Carpeta 1 — EC2 con Python + Pandas

1. Inicializa Terraform dentro de la carpeta del proyecto:
   terraform init
2. Despliega la infraestructura:
   terraform apply -auto-approve
3. Cuando termine, consulta el ID de la instancia y la IP pública:
   terraform output
4. Conéctate a la instancia con AWS SSM (no es necesario abrir puertos de SSH):
   aws ssm start-session --target <INSTANCE_ID>
5. Una vez dentro de la máquina, valida que Python está instalado:
   python3 --version
6. Verifica que Pandas se instaló correctamente:
   python3 -c "import pandas as pd; print(pd.**version**)"

PANDAS

```
Location: /usr/local/lib/python3.10/dist-packages
Requires: numpy, python-dateutil, pytz, tzdata
Required-by:
$ Santiago
sh: 5: Santiago: not found
$ python3
Python 3.10.12 (main, Aug 15 2025, 14:32:43) [GCC 11.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import pandas as pd
print(pd.__version__)

df = pd.DataFrame({"A": [1, 2, 3], "B": ["x", "y", "z"]})
print(df)
>>> print(pd.__version__)
2.3.3
>>>
>>> df = pd.DataFrame({"A": [1, 2, 3], "B": ["x", "y", "z"]})
>>> print(df)
   A  B
0  1  x
1  2  y
2  3  z
>>>
```

Carpeta 2 — EC2 con Python + Polars

1. Inicializa Terraform en esta carpeta:
   terraform init
2. Crea la instancia automáticamente:
   terraform apply -auto-approve
3. Revisa los outputs para obtener el ID de la instancia:
   terraform output
4. Conéctate a la instancia con SSM:
   aws ssm start-session --target <INSTANCE_ID>
5. Verifica instalación de Python:
   python3 --version
6. Comprueba que Polars está disponible:
   python3 -c "import polars as pl; print(pl.version)"

POLARS

```
>>>
$ print(df.to_pandas())
sh: 20: Syntax error: word unexpected (expecting ")")
$ python3
Python 3.10.12 (main, Aug 15 2025, 14:32:43) [GCC 11.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import polars as pl

df = pl.DataFrame({
    "id": [1, 2, 3],
  >>>
>>> df = pl.DataFrame({
...     "id": [1, 2, 3],
...     "nombre": ["Alice", "Bob", "Charlie"],
...     "edad": [25, 30, 35]
... })
>>>
>>> pl.Config.set_tbl_formatting("ASCII_FULL")  # salida más limpia
<class 'polars.config.Config'>i
>>> print(df)
shape: (3, 3)
+-----+---------+------+
| id  | nombre  | edad |
| --- | ---     | ---  |
| i64 | str     | i64  |
+=====================+
| 1   | Alice   | 25   |
|-----+---------+------|
| 2   | Bob     | 30   |
|-----+---------+------|
| 3   | Charlie | 35   |
+-----+---------+------+
>>>
```

Carpeta 3 — EC2 con Python + DuckDB

1. Inicia Terraform en la carpeta correspondiente:
   terraform init
2. Aplica el despliegue:
   terraform apply -auto-approve
3. Verifica el ID de la instancia y conéctate con SSM:
   terraform output
   aws ssm start-session --target <INSTANCE_ID>
4. Dentro de la instancia, revisa que Python está disponible:
   python3 --version
5. Valida la instalación de DuckDB:
   python3 -c "import duckdb; print(duckdb.version)"
6. Prueba con un ejemplo sencillo en un script test_duckdb.py:

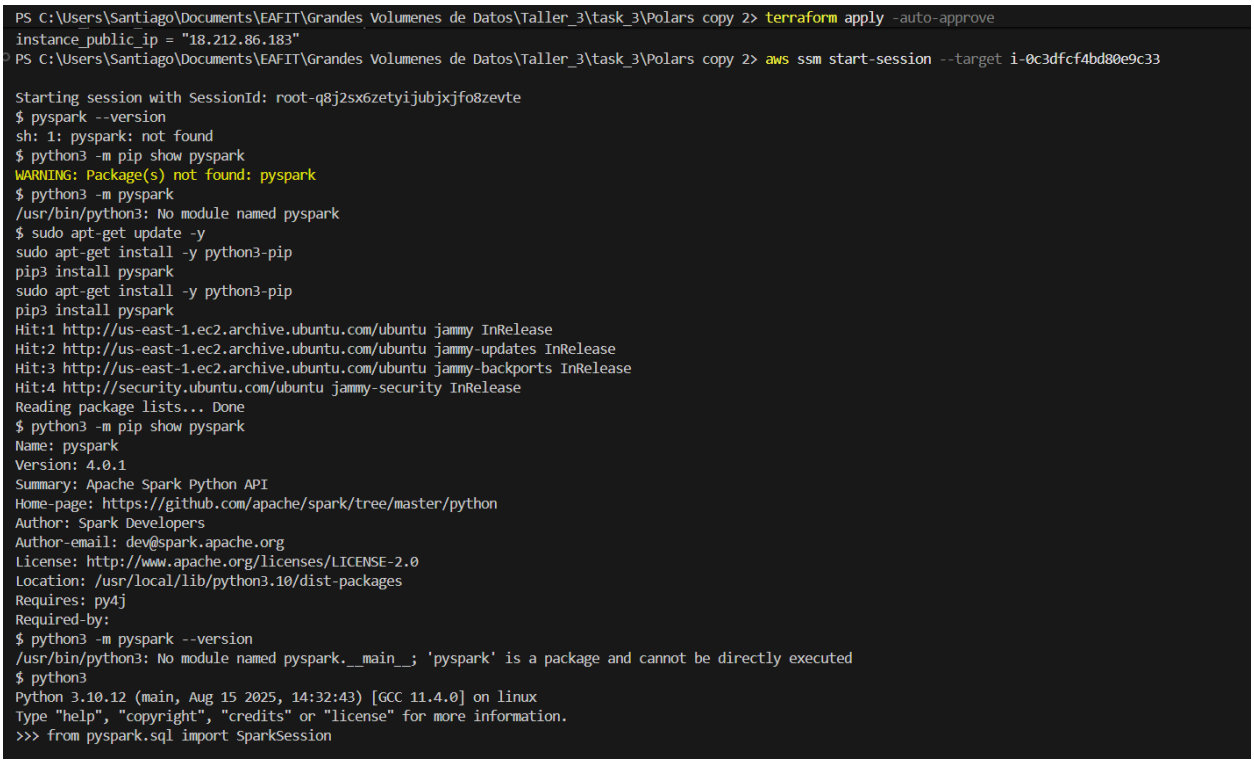
DUCK

```
> print(resultado)
> EOF
> cat > test_duckdb.py << 'EOF'
> import duckdb
> import pandas as pd
>
df>  = pd.DataFrame({
>     "id": [1, 2, 3, 4],
>     "nombre": ["Alice", "Bob", "Charlie", "Diana"],
>     "edad": [25, 30, 35, 40]
}> )
>
> con = duckdb.connect()
> con.register("personas", df)
>
> resultado = con.execute("SELECT nombre, edad FROM personas WHERE edad > 30").fetchdf()
>
> print("Resultado de DuckDB:")
> print(resultado)
E> OF
> python3 test_duckdb.py
> apt-get update -y
> apt-get install -y python3 python3-pip
> python3 -m pip install --upgrade pip setuptools wheel
p> ython3 -m pip install duckdb
> apt-get update -y
> apt-get install -y python3 python3-pip
p> ython3 -m pip install --upgrade pip setuptools wheel
> python3 -m pip install duckdb
> apt-get update -y
> apt-get install -y python3 python3-pip
> python3 -m pip install --upgrade pip setuptools wheel
> python3 -m pip install duckdb
> apt-get update -y
> apt-get install -y python3 python3-pip
p> ython3 -m pip install --upgrade pip setuptools wheel
> python3 -m pip install duckdb
> python3 -c "import duckdb; print(duckdb.__version__)"
```

Carpeta 4 — EC2 con Python + Spark (PySpark)

1. Ejecuta Terraform para crear la instancia:
   terraform init
   terraform apply -auto-approve
2. Obtén el ID de la instancia creada:
   terraform output
3. Conéctate con SSM:
   aws ssm start-session --target <INSTANCE_ID>
4. Verifica la instalación de PySpark dentro de la máquina:
   python3 -c "import pyspark; print(pyspark.version)"

SPARK



Carpeta 5 — Cluster EMR con Spark distribuido

1. Crear el cluster con Terraform

---

terraform init

terraform apply -auto-approve

Espera a que el cluster esté en estado WAITING o RUNNING.

2. Obtener el ID del cluster

---

aws emr list-clusters --active

** Copia el valor de ClusterId.

3. Verificar estado del cluster

---

aws emr describe-cluster --cluster-id <CLUSTER_ID> --query "Cluster.Status.State"

👉 Debe responder WAITING o RUNNING.

4. Obtener el ID de la instancia master

---

aws emr list-instances --cluster-id <CLUSTER_ID> --instance-group-types MASTER --query "Instances[*].Ec2InstanceId" --output text

5. Conectarse al nodo master con SSM (recomendado)

---

aws ssm start-session --target <MASTER_INSTANCE_ID>

👉 Ejemplo:

aws ssm start-session --target i-0a6ef27cf4be79182

6. Validar Spark en el nodo master

---

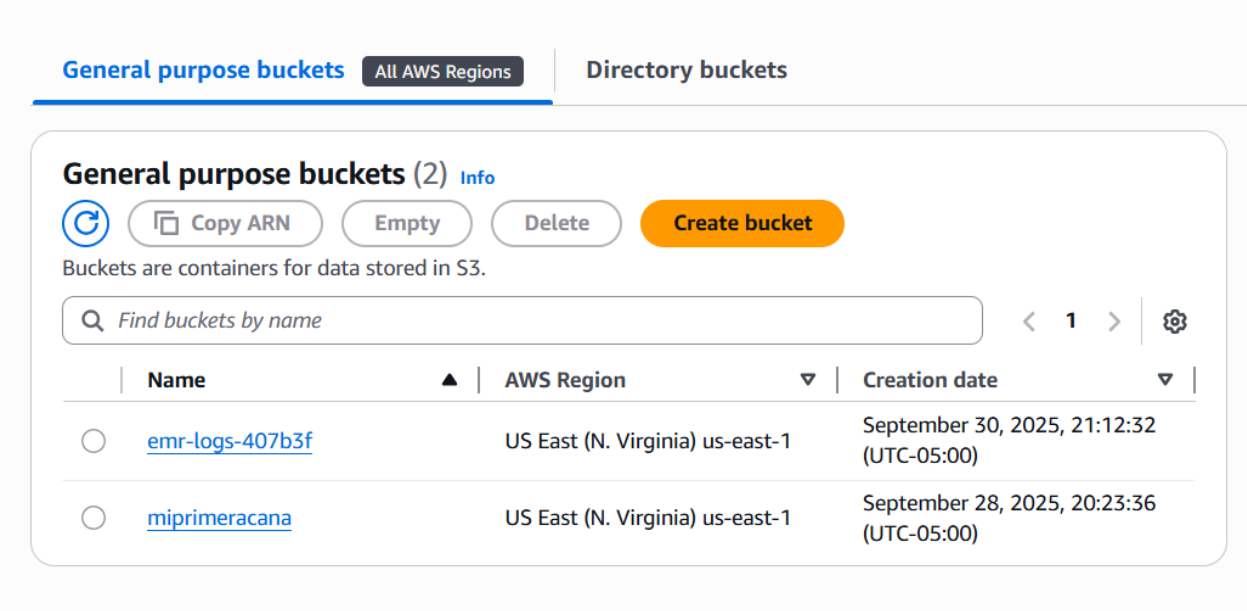
pyspark --version

EMR

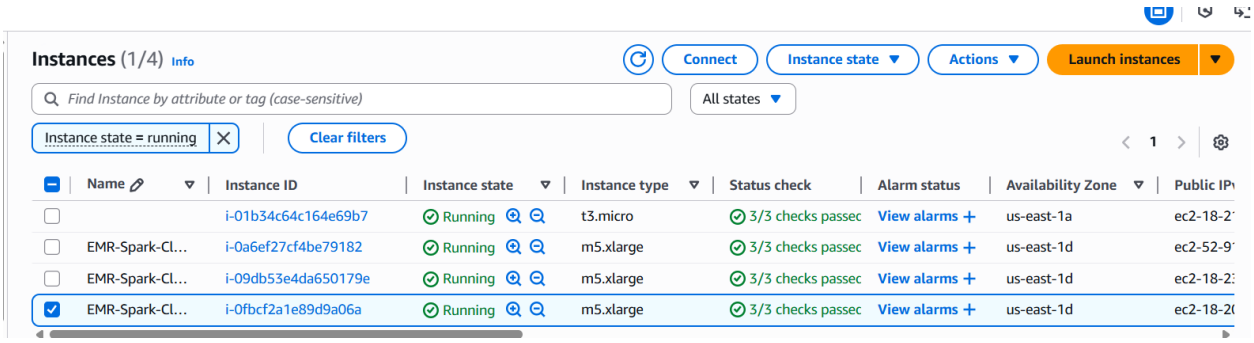
```
PS C:\Users\Santiago\Documents\EAFIT\Grandes Volumenes de Datos\Taller_3\task_3\EMR> aws emr describe-cluster --cluster-id j-3617LDYAXQIF8 --query "Cluster.Stat
us.State"
"WAITING"

PS C:\Users\Santiago\Documents\EAFIT\Grandes Volumenes de Datos\Taller_3\task_3\EMR> aws emr list-instances --cluster-id j-3617LDYAXQIF8 --instance-group-types
MASTER --query "Instances[*].Ec2InstanceId" --output text
i-0a6ef27cf4be79182

PS C:\Users\Santiago\Documents\EAFIT\Grandes Volumenes de Datos\Taller_3\task_3\EMR> aws ssm start-session --target i-0a6ef27cf4be79182

Starting session with SessionId: root-hol3ijojy7hoxl3jbvil5id888
sh-4.2$ python3 --version
pyspark --vePython 3.7.16
sh-4.2$ pyspark --version
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 3.4.1-amzn-2
      /_/

Using Scala version 2.12.15, OpenJDK 64-Bit Server VM, 1.8.0_462
Branch
Compiled by user release on 2024-03-14T04:28:38Z
Revision
Url
Type --help for more information.
sh-4.2$
```

AWS SCREENSHOTS

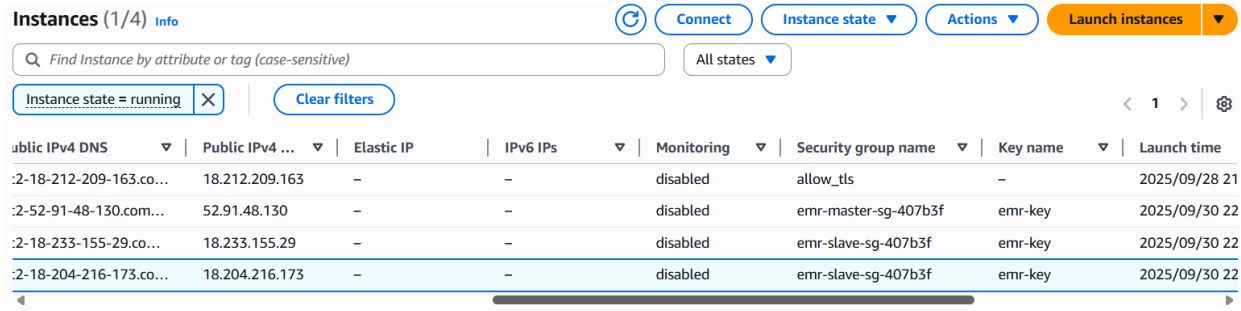
S3



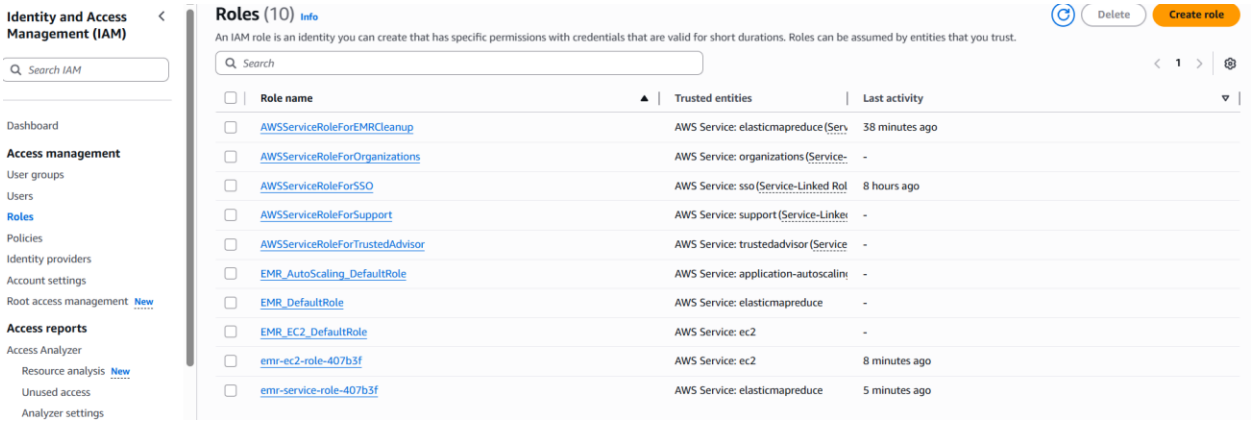
EC2



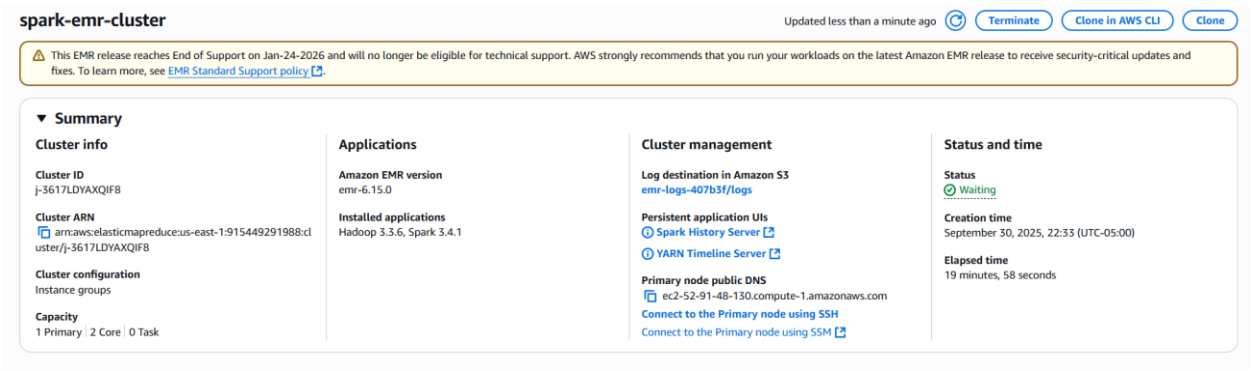
3 INSTANCIAS PARA EL EMR



IAM ROLES



EMR


spark-emr-cluster
Updated less than a minute ago
Terminate
Clone in AWS CLI
Clone
This EMR release reaches End of Support on Jan-24-2026 and will no longer be eligible for technical support. AWS strongly recommends that you run your workloads on the latest Amazon EMR release to receive security-critical updates and fixes. To learn more, see EMR Standard Support policy.
Summary
Cluster info
Cluster ID
j-3617LDYAXQIF8
Cluster ARN
arn:aws:elasticmapreduce:us-east-1:915449291988:cluster/j-3617LDYAXQIF8
Cluster configuration
Instance groups
Capacity
1 Primary | 2 Core | 0 Task
Applications
Amazon EMR version
emr-6.15.0
Installed applications
Hadoop 3.3.6, Spark 3.4.1
Cluster management
Log destination in Amazon S3
emr-logs-407b3f/logs
Persistent application UIs
Spark History Server
YARN Timeline Server
Primary node public DNS
ec2-52-91-48-130.compute-1.amazonaws.com
Connect to the Primary node using SSH
Connect to the Primary node using SSM
Status and time
Status
Waiting
Creation time
September 30, 2025, 22:33 (UTC-05:00)
Elapsed time
19 minutes, 58 seconds