Infrastructure as Code w/CloudFormation

In this assignment, you will update the CloudFormation template for the application stack to add the following resources:

Private Subnets for RDS

* Create 3 private subnets each in a different availability zone in the same region in the same VPC. in the VPC the CloudFormation template creates.
* Create a private [route table](https://docs.aws.amazon.com/vpc/latest/userguide/VPC_Route_Tables.html). Attach private subnets to this route table.
* Create a private route in the private route table.
* Do not add a route for 0.0.0.0/0.

DB Security Group

* Create an EC2 security group for your RDS instances.
* Add ingress rule to allow TCP traffic on port 3306 for MySQL or 5432 for PostgreSQL.
* The Source of the traffic should be the application security group.
  + Restrict access to the instance from the internet.
* This security group will be referred to as the database security group.

S3 Bucket

* Create a private S3 bucket with the bucket name <random-string>.dev.domain.tld or <random-string>.prod.domain.tld depending on the environment.
* Make sure CloudFormation can delete the bucket even if it is not empty.
  + To delete all objects from the bucket manually using CLI before you delete the bucket you can use the following AWS CLI command that may work for removing all objects from the bucket. aws s3 rm s3://bucket-name --recursive.
* Enable [default encryption for S3 Buckets](https://docs.aws.amazon.com/AmazonS3/latest/dev/bucket-encryption.html).
* Create a lifecycle policy for the bucket to transition objects from STANDARD storage class to STANDARD\_IA storage class after 30 days.

RDS Parameter Group

A DB parameter group acts as a container for engine configuration values that are applied to one or more DB instances. Create a new parameter group to match your database (Postgres or MySQL) and its version. Then RDS DB instance must use the new parameter group and not the default parameter group.

RDS Instance

WARNING: Setting Public accessibility to true will expose your instance to the internet.

Your RDS instance should be created with the following configuration. You may use default value/settings for any property not mentioned below.

| **Property** | **Value** |
| --- | --- |
| Database Engine | MySQL/PostgreSQL |
| DB Instance Class | db.t3.micro |
| Multi-AZ deployment | No |
| DB instance identifier | csye6225 |
| Master username | csye6225 |
| Master password | *pick a strong password* |
| Subnet group | Private subnet for RDS instances |
| Public accessibility | No |
| Database name | csye6225 |

Database security group should be attached to this RDS instance.

User Data

1. EC2 instance should be launched with [user data](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/user-data.html).
2. Database username, password, hostname, and S3 bucket name should be passed to the web application using [user data](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/user-data.html).
3. S3 bucket name must be passed to the application via EC2 user data.

IAM Policy

WebAppS3 the policy will allow EC2 instances to perform S3 buckets. This is required for applications on your EC2 instance to talk to the S3 bucket.

Note: Replace \* with appropriate permissions for the S3 bucket to create security policies.

{

"Version": "2012-10-17",

"Statement": [

{

"Action": [

"s3:\*"

],

"Effect": "Allow",

"Resource": [

"arn:aws:s3:::YOUR\_BUCKET\_NAME",

"arn:aws:s3:::YOUR\_BUCKET\_NAME/\*"

]

}

]

}

IAM Role

Create an IAM role EC2-CSYE6225 for the EC2 service and attach the WebAppS3 policy to it. You will attach this role to your EC2 instance.

Web Application

Web application’s database must be the RDS instance launched by the CloudFormation template when running on the EC2 instance. You can no longer install/use the local database on the EC2 instance.

**Swagger Docs:**

**RESTful API Endpoints To Be Implemented**: <https://app.swaggerhub.com/apis-docs/spring2022-csye6225/app/a03>

About the field data types in swagger docs:

1. If a field has readOnly set to true, the value in this field is populated by your application. Example fields are timestamp and id. readOnly properties are included in responses but not in requests.
2. If a field has writeOnly set to true, the value in this field is provided by the API caller in POST or PUT request but these fields are not part of GET request. An example is the password field. writeOnly properties may be sent in requests but not in responses.
3. multipleOf keyword is used to specify that a number must be the multiple of another number.
4. The minimum and maximum keywords are used to specify the range of possible values.

**API Requirements**

1. All API request/response payloads should be in JSON.
2. No UI should be implemented for the application.
3. As a user, I expect all APIs calls to return with a proper [HTTP status code](https://en.wikipedia.org/wiki/List_of_HTTP_status_codes).
4. Users should be authenticated & authorized where applicable.
5. Users can add a picture to their profile.
   1. The application must support popular image formats such as png, jpg, and jpeg.
   2. Each image must be stored in an S3 bucket. Object metadata must be stored in the database. See AWS docs for available [metadata](https://docs.aws.amazon.com/AmazonS3/latest/dev/UsingMetadata.html#object-metadata).
   3. Users can (hard) delete pictures attached to their profile.
      1. Image must be deleted from the S3 bucket.
   4. Users can also update the profile picture.
      1. The old picture must be deleted from the S3 bucket on update.
6. Users can add/delete/update profile pictures to their profile only. They should not be able to add/delete/update profile pictures to other users' accounts.
7. S3 credentials should not be hardcoded anywhere and the application must be able to access S3 using the IAM role attached to the EC2 instance.

Documentation

AWS

* [cloud-init](https://cloudinit.readthedocs.io/en/latest/)
* [Running commands on your Linux instance at launch](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/user-data.html).
* [User data and shell scripts](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/user-data.html#user-data-shell-scripts)
* [User data and cloud-init directives](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/user-data.html#user-data-cloud-init)

Submission

The assignment will be considered late if commits are made to the **main** branch after the due date.

1. All work for this assignment must be done on the feature branch in your fork and merged to main when you are dev complete.
2. The feature and main branches must be in sync.
3. Submit your code from **all** repositories in this assignment. **Read the instructions carefully to create your zip file.**
   1. Create a folder with the naming convention **firstname\_lastname\_neuid**
   2. In the **firstname\_lastname\_neuid**clone all of your GitHub (organization) repositories with the **git clone**command. It is important that you clone the repos so that your commit history and branches are preserved.
   3. Once you have cloned all of your repositories, you will create a zip of the **firstname\_lastname\_neuid\_a\_##** directory. The zip file should be **firstname\_lastname\_neuid\_a\_##.zip** where **##** is the assignment number.
   4. Now unzip the zip file in some other directory and confirm the content of the zip files.
   5. Upload the Zip to this assignment.
4. You are allowed to resubmit. If you think there may be an issue with the ZIP file, feel free to submit it again. Only the latest will be used grading.

Grading Guidelines

The following guidelines are for information only. They are subject to change at the discretion of the instructor and TA.

10% penalty will be imposed if the application crashes due to unhandled exceptions/errors.

Previous Assignment Objectives

* TAs must verify that students have resolved all open issues from the previous assignment(s).

Git Repository Content Check (10% Penalty)

* Check the repository for any IDE-specific files. IDE configuration files must not be in the repository.
  + Verify their .gitignore configuration.
* Check the repository for build artifacts such as .class, .jar, .war files and build, node\_modules directory. None of these should be checked into the repository.
* Check for dependencies. Dependencies from the Maven repository or npm should not be committed to the git repository.

Pre-requisite for Demo (5% Penalty)

* Delete all VPCs you may have created in your demo AWS account.

Application Crash (10% Penalty)

* The application should not crash or return *500 Internal Server Errors* during the demo.

Validate and Build AMIs (5%)

* Verify that AMI does not have a database installed.
  + It should not have MySQL/PostgreSQL server installed on it. It is OK to have MySQL/PostgreSQL client installed on it.
* Verify this by looking at AMI provisioners.
* Students must build a new AMI for the demo.
* Changes to AMIs are not allowed during the demo.
* AMI repo’s main branch should not have any changes after this assignment’s deadline.

Infrastructure as Code (55%)

* The demo AWS account must not have any infrastructure set up prior to the demo.
* Deleting the CloudFormation stack should delete all of the resources it has created including the S3 bucket.
  + Students can remove objects from the S3 bucket using AWS CLI but not AWS Console.

Networking

* All networking resources should be created using the CloudFormation template as part of the application stack.
* Default VPC should not be used for any resource other than building AMIs.

IAM Policy

* Should be created by CloudFormation template as part of the application stack..
* Ensure that this policy does not exist before the infrastructure is created by the CloudFormation template as part of the application stack..
* Ensure that Resource section is not set to \*. It should have the name of the bucket created by the CloudFormation template as part of the application stack.

IAM Role

* Ensure that role does not exist before the infrastructure is created by the CloudFormation template as part of the application stack.
* The IAM role should be created by the CloudFormation template as part of the application stack.
* The IAM role should allow EC2 service to call S3 service and the above policy should be attached to it.

Security Group for Application Servers

* Ensure that the Security Group for Application Servers does not exist before the infrastructure is created by the CloudFormation template as part of the application stack.
* Verify that it only opens required ports and not all ports.

Security Group for Database Servers

* Ensure that the Security Group for Database Servers does not exist before the infrastructure is created by the CloudFormation template as part of the application stack.
* Verify that it only opens required ports and the ingress source is set to the application security group.
* It should not allow traffic from the internet.

S3 Bucket

* S3 bucket to store attachments is created using the CloudFormation template as part of the application stack.
* S3 bucket should be private.
* S3 bucket should be encrypted.
* S3 bucket has lifecycle policies set up to transition all objects from STANDARD storage class to standard\_IA storage class after 30 days.

RDS Instance

* RDS instance should be created in the database server security group.
* RDS instance should not have a public IP address.
* RDS instance should be launched in the VPC created by the CloudFormation template as part of the application stack.
* RDS instance should use the custom RDS DB parameter group.

EC2 Instance

* EC2 instance should be created using the AMI built for this demo.
* EC2 instance should be created in the application server security group created as part of the CloudFormation template as part of the application stack.
* EC2 instance must have a root volume larger than 8GB.
* EC2 instance should be launched in the VPC created by CloudFormation.
* RDS and S3 bucket information is provided to EC2 instance using userdata.
* IAM role created by the CloudFormation template must be attached to the EC2 instance.

Web Application (40%)

* Students cannot install and configure Git or AWS CLI on the EC2 instance.
* Development tools such as Maven, Gradle, etc. should NOT be installed on the EC2 instance.
* No database should be installed on the EC2 instance.
* Verify data is stored in the RDS instance.
  + Ask students to SSH into the EC2 instance and then connect to the RDS instance using the database client.

Bootstrapping Database

* When the application is started for the first time, it should create all the database objects it needs.
* Students are not allowed to run SQL scripts or queries to bootstrap the database.

Application Configuration

* The web application should configure itself from the env/properties file created by the userdata.
* No edits are allowed to the source code uploaded to the EC2 instance.
* No edits are allowed to the source code when it is built on a student’s laptop. Verify that the code has not been modified and matches the code on their master branch.

Profile Image

* The user should be able to add a profile picture.
* The user should be able to delete a profile picture.
  + Verify that image is deleted from the S3 bucket.
* The user should be able to update their profile picture.
  + This should result in the old picture being deleted from the S3 bucket.
* Multiple users can add the same image to their profile.
  + A separate copy of the image must be stored for each user.
  + If one user deletes the image, it should not impact other users.
* Verify users cannot update/delete other users' profile pictures.