DCONAP – Data Center Outage Notification And Prediction

# Comprehensive Docker Documentation for Full-Stack Data Platform

This document provides an in-depth explanation of the Docker configuration used for a full-stack data platform, which includes a variety of services essential for data engineering, orchestration, streaming, monitoring, and visualization. It includes detailed documentation of the Dockerfile, docker-compose.yml, project structure, and supporting resources.

## Project Structure

project-root/

├── docker/

│ ├── requirements.txt # Python dependencies for the dev container

│ ├── entrypoint.sh # Entry script for the dev container

│ ├── docker-compose.yml # Compose configuration file

│ └── Dockerfile

└── baseline/ # Source code mounted into the dev container

# Dockerfile Documentation

The Dockerfile defines a custom development container that is suitable for data science and data engineering. It includes Python, JupyterLab, Hadoop, and other essential tools.

### Base Image

* FROM python:3.9: A stable Python version that is compatible with most libraries and tools used in data engineering.

### System Dependencies

* RUN apt-get install:
  + adduser, tini: For secure user management and signal forwarding.
  + default-jre: Required for Hadoop (Java-based tool).
  + jq, curl: Utilities for API interaction and file downloading.
  + apt-get clean, rm -rf /var/lib/apt/lists/\*: Reduce image size.

### Python Dependencies

* COPY docker/requirements.txt /tmp/: Stages Python package requirements.
* RUN pip install:
  + Installs Python packages required for development and data science.
  + Upgrades pip, setuptools, and wheel for compatibility.

### User and Group Configuration

* ARG UID**,** GID: Allow configuration of user and group IDs at build time.
* RUN addgroup**,** adduser: Create a non-root user (jupyteruser) for better security.
* USER jupyteruser: Prevents the container from running as root.

### Working Directory

* WORKDIR /home/jupyteruser: Default working directory for the user.

### Code and Ownership

* COPY --chown=jupyteruser:jupyteruser baseline/ /home/jupyteruser/: Copies application source code while assigning proper ownership.

### Ports and Volumes

* EXPOSE 8888: JupyterLab's default port.
* VOLUME $JUPYTER\_MOUNT\_PATH: Supports volume mounting from the host.

### Entrypoint and Execution

* COPY entrypoint.sh, RUN chmod +x: Add and make the entry script executable.
* ENTRYPOINT ["/entrypoint.sh"]: Specifies the start-up script.
* CMD ["jupyter-lab"...]: Launches JupyterLab by default.

### Hadoop Setup

* ARG HADOOP\_VERSION**,** HADOOP\_URL: Set during build for flexibility.
* RUN curl -L: Downloads and installs Hadoop.
* ENV PATH **and** .bashrc: Ensures Hadoop CLI is accessible.

# Docker Compose (docker-compose.yml) Documentation

Defines and orchestrates multiple services essential for the data platform.

### Compose Version

* version: "3.8": Supports advanced features like profiles and healthchecks.

### Services Overview

#### 1. ****PostgreSQL (****postgres****)****

* Relational database for structured data.
* Uses postgres:14 image.
* Mounts volume postgres\_data.
* Exposes port 5432.

#### 2. ****Redis (****redis****)****

* In-memory data store and cache.
* Used for fast key-value access.
* Image: redis:6.2.

#### 3. ****GitLab (****gitlab****)****

* DevOps lifecycle tool for version control and CI/CD.
* Useful for collaborative development.

#### 4. ****Apache Spark (****spark-master****,**** spark-worker****)****

* Distributed computation engine for batch processing.
* Provides master-worker architecture for parallel processing.

#### 5. ****Apache Airflow (****airflow****)****

* DAG-based workflow orchestrator.
* Schedules ETL, data pipelines.
* Uses apache/airflow image with webserver and scheduler.

#### 6. ****Grafana (****grafana****)****

* Interactive dashboards for visual monitoring.
* Data sources include Prometheus, PostgreSQL.

#### 7. ****Apache Kafka (****kafka****), Zookeeper (****zookeeper****)****

* Kafka: Distributed event streaming platform.
* Zookeeper: Kafka dependency for cluster coordination.

#### 8. ****Apache Flink (****flink-jobmanager****,**** flink-taskmanager****)****

* Real-time stream processing framework.
* Works alongside Kafka for real-time analytics.

#### 9. ****Apache Superset (****superset****)****

* Modern data exploration and visualization platform.

#### 10. ****Prometheus (****prometheus****)****

* Metrics collection and monitoring.
* Scrapes data from various services.

#### 11. ****Development Container (****dev****)****

* Based on the custom Dockerfile.
* Includes Hadoop, Spark, Python tools, and JupyterLab.
* Mounts local baseline/ code into container.
* Launches JupyterLab.

## Profiles

Profiles in Docker Compose help modularize the deployment:

* base: Essential infrastructure (Postgres, Redis).
* realtime: Kafka, Flink, Zookeeper for real-time streaming.
* batch: Airflow for batch ETL workflows.
* web: Superset and Redis for data viz and caching.
* monitoring: Prometheus and Grafana for system observability.
* spark: Spark master and worker for distributed computing.
* dev: JupyterLab-based dev environment.

**Purpose of Profiles:**

* Enable selective deployment.
* docker compose --profile dev up launches only the development environment.
* Avoid unnecessary resource usage during development/testing.

## Volumes

Used for data persistence.

* postgres\_data: PostgreSQL persistent storage.
* grafana\_data: Grafana configuration and dashboards.

## Network

* datacenter\_network: Shared network bridge for all containers to interact seamlessly.

## Summary

This Docker setup allows:

* Isolated, reproducible development using JupyterLab.
* Batch and real-time processing via Airflow, Spark, Kafka, Flink.
* Rich monitoring with Prometheus and Grafana.
* DevOps and visualization via GitLab and Superset.
* Modular architecture using Compose profiles.

It is designed to support the full lifecycle of data engineering projects, from ingestion and transformation to analysis and visualization.