**Homework #1 Web Application**

**Serverless Seekers**

**-Dhruv Khut**

**-Pratik Kamanahalli Mallikarjuna**

**-Rahul Dhingra**

**-Nitya Reddy Yerram**

This document provides a comprehensive guide on setting up, deploying, and cleaning up an AWS-based web application that utilizes AWS Directory Service for authentication and role-based access control. The application was deployed using AWS Elastic Beanstalk and integrated with AWS AD for secure authentication and authorization.

**Setup Details**

**1. AWS Directory Service Setup**

* Signed up for AWS Directory Service Free Trial.
* Created an AWS Managed Microsoft AD instance in a specified VPC.
* Configured users and groups to manage role-based access.
* Enabled security groups to allow LDAP/LDAPS traffic.

**2. Elastic Beanstalk Deployment**

* Created an Elastic Beanstalk application for hosting the web app.
* Configured the environment to use a custom VPC and subnets.
* Set environment variables for LDAP authentication.
* Ensured security group settings allowed communication with AWS Directory Service.

**Implementation**

**1. Authentication Setup (LDAP)**

**Code Snippet:**

*const LDAP\_OPTIONS = {*

*server: {*

*url: process.env.LDAP\_URL || 'ldap://your-directory.example.com:389',*

*bindDN: process.env.LDAP\_BIND\_DN || 'CN=readonly,OU=Users,DC=example,DC=com',*

*bindCredentials: process.env.LDAP\_BIND\_PASSWORD || 'password',*

*searchBase: 'OU=Users,DC=example,DC=com',*

*searchFilter: '(sAMAccountName={{username}})'*

*}*

*};*

**2. Role-Based Access Control**

**Middleware to Check User Roles:**

*function checkRole(requiredGroup) {*

*return (req, res, next) => {*

*const groups = Array.isArray(req.user.memberOf) ? req.user.memberOf : [req.user.memberOf];*

*const hasRole = groups.some(group => group.includes(requiredGroup));*

*hasRole ? next() : res.status(403).send('Access Denied');*

*};*

*}*

**3. Screenshots**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A computer screen with a white screen

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

A screenshot of a computer

AI-generated content may be incorrect.

**Deployment Steps**

1. **Upload Code to Elastic Beanstalk**
   * Used AWS CLI: eb init and eb deploy commands.
2. **Verify Application Logs**
   * Checked application and health logs in AWS Elastic Beanstalk.
3. **Load Balancer Health Check**
   * Configured health checks for proper request routing.
4. **Environment Variables Setup**
   * Defined LDAP-related credentials and configurations securely.

**Scalability (Extra Credit)**

* Configured AWS Elastic Beanstalk to auto-scale instances based on traffic load.
* Integrated Amazon CloudWatch for real-time monitoring and alerts.
* Performed a load test using Loader.io to simulate user traffic.

**Resource Cleanup**

To ensure no unnecessary AWS costs, the following resources were deleted:

1. **Delete Elastic Beanstalk Environment**

aws elasticbeanstalk terminate-environment --environment-name aws-ad-webapp-env

1. **Delete Elastic Beanstalk Application**

aws elasticbeanstalk delete-application --application-name my-app

1. **Delete RDS Database (if created)**

aws rds delete-db-instance --db-instance-identifier mydatabase --skip-final-snapshot

1. **Delete S3 Buckets**

aws s3 rb s3://my-bucket-name --force

1. **Delete Security Groups & IAM Roles**

aws ec2 delete-security-group --group-id sg-12345678

aws iam delete-role --role-name aws-elasticbeanstalk-ec2-role

1. **Delete Load Balancers (if left behind)**

aws elb delete-load-balancer --load-balancer-name my-load-balancer

1. **Delete VPC (If created for this project)**

aws ec2 delete-vpc --vpc-id vpc-12345678

1. **Final Billing Check**
   * Verified AWS **Billing Dashboard** to ensure all resources were removed.

**Link to Code Repository**

* <https://github.com/PratikKM2002/Web-Application>