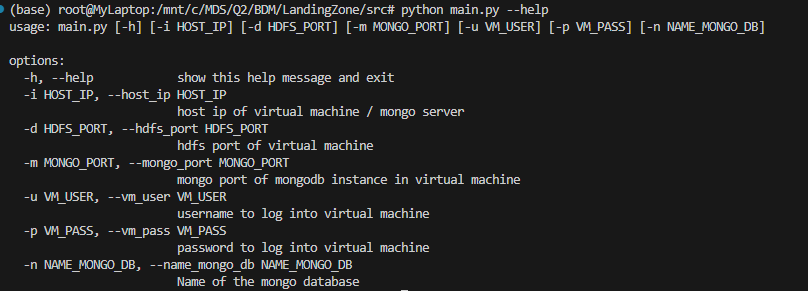
## QuickStart instructions

In order to work with the repository, we will need to have installed python3 in our case we used Python 3.11. First, we will have to install all dependencies by using the requirements.txt file in the root directory of the repository. After that we must have all the information, we want to load in the resources folder (not the unemployment data which is pulled by an API). Subsequently, all the programs can be processed through the main.py, we can modify the following parameters to maximize reusability as follows:



# Temporal Landing Zone

## Architecture

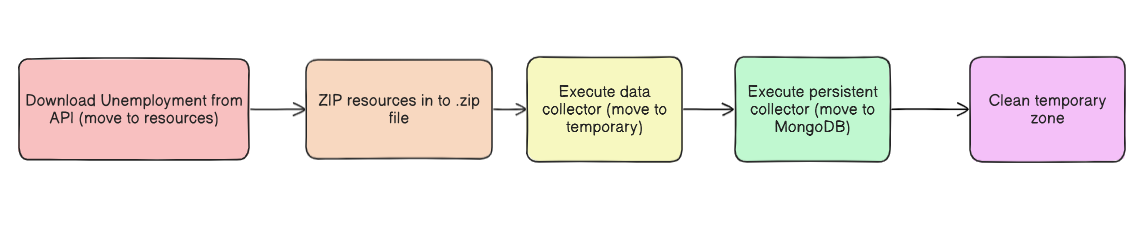
The heart of this architecture lies within the HDFS, residing on the virtual machine. This choice of HDFS as the foundation because of its robustness and distributed storage capability. HDFS stands out for its ability to handle various file types without necessitating modifications. This feature proves invaluable in managing source heterogeneity, HDFS accommodates diverse data formats effortlessly.

One of the key features of HDFS is automatic replication, ensuring data security and availability. In this architecture, HDFS replicates data 3 times which provides from fault tolerance. Additionally, data distribution across available machines occurs randomly.

The architecture is very simple to scale by simply adding extra nodes and increasing capacity of compute and storage although in this demo is not shown as we only use 1 VM.

The lowest level of abstraction in the whole architecture will be the file, not the row/tuple. So, we will move the data by files in the temporal zone. Storing original formats of source files will guarantee flexibility and resistance to changes in data sources over time.

The landing zone resides within the HDFS, accessible at /user/temporary. In order to copy the data in the landing zone we will download, locally, the unemployment information through an API, after that we will ZIP the whole contents and upload the information to the VM. Subsequently, we will copy the information from the zip file (uncompressed) to the Landing Zone. We can also see how it works in the following diagram:



## Assumptions

Todo el proceso se ejecuta de una sola vez utilizando el main.py , pero se puede ejecutar por partes también.

Hemos usado una api para obtener los datos del 4 dataset. En este script lo que hacemos principalmente es guardar el archivo en un entorno local, luego juntarlo con el data.zip que ya teníamos previamente y cargarlo en la maquina virtual siguiendo el mismo proceso que los otros datasets. Para correr este script tenemos que tener instalados los siguientes paquetes (…)

Explicar lo de los data to persistent

.

.

Main.py ejecuta todos los scripts, primero se descarga los archivos, luego lo comprime y lo mueve al fichero de resources y finalmente se comprime todo y se envia a la maquina virtual. Aquí se ejecuta el data collector que lo mueve al temporal landing zone (que nosotros hemos escogido hdfs) y luego se ejecuta el persistent collector. 1 Crea un objeto set que mira si el fichero existe, si esta no lo sube. Entonces la primera vez que lo ejecutes lo cargara todo pero en la segunda ya no 2

1 La forma de trackearlo se ha hecho con un .pickle ya que es mas fácil hacerlo así y esta mas integrado en Python ya que pickle es la forma de guardar objetos de Python

2 La forma en la que lo hemos cargado no es la ideal ya que al final estamos usando Python y Python no paraleliza. Para substituirlo se debería usar spark o cualquier otro framework que sea paralelizado. Lo hemos intentado pero no hemos podido conectar la master con nuestra maquina posiblemente por incompatibilidad de versiones. Si el data incrementara se necesitarían soluciones mas optimas.

**1 temporal landing zone**

The temporary landing zone consists of a directory within a Hadoop Distributed File System (HDFS) on a virtual machine. HDFS was chosen as the technology for the temporal landing zone as it allows the data to be stored in distributed fashion and supports all different types of files, without any modifications, thus swiftly handling source heterogeneity’s

Furthermore, HDFS automatically produces replicas (in this case 3) of the data, which provides a security to the data storage as well as better availability [1]. As HDFS distributes data randomly across the machines available, it is not needed to design a key which evenly distributes the data in this zone. Moreover, this implies good scalabilty as extra nodes can easily be added and thus provide improved storage and processing capabilities [1].

Another nice property of HFDS is its rebalancing feature, which helps storing data evenly across nodes, preventing one to have much more data compared to others.

Within the HDSF, the temporary landing zone can be found under /user/temporary.

DIBUJO DE TEMPORARY