Project Documentation

Project Title:

Deploying Web Application on EC2

Project Overview:

Deploy a web application on an Amazon EC2 instance using a chosen programming language and web framework. This project includes setting up the EC2 instance, installing necessary software, configuring security groups, and deploying the web application code. Additionally, it involves using Elastic IP for a static IP address and Elastic Load Balancer for load balancing to ensure high availability and reliability.

Project Team:

1. **Vivaan Johri(TL) -** Oversees the project, sets up the EC2 instance, and manages overall deployment. Ensures that all project objectives are met and coordinates the team's activities.
2. **Anisha Shankar Gore -** Responsible for implementing features of the web application, ensuring the application runs smoothly, and contributing to the deployment process.
3. **G Srimanjunath -** Focuses on testing the application, maintaining code quality, and ensuring the application is robust and free of critical bugs.

Project Technologies:

- **AWS EC2**: The primary cloud service used to host and run web applications.
- **Git and GitHub**: Version control system and repository hosting service for collaborative development and code management.
- **Shell Scripting**: Used for automating setup, installation, and deployment tasks on the EC2 instance.

- **Web Framework:** The framework used to develop the web application, depending on the chosen programming language. (Amazon Linux 2023 AMI)

- **Elastic IP and Elastic Load Balancer**: Services used to provide a static IP address and distribute incoming traffic across multiple instances, respectively.

Project Setup:

Create and Configure EC2 Instance:

- Launch an EC2 instance (e.g., t3.micro) from the AWS Management Console.
- Allocate and associate an Elastic IP to the instance for a persistent public IP address.
- Configure security groups to allow necessary traffic (e.g., SSH for remote access, HTTP/HTTPS for web traffic).

Install Required Software:

- Connect to the EC2 instance via SSH using a terminal.
- Update the system packages and install necessary software, such as Git for version control and ----- for the web framework.

Set Up SSH Access:

- Create .ssh directories and authorized keys files for Vivaan and Manju.
- Add their public SSH keys to the authorized keys files to allow secure SSH access.

Clone Repository and Set Up Branches:

- Clone the project repository from GitHub to the EC2 instance.
- Create and switch to individual branches for each team member to work on their parts of the project independently.

Deploy the Application:

- Install all dependencies required by the web application.
- Run the application on the EC2 instance and test it to ensure it is functioning as expected.

Project Resources/Reference:

- **Human Resources**: Anisha Shankar Gore, Vivaan Johri (TL), G Srimanjunath
- **Materials**: AWS EC2 instance, Elastic IP, Security Groups
- **Tools**: GitHub for version control, Shell for scripting, Text Editors (e.g., nano, vim) for editing configuration files
- **Links**:
  - AWS EC2 Documentation
  - GitHub Documentation
  - https://www.free-css.com/assets/files/free-css-templates/download/page296/mediplus-lite.zip
  - https://www.youtube.com/watch?v=gWVIIU1ev0Y

    - https://www.youtube.com/watch?v=Islmm-LMu38
    - https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/elastic-ip-addresses-eip.html
    - https://docs.aws.amazon.com/elasticloadbalancing/

Project Risks:

- *Security Risks*

There is a potential for unauthorized access or data breaches if the EC2 instance and application are not properly secured. To mitigate this risk:

- **Configure Security Groups Properly**: Only allow necessary traffic (e.g., SSH, HTTP/HTTPS).
- **Manage SSH Keys Securely**: Use strong SSH keys, restrict access to authorized users only, and regularly update keys.
- **Regular Updates and Patching**: Ensure the operating system and all installed software are up to date with the latest security patches.

- *Deployment Issues:*

Errors during deployment could cause the application to fail or malfunction. To mitigate this risk:

- **Thorough Testing**: Conduct comprehensive testing in a staging environment before deploying changes to production.
- **Version Control**: Use GitHub for version control to track changes, collaborate effectively, and revert to previous stable versions if necessary.
- **Automated Deployment Scripts**: Use shell scripts to automate the deployment process, reducing the chance of human error and ensuring consistency