**Implementación Detallada de la Arquitectura de Big Data en un Entorno Local con Docker y Docker Compose**

Esta guía te proporcionará un plan detallado para configurar y probar una arquitectura de Big Data utilizando Docker y Docker Compose en un entorno local. La implementación cubrirá HDFS, Apache Spark, Apache Kafka, Apache Airflow, Prometheus, Grafana y Jenkins.

**1. Preparación del Entorno**

1. **Instalación de Docker:**
   * **Windows:** Guía de instalación
   * **Mac:** Guía de instalación
   * **Linux:** Guía de instalación
2. **Instalación de Docker Compose:**
   * **Windows y Mac:** Docker Desktop incluye Docker Compose.
   * **Linux:**

bash

Copiar código

sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

sudo chmod +x /usr/local/bin/docker-compose

docker-compose --version

1. **Crear el Directorio del Proyecto:**

bash

Copiar código

mkdir bigdata-poc

cd bigdata-poc

**2. Creación del Archivo docker-compose.yml**

Este archivo define todos los servicios que forman parte de la arquitectura.

yaml

Copiar código

version: '3.8'

services:

# HDFS

hdfs-namenode:

image: bde2020/hadoop-namenode:2.0.0-hadoop3.2.1-java8

environment:

- CLUSTER\_NAME=test

- CORE\_CONF\_fs\_defaultFS=hdfs://namenode:8020

ports:

- "9870:9870"

volumes:

- namenode:/hadoop/dfs/name

networks:

- bigdata

hdfs-datanode:

image: bde2020/hadoop-datanode:2.0.0-hadoop3.2.1-java8

environment:

- CORE\_CONF\_fs\_defaultFS=hdfs://namenode:8020

volumes:

- datanode:/hadoop/dfs/data

depends\_on:

- hdfs-namenode

networks:

- bigdata

# Spark

spark:

image: bitnami/spark:latest

environment:

- SPARK\_MODE=master

- SPARK\_MASTER\_NAME=spark-master

- SPARK\_MASTER\_PORT=7077

ports:

- "8080:8080"

volumes:

- spark-data:/bitnami/spark

networks:

- bigdata

spark-worker:

image: bitnami/spark:latest

environment:

- SPARK\_MODE=worker

- SPARK\_MASTER\_URL=spark://spark-master:7077

depends\_on:

- spark

volumes:

- spark-worker-data:/bitnami/spark

networks:

- bigdata

# Kafka y Zookeeper

zookeeper:

image: wurstmeister/zookeeper:latest

ports:

- "2181:2181"

networks:

- bigdata

kafka:

image: wurstmeister/kafka:latest

ports:

- "9092:9092"

environment:

KAFKA\_ZOOKEEPER\_CONNECT: zookeeper:2181

KAFKA\_ADVERTISED\_LISTENERS: PLAINTEXT://kafka:9092

KAFKA\_OFFSETS\_TOPIC\_REPLICATION\_FACTOR: 1

depends\_on:

- zookeeper

networks:

- bigdata

# Airflow

postgres:

image: postgres:latest

environment:

POSTGRES\_USER: airflow

POSTGRES\_PASSWORD: airflow

POSTGRES\_DB: airflow

ports:

- "5432:5432"

networks:

- bigdata

airflow:

image: puckel/docker-airflow:latest

restart: always

environment:

AIRFLOW\_\_CORE\_\_EXECUTOR: LocalExecutor

AIRFLOW\_\_CORE\_\_SQL\_ALCHEMY\_CONN: postgresql+psycopg2://airflow:airflow@postgres:5432/airflow

AIRFLOW\_\_CORE\_\_FERNET\_KEY: dGhpcyBpcyBhIHZlcnkgc2VjdXJlIGtleQ==

AIRFLOW\_\_CORE\_\_LOAD\_EXAMPLES: 'True'

ports:

- "8080:8080"

volumes:

- ./dags:/usr/local/airflow/dags

depends\_on:

- postgres

networks:

- bigdata

# Prometheus y Grafana

prometheus:

image: prom/prometheus:latest

ports:

- "9090:9090"

volumes:

- ./prometheus:/etc/prometheus

networks:

- bigdata

grafana:

image: grafana/grafana:latest

ports:

- "3000:3000"

volumes:

- grafana-storage:/var/lib/grafana

networks:

- bigdata

# Jenkins

jenkins:

image: jenkins/jenkins:lts

ports:

- "8081:8081"

- "50000:50000"

volumes:

- jenkins\_home:/var/jenkins\_home

networks:

- bigdata

volumes:

namenode:

datanode:

spark-data:

spark-worker-data:

grafana-storage:

jenkins\_home:

networks:

bigdata:

**3. Ejecución de Docker Compose**

Ejecuta el siguiente comando para iniciar todos los servicios definidos en el archivo docker-compose.yml:

bash

Copiar código

docker-compose up -d

**4. Interacción y Pruebas**

**HDFS**

* **Subir Archivos:**

bash

Copiar código

docker exec -it $(docker ps -qf "name=hdfs-namenode") /bin/bash

hdfs dfs -mkdir /user/test

hdfs dfs -put local\_file.txt /user/test/

* **Leer Archivos:**

bash

Copiar código

hdfs dfs -cat /user/test/local\_file.txt

**Spark**

* **Ejecutar Jobs:**

bash

Copiar código

docker exec -it $(docker ps -qf "name=spark") /bin/bash

spark-submit --master spark://spark-master:7077 your\_spark\_job.py

**Kafka**

* **Producir Mensajes:**

bash

Copiar código

docker exec -it $(docker ps -qf "name=kafka") /bin/bash

kafka-console-producer.sh --broker-list kafka:9092 --topic test

* **Consumir Mensajes:**

bash

Copiar código

kafka-console-consumer.sh --bootstrap-server kafka:9092 --topic test --from-beginning

**Airflow**

* **Acceder a la Interfaz Web:** <http://localhost:8080>
* **Subir DAGs:** Coloca tus DAGs en el directorio ./dags local.

**Prometheus y Grafana**

* **Acceder a Prometheus:** http://localhost:9090
* **Acceder a Grafana:** http://localhost:3000

**Jenkins**

* **Acceder a Jenkins:** http://localhost:8081

**Conclusión**

Este plan detallado te permite simular la implementación de una arquitectura de Big Data en un entorno local utilizando Docker y Docker Compose. Al seguir estos pasos, puedes validar la integración y funcionamiento de cada componente antes de desplegar la arquitectura en un entorno de producción. Asegúrate de realizar pruebas exhaustivas y documentar cualquier ajuste o problema que encuentres durante el proceso.

Para mantener esta arquitectura funcional y accesible para los desarrolladores en una organización, considera los siguientes puntos:

* **Configurar DNS interno** para resolver los nombres de dominio internos a las IPs de los servidores.
* **Implementar autenticación y autorización** en cada servicio para garantizar la seguridad.
* **Proporcionar documentación detallada** y **capacitación** para los desarrolladores sobre cómo acceder y utilizar cada servicio.
* **Configurar una VPN** para los desarrolladores que accedan desde fuera de la red interna de la organización.