# MLOps Project: Cat vs Dog Classifier

## Table of Contents

1. [Introduction](#introduction)
2. [Project Structure](#project-structure)
3. [Environment Setup](#environment-setup)
4. [Deployment Details](#deployment-details)
5. [Key Features](#key-features)
6. [API Endpoints](#api-endpoints)
7. [Ingress Configuration](#ingress-configuration)
8. [Conclusion](#conclusion)

## Introduction

This project demonstrates an end-to-end MLOps implementation with three main components: machine learning model development, backend API service, and frontend web application.

## Project Structure

mlops\_project/  
├── ml\_part/ # Machine learning model development  
│ ├── config.py # Configuration parameters  
│ ├── data\_preprocessing.py # Data download and preprocessing  
│ ├── model.py # Model definition and training  
│ ├── checkpoints/ # Saved model files  
│ │ └── model.h5 # Trained model weights  
│ ├── data/ # Dataset directory  
│ │ ├── train/ # Training images  
│ │ │ ├── cats/ # Cat training images  
│ │ │ └── dogs/ # Dog training images  
│ │ └── test/ # Test images  
│ │ ├── cats/ # Cat test images  
│ │ └── dogs/ # Dog test images  
│ ├── plots/ # Training visualization plots  
│ └── README.md # ML documentation  
│  
├── backend/ # API services for model serving  
│ ├── main.py # FastAPI application with endpoints  
│ ├── requirements.txt # Backend dependencies  
│ └── README.md # Backend documentation  
│  
├── frontend/ # User interface  
│ ├── src/ # React source code  
│ │ ├── components/ # React components  
│ │ │ ├── Classifier.tsx # Image classification component  
│ │ │ ├── Performance.tsx # Model performance dashboard  
│ │ │ └── Performance.css # Dashboard styles  
│ │ ├── App.tsx # Main application component  
│ │ ├── App.css # Application styles  
│ │ ├── index.tsx # React entry point  
│ │ └── index.css # Global styles  
│ ├── public/ # Static public assets  
│ ├── package.json # Frontend dependencies  
│ ├── tsconfig.json # TypeScript configuration  
│ └── README.md # Frontend documentation  
│  
├── activate\_env.sh # Script to activate virtual environment  
├── requirements.txt # Core Python dependencies  
└── README.md # Main project documentation

## Environment Setup

This project uses a Python virtual environment located at ~/mlops\_env to manage dependencies for both the ML and backend components.

### Activating the Environment

To activate the virtual environment, run:

source activate\_env.sh

### Dependencies

The main dependencies installed in the environment are: - tensorflow - numpy - matplotlib - requests - scikit-learn - pillow - pillow-avif-plugin - fastapi - uvicorn

## Deployment Details

### Ansible Playbook

The Ansible playbook automates the deployment of the Cat vs Dog Classifier to Kubernetes. Key tasks include: - Checking and starting Minikube - Pulling and loading Docker images - Applying Kubernetes deployments and services - Configuring monitoring components with Prometheus and Grafana

### Kubernetes Configurations

* **Backend Deployment**: Exposes the API on port 8000
* **Frontend Deployment**: Exposes the web application on port 80
* **Prometheus and Grafana**: Configured for monitoring and visualization

## Key Features

1. **Machine Learning**
   * CNN-based cat vs dog image classifier
   * Data preprocessing and augmentation
   * Model training and evaluation
2. **Backend API**
   * Image upload and classification endpoint
   * Model performance tracking
   * User feedback collection
3. **Frontend Application**
   * Image upload and preview
   * Classification results display
   * Performance dashboard with metrics
   * User feedback submission

## API Endpoints

* POST /predict/ - Upload and classify an image
* GET /performance/ - Get model performance metrics
* POST /feedback/ - Provide feedback on predictions

## Ingress Configuration

The ingress setup allows external access to the frontend and backend services: - **Frontend**: Accessible at catvsdog.example.com and catvsdogclasifier.com - **Backend**: Accessible via /backend path on the same hosts

## Conclusion

This project successfully demonstrates the integration of machine learning, backend services, and frontend applications in a Kubernetes environment, with robust monitoring and deployment automation. Future improvements could include scaling the model to handle more classes and optimizing the deployment for production environments.