Yash Mutatkar **Nov 12 2024**

Documentation: Configuring AWS Elastic Beanstalk for Application Deployment

Project: Deploying Application using AWS Elastic Beanstalk

Overview

In this project, I set up and configured AWS Elastic Beanstalk to deploy a web application. The primary goal was to leverage AWS Elastic Beanstalk's features, such as simplified environment setup, automatic scaling, and easy management, to streamline deployment and reduce infrastructure management overhead.

Step-by-Step Guide

---

1. Setting Up the AWS Elastic Beanstalk Environment

- **AWS Elastic Beanstalk Console**:
    - I accessed the AWS Management Console, navigated to **Elastic Beanstalk**, and started setting up the environment for the deployment.
    - Selected **Create a new environment** to initiate the setup process.
- **Choosing the Application Platform**:
    - Selected the appropriate platform for my application, such as **Node.js**, **Python**, **Java**, or any other language based on the application stack.
    - Elastic Beanstalk automatically configures the server environment based on the platform chosen.

---

2. Application Setup and Upload

- **Creating an Application**:
    - I created a new application in the Elastic Beanstalk console, giving it a unique name for easy identification.
- **Uploading Application Code**:
    - Uploaded the application code as a `.zip` file containing the app’s source code and dependencies.
    - Elastic Beanstalk can automatically detect dependencies for supported platforms and install them as part of the environment configuration.

---

3. Configuring Environment Settings

- **Setting Up Environment Variables**:

    - Defined necessary environment variables in the Elastic Beanstalk configuration panel. These variables were crucial for storing sensitive information like API keys, database URLs, and other configuration data.
- **Specifying Instance Type and Scaling**:
    - Configured the instance type (e.g., t2.micro, t2.medium) based on application requirements.
    - Enabled **Auto-Scaling** to allow Elastic Beanstalk to automatically manage the number of instances running based on traffic.
- **Load Balancer and Network Configuration**:
    - Configured the load balancer to manage incoming traffic and distribute it across instances.
    - Specified VPC, subnets, and security groups to control network access to my application.

---

4. Deploying the Application

- **Deploying the Application Version**:
    - Elastic Beanstalk allows multiple application versions to be created and stored. I selected the version I wanted to deploy and launched it.
    - The environment status changed to **Running**, indicating a successful deployment.

---

5. Monitoring and Managing the Environment

- **Monitoring Tools**:
    - Used the Elastic Beanstalk dashboard to monitor environment health, logs, and metrics.
    - Elastic Beanstalk integrates with **CloudWatch** for detailed monitoring, providing insights into CPU utilization, memory usage, and request counts.
- **Rolling Updates**:
    - Configured rolling updates to ensure that the deployment of new versions doesn’t disrupt active user sessions.

---

6. Setting Up CI/CD Pipeline (Optional)

- **Continuous Integration/Continuous Deployment (CI/CD)**:
    - Configured a CI/CD pipeline (optional but recommended) using **AWS CodePipeline** and **AWS CodeBuild** to automate future deployments.
    - This setup enabled automatic deployment of new code versions to Elastic Beanstalk whenever changes were pushed to the repository.

---

7. Testing and Finalization

- **Testing the Deployment**:
    - Once deployed, I tested the application by accessing the generated URL provided by Elastic Beanstalk.
    - Ensured that the application was responsive and that all configurations, like database connections and API integrations, were functional.
- **Finalizing the Deployment**:
    - After confirming everything worked as expected, I finalized the deployment and documented all configurations for future reference.

Conclusion

Completing this task, I gained hands-on experience with AWS Elastic Beanstalk's deployment workflow, configuration management, and environment customization. This deployment setup helps to maintain application stability while simplifying infrastructure management for scaling and monitoring.