

Before you turn this problem in, make sure everything runs as expected. First, **restart the kernel** (in the menubar, select Kernel→Restart) and then **run all cells** (in the menubar, select Cell→Run All).

Make sure you fill in any place that says `YOUR CODE HERE` or "YOUR ANSWER HERE", as well as your name and collaborators below:

In [ ]:

```
NAME = "Vicent Ripoll Ramírez"
COLLABORATORS = ""
```

## PEC 2: Ejecución de trabajos mediante el gestor de recursos YARN

### 0 Nota

En muchos de los ejercicios se deberán realizar **capturas de pantalla que justifiquen las respuestas**. Las capturas de pantalla se pueden realizar con las herramientas del sistema operativo que estéis utilizando. Para copiar la imagen en el portapapeles podéis utilizar diferentes métodos: "Herramienta de Recortes" en Windows, "Imprimir pantalla", Ctrl+C al seleccionar una imagen, etc.... Las imágenes una vez capturadas se pueden pegar directamente en las celdas de respuesta, mediante Ctrl+V o con el menú contextual que aparece con el botón derecho del ratón, lo que pegará la imagen que esté en el portapapeles. Para ver la imagen se debe ejecutar la celda.

### 1 Introducción

Hasta ahora en los ejercicios realizados hemos visto como ejecutar trabajos en un entorno Big Data pero no nos hemos fijado en como estos trabajan, interactúan y se ejecutan dentro del sistema. Este es precisamente el objetivo de este ejercicio.

Antes de empezar con el anunciado de la PEC, vamos a revisar un poco la composición del sistema Big Data que estáis usando. En la siguiente imagen, mediante el Cloudera Manager, el gestor del sistema que utilizamos [[CDH Components](https://www.cloudera.com/products/open-source/apache-hadoop/key-cdh-components.html)] (<https://www.cloudera.com/products/open-source/apache-hadoop/key-cdh-components.html>), podemos ver como está configurado el gestor de recursos Yarn. Es interesante ver (Figura 1) que en nuestro caso el clúster está compuesto por 4 máquinas, una principal, y más grande, y tres más modestas. Podéis ver las características de las máquinas en las diversas columnas de la izquierda. En vuestro caso, cuando accedéis al sistema, lo hacéis mediante la máquina "eimtcld.uoc.edu".

The screenshot shows the Cloudera Manager interface with the 'Hosts' tab selected. The page displays a list of hosts under the 'eimtCLD' cluster. The hosts are: Cloudera02, Cloudera03, Cloudera04, and eimtcld.uoc.edu. All hosts are in a 'Good Health' status. The table includes columns for Status, Name, IP, Roles, Commission State, Last Heartbeat, Load Average, Disk Usage, Physical Memory, and Swap Space.

Status	Name	IP	Roles	Commission State	Last Heartbeat	Load Average	Disk Usage	Physical Memory	Swap Space
Good Health	Cloudera02	10.20.30.2	9 Role(s)	Commissioned	4.04s ago	0.07 0.08 0.03	149 GiB / 491.2 GiB	3.9 GiB / 30.7 GiB	0 B / 953 MiB
Good Health	Cloudera03	10.20.30.3	9 Role(s)	Commissioned	4.44s ago	0.06 0.04 0.00	148.9 GiB / 491.2 GiB	3.9 GiB / 30.7 GiB	0 B / 953 MiB
Good Health	Cloudera04	10.20.30.4	9 Role(s)	Commissioned	6.18s ago	0.01 0.02 0.00	149 GiB / 491.2 GiB	3.9 GiB / 30.7 GiB	0 B / 953 MiB
Good Health	eimtcld.uoc.edu	213.73.35.119	28 Role(s)	Commissioned	5.75s ago	0.71 0.65 0.69	219.1 GiB / 491.2 GiB	31.1 GiB / 124.9 GiB	0 B / 953 MiB

Figura 1: Descripción del sistema Cloudera instalado para la realización de las prácticas.

En la Figura 2 podéis ver la configuración del resource manager usado, el Yarn. Vemos que cada una de las máquinas está corriendo un "Node manager" de Yarn gestionado por el "Resource Manager" que se ejecuta en la máquina principal. Si no recordáis bien estos conceptos podéis revisar la Sección 5 del contenido teórico del módulo actual.

Por lo tanto, cualquier proceso que se ejecute mediante Yarn se va a ejecutar (si lo considera necesario) utilizando todos los recursos del sistema. En este ejercicio vamos a ver como usar Yarn, como analizar las características del programa ejecutado y como ha finalizado la ejecución.

The screenshot shows the Cloudera Manager interface with the 'YARN (MR2 Included)' tab selected. The page displays the configuration of Yarn roles. The roles are: JobHistory Server, NodeManager, and ResourceManager (Active). All roles are in a 'Good Health' status. The table includes columns for Role Type, State, Host, Commission State, and Role Group.

Role Type	State	Host	Commission State	Role Group
JobHistory Server	Started	eimtcld.uoc.edu	Commissioned	JobHistory Server Default Group
NodeManager	Started	eimtcld.uoc.edu	Commissioned	NodeManager Default Group
NodeManager	Started	Cloudera02	Commissioned	NodeManager Group 2
NodeManager	Started	Cloudera04	Commissioned	NodeManager Group 1
NodeManager	Started	Cloudera03	Commissioned	NodeManager Group 3
ResourceManager (Active)	Started	eimtcld.uoc.edu	Commissioned	ResourceManager Default Group

Figura 2: Configuración del gestor de recursos Yarn en la máquina usada en la asignatura.

Si revisamos la configuración del Spark Context que estábamos usando hasta ahora en los notebooks:

```
sc = pyspark.SparkContext(master="local[1]",
appName="PAC1_vuestro_loginUOC")
```

podemos ver que el parámetro `master="local[1]"` indica que el programa sólo se ejecutará usando la máquina `local[1]`. Para entender bien los parámetros de configuración, se recomienda revisar el manual de referencia [\[Spark Programming Guide\]](https://spark.apache.org/docs/2.1.1/programming-guide.html#initializing-spark) (<https://spark.apache.org/docs/2.1.1/programming-guide.html#initializing-spark>). Por lo tanto, en los ejercicios ejecutados mediante notebooks Jupyter, no estamos utilizando el gestor de recursos, sino que ejecutamos los programas solamente utilizando la máquina principal. Para configurar el Spark Context mediante el gestor de recursos solo haría falta cambiar el parámetro `master="local[1]"` por `master="yarn"`. Aun así, en esta PEC nosotros vamos a lanzar los programas directamente desde la terminal de Linux mediante el comando `spark-submit`.

## 2 Ejecución de Spark (2.5 puntos)

### 2.1 Descripción del programa SparkPi (0.5 puntos)

El este ejercicio práctico vamos a trabajar sobre un programa ya compilado y preparado para ejecutarse. El programa que vamos a usar es un ejemplo que viene incorporado en la instalación de Cloudera, se suele usar como ejemplo base para realizar tareas de comprobación de operatividad del sistema y rendimiento. Podéis encontrar el programa en:

```
/opt/cloudera/parcels/CDH/lib/spark/examples/jars/spark-examples_2.11-2.4.0-cdh6.2.0.jar
```

Como podéis ver el programa está construido en Java, pero no vamos a modificarlo. Solo lo vamos a ejecutar en el siguiente apartado.

En este primer apartado se pide que describáis:

¿Qué hace el programa?

Da una aproximación del valor de pi.

```
Seleccíonar vripollr@Cloudera01: ~
3, 4))
22/10/27 17:01:35 INFO scheduler.TaskSchedulerImpl: Adding task set 0.0 with 5 tasks
22/10/27 17:01:35 INFO scheduler.TaskSetManager: Starting task 0.0 in stage 0.0 (TID 0, localhost, executor driver, partition 0, PROCESS_LOCAL, 7730 bytes)
22/10/27 17:01:35 INFO scheduler.TaskSetManager: Starting task 1.0 in stage 0.0 (TID 1, localhost, executor driver, partition 1, PROCESS_LOCAL, 7732 bytes)
22/10/27 17:01:35 INFO scheduler.TaskSetManager: Starting task 2.0 in stage 0.0 (TID 2, localhost, executor driver, partition 2, PROCESS_LOCAL, 7732 bytes)
22/10/27 17:01:35 INFO scheduler.TaskSetManager: Starting task 3.0 in stage 0.0 (TID 3, localhost, executor driver, partition 3, PROCESS_LOCAL, 7732 bytes)
22/10/27 17:01:35 INFO scheduler.TaskSetManager: Starting task 4.0 in stage 0.0 (TID 4, localhost, executor driver, partition 4, PROCESS_LOCAL, 7732 bytes)
22/10/27 17:01:35 INFO executor.Executor: Running task 0.0 in stage 0.0 (TID 0)
22/10/27 17:01:35 INFO executor.Executor: Running task 3.0 in stage 0.0 (TID 3)
22/10/27 17:01:35 INFO executor.Executor: Running task 2.0 in stage 0.0 (TID 2)
22/10/27 17:01:35 INFO executor.Executor: Running task 4.0 in stage 0.0 (TID 4)
22/10/27 17:01:35 INFO executor.Executor: Fetching spark://Cloudera01:34642/jars/spark-examples_2.11-2.4.0-cdh6.2.0.jar with timestamp 1666882893562
22/10/27 17:01:35 INFO client.TransportClientFactory: Successfully created connection to Cloudera01/213.73.35.119:34642 after 30 ms (0 ms spent in bootstraps)
22/10/27 17:01:35 INFO util.Utils: Fetching spark://Cloudera01:34642/jars/spark-examples_2.11-2.4.0-cdh6.2.0.jar to /tmp/spark-25ba8162-3fd8-4e93-8796-19ce554f43f6/userFiles-0573cbc6-8849-4ce1-b15f-615fda6ce05f/fetchFileTemp1691085300152030699.tmp
22/10/27 17:01:35 INFO executor.Executor: Adding file:/tmp/spark-25ba8162-3fd8-4e93-8796-19ce554f43f6/userFiles-0573cbc6-8849-4ce1-b15f-615fda6ce05f/spark-examples_2.11-2.4.0-cdh6.2.0.jar to class loader
22/10/27 17:01:36 INFO executor.Executor: Finished task 1.0 in stage 0.0 (TID 1). 713 bytes result sent to driver
22/10/27 17:01:36 INFO executor.Executor: Finished task 2.0 in stage 0.0 (TID 2). 713 bytes result sent to driver
22/10/27 17:01:36 INFO scheduler.TaskSetManager: Finished task 2.0 in stage 0.0 (TID 2) in 516 ms on localhost (executor driver) (1/5)
22/10/27 17:01:36 INFO scheduler.TaskSetManager: Finished task 1.0 in stage 0.0 (TID 1) in 527 ms on localhost (executor driver) (2/5)
22/10/27 17:01:36 INFO executor.Executor: Finished task 3.0 in stage 0.0 (TID 3). 713 bytes result sent to driver
22/10/27 17:01:36 INFO executor.Executor: Finished task 0.0 in stage 0.0 (TID 0). 713 bytes result sent to driver
22/10/27 17:01:36 INFO executor.Executor: Finished task 4.0 in stage 0.0 (TID 4). 713 bytes result sent to driver
22/10/27 17:01:36 INFO scheduler.TaskSetManager: Finished task 4.0 in stage 0.0 (TID 4) in 537 ms on localhost (executor driver) (3/5)
22/10/27 17:01:36 INFO scheduler.TaskSetManager: Finished task 3.0 in stage 0.0 (TID 3) in 544 ms on localhost (executor driver) (4/5)
22/10/27 17:01:36 INFO scheduler.TaskSetManager: Finished task 0.0 in stage 0.0 (TID 0) in 586 ms on localhost (executor driver) (5/5)
22/10/27 17:01:36 INFO scheduler.TaskSchedulerImpl: Removed TaskSet 0.0, whose tasks have all completed, from pool
22/10/27 17:01:36 INFO scheduler.DAGScheduler: ResultStage 0 (reduce at SparkPi.scala:38) finished in 0,955 s
22/10/27 17:01:36 INFO scheduler.DAGScheduler: Job 0 finished: reduce at SparkPi.scala:38, took 1,019363 s
Pi is roughly 3.142742285484571
22/10/27 17:01:36 INFO spark.MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
22/10/27 17:01:36 INFO memory.MemoryStore: MemoryStore cleared
22/10/27 17:01:36 INFO storage.BlockManager: BlockManager stopped
22/10/27 17:01:36 INFO storage.BlockManagerMaster: BlockManagerMaster stopped
22/10/27 17:01:36 INFO scheduler.OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
22/10/27 17:01:36 INFO spark.SparkContext: Successfully stopped SparkContext
22/10/27 17:01:36 INFO util.ShutdownHookManager: Shutdown hook called
22/10/27 17:01:36 INFO util.ShutdownHookManager: Deleting directory /tmp/spark-25ba8162-3fd8-4e93-8796-19ce554f43f6
22/10/27 17:01:36 INFO util.ShutdownHookManager: Deleting directory /tmp/spark-7cb0414d-225c-4d57-a95c-83e698756c82
vripollr@Cloudera01: $
```

¿Cómo lo hace?

Mediante la aplicación Spark Pi, que implementa una variación del problema clásico de probabilidad de la "Aguja de Bufón". Concretamente, define el cuadrado de vértices (0,0), (0,1), (1,0) y (1,1) así como la circunferencia inscrita, centrada en (0.5,0.5). El programa genera puntos en este cuadrado y cuenta cuántos de ellos caen en la circunferencia. Nótese que el área del cuadrado es 1 y la de la circunferencia  $\pi/4$ . La ley de grandes números asegura que los ensayos que realiza el programa tienden al ratio entre ambas áreas, que es  $\pi/4$ . Por este motivo, el programa finaliza multiplicando por 4.

```
Seleccionar vripollr@Cloudera01: ~
vripollr@Cloudera01: $ spark-submit --class org.apache.spark.examples.SparkPi --master="yarn" /opt/cloudera/parcels/CDH/lib/spark/examples/jars/spark-examples_2.11-2.4.0-cdh6.2.0.jar 5
22/10/27 17:35:10 INFO spark.SparkContext: Running Spark version 2.4.0-cdh6.2.0
22/10/27 17:35:10 INFO logging.DriverLogger: Added a local log appender at: /tmp/spark-92314d20-bc56-43a2-b682-9579169dacb5/_driver_logs/_driver.log
22/10/27 17:35:10 INFO spark.SparkContext: Submitted application: Spark Pi
22/10/27 17:35:10 INFO spark.SecurityManager: Changing view acls to: vripollr
22/10/27 17:35:10 INFO spark.SecurityManager: Changing modify acls to: vripollr
22/10/27 17:35:10 INFO spark.SecurityManager: Changing view acls groups to:
22/10/27 17:35:10 INFO spark.SecurityManager: Changing modify acls groups to:
22/10/27 17:35:10 INFO spark.SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users with view permissions: Set(vripollr); groups with view permissions: Set(); user
$ with modify permissions: Set(vripollr); groups with modify permissions: Set()
22/10/27 17:35:10 INFO util.Utils: Successfully started service 'sparkDriver' on port 37090.
22/10/27 17:35:10 INFO spark.SparkEnv: Registering MapOutputTracker
22/10/27 17:35:10 INFO spark.SparkEnv: Registering BlockManagerMaster
22/10/27 17:35:10 INFO storage.BlockManagerMasterEndpoint: Using org.apache.spark.storage.DefaultTopologyMapper for getting topology information
22/10/27 17:35:10 INFO storage.BlockManagerMasterEndpoint: BlockManagerMasterEndpoint up
22/10/27 17:35:10 INFO storage.DiskBlockManager: Created local directory at /tmp/blockmgr-b4a2fa31-0408-4065-a9ba-52faf8c06d30
22/10/27 17:35:10 INFO memory.MemoryStore: MemoryStore started with capacity 366.3 MB
22/10/27 17:35:10 INFO spark.SparkEnv: Registering OutputCommitCoordinator
22/10/27 17:35:10 INFO spark.SparkContext: Added JAR file:/opt/cloudera/parcels/CDH/lib/spark/examples/jars/spark-examples_2.11-2.4.0-cdh6.2.0.jar at spark://Cloudera01:37090/jars/spark-examp
ls_2.11-2.4.0-cdh6.2.0.jar with timestamp 1666884910911
22/10/27 17:35:10 INFO util.Utils: Using initial executors = 0, max of spark.dynamicAllocation.initialExecutors, spark.dynamicAllocation.minExecutors and spark.executor.instances
22/10/27 17:35:11 INFO client.RMProxy: Connecting to ResourceManager at eimtcld.uoc.edu/213.73.35.119:8032
22/10/27 17:35:11 INFO yarn.Client: Requesting a new application from cluster with 3 NodeManagers
22/10/27 17:35:11 INFO conf.Configuration: resource-types.xml not found
22/10/27 17:35:11 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
22/10/27 17:35:11 INFO yarn.Client: Verifying our application has not requested more than the maximum memory capability of the cluster (3783 MB per container)
22/10/27 17:35:11 INFO yarn.Client: Will allocate AM container, with 896 MB memory including 384 MB overhead
22/10/27 17:35:11 INFO yarn.Client: Setting up container launch context for our AM
22/10/27 17:35:11 INFO yarn.Client: Setting up the launch environment for our AM container
22/10/27 17:35:11 INFO yarn.Client: Preparing resources for our AM container
22/10/27 17:35:12 INFO yarn.Client: Uploading resource file:/tmp/spark-92314d20-bc56-43a2-b682-9579169dacb5/_spark_conf_2711888547024466365.zip -> hdfs://eimtcld.uoc.edu:8020/user/vripollr/.
sparkStaging/application_1662730125715_0159/_spark_conf_.zip
22/10/27 17:35:12 INFO spark.SecurityManager: Changing view acls to: vripollr
22/10/27 17:35:12 INFO spark.SecurityManager: Changing modify acls to: vripollr
22/10/27 17:35:12 INFO spark.SecurityManager: Changing view acls groups to:
22/10/27 17:35:12 INFO spark.SecurityManager: Changing modify acls groups to:
22/10/27 17:35:12 INFO yarn.Client: Submitting application application_1662730125715_0159 to ResourceManager
22/10/27 17:35:12 INFO impl.YarnClientImpl: Submitted application application_1662730125715_0159
22/10/27 17:35:13 INFO yarn.Client: Application report for application_1662730125715_0159 (state: ACCEPTED)
22/10/27 17:35:13 INFO yarn.Client:
client token: N/A
diagnostics: AM container is launched, waiting for AM container to Register with RM
ApplicationMaster host: N/A
ApplicationMaster RPC port: -1
queue: root.users.vripollr
start time: 1666884912534
final status: UNDEFINED
tracking URL: http://eimtcld.uoc.edu:8088/proxy/application_1662730125715_0159/
```

El código se puede consultar en <https://github.com/apache/spark/blob/master/examples/src/main/python/pi.py>.  
(<https://github.com/apache/spark/blob/master/examples/src/main/python/pi.py>).

```

18 import sys
19 from random import random
20 from operator import add
21
22 from pyspark.sql import SparkSession
23
24
25 if __name__ == "__main__":
26     """
27     Usage: pi [partitions]
28     """
29     spark = SparkSession\
30         .builder\
31         .appName("PythonPi")\
32         .getOrCreate()
33
34     partitions = int(sys.argv[1]) if len(sys.argv) > 1 else 2
35     n = 100000 * partitions
36
37     def f(_: int) -> float:
38         x = random() * 2 - 1
39         y = random() * 2 - 1
40         return 1 if x ** 2 + y ** 2 <= 1 else 0
41
42     count = spark.sparkContext.parallelize(range(1, n + 1), partitions).map(f).reduce(add)
43     print("Pi is roughly %f" % (4.0 * count / n))
44
45     spark.stop()

```

---

¿Qué parámetros recibe para su ejecución y para que sirven?

Recibe el número de puntos que se desean generar dentro del cuadrado. Este número sirve para estimar el número pi mediante el procedimiento que se ha descrito. Cuanto mayor sea el número, la aproximación será más exacta.

## 2.2 Ejecución (1 puntos)

Revisad la documentación de Cloudera [[Running Spark applications on YARN](https://docs.cloudera.com/runtime/7.2.1/running-spark-applications/topics/spark-running-spark-apps-on-yarn.html)] (<https://docs.cloudera.com/runtime/7.2.1/running-spark-applications/topics/spark-running-spark-apps-on-yarn.html>) y ejecutad el programa usando Yarn en modo cliente, con una memoria por ejecutor de 1 gigabyte y configurando el programa SparkPi para usar 10 particiones. Debéis ejecutar el programa mediante una terminal (logueados vía ssh o directamente con el terminal de Jupyter), mediante el comando spark-submit.

Una vez ejecutado, el resultado esperado debe ser parecido al siguiente:



```

20/09/19 17:31:54 INFO scheduler.TaskSetManager: Starting task 7.0 in stage 0.0 (TID 7, Cloudera01, executor
20/09/19 17:31:54 INFO scheduler.TaskSetManager: Finished task 6.0 in stage 0.0 (TID 6) in 51 ms on Cloudera0
20/09/19 17:31:54 INFO scheduler.TaskSetManager: Starting task 8.0 in stage 0.0 (TID 8, Cloudera01, executor
20/09/19 17:31:54 INFO scheduler.TaskSetManager: Finished task 7.0 in stage 0.0 (TID 7) in 53 ms on Cloudera0
20/09/19 17:31:54 INFO scheduler.TaskSetManager: Starting task 9.0 in stage 0.0 (TID 9, Cloudera01, executor
20/09/19 17:31:54 INFO scheduler.TaskSetManager: Finished task 8.0 in stage 0.0 (TID 8) in 307 ms on Cloudera
20/09/19 17:31:54 INFO scheduler.TaskSetManager: Finished task 9.0 in stage 0.0 (TID 9) in 90 ms on Cloudera0
20/09/19 17:31:54 INFO cluster.YarnScheduler: Removed TaskSet 0.0, whose tasks have all completed, from pool
20/09/19 17:31:54 INFO scheduler.DAGScheduler: ResultStage 0 (reduce at SparkPi.scala:38) finished in 4,997 s
20/09/19 17:31:54 INFO scheduler.DAGScheduler: Job 0 finished: reduce at SparkPi.scala:38, took 5,078599 s
Pi is roughly 3.14017514017514
20/09/19 17:31:54 INFO cluster.YarnClientSchedulerBackend: Interrupting monitor thread
20/09/19 17:31:54 INFO cluster.YarnClientSchedulerBackend: Shutting down all executors
20/09/19 17:31:54 INFO cluster.YarnSchedulerBackend$YarnDriverEndpoint: Asking each executor to shut down
20/09/19 17:31:54 INFO cluster.SchedulerExtensionServices: Stopping SchedulerExtensionServices
(serviceOption=None,
 services=List(),
 started=false)
20/09/19 17:31:54 INFO cluster.YarnClientSchedulerBackend: Stopped
20/09/19 17:31:54 INFO spark.MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
20/09/19 17:31:54 INFO memory.MemoryStore: MemoryStore cleared
20/09/19 17:31:54 INFO storage.BlockManager: BlockManager stopped
20/09/19 17:31:54 INFO storage.BlockManagerMaster: BlockManagerMaster stopped

```

spark-submit --class org.apache.spark.examples.SparkPi --master yarn --executor-memory 1G --deploy-mode client /opt/cloudera/parcels/CDH/lib/spark/examples/jars/spark-examples\_2.11-2.4.0-cdh6.2.0.jar 10

vrpollr@Cloudera01: ~

```

22/10/27 18:00:04 INFO scheduler.TaskSetManager: Finished task 1.0 in stage 0.0 (TID 1) in 2004 ms on Cloudera
22/10/27 18:00:04 INFO scheduler.TaskSetManager: Starting task 8.0 in stage 0.0 (TID 8, Cloudera01, executor 1
22/10/27 18:00:04 INFO scheduler.TaskSetManager: Finished task 0.0 in stage 0.0 (TID 0) in 2143 ms on Cloudera
22/10/27 18:00:04 INFO scheduler.TaskSetManager: Starting task 9.0 in stage 0.0 (TID 9, Cloudera01, executor 3
22/10/27 18:00:04 INFO storage.BlockManagerMasterEndpoint: Registering block manager Cloudera01:45846 with 408
22/10/27 18:00:04 INFO scheduler.TaskSetManager: Finished task 2.0 in stage 0.0 (TID 2) in 2032 ms on Cloudera
22/10/27 18:00:04 INFO scheduler.TaskSetManager: Finished task 7.0 in stage 0.0 (TID 7) in 166 ms on Cloudera0
22/10/27 18:00:04 INFO scheduler.TaskSetManager: Finished task 9.0 in stage 0.0 (TID 9) in 129 ms on Cloudera0
22/10/27 18:00:04 INFO scheduler.TaskSetManager: Finished task 8.0 in stage 0.0 (TID 8) in 202 ms on Cloudera0
22/10/27 18:00:04 INFO storage.BlockManagerMasterEndpoint: Registering block manager Cloudera01:39339 with 408
22/10/27 18:00:04 INFO storage.BlockManagerMasterEndpoint: Registering block manager Cloudera01:42593 with 408
22/10/27 18:00:05 INFO cluster.YarnSchedulerBackend$YarnDriverEndpoint: Registered executor NettyRpcEndpointRe
22/10/27 18:00:05 INFO spark.ExecutorAllocationManager: New executor 9 has registered (new total is 8)
22/10/27 18:00:05 INFO storage.BlockManagerInfo: Added broadcast_0_piece0 in memory on Cloudera01:39708 (size:
22/10/27 18:00:05 INFO storage.BlockManagerMasterEndpoint: Registering block manager Cloudera01:35632 with 408
22/10/27 18:00:05 INFO scheduler.TaskSetManager: Finished task 3.0 in stage 0.0 (TID 3) in 1524 ms on Cloudera
22/10/27 18:00:05 INFO storage.BlockManagerInfo: Added broadcast_0_piece0 in memory on Cloudera01:45846 (size:
22/10/27 18:00:05 INFO storage.BlockManagerInfo: Added broadcast_0_piece0 in memory on Cloudera01:39339 (size:
22/10/27 18:00:05 INFO cluster.YarnSchedulerBackend$YarnDriverEndpoint: Registered executor NettyRpcEndpointRe
22/10/27 18:00:05 INFO spark.ExecutorAllocationManager: New executor 8 has registered (new total is 9)
22/10/27 18:00:05 INFO storage.BlockManagerInfo: Added broadcast_0_piece0 in memory on Cloudera01:42593 (size:
22/10/27 18:00:05 INFO cluster.YarnSchedulerBackend$YarnDriverEndpoint: Registered executor NettyRpcEndpointRe
22/10/27 18:00:05 INFO spark.ExecutorAllocationManager: New executor 10 has registered (new total is 10)
22/10/27 18:00:05 INFO scheduler.TaskSetManager: Finished task 4.0 in stage 0.0 (TID 4) in 1675 ms on Cloudera
22/10/27 18:00:05 INFO storage.BlockManagerMasterEndpoint: Registering block manager Cloudera01:42241 with 408
22/10/27 18:00:05 INFO scheduler.TaskSetManager: Finished task 5.0 in stage 0.0 (TID 5) in 1621 ms on Cloudera
22/10/27 18:00:06 INFO scheduler.TaskSetManager: Finished task 6.0 in stage 0.0 (TID 6) in 1592 ms on Cloudera
22/10/27 18:00:06 INFO cluster.YarnScheduler: Removed TaskSet 0.0, whose tasks have all completed, from pool
22/10/27 18:00:06 INFO scheduler.DAGScheduler: ResultStage 0 (reduce at SparkPi.scala:38) finished in 8,521 s
22/10/27 18:00:06 INFO scheduler.DAGScheduler: Job 0 finished: reduce at SparkPi.scala:38, took 8,579657 s
22/10/27 18:00:06 INFO storage.BlockManagerMasterEndpoint: Registering block manager Cloudera01:43295 with 408
Pi is roughly 3.1404311404311405
22/10/27 18:00:06 INFO cluster.YarnClientSchedulerBackend: Interrupting monitor thread
22/10/27 18:00:06 INFO cluster.YarnClientSchedulerBackend: Shutting down all executors
22/10/27 18:00:06 INFO cluster.YarnSchedulerBackend$YarnDriverEndpoint: Asking each executor to shut down
22/10/27 18:00:06 INFO cluster.SchedulerExtensionServices: Stopping SchedulerExtensionServices
(serviceOption=None,
 services=List(),
 started=false)
22/10/27 18:00:06 INFO cluster.YarnClientSchedulerBackend: Stopped
22/10/27 18:00:06 INFO spark.MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
22/10/27 18:00:06 INFO memory.MemoryStore: MemoryStore cleared
22/10/27 18:00:06 INFO storage.BlockManager: BlockManager stopped
22/10/27 18:00:06 INFO storage.BlockManagerMaster: BlockManagerMaster stopped

```

## 2.3 Modos de ejecución (1 puntos)

¿Qué diferencia hay entre ejecutar una aplicación de Spark en YARN en modo cliente o modo clúster? ¿En que casos utilizarías cada uno de estos modos?

Tal y como se explica en <https://spark.apache.org/docs/1.6.2/running-on-yarn.html#configuration> (<https://spark.apache.org/docs/1.6.2/running-on-yarn.html#configuration>), en el modo clúster el driver de Spark se ejecuta en el application master, mientras que en el modo cliente el driver se ejecuta en el proceso del cliente. En otras palabras, en modo cliente cuando un usuario envía un spark submit (formado por el driver program y los executor program), el driver program se ejecuta en el edge node ocupando memoria. En el modo cluster, el driver aunque es enviado al edge node, se envía a las máquinas del clúster para ser ejecutado en función de los recursos disponibles. Para trabajar en modo producción es preferible el modo clúster porque incluso si la cuenta con que se trabaja se desactiva el programa se seguirá ejecutado. En el modo cliente esto no sucede. Así, se prefiere el modo cliente en tareas menos importantes como las de testeo. Como consecuencia del funcionamiento del modo cliente, una desventaja importante de este es que si hay varias personas enviando tareas de Spark al edge node, todos los drivers correspondientes a estas tareas serán ejecutados en este mismo edge node lo que puede implicar una falta de recursos para algunos de estos drivers. En el modo clúster, los drivers no se ejecutan en el edge node sino que es el edge node el que gestiona donde tienen que ejecutarse estos drivers. Esta gestión permite distribuir los recursos de una forma más eficiente. De nuevo, en tareas de producción es preferible el modo clúster debido a que habrá menos posibilidades de encontrarse con problemas de memoria.

### 3 Spark History Server (5 puntos)

En este bloque de preguntas vamos a trabajar con la herramienta de monitorización que ofrece Spark. Entrar en la Web UI del Spark history Server [[Monitoring Spark Applications](https://docs.cloudera.com/documentation/enterprise/6/6.2/topics/operation_spark_applications.html)] ([https://docs.cloudera.com/documentation/enterprise/6/6.2/topics/operation\\_spark\\_applications.html](https://docs.cloudera.com/documentation/enterprise/6/6.2/topics/operation_spark_applications.html)) habilitando primero un túnel SSH:

`http://localhost:18088`

y contestar a las siguientes preguntas (todas las respuestas se deben acompañar de evidencias con capturas de pantalla). Revisad con cuidado la información que os da el Spark History Server ya que no solo aparecen vuestros programas.

#### 3.1 ¿Cuál ha sido el identificador de aplicación que nos ha asignado YARN cuando hemos ejecutado nuestro programa? (0.5 punto)

`application_1662730125715_0167`

← → ↻ ⓘ localhost:18088

**History Server**  
2.4.0-cdh6.2.0

Event log directory: hdfs://eimtcld.uoc.edu:8020/user/spark/applicationHistory  
Last updated: 2022-10-27 18:21:27  
Client local time zone: Europe/Madrid

Show  entries

Search:

App ID	App Name	Started	Completed	Duration	Spark User	Last Updated	Event Log
application_1662730125715_0168	Spark Pi	2022-10-27 18:17:45	2022-10-27 18:17:57	12 s	gromerof1974	2022-10-27 18:17:57	<a href="#">Download</a>
local-1666876605761	dpadillas	2022-10-27 15:16:45	2022-10-27 18:10:06	2.9 h	dpadillas	2022-10-27 18:10:06	<a href="#">Download</a>
local-1666865473386	cruiz82	2022-10-27 12:11:12	2022-10-27 18:10:05	6.0 h	cruiz82	2022-10-27 18:10:05	<a href="#">Download</a>
application_1662730125715_0167	Spark Pi	2022-10-27 17:59:51	2022-10-27 18:00:06	15 s	vrpollr	2022-10-27 18:00:06	<a href="#">Download</a>
application_1662730125715_0166	Spark Pi	2022-10-27 17:40:27	2022-10-27 17:41:23	56 s	fpintoh	2022-10-27 17:41:23	<a href="#">Download</a>
application_1662730125715_0164	Spark Pi	2022-10-27 17:39:20	2022-10-27 17:39:40	20 s	vrpollr	2022-10-27 17:39:40	<a href="#">Download</a>
application_1662730125715_0162	Spark Pi	2022-10-27 17:37:34	2022-10-27 17:37:46	12 s	vrpollr	2022-10-27 17:37:46	<a href="#">Download</a>
application_1662730125715_0161	Spark Pi	2022-10-27 17:36:55	2022-10-27 17:37:22	27 s	fpintoh	2022-10-27 17:37:22	<a href="#">Download</a>
application_1662730125715_0160	Spark Pi	2022-10-27 17:35:25	2022-10-27 17:36:03	38 s	fpintoh	2022-10-27 17:36:03	<a href="#">Download</a>
application_1662730125715_0159	Spark Pi	2022-10-27 17:35:10	2022-10-27 17:35:26	16 s	vrpollr	2022-10-27 17:35:26	<a href="#">Download</a>
application_1662730125715_0158	Spark Pi	2022-10-27 17:32:46	2022-10-27 17:33:25	38 s	fpintoh	2022-10-27 17:33:25	<a href="#">Download</a>
application_1662730125715_0157	Spark Pi	2022-10-27 17:28:02	2022-10-27 17:28:13	12 s	svila0	2022-10-27 17:28:14	<a href="#">Download</a>
application_1662730125715_0156	Spark Pi	2022-10-27 17:22:58	2022-10-27 17:23:39	40 s	fpintoh	2022-10-27 17:23:39	<a href="#">Download</a>

### 3.2 ¿Qué tiempo ha durado la ejecución? (0.5 punto)

15 segundos

← → ↻ ⓘ localhost:18088

**History Server**  
2.4.0-cdh6.2.0

Event log directory: hdfs://eimtcld.uoc.edu:8020/user/spark/applicationHistory  
Last updated: 2022-10-27 18:21:27  
Client local time zone: Europe/Madrid

Show  entries

Search:

App ID	App Name	Started	Completed	Duration	Spark User	Last Updated	Event Log
application_1662730125715_0168	Spark Pi	2022-10-27 18:17:45	2022-10-27 18:17:57	12 s	gromerof1974	2022-10-27 18:17:57	<a href="#">Download</a>
local-1666876605761	dpadillas	2022-10-27 15:16:45	2022-10-27 18:10:06	2.9 h	dpadillas	2022-10-27 18:10:06	<a href="#">Download</a>
local-1666865473386	cruiz82	2022-10-27 12:11:12	2022-10-27 18:10:05	6.0 h	cruiz82	2022-10-27 18:10:05	<a href="#">Download</a>
application_1662730125715_0167	Spark Pi	2022-10-27 17:59:51	2022-10-27 18:00:06	15 s	vrpollr	2022-10-27 18:00:06	<a href="#">Download</a>
application_1662730125715_0166	Spark Pi	2022-10-27 17:40:27	2022-10-27 17:41:23	56 s	fpintoh	2022-10-27 17:41:23	<a href="#">Download</a>
application_1662730125715_0164	Spark Pi	2022-10-27 17:39:20	2022-10-27 17:39:40	20 s	vrpollr	2022-10-27 17:39:40	<a href="#">Download</a>
application_1662730125715_0162	Spark Pi	2022-10-27 17:37:34	2022-10-27 17:37:46	12 s	vrpollr	2022-10-27 17:37:46	<a href="#">Download</a>
application_1662730125715_0161	Spark Pi	2022-10-27 17:36:55	2022-10-27 17:37:22	27 s	fpintoh	2022-10-27 17:37:22	<a href="#">Download</a>
application_1662730125715_0160	Spark Pi	2022-10-27 17:35:25	2022-10-27 17:36:03	38 s	fpintoh	2022-10-27 17:36:03	<a href="#">Download</a>
application_1662730125715_0159	Spark Pi	2022-10-27 17:35:10	2022-10-27 17:35:26	16 s	vrpollr	2022-10-27 17:35:26	<a href="#">Download</a>
application_1662730125715_0158	Spark Pi	2022-10-27 17:32:46	2022-10-27 17:33:25	38 s	fpintoh	2022-10-27 17:33:25	<a href="#">Download</a>
application_1662730125715_0157	Spark Pi	2022-10-27 17:28:02	2022-10-27 17:28:13	12 s	svila0	2022-10-27 17:28:14	<a href="#">Download</a>
application_1662730125715_0156	Spark Pi	2022-10-27 17:22:58	2022-10-27 17:23:39	40 s	fpintoh	2022-10-27 17:23:39	<a href="#">Download</a>

### 3.3 ¿Cuántos executors han intervenido durante nuestra ejecución y en que nodo se han levantado? (0.5 punto)

Si se ejecuta el siguiente código: `spark-submit --class org.apache.spark.examples.SparkPi --master yarn --executor-memory 1G --deploy-mode client /opt/cloudera/parcels/CDH/lib/spark/examples/jars/spark-examples_2.11-2.4.0-cdh6.2.0.jar 10` Se obtiene:



Se observa que se han creado 10, pero en las 10 tasks que se querían aplicar han intervenido 5: executors 1, 2, 5, 6 y 7. El executor 1 se ha levantado en el nodo Cloudera 03 El executor 2, en Cloudera 03 El executor 5, en Cloudera 01 El executor 6, en Cloudera 01 El executor 7, en Cloudera 01

Se ejecutan 10. Se observa que el TaskSetManager gestiona el inicio y el fin de 10 tasks: task 0.0, 1.0,..., 9.0.

## punto)

Si se consulta el listado de transformaciones en <https://spark.apache.org/docs/latest/rdd-programming-guide.html#transformations> (<https://spark.apache.org/docs/latest/rdd-programming-guide.html#transformations>) y se utiliza el comando ctrl+f dentro del terminal para buscar coincidencias, se encuentran únicamente transformaciones de tipo map():

The screenshot shows a terminal window with Spark logs. A search dialog box is open over the logs, with the word "map" entered in the search field. The dialog box has a "Buscar" (Search) button, a "Buscar siguiente" (Search next) button, and a "Cancelar" (Cancel) button. The logs show various Spark events, including "MapPartitionsRDD[1] at map at SparkPi.scala:34", which is highlighted in red in the original image.

## 3.6 ¿Cuántas acciones se han invocado durante nuestro programa? (1 punto)

Si se consulta el listado de acciones en <https://spark.apache.org/docs/latest/rdd-programming-guide.html#actions> (<https://spark.apache.org/docs/latest/rdd-programming-guide.html#actions>) y se utiliza el comando ctrl+f dentro del terminal para buscar coincidencias, se encuentra únicamente la acción reduce():





Como ya sabéis, YARN es el encargado de manejar los recursos de los nodos que forman el clúster de manera global y es él quien tiene la responsabilidad de distribuir las aplicaciones para que se ejecuten de manera paralelizada y balancear la carga entre ellas.

Una vez enviada una aplicación a YARN en modo cluster, esta se ejecuta bajo su control, por eso, si queremos cancelar una aplicación, no basta con hacer un `ctrl+c` sino que debemos indicárselo a YARN.

Para esta tarea y muchas otras, tenéis disponible el comando `yarn` en la terminal [\[YARN Commands\]](https://hadoop.apache.org/docs/stable/hadoop-yarn/hadoop-yarn-site/YarnCommands.html) (<https://hadoop.apache.org/docs/stable/hadoop-yarn/hadoop-yarn-site/YarnCommands.html>). En este ejercicio vamos a ver algunas de las posibilidades que ofrece.

Todas las respuestas a las preguntas siguientes deben acompañarse de evidencias con capturas de pantalla.

Para este ejercicio, vamos a abrir dos terminales dentro del JupyterLab, uno en cada pestaña. En el primero, volveremos a ejecutar la aplicación SparkPi en modo clúster incrementando el número de muestras de 10 a 10000.

En el segundo, y mientras SparkPi se está ejecutando, debéis consultar las aplicaciones que YARN está ejecutando mediante el comando `yarn`.

¿Cuál es la ID que se ha asignado (1.25 puntos)?

En la primer pestaña se ha aplicado el comando `spark-submit --class org.apache.spark.examples.SparkPi --master yarn --executor-memory 1G --deploy-mode cluster /opt/cloudera/parcels/CDH/lib/spark/examples/jars/spark-examples_2.11-2.4.0-cdh6.2.0.jar 10000` En la segunda, mientras se ejecutaba el anterior, se ha aplicado el comando `yarn application -list -appStates RUNNING`

```

22/10/29 19:56:29 INFO yarn.Client: Application report for application_1
662730125715_0705 (state: RUNNING)
22/10/29 19:56:30 INFO yarn.Client: Application report for application_1
662730125715_0705 (state: RUNNING)
22/10/29 19:56:31 INFO yarn.Client: Application report for application_1
662730125715_0705 (state: RUNNING)
22/10/29 19:56:32 INFO yarn.Client: Application report for application_1
662730125715_0705 (state: RUNNING)
22/10/29 19:56:33 INFO yarn.Client: Application report for application_1
662730125715_0705 (state: RUNNING)
22/10/29 19:56:34 INFO yarn.Client: Application report for application_1
662730125715_0705 (state: RUNNING)
22/10/29 19:56:35 INFO yarn.Client: Application report for application_1
662730125715_0705 (state: RUNNING)
22/10/29 19:56:36 INFO yarn.Client: Application report for application_1
662730125715_0705 (state: FINISHED)
22/10/29 19:56:36 INFO yarn.Client:
client token: N/A
diagnostics: N/A
ApplicationMaster host: Cloudera01
ApplicationMaster RPC port: 37970
queue: root.users.vripollr
start time: 1667066162306
final status: SUCCEEDED
tracking URL: http://eimtcld.uoc.edu:8088/proxy/application_166
2730125715_0705/
user: vripollr
22/10/29 19:56:36 INFO util.ShutdownHookManager: Shutdown hook called
22/10/29 19:56:36 INFO util.ShutdownHookManager: Deleting directory /tmp
/spark-f3e689d3-6765-4a8a-b0e4-b7abeea18013
22/10/29 19:56:36 INFO util.ShutdownHookManager: Deleting directory /tmp
/spark-aa69acal-3086-4d8f-a027-e05bc2aece8a
vripollr@Cloudera01:~$

ApplicationId can be passed using 'appI
d'
option.
vripollr@Cloudera01:~$ yarn application -list ^CappStates RUNNING
vripollr@Cloudera01:~$ yarn application -list -appStates RUNNING
WARNING: YARN_OPTS has been replaced by HADOOP_OPTS. Using value of YARN
_OPTS.
22/10/29 19:53:23 INFO client.RMProxy: Connecting to ResourceManager at
eimtcld.uoc.edu/213.73.35.119:8032
Total number of applications (application-types: [], states: [RUNNING]) a
nd tags: [{}]:1
Application-Id      Application-Name      Application-
Type              User              Queue              State              F
inal-State          Progress            Tracking-URL
application_1662730125715_0702      Spark Pi              S
PARK              vripollr      root.users.vripollr      RUNNING
UNDEFINED              10% http://eimtcld.uoc.edu:18088/his
tory/application_1662730125715_0702/1
vripollr@Cloudera01:~$ yarn application -list -appStates RUNNING
WARNING: YARN_OPTS has been replaced by HADOOP_OPTS. Using value of YARN
_OPTS.
22/10/29 19:56:11 INFO client.RMProxy: Connecting to ResourceManager at
eimtcld.uoc.edu/213.73.35.119:8032
Total number of applications (application-types: [], states: [RUNNING]) a
nd tags: [{}]:1
Application-Id      Application-Name      Application-
Type              User              Queue              State              F
inal-State          Progress            Tracking-URL
application_1662730125715_0705      org.apache.spark.examples.SparkPi
SPARK              vripollr      root.users.vripollr
RUNNING              UNDEFINED              10% http://eimtcld.u
oc.edu:18088/history/application_1662730125715_0705/1
vripollr@Cloudera01:~$

```

El ID asignado es `application_1662730125715_0705`

Una vez obtengáis la ID, mientras nuestra aplicación esta todavía en ejecución (tarda 40 segundos en ejecutarse), utilizad el comando `yarn` para parar su ejecución.

¿Qué comando habéis ejecutado (1.25 puntos)?

Para parar la aplicación se puede usar el comando `yarn application -kill <ID de la aplicación>`. Nótese que en cada `spark-submit` la ID cambia, en la siguiente imagen primero se ha tenido que aplicar el comando `yarn`



application -list -appStates RUNNING para conocer la ID actual y luego finalizarla con yarn application -kill application\_1662730125715\_0712:

jupyterLogoutControl Panel

```

)
    at org.apache.hadoop.hdfs.DFSClient.getFileInfo(DFSClient.java:1
622)
    at org.apache.hadoop.hdfs.DistributedFileSystem$29.doCall(Distri
butedFileSystem.java:1495)
    at org.apache.hadoop.hdfs.DistributedFileSystem$29.doCall(Distri
butedFileSystem.java:1492)
    at org.apache.hadoop.fs.FileSystemLinkResolver.resolve(FileSyste
mLinkResolver.java:81)
    at org.apache.hadoop.hdfs.DistributedFileSystem.getFileStatus(Di
stributedFileSystem.java:1507)
    at org.apache.hadoop.fs.FileSystem.exists(FileSystem.java:1668)
    at org.apache.spark.scheduler.EventLoggingListener.stop(EventLog
gingListener.scala:295)
    at org.apache.spark.SparkContext$$anonfun$stop$9$$anonfun$apply$
mcV$sp$7.apply(SparkContext.scala:1974)
    at org.apache.spark.SparkContext$$anonfun$stop$9$$anonfun$apply$
mcV$sp$7.apply(SparkContext.scala:1974)
    at scala.Option.foreach(Option.scala:257)
    at org.apache.spark.SparkContext$$anonfun$stop$9.apply$mcV$sp$(Sp
arkContext.scala:1974)
    at org.apache.spark.util.Utils$.tryLogNonFatalError(Utils.scala:
1343)
    at org.apache.spark.SparkContext.stop(SparkContext.scala:1973)
    at org.apache.spark.scheduler.cluster.YarnClientSchedulerBackend
$MonitorThread.run(YarnClientSchedulerBackend.scala:122)
22/10/29 20:17:31 INFO spark.MapOutputTrackerMasterEndpoint: MapOutputTr
ackerMasterEndpoint stopped!
22/10/29 20:17:31 INFO memory.MemoryStore: MemoryStore cleared
22/10/29 20:17:31 INFO storage.BlockManager: BlockManager stopped
22/10/29 20:17:31 INFO storage.BlockManagerMaster: BlockManagerMaster st
opped
vripollr@Cloudera01:~$

```

jupyterLogoutControl Panel

```

nd tags: []):1
Application-Id      Application-Name      Application-
Type      User      Queue      State      Application-
inal-State      Progress      Tracking-URL      F
application_1662730125715_0711      Spark Pi      S
PARK      vripollr      root.users.vripollr      RUNNING
UNDEFINED      10% http://eimtcld.uoc.edu:18088/his
tory/application_1662730125715_0711/1
vripollr@Cloudera01:~$ yarn application -kill application ^C
vripollr@Cloudera01:~$ yarn application -list -appStates RUNNING
WARNING: YARN_OPTS has been replaced by HADOOP_OPTS. Using value of YARN
_OPTS.
22/10/29 20:17:17 INFO client.RMPProxy: Connecting to ResourceManager at
eimtcld.uoc.edu/213.73.35.119:8032
Total number of applications (application-types: [], states: [RUNNING]) a
nd tags: []):1
Application-Id      Application-Name      Application-
Type      User      Queue      State      Application-
inal-State      Progress      Tracking-URL      F
application_1662730125715_0712      Spark Pi      S
PARK      vripollr      root.users.vripollr      RUNNING
UNDEFINED      0% http://eimtcld.uoc.edu:18088/his
tory/application_1662730125715_0712/1
vripollr@Cloudera01:~$ yarn application -kill application_1662730125715_
0712
WARNING: YARN_OPTS has been replaced by HADOOP_OPTS. Using value of YARN
_OPTS.
22/10/29 20:17:30 INFO client.RMPProxy: Connecting to ResourceManager at
eimtcld.uoc.edu/213.73.35.119:8032
Killing application application_1662730125715_0712
22/10/29 20:17:30 INFO impl.YarnClientImpl: Killed application applicati
on_1662730125715_0712
vripollr@Cloudera01:~$

```