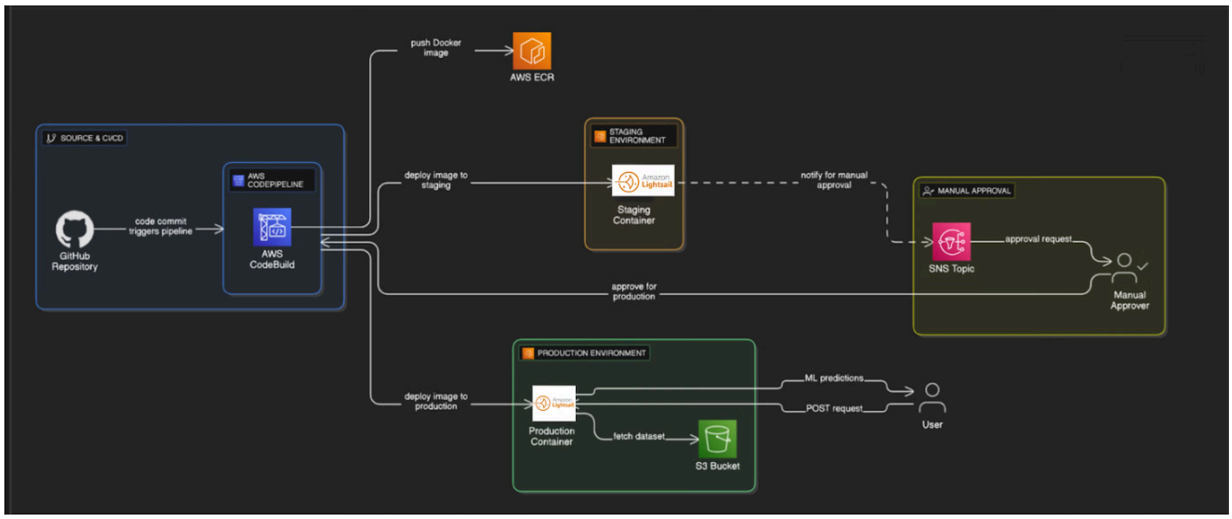
(/user/dashboard)

CI/CD Deployment of Gaussian Process Time Series Models

Architecture Diagram



Project Details Notes Forum Download Codes

In this MLOps project, you will learn to build an automated, end-to-end MLOps pipeline to deploy a Gaussian Process Time Series forecasting model on AWS. Using Terraform for infrastructure provisioning,

the solution implements CI/CD pipelines that support safe, reproducible staging and production environments.

What will you learn?

- Learn how to build a workflow for automating model deployments on AWS
- Learn how to use Terraform to provision AWS infrastructure automatically
- Understanding the process of containerizing ML applications with Docker
- How to implement CI/CD using AWS CodePipeline for testing and deployment?
- Understand the deployment process on Amazon Lightsail
- Learn how to create separate staging and production environments for controlled releases
- How to integrate approvals through Amazon SNS for promoting tested builds to prod?
- Learn how to automate image deployment through task definitions and ECS service updates
- Understanding the integration of AWS S3 as the backend for input data in ML applications

- How to validate a deployed ML Application by making a POST Request?
- Gain practical experience in building a production-ready MLOps pipeline

Project Description

Business Overview
In today's fast-paced, data-driven environment, the deployment of machine learning models efficiently and reliably is crucial for informed business decision-making. Time series forecasting plays a crucial role in areas such as demand prediction, financial analysis, and operational planning.
This project demonstrates how to operationalize a Gaussian Process Time Series Model on AWS. The practice of deploying the model in a staging environment first, organizations can validate its performance before moving to production. All infrastructure and services in this project were provisioned and managed using Terraform alongside UI, to ensure a structured and automated setup for both staging and production environments.
We recommend having a basic understanding of the AWS MLOps Project for Gaussian Process Time Series Modeling (../../../project-use-case/gaussian-model-time-series-mlops-aws) before starting this project.

Aim:
This project aims to deploy a Gaussian Process Time Series model on AWS, which covers the full lifecycle from model creation and containerization to deployment and monitoring with automated CI/CD pipelines. The key to using AWS services such as CodePipeline, CodeBuild, ECR, EC2, and Lightsail is to automate the workflow, whereas Terraform provisions the entire infrastructure automatically.

Prerequisite Project:
1. Build Time Series Models for Gaussian Processes in Python (../../../project-use-case/gaussian-model-time-series-python)
2. AWS MLOps Project for Gaussian Process Time Series Modeling (../../../project-use-case/gaussian-model-time-series-mlops-aws)

Prerequisites:
It is recommended to have a basic knowledge of the following:
1. Python
2. Flask
3. Basic understanding of deployment
4. Basics of AWS

5. Terraform

Data Description

(/user/dashboard)

CI/CD Deployment of Gaussian Process Time Series Models

The dataset is “Call-centres” data. This data is at the month level, wherein the calls are segregated at the domain level as the call centre operates for various domains. There are also external regressors like no of channels and no of phone lines, which essentially indicate the traffic prediction of the in-house analyst and the resources available. The total number of rows is 132, and the number of columns is 8:- Month, healthcare, telecom, banking, technology, insurance, no of phone lines, and no of channels.

Tech Stack

➔ **Language:** Python

➔ **Libraries:** Flask, numpy, pandas, boto3, seaborn, scipy, matplotlib

➔ **Services:** AWS ECR, AWS CodePipeline, AWS CodeBuild, AWS Lightsail, AWS SNS, Docker

➔ **Tool:** Terraform

Note:

- We recommend monitoring AWS Services and deleting those that are not in use after project execution, whether they were created using the UI or Terraform.
- AWS services used in this project may incur charges. It is essential to review the official AWS documentation and check Cost Breakdown documentation to understand the pricing structure and potential costs associated with different resources and usage patterns.

Approach:

In this project, we build an end-to-end MLOps pipeline to automate the deployment and management of a containerized Gaussian Process Time Series model on AWS Lightsail.

1. Infrastructure Provisioning

- All AWS resources, including CodePipeline, CodeBuild, ECR, Lightsail containers, and S3, were provisioned automatically using Terraform.
- This ensured a fully automated and consistent setup across both staging and production environments.

2. Source Control and CI/CD Triggering

- The application source code, configuration files, and Terraform scripts were stored in a GitHub repository.
- Each new commit to the repository automatically triggered the AWS CodePipeline, which packaged the source code and forwarded it to the build stage..

3. Staging Environment Deployment

- AWS CodeBuild (Staging) built the Docker image for the ML application and pushed it to the Amazon ECR staging repository.
- Once deployed, Amazon SNS sent a manual approval request to validate the application’s behavior and performance before promotion to production.
- Amazon Lightsail (Staging) deployed the Docker image into a staging container service, where the application was tested in a staging environment.

(/user/dashboard)

CI/CD Deployment of Gaussian Process Time Series Models

4. Production Environment Deployment

- After manual approval by the reviewer, AWS CodeBuild (Production) reused the approved Docker image from the staging environment and pushed it to the production ECR repository.
- Lightsail (Production) pulled this tested image, deployed it into a production container service, and made the Flask-based API publicly accessible.

5. User Request and Model Inference Flow

- When a user sends a POST request to the production API endpoint, the Flask application hosted on the Lightsail container retrieves the data from Amazon S3.
- The Gaussian Process Time Series model processes this data and generates predictions, which are then returned to the user through the API in real time.