**Web Services Report**

**Solution Architecture**

In this solution, the goal is to build a system where users can input data (such as age, job, marital status, education, etc.) through a web interface. Based on the input, the system will predict whether the client will subscribe to a term deposit. The solution involves a trained machine learning model (e.g., SVM or Logistic Regression) that provides this prediction.

Diagram -[Solution Architecture Diagram](https://drive.google.com/drive/u/0/folders/1DNd04I1KF25pBtN8kobAQcNq9LIeqP3S)

**Flow of Requests:**

**1. User Input:**

* The user enters data (age, job, marital status, etc.) using dropdowns and submits it.

**2. Request to Application:**

* The form data is sent via a POST request to the Flask web application running in the backend.
* Flask receives the input and preprocesses it (e.g., handling missing values, encoding categorical variables, and scaling numerical values).

**3. Model Prediction:**

* The preprocessed data is passed to the trained machine learning model.
* The model predicts whether the client will subscribe to the term deposit.
* The prediction (e.g., "Yes" or "No") is returned.

**4. Return Prediction to User:**

* The prediction result is sent back as a response to the user’s web browser.

**Data Flow:**

* **Input Data:** User-provided values (age, job, marital status, etc.) are sent from the web application.
* **Preprocessed Data:** After encoding and scaling, the data is passed to the trained model.
* **Prediction Result:** The model returns a prediction, which is then displayed in the web application.

**Deployment Architecture**

The solution is deployed on AWS ECS (Elastic Container Service) to manage and scale the web application efficiently. Several AWS services are integrated to ensure the application's reliability and performance.

Diagram -[Deployment Architecture Diagram](https://drive.google.com/drive/u/0/folders/1DNd04I1KF25pBtN8kobAQcNq9LIeqP3S)

**Flow of Requests:**

**1. User Access:**

* The user accesses the deployed web application through a public URL.
* The application is hosted on AWS ECS and exposed via an Elastic Load Balancer (ELB) to handle traffic distribution.

**2. ECS Container:**

* The user’s request is routed to the ECS container running the web application.
* ECS automatically manages the container’s scaling and availability.

**3. Model Inference:**

* The application accesses the pre-trained model, which is embedded within the container.
* The model processes the user input and makes a prediction.

**4. Response Back to User:**

* The prediction result is returned through the Elastic Load Balancer (ELB) and displayed in the user’s browser.

**Data Flow:**

* **Input Data:** User-provided values (age, job, marital status, etc.) are sent from the web application to the web application.
* **Preprocessed Data:** After encoding and scaling, the data is passed to the trained model.
* **Prediction Result:** The model returns a prediction, which is then displayed in the web application.

**CI-CD Pipeline Process**

The CI/CD (Continuous Integration and Continuous Deployment) pipeline automates the process of building, testing, and deploying the web application to AWS ECS. Below are the steps in the pipeline.

Diagram - [CI-CD Pipeline Diagram](https://drive.google.com/drive/u/0/folders/1DNd04I1KF25pBtN8kobAQcNq9LIeqP3S)

**Process:**

1. **Code Commit (Source Control - GitHub):**

* Developers push the latest code changes to a version control repository (GitHub).

1. **Continuous Integration (AWS CodeBuild):**

* The AWS CodeBuild workflow is triggered after code changes are detected.
* It pulls the latest code from GitHub.
* It runs automated tests, including unit tests and integration tests.
* After tests, it builds a Docker image of the application.

1. **Docker Image Push (AWS ECR):**

* The Docker image is tagged and pushed to AWS ECR (Elastic Container Registry) for storage.

1. **Continuous Deployment (AWS ECS):**

* AWS ECS (Elastic Container Service) fetches the latest Docker image from ECR.
* ECS updates the running container with the new version of the app.
* The application is automatically restarted, ensuring zero downtime.

1. **Deployment Validation:**

* Health checks are performed to ensure the updated application is functioning correctly.
* If the deployment fails, an automatic rollback is triggered to restore the previous stable version.