Managing Swarm clusters on AWS

- Docker Machine
- Set-Up instructions
    - Link to code
    - Starting Consul
    - Starting Swarm Master
    - Joining workers
    - Checking the running instances
    - Removing instances
- Spark Application
    - Stack
    - Set up environment
    - Deploy application
    - Listing stack details
    - Listing running containers
    - Run Spark Application (Spark Pi):
        - Console output
    -

Docker Machine

Docker Machine Overview

To provision hosts in the cloud, drivers are available for various cloud providers including AWS: Provisioning in the cloud

AWS amazonec2 driver specific settings can be found here: AWS driver

Among a myriad of choices to run containers on AWS, docker-machine seems reasonable choice to explore and experiment with, especially in the prototyping, dev/test set up.The true benefit of docker-machine in this set up is installing Docker Engine on remote node and the ability to communicate seamlessly with the remote cluster with our local Docker client. This is a powerful set up for flexibility of experimentation.

Set-Up instructions

Docker needs certain ports to be open so that the containers can communicate. To have a separate security group for the swarm cluster, make a security group via AWS console. Then create a dummy machine with setting all the ports and then delete it as shown in the following script

```
docker-machine create --driver amazonec2 \
        --amazonec2-security-group $MY_SECURITY_GROUP \
        --amazonec2-access-key $MY_ACCESS_KEY \
        --amazonec2-secret-key $MY_SECRET_ACCESS_KEY \
        --amazonec2-region eu-central-1 \
        --amazonec2-vpc-id $MY_VPC_ID \
        --amazonec2-open-port 2377 \
        --amazonec2-open-port 7946 \
        --amazonec2-open-port 4789 \
        --amazonec2-open-port 7946/udp --amazonec2-open-port 4789/udp \
        --amazonec2-open-port 8080 --amazonec2-open-port 80 \
        --amazonec2-open-port 8300 --amazonec2-open-port 8301 \
        --amazonec2-open-port 8302 --amazonec2-open-port 8400 \
        --amazonec2-open-port 8500 --amazonec2-open-port 8600 \
        --amazonec2-open-port 443 ports


docker-machine rm --force ports
```

The following are the most important ports: (Borrowed from a concise write-up here: Docker Swarm Ports)

- TCP port 2377 for cluster management & raft sync communications
- TCP and UDP port 7946 for "control plane" gossip discovery communication
- UDP port 4789 for "data plane" VXLAN overlay network traffic

With some scripting and parsing json for aws describe-instances, the process to set up a docker swarm (manager and workers) is running the following scripts:

Link to code

Mercateo: swarming_spark_aws on Bitbucket

Github: swarming_spark_aws on GitHub vandatud account

Starting Consul

Using the script start_consul.sh you can start an EC2 instance with consul set up for the cluster. The following code snippet shows the console output.

```
$ ./start_consul.sh ~/path/to/aws_credentials
Running pre-create checks...
Creating machine...
(c1consul) Launching instance...
Waiting for machine to be running, this may take a few minutes...
Detecting operating system of created instance...
Waiting for SSH to be available...
Detecting the provisioner...
Provisioning with ubuntu(systemd)...
Installing Docker...
Copying certs to the local machine directory...
Copying certs to the remote machine...
Setting Docker configuration on the remote daemon...
Checking connection to Docker...
Docker is up and running!
To see how to connect your Docker Client to the Docker Engine running on
this virtual machine, run: docker-machine env c1consul
Unable to find image 'progrium/consul:latest' locally
latest: Pulling from progrium/consul
c862d82a67a2: Pull complete
0e7f3c08384e: Pull complete
0e221e32327a: Pull complete
09a952464e47: Pull complete
60a1b927414d: Pull complete
4c9f46b5ccce: Pull complete
417d86672aa4: Pull complete
b0d47ad24447: Pull complete
fd5300bd53f0: Pull complete
a3ed95caeb02: Pull complete
d023b445076e: Pull complete
ba8851f89e33: Pull complete
5d1cefca2a28: Pull complete
Digest:
sha256:8cc8023462905929df9a79ff67ee435a36848ce7a10f18d6d0faba9306b97274
Status: Downloaded newer image for progrium/consul:latest
0147d387b812373dc5702eb4de4dd64338e2377f8fbc9314c138c36f8a3f7558
```

Starting Swarm Master

Start the swarm manager using the start_master.sh script, the console output would be as below

```
$ ./start_master.sh ~/path/to/aws_credentials
Running pre-create checks...
Creating machine...
(c1-master) Launching instance...
Waiting for machine to be running, this may take a few minutes...
Detecting operating system of created instance...
Waiting for SSH to be available...
Detecting the provisioner...
Provisioning with ubuntu(systemd)...
Installing Docker...
Copying certs to the local machine directory...
Copying certs to the remote machine...
Setting Docker configuration on the remote daemon...
Configuring swarm...
Checking connection to Docker...
Docker is up and running!
To see how to connect your Docker Client to the Docker Engine running on
this virtual machine, run: docker-machine env c1-master
Swarm initialized: current node (ja6zg76kqgbhs9f7gjax9vyj4) is now a
manager.
To add a worker to this swarm, run the following command:

    docker swarm join --token
SWMTKN-1-3w94lbs4j2j9srxbidatqhbgah9s5rawdttxls2datots16hp8-abam7xwjmxt541
l11mwzwxww4 172.31.15.194:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and
follow the instructions.
```

Joining workers

Join worker using the script start_workers.sh. It has currently 4 workers for demonstration. This can be added as a parameter to make number of workers configurable.

```
$ ./start_worker.sh ~/VANDA/swarming-spark-aws/aws_credentials
Got Swarm Token :
SWMTKN-1-3w94lbs4j2j9srxbidatqhbgah9s5rawdttxls2datots16hp8-abam7xwjmxt541
l11mwzwxww4
Running pre-create checks...
Running pre-create checks...
Running pre-create checks...
Running pre-create checks...
Creating machine...
Creating machine...
Creating machine...
Creating machine...
(c1-worker-2) Launching instance...
(c1-worker-4) Launching instance...
(c1-worker-1) Launching instance...
(c1-worker-3) Launching instance...
```

```
Waiting for machine to be running, this may take a few minutes...
Waiting for machine to be running, this may take a few minutes...
Waiting for machine to be running, this may take a few minutes...
Detecting operating system of created instance...
Waiting for SSH to be available...
Detecting operating system of created instance...
Waiting for SSH to be available...
Detecting operating system of created instance...
Waiting for SSH to be available...
Waiting for machine to be running, this may take a few minutes...
Detecting operating system of created instance...
Waiting for SSH to be available...
Detecting the provisioner...
Provisioning with ubuntu(systemd)...
Detecting the provisioner...
Detecting the provisioner...
Provisioning with ubuntu(systemd)...
Detecting the provisioner...
Provisioning with ubuntu(systemd)...
Provisioning with ubuntu(systemd)...
Installing Docker...
Installing Docker...
Installing Docker...
Installing Docker...
Copying certs to the local machine directory...
Copying certs to the remote machine...
Setting Docker configuration on the remote daemon...
Copying certs to the local machine directory...
Copying certs to the local machine directory...
Copying certs to the local machine directory...
Copying certs to the remote machine...
Configuring swarm...
Copying certs to the remote machine...
Copying certs to the remote machine...
Setting Docker configuration on the remote daemon...
Setting Docker configuration on the remote daemon...
Setting Docker configuration on the remote daemon...
Configuring swarm...
Configuring swarm...
Checking connection to Docker...
Docker is up and running!
To see how to connect your Docker Client to the Docker Engine running on
this virtual machine, run: docker-machine env c1-worker-1
Configuring swarm...
This node joined a swarm as a worker.
Checking connection to Docker...
Docker is up and running!
To see how to connect your Docker Client to the Docker Engine running on
this virtual machine, run: docker-machine env c1-worker-3
This node joined a swarm as a worker.
Checking connection to Docker...
Docker is up and running!
To see how to connect your Docker Client to the Docker Engine running on
```

```
this virtual machine, run: docker-machine env c1-worker-4
Checking connection to Docker...
Docker is up and running!
To see how to connect your Docker Client to the Docker Engine running on
```

```
this virtual machine, run: docker-machine env c1-worker-2
This node joined a swarm as a worker.
This node joined a swarm as a worker.
```

Checking the running instances

Use docker-machine ls to view all the docker swarm hosts that we just started on AWS

```
$ docker-machine ls
NAME          ACTIVE   DRIVER       STATE     URL
SWARM                  DOCKER        ERRORS
c1-master     -        amazonec2    Running   tcp://35.159.8.126:2376
c1-master (master)   v17.06.0-ce
c1-worker-1   -        amazonec2    Running   tcp://52.58.41.39:2376
c1-master            v17.06.0-ce
c1-worker-2   -        amazonec2    Running   tcp://35.158.50.16:2376
c1-master            v17.06.0-ce
c1-worker-3   -        amazonec2    Running   tcp://52.58.66.184:2376
c1-master            v17.06.0-ce
c1-worker-4   -        amazonec2    Running   tcp://35.158.196.135:2376
c1-master            v17.06.0-ce
c1consul      -        amazonec2    Running   tcp://35.158.192.194:2376
v17.06.0-ce
```

Removing instances

```
docker-machine rm -f <instance name>


docker-machine rm -f c1-master
docker-machine rm -f c1-worker-1
...and so on!
```

Spark Application

Stack

Stacks are a convenient way to deploy multiple services. Details of the concept: Service stacks

To create 'services' that our swarm infrastructure could run: a network, spark master, spark worker, and a ui proxy for Spark UI. This is typically done using a yaml file like the following. It is called a stack file and is used to deploy complex distributed applications in docker at scale, as multiple linked services.

```
```

docker-stack.yml

```
version: '3'

networks:
  spark:
    driver: overlay

services:
  master:
    image: gettyimages/spark:2.1.0-hadoop-2.7
    command: bin/spark-class org.apache.spark.deploy.master.Master -h
master
    hostname: master
    environment:
      MASTER: spark://master:7077
      SPARK_CONF_DIR: /spark/conf
      SPARK_PUBLIC_DNS: localhost
      SPARK_MASTER_WEBUI_PORT: 8080
    ports:
      - 4040:4040
      - 6066:6066
      - 7077:7077
      - 8080:8080
    depends_on:
      - default
    networks:
      - spark
    deploy:
        replicas: 1
        placement:
          constraints:
            - node.role == manager
        update_config:
            parallelism: 1
        restart_policy:
            condition: on-failure

worker:
    image: gettyimages/spark:2.1.0-hadoop-2.7
    command: bin/spark-class org.apache.spark.deploy.worker.Worker
spark://master:7077
    hostname: worker
    depends_on:
      - default
      - master
    environment:
      SPARK_CONF_DIR: /spark/conf
      SPARK_WORKER_CORES: 2
      SPARK_WORKER_MEMORY: 1g
      SPARK_WORKER_PORT: 8881
      SPARK_WORKER_WEBUI_PORT: 8081
      SPARK_PUBLIC_DNS: localhost
```

```
    networks:
      - spark
    ports:
      - 8081:8081
      - 8881:8881
    deploy:
        mode: global
        update_config:
            parallelism: 1
        restart_policy:
            condition: on-failure
spark-ui-proxy:
    image: pschatzmann/spark-ui-proxy
```

```
    command: master:8080 9006
    depends_on:
      - default
```

Set up environment

Run the following command to configure your shell.

```
eval $(docker-machine env c1-master)
```

Deploy application

Using the following command, the application is deployed to the swarm. The application name is my_spark_swarm. The stack file used is the one described above.

```
$ docker stack deploy --compose-file docker-stack.yml my_spark_swarm
Creating network my_spark_swarm_spark
Creating service my_spark_swarm_master
Creating service my_spark_swarm_worker
Creating service my_spark_swarm_spark-ui-proxy
```

Listing stack details

The docker stack commands can be normally used from the configured shell locally. All the containers are running remotely on AWS

```
$ docker stack ps my_spark_swarm
ID                  NAME
IMAGE                                 NODE                DESIRED STATE
CURRENT STATE              ERROR               PORTS
7023gab4nhx2        my_spark_swarm_worker.7r9woyef4s9dhrnrj9d3krkn2
gettyimages/spark:2.1.0-hadoop-2.7   c1-worker-2         Running
Preparing 25 seconds ago

ruvnke60r9qw        my_spark_swarm_worker.sig82uymtbtugckjhrksug3h8
gettyimages/spark:2.1.0-hadoop-2.7   c1-worker-4         Running
Preparing 25 seconds ago

az7dg1nbv8ao        my_spark_swarm_worker.v4gtl7u9oosyeov6141gj42jq
gettyimages/spark:2.1.0-hadoop-2.7   c1-worker-3         Running
Preparing 25 seconds ago

rivk6rn8m8ey        my_spark_swarm_worker.jiw19pssuetpgys13k1smam5w
gettyimages/spark:2.1.0-hadoop-2.7   c1-worker-1         Running
Preparing 25 seconds ago

swfu7lkd7bdm        my_spark_swarm_worker.ja6zg76kqgbhs9f7gjax9vyj4
gettyimages/spark:2.1.0-hadoop-2.7   c1-master           Running
Preparing 25 seconds ago

jskkpklqgfp5        my_spark_swarm_spark-ui-proxy.1
pschatzmann/spark-ui-proxy:latest    c1-master           Running
Preparing 22 seconds ago

c08dc127fj6a        my_spark_swarm_master.1
gettyimages/spark:2.1.0-hadoop-2.7   c1-master           Running
Preparing 27 seconds ago
```

Listing running containers

```
$ docker ps

CONTAINER ID        IMAGE                                  COMMAND
CREATED             STATUS              PORTS
NAMES

acde79a263bb        gettyimages/spark:2.1.0-hadoop-2.7   "bin/spark-class
o..."   About an hour ago   Up About an hour
my_spark_swarm_worker.ja6zg76kqgbhs9f7gjax9vyj4.swfu7lkd7bdm6o62sfjpfdwlb

3a97aaf05707        gettyimages/spark:2.1.0-hadoop-2.7   "bin/spark-class
o..."   About an hour ago   Up About an hour
my_spark_swarm_master.1.c08dc127fj6akarot0spt7rvg

853ba411ea2e        pschatzmann/spark-ui-proxy:latest    "python
/spark-ui-..."   About an hour ago   Up About an hour    80/tcp
my_spark_swarm_spark-ui-proxy.1.jskkpklqgfp57m65g68mtfqg1

e4cafe610c83        swarm:latest                         "/swarm join
--adv..."   2 hours ago         Up 2 hours          2375/tcp
swarm-agent

79e059304b62        swarm:latest                         "/swarm manage
--t..."   2 hours ago         Up 2 hours          2375/tcp,
0.0.0.0:4000->4000/tcp   swarm-agent-master
```

Run Spark Application (Spark Pi):

Spark Pi can now run on our swarming Spark installation with the follwing command: Note the value of the --net option, it is the name of the Spark master (See docker ps above)

```
docker run
--net=container:my_spark_swarm_master.1.c08dc127fj6akarot0spt7rvg
--entrypoint spark-submit gettyimages/spark:2.1.0-hadoop-2.7 --master
spark://master:7077  --class org.apache.spark.examples.SparkPi
/usr/spark-2.1.0/examples/jars/spark-examples_2.11-2.1.0.jar
```

Console output

```
docker run
--net=container:my_spark_swarm_master.1.c08dc127fj6akarot0spt7rvg
--entrypoint spark-submit gettyimages/spark:2.1.0-hadoop-2.7 --master
spark://master:7077  --class org.apache.spark.examples.SparkPi
/usr/spark-2.1.0/examples/jars/spark-examples_2.11-2.1.0.jar

17/08/16 09:09:51 INFO spark.SparkContext: Running Spark version 2.1.0
17/08/16 09:09:52 WARN util.NativeCodeLoader: Unable to load native-hadoop
```

```
library for your platform... using builtin-java classes where applicable
17/08/16 09:09:53 INFO spark.SecurityManager: Changing view acls to: root
17/08/16 09:09:53 INFO spark.SecurityManager: Changing modify acls to: root
17/08/16 09:09:53 INFO spark.SecurityManager: Changing view acls groups to:

17/08/16 09:09:53 INFO spark.SecurityManager: Changing modify acls groups
to:
17/08/16 09:09:53 INFO spark.SecurityManager: SecurityManager:
authentication disabled; ui acls disabled; users  with view permissions:
Set(root); groups with view permissions: Set(); users  with modify
permissions: Set(root); groups with modify permissions: Set()
17/08/16 09:09:54 INFO util.Utils: Successfully started service
'sparkDriver' on port 36787.
17/08/16 09:09:54 INFO spark.SparkEnv: Registering MapOutputTracker
17/08/16 09:09:54 INFO spark.SparkEnv: Registering BlockManagerMaster
17/08/16 09:09:54 INFO storage.BlockManagerMasterEndpoint: Using
org.apache.spark.storage.DefaultTopologyMapper for getting topology
information
17/08/16 09:09:54 INFO storage.BlockManagerMasterEndpoint:
BlockManagerMasterEndpoint up
17/08/16 09:09:54 INFO storage.DiskBlockManager: Created local directory at
/tmp/blockmgr-c274c8c9-f26b-4830-b169-701f42e1a5c2
17/08/16 09:09:54 INFO memory.MemoryStore: MemoryStore started with
capacity 413.9 MB
17/08/16 09:09:55 INFO spark.SparkEnv: Registering OutputCommitCoordinator
17/08/16 09:09:55 INFO util.log: Logging initialized @6772ms
17/08/16 09:09:55 INFO server.Server: jetty-9.2.z-SNAPSHOT
17/08/16 09:09:55 INFO handler.ContextHandler: Started
o.s.j.s.ServletContextHandler@7889a1ac{/jobs,null,AVAILABLE}
17/08/16 09:09:55 INFO handler.ContextHandler: Started
o.s.j.s.ServletContextHandler@3aee3976{/jobs/json,null,AVAILABLE}
17/08/16 09:09:55 INFO handler.ContextHandler: Started
o.s.j.s.ServletContextHandler@5ef8df1e{/jobs/job,null,AVAILABLE}
17/08/16 09:09:55 INFO handler.ContextHandler: Started
o.s.j.s.ServletContextHandler@27cf3151{/jobs/job/json,null,AVAILABLE}
17/08/16 09:09:55 INFO handler.ContextHandler: Started
o.s.j.s.ServletContextHandler@127e70c5{/stages,null,AVAILABLE}
17/08/16 09:09:55 INFO handler.ContextHandler: Started
o.s.j.s.ServletContextHandler@5910de75{/stages/json,null,AVAILABLE}
17/08/16 09:09:55 INFO handler.ContextHandler: Started
o.s.j.s.ServletContextHandler@4108fa66{/stages/stage,null,AVAILABLE}
17/08/16 09:09:55 INFO handler.ContextHandler: Started
o.s.j.s.ServletContextHandler@1f130eaf{/stages/stage/json,null,AVAILABLE}
17/08/16 09:09:55 INFO handler.ContextHandler: Started
o.s.j.s.ServletContextHandler@7e0aadd0{/stages/pool,null,AVAILABLE}
17/08/16 09:09:55 INFO handler.ContextHandler: Started
o.s.j.s.ServletContextHandler@21362712{/stages/pool/json,null,AVAILABLE}
17/08/16 09:09:55 INFO handler.ContextHandler: Started
o.s.j.s.ServletContextHandler@27eb3298{/storage,null,AVAILABLE}
17/08/16 09:09:55 INFO handler.ContextHandler: Started
o.s.j.s.ServletContextHandler@200a26bc{/storage/json,null,AVAILABLE}
17/08/16 09:09:55 INFO handler.ContextHandler: Started
o.s.j.s.ServletContextHandler@bc57b40{/storage/rdd,null,AVAILABLE}
```

```
17/08/16 09:09:55 INFO handler.ContextHandler: Started
o.s.j.s.ServletContextHandler@1b5bc39d{/storage/rdd/json,null,AVAILABLE}
17/08/16 09:09:55 INFO handler.ContextHandler: Started
o.s.j.s.ServletContextHandler@655a5d9c{/environment,null,AVAILABLE}
17/08/16 09:09:55 INFO handler.ContextHandler: Started
o.s.j.s.ServletContextHandler@1494b84d{/environment/json,null,AVAILABLE}
17/08/16 09:09:55 INFO handler.ContextHandler: Started
o.s.j.s.ServletContextHandler@34abdee4{/executors,null,AVAILABLE}
17/08/16 09:09:55 INFO handler.ContextHandler: Started
o.s.j.s.ServletContextHandler@71a9b4c7{/executors/json,null,AVAILABLE}
17/08/16 09:09:55 INFO handler.ContextHandler: Started
o.s.j.s.ServletContextHandler@4628b1d3{/executors/threadDump,null,AVAILABL
E}
17/08/16 09:09:55 INFO handler.ContextHandler: Started
o.s.j.s.ServletContextHandler@77cf3f8b{/executors/threadDump/json,null,AVA
ILABLE}
17/08/16 09:09:55 INFO handler.ContextHandler: Started
o.s.j.s.ServletContextHandler@1df98368{/static,null,AVAILABLE}
17/08/16 09:09:55 INFO handler.ContextHandler: Started
o.s.j.s.ServletContextHandler@21ca139c{/,null,AVAILABLE}
17/08/16 09:09:55 INFO handler.ContextHandler: Started
o.s.j.s.ServletContextHandler@226f885f{/api,null,AVAILABLE}
17/08/16 09:09:55 INFO handler.ContextHandler: Started
o.s.j.s.ServletContextHandler@2cd2c8fe{/jobs/job/kill,null,AVAILABLE}
17/08/16 09:09:55 INFO handler.ContextHandler: Started
o.s.j.s.ServletContextHandler@7d61eccf{/stages/stage/kill,null,AVAILABLE}
17/08/16 09:09:55 INFO server.ServerConnector: Started
ServerConnector@3bedc33d{HTTP/1.1}{0.0.0.0:4040}
17/08/16 09:09:55 INFO server.Server: Started @7246ms
17/08/16 09:09:55 INFO util.Utils: Successfully started service 'SparkUI'
on port 4040.
17/08/16 09:09:56 INFO ui.SparkUI: Bound SparkUI to 0.0.0.0, and started at
http://10.0.0.3:4040
17/08/16 09:09:56 INFO spark.SparkContext: Added JAR
file:/usr/spark-2.1.0/examples/jars/spark-examples_2.11-2.1.0.jar at
spark://10.0.0.3:36787/jars/spark-examples_2.11-2.1.0.jar with timestamp
1502874596164
17/08/16 09:09:56 INFO client.StandaloneAppClient$ClientEndpoint:
Connecting to master spark://master:7077...
17/08/16 09:09:56 INFO client.TransportClientFactory: Successfully created
connection to master/10.0.0.3:7077 after 81 ms (0 ms spent in bootstraps)
17/08/16 09:09:57 INFO cluster.StandaloneSchedulerBackend: Connected to
Spark cluster with app ID app-20170816090957-0000
17/08/16 09:09:57 INFO util.Utils: Successfully started service
'org.apache.spark.network.netty.NettyBlockTransferService' on port 40742.
17/08/16 09:09:57 INFO netty.NettyBlockTransferService: Server created on
10.0.0.3:40742
17/08/16 09:09:57 INFO storage.BlockManager: Using
org.apache.spark.storage.RandomBlockReplicationPolicy for block replication
policy
17/08/16 09:09:57 INFO storage.BlockManagerMaster: Registering BlockManager
BlockManagerId(driver, 10.0.0.3, 40742, None)
17/08/16 09:09:57 INFO storage.BlockManagerMasterEndpoint: Registering
```

```
block manager 10.0.0.3:40742 with 413.9 MB RAM, BlockManagerId(driver,
10.0.0.3, 40742, None)
17/08/16 09:09:57 INFO storage.BlockManagerMaster: Registered BlockManager
BlockManagerId(driver, 10.0.0.3, 40742, None)
17/08/16 09:09:57 INFO storage.BlockManager: Initialized BlockManager:
BlockManagerId(driver, 10.0.0.3, 40742, None)
17/08/16 09:09:57 INFO client.StandaloneAppClient$ClientEndpoint: Executor
added: app-20170816090957-0000/0 on worker-20170816075723-10.0.0.5-8881
(10.0.0.5:8881) with 2 cores
17/08/16 09:09:57 INFO cluster.StandaloneSchedulerBackend: Granted executor
ID app-20170816090957-0000/0 on hostPort 10.0.0.5:8881 with 2 cores, 1024.0
MB RAM
17/08/16 09:09:57 INFO client.StandaloneAppClient$ClientEndpoint: Executor
added: app-20170816090957-0000/1 on worker-20170816075723-10.0.0.8-8881
(10.0.0.8:8881) with 2 cores
17/08/16 09:09:57 INFO cluster.StandaloneSchedulerBackend: Granted executor
ID app-20170816090957-0000/1 on hostPort 10.0.0.8:8881 with 2 cores, 1024.0
MB RAM
17/08/16 09:09:57 INFO client.StandaloneAppClient$ClientEndpoint: Executor
added: app-20170816090957-0000/2 on worker-20170816075723-10.0.0.6-8881
(10.0.0.6:8881) with 2 cores
17/08/16 09:09:57 INFO cluster.StandaloneSchedulerBackend: Granted executor
ID app-20170816090957-0000/2 on hostPort 10.0.0.6:8881 with 2 cores, 1024.0
MB RAM
17/08/16 09:09:57 INFO client.StandaloneAppClient$ClientEndpoint: Executor
added: app-20170816090957-0000/3 on worker-20170816075741-10.0.0.7-8881
(10.0.0.7:8881) with 2 cores
17/08/16 09:09:57 INFO cluster.StandaloneSchedulerBackend: Granted executor
ID app-20170816090957-0000/3 on hostPort 10.0.0.7:8881 with 2 cores, 1024.0
MB RAM
17/08/16 09:09:57 INFO client.StandaloneAppClient$ClientEndpoint: Executor
added: app-20170816090957-0000/4 on worker-20170816075721-10.0.0.9-8881
(10.0.0.9:8881) with 2 cores
17/08/16 09:09:57 INFO cluster.StandaloneSchedulerBackend: Granted executor
ID app-20170816090957-0000/4 on hostPort 10.0.0.9:8881 with 2 cores, 1024.0
MB RAM
17/08/16 09:09:58 INFO client.StandaloneAppClient$ClientEndpoint: Executor
updated: app-20170816090957-0000/0 is now RUNNING
17/08/16 09:09:58 INFO client.StandaloneAppClient$ClientEndpoint: Executor
updated: app-20170816090957-0000/2 is now RUNNING
17/08/16 09:09:58 INFO client.StandaloneAppClient$ClientEndpoint: Executor
updated: app-20170816090957-0000/1 is now RUNNING
17/08/16 09:09:58 INFO client.StandaloneAppClient$ClientEndpoint: Executor
updated: app-20170816090957-0000/4 is now RUNNING
17/08/16 09:09:58 INFO client.StandaloneAppClient$ClientEndpoint: Executor
updated: app-20170816090957-0000/3 is now RUNNING
17/08/16 09:09:58 INFO handler.ContextHandler: Started
o.s.j.s.ServletContextHandler@5e76a2bb{/metrics/json,null,AVAILABLE}
17/08/16 09:09:58 INFO cluster.StandaloneSchedulerBackend: SchedulerBackend
is ready for scheduling beginning after reached
minRegisteredResourcesRatio: 0.0
17/08/16 09:09:59 INFO internal.SharedState: Warehouse path is
'file:/usr/spark-2.1.0/spark-warehouse'.
```

```
17/08/16 09:09:59 INFO handler.ContextHandler: Started
o.s.j.s.ServletContextHandler@18fdb6cf{/SQL,null,AVAILABLE}
17/08/16 09:09:59 INFO handler.ContextHandler: Started
o.s.j.s.ServletContextHandler@60baef24{/SQL/json,null,AVAILABLE}
17/08/16 09:09:59 INFO handler.ContextHandler: Started
o.s.j.s.ServletContextHandler@6ad5923a{/SQL/execution,null,AVAILABLE}
17/08/16 09:09:59 INFO handler.ContextHandler: Started
o.s.j.s.ServletContextHandler@43b0ade{/SQL/execution/json,null,AVAILABLE}
17/08/16 09:09:59 INFO handler.ContextHandler: Started
o.s.j.s.ServletContextHandler@4565a70a{/static/sql,null,AVAILABLE}
17/08/16 09:10:05 INFO
cluster.CoarseGrainedSchedulerBackend$DriverEndpoint: Registered executor
NettyRpcEndpointRef(null) (10.0.0.8:48314) with ID 1
17/08/16 09:10:05 INFO spark.SparkContext: Starting job: reduce at
SparkPi.scala:38
17/08/16 09:10:05 INFO
cluster.CoarseGrainedSchedulerBackend$DriverEndpoint: Registered executor
NettyRpcEndpointRef(null) (10.0.0.5:34582) with ID 0
17/08/16 09:10:05 INFO
cluster.CoarseGrainedSchedulerBackend$DriverEndpoint: Registered executor
NettyRpcEndpointRef(null) (10.0.0.9:42698) with ID 4
17/08/16 09:10:05 INFO
cluster.CoarseGrainedSchedulerBackend$DriverEndpoint: Registered executor
NettyRpcEndpointRef(null) (10.0.0.6:60914) with ID 2
17/08/16 09:10:05 INFO storage.BlockManagerMasterEndpoint: Registering
block manager 10.0.0.8:33023 with 413.9 MB RAM, BlockManagerId(1, 10.0.0.8,
33023, None)
17/08/16 09:10:05 INFO storage.BlockManagerMasterEndpoint: Registering
block manager 10.0.0.5:37643 with 413.9 MB RAM, BlockManagerId(0, 10.0.0.5,
37643, None)
17/08/16 09:10:05 INFO scheduler.DAGScheduler: Got job 0 (reduce at
SparkPi.scala:38) with 2 output partitions
17/08/16 09:10:05 INFO scheduler.DAGScheduler: Final stage: ResultStage 0
(reduce at SparkPi.scala:38)
17/08/16 09:10:05 INFO scheduler.DAGScheduler: Parents of final stage:
List()
17/08/16 09:10:05 INFO storage.BlockManagerMasterEndpoint: Registering
block manager 10.0.0.6:34005 with 413.9 MB RAM, BlockManagerId(2, 10.0.0.6,
34005, None)
17/08/16 09:10:05 INFO scheduler.DAGScheduler: Missing parents: List()
17/08/16 09:10:05 INFO storage.BlockManagerMasterEndpoint: Registering
block manager 10.0.0.9:45561 with 413.9 MB RAM, BlockManagerId(4, 10.0.0.9,
45561, None)
17/08/16 09:10:05 INFO scheduler.DAGScheduler: Submitting ResultStage 0
(MapPartitionsRDD[1] at map at SparkPi.scala:34), which has no missing
parents
17/08/16 09:10:07 INFO memory.MemoryStore: Block broadcast_0 stored as
values in memory (estimated size 1832.0 B, free 413.9 MB)
17/08/16 09:10:07 INFO memory.MemoryStore: Block broadcast_0_piece0 stored
as bytes in memory (estimated size 1172.0 B, free 413.9 MB)
17/08/16 09:10:07 INFO storage.BlockManagerInfo: Added broadcast_0_piece0
in memory on 10.0.0.3:40742 (size: 1172.0 B, free: 413.9 MB)
17/08/16 09:10:07 INFO spark.SparkContext: Created broadcast 0 from
```

```
broadcast at DAGScheduler.scala:996
17/08/16 09:10:07 INFO scheduler.DAGScheduler: Submitting 2 missing tasks
from ResultStage 0 (MapPartitionsRDD[1] at map at SparkPi.scala:34)
17/08/16 09:10:07 INFO scheduler.TaskSchedulerImpl: Adding task set 0.0
with 2 tasks
17/08/16 09:10:08 INFO scheduler.TaskSetManager: Starting task 0.0 in stage
0.0 (TID 0, 10.0.0.9, executor 4, partition 0, PROCESS_LOCAL, 6024 bytes)
17/08/16 09:10:08 INFO scheduler.TaskSetManager: Starting task 1.0 in stage
0.0 (TID 1, 10.0.0.5, executor 0, partition 1, PROCESS_LOCAL, 6024 bytes)
17/08/16 09:10:10 INFO storage.BlockManagerInfo: Added broadcast_0_piece0
in memory on 10.0.0.9:45561 (size: 1172.0 B, free: 413.9 MB)
17/08/16 09:10:10 INFO storage.BlockManagerInfo: Added broadcast_0_piece0
in memory on 10.0.0.5:37643 (size: 1172.0 B, free: 413.9 MB)
17/08/16 09:10:11 INFO scheduler.TaskSetManager: Finished task 1.0 in stage
0.0 (TID 1) in 2590 ms on 10.0.0.5 (executor 0) (1/2)
17/08/16 09:10:11 INFO scheduler.TaskSetManager: Finished task 0.0 in stage
0.0 (TID 0) in 3077 ms on 10.0.0.9 (executor 4) (2/2)
17/08/16 09:10:11 INFO scheduler.TaskSchedulerImpl: Removed TaskSet 0.0,
whose tasks have all completed, from pool
17/08/16 09:10:11 INFO scheduler.DAGScheduler: ResultStage 0 (reduce at
SparkPi.scala:38) finished in 3.138 s
17/08/16 09:10:11 INFO scheduler.DAGScheduler: Job 0 finished: reduce at
SparkPi.scala:38, took 6.065403 s
Pi is roughly 3.141275706378532
17/08/16 09:10:11 INFO server.ServerConnector: Stopped
ServerConnector@3bedc33d{HTTP/1.1}{0.0.0.0:4040}
17/08/16 09:10:11 INFO handler.ContextHandler: Stopped
o.s.j.s.ServletContextHandler@7d61eccf{/stages/stage/kill,null,UNAVAILABLE}
17/08/16 09:10:11 INFO handler.ContextHandler: Stopped
o.s.j.s.ServletContextHandler@2cd2c8fe{/jobs/job/kill,null,UNAVAILABLE}
17/08/16 09:10:11 INFO handler.ContextHandler: Stopped
o.s.j.s.ServletContextHandler@226f885f{/api,null,UNAVAILABLE}
17/08/16 09:10:11 INFO handler.ContextHandler: Stopped
o.s.j.s.ServletContextHandler@21ca139c{/,null,UNAVAILABLE}
17/08/16 09:10:11 INFO handler.ContextHandler: Stopped
o.s.j.s.ServletContextHandler@1df98368{/static,null,UNAVAILABLE}
17/08/16 09:10:11 INFO handler.ContextHandler: Stopped
o.s.j.s.ServletContextHandler@77cf3f8b{/executors/threadDump/json,null,UNA
VAILABLE}
17/08/16 09:10:11 INFO handler.ContextHandler: Stopped
o.s.j.s.ServletContextHandler@4628b1d3{/executors/threadDump,null,UNAVAILA
BLE}
17/08/16 09:10:11 INFO handler.ContextHandler: Stopped
o.s.j.s.ServletContextHandler@71a9b4c7{/executors/json,null,UNAVAILABLE}
17/08/16 09:10:11 INFO handler.ContextHandler: Stopped
o.s.j.s.ServletContextHandler@34abdee4{/executors,null,UNAVAILABLE}
17/08/16 09:10:11 INFO handler.ContextHandler: Stopped
o.s.j.s.ServletContextHandler@1494b84d{/environment/json,null,UNAVAILABLE}
17/08/16 09:10:11 INFO handler.ContextHandler: Stopped
o.s.j.s.ServletContextHandler@655a5d9c{/environment,null,UNAVAILABLE}
17/08/16 09:10:11 INFO handler.ContextHandler: Stopped
o.s.j.s.ServletContextHandler@1b5bc39d{/storage/rdd/json,null,UNAVAILABLE}
17/08/16 09:10:11 INFO handler.ContextHandler: Stopped
```

```
o.s.j.s.ServletContextHandler@bc57b40{/storage/rdd,null,UNAVAILABLE}
17/08/16 09:10:11 INFO handler.ContextHandler: Stopped
o.s.j.s.ServletContextHandler@200a26bc{/storage/json,null,UNAVAILABLE}
17/08/16 09:10:11 INFO handler.ContextHandler: Stopped
o.s.j.s.ServletContextHandler@27eb3298{/storage,null,UNAVAILABLE}
17/08/16 09:10:11 INFO handler.ContextHandler: Stopped
o.s.j.s.ServletContextHandler@21362712{/stages/pool/json,null,UNAVAILABLE}
17/08/16 09:10:11 INFO handler.ContextHandler: Stopped
o.s.j.s.ServletContextHandler@7e0aadd0{/stages/pool,null,UNAVAILABLE}
17/08/16 09:10:11 INFO handler.ContextHandler: Stopped
o.s.j.s.ServletContextHandler@1f130eaf{/stages/stage/json,null,UNAVAILABLE}
17/08/16 09:10:11 INFO handler.ContextHandler: Stopped
o.s.j.s.ServletContextHandler@4108fa66{/stages/stage,null,UNAVAILABLE}
17/08/16 09:10:11 INFO handler.ContextHandler: Stopped
o.s.j.s.ServletContextHandler@5910de75{/stages/json,null,UNAVAILABLE}
17/08/16 09:10:11 INFO handler.ContextHandler: Stopped
o.s.j.s.ServletContextHandler@127e70c5{/stages,null,UNAVAILABLE}
17/08/16 09:10:11 INFO handler.ContextHandler: Stopped
o.s.j.s.ServletContextHandler@27cf3151{/jobs/job/json,null,UNAVAILABLE}
17/08/16 09:10:11 INFO handler.ContextHandler: Stopped
o.s.j.s.ServletContextHandler@5ef8df1e{/jobs/job,null,UNAVAILABLE}
17/08/16 09:10:11 INFO handler.ContextHandler: Stopped
o.s.j.s.ServletContextHandler@3aee3976{/jobs/json,null,UNAVAILABLE}
17/08/16 09:10:11 INFO handler.ContextHandler: Stopped
o.s.j.s.ServletContextHandler@7889a1ac{/jobs,null,UNAVAILABLE}
17/08/16 09:10:11 INFO ui.SparkUI: Stopped Spark web UI at
http://10.0.0.3:4040
17/08/16 09:10:11 INFO cluster.StandaloneSchedulerBackend: Shutting down
all executors
17/08/16 09:10:11 INFO
cluster.CoarseGrainedSchedulerBackend$DriverEndpoint: Asking each executor
to shut down
17/08/16 09:10:12 INFO spark.MapOutputTrackerMasterEndpoint:
MapOutputTrackerMasterEndpoint stopped!
17/08/16 09:10:12 INFO memory.MemoryStore: MemoryStore cleared
17/08/16 09:10:12 INFO storage.BlockManager: BlockManager stopped
17/08/16 09:10:12 INFO storage.BlockManagerMaster: BlockManagerMaster
stopped
17/08/16 09:10:12 INFO
scheduler.OutputCommitCoordinator$OutputCommitCoordinatorEndpoint:
OutputCommitCoordinator stopped!
17/08/16 09:10:12 INFO spark.SparkContext: Successfully stopped
SparkContext
```

```
17/08/16 09:10:12 INFO util.ShutdownHookManager: Shutdown hook called
17/08/16 09:10:12 INFO util.ShutdownHookManager: Deleting directory
/tmp/spark-b55233b9-a809-46e5-bdc5-b5fe8b537c05
```