CS 696 Intro to Big Data: Tools and Methods
Fall Semester, 2019
Doc 17 Spark on AWS
Mar 28, 2019

Running Program on AWS EMR

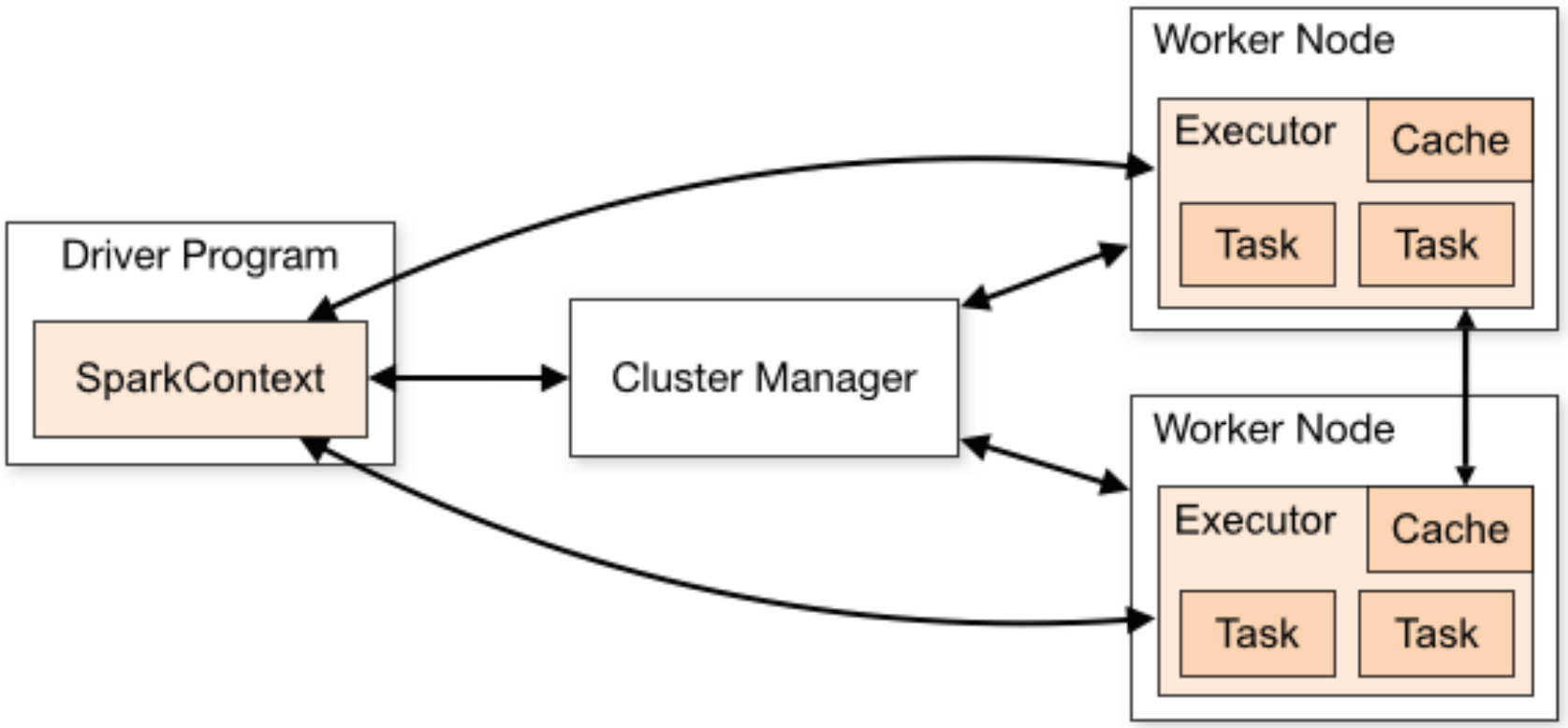
Make sure program runs locally

Create program file containing code

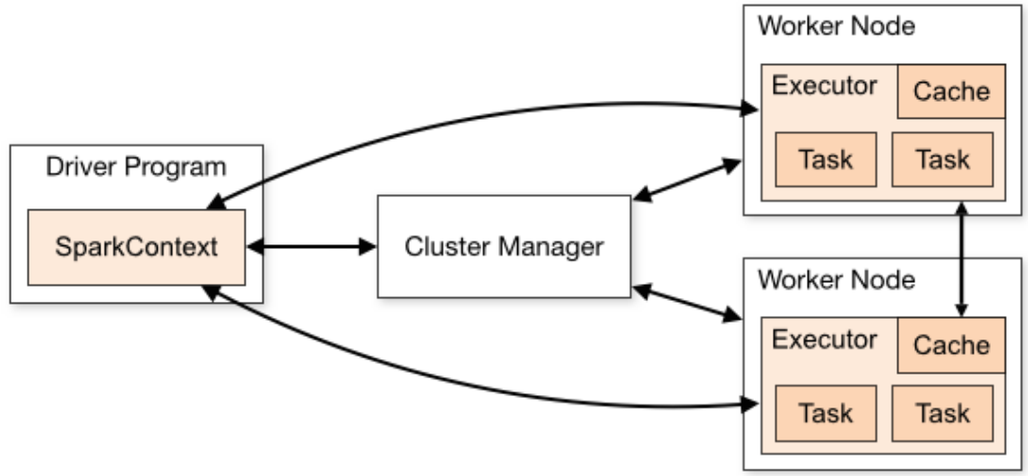
Create s3 bucket(s) for
- program file file
- logs
- input
- output

Upload program & data files to s3

Spark Components



Terms



Application

User program built on Spark

Driver program + executors

Driver program

The process running the main() function of the application and creating the SparkContext

Cluster manager

External service for acquiring resources on the cluster

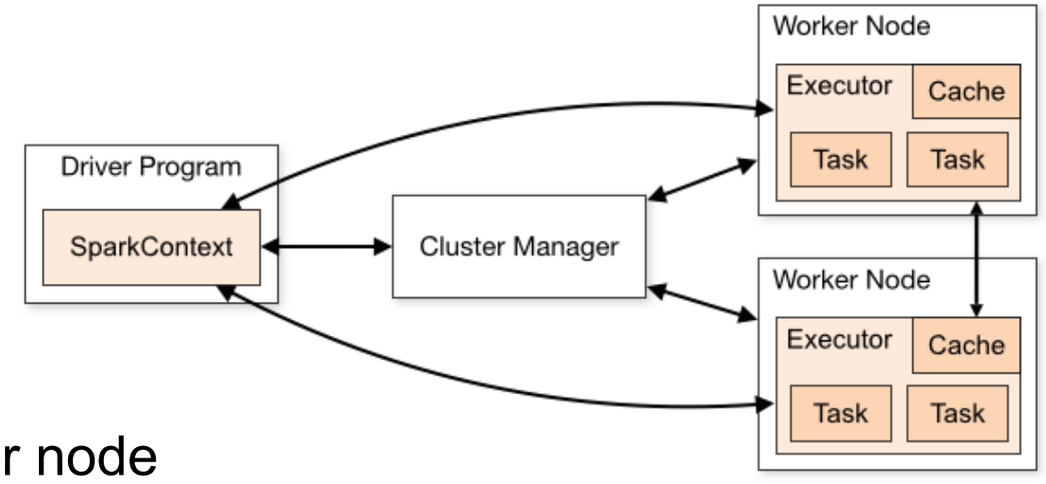
Deploy mode

Where the driver process runs

"cluster" - the driver inside of the cluster

"client" - the driver outside of the cluster

Terms



Executor

A process launched for an application on a worker node

Runs tasks and keeps data in memory or disk storage across them.

Each application has its own executors

Task

A unit of work that will be sent to one executor

Job

A parallel computation consisting of multiple tasks

Gets spawned in response to a Spark action (e.g. save, collect)

Stage

Job divided into smaller tasks called stages

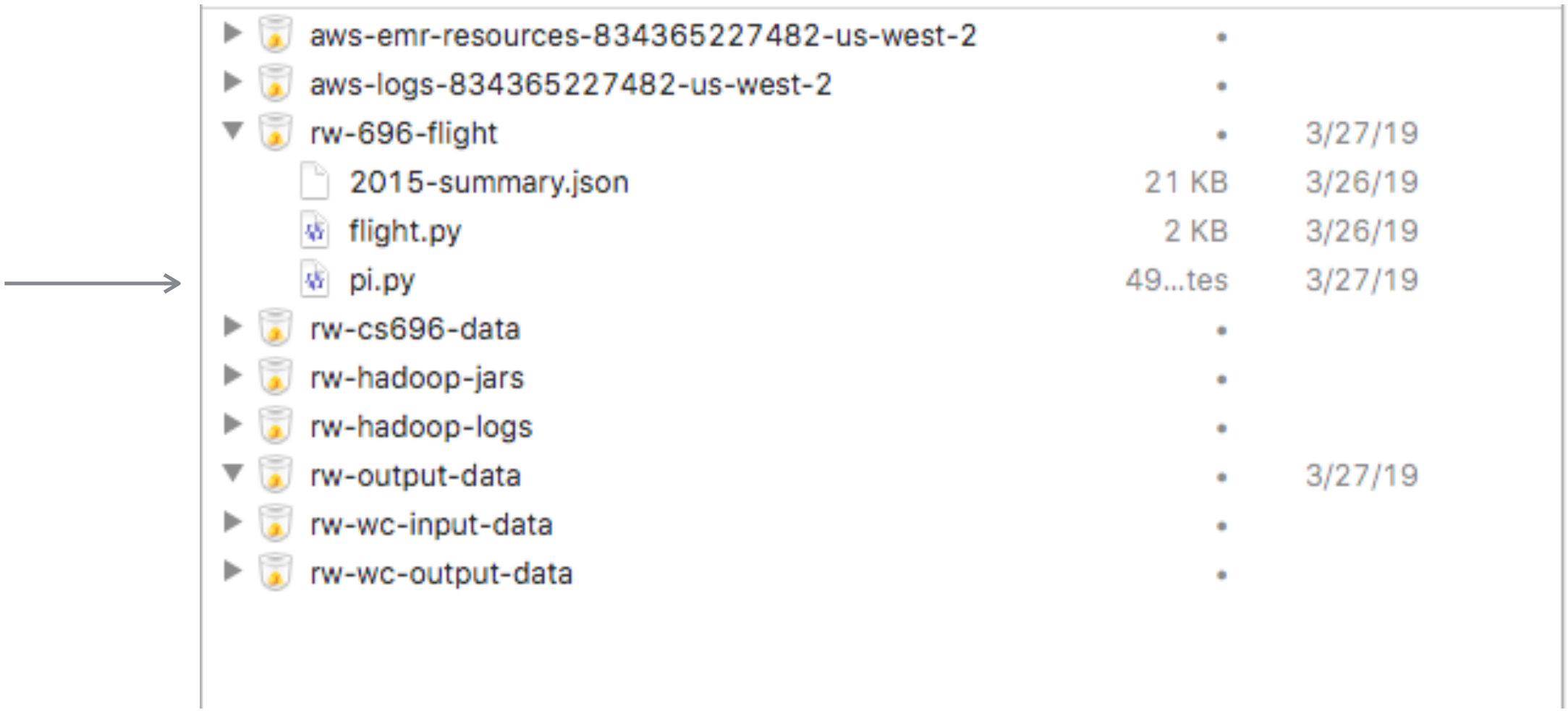
Depend on each other

Test Program 1 - Pi

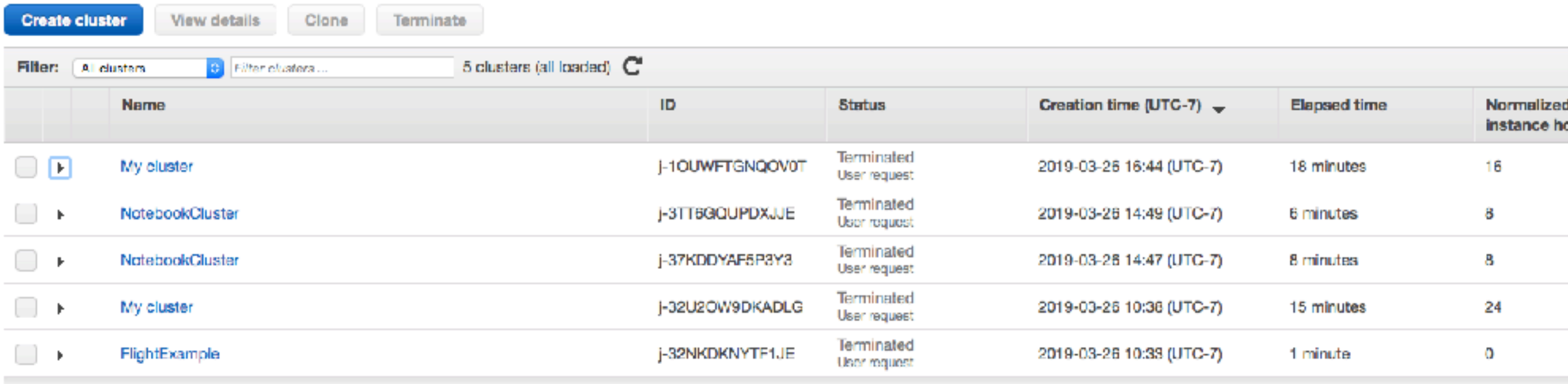
Designed to have no
- Command line dependancies
- No input or output files

```
from random import random
from operator import add

from pyspark import SparkContext

if __name__ == "__main__":
    sc = SparkContext(appName="PythonPi")
    partitions =  3
    n = 100000 * partitions

    def f(_):
        x = random() * 2 - 1
        y = random() * 2 - 1
        return 1 if x ** 2 + y ** 2 < 1 else 0

    count = sc.parallelize(range(1, n + 1), partitions).map(f).reduce(add)
    print("Pi is roughly %f" % (4.0 * count / n))

    sc.stop()
```

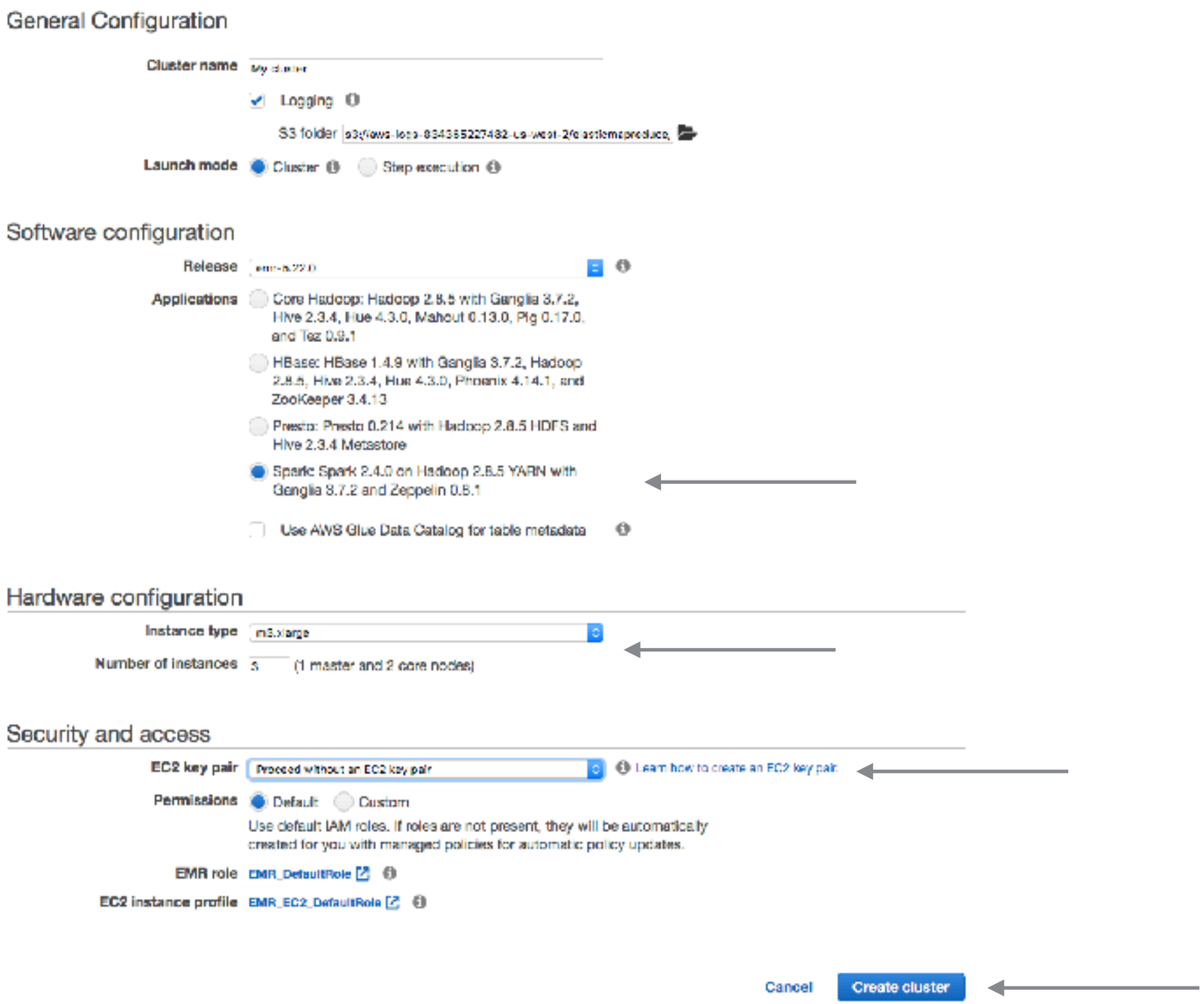
My S3 Buckets



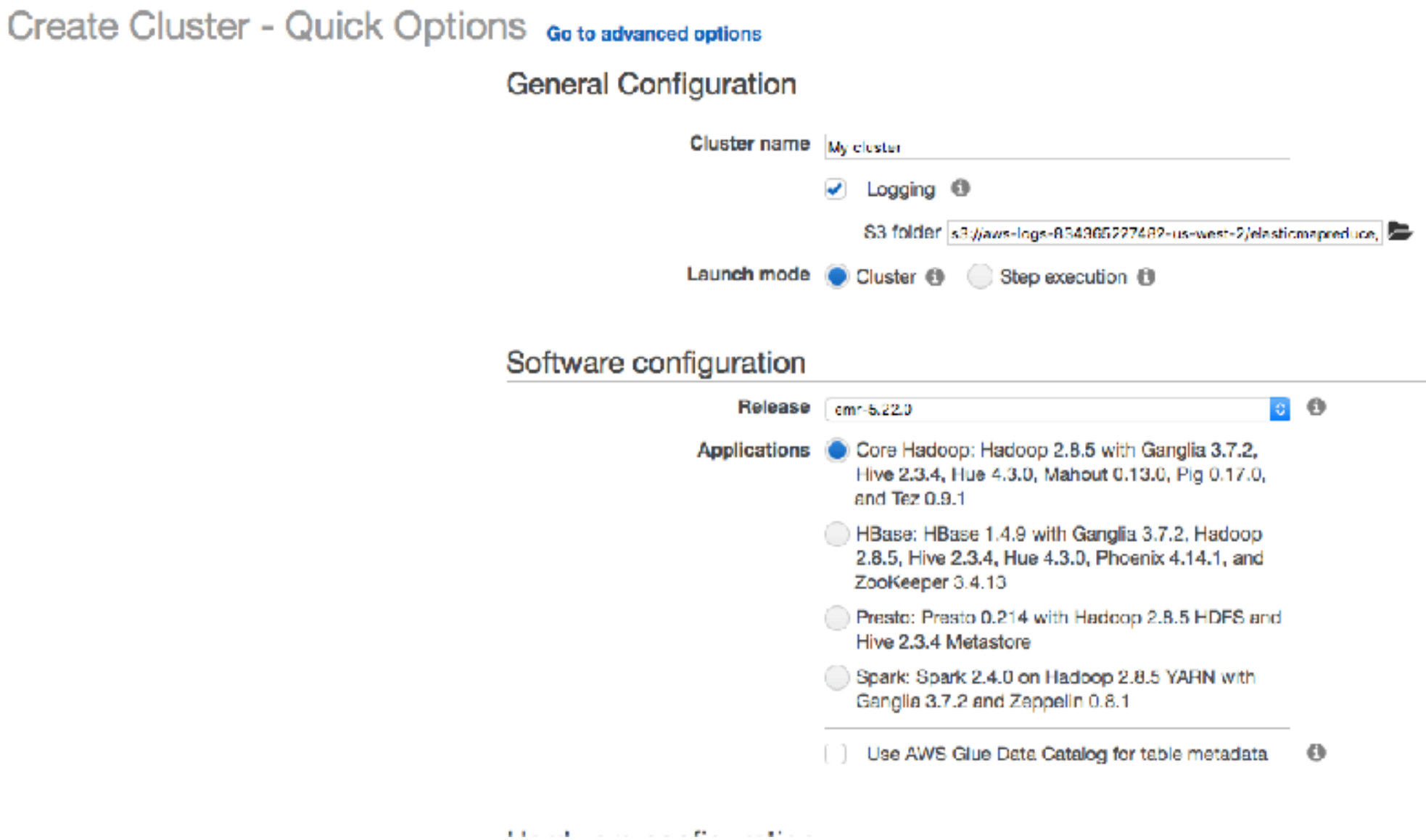
Spark on AWS - EMR Console



Using Quick Options



Use Advanced Options



Advanced Options

Software Configuration

Release emr-5.22.0

☑ Hadoop 2.8.5	☐ Zeppelin 0.8.1	☐ Livy 0.5.0
☐ JupyterHub 0.9.4	☐ Tez 0.9.1	☐ Flink 1.7.1
☐ Ganglia 3.7.2	☐ HBase 1.4.9	☐ Pig 0.17.0
☐ Hive 2.3.4	☐ Presto 0.215	☐ ZooKeeper 3.4.13
☐ MXNet 1.3.1	☐ Sqoop 1.4.7	☐ Mahout 0.13.0
☐ Hue 4.3.0	☐ Phoenix 4.14.1	☐ Oozie 5.1.0
☑ Spark 2.4.0	☐ HCatalog 2.3.4	☐ TensorFlow 1.12.0

AWS Glue Data Catalog settings (optional)

☐ Use for Spark table metadata

Edit software settings

◉ Enter configuration ○ Load JSON from S3

```
classification=config-file-name,properties=[myKey1=myValue1,myKey2=myValue2]
```

Add steps (optional)

Step type Spark application **Configure**

☐ Auto-terminate cluster after the last step is completed

Advanced Options

Hardware Configuration

If you need more than 20 EC2 instances, see this topic.

Instance group configuration

- **Uniform instance groups** — *Specify a single instance type and purchasing option for each node type.*
- **Instance fleets** — *Specify target capacity and how Amazon EMR fulfills it for each node type. Mix instance types and purchasing options. Learn more*

Network vpc-9a87d6fe (172.31.0.0/16) (default) Create a VPC

EC2 Subnet subnet-0f55190b | Default in us-west-2a

Root device EBS volume size 10 GiB

Choose the instance type, number of instances, and a purchasing option. You can choose to use On-Demand Instances, Spot Instances, or both. The instance type and purchasing option apply to all EC2 instances in each instance group, and you can only specify these options for an instance group when you create it. Learn more about instance purchasing options

Node type	Instance type	Instance count	Purchasing option	Auto Scaling
Master Master - 1	**m3.xlarge** 8 vCore, 15 GiB memory, 80 SSD GB storage EBS Storage: none Add configuration settings	1 Instances	On-demand Spot Use on-demand as max price	Not available for Master
Core Core - 2	**m3.xlarge** 8 vCore, 15 GiB memory, 80 SSD GB storage EBS Storage: none Add configuration settings	2 Instances	On-demand Spot Use on-demand as max price	Not enabled
Task Task - 3	**m3.xlarge** 8 vCore, 15 GiB memory, 80 SSD GB storage EBS Storage: none Add configuration settings	0 Instances	On-demand Spot Use on-demand as max price	Not enabled

+ Add task instance group

Cancel Previous Next

Advanced Options

General Options

Cluster name CourseSlides

Logging

S3 folder s3://aws-logs-834365227482-us-west-2/elasticmapreduce/

Debugging

Termination protection

Tags

Key	Value (optional)
Add a key to create a tag	

Additional Options

EMRFS consistent view

Custom AMI ID None

▸ Bootstrap Actions

Cancel Previous Next

Advanced Options

Security Options

EC2 key pair Proceed without an EC2 key pair

Cluster visible to all IAM users in account

This feature will be deprecated soon.

Permissions

Default Custom

Use default IAM roles. If roles are not present, they will be automatically created for you with managed policies for automatic policy updates.

EMR role EMR_DefaultRole

EC2 instance profile EMR_EC2_DefaultRole

Auto Scaling role EMR_AutoScaling_DefaultRole

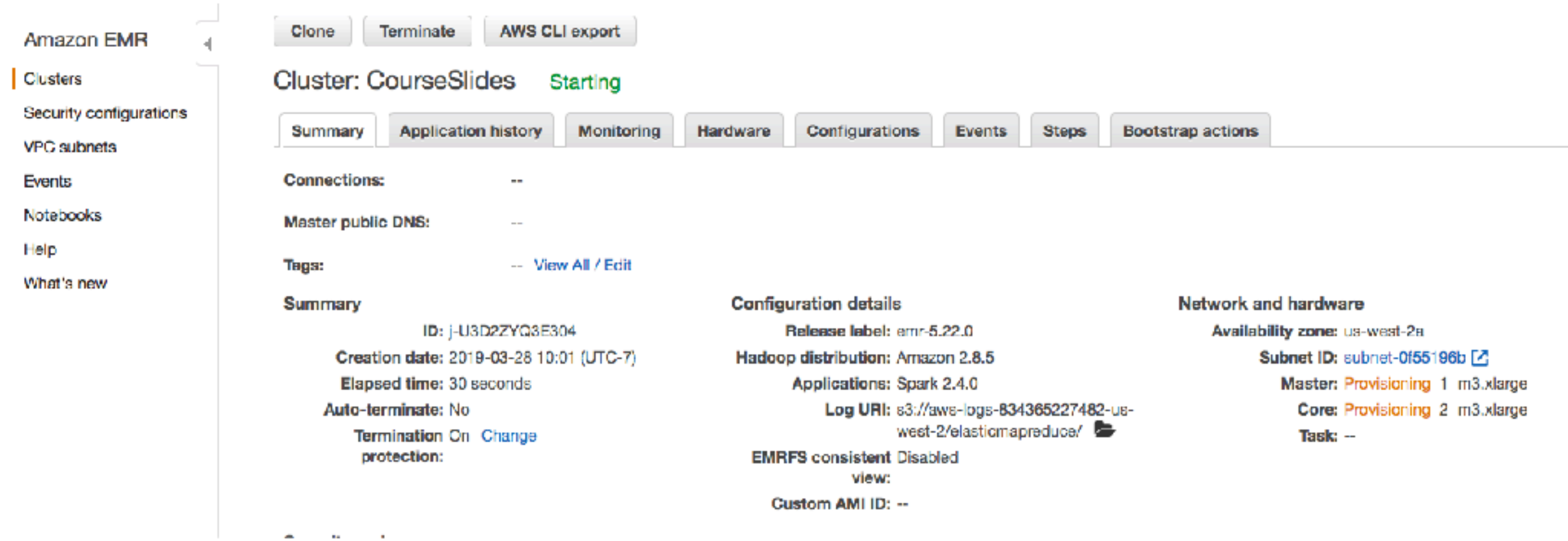
▸ Authentication and encryption

▸ EC2 security groups

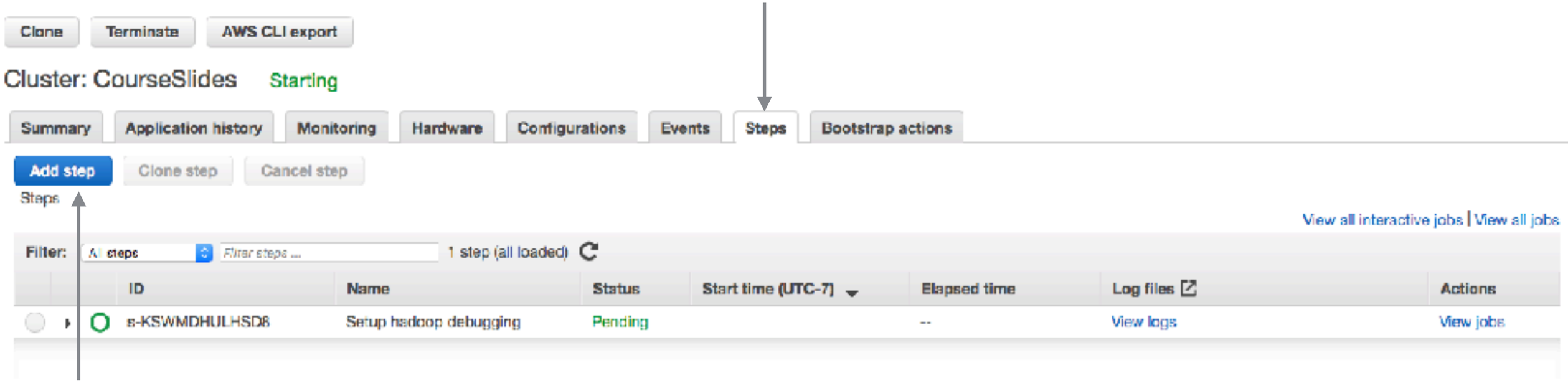
No EC2 key pair has been selected, so you will not be able to SSH to this cluster. Learn how to create an EC2 Key Pair.

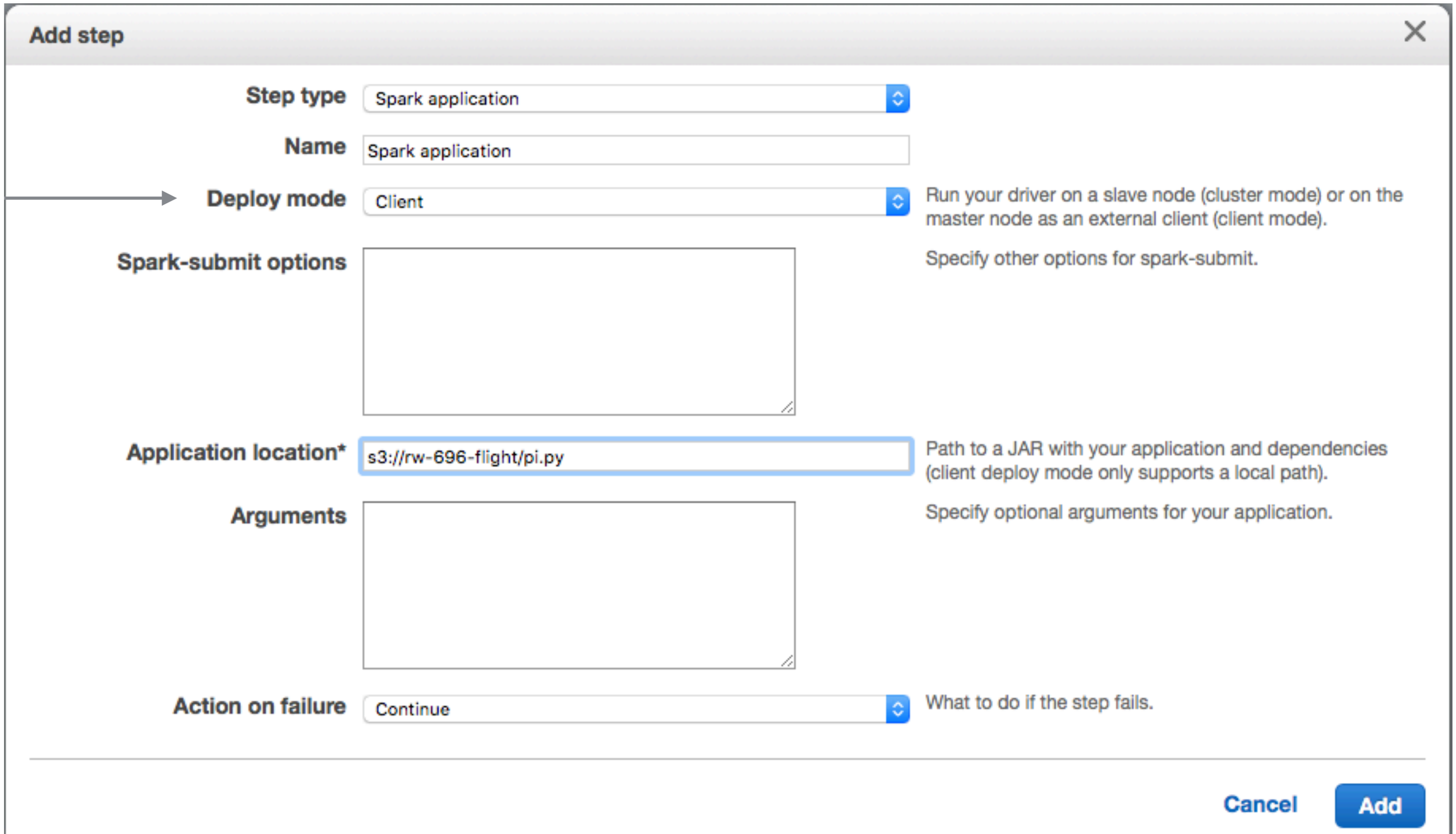
Cancel Previous Create cluster

Cluster Created - Either Quick or Advanced



Adding a Step





Modes - client or cluster

Either works

client mode gives access to standard out

Clone | Terminate | AWS CLI export

Cluster: CourseSlides Starting Configuring cluster software

Summary | Application history | Monitoring | Hardware | Configurations | Events | Steps | Bootstrap actions

Add step | Clone step | Cancel step

Steps

View all interactive jobs | View all jobs

Filter: All steps | Filter steps... | 2 steps (all loaded)

	ID	Name	Status	Start time (UTC-7)	Elapsed time	Log files	Actions
▸	s-3Q0ZVOAZVV3VR	Spark application	Pending		--	View logs	View jobs
▸	s-KSWMDHULHSD8	Setup hadoop debugging	Pending		--	View logs	View jobs

Cluster: CourseSlides Waiting Cluster ready after last step completed.

Summary | Application history | Monitoring | Hardware | Configurations | Events | Steps | Bootstrap actions

Add step | Clone step | Cancel step

Steps

View all interactive jobs | View

Filter: All steps | Filter steps ... | 2 steps (all loaded)

ID	Name	Status	Start time (UTC-7)	Elapsed time	Log files	Actions
s-3Q0ZVOAZVV3VR	Spark application	Completed	2019-03-28 10:05 (UTC-7)	36 seconds	View logs	View jobs
s-KSWMDHULHSD8	Setup hadoop debugging	Completed	2019-03-28 10:05 (UTC-7)	2 seconds	View logs	View jobs

Add step | Clone step | Cancel step

Steps

View all interactive jobs | View

Filter: All steps | Filter steps ... | 2 steps (all loaded)

ID	Name	Status	Start time (UTC-7)	Elapsed time	Log files	Actions
s-3Q0ZVOAZVV3VR	Spark application	Completed	2019-03-28 10:05 (UTC-7)	36 seconds	controller | syslog | stderr | stdout	View jobs
s-KSWMDHULHSD8	Setup hadoop debugging	Completed	2019-03-28 10:05 (UTC-7)	2 seconds	View logs	View jobs

```
Pi is roughly 3.138120
```

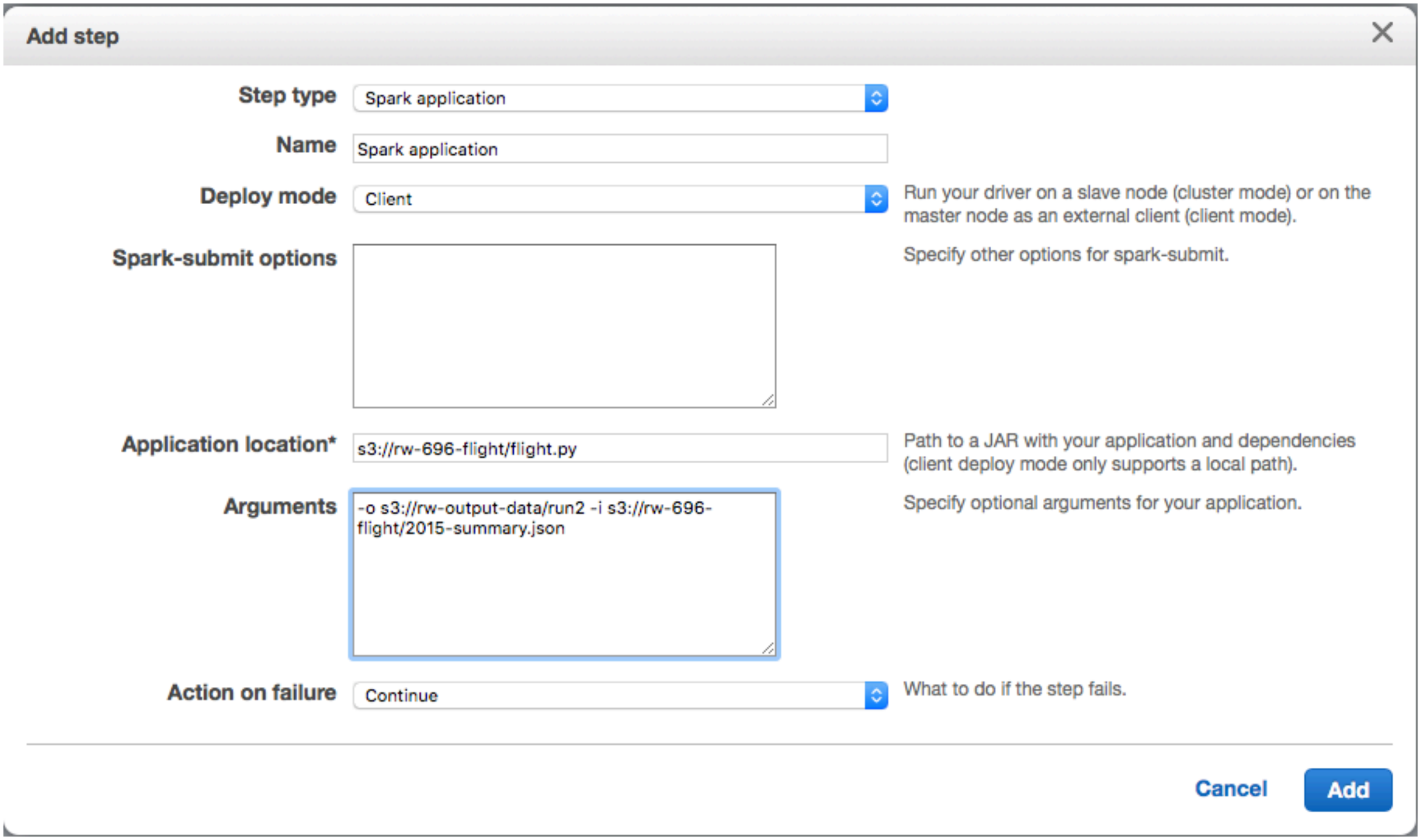
Example 2

```
def flight(input, output):
    import pyspark.sql.functions as F
    from pyspark.sql import SparkSession
    spark = SparkSession.builder \
        .appName("Fight") \
        .getOrCreate()

    flight_df = spark.read.json(input)

    grouped_df = flight_df.groupBy('DEST_COUNTRY_NAME').agg(F.sum('count'))
    grouped_df.write.format('csv').save(output)

def files_from_args():
        import argparse
        parser = argparse.ArgumentParser()
        parser.add_argument('-i', '--input', default='input')
        parser.add_argument('-o', '--output',default='output')
        args = parser.parse_args()
        return (args.input, args.output)

if __name__ == "__main__":
        inputfile, outputfile = files_from_args()
        flight(inputfile, outputfile)
```

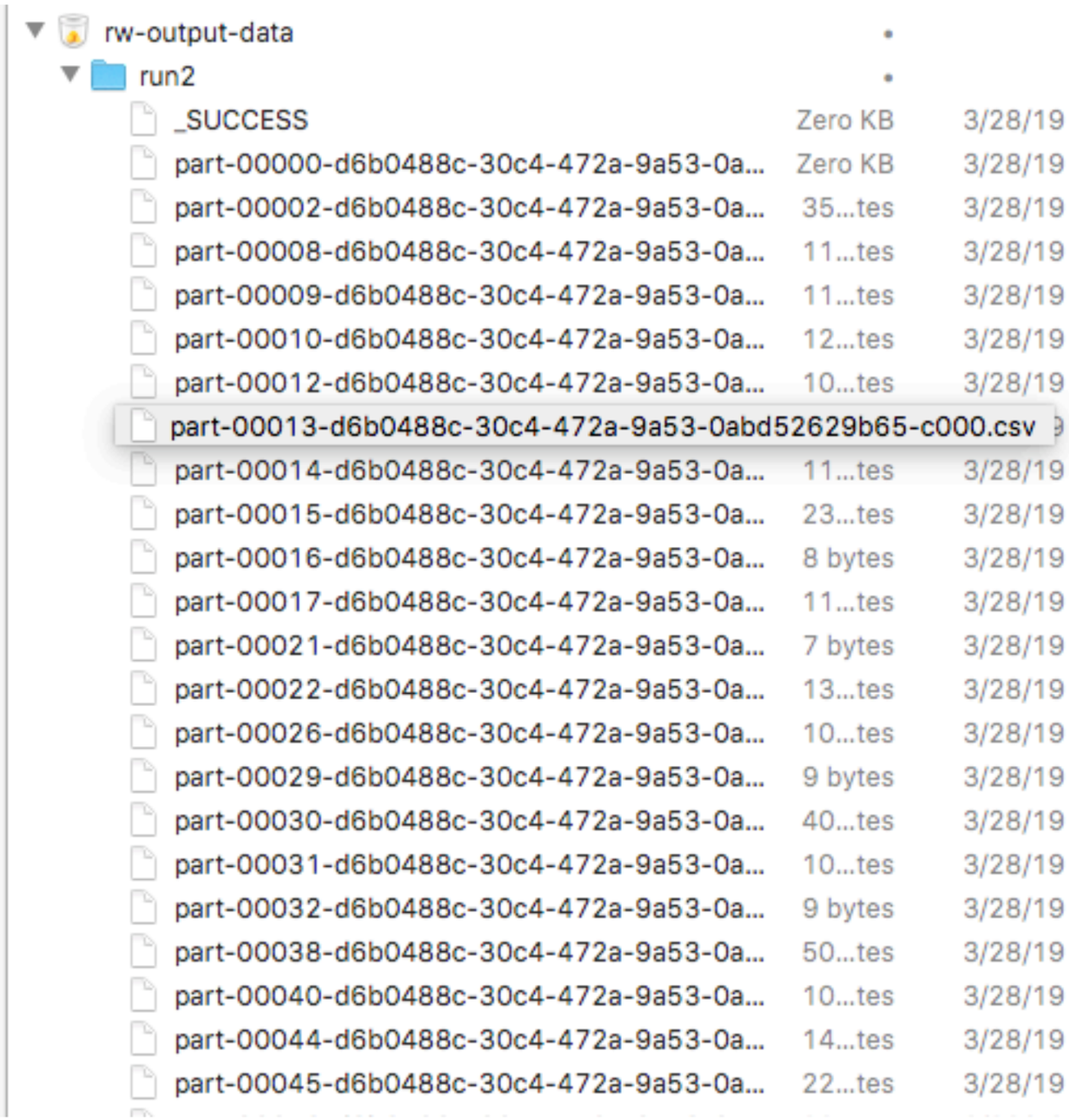
S3 Buckets


aws-emr-resources-834365227482-us-west-2
aws-logs-834365227482-us-west-2
rw-696-flight
2015-summary.json
21 KB
3/26/19
flight.py
2 KB
3/26/19
pi.py
49...tes
3/27/19
rw-cs696-data
rw-hadoop-jars

Added Step


Add step
Step type
Spark application
Name
Spark application
Deploy mode
Client
Run your driver on a slave node (cluster mode) or on the master node as an external client (client mode).
Spark-submit options
Specify other options for spark-submit.
Application location*
s3://rw-696-flight/flight.py
Path to a JAR with your application and dependencies (client deploy mode only supports a local path).
Arguments
-o s3://rw-output-data/run2 -i s3://rw-696-flight/2015-summary.json
Specify optional arguments for your application.
Action on failure
Continue
What to do if the step fails.
Cancel
Add

S3 output


rw-output-data
run2
_SUCCESS Zero KB 3/28/19
part-00000-d6b0488c-30c4-472a-9a53-0a... Zero KB 3/28/19
part-00002-d6b0488c-30c4-472a-9a53-0a... 35...tes 3/28/19
part-00008-d6b0488c-30c4-472a-9a53-0a... 11...tes 3/28/19
part-00009-d6b0488c-30c4-472a-9a53-0a... 11...tes 3/28/19
part-00010-d6b0488c-30c4-472a-9a53-0a... 12...tes 3/28/19
part-00012-d6b0488c-30c4-472a-9a53-0a... 10...tes 3/28/19
part-00013-d6b0488c-30c4-472a-9a53-0abd52629b65-c000.csv
part-00014-d6b0488c-30c4-472a-9a53-0a... 11...tes 3/28/19
part-00015-d6b0488c-30c4-472a-9a53-0a... 23...tes 3/28/19
part-00016-d6b0488c-30c4-472a-9a53-0a... 8 bytes 3/28/19
part-00017-d6b0488c-30c4-472a-9a53-0a... 11...tes 3/28/19
part-00021-d6b0488c-30c4-472a-9a53-0a... 7 bytes 3/28/19
part-00022-d6b0488c-30c4-472a-9a53-0a... 13...tes 3/28/19
part-00026-d6b0488c-30c4-472a-9a53-0a... 10...tes 3/28/19
part-00029-d6b0488c-30c4-472a-9a53-0a... 9 bytes 3/28/19
part-00030-d6b0488c-30c4-472a-9a53-0a... 40...tes 3/28/19
part-00031-d6b0488c-30c4-472a-9a53-0a... 10...tes 3/28/19
part-00032-d6b0488c-30c4-472a-9a53-0a... 9 bytes 3/28/19
part-00038-d6b0488c-30c4-472a-9a53-0a... 50...tes 3/28/19
part-00040-d6b0488c-30c4-472a-9a53-0a... 10...tes 3/28/19
part-00044-d6b0488c-30c4-472a-9a53-0a... 14...tes 3/28/19
part-00045-d6b0488c-30c4-472a-9a53-0a... 22...tes 3/28/19

Warning on AWS

It can take 5-10 minutes to start cluster

Logs do not show your logging statements

When you configure Steps incorrectly they fail
  Error messages are not very helpful

SSH to your Master Node

Create Amazon EC2 Key pair

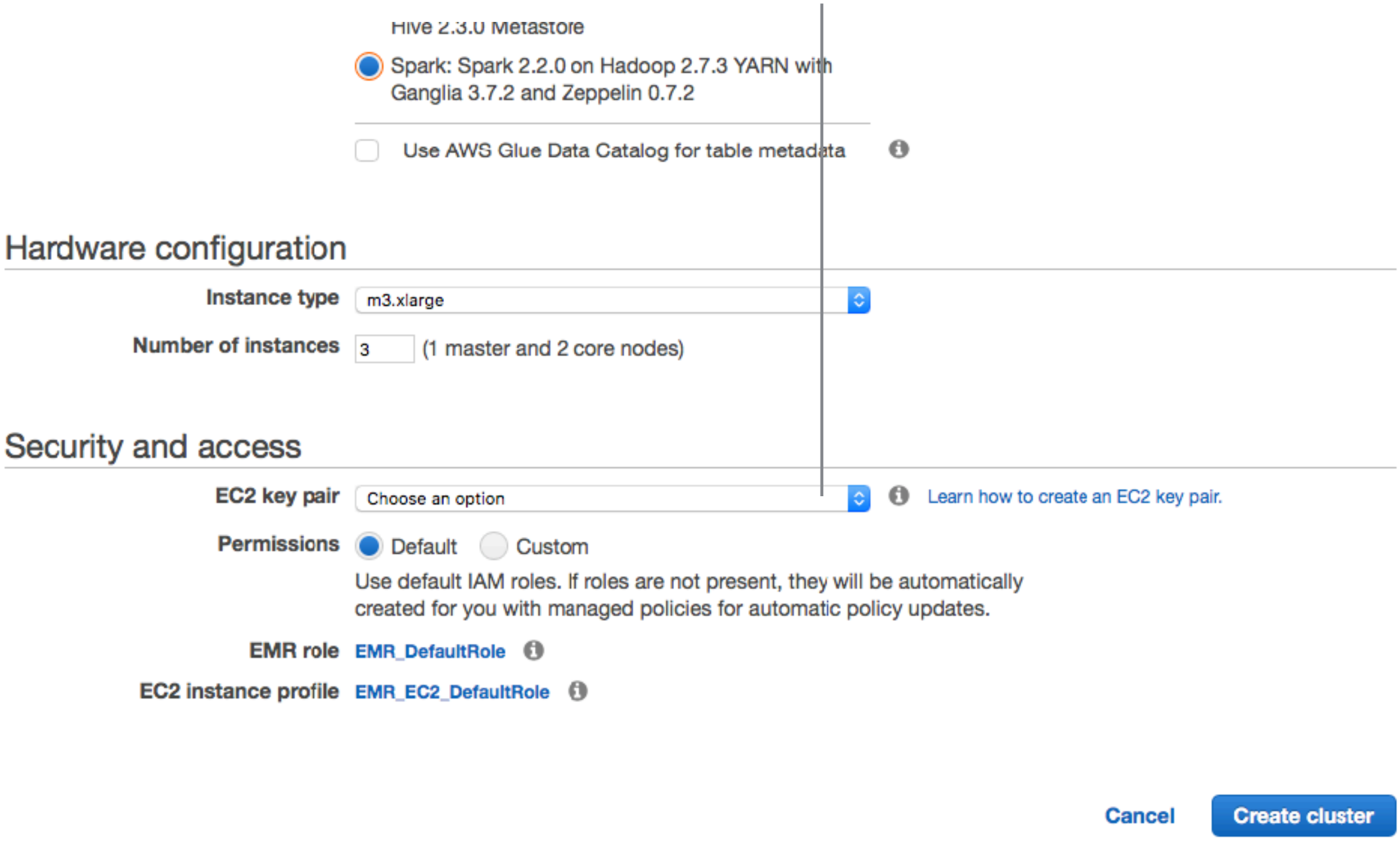
Instructions

http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-key-pairs.html#having-ec2-create-your-key-pair

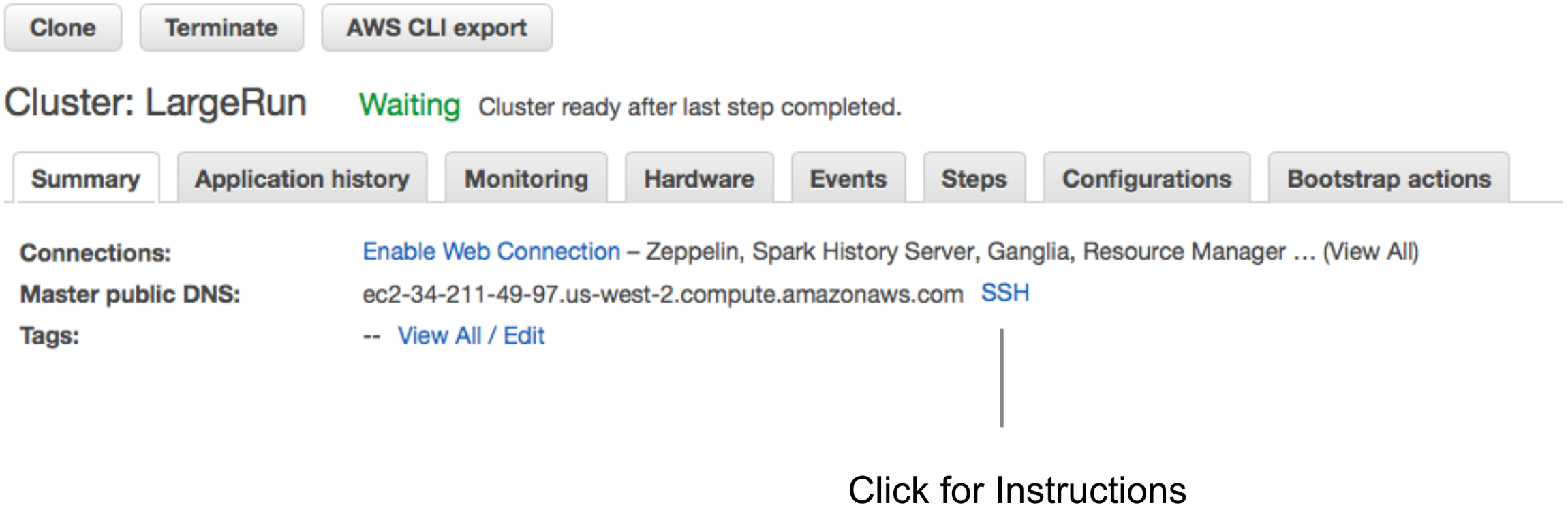
Open EC2 Dashboard - Select Key Pairs

SSH to your Master Node

In Create Cluster - Quick Options



SSH to your Master Node



Click for Instructions

Command-line Tools

Flintrock

Open-source command-line tool for launching Apache Spark clusters

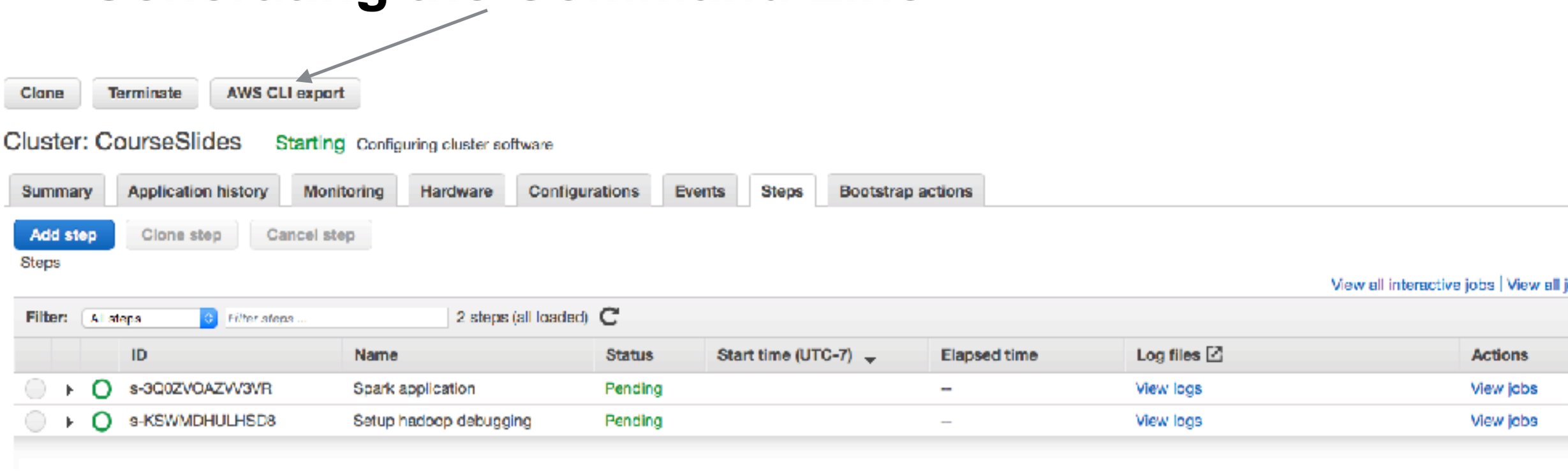
https://github.com/nchammas/flintrock

aws cli

Amazon's command line tool

https://aws.amazon.com/cli/

Generating the Command Line



AWS CLI export

```
aws emr create-cluster --termination-protected --applications Name=Hadoop Name=Spark --ec2-attributes
'{"InstanceProfile":"EMR_EC2_DefaultRole","SubnetId":"subnet-0f55196b","EmrManagedSlaveSecurityGroup":"sg-
65bffa1c","EmrManagedMasterSecurityGroup":"sg-62bffa1b"}' --release-label emr-5.22.0 --log-uri 's3n://aws-
logs-834365227482-us-west-2/elasticmapreduce/' --steps '[{"Args":["spark-submit","--deploy-
mode","client","s3://rw-696-flight/pi.py"],"Type":"CUSTOM_JAR","ActionOnFailure":"CONTINUE","Jar":"command-
runner.jar","Properties":"","Name":"Spark application"}]' --instance-groups
'[{"InstanceCount":1,"InstanceGroupType":"MASTER","InstanceType":"m3.xlarge","Name":"Master - 1"},
{"InstanceCount":2,"InstanceGroupType":"CORE","InstanceType":"m3.xlarge","Name":"Core - 2"}]' --auto-scaling-
role EMR_AutoScaling_DefaultRole --ebs-root-volume-size 10 --service-role EMR_DefaultRole --enable-
debugging --name 'CourseSlides' --scale-down-behavior TERMINATE_AT_TASK_COMPLETION --region us-west-2
```

Hadoop Ecosystem

Hadoop
- HDFS
- MapReduce
- YARN

Tez

Pig

Hive

Hbase

Sqoop

Oozie

Falcon

Spark

ZooKeeper

Mahout

Phoenix

BigTop

+ others

Apache Pig

Programming Map-Reduce can be low level

Apache Pig - high-level platform for creating programs for Hadoop

Pig Latin

```
input_lines = LOAD '/tmp/my-copy-of-all-pages-on-internet' AS (line:chararray);
words = FOREACH input_lines GENERATE FLATTEN(TOKENIZE(line)) AS word;
filtered_words = FILTER words BY word MATCHES '\\w+';
word_groups = GROUP filtered_words BY word;
word_count = FOREACH word_groups GENERATE COUNT(filtered_words) AS
                        count, group AS word;

ordered_word_count = ORDER word_count BY count DESC;
STORE ordered_word_count INTO '/tmp/number-of-words-on-internet';
```

Apache Hive

SQL is common way to interact with data

Hive provides SQL like query language for HDFS, Amazon S3 data

HiveQL - converted into MapReduce

```
DROP TABLE IF EXISTS docs;
CREATE TABLE docs (line STRING);
LOAD DATA INPATH 'input_file' OVERWRITE INTO TABLE docs;
CREATE TABLE word_counts AS
SELECT word, count(1) AS count FROM
 (SELECT explode(split(line, '\s')) AS word FROM docs) temp
GROUP BY word
ORDER BY word;
```

Apache HBase

BigTable for Hadoop

Non-relational distributed database

Fault-tolerant way of storing large quantites of sparse data

Apache Sqoop

People have data in non-hadoop databases

Sqoop

Transferring data between relational databases & Hadoop

Apache Phoenix

But SQL is common

Phoenix

- Massively parallel relational database for Hadoop
- Uses HBase to store data

Apache Spark

Hadoop has latency issues - reads data from disk
MapReduce is not conducive to solving all problems

Spark
- Uses distributed shared memory: Resilient distributed dataset (RDD)
- Iterative algorithms
- Implemented in Scala

Spark Core
Spark SQL
- Dataframes & SQL

Spark Streaming
Spark MLlib
- Machine learning

Apache Mahout

Hadoop does not have machine learning libraries

Mahout

- Environment for quickly creating scalable machine learning applications
- Samsara - R-line syntax & environment

Apache Flink, Apache Storm

Hadoop does batch jobs

Spark streaming has delays

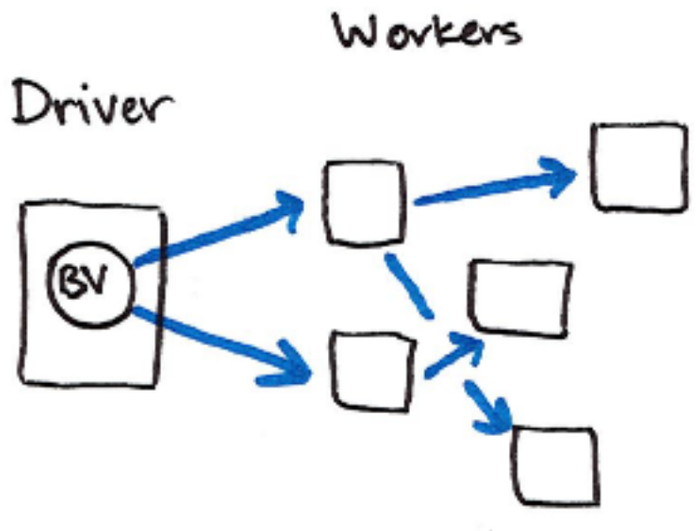
Fling & Storm

Each calin to have high throughput and low latency streaming

Distributed Variables

Broadcast

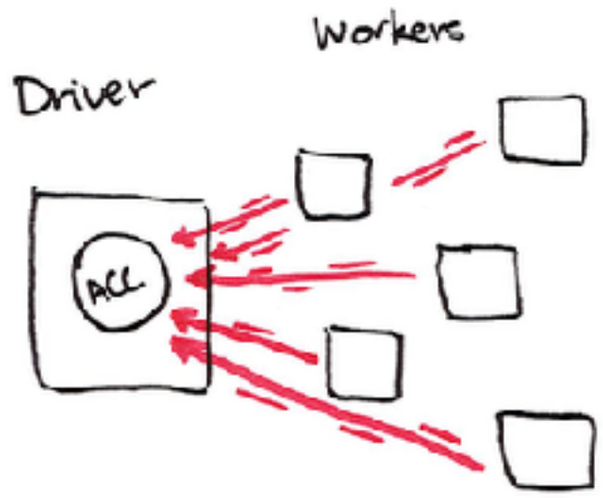
Read-only data shared among workers



Accumulator

Write only by workers

Read only on master



Broadcast Example

```
from pyspark.sql import SparkSession

spark = SparkSession\
        .builder\
        .appName("variables")\
        .getOrCreate()

courseSize = 45
courseSizeBroadcast = spark.sparkContext.broadcast(courseSize)

courseSizeBroadcast.value

data = spark.sparkContext.parallelize((1,2,3,4,5,6,7,8), 2)
data.map(lambda x: x + courseSizeBroadcast.value).collect()
```

Using ComplexType

```
sampleMap =  {'a': 10,'bat':1 }
sampleBroadCast = spark.sparkContext.broadcast(sampleMap)
sampleBroadCast.value
```

```
import org.apache.spark.sql.SparkSession
val blockSize =  "4096"
val spark = SparkSession.builder().
      appName("Broadcast Test").
      config("spark.broadcast.blockSize", blockSize).
      getOrCreate()

val sc = spark.sparkContext
val slices =  2
val num =  10000000

val arr1 = (0 until num).toArray

for (i <- 0 until 3) {
    println("Iteration " + i)
    println("===========")
    val startTime = System.nanoTime
    val barr1 = sc.broadcast(arr1)
    val observedSizes = sc.parallelize(1 to 10, slices).map(_ => barr1.value.length)
   observedSizes.collect().foreach(i => println(i))
    println("Iteration %d took %.0f milliseconds".format(i, (System.nanoTime - startTime) / 1E6))
}
```

Accumulator Example

```
from pyspark.sql import SparkSession

spark = SparkSession\
        .builder\
        .appName("variables")\
        .getOrCreate()

counter = spark.sparkContext.accumulator(0)
def count(item):
    global counter
    print("item: ", item.id)
    counter.add(1)

df = spark.range(16)
smaller = df.coalesce(4)
smaller.foreach(count)
counter.value
```

Output

16

Accumulator

add()

value

Numbers only

Can create custom accumulators

Machine Learning in Spark

MLlib

RDD-based
org.apache.spark.mllib
Maintenance mode

DataFrame based (Spark ML)
org.apache.spark.ml
Pipelines
Inspired by Python scikit-learn

Classification
Regression
Clustering
Collaborative Filtering
Dimension reduction
Linear Algebra
Statistics

http://spark.apache.org/docs/latest/ml-guide.html

MLlib: Main Guide

- Basic statistics
- Data sources
- Pipelines
- Extracting, transforming and selecting features
- Classification and Regression
- Clustering
- Collaborative filtering
- Frequent Pattern Mining
- Model selection and tuning
- Advanced topics

Machine Learning Library (MLlib) Guide

MLlib is Spark's machine learning (ML) library. Its goal is to make practical machine learning scalable provides tools such as:

- ML Algorithms: common learning algorithms such as classification, regression, clustering, and co
- Featurization: feature extraction, transformation, dimensionality reduction, and selection
- Pipelines: tools for constructing, evaluating, and tuning ML Pipelines
- Persistence: saving and load algorithms, models, and Pipelines
- Utilities: linear algebra, statistics, data handling, etc.

Announcement: DataFrame-based API is prima

The MLlib RDD-based API is now in maintenance mode.

As of Spark 2.0, the RDD-based APIs in the `spark.mllib` package have entered maintenance mode.

Python Examples

$SPARK_INSTALL_DIR/examples/src/main/python/ml

aft_survival_regression.py
als_example.py
binarizer_example.py
bisecting_k_means_example.py
bucketed_random_projection_lsh_example.py
bucketizer_example.py
chi_square_test_example.py
chisq_selector_example.py
correlation_example.py
count_vectorizer_example.py
cross_validator.py
dataframe_example.py
dct_example.py
decision_tree_classification_example.py
decision_tree_regression_example.py
elementwise_product_example.py
estimator_transformer_param_example.py
feature_hasher_example.py
fpgrowth_example.py
gaussian_mixture_example.py
generalized_linear_regression_example.py
gradient_boosted_tree_classifier_example.py
gradient_boosted_tree_regressor_example.py
imputer_example.py
index_to_string_example.py
isotonic_regression_example.py
kmeans_example.py
lda_example.py
linear_regression_with_elastic_net.py
linearsvc.py
logistic_regression_summary_example.py
logistic_regression_with_elastic_net.py
max_abs_scaler_example.py
min_hash_lsh_example.py
min_max_scaler_example.py
multiclass_logistic_regression_with_elastic_
multilayer_perceptron_classification.py
n_gram_example.py
naive_bayes_example.py
normalizer_example.py
one_vs_rest_example.py
onehot_encoder_estimator_example.py
pca_example.py
pipeline_example.py
polynomial_expansion_example.py
prefixspan_example.py
quantile_discretizer_example.py
random_forest_classifier_example.py