

MLOps Architecture Explorer

An interactive guide to the California Housing Price Prediction Pipeline.

[Summary](#)[Architecture](#)[Workflow](#)[Deployment](#)[Contributors](#)

Project Summary

This project implements a complete MLOps workflow for the California Housing dataset, enabling automated, reproducible, and production-ready model deployment. The pipeline spans from dataset preparation to monitoring in production, ensuring that the deployed model consistently delivers high-quality predictions.

1. Data Versioning and Preprocessing

The California Housing dataset was loaded and preprocessed to handle missing values, normalize features, and prepare target variables. A clean, modular directory structure was maintained for data, source code, and models, with dataset versioning managed using DVC to ensure reproducibility.

2. Model Development and Experiment Tracking

Two regression models, Linear Regression and Decision Tree Regressor, were trained. All experiments were tracked using MLflow, recording parameters, metrics, and artifacts. The best model, evaluated on the highest R^2 score, was registered in the MLflow Model Registry for deployment.

3. API Development and Containerization

A FastAPI-based prediction API was developed to accept JSON input and return predicted house values. The entire service was packaged into a Docker container for consistent deployment across any environment.

4. CI/CD Automation with GitHub Actions

A GitHub Actions workflow was implemented to automate linting, testing, building Docker images, and pushing them to Docker Hub. This ensures

zero manual intervention from code commit to service deployment.

5. Logging and Monitoring

Structured logging was implemented for all incoming prediction requests and their outputs, stored for auditing. An optional `/metrics` endpoint was added to expose runtime performance statistics, supporting integration with monitoring tools like Prometheus and Grafana.

Outcome

This project delivers an industry-standard MLOps pipeline that automates the entire machine learning lifecycle. The approach ensures reproducibility, scalability, and maintainability while selecting and deploying the best regression model based on R^2 score, making it ready for real-world production use.

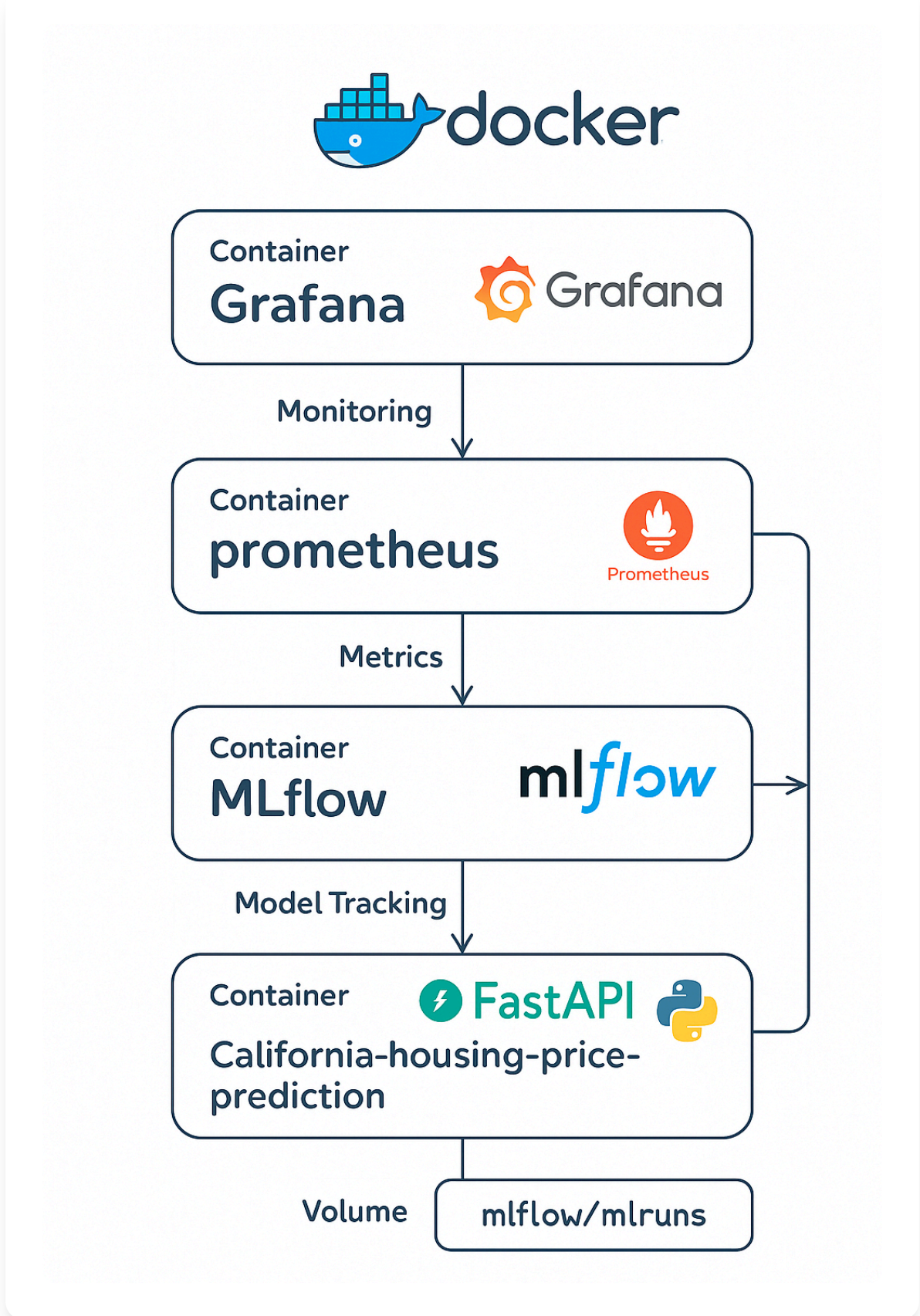


[View on GitHub](#)

[MLOps Architecture Explorer Summary](#)


System Architecture

Below is a static overview of the system. For an interactive experience, click on any component in the diagram further down to reveal its details.



Interactive Diagram

Click on a component to learn more.

 **Docker**
Foundation **Grafana**
Visualization **Prometheus**
Metrics **MLflow**
Model Tracking **FastAPI**
Prediction Service **Persistent Volume**
Data Storage

End-to-End Workflow

1. Model Training

A training script logs parameters, metrics, and artifacts to MLflow, which stores the data in the persistent volume.

2. Model Deployment

FastAPI loads the latest registered model from MLflow for inference and exposes prediction APIs.

3. Monitoring & Visualization

Prometheus scrapes metrics from FastAPI, and Grafana displays this data in real-time dashboards.

Deployment Guide

Pre-requisites:

Ensure that both **Docker** and **Docker Compose** are installed on your system.

Run the Stack:

To start the entire application stack, run the following command in your terminal:

```
docker-compose up -d
```

Access Services:

- **MLflow UI:** <http://103.120.179.184:5000>
- **FastAPI Docs:** <http://103.120.179.184:8000/docs>
- **Prometheus UI:** <http://103.120.179.184:9090>
- **Grafana UI:** <http://103.120.179.184:3000>

Contributors

Atul Bhatia

2023ac05893@wilp.bits-pilani.ac.in

Girish Satyam

2023ad05061@wilp.bits-pilani.ac.in

Vrushab S

2023ad05052@wilp.bits-pilani.ac.in

Chandrashekhar Kumar

2023ac05042@wilp.bits-pilani.ac.in