World Skills India - Cloud Computing Test Project for State Hackathon

Jan 2020

aws partner network
Training Partner

ETHNUS
151/17/1, SST Chambers, Second Floor,
36th Cross Road, Jayanagar 5th Block,
Bangalore, Karnataka - 560041

aws academy
Accredited Educator

aws training and certification
Authorized Instructor

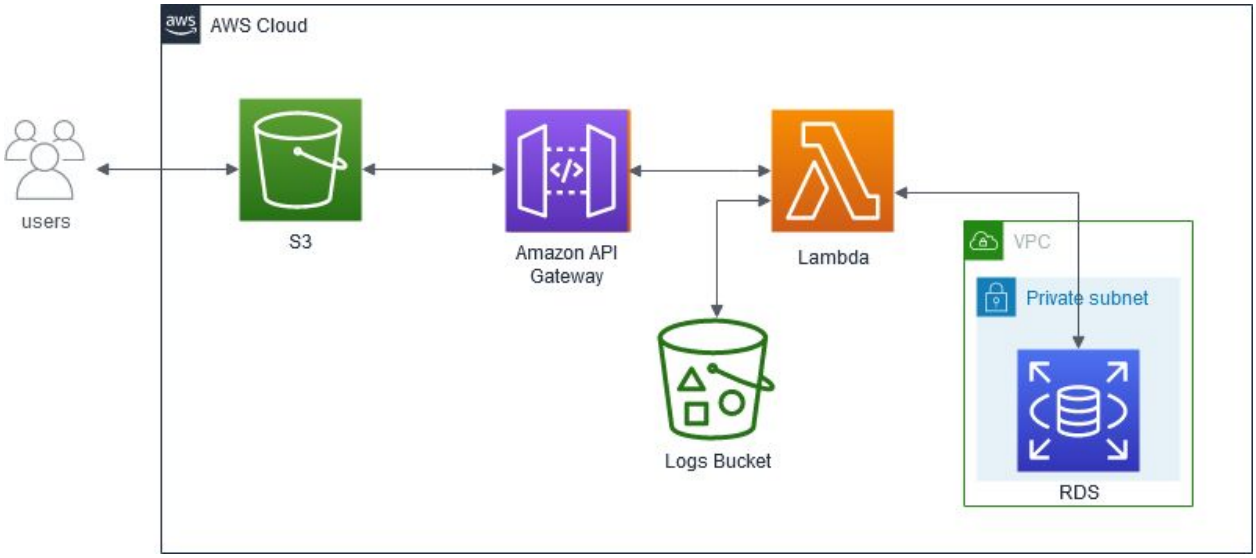
Leveraging Managed Serverless Services

Description: AWS provides a spectrum of managed services that cater to object storage, serverless computing, API management, databases and more.

The requirement is to leverage these services to create a fully managed, serverless solution that can do the following:

1. Host a static website on S3 that simply takes in username and password in the first page.
   The second page will be a container for displaying access logs of the same user in a well defined view.
2. Configure an API using API Gateway that calls a Lambda function which validates the user input.
3. On successful validation, the details are checked from a table in RDS, and if successful, should authenticate the user. (You can use cookies/DynamoDB/ElastiCache etc to manage the session information)
4. A friendly welcome message should be given to the user when the authentication is successful, otherwise an appropriate error should be displayed.
5. Each such login attempt should capture as much user metadata and write it to a log file on S3 in a different bucket.
6. After logging in, the user should be able to access his/her login attempts logs.
7. A logout button should clear the session information and log out the user and redirect to the login page.

Reference Architecture



- A simple static website has to hosted S3 (refer image below)
- Configure API Gateway to connect to a Lambda which in turns performs a simple username and password validation against RDS hosted inside the same VPC in private subnet.
- Do not store the password as plain text. You may use any known hashing algorithms such as SHA.
- Pre populate the user table with usernames and password hashes

  Or

  Create a lambda function to populate usernames and password hashes into the user table
- All successful and failed attempts should be logged to S3 in a different bucket than the bucket used for the static website.

Tips:

- Make extensive use of javascript libraries like bootstrap.js/jquery etc to ease your front end development.
- Decouple the services and chain the flow as needed.
- Use secure communication such as HTTPS/SSL/TLS for all data in transit.
- Use any language for lambda functions with which you are comfortable.
- Use AWS architecting best practices to build the solution.
- Consider the security best practices in all stages.

Bonus 1: Create a log viewer page that can consolidate the logs from S3.
Bonus 2: A master view (with admin privileges) that shows all user access logs optionally with masked email addresses.

Sample Login Page:

Sample Table DDL:

```
CREATE TABLE users (
      UserId int NOT NULL AUTO_INCREMENT,
      Password varchar(255) NOT NULL,
      PRIMARY KEY (UserId)
);
```

Sample Lambda script for user authentication (node.js):

```
let AWS = require('aws-sdk');
const rds = new AWS.RDS(...);

exports.handler = function (event, context, callback) {
    let passwordHash = // hash the event.password
    let successfullyLoggedIn = false;
      let inserts = [event.email, passwordHash];
      rds.query({
            instanceIdentifier: 'authDatabase',
            query: 'SELECT * FROM users WHERE Email = ? AND Password = ?',
            inserts: inserts
      }, function (error, results, connection) {
            if (error) {
                  throw error;
            } else {
                  successfullyLoggedIn = results.length > 0;
            }

            connection.end();
            callback(null, successfullyLoggedIn);
      });
}
```