

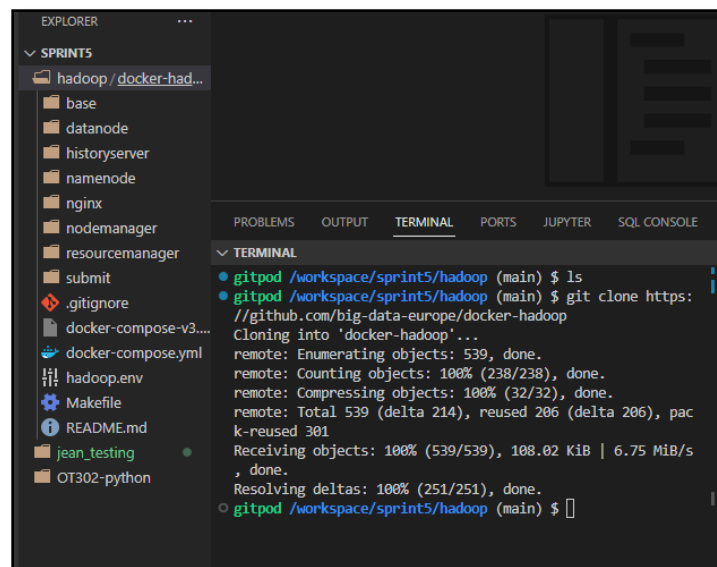
1) GitPod

Para la configuración del ambiente de desarrollo nos basamos en el servidor de gitpod, este servidor proporciona un IDE de trabajo con todos los elementos para desplegar la mayoría de los trabajos usando Docker.

- <https://gitpod.io/>

2) Imagen Docker Hadoop

Luego de abrir nuestro workspace utilizamos la imagen de Big-Data-Europe que constantemente recibe soporte, por lo cual abrimos nuestra terminal bash y la clonamos.



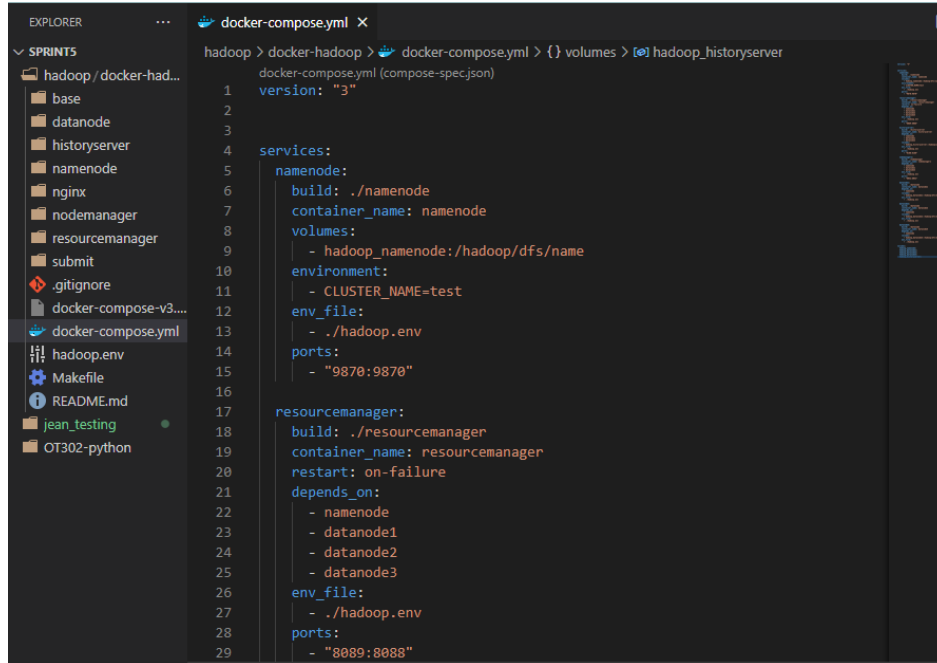
The screenshot shows a GitPod workspace interface. On the left, the 'EXPLORER' panel displays a file tree for a project named 'SPRINTS'. The tree includes a folder 'hadoop/docker-had...' and several subfolders like 'base', 'datanode', 'historyserver', 'namenode', 'nginx', 'nodemanager', 'resource manager', 'submit', and files like '.gitignore', 'docker-compose-v3...', 'docker-compose.yml', 'hadoop.env', 'Makefile', and 'README.md'. The main panel shows a 'TERMINAL' tab with the following output:

```
gitpod /workspace/sprint5/hadoop (main) $ ls
gitpod /workspace/sprint5/hadoop (main) $ git clone https://github.com/big-data-europe/docker-hadoop
Cloning into 'docker-hadoop'...
remote: Enumerating objects: 539, done.
remote: Counting objects: 100% (238/238), done.
remote: Compressing objects: 100% (32/32), done.
remote: Total 539 (delta 214), reused 206 (delta 206), pack-reused 301
Receiving objects: 100% (539/539), 108.02 KiB | 6.75 MiB/s, done.
Resolving deltas: 100% (251/251), done.
gitpod /workspace/sprint5/hadoop (main) $
```

- <https://github.com/big-data-europe/docker-hadoop>
- Git clone <https://github.com/big-data-europe/docker-hadoop>

3) Configuración del docker-compose.yml

Con la imagen de Docker para Hadoop en el servidor podemos usar docker-compose para configurar el clúster local de Hadoop. Se reemplaza el docker-compose.yml por el archivo siguiente teniendo en cuenta que solo se cambia el número de la versión.



```
1 version: "3"
2
3
4 services:
5   namenode:
6     build: ./namenode
7     container_name: namenode
8     volumes:
9       - hadoop_namenode:/hadoop/dfs/name
10    environment:
11      - CLUSTER_NAME=test
12    env_file:
13      - ./hadoop.env
14    ports:
15      - "9870:9870"
16
17   resourcemanager:
18     build: ./resourcemanager
19     container_name: resourcemanager
20     restart: on-failure
21     depends_on:
22       - namenode
23       - datanode1
24       - datanode2
25       - datanode3
26     env_file:
27       - ./hadoop.env
28     ports:
29       - "8089:8088"
```

➤ <https://gist.github.com/nathan815/a938b3f7a4d06b2811cf2b1a917800e1>

En este mismo archivo podemos establecer la configuración en la cual se mostrará el UI de Hadoop.

Ejemplo:

Ports :

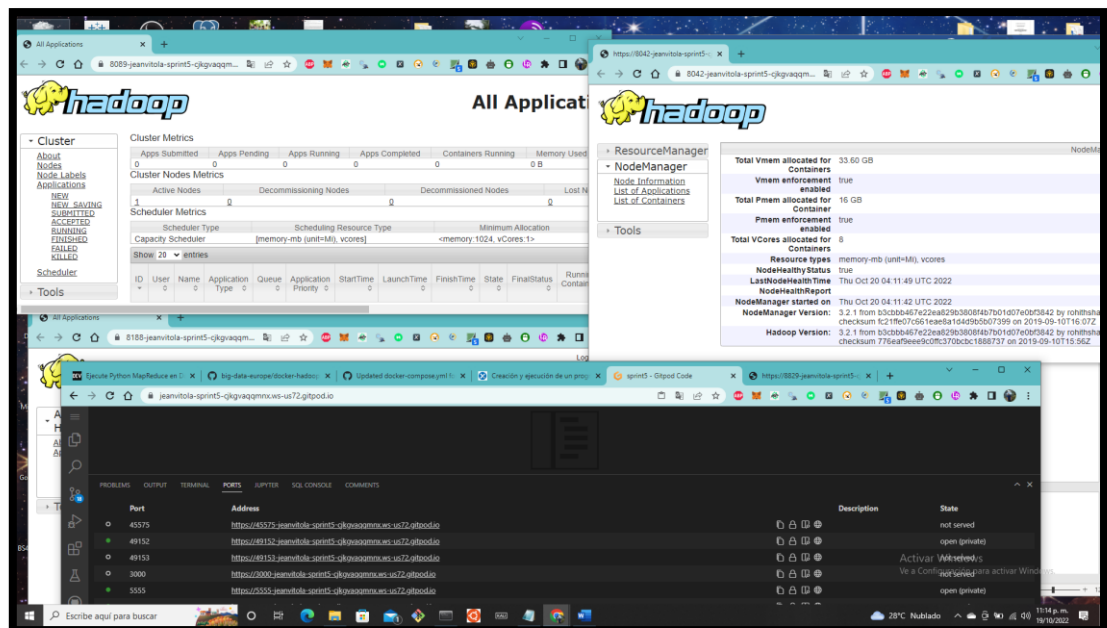
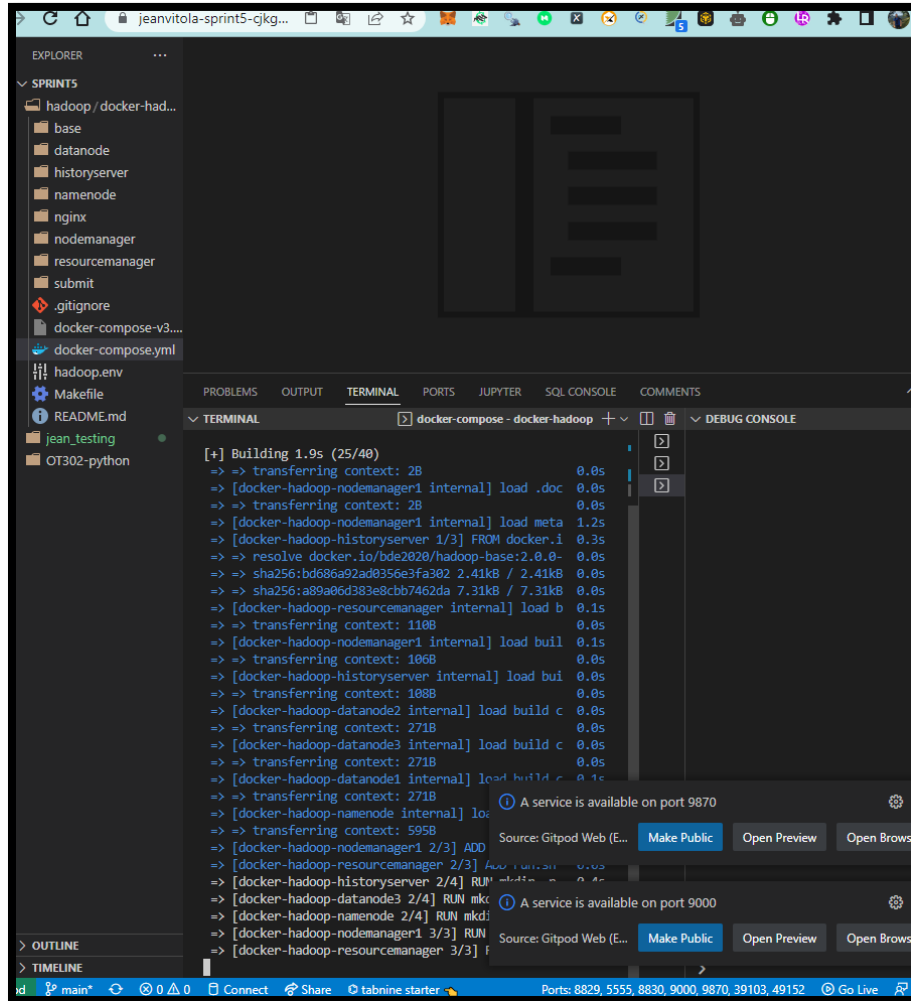
- "8042:8042"

4) Iniciar el container hadoop

Una vez que se configura el clúster de Hadoop con un nodo maestro que está en el archivo anteriormente mencionado se procede a levantar el contenedor de Hadoop. Abrimos la terminal de bash de gitpod y ejecutamos lo siguiente

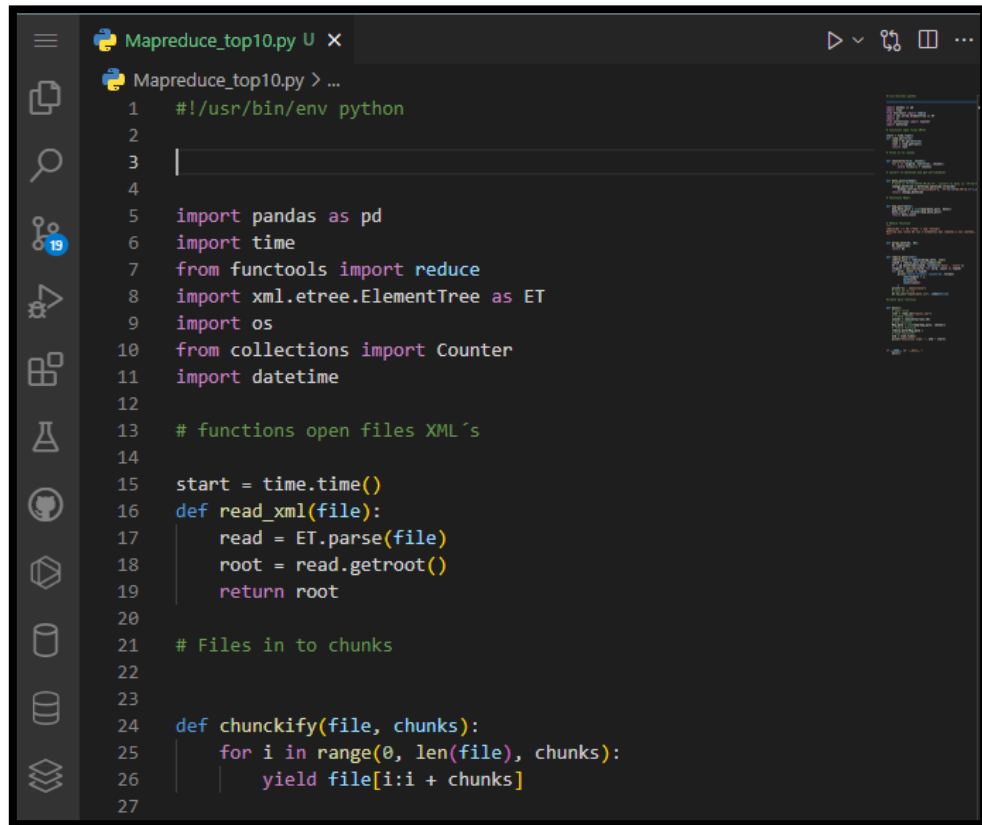
- docker-compose up -d
- docker-compose up

Nota: Hay que tener en cuenta que al momento de ejecutar el comando debemos verificar que estemos en la ruta correcta, ya que muchas veces en la terminal muestra error cuando ejecutamos el comando por fuera de la ruta



Ejecutando Python MapReduce

Para la aplicación utilizamos las funciones de Mapreduce hechas en anteriores ocasiones para ejecutarlas dentro del Hadoop.

A screenshot of a code editor window titled 'Mapreduce_top10.py U x'. The editor shows a Python script with the following content:

```
1  #!/usr/bin/env python
2
3
4
5  import pandas as pd
6  import time
7  from functools import reduce
8  import xml.etree.ElementTree as ET
9  import os
10 from collections import Counter
11 import datetime
12
13 # functions open files XML's
14
15 start = time.time()
16 def read_xml(file):
17     read = ET.parse(file)
18     root = read.getroot()
19     return root
20
21 # Files in to chunks
22
23
24 def chunkify(file, chunks):
25     for i in range(0, len(file), chunks):
26         yield file[i:i + chunks]
27
```

Guardamos el archivo y en la terminal le damos permisos para poder ejecutar el Mapreduce :

- `chmod u+x Mapreduce_top10.py`

Agregamos el archivo de postst.xml a la carpeta donde esta nuestro hadoop

Configuración interna de Docker para ejecutar los archivos

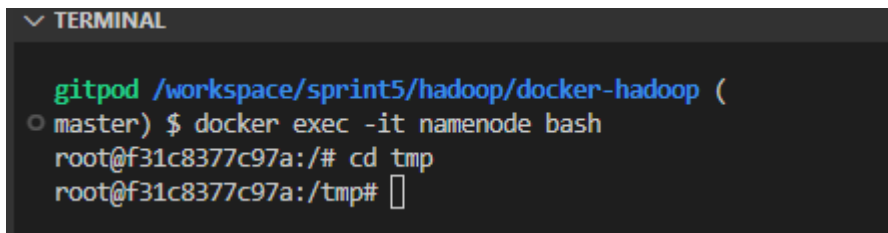
Nos ubicamos en la termina y en el bash ejecutamos la siguiente función para acceder al bash de docker :

- `docker exec -it namenode bash`
- `hdfs dfs -ls / <---` Listar carpeta del nodo raíz
- `hdfs dfs -mkdir -p /user/root <----` crear carpeta root

Ahora pasamos los archivos del Mapreducer y del posts.xml al container de docker.

- Ejecutamos el comando “exit” dentro del bash de docker para volver a nuestra terminal
- `docker cp Mapreduce_top10.py namenode:/tmp`
- `docker cp posts.xml namenode:/tmp`

Ahora ingresamos nuevamente al bash de docker :



```
✓ TERMINAL
gitpod /workspace/sprint5/hadoop/docker-hadoop (
○ master) $ docker exec -it namenode bash
root@f31c8377c97a:/# cd tmp
root@f31c8377c97a:/tmp#
```

- `docker exec -it namenode bash`
- `hdfs dfs -put posts.xml /user/root/input` ← Copio el archivo hacia la carpeta input
- `hadoop jar /opt/hadoop-3.2.1/share/hadoop/tools/lib/hadoop-streaming-3.2.1.jar \`
`- Mapreduce_top10.py - Mapreduce_top10.py \`
`-input input -output output`