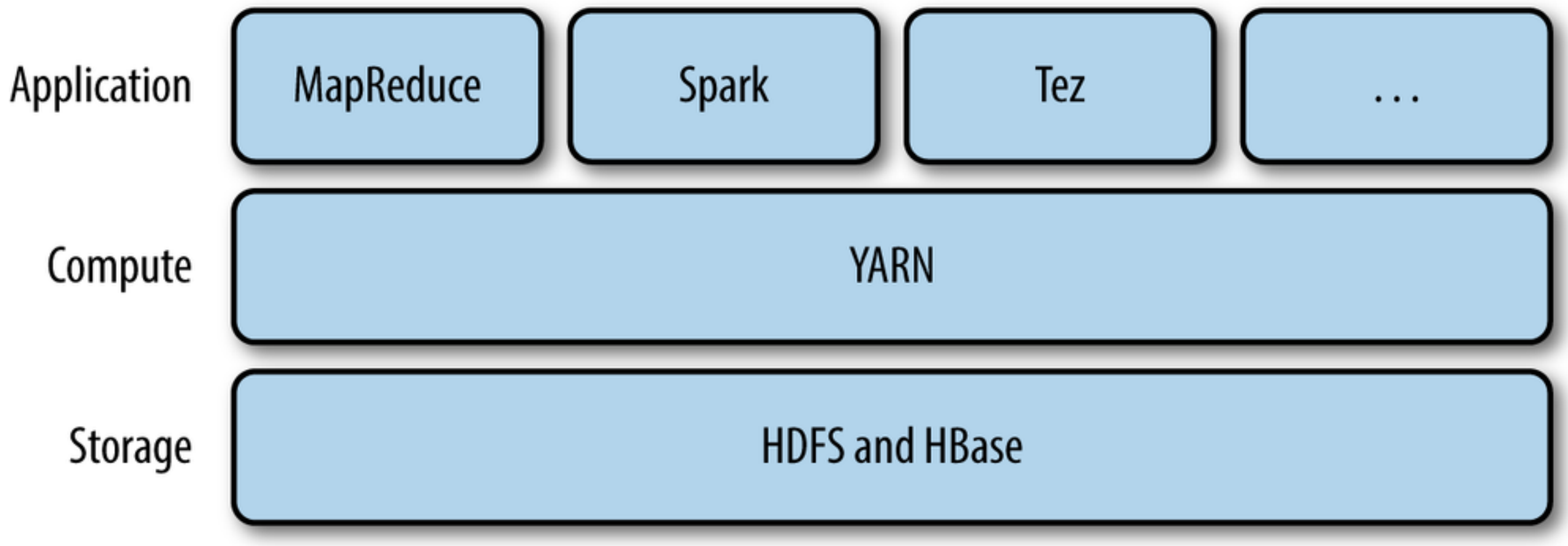
CS 696 Intro to Big Data: Tools and Methods
Fall Semester, 2016
Doc 20 Yarn, ToolRunner, AWS EMR
Nov 8, 2016

YARN

YARN

How to schedule jobs on a cluster
- Multiple requests at same time

Each request requires
- Different amount/type of resources
- Runs different length of time

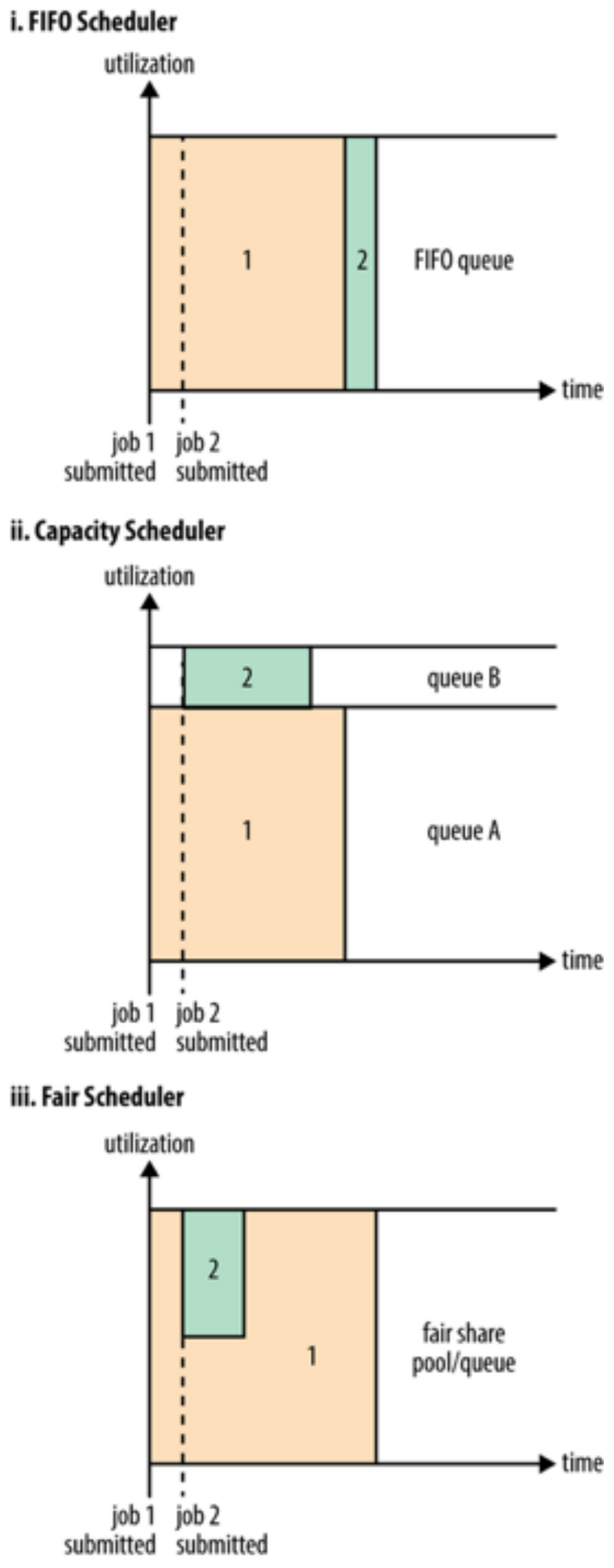


Yarn Scheduling Algorithms

FIFO

Capacity

Fair



Yarn FIFO Scheduler

Jobs are run in the order they are submitted

YARN Capacity Scheduler

Each group
- Assigned a part of the cluster
- Has separate queue for jobs with quota of resources available

Queue elasticity
- If parts of cluster are idle a queue may be assigned more than its quota
- When demand increases wait until jobs are finished to return resources to proper queue

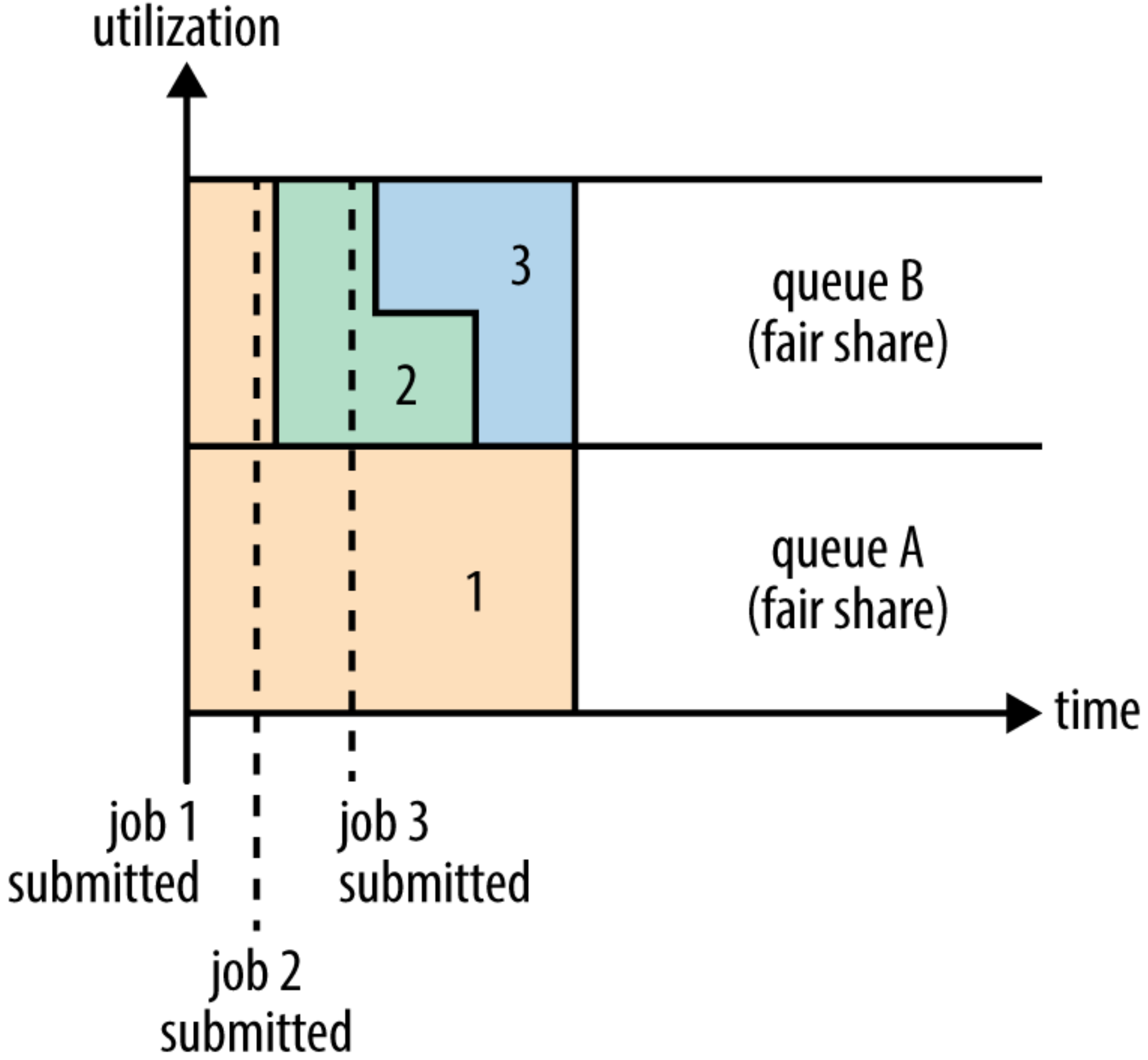
YARN Fair Scheduler

Each user/group has separate job queue

Configure what amount or resource is fair for each user

When new requests arrive
- Wait until resources are freed up
- Preempt running jobs

Each queue can have different scheduling algorithms


utilization
3
2
1
queue B
(fair share)
queue A
(fair share)
time
job 1
submitted
job 2
submitted
job 3
submitted

Delay Scheduling

What happens when a job requests a node that is busy

- Mode resources on given node to another node

Each node sends heartbeat to YARN resource manager

- Current status
- Each heartbeat is scheduling opportunity

Delay scheduling

- When requested node in busy
- Wait a given number of heartbeats before scheduling the job

Which Resource

Each job request
- CPUs
- Memory

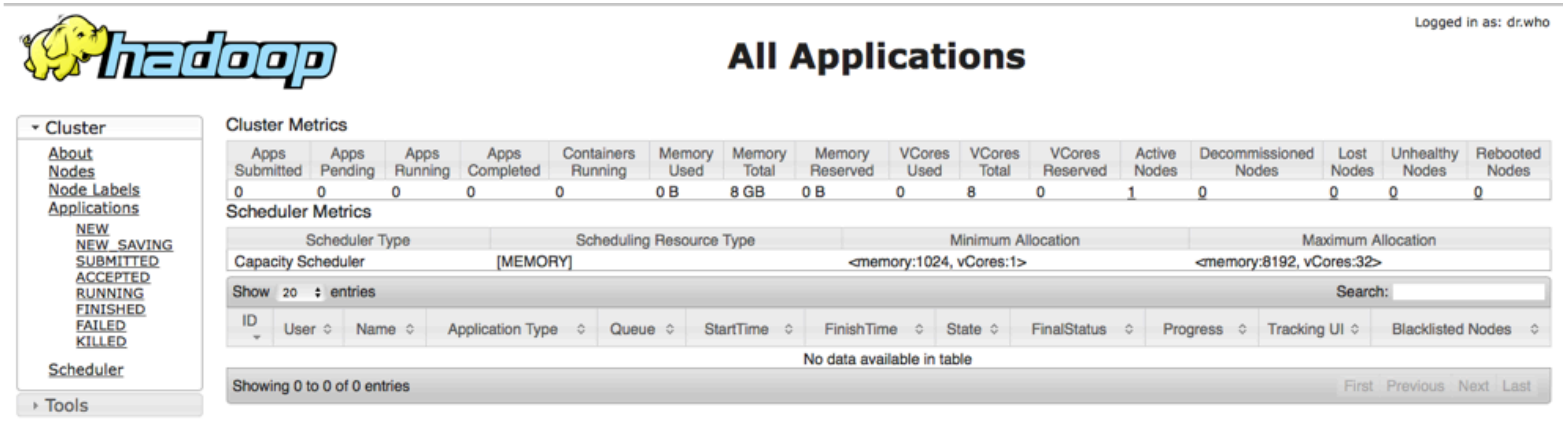
Which resource rquirement to use to determine how much of cluster is needed?
- Default is memory

Dominant Resource Fairness
- Uses the dominant resource
- YARN can be configured to use

Running YARN

start-yarn.sh

ResourceManager - http://localhost:8088/



Logged in as: dr.who

All Applications

- Cluster
  - About
  - Nodes
  - Node Labels
  - Applications
    - NEW
    - NEW_SAVING
    - SUBMITTED
    - ACCEPTED
    - RUNNING
    - FINISHED
    - FAILED
    - KILLED
  - Scheduler
- Tools

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes
0	0	0	0	0	0 B	8 GB	0 B	0	8	0	1	0	0	0	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation
Capacity Scheduler	[MEMORY]	<memory:1024, vCores:1>	<memory:8192, vCores:32>

Show 20 entries Search:

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Progress	Tracking UI	Blacklisted Nodes
No data available in table											

Showing 0 to 0 of 0 entries

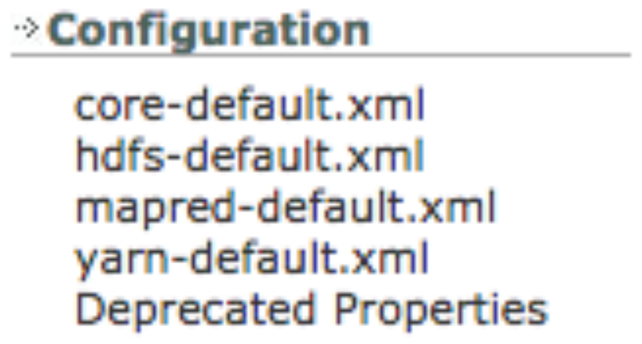
First Previous Next Last

MRUnit

MRUnit

- Junit for testing MapReduce functions

- Moved to Apache Attic in April 2016



Configuration Files

core-site.xml
hdfs-site.xml
httpfs-site.xml
mapred-site.xml
yarn-site.xml

http://hadoop.apache.org/docs/r2.7.3/

core.-site.xml	Default value
hadoop.tmp.dir	/tmp/hadoop-${user.name}

```
<property>
    <name>hadoop.tmp.dir</name>
    <value>/Your/path/here</value>
 </property>
```

Reading Configurations files

```
Configuration conf = new Configuration();
   conf.addResource("configuration-1.xml");
   assertThat(conf.get("color"), is("yellow"));
   assertThat(conf.getInt("size", 0), is(10));
   assertThat(conf.get("breadth", "wide"), is("wide"));
```

```
<configuration>
  <property>
    <name>color</name>
    <value>yellow</value>
    <description>Color</description>
  </property>

  <property>
    <name>size</name>
    <value>10</value>
    <description>Size</description>
  </property>

  <property>
    <name>weight</name>
    <value>heavy</value>
    <final>true</final>
    <description>Weight</description
  </property>
```

Variable Expansion

```
assertThat(conf.get("size-weight"), is("12,heavy"));
```

```
<configuration>
  <property>
    <name>size</name>
    <value>10</value>
    <description>Size</description>
  </property>

  <property>
    <name>weight</name>
    <value>heavy</value>
    <final>true</final>
    <description>Weight</description>
  </property>

  <property>
    <name>size-weight</name>
    <value>${size},${weight}</value>
    <description>Size and weight</descript
  </property>
</configuration>
```

Setting Configuration files

Some commands support -conf flag

```
hadoop fs -conf conf/hadoop-localhost.xml -ls .
```

Comands that do not support -conf

```
hadoop jar
hadoop className args
```

Setting HADOOP_CONF_DIR

Set environmental HADOOP_CONF_DIR

Set one directory for
- Standalone
- Pseudo-Distributed

Tool & ToolRunner

```
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;
```

Tool - interface for handling generic command-line options

    run(String[] args)

ToolRunner - Utility to run classes implementing Tool

    Uses GenericOptionsParser to parse command line options

Configured

```
import org.apache.hadoop.conf.Configured;


  getConf()
  setConf(Configuration conf)
```

Configuration

import org.apache.hadoop.conf.Configuration;

Contains get/set methods to get/set values in a configuration
    getBoolean, getClass, getDouble, getFile, getFloat, getPassword, etc.

Contains methods to add configuration files
    addDefaultResource(String)
    addResource(Configuration)
    addResource(Path)

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;
import java.util.Map.Entry;

public class ConfigurationPrinter extends Configured implements Tool{
    static {
        Configuration.addDefaultResource("/Java/hadoop-2.7.3/etc/hadoop/core-site.xml®"); }

    @Override
    public int run(String[] args) throws Exception {
        Configuration conf = getConf();
        int ftpPort = conf.getInt("fs.ftp.host.port", 23);
        for (Entry<String, String> entry: conf) {
            System.out.printf("%s=%s\n", entry.getKey(), entry.getValue());
        }
        return 0;
    }

    public static void main(String[] args) throws Exception {
        System.out.println("Start");
        int exitCode = ToolRunner.run(new ConfigurationPrinter(), args);
        System.exit(exitCode);
    }
}
```

ToolRunner options

Using Tool Runner To Run Jobs

```
public class WordCount extends Configured implements Tool {

    public static class TokenizerMapper
            extends Mapper<Object, Text, Text, IntWritable>{

        private final static IntWritable one = new IntWritable(1);
        private Text word = new Text();

        public void map(Object key, Text value, Context context
        ) throws IOException, InterruptedException {
            StringTokenizer itr = new StringTokenizer(value.toString());
            while (itr.hasMoreTokens()) {
                word.set(itr.nextToken());
                context.write(word, one);
            }
        }
    }
```

Using Tool Runner To Run Jobs

```
public static class IntSumReducer
        extends Reducer<Text,IntWritable,Text,IntWritable> {
    private IntWritable result = new IntWritable();

    public void reduce(Text key, Iterable<IntWritable> values,
                    Context context
    ) throws IOException, InterruptedException {
        int sum = 0;
        for (IntWritable val : values) {
            sum += val.get();
        }
        result.set(sum);
        context.write(key, result);
    }
}
```

```
@Override
public int run(String[] args) throws Exception {
    if (args.length != 2) {
        System.err.printf("Usage: %s [generic options] <input> <output>\n",
                getClass().getSimpleName());
        ToolRunner.printGenericCommandUsage(System.err);
        return -1;
    }

    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "word count");
    job.setJarByClass(WordCount.class);
    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(IntSumReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    job.setOutputValueClass(IntWritable.class);
    return job.waitForCompletion(true) ? 0 : 1;
}
```

The Main

```
    public static void main(String[] args) throws Exception {
        int exitCode = ToolRunner.run(new WordCount(), args);
        System.exit(exitCode);
    }
}
```

The Imports

```
import java.io.IOException;
import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;
```

Amazon EMR

Amazon Elastic Map-Reduce (EMR)

Hadoop, Hive, Spark, etc on Cluster

Predefined set of languages/tools available

Can create cluster of machines

https://aws.amazon.com

- Create new account
- Get 12 months free access

AWS Free Tier

12 months free

EC2 - compute instances
- 740 hours per month
- Billed in hour increments
- Billed per instance

S3 - storage
- 5 GB
- 20,000 Get requests

RDS - MySQL, PostgresSQL, SQL Sever
- 20 GB
- 750 hours

EC2 Container - Docker images
- 500 MB

EC2 Pricing

On Demand

m1.meduim	8.7 cents per Hour
m1.large	17.5 cents
m4.large	12 cents per hour

Spot Prices

m1.meduim	0.8 cents per hour
m4.large	2.12 cents per hour

Basic Outline

Develop & test Hadoop locally

- Use ToolRunner to run Job

Upload data to S3

Configure & launch cluster

- AWS Management Console
- AWS CLI
- SDKs

Monitor cluster

Make sure you terminate cluster when done

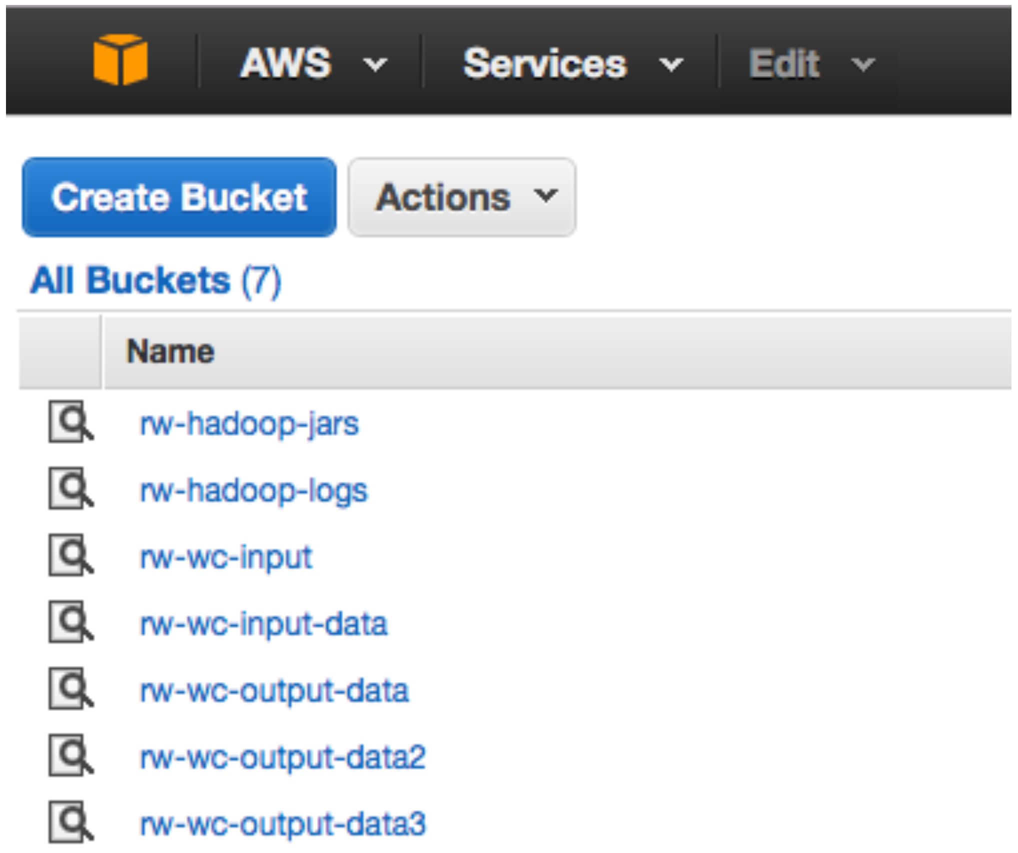
S3

Files are stored in buckets

Bucket names are global

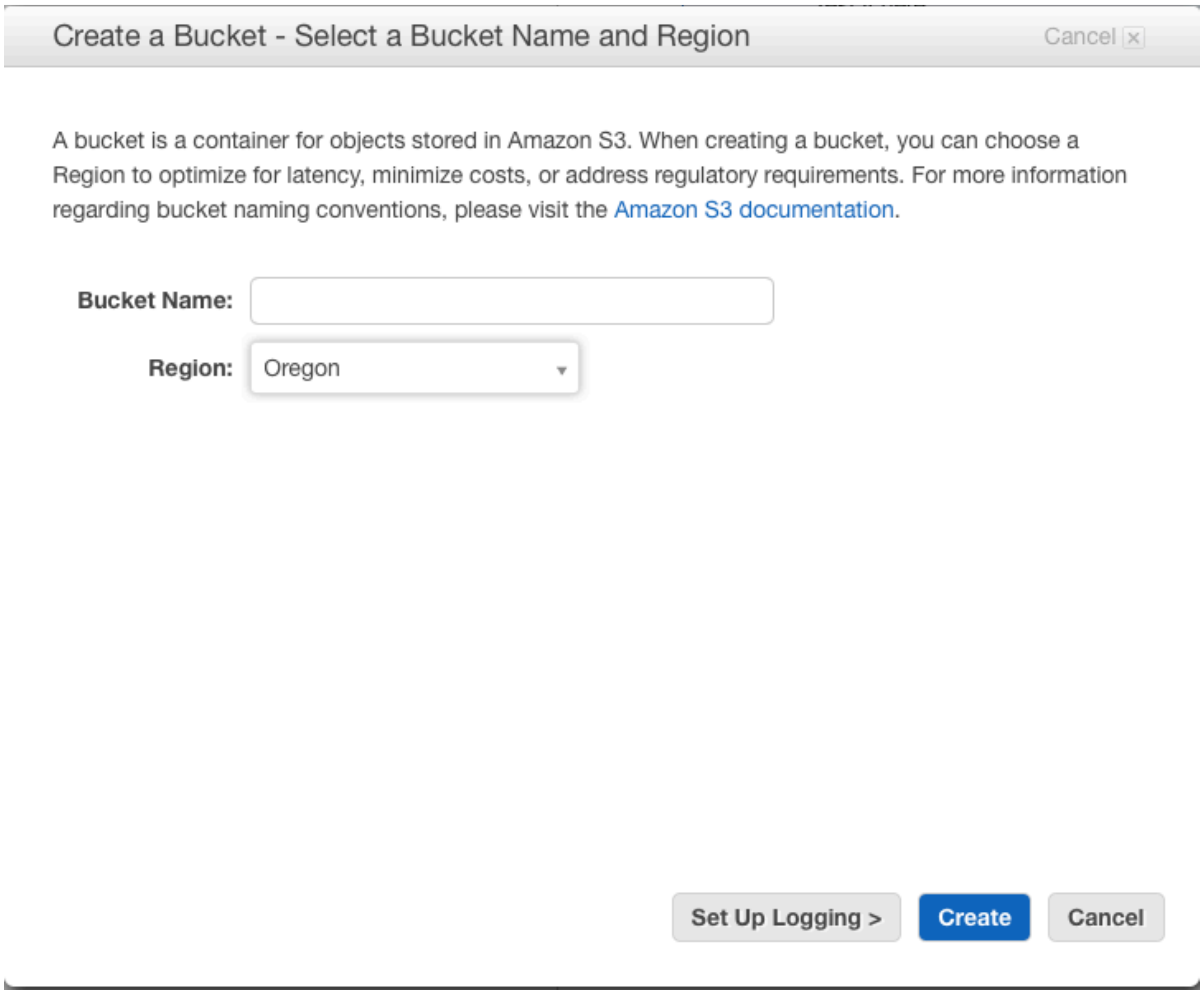
Supports
- s3 - files divided in to block
- s3n

Accessing files
- S3 console
- Third party



Name
rw-hadoop-jars
rw-hadoop-logs
rw-wc-input
rw-wc-input-data
rw-wc-output-data
rw-wc-output-data2
rw-wc-output-data3

S3 Creating a Bucket


Create a Bucket - Select a Bucket Name and Region

A bucket is a container for objects stored in Amazon S3. When creating a bucket, you can choose a Region to optimize for latency, minimize costs, or address regulatory requirements. For more information regarding bucket naming conventions, please visit the Amazon S3 documentation.

Bucket Name:

Region: Oregon

Running WordCount on AWS EMR

Make sure program runs locally

Use ToolRunner to run job

Create jar file for hadoop code

Create s3 buckets for

- jar file
- logs
- input
- output

Upload jar & data files to s3

Example

Source code: see slides 24-28

Compile the program

Create jar file with manifest file

Adding manifest file to jar file

https://docs.oracle.com/javase/tutorial/deployment/jar/appman.html

Manifest.txt

```
Main-Class: WordCount
```

```
jar cfm WodsCount.jar Manifest.txt listClassFiles
```

S3 buckets for Example

rw-hadoop-jars
    WordCount.jar

rw-wc-input-data
    file01
    file02

rw-hadoop-logs

rw-wc-output-data

file01

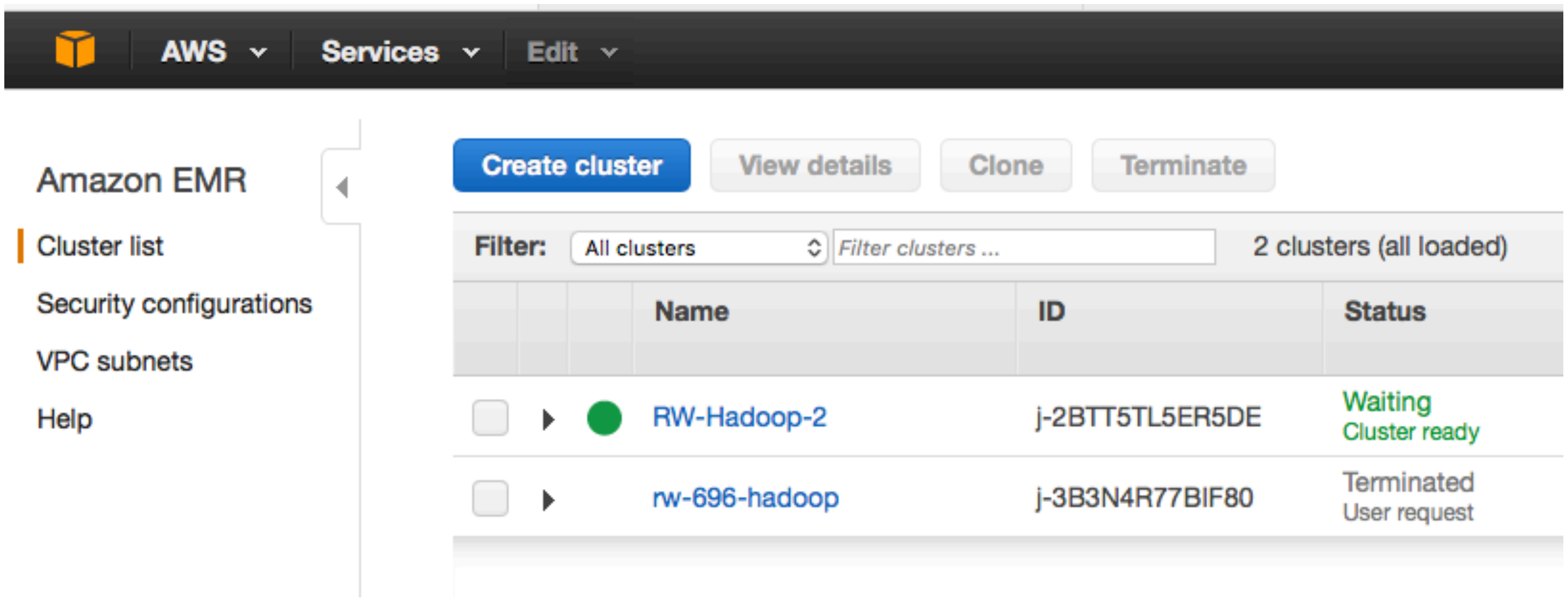
Hello World Bye World

file02

Hello Hadoop Goodbye Hadoop

Creating a Cluster

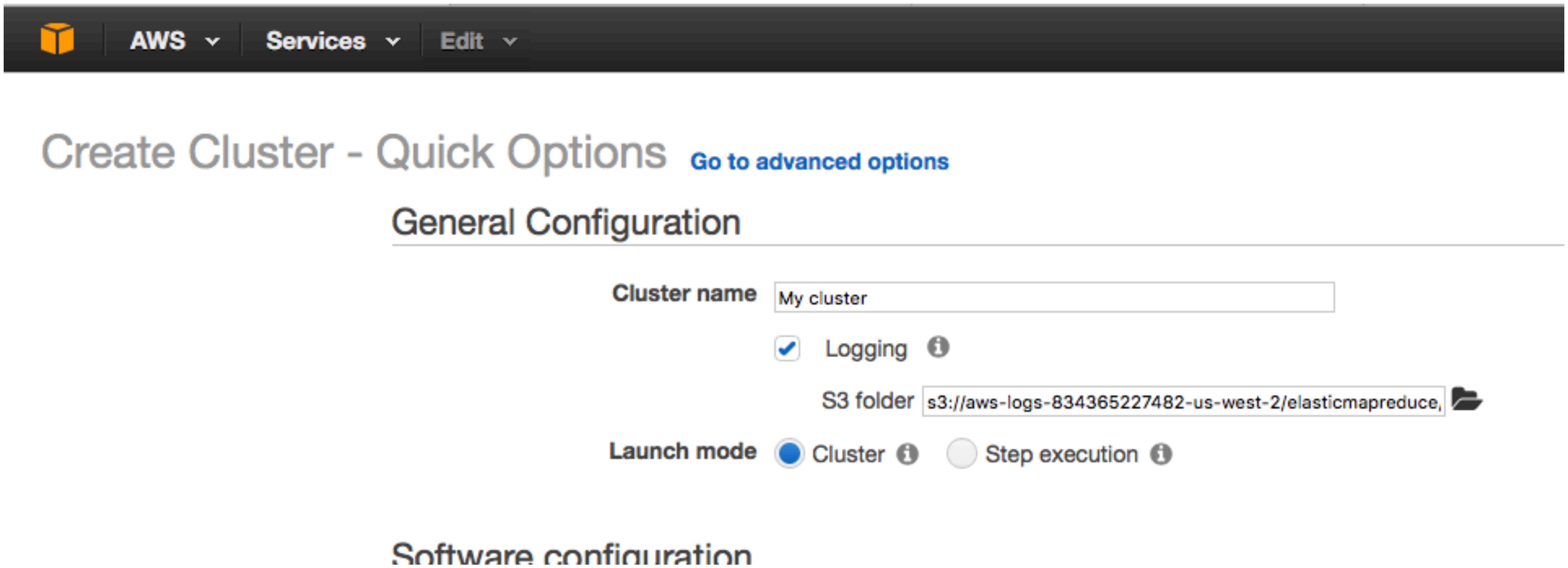
AWS EMR managemet console



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

Name	ID	Status
RW-Hadoop-2	j-2BTT5TL5ER5DE	Waiting Cluster ready
rw-696-hadoop	j-3B3N4R77BIF80	Terminated User request

Select Advanced Options



Create Cluster - Quick Options Go to advanced options

General Configuration

Cluster name: My cluster

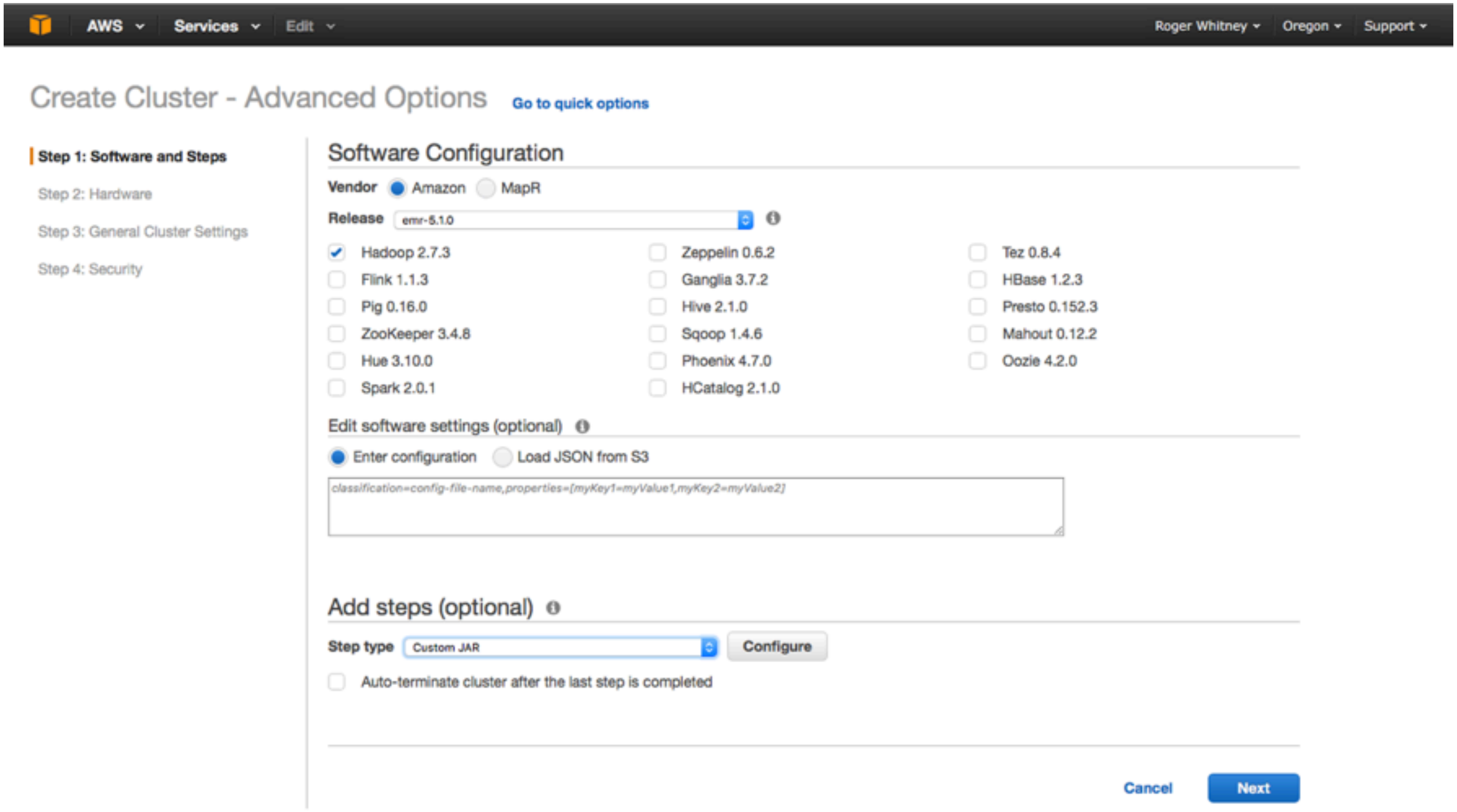
[✓] Logging

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

Launch mode: [● Cluster] ○ Step execution

Software configuration

Add steps - Select Custom Jar



Create Cluster - Advanced Options

Step 1: Software and Steps
Step 2: Hardware
Step 3: General Cluster Settings
Step 4: Security

Software Configuration

Vendor [✓ Amazon] MapR

Release emr-5.1.0

✓ Hadoop 2.7.3	☐ Zeppelin 0.6.2	☐ Tez 0.8.4
☐ Flink 1.1.3	☐ Ganglia 3.7.2	☐ HBase 1.2.3
☐ Pig 0.16.0	☐ Hive 2.1.0	☐ Presto 0.152.3
☐ ZooKeeper 3.4.8	☐ Sqoop 1.4.6	☐ Mahout 0.12.2
☐ Hue 3.10.0	☐ Phoenix 4.7.0	☐ Oozie 4.2.0
☐ Spark 2.0.1	☐ HCatalog 2.1.0	

Edit software settings (optional)

[✓ Enter configuration] Load JSON from S3

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

Add steps (optional)

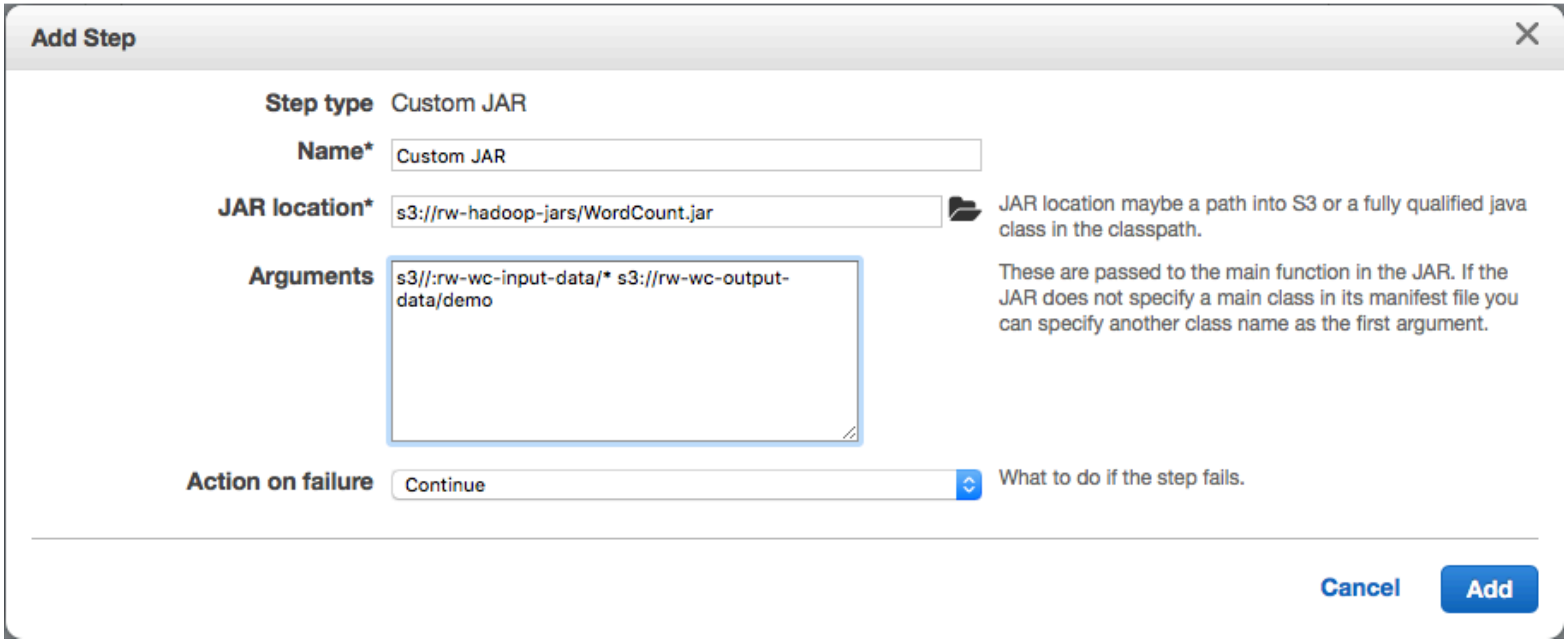
Step type Custom JAR

☐ Auto-terminate cluster after the last step is completed

For development leave Auto-terminate cluster off

Once developed can turn on

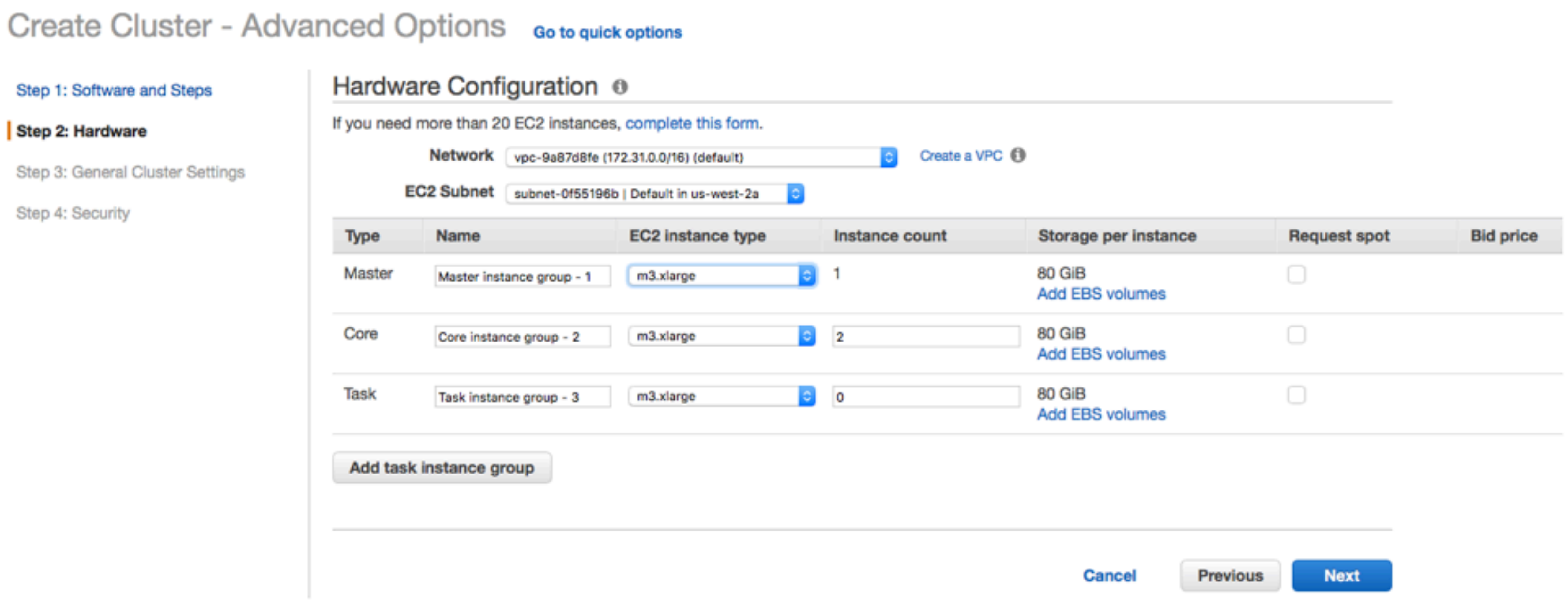
Custom Jar



Add Step

Step type	Custom JAR	
Name*	Custom JAR	
JAR location*	s3://rw-hadoop-jars/WordCount.jar	JAR location maybe a path into S3 or a fully qualified java class in the classpath.
Arguments	s3//:rw-wc-input-data/* s3://rw-wc-output-data/demo	These are passed to the main function in the JAR. If the JAR does not specify a main class in its manifest file you can specify another class name as the first argument.
Action on failure	Continue	What to do if the step fails.

Selecting hardware



Create Cluster - Advanced Options

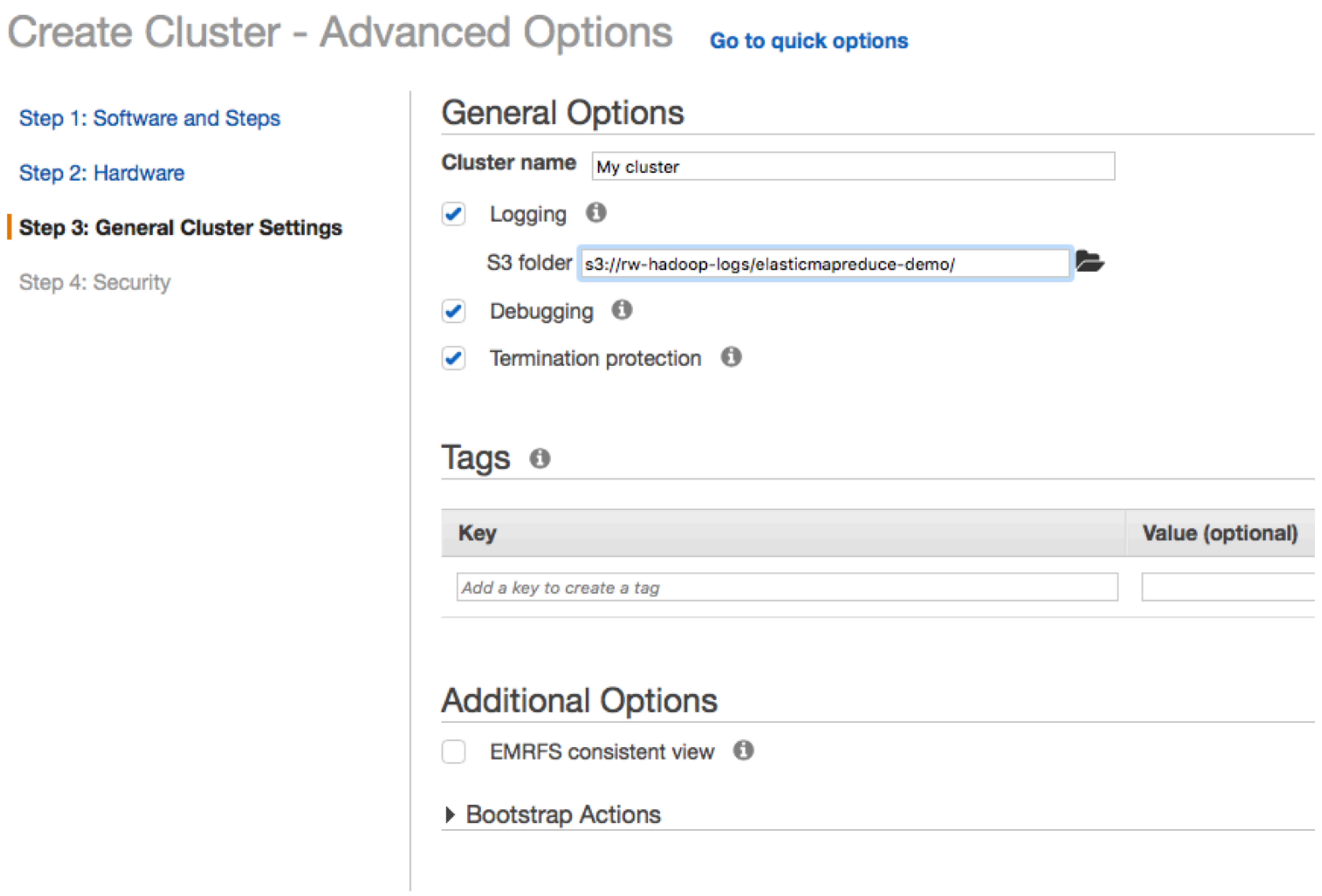
Hardware Configuration

If you need more than 20 EC2 instances, complete this form.

Network: vpc-9a87d8fe (172.31.0.0/16) (default)

EC2 Subnet: subnet-0f55196b | Default in us-west-2a

Type	Name	EC2 instance type	Instance count	Storage per instance	Request spot	Bid price
Master	Master instance group - 1	m3.xlarge	1	80 GiB Add EBS volumes	☐	
Core	Core instance group - 2	m3.xlarge	2	80 GiB Add EBS volumes	☐	
Task	Task instance group - 3	m3.xlarge	0	80 GiB Add EBS volumes	☐	



Create Cluster - Advanced Options

General Options

Cluster name: My cluster

[✓] Logging

S3 folder: s3://rw-hadoop-logs/elasticmapreduce-demo/

[✓] Debugging

[✓] Termination protection

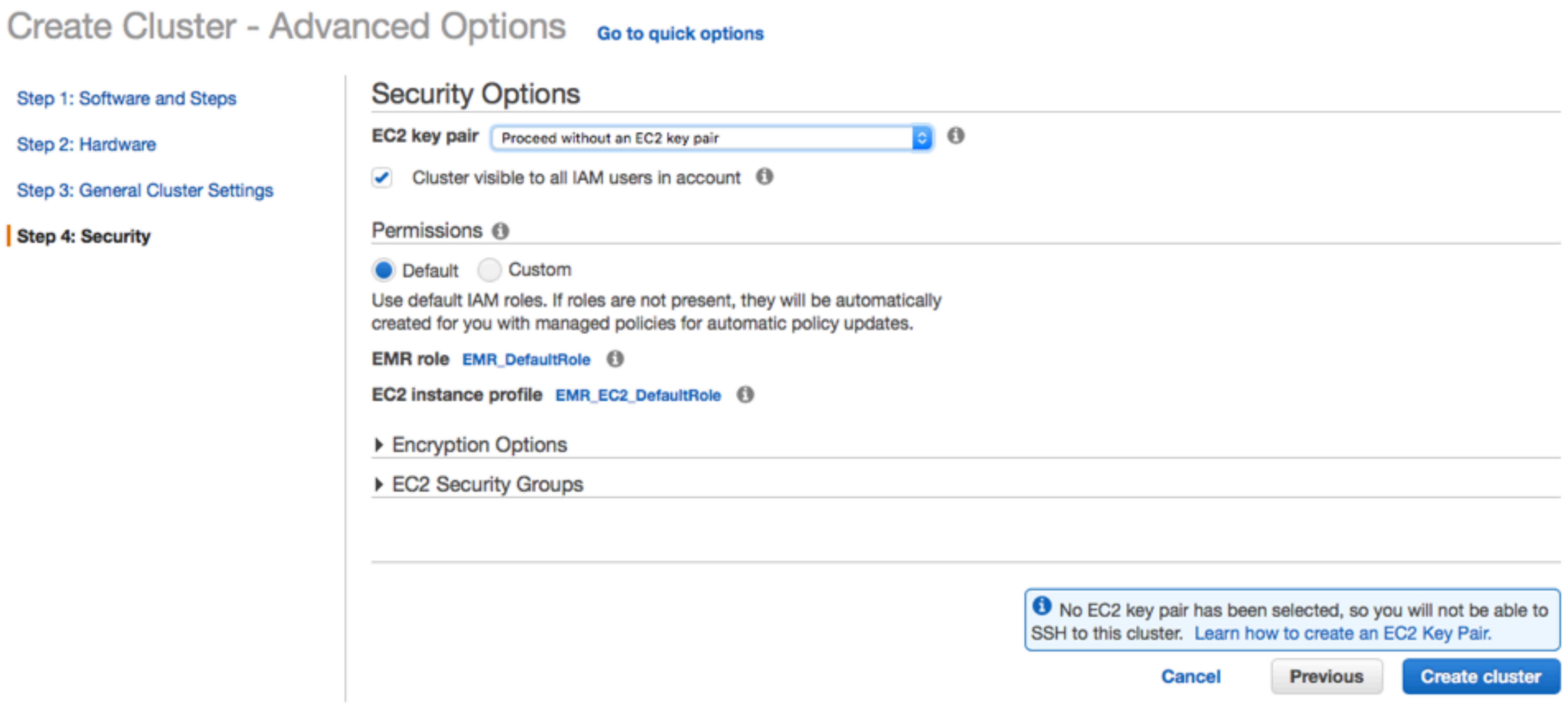
Tags

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

Additional Options

[ ] EMRFS consistent view

Security Options



Create Cluster - Advanced Options

Security Options

EC2 key pair: Proceed without an EC2 key pair

✓ Cluster visible to all IAM users in account

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.

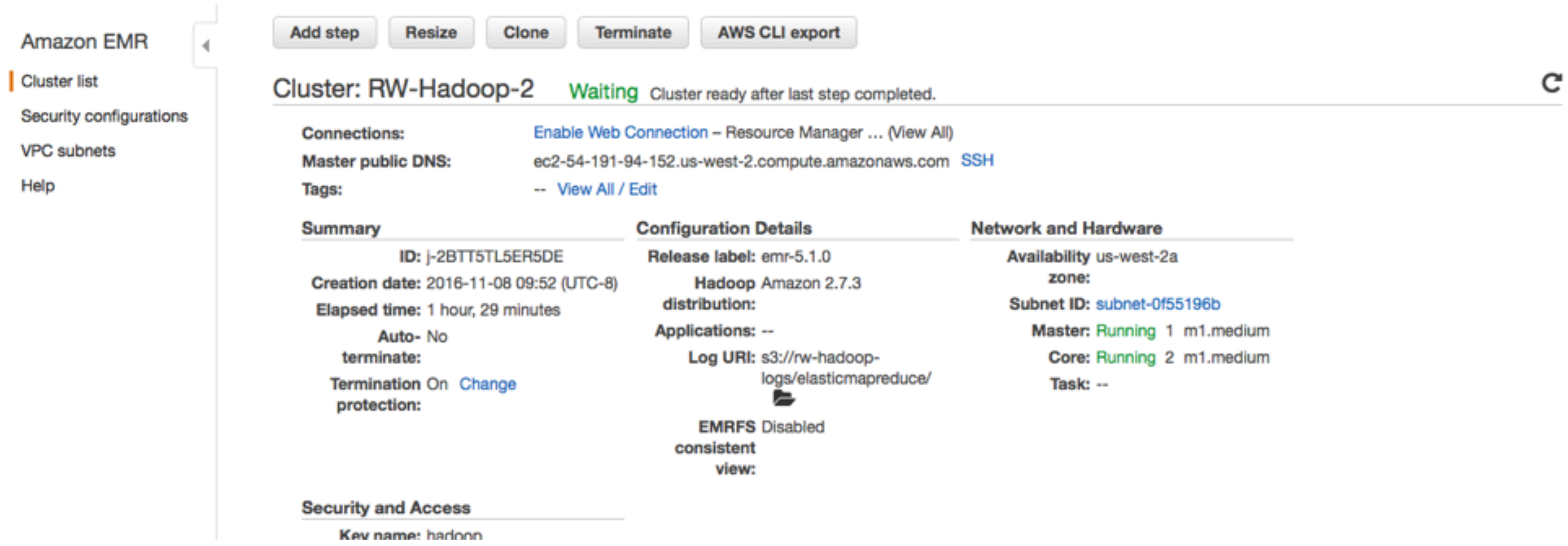
EMR role: EMR_DefaultRole

EC2 instance profile: EMR_EC2_DefaultRole

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.

With EC2 pair you can ssh into cluster

Monitoring Running Cluster



Cluster: RW-Hadoop-2 Waiting Cluster ready after last step completed.

Connections: Enable Web Connection – Resource Manager ... (View All)
Master public DNS: ec2-54-191-94-152.us-west-2.compute.amazonaws.com SSH
Tags: -- View All / Edit

Summary

ID:	j-2BTT5TL5ER5DE
Creation date:	2016-11-08 09:52 (UTC-8)
Elapsed time:	1 hour, 29 minutes
Auto-terminate:	No
Termination protection:	On Change

Configuration Details

Release label:	emr-5.1.0
Hadoop distribution:	Amazon 2.7.3
Applications:	--
Log URI:	s3://rw-hadoop-logs/elasticmapreduce/
EMRFS consistent view:	Disabled

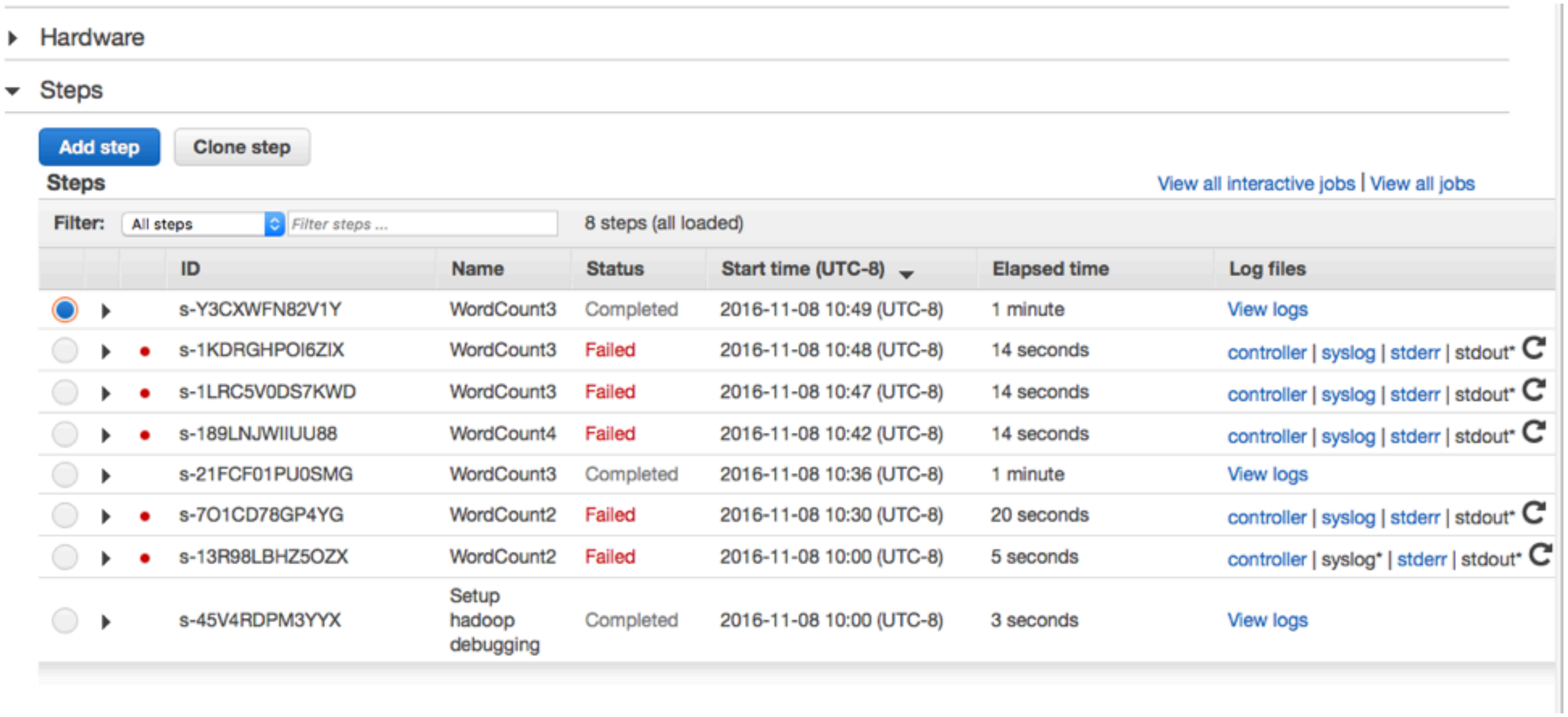
Network and Hardware

Availability zone:	us-west-2a
Subnet ID:	subnet-0f55196b
Master:	Running 1 m1.medium
Core:	Running 2 m1.medium
Task:	--

Security and Access

Key name: hadoop

Adding more steps (jobs)



Steps

8 steps (all loaded)

			ID	Name	Status	Start time (UTC-8)	Elapsed time	Log files
◉	▸		s-Y3CXWFN82V1Y	WordCount3	Completed	2016-11-08 10:49 (UTC-8)	1 minute	View logs
○	▸	●	s-1KDRGHPOI6ZIX	WordCount3	Failed	2016-11-08 10:48 (UTC-8)	14 seconds	controller | syslog | stderr | stdout*
○	▸	●	s-1LRC5V0DS7KWD	WordCount3	Failed	2016-11-08 10:47 (UTC-8)	14 seconds	controller | syslog | stderr | stdout*
○	▸	●	s-189LNJWIIUU88	WordCount4	Failed	2016-11-08 10:42 (UTC-8)	14 seconds	controller | syslog | stderr | stdout*
○	▸		s-21FCF01PU0SMG	WordCount3	Completed	2016-11-08 10:36 (UTC-8)	1 minute	View logs
○	▸	●	s-7O1CD78GP4YG	WordCount2	Failed	2016-11-08 10:30 (UTC-8)	20 seconds	controller | syslog | stderr | stdout*
○	▸	●	s-13R98LBHZ5OZX	WordCount2	Failed	2016-11-08 10:00 (UTC-8)	5 seconds	controller | syslog* | stderr | stdout*
○	▸		s-45V4RDPM3YYX	Setup hadoop debugging	Completed	2016-11-08 10:00 (UTC-8)	3 seconds	View logs

Example Input & Output

Input

file01

```
Hello World Bye World
```

file02

```
Hello Hadoop Goodbye Hadoop
```

part-r-00000

```
Hello     2
```

part-r-00001

```
Bye                    1
Goodbye     1
```

part-r-00002

```
Hadoop 2
World    2
```

Hadoop MapReduce v2 Cookbook 2nd Edition

Contains instructions with are mainly correct for various operations

To Do list

Selecting Scheduler
Chaining Jobs
Different types of input
Basic MapReduce Algorithm Design
- Pairs & Stripes
- Secondary Sorting

Performance issues