AWS Hands-on Training

Purpose of This Assignment

The purpose of this assignment is as follows.

1. **To understand how to use basic AWS services.**
2. **To acquire practical infrastructure-building skills.**

Ultimately, the goal is to reach a level where you can take charge of infrastructure construction in real-world projects.

Assignment Overview

In this assignment, you will build a WordPress site using a standard configuration that our company also employs in real-world projects.

Important Notes

- Please enable a virtual MFA device for the provided IAM user.
- Use the ap-northeast-1 region.
- Wherever {IAM username} is mentioned, replace it with your own IAM username.
- If you are instructed to add tags, ensure they are applied, as resources may not be created without the correct tags.
- You will not be able to work if you access from an IP address other than COOSY's. Make sure you are connected to COOSY's network before starting.
- EC2 instances will automatically stop every day at 11:00 PM Japan time and will automatically start at 8:00 AM on weekdays.

Level1

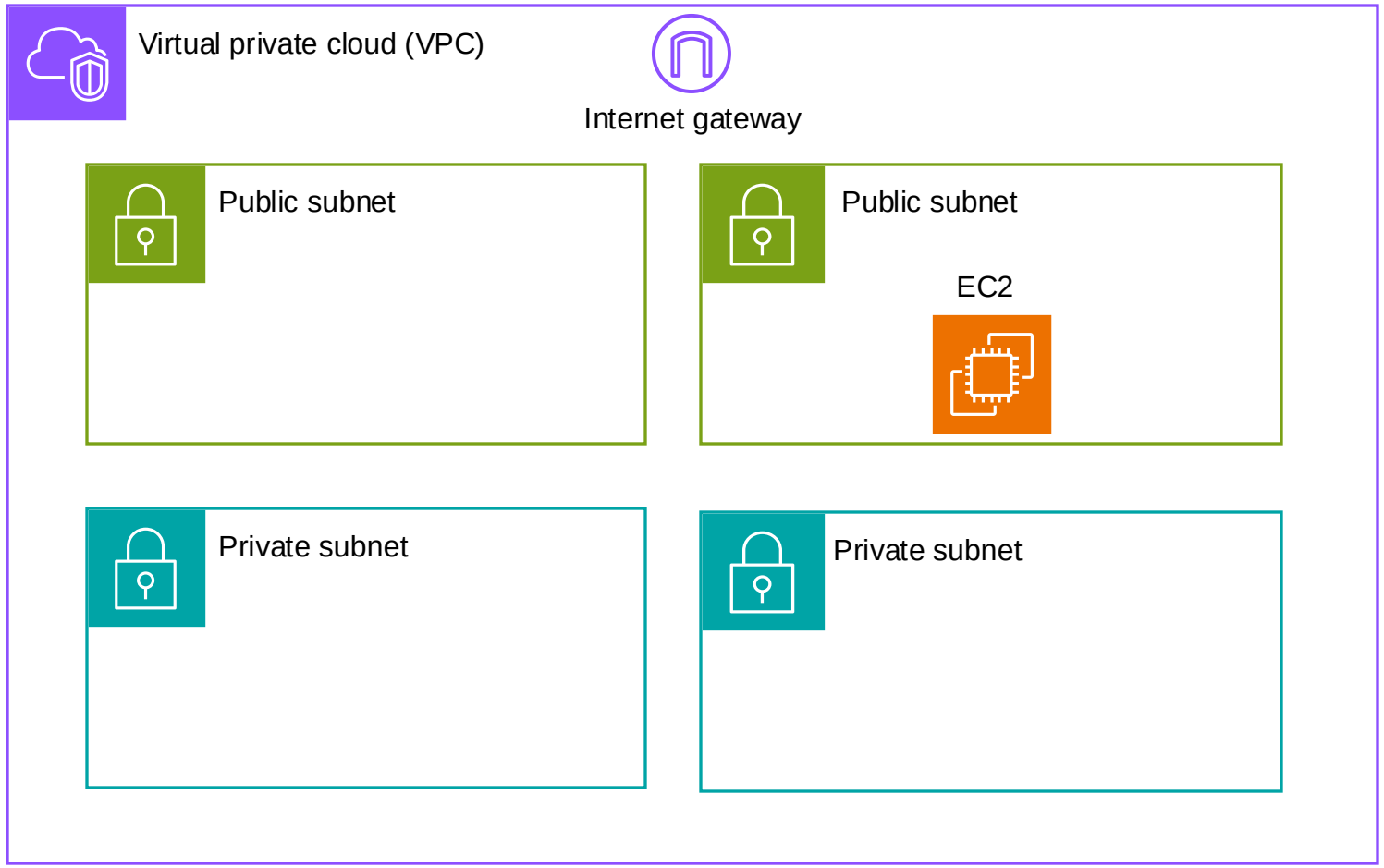
1-1 Building Network Resources and EC2 Instances

Resources to Create

- Security Group (for EC2 and ALB)

Note: ALB will be created in a later step

- EC2 instance



Requirements

1. **Security Groups**
    - For ALB
        - **Name:** coosy-training-alb-{IAM username}
        - **VPC**: coosy-training
        - **Inbound:** Allow all traffic on port 80
        - **Outbound:** Allow all traffic
        - **Tags**
            - **Key**: owner, **Value**: {IAM username}
    - For EC2
        - **Name:** coosy-training-ec2-{IAM username}
        - **VPC**: coosy-training
        - **Inbound:** Allow only traffic on port 80 from the ALB security group
        - **Outbound:** Allow all traffic
        - **Tags**
            - **Key:** owner, **Value:** {IAM username}

2. **EC2**
    - **Name**: coosy-training-{IAM username}
    - **AMI**: Amazon Linux 2023 AMI
    - **Instance type**: t2.micro
    - **Key pair:** Proceed without a key pair (*Access to EC2 is to be done using Session Manager.)
    - **VPC**: coosy-training
    - **Subnet**: coosy-training-public-1c Or coosy-training-public-1a

    *From a security perspective, it is preferable to place the EC2 instance in a private subnet. However, this would require a NAT Gateway for external communication, which incurs additional costs. Therefore, in this case, the instance will be placed in a public subnet.

    - **Auto-assign public IP**: Enable
    - **Security group**: use the one for EC2 created in step 1.
    - **Termination protection** : Enable
    - **Tags**
        - **Key**: owner, **Value**: {IAM username}
        - **Key**: AutoStartStop, **Value**: enabled

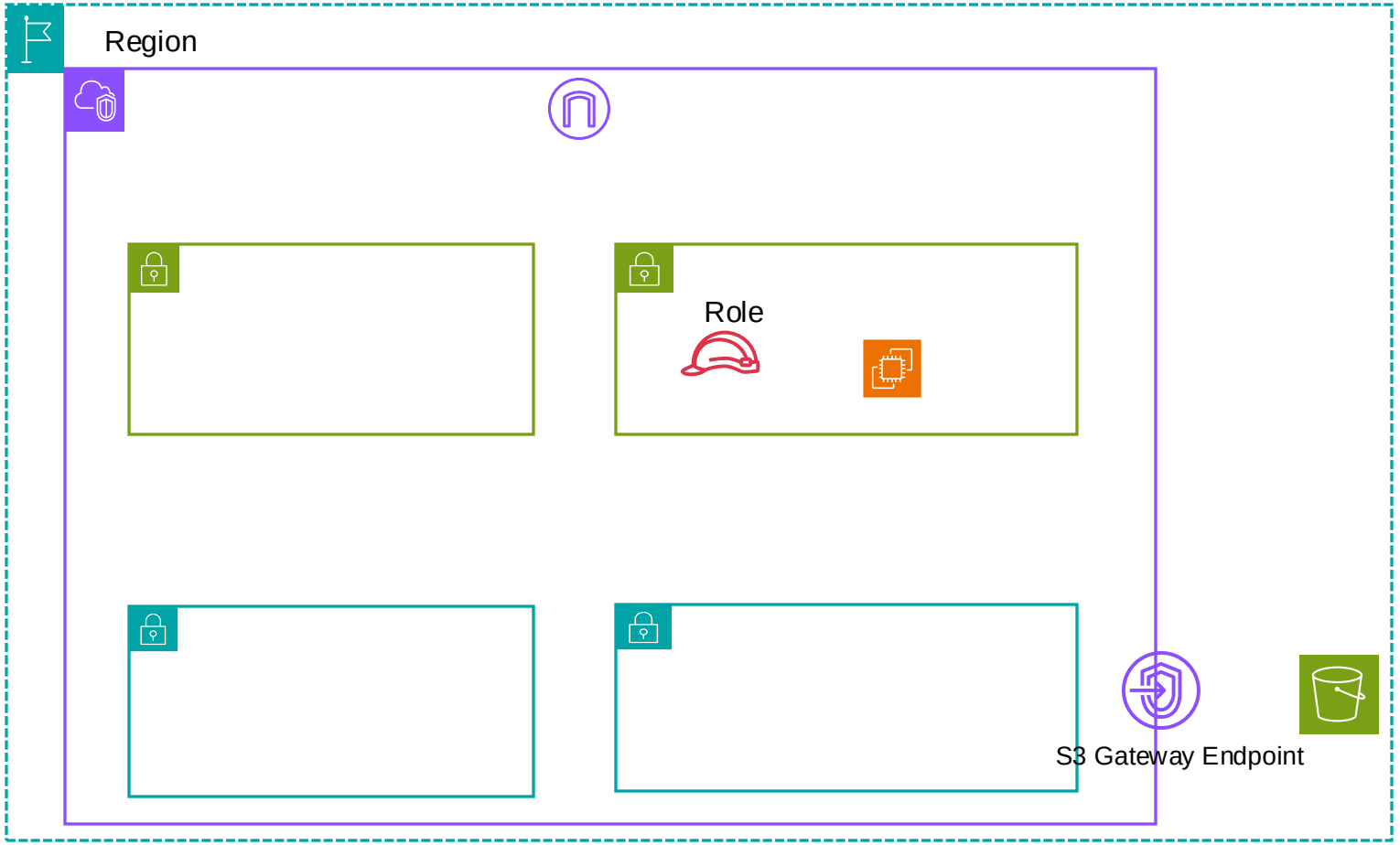
Tasks to be performed after resource creation

1. Connect to EC2 using Session Manager
2. Update dnf packages
3. Install & start & enable autostart for Apache
4. Install & start & enable autostart for MariaDB

1-2 Creation of resources related to the S3 bucket

Resources to Create

- S3 bucket
- IAM Role(For EC2)
- IAM Policy(For EC2)



Requirements

1. **S3 Bucket**
   - **Name**: coosy-training-{IAM usernmae}-{Today's date (YYYYmmdd)}

   e.g, coosy-training-ichiro-suzuki-20240823
2. **IAM Policy**
   - Allow all S3 actions limited to the S3 bucket created in step 1
   - **Name**: coosy-training-{IAM username}-ec2-policy
   - **Tags**
     - **Key**: owner, **Value**: {IAM username}
3. **IAM Role**
   - **Service or use case**: EC2
   - **Policy**: the policy created in step 2
   - **Permission Boundary**: AmazonS3FullAccess
   - **Name**: coosy-training-{IAM username}-ec2-role
   - **Tags**
     - **Key**: owner, **Value**: {IAM username}

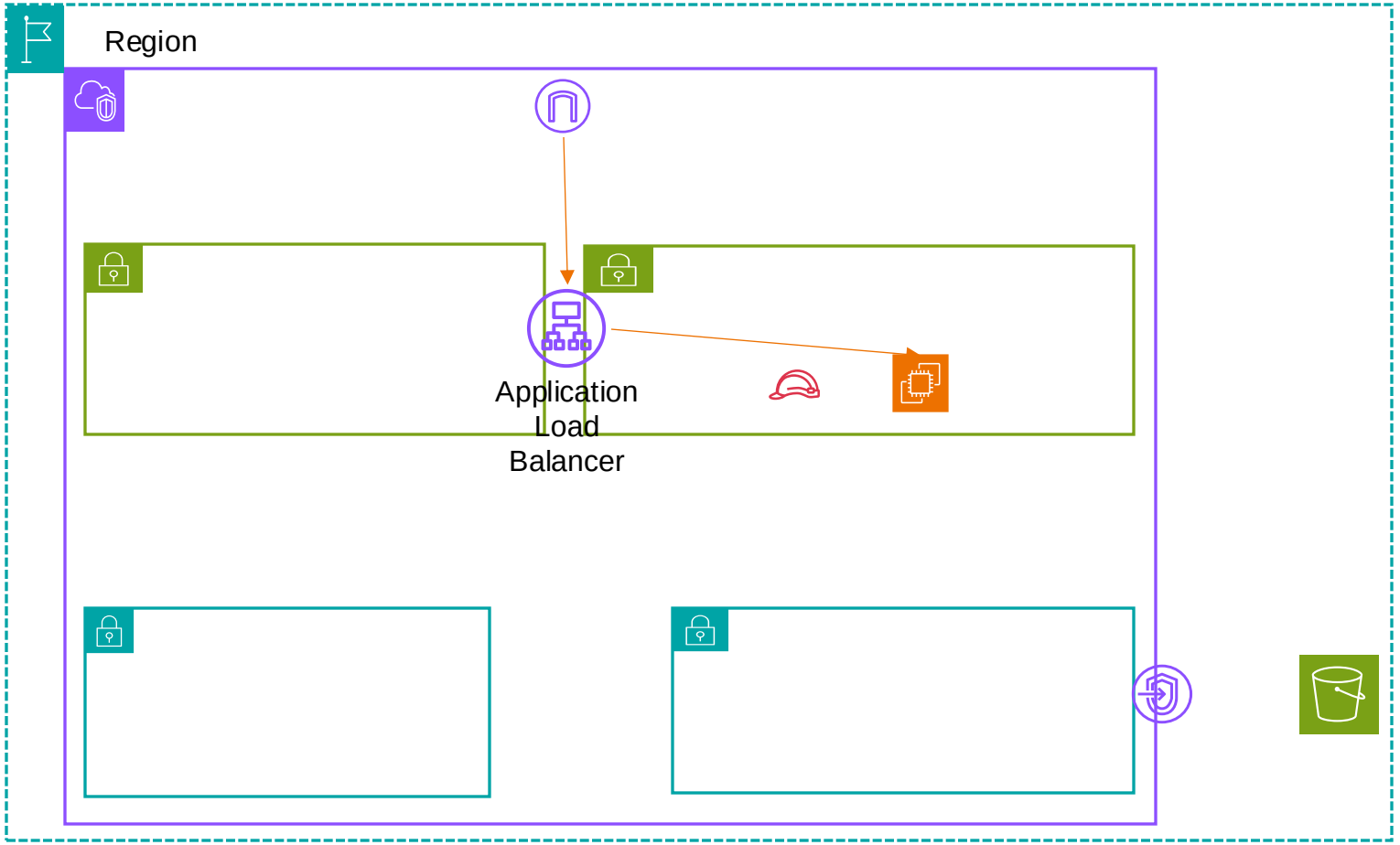
Tasks to be performed after resource creation

1. Attach the IAM role from step 3 to the EC2 instance
2. Connect to the EC2 instance using Session Manager, and verify if the EC2 instance can operate the S3 bucket by running the following command (check if the region can be retrieved).
   - aws s3api get-bucket-location --bucket {bucket name}

1-3 Creation of resources related to the ALB

Resources to Create

- ALB
- ALB Listener
- Listener Rule
- Target Group



Requirements

1. **TargetGroup**
   - **Name**: tg-{IAM username}
   - **Protocol**: HTTP, Port: 80
   - **VPC**: coosy-training
   - **Tags**
     - **Key**: owner , **Value**: {IAM username}
   - Register the EC2 instance created in step 1-1 as a target
2. **ALB**
   - **Name**: alb-{IAM username}
   - **VPC**: coosy-training
   - **Subnets**
     - coosy-training-public-1a
     - coosy-training-public-1c
   - **Security Groups**: the security group for ELB created in step 1-1
   - **Listener**
     - **Protocol**: HTTP
     - **Port**: 80
     - **Forward to**: the target group created in 1
     - **Listener Tags**
       - **Key**: owner, **Value**: {IAM username}
   - **Load balancer tags**
     - **Key**: owner, **Value**: {IAM username}

4. **ListerRule**
    - **Name**: AllowOnlyCoosy
    - **Tags**
        - **Key**: owner, **Value**: {IAM username}
    - **Condition**
        - **Type**: source ip
        - **Source IP** : 203.140.138.74/32, 116.82.244.198/32
    - **Routing**: Forward to target group
    - Target group: the targetgroup created in 1

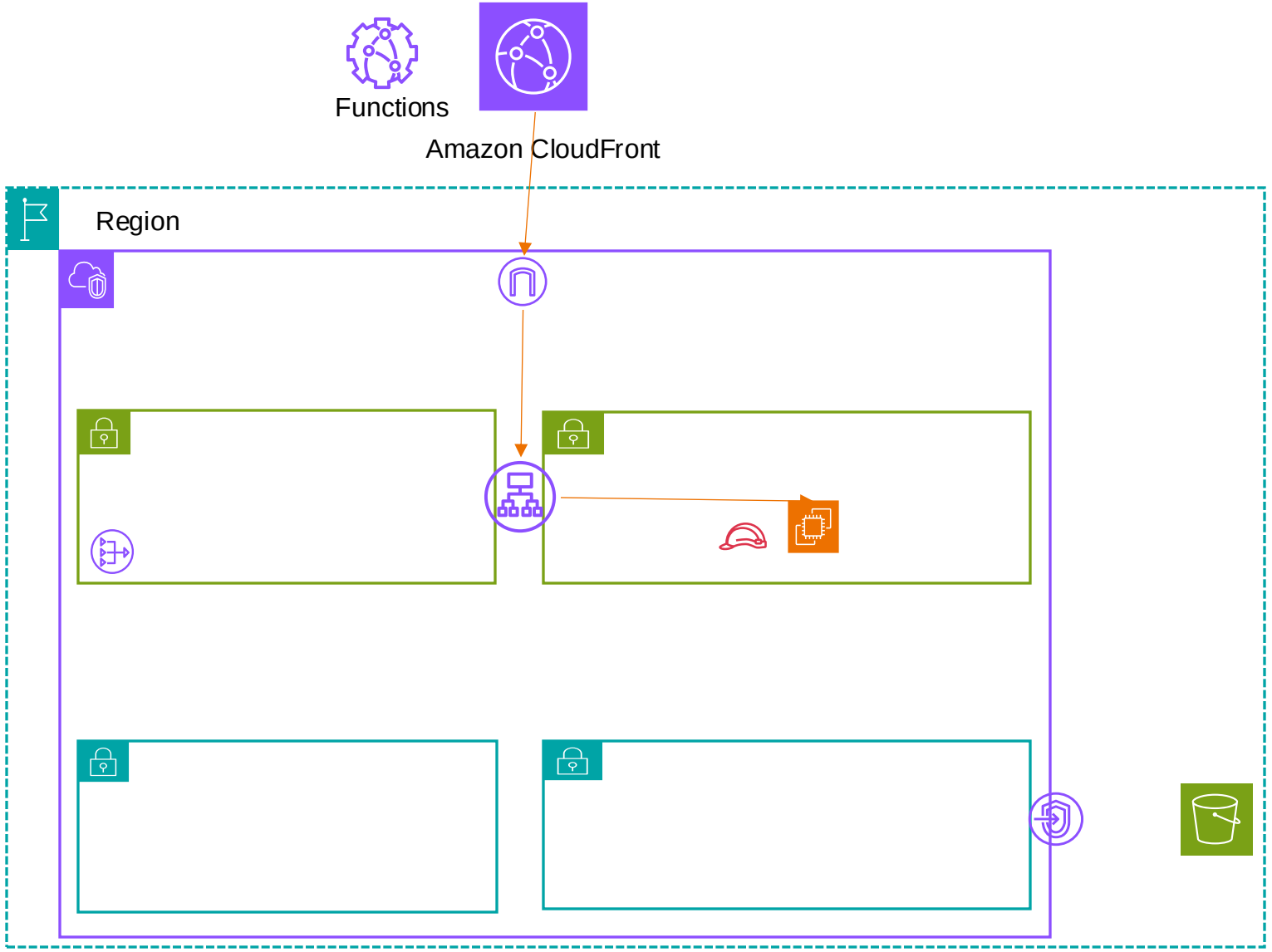
Tasks to be performed after resource creation

1. Edit the default listener rule
    - **default action**: return fixed response
    - **response code**: 403
    - **content**: Acccess Denied
2. Access the ALB domain and check if the Apache welcome page is displayed
3. Verify that a 403 response is returned when accessing from any IP other than COOSY's IP

1-4 Creation of resources related to the CloudFront

Resources to Create

- CloudFront distribution
- CloudFront Function(For basic auth)
- OAC(Origin Access Control)



Requirements

1. **CloudFront distribution**
    - **Origin domain**: The ALB created in step 1-3
    - **Protocol**: HTTP only
    - **Custom Header**: Set an arbitrary string of 12 characters or more using the key 'X-Auth-Token’
    - **Default cache behaivor**
        - **Viewer protocol policy**: HTTPS only
        - **Cache policy**: CachingOptimized
        - **Origin request policy**: Managed-AllViewerAndCloudFrontHeaders-2022-06
    - **WAF**: Disable
    - **Price class** : Use only North America and Europe

2. **CloudFront Function**
    - Associate with all viewer requests of the CloudFront distribution created in step 1
    - **Name**: basic-auth-{IAM username}
    - * The method to perform Basic Authentication with a CloudFront Function can be found by searching, so please look it up.

Tasks to be performed after resource creation

1. Set a tag with the key: owner and value: {IAM username} on the created CloudFront distribution
2. Add the S3 bucket created in step 1-2 as an origin to the CloudFront distribution.
   - **Origin Access**: Origin access control settings(Please create a new OAC)
   - **OAC Name**: oac-{IAM username}

     ※When you create the OAC, a button to copy the S3 bucket policy will be displayed. Please copy it and apply it to the relevant S3 bucket.
3. Add behaivors
   1. **For WordPress Dashboard**
      - **Path pattern**: /wp-admin/*
      - **Origin**: the ALB created in step 1-3
      - **Viewer protocol policy**: HTTPS only
      - **Allowed HTTP method**: GET, HEAD, OPTIONS, PUT, POST, PATCH, DELETE
      - **Cache policy**: CachingDisabled
      - **Origin request policy**: Managed-AllViewerAndCloudFrontHeaders-2022-06
      - **Function Associations**
        - **Viewer request**: The CloudFront Functions created earlier
   2. **For uploads directory**
      - **Path pattern**: /wp-content/uploads/*
      - **Viewer protocol policy**: HTTPS only
      - **Allowed HTTP method**: GET, HEAD, OPTIONS, PUT, POST, PATCH, DELETE
      - **Cache policy**: CachingOptimized
      - **Function Associations**
        - **Viewer request**: The CloudFront Functions created earlier

3. **For login page**
    - **Path pattern**: /wp-login*
    - **Origin**: the ALB created in step 1-3
    - **Viewer protocol policy**: HTTPS only
    - **Allowed HTTP method**: GET, HEAD, OPTIONS, PUT, POST, PATCH, DELETE
    - **Cache policy**: CachingDisabled
    - **Origin request policy**: Managed-AllViewerAndCloudFrontHeaders-2022-06
    - **Function Associations**
        - **Viewer request**: The CloudFront Functions created earlier

4. Add and edit conditions in the listener rules for the ALB
    - Edit the conditions for AllowOnlyCoosy
        - Condition type: HTTP header
        - HTTP header name: CloudFront-Viewer-Address
        - HTTP header value
            - 203.140.138.74:*
            - 116.82.244.198:*

    - Add a condition for AllowOnlyCoosy
        - Condition type: HTTP header
        - HTTP header name: X-Auth-Token
        - HTTP header value: The value set in the custom header of CloudFront

5. Access the page
    - Access the CloudFront domain and verify that the Apache welcome page is displayed (also confirm that Basic Authentication is functioning)
    - Access the ALB domain and confirm that a 403 response is returned

1-5 Server setup & WordPress installation

Tasks

1. **Setup of PHP and Apache**
    1. **Setup PHP**
        1. Install php, php-mysqlnd, php-gd, php-intl, php-zip
        2. Start php-fpm and enable autostart
    2. **Change the Apache configuration**
        1. Create a directory for vhosts configuration (/etc/httpd/conf.d/vhosts)
        2. Place a configuration file in /etc/httpd/conf.d/ that includes the setting to load .conf files under /etc/httpd/conf.d/vhosts.
        3. Create a config file with the vhosts settings in /etc/httpd/conf.d/
            - Write the configuration here to pass access to PHP files to php-fpm
        4. Restart the Apache
2. **Setup DB**
    1. **Run mysql_secure_installation**
        - Set the root user password at this time
    2. Create a MySQL user for WordPress and a database for WordPress
    3. Grant all privileges on the WordPress database to the MySQL user for WordPress

3. **WordPress installation**
    1. Install git
    2. Clone the Git repository containing the WordPress source into /var/www/html
4. **Setup WordPress**
    1. Copy /var/www/html/wp-config-sample.php to create /var/www/html/wp-config.php
    2. Enter the various configuration values in `wp-config.php`.
    3. Add the following to `wp-config.php`.

```
if (isset($_SERVER['HTTP_CLOUDFRONT_FORWARDED_PROTO']) && $_SERVER['HTTP_CLOUDFRONT_FORWARDED_PROTO'] === 'https') {
    $_SERVER['HTTPS'] = 'on';
}
```

    - Access `{CloudFront domain}/wp-admin/` and complete the WordPress setup.
5. **Install and set up WP Offload Media Lite**
    - Install and set up the WP Offload Media Lite plugin, which uploads images to S3.
    - Configure it to use the server's IAM role instead of access keys.
    - Verify that images are being uploaded to S3.

Level 1 submission

- The URL of the WordPress admin dashboard
- Basic Authentication credentials
- WordPress username and password
- DB root user password

Level2

2-1 Server configuration

Tasks

1. **Install and configure various middleware**
    - Middleware to install
        1. rsyslog
        2. cronie
        3. htop

    * For middleware that supports autostart, enable the autostart setting.
2. Configure the server's time zone
    - Timezone: Asia/Tokyo

3. **Apache configuration (security-related)**
    1. Configure Apache to not return its version.
    2. Disable the default content (welcome page).
    3. Disable access to the `icons` directory.
4. **PHP configuration**
    - Configure PHP to not include its version in the response headers.
    - Set the PHP timezone to Asia/Tokyo.
5. **Logrotate configuration**
    - Set the rotation to weekly and keep 52 generations.
    - Enable compression for httpd logs (also enable `delaycompress`).

6. Apache Configuration(Other)

- In the configuration file created in `/etc/httpd/conf.d/vhosts/`, specify settings to meet the following requirements:
  - Ensure that the client's IP address is output in the access log's IP address field.
  - Exclude ELB health check requests from the access logs.

Level3

3-1 Create a CloudFormation template

Tasks

1. Convert the resources created up to Level 2 into a CloudFormation template
    - **Stack name**: Starts with “{IAM username}-”
    - Add "-cfn" to the end of each resource name created up to Level 2 (e.g., coosy-training-ec2-{IAM username}-cfn).
    - Ensure that attributes other than resource names, such as tags, are the same as those created up to Level 2.
    - Ensure that the following actions are automatically executed when the EC2 instance first starts. * (Using AWS::CloudFormation::Init is recommended)
        1. **Install various middleware, enable autostart, and start them**
            - php
            - php-mysqlnd
            - php-fpm
            - apache
            - mariadb
            - git
            - rsyslog
            - htop
            - cronie

2. **Create Apache configuration files**
    - Create a directory for vhosts configuration (/etc/httpd/conf.d/vhosts)
    - Place a configuration file in /etc/httpd/conf.d/ that includes the setting to load .conf files under /etc/httpd/conf.d/vhosts.
    - Create a config file with the vhosts settings in /etc/httpd/conf.d/
    - Ensure that the client's IP address is output in the access log's IP address field.
    - Exclude ELB health check requests from the access logs.
    - Configure Apache to not return its version.
    - Disable the default content (welcome page).
    - Disable access to the `icons` directory.
3. **PHP configuration**
    - Configure PHP to not include its version in the response headers.
    - Set the PHP timezone to Asia/Tokyo.
4. **Logrotate configuration**

- Set the rotation to weekly and keep 52 generations.
- Enable compression for httpd logs (also enable `delaycompress`).

* It is acceptable to split the template files into multiple files.

Level 3 submission

- CloudFormation template