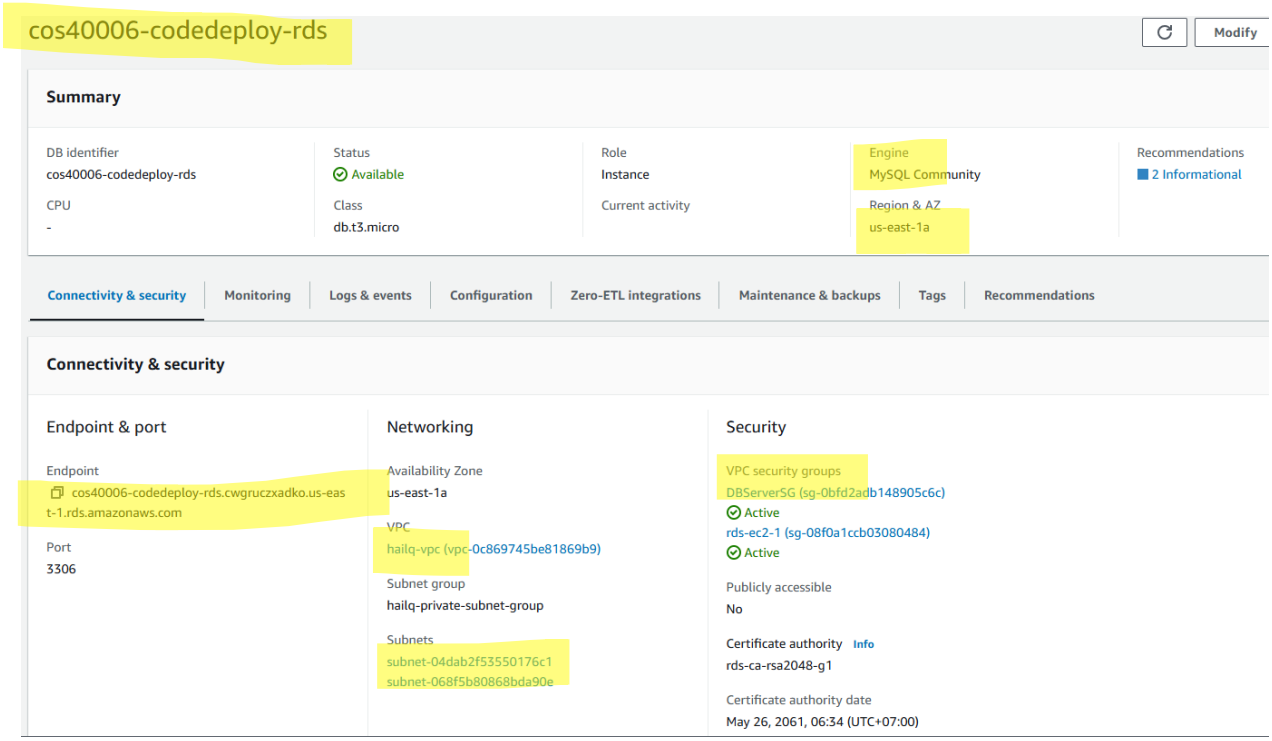
Targets in the listed zones and subnets are available for traffic from the load balancer using the IP addresses shown.

VPC
vpc-0c869745be81869b9
IPv4 VPC CIDR: 10.0.0.0/16
IPv6 : -

Load balancer IP address type
IPv4

Re-install WordPress using an external database

Create an AWS RDS database instance



Modify the settings in the wp-config.php file corresponding to the RDS database instance

```
// ** Database settings - You can get this info from your web host ** //
/** The name of the database for WordPress */
define( 'DB_NAME', 'test' );

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

/** Database password */
define( 'DB_PASSWORD', 'Hai204fromtheocean' );

/** Database hostname */
define( 'DB_HOST', 'cos40006-codedeploy-rds.cwgruczxadko.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', '' );
```

Modify the scripts/install_dependencies.sh file to install mysql-client instead of mariadb-server

```
#!/bin/bash
sudo dnf install php php-cli php-common php-mysqlnd php-json php-xml -y
# sudo dnf install mariadb105-server -y
sudo wget https://dev.mysql.com/get/mysql80-community-release-el9-1.noarch.rpm
sudo dnf install mysql80-community-release-el9-1.noarch.rpm -y
sudo rpm --import https://repo.mysql.com/RPM-GPG-KEY-mysql-2023
sudo dnf install mysql-community-client -y
sudo rm mysql80-community-release-el9-1.noarch.rpm
```

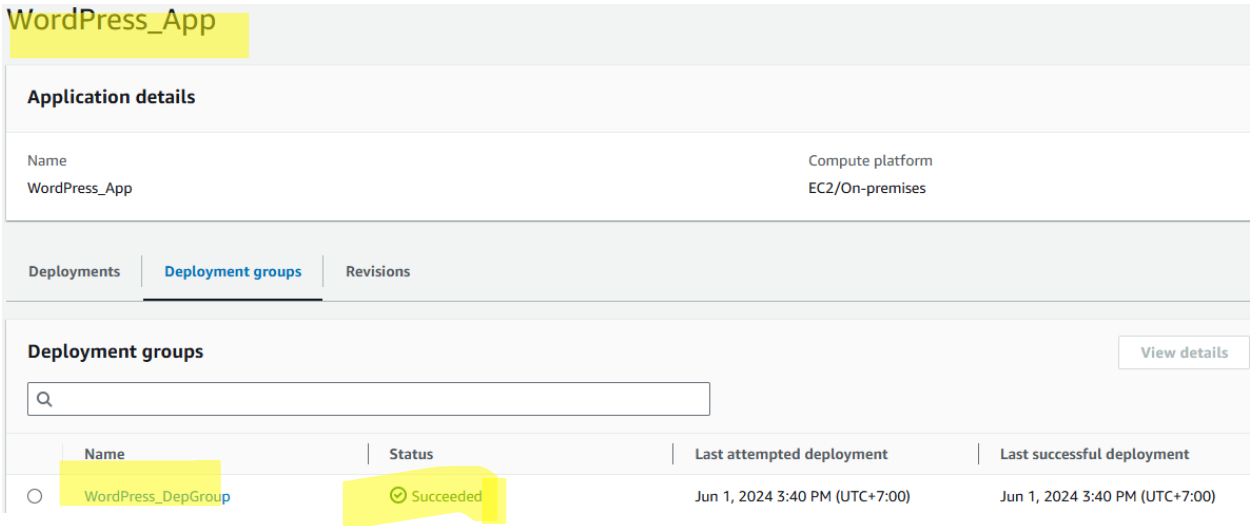
Modify the scripts/create_test_db.sh correspondingly to the RDS instance

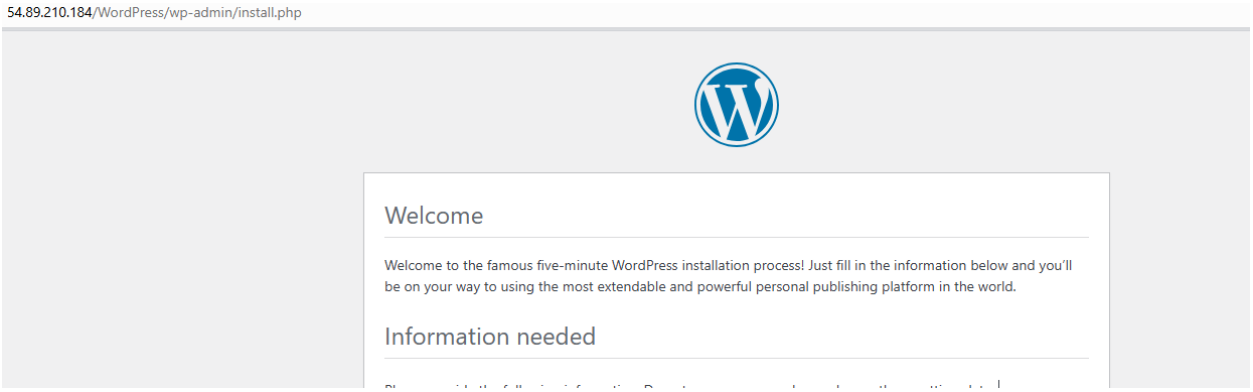
```
#!/bin/bash

# Connect to the MySQL RDS instance and create a new database if it doesn't already exist
mysql -u admin -pHai204fromtheocean -h cos40006-codedeploy-rds.cwgruczxadko.us-east-1.rds.amazonaws.com -P 3306 <<CREATE_TEST_DB
CREATE DATABASE IF NOT EXISTS test;
CREATE_TEST_DB
```

Modify the scripts/start_server.sh to no longer start a local database service

```
#!/bin/bash
# sudo systemctl start mariadb.service
sudo systemctl start httpd.service
sudo systemctl start php-fpm.service
```

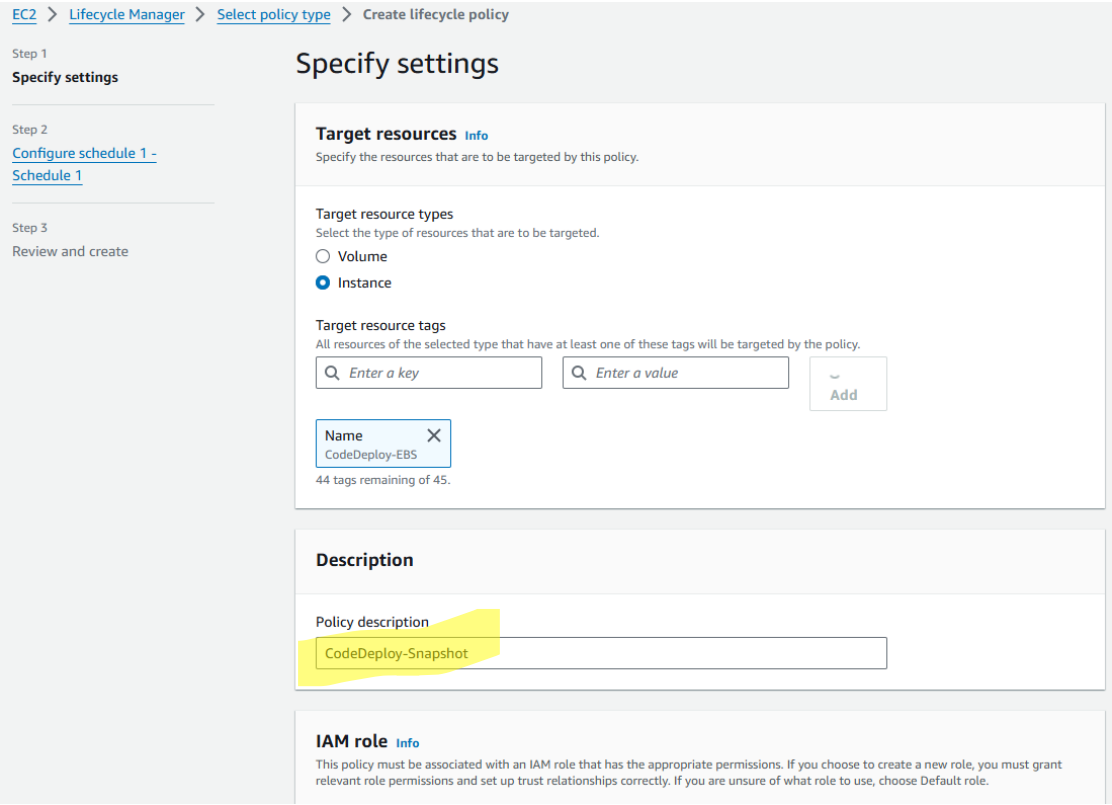
Deploy similarly to Task 2.1





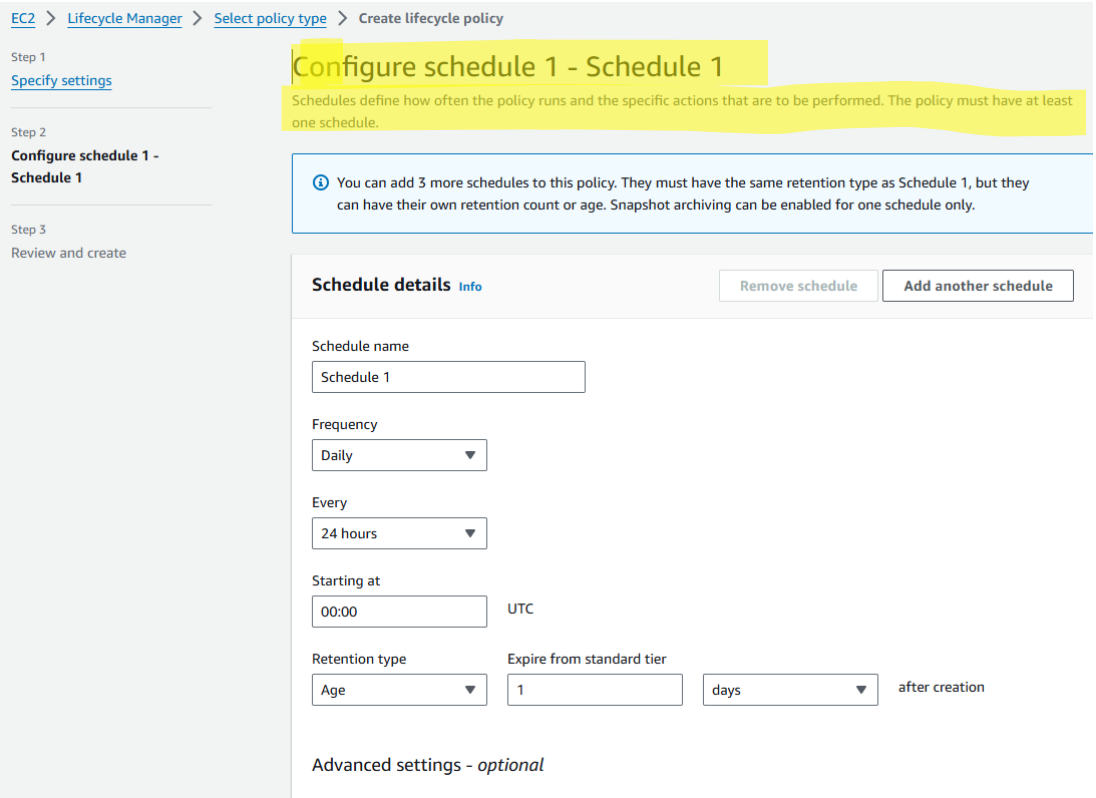
Backup instance to S3

To back up the instance, AWS provides the EBS snapshot creation. The snapshot is then stored in S3.

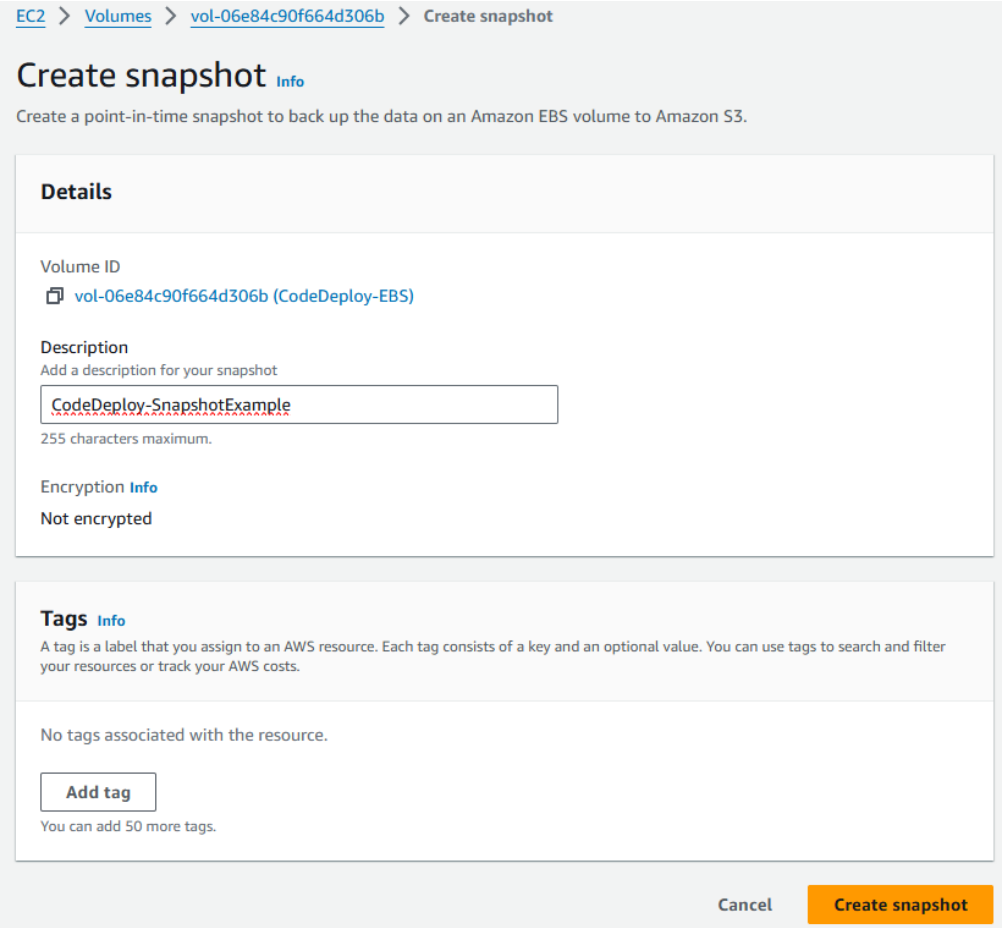


We create a schedule, specifically:

- **A snapshot is taken every day at 12 am**
- **The snapshot remains in S3 for 24 hours**

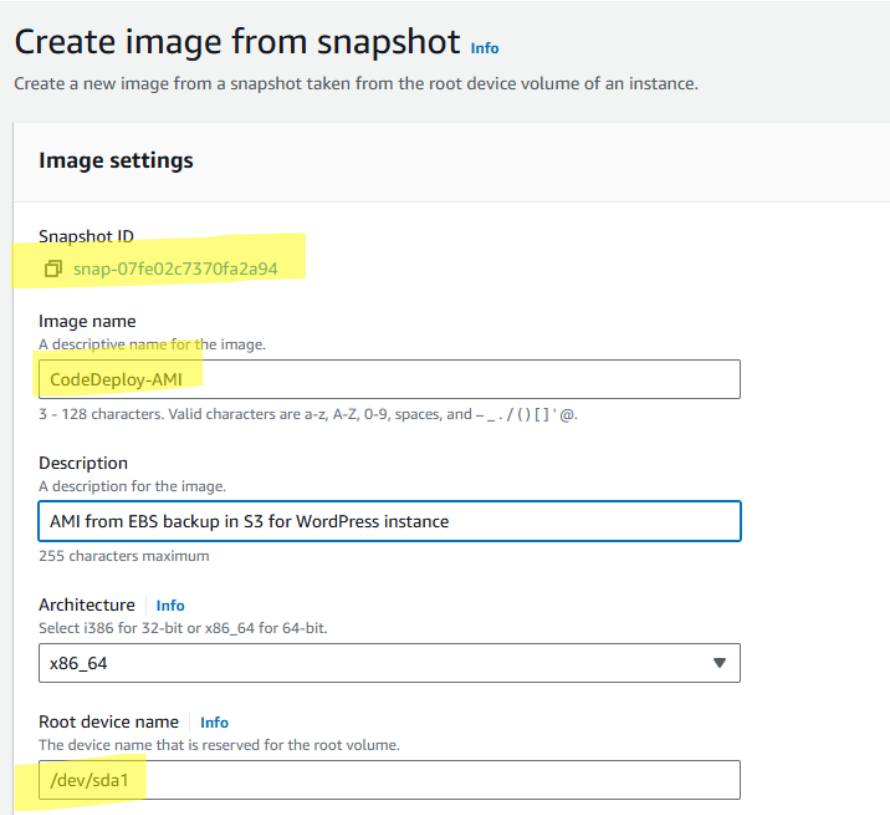


Now we ensure the backup is always the most up-to-date. For testing purposes, we create a snapshot manually

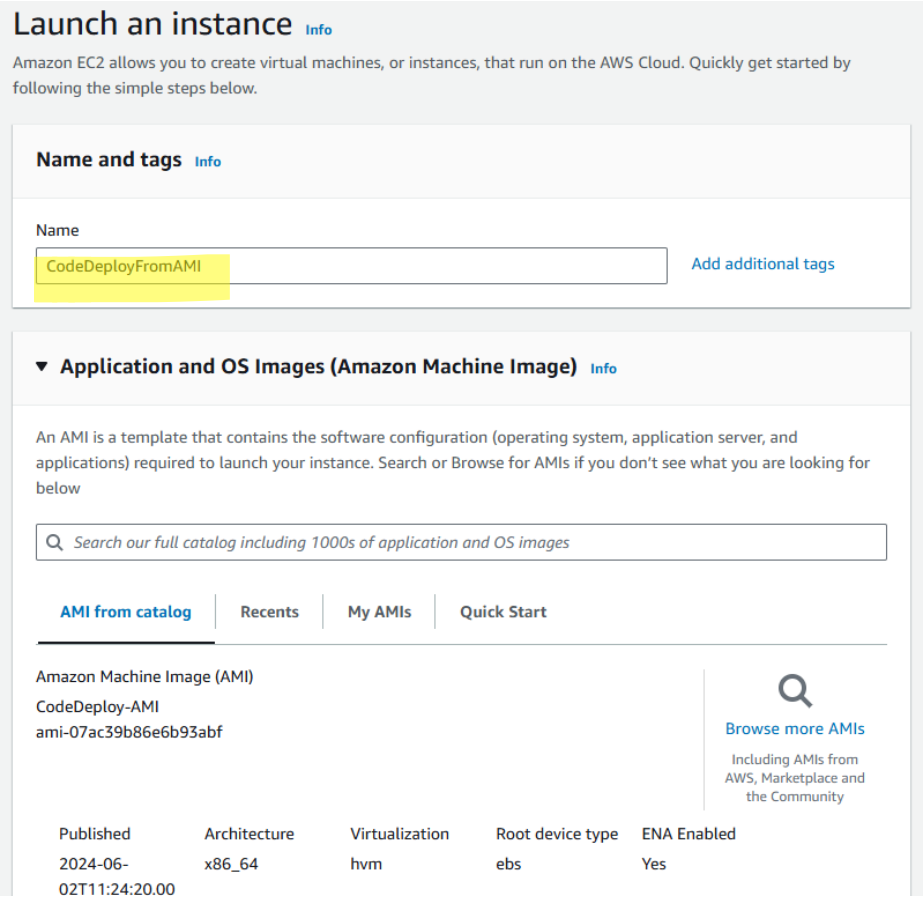


Create an instance from S3 backup

From the snapshot taken above, create an AWS AMI out of it



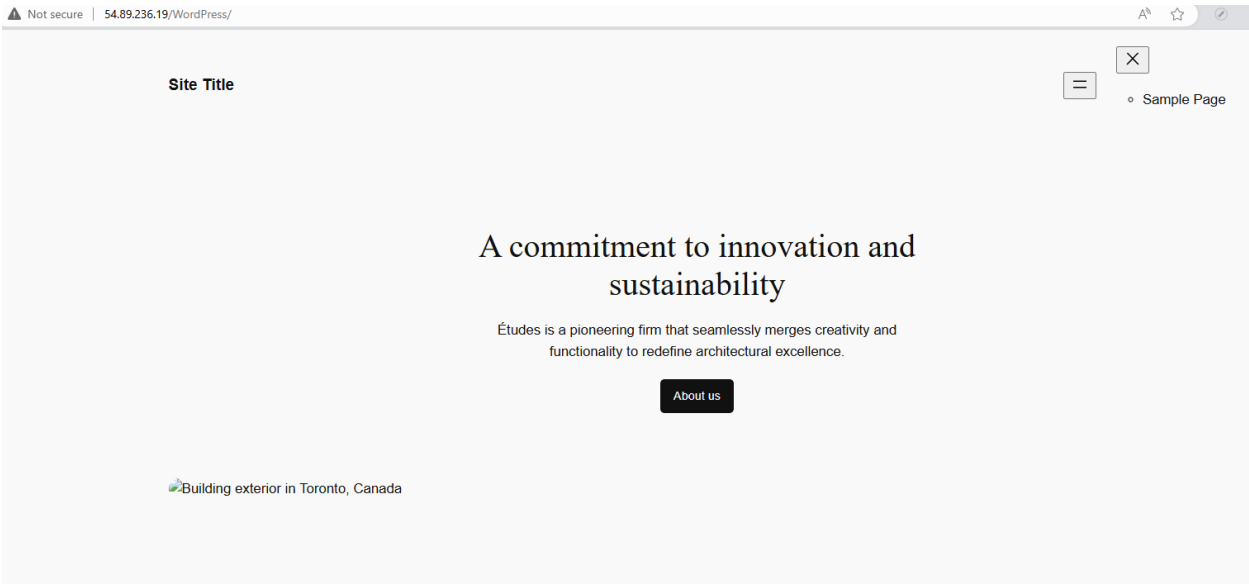
From the AMI create we can create an EC2 instance with an exact copy of the files in the CodeDeployDemo instance

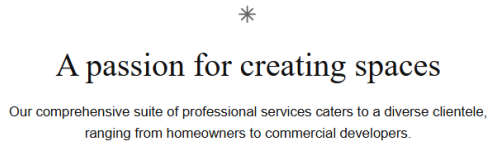


In order to listen on http request, the newly created instance must start httpd and php-frm service, add these commands to the data executed on boot up

```
#!/bin/bash
# sudo systemctl start mariadb.service
sudo systemctl start httpd.service
sudo systemctl start php-fpm.service
```

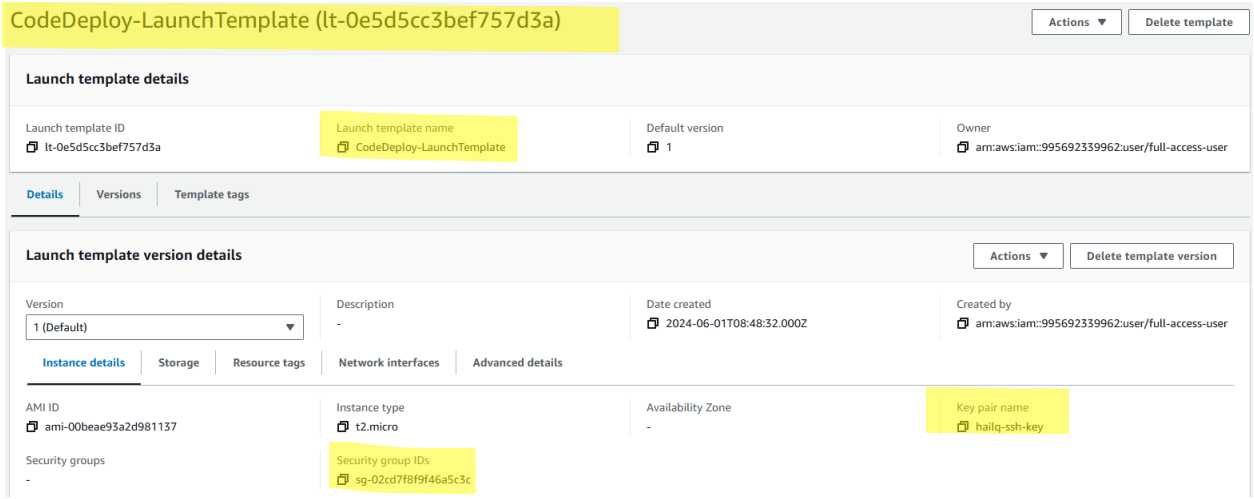
Now the WordPress site should be available for http request





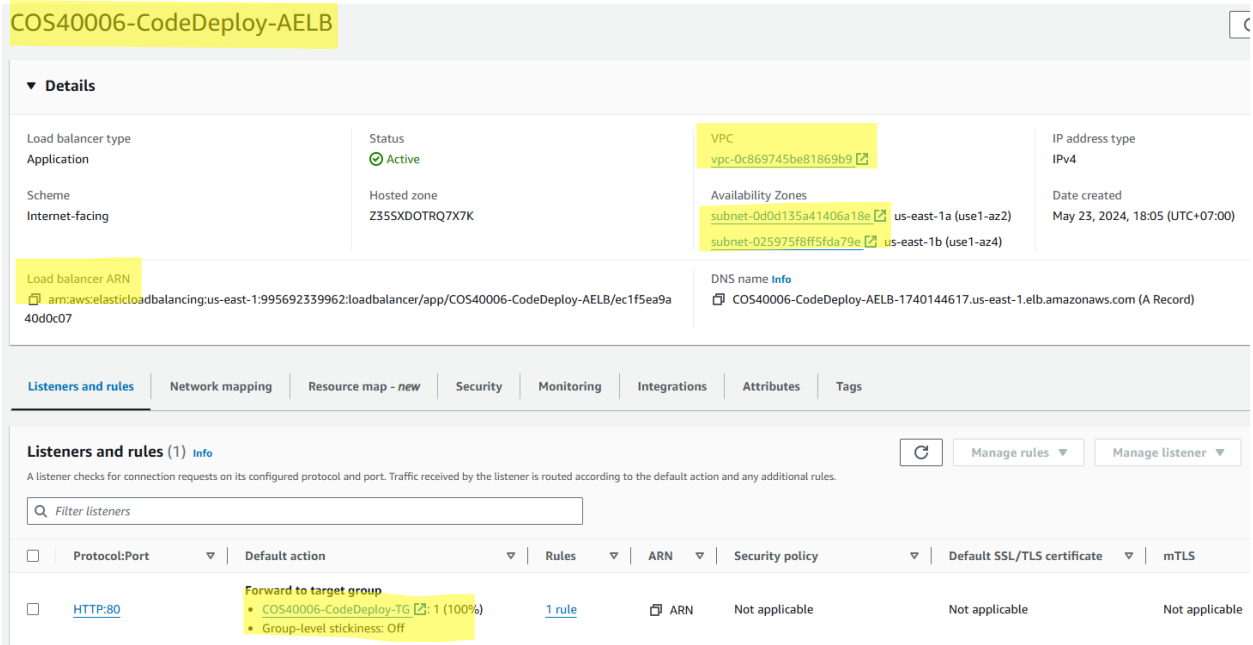
Task 2.3

Create a Launch Template (formerly Launch Configuration* now deprecated)

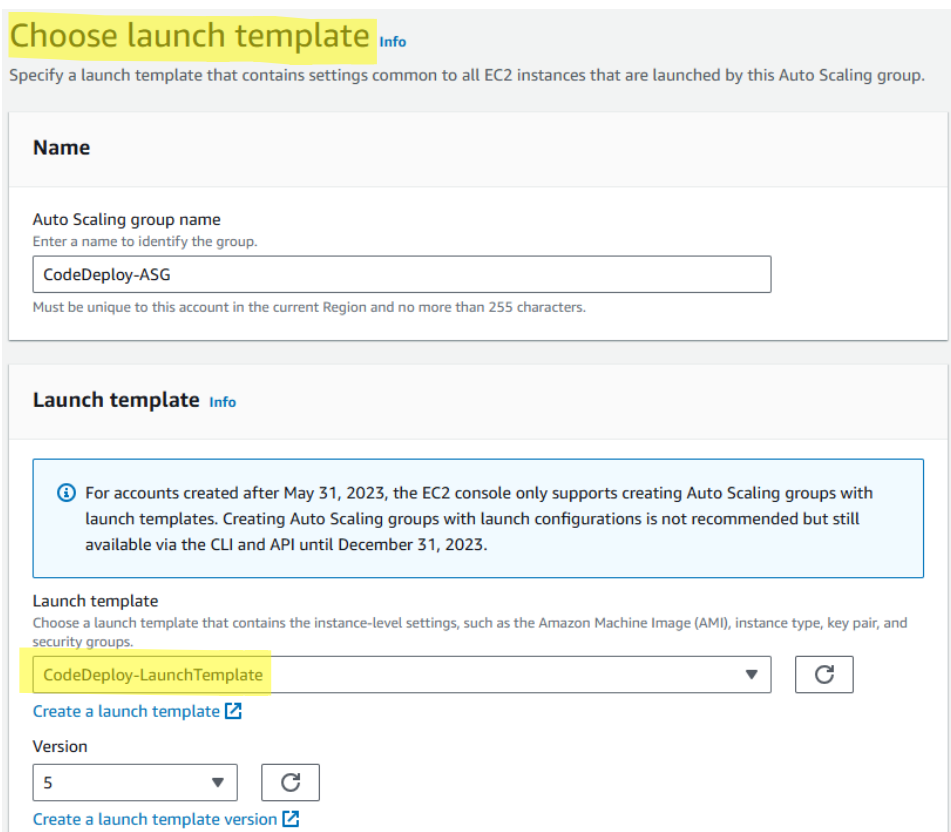


Create an ASG

Create an elastic application load balancer at layer 7 to prepare for ASG creation



Configure ASG to launch instance



Choose instance launch options Info

Choose the VPC network environment that your instances are launched into, and customize the instance types and purchase options.

Instance type requirements Info

Override launch template

You can keep the same instance attributes or instance type from your launch template, or you can choose to override the launch template by specifying different instance attributes or manually adding instance types.

Launch template	Version	Description
CodeDeploy-LaunchTemplate lt-0e5d5cc3bef757d3a	5	-

Instance type

t2.micro

Network Info

For most applications, you can use multiple Availability Zones and let EC2 Auto Scaling balance your instances across the zones. The default VPC and default subnets are suitable for getting started quickly.

VPC

Choose the VPC that defines the virtual network for your Auto Scaling group.

vpc-0c869745be81869b9 (hailq-vpc)
10.0.0.0/16

Create a VPC

Availability Zones and subnets

Define which Availability Zones and subnets your Auto Scaling group can use in the chosen VPC.

Select Availability Zones and subnets

us-east-1a | subnet-0d0d135a41406a18e (hailq-

Configure advanced options - *optional* Info

Integrate your Auto Scaling group with other services to distribute network traffic across multiple servers using a load balancer or to establish service-to-service communications using VPC Lattice. You can also set options that give you more control over health check replacements and monitoring.

Load balancing Info

Use the options below to attach your Auto Scaling group to an existing load balancer, or to a new load balancer that you define.

- ○ **No load balancer**
  Traffic to your Auto Scaling group will not be fronted by a load balancer.
- ◉ **Attach to an existing load balancer**
  Choose from your existing load balancers.
- ○ **Attach to a new load balancer**
  Quickly create a basic load balancer to attach to your Auto Scaling group.

Attach to an existing load balancer

Select the load balancers that you want to attach to your Auto Scaling group.

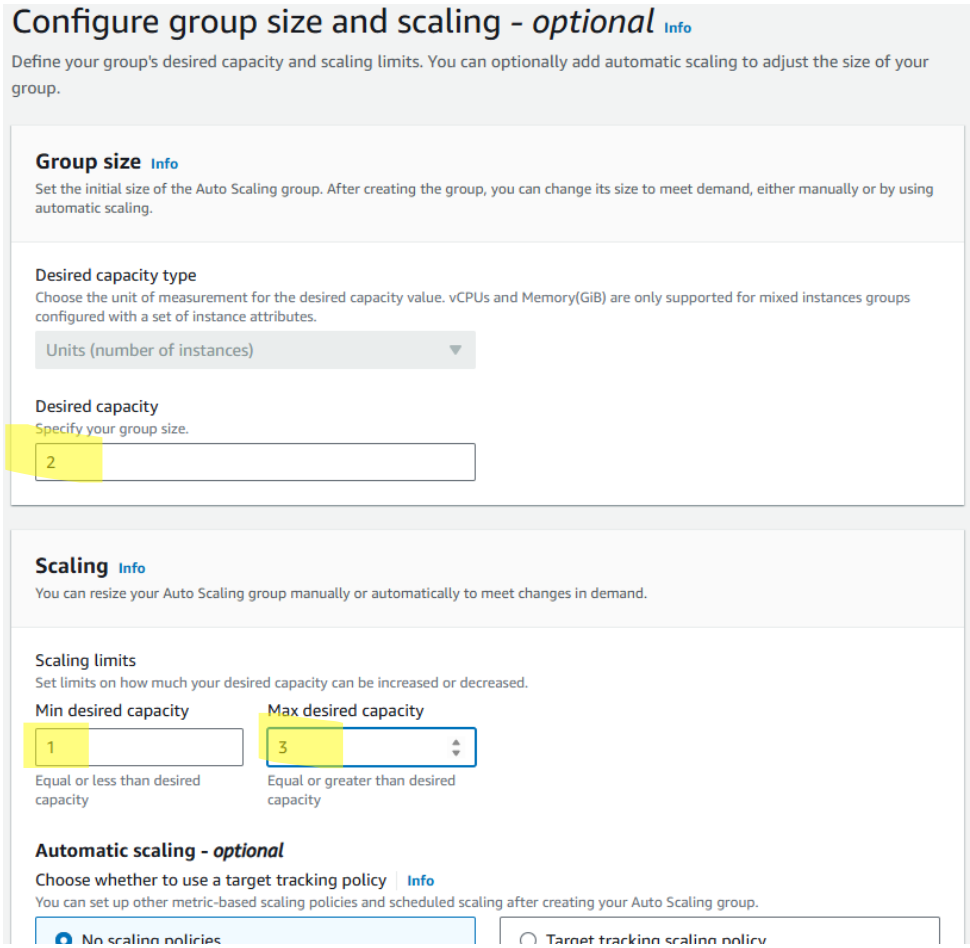
- ◉ **Choose from your load balancer target groups**
  This option allows you to attach Application, Network, or Gateway Load Balancers.
- ○ **Choose from Classic Load Balancers**

Existing load balancer target groups

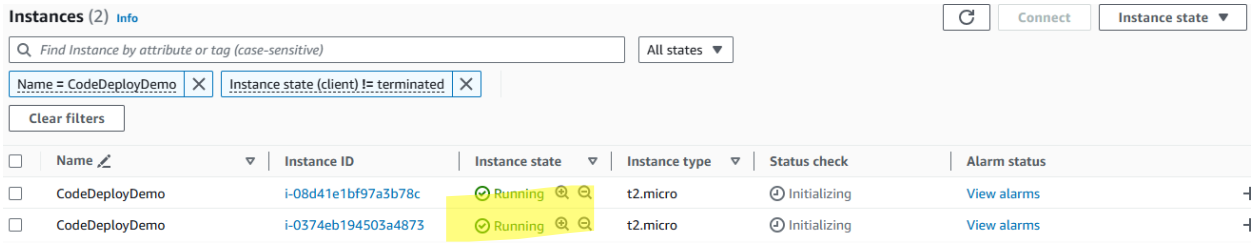
Only instance target groups that belong to the same VPC as your Auto Scaling group are available for selection.

Select target groups ▼ [↻]

COS40006-CodeDeploy-TG | HTTP ×
Application Load Balancer: COS40006-CodeDeploy-AELB



After creating ASG, 2 instances is initialized as specified in settings



Test scaling

On the instances, install "stress" with

```
sudo dnf install -y epel-release
sudo dnf install -y stress
```

Then overload CPU usage of the instances by

```
stress --cpu 4 --timeout 300
```

Now a new instance is automatically created

Name = CodeDeployDemo | Instance state (client) != terminated

Clear filters

Name	Instance ID	Instance state	Instance type	Status check	Alarm status
CodeDeployDemo	i-08d41e1bf97a3b78c	Running	t2.micro	...	View alarms
CodeDeployDemo	i-0374eb194503a4873	Running	t2.micro	...	View alarms
CodeDeployDemo	i-0f7d344d9d0737feb	Running	t2.micro	...	View alarms

Task 2.4

Connecting to instances via SSH

```
PS C:\Users\vanla> ubuntu
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 5.15.146.1-microsoft-standard-WSL2 x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
   just raised the bar for easy, resilient and secure K8s cluster deployment.

   https://ubuntu.com/engage/secure-kubernetes-at-the-edge

This message is shown once a day. To disable it please create the
/home/hailq/.hushlogin file.
hailq@lequanghai:~$ ls
CodeDeployDemo-EC2-Permissions.json               full-access-user_access-keys.txt
CodeDeployDemo-EC2-Trust.json                     hailq-ssh-key.pem
full-access-user_access-keys.csv:Zone.Identifier  hailq-ssh-key.pem:Zone.Identifier
hailq@lequanghai:~$ ssh -i "hailq-ssh-key.pem" ec2-user@ec2-44-201-158-110.compute-1.amazonaws.com
The authenticity of host 'ec2-44-201-158-110.compute-1.amazonaws.com (44.201.158.110)' can't be established.
ED25519 key fingerprint is SHA256:aDBTcuUxABJ+6Y+Cx2dr1p/Por/4AcDNRZMqU9z0lDc.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-44-201-158-110.compute-1.amazonaws.com' (ED25519) to the list of known hosts.
       #_
       ~\_  ####_        Amazon Linux 2023
      ~~  \_#####\
      ~~     \###|
      ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
       ~~       V~' '->
        ~~~         /
          ~~._.   _/
             _/ _/
           _/m/'
[ec2-user@ip-10-0-0-196 ~]$
```

All the tasks above can be automated completely via AWS CLI

Before starting, to avoid error, delete the existing resources (perhaps from last deployment, etc.)

```
aws deploy delete-application --application-name WordPress_App
aws deploy create-application --application-name WordPress_App
aws s3 rm s3://cos40006-code-deploy-bucket/WordPressApp.zip
```

Create a AWS CodeDeploy Application via CLI

```
aws deploy create-application --application-name WordPress_App
```

Push the current directory to S3 and attach it with the Application created via CLI

```
aws deploy push \
  --application-name WordPress_App --s3-location s3://cos40006-code-deploy-bucket/WordPressApp.zip --ignore-hidden-files
```

Create a deployment group for the application

```
aws deploy create-deployment-group \
  --application-name WordPress_App \
  --deployment-group-name WordPress_DepGroup \
  --deployment-config-name CodeDeployDefault.OneAtATime \
  --ec2-tag-filters Key=Name,Value=CodeDeployDemo,Type=KEY_AND_VALUE \
  --service-role-arn arn:aws:iam::995692339962:role/CodeDeployServiceRole
```

Deploy the application from the using that deployment group

```
aws deploy create-deployment \
  --application-name WordPress_App \
  --deployment-config-name CodeDeployDefault.OneAtATime \
  --deployment-group-name WordPress_DepGroup \
  --s3-location bucket=cos40006-code-deploy-bucket,bundleType=zip,key=WordPressApp.zip
```

Create a launch template via CLI

```
aws ec2 create-launch-template \
  --launch-template-name CodeDeploy-LaunchTemplate \
  --version-description "Version 1" \
  --launch-template-data '{
      "ImageId": "ami-0abcdef1234567890",
      "InstanceType": "t2.micro",
      "KeyName": "hailq-ssh-key",
      "SecurityGroupIds": ["sg-02cd7f8f9f46a5c3c"],
      "UserData": "IyEvYmluL2Jhc2gKc3VkbyBkbmYgaW5zdGFsbCAtWSBlcGVsLXJlbGVhc2UKc3VkbyBkbmYgaW5zdGFsbCAtWSBzdHJlc3MKc3RyZXNzIC0tY3B1IDQgLS10aW1lb3V0I
      "TagSpecifications": [{
          "ResourceType": "instance",
          "Tags": [{
              "Key": "Name",
              "Value": "CodeDeployDemo"
          }]
      }]
  }'
```

Launch an EC2 instance from the created launch template

```
aws ec2 run-instances \
    --launch-template LaunchTemplateName=CodeDeploy-LaunchTemplate,Version=1
```

Create an application elastic load balancer

```
aws elb create-load-balancer \
    --load-balancer-name CodeDeploy-AELB \
    --listeners "Protocol=HTTP,LoadBalancerPort=80,InstanceProtocol=HTTP,InstancePort=80" \
    --subnets subnet-0d0d135a41406a18e subnet-025975f8ff5fda79e \
    --security-groups sg-0fdd436de6b43382b
```

Finally create an auto scaling group via CLI

```
aws autoscaling create-auto-scaling-group \
    --auto-scaling-group-name CodeDeploy-ASG \
    --min-size 2 \
    --max-size 5 \
    --desired-capacity 3 \
    --vpc-zone-identifier subnet-0d0d135a41406a18e subnet-025975f8ff5fda79e \
    --availability-zones us-east-1a us-east-1b \
```