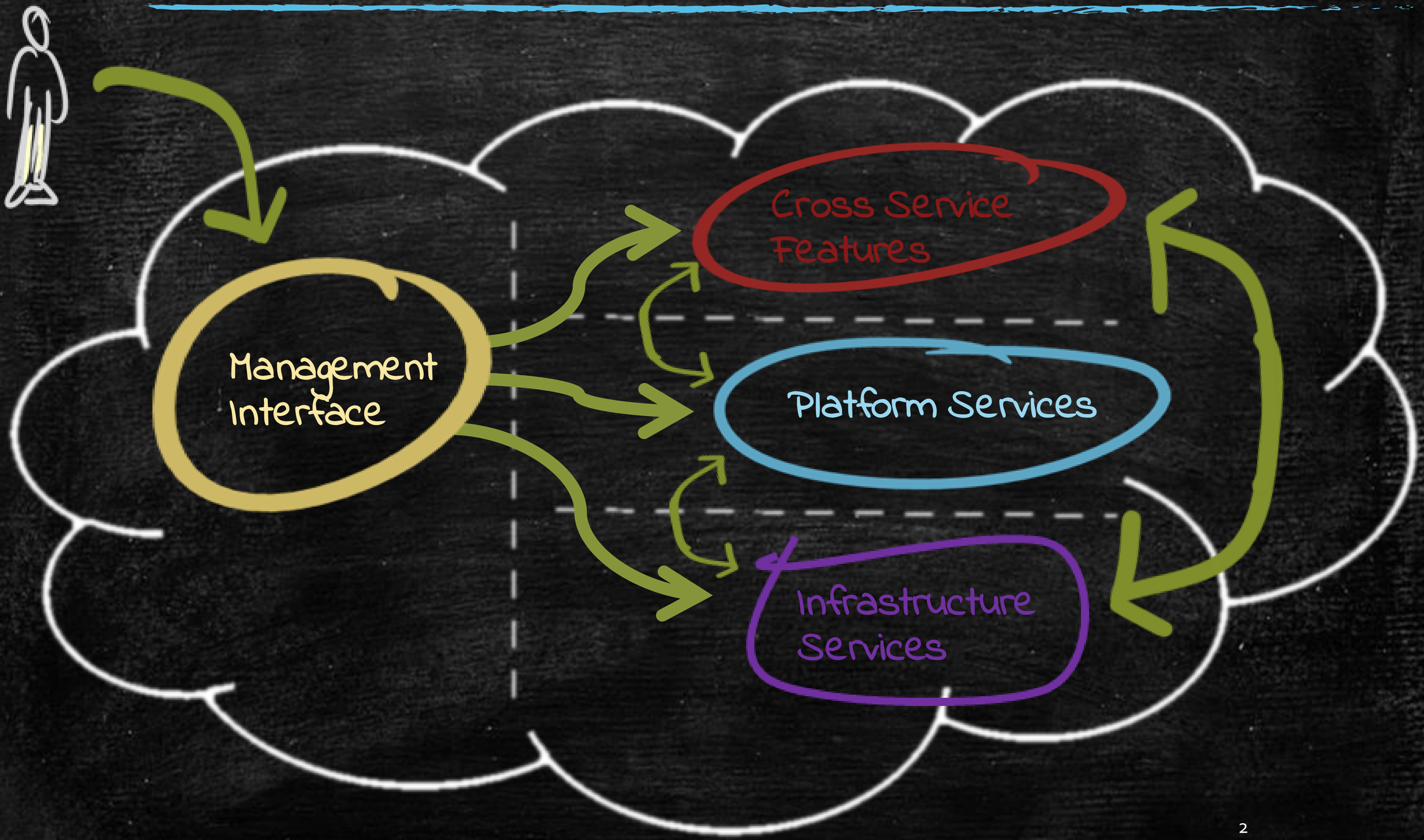
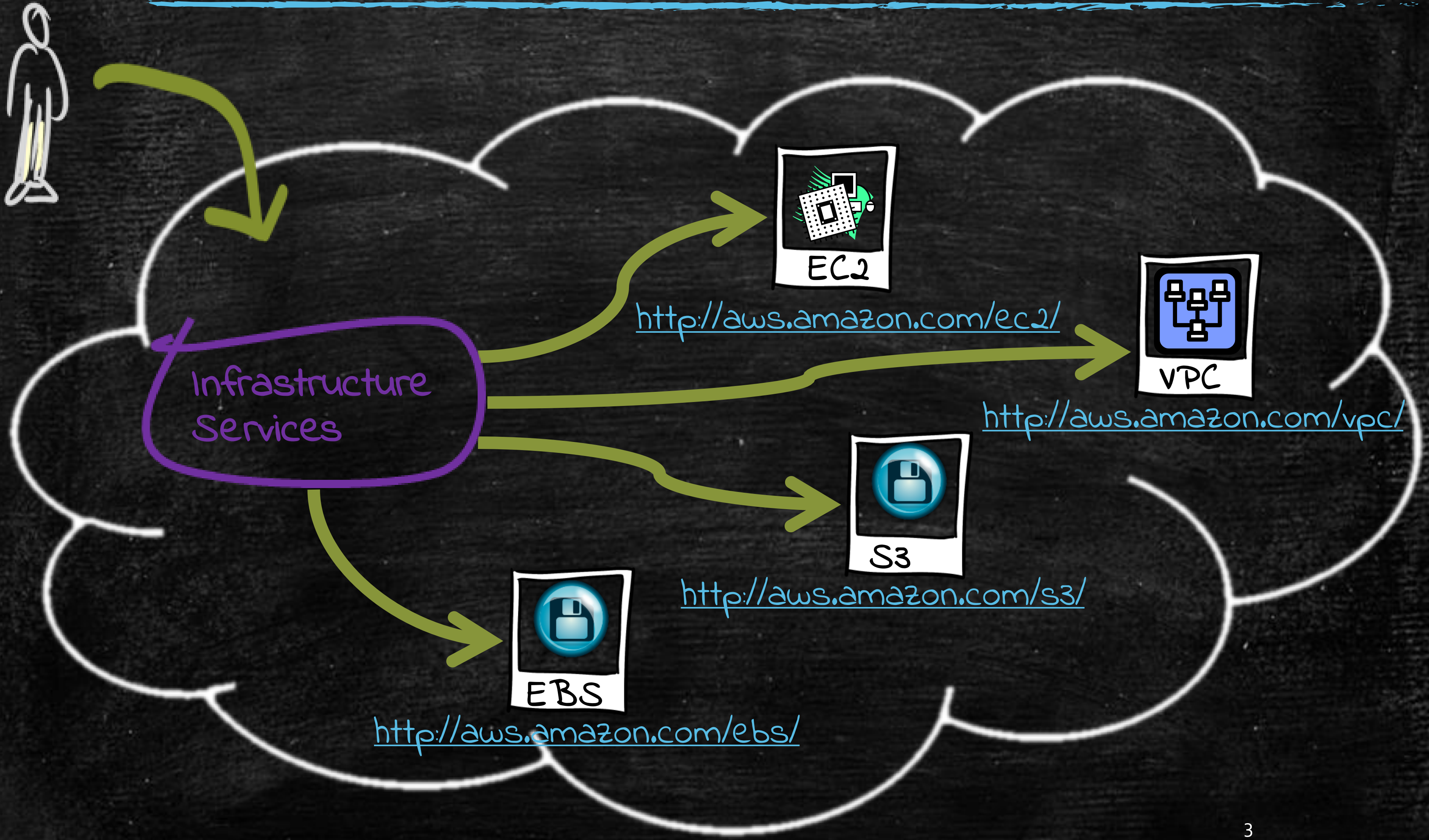
Elastic Map Reduce

Shadi Khalifa
Database Systems Laboratory (DSL)
khalifa@cs.queensu.ca

The Amazon Web Services Universe



Infrastructure Services



http://aws.amazon.com/ec2/

http://aws.amazon.com/vpc/

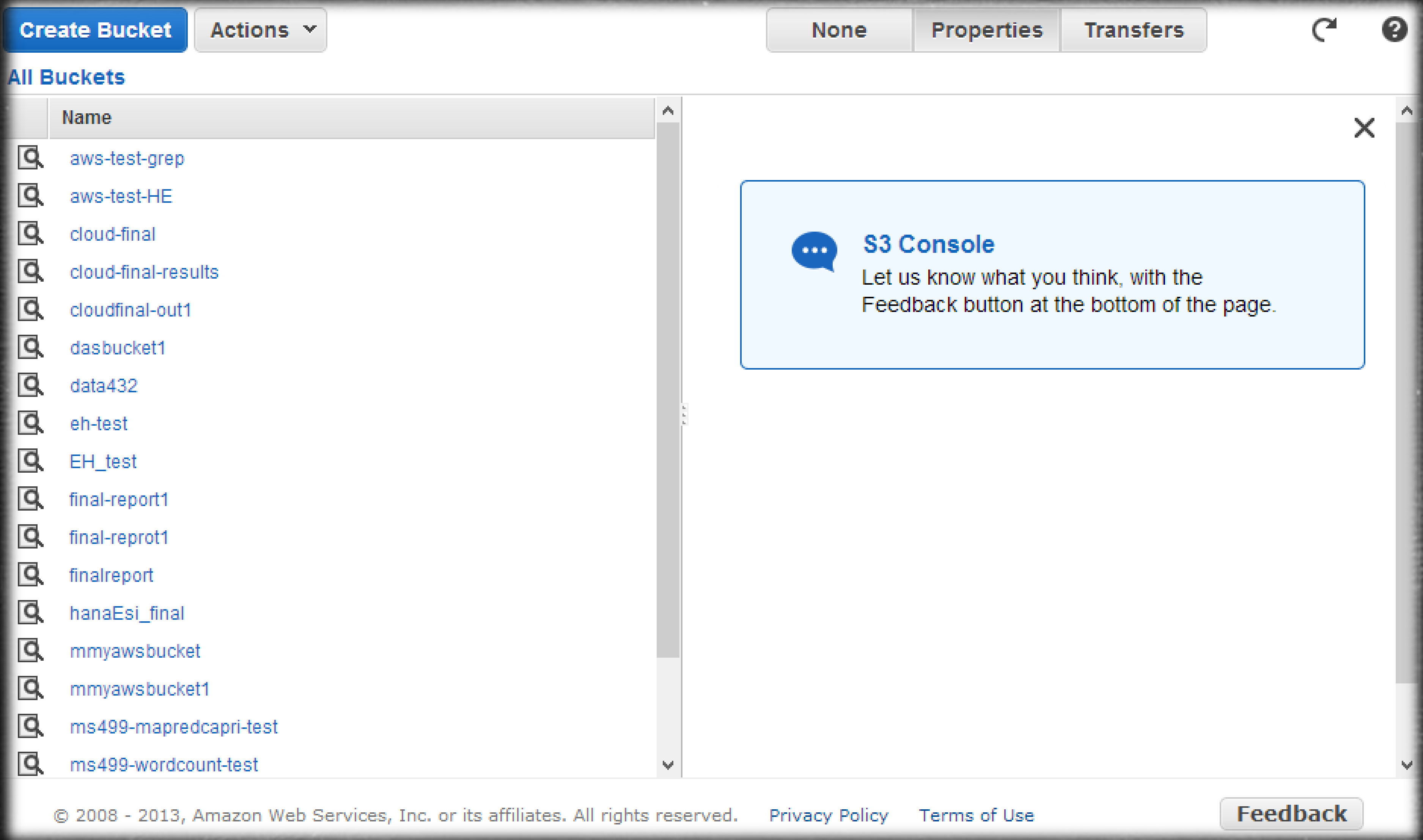
http://aws.amazon.com/s3/

http://aws.amazon.com/ebs/

S3
Scalable Storage in the Cloud

Amazon Simple Storage Service (S3)

- Amazon S3 provides a simple web services interface that can be used to store and retrieve any amount of data, at any time, from anywhere on the web.
- write, read, and delete objects containing from 1 byte to 5 terabytes of data each. The number of objects you can store is unlimited.
- Each object is stored in a bucket and retrieved via a unique, developer-assigned key.
  - A bucket can be stored in one of several Regions.
  - You can choose a Region to optimize for latency, minimize costs, or address regulatory requirements.
  - objects stored in a Region never leave the Region unless you transfer them out.
- Authentication mechanisms are provided to ensure that data is kept secure from unauthorized access.
  - objects can be made private or public, and rights can be granted to specific users.
- S3 charges based on per GB-month AND per I/O requests AND per data modification requests.

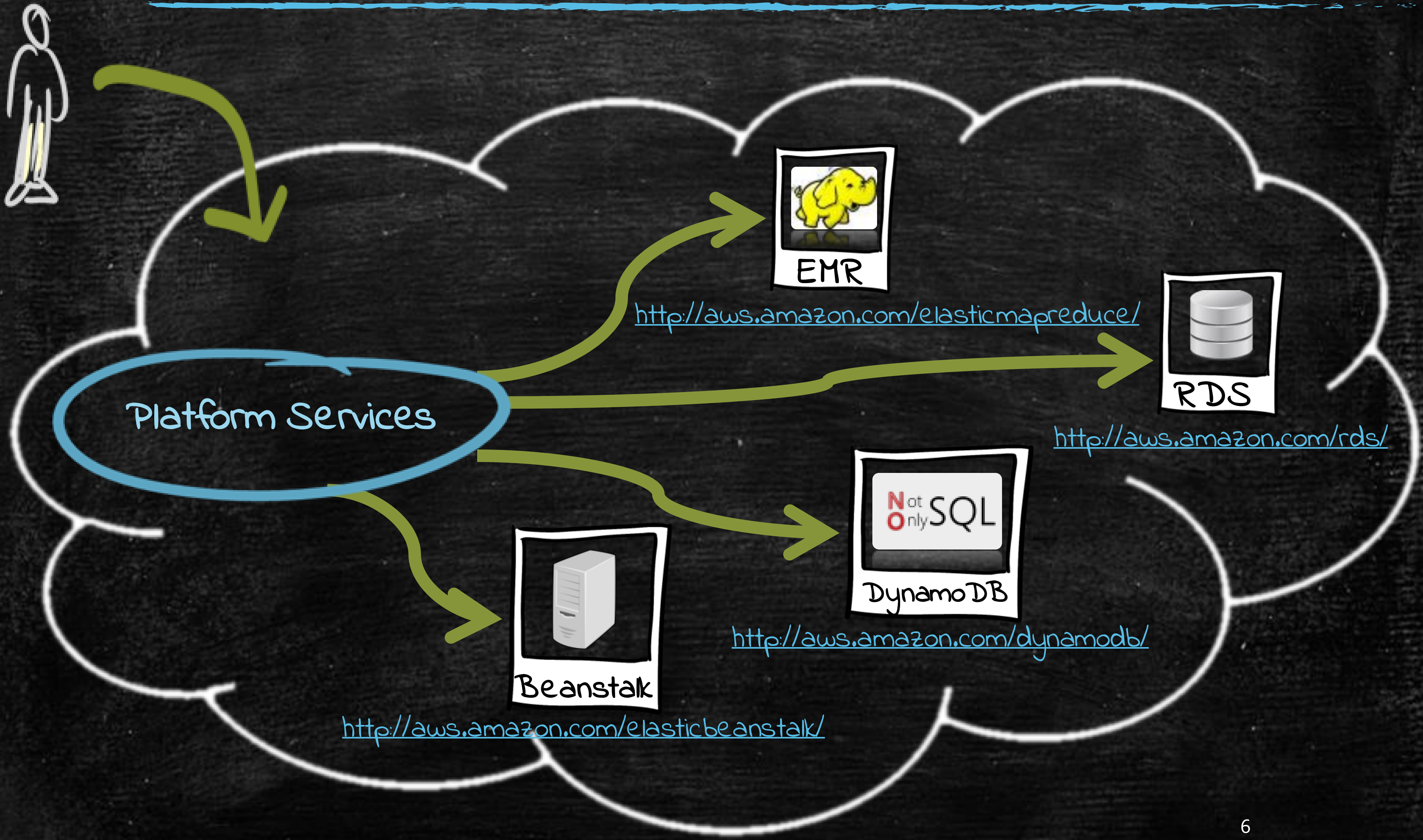


aws-test-grep
aws-test-HE
cloud-final
cloud-final-results
cloudfinal-out1
dasbucket1
data432
eh-test
EH_test
final-report1
final-reprot1
finalreport
hanaEsi_final
mmyawsbucket
mmyawsbucket1
ms499-mapredcapri-test
ms499-wordcount-test

S3 Console
Let us know what you think, with the Feedback button at the bottom of the page.

© 2008 - 2013, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Platform Services

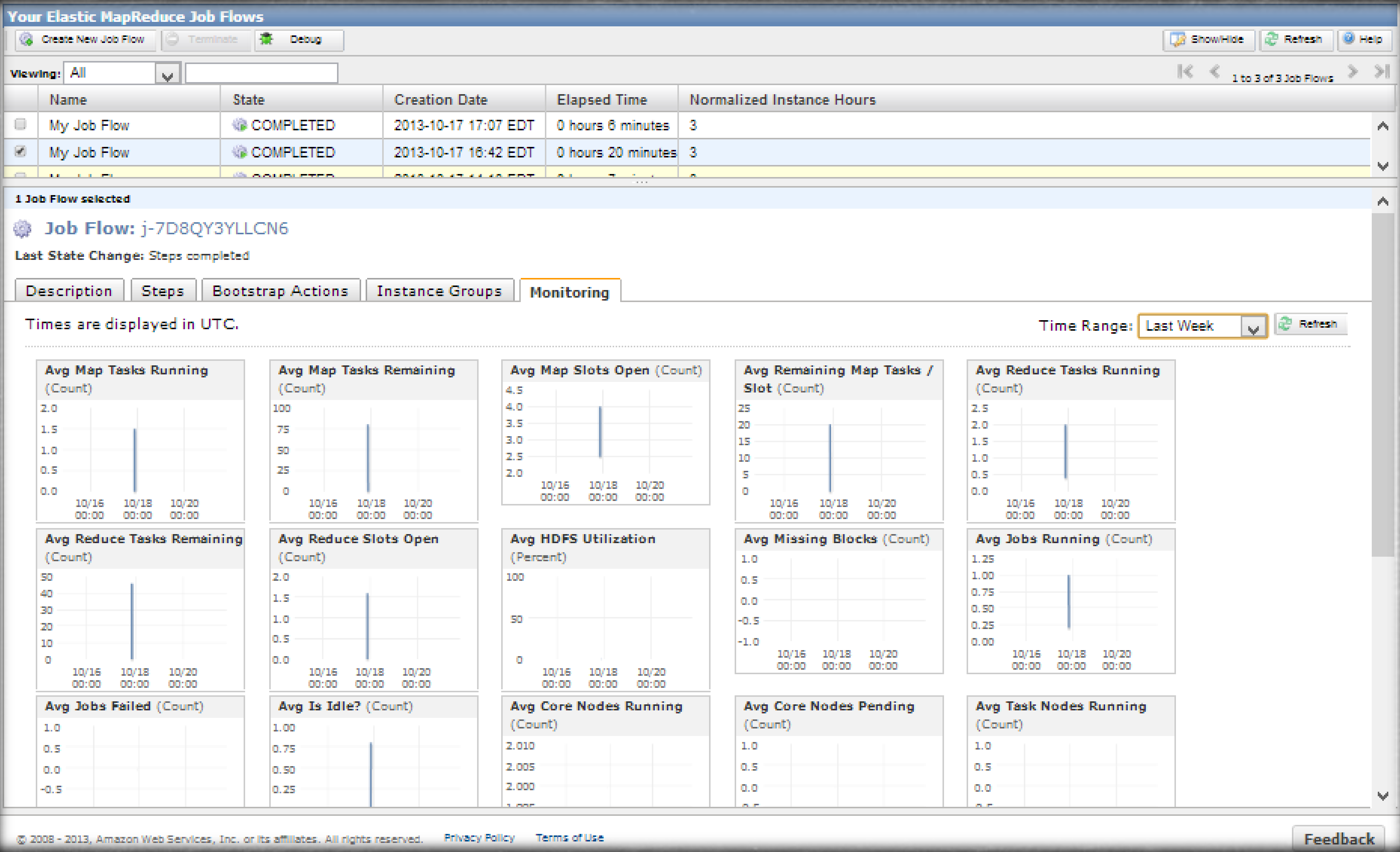

Platform Services
EMR
http://aws.amazon.com/elasticmapreduce/
RDS
http://aws.amazon.com/rds/
Not Only SQL
DynamoDB
http://aws.amazon.com/dynamodb/
Beanstalk
http://aws.amazon.com/elasticbeanstalk/

Platform Services

Elastic MapReduce
Managed Hadoop Framework

Amazon Elastic MapReduce (EMR)

- Amazon EMR is a web service that makes it easy to quickly and cost-effectively process vast amounts of data using Hadoop.
- Amazon EMR distribute the data and processing across a resizable cluster of Amazon EC2 instances.
- With Amazon EMR you can launch a persistent cluster that stays up indefinitely or a temporary cluster that terminates after the analysis is complete.
- Amazon EMR supports a variety of Amazon EC2 instance types and Amazon EC2 pricing options (On-Demand, Reserved, and Spot).
- When launching an Amazon EMR cluster (also called a "job flow"), you choose how many and what type of Amazon EC2 Instances to provision.
- The Amazon EMR price is in addition to the Amazon EC2 price.
- Amazon EMR is used in a variety of applications, including log analysis, web indexing, data warehousing, machine learning, financial analysis, scientific simulation, and bioinformatics.



Your Elastic MapReduce Job Flows
Create New Job Flow
Terminate
Debug
Show/Hide
Refresh
Help
Viewing: All
1 to 3 of 3 Job Flows

	Name	State	Creation Date	Elapsed Time	Normalized Instance Hours
☐	My Job Flow	COMPLETED	2013-10-17 17:07 EDT	0 hours 6 minutes	3
☑	My Job Flow	COMPLETED	2013-10-17 16:42 EDT	0 hours 20 minutes	3

1 Job Flow selected

Job Flow: j-7D8QY3YLLCN6

Last State Change: Steps completed

Description
Steps
Bootstrap Actions
Instance Groups
Monitoring

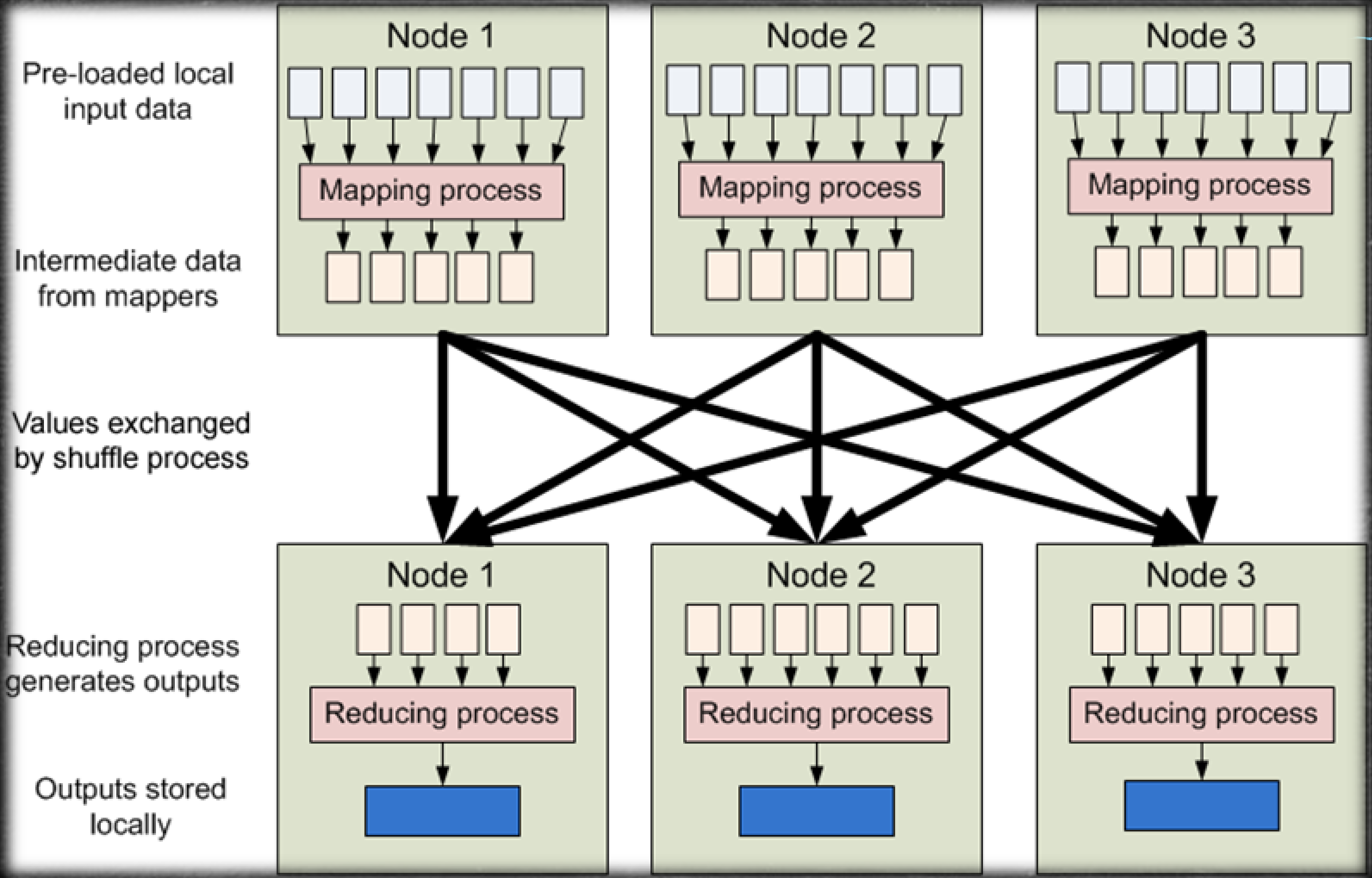
Times are displayed in UTC.

Time Range: Last Week

Refresh
Avg Map Tasks Running (Count)
Avg Map Tasks Remaining (Count)
Avg Map Slots Open (Count)
Avg Remaining Map Tasks / Slot (Count)
Avg Reduce Tasks Running (Count)
Avg Reduce Tasks Remaining (Count)
Avg Reduce Slots Open (Count)
Avg HDFS Utilization (Percent)
Avg Missing Blocks (Count)
Avg Jobs Running (Count)
Avg Jobs Failed (Count)
Avg Is Idle? (Count)
Avg Core Nodes Running (Count)
Avg Core Nodes Pending (Count)
Avg Task Nodes Running (Count)
10/16 00:00
10/18 00:00
10/20 00:00

© 2008 - 2013, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

What is Hadoop?



Example: Word Count

- Objective: Count the number of occurrences of each word in the provided input files.
- How it works
  - In the Map phase the text is tokenized into words then we form a key value pair with these words where the key being the word itself and value is set to '1'.
  - In the reduce phase the keys are grouped together and the values for the same key are added.

Input Files

Apple Orange Mango
Orange Grapes Plum

Apple Plum Mango
Apple Apple Plum

Example 2: Joins with MapReduce

UserDetails.txt

123 456, Jim
456 123, Tom
789 123, Harry
789 456, Richa

DeliveryDetails.txt

123 456, Delivered
456 123, Pending
789 123, Failed
789 456, Resend

Expected Output

Jim, Delivered
Tom, Pending
Harry, Failed
Richa, Resend

- We have 2 input files as follows:
  - UserDetails.txt : Every record is of the format 'mobile number , consumer name'.
  - DeliveryDetails.txt: Every record is of the format 'mobile number, delivery status'.
- Objective: Associate the customer name with the delivery status.

Formulating as a MapReduce

files, we need 2 Map functions (one

er

cu r N

ber

- Value: 'DR' for the

UserDetails.txt

<123 456, CD~Jim>
<456 123, CD~Tom>
<789 123, CD~Harry>
<789 456, CD~Richa>

DeliveryDetails.txt

<123 456, DR~Delivered>
<456 123, DR~Pending>
<789 123, DR~Failed>
<789 456, DR~Resend>

Reduce Input

<123 456, CD~Jim>
<123 456, DR~Delivered>

<456 123, CD~Tom>
<456 123, DR~Pending>

Reduce Output

<Jim, Delivered>

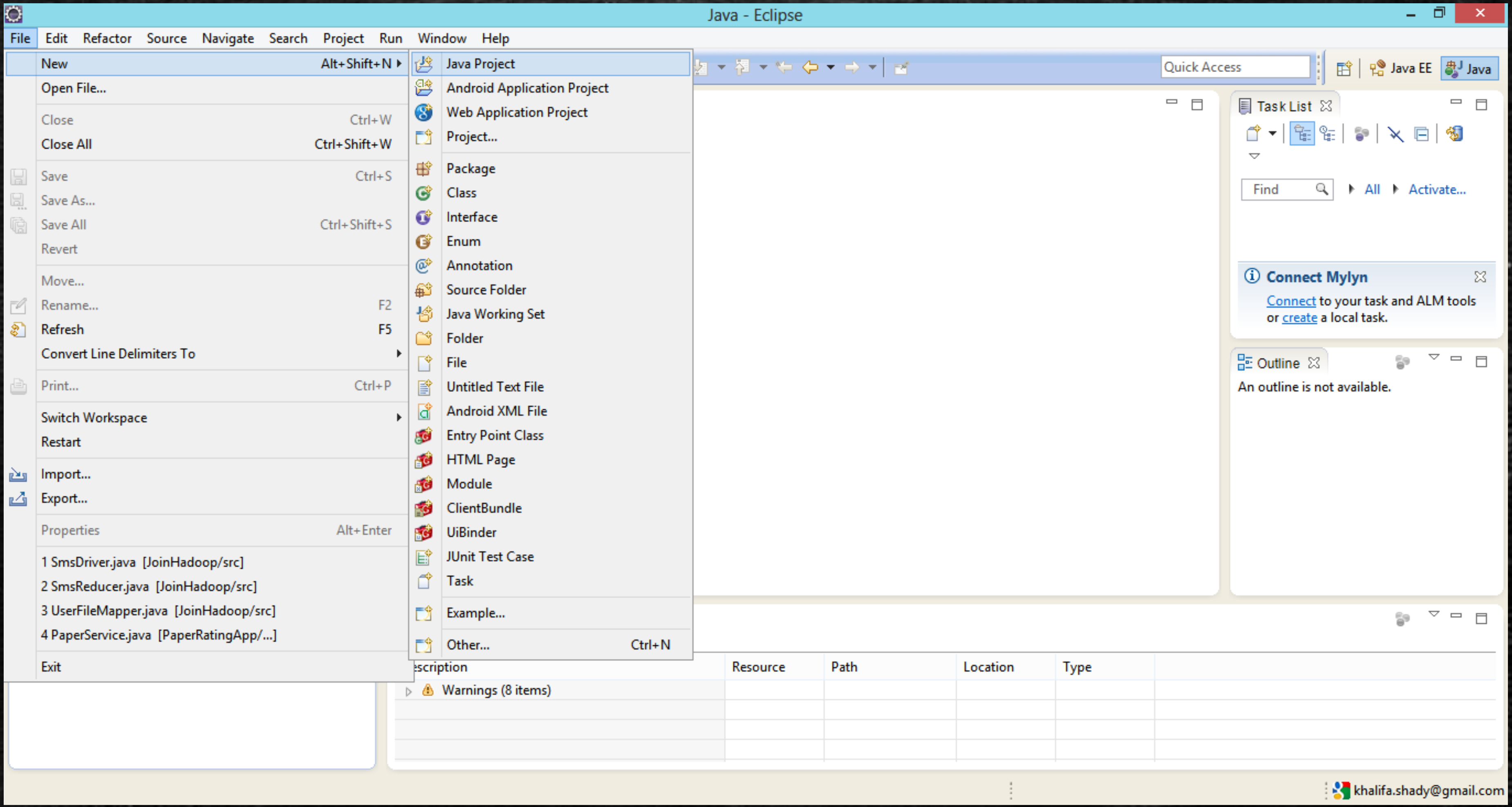
<Tom, Pending>

- On the reducer, every s (for simplicity) one with prefix 'CD' and other 'DR'.
- From CD get the customer name corresponding to the cell number (input key) and from DR get the status.
- The output Key values from the reducer would be as follows
  - Key : Customer Name
  - Value : Status Message

Tools you will need

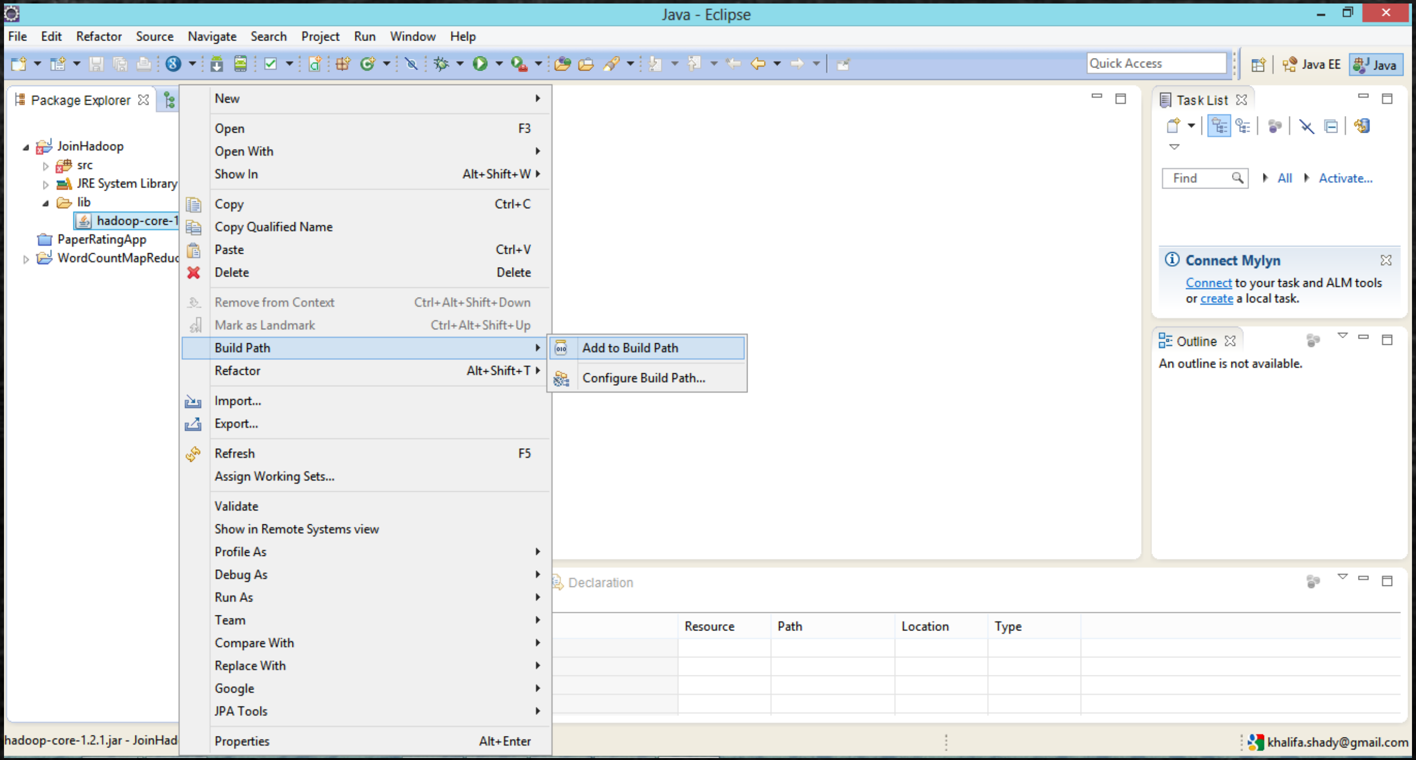
- Eclipse IDE for Java EE Developers
  - http://www.eclipse.org/downloads/packages/eclipse-ide-java-ee-developers/keplersr1
- hadoop-core-1.2.1.jar
  - http://mvnrepository.com/artifact/org.apache.hadoop/hadoop-core/1.2.1
- Amazon web Services Account
  - https://console.aws.amazon.com/elasticmapreduce/home?region=us-east-1

Create a new Java Project



Add Hadoop jar to the project

- Create a 'lib' folder in the project.
- Copy and paste the Hadoop jar into the 'lib' folder.
- Add the jar to the project build path.



Map classes: 1) UserFileMapper

```
public class UserFileMapper extends MapReduceBase implements Mapper<LongWritable, Text,
Text, Text>
{
    //varia                    er Details
    private                    tomerName,fileTag="CD~";

    /* map                     nsu                        me
pairs
       Key(
       Valu                    to indicate the source of                       e
customer de                    Name
     */
    public void map(LongWritable key, Text value, OutputCollector<Text, Text> output,
Reporter reporter) throws IOException
    {
       //taking one line/record at a time and parsing them into key value pairs
        String line = value.toString();
        String splitarray[] = line.split(",");
        cellNumber = splitarray[0].trim();
        customerName = splitarray[1].trim();

      //sending the key value pair out of mapper
        output.collect(new Text(cellNumber), new Text(fileTag+customerName));
     }
}
```

UserDetails.txt

123 456, Jim
456 123, Tom
789 123, Harry
789 456, Richa

UserDetails.txt

<123 456, CD~Jim>
<456 123, CD~Tom>
<789 123, CD~Harry>
<789 456, CD~Richa>

Map classes: 2) DeliveryFileMapper

```
public class DeliveryFileMapper extends MapReduceBase implements Mapper<LongWritable,
Text, Text, Text>
{
    //                              ry report
    pr                              iveryCode,fileTag="DR~";

   /*                               iv                       e
    Ke
    Va                              o indicate the source of
delive                              de*/

    public void map(LongWritable key, Text value, OutputCollector<Text, Text> output,
Reporter reporter) throws IOException
    {
       //taking one line/record at a time and parsing them into key value pairs
        String line = value.toString();
        String splitarray[] = line.split(",");
        cellNumber = splitarray[0].trim();
        deliveryCode = splitarray[1].trim();

        //sending the key value pair out of mapper
        output.collect(new Text(cellNumber), new Text(fileTag+deliveryCode));
     }
}
```

DeliveryDetails.txt

123 456, Delivered
456 123, Pending
789 123, Failed
789 456, Resend

DeliveryDetails.txt

<123 456, DR~Delivered>
<456 123, DR~Pending>
<789 123, DR~Failed>
<789 456, DR~Resend>

Reduce Class

```
publi                         apReduceBase implements Reducer<Text, Text, Text, Text> {

                              n process
                              e.deliveryReport;



                              y, Iterator<Text> values,
outpu                          IOException
    {
        while (values.hasNext())
        {
             String currValue = values.next().toString();
             String valueSplitted[] = currValue.split("~");
             /*identifying the record source that corresponds to a cell number
             and parses the values accordingly*/
             if(valueSplitted[0].equals("CD"))
             {
               customerName=valueSplitted[1].trim();
             }
             else if(valueSplitted[0].equals("DR"))
             {
               deliveryReport = valueSplitted[1].trim();
             }
        }
          output.collect(new Text(customerName), new Text(deliveryReport));
    }
}
```

Reduce Input

<123 456, CD~Jim>

<123 456, DR~Delivered>

<456 123, CD~Tom>

<456 123, DR~Pending>

Reduce Output

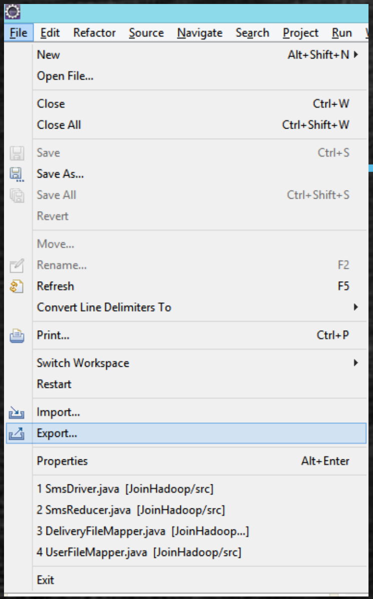
<Jim, Delivered>

<Tom, Pending>

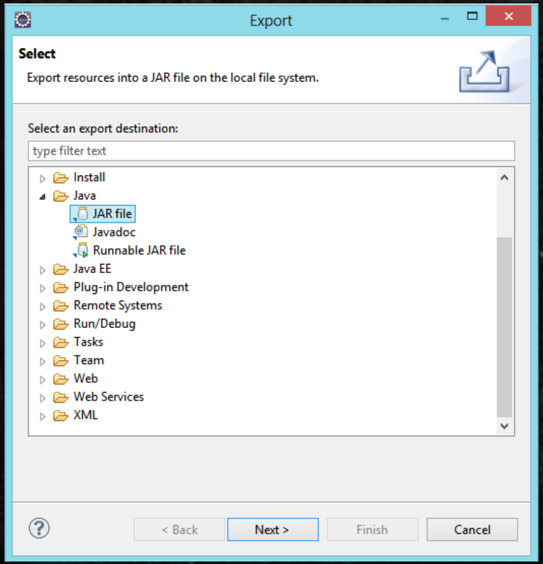
Driver Class

```
public class SmsDriver extends Configured implements Tool
{
       public int run(String[] args) throws Exception {

              //get the configuration parameters and assigns a job name
              JobConf conf = new JobConf(getConf(), SmsDriver.class);
              conf.setJobName("SMS Reports");

              //setting key value types for mapper and reducer outputs
              conf.setOutputKeyClass(Text.class);
              conf.setOutputValueClass(Text.class);

              //specifying the custom reducer class
              conf.setReducerClass(SmsReducer.class);

// If only one Mapper exists
  //   conf.setMapperClass(Mapper.class);
  //   FileInputFormat.addInputPath(conf, newPath(args[0]));
//Specifying the input directories(@ runtime) and Mappers independently for inputs from multiple sources
    MultipleInputs.addInputPath(conf, new Path(args[0]), TextInputFormat.class, UserFileMapper.class);
    MultipleInputs.addInputPath(conf, new Path(args[1]), TextInputFormat.class, DeliveryFileMapper.class);

              //Specifying the output directory @ runtime
              FileOutputFormat.setOutputPath(conf, new Path(args[2]));

              JobClient.runJob(conf);
              return 0;
       }

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

Export Project to Jar



File Edit Refactor Source Navigate Search Project Run
New Alt+Shift+N
Open File...
Close Ctrl+W
Close All Ctrl+Shift+W
Save Ctrl+S
Save As...
Save All Ctrl+Shift+S
Revert
Move...
Rename... F2
Refresh F5
Convert Line Delimiters To
Print... Ctrl+P
Switch Workspace
Restart
Import...
Export...
Properties Alt+Enter
1 SmsDriver.java [JoinHadoop/src]
2 SmsReducer.java [JoinHadoop/src]
3 DeliveryFileMapper.java [JoinHadoop...]
4 UserFileMapper.java [JoinHadoop/src]
Exit


Export

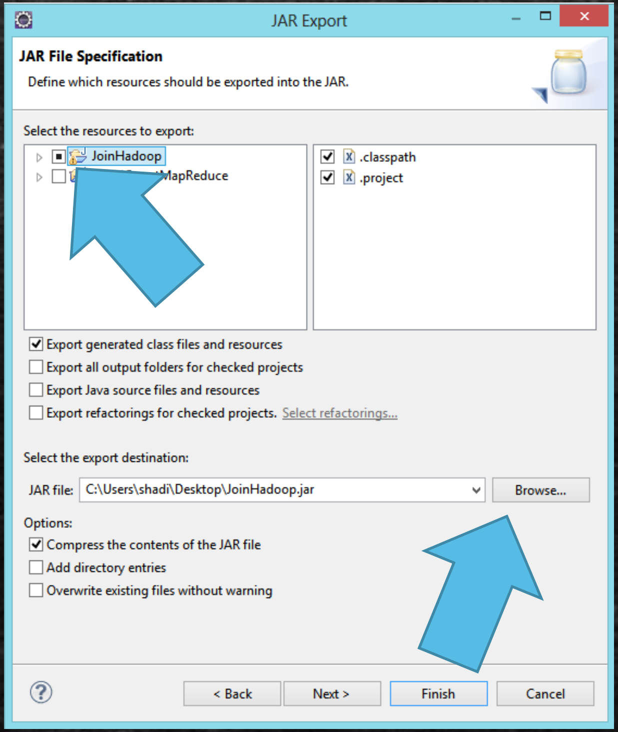
Select

Export resources into a JAR file on the local file system.

Select an export destination:

type filter text

- Install
- Java
  - JAR file
  - Javadoc
  - Runnable JAR file
- Java EE
- Plug-in Development
- Remote Systems
- Run/Debug
- Tasks
- Team
- Web
- Web Services
- XML


JAR Export

JAR File Specification

Define which resources should be exported into the JAR.

Select the resources to export:

- ■ JoinHadoop
- ☐ MapReduce
- ✓ .classpath
- ✓ .project

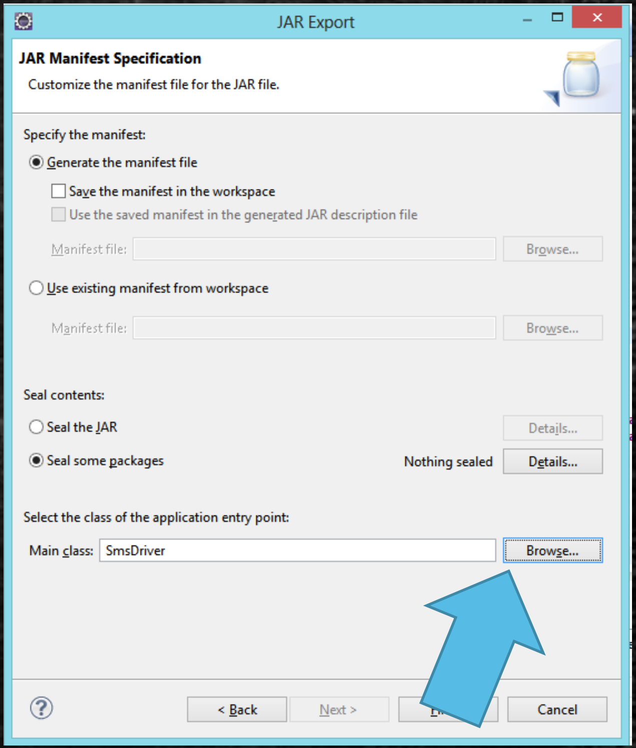
- ✓ Export generated class files and resources
- ☐ Export all output folders for checked projects
- ☐ Export Java source files and resources
- ☐ Export refactorings for checked projects. Select refactorings...

Select the export destination:

JAR file: C:\Users\shadi\Desktop\JoinHadoop.jar

Options:

- ✓ Compress the contents of the JAR file
- ☐ Add directory entries
- ☐ Overwrite existing files without warning


JAR Export

JAR Manifest Specification

Customize the manifest file for the JAR file.

Specify the manifest:

- ● Generate the manifest file
  - ☐ Save the manifest in the workspace
  - ☐ Use the saved manifest in the generated JAR description file
  - Manifest file:
- ○ Use existing manifest from workspace
  - Manifest file:

Seal contents:

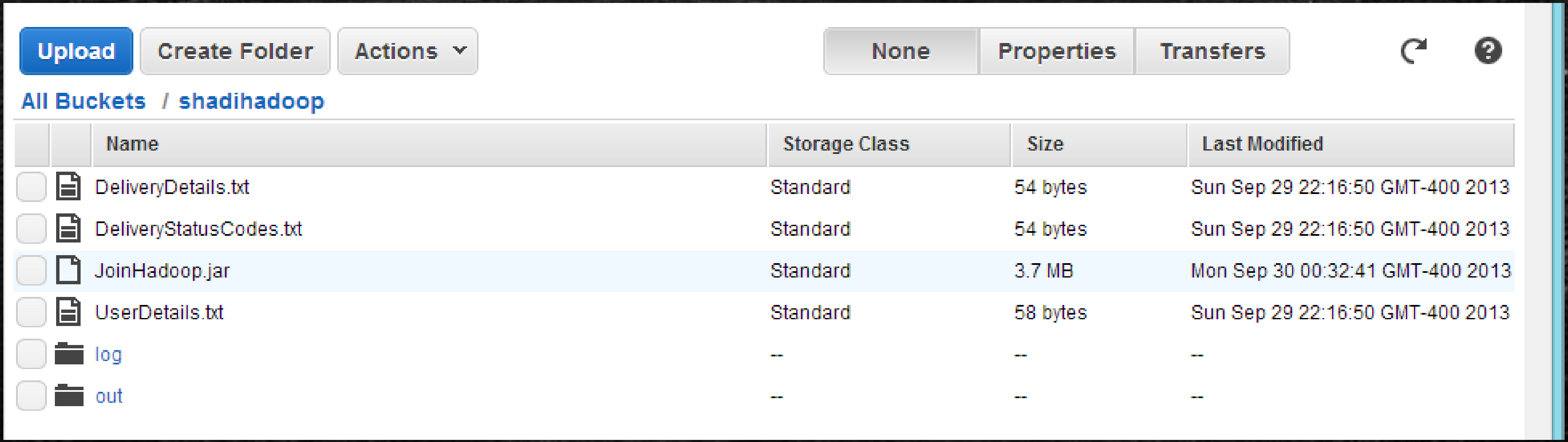
- ○ Seal the JAR
- ● Seal some packages — Nothing sealed

Select the class of the application entry point:

Main class: SmsDriver

Uploading Project Jar and Data to S3

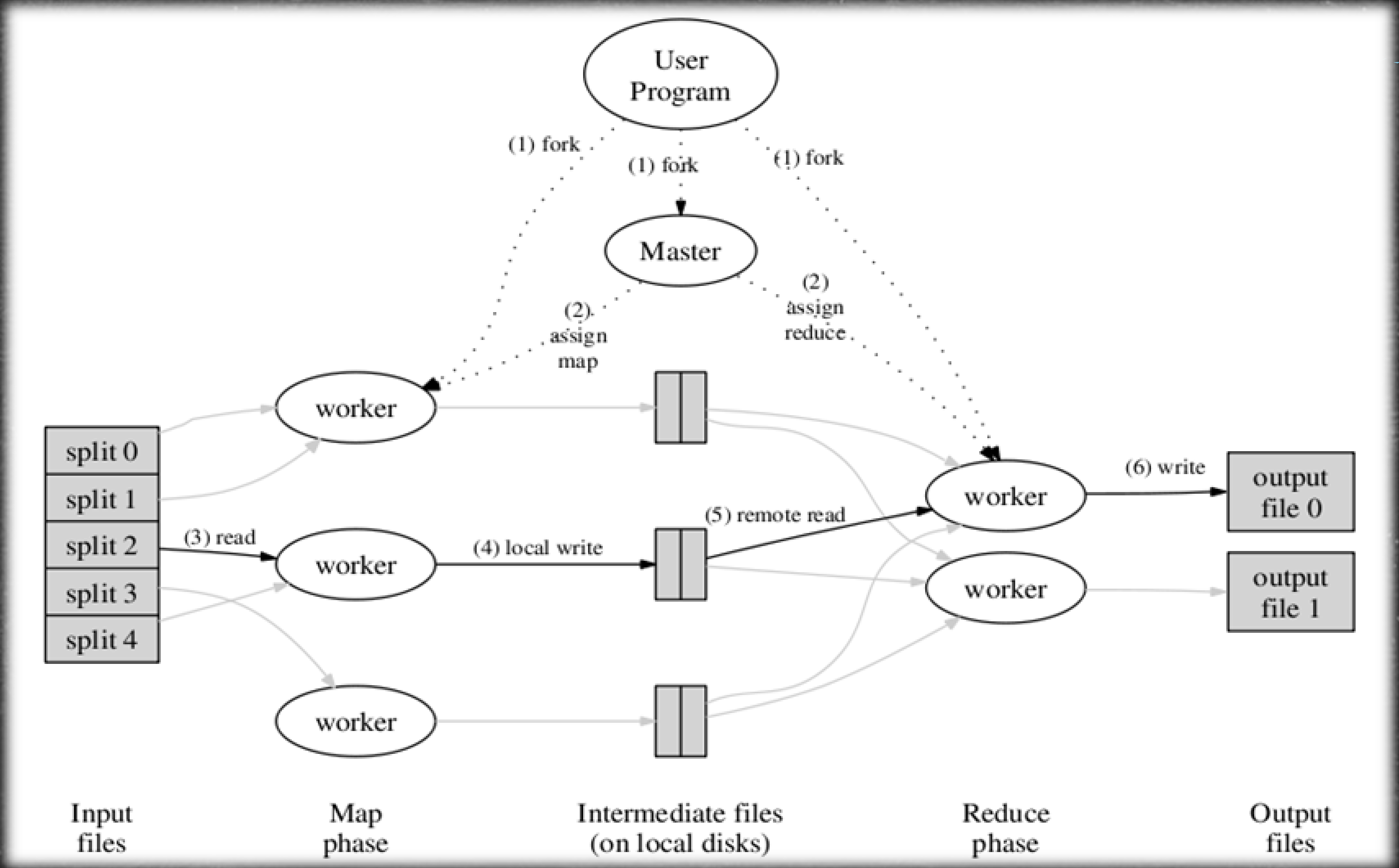
- https://console.aws.amazon.com/s3/home?region=us-east-1



All Buckets / shadihadoop

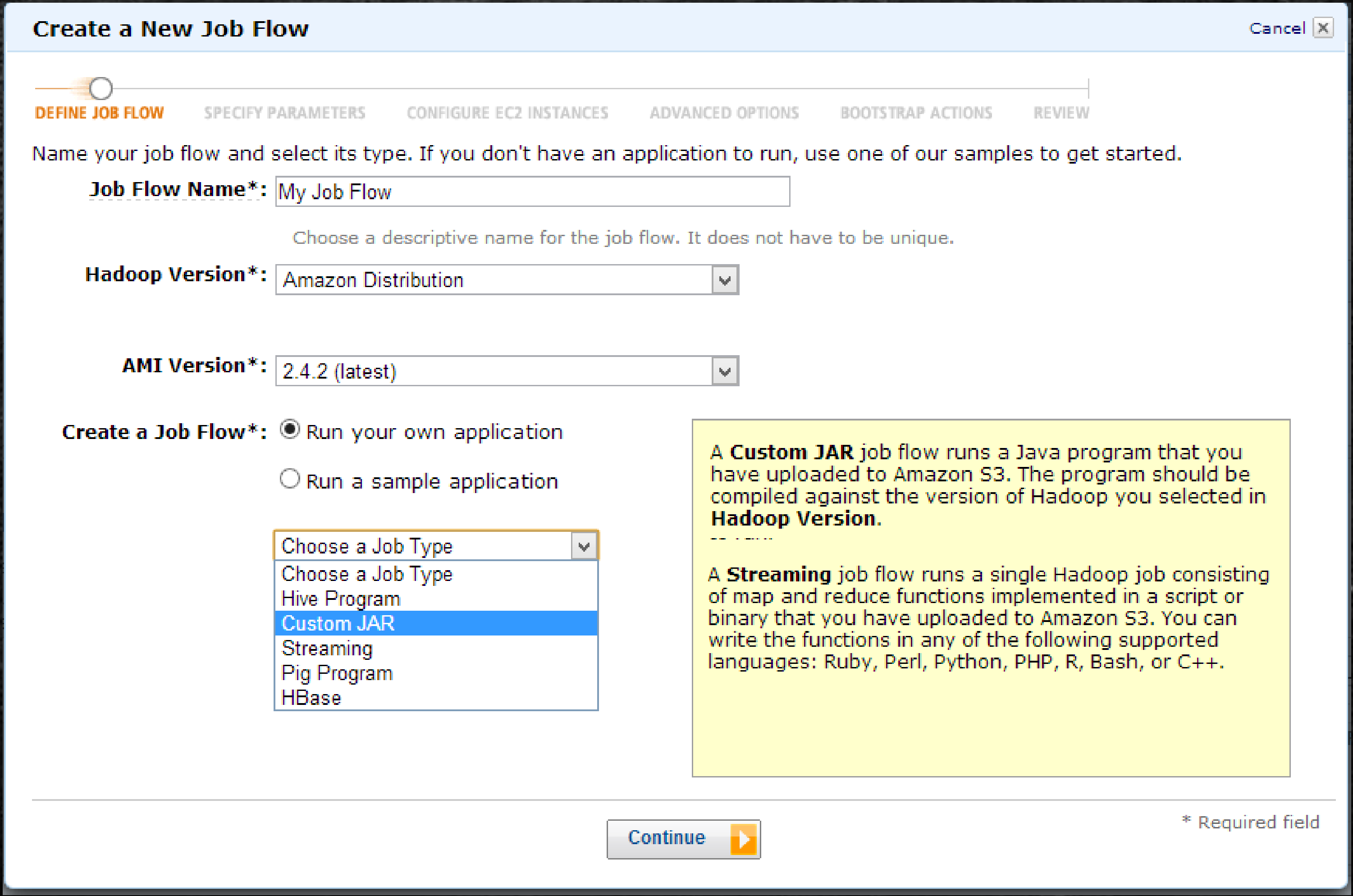
Name	Storage Class	Size	Last Modified
DeliveryDetails.txt	Standard	54 bytes	Sun Sep 29 22:16:50 GMT-400 2013
DeliveryStatusCodes.txt	Standard	54 bytes	Sun Sep 29 22:16:50 GMT-400 2013
JoinHadoop.jar	Standard	3.7 MB	Mon Sep 30 00:32:41 GMT-400 2013
UserDetails.txt	Standard	58 bytes	Sun Sep 29 22:16:50 GMT-400 2013
log	--	--	--
out	--	--	--

Hadoop Execution overview



Using Amazon EMR

https://console.aws.amazon.com/elasticmapreduce/home?region=us-east-1



Name your job flow and select its type. If you don't have an application to run, use one of our samples to get started.

Job Flow Name*: My Job Flow

Choose a descriptive name for the job flow. It does not have to be unique.

Hadoop Version*: Amazon Distribution

AMI Version*: 2.4.2 (latest)

Create a Job Flow*: [✓ Run your own application] Run a sample application

Choose a Job Type
- Choose a Job Type
- Hive Program
- Custom JAR
- Streaming
- Pig Program
- HBase

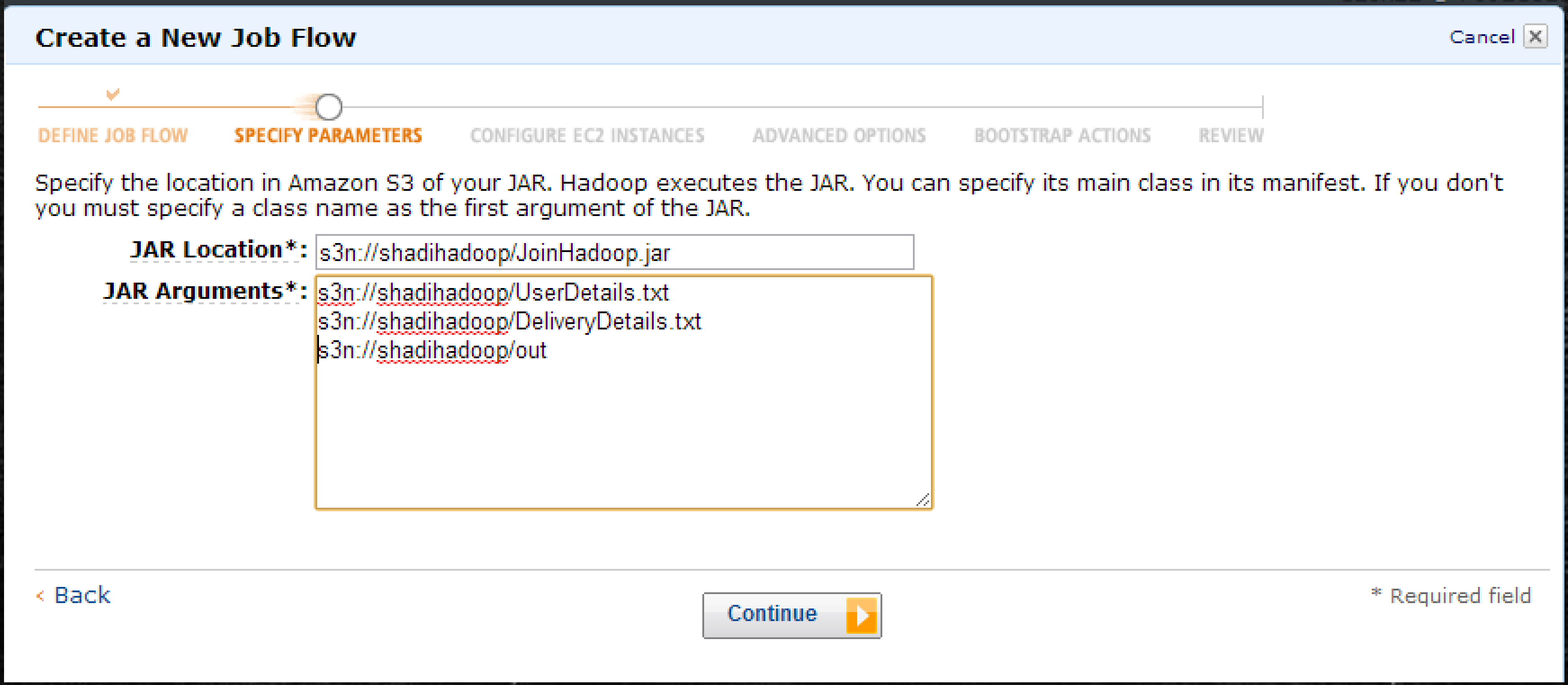
A **Custom JAR** job flow runs a Java program that you have uploaded to Amazon S3. The program should be compiled against the version of Hadoop you selected in **Hadoop Version**.

A **Streaming** job flow runs a single Hadoop job consisting of map and reduce functions implemented in a script or binary that you have uploaded to Amazon S3. You can write the functions in any of the following supported languages: Ruby, Perl, Python, PHP, R, Bash, or C++.

* Required field

Using Amazon EMR

Make Sure that the output folder (3rd argument in our example) does NOT exists on S3 (MapReduce will create it)



Create a New Job Flow

Specify the location in Amazon S3 of your JAR. Hadoop executes the JAR. You can specify its main class in its manifest. If you don't you must specify a class name as the first argument of the JAR.

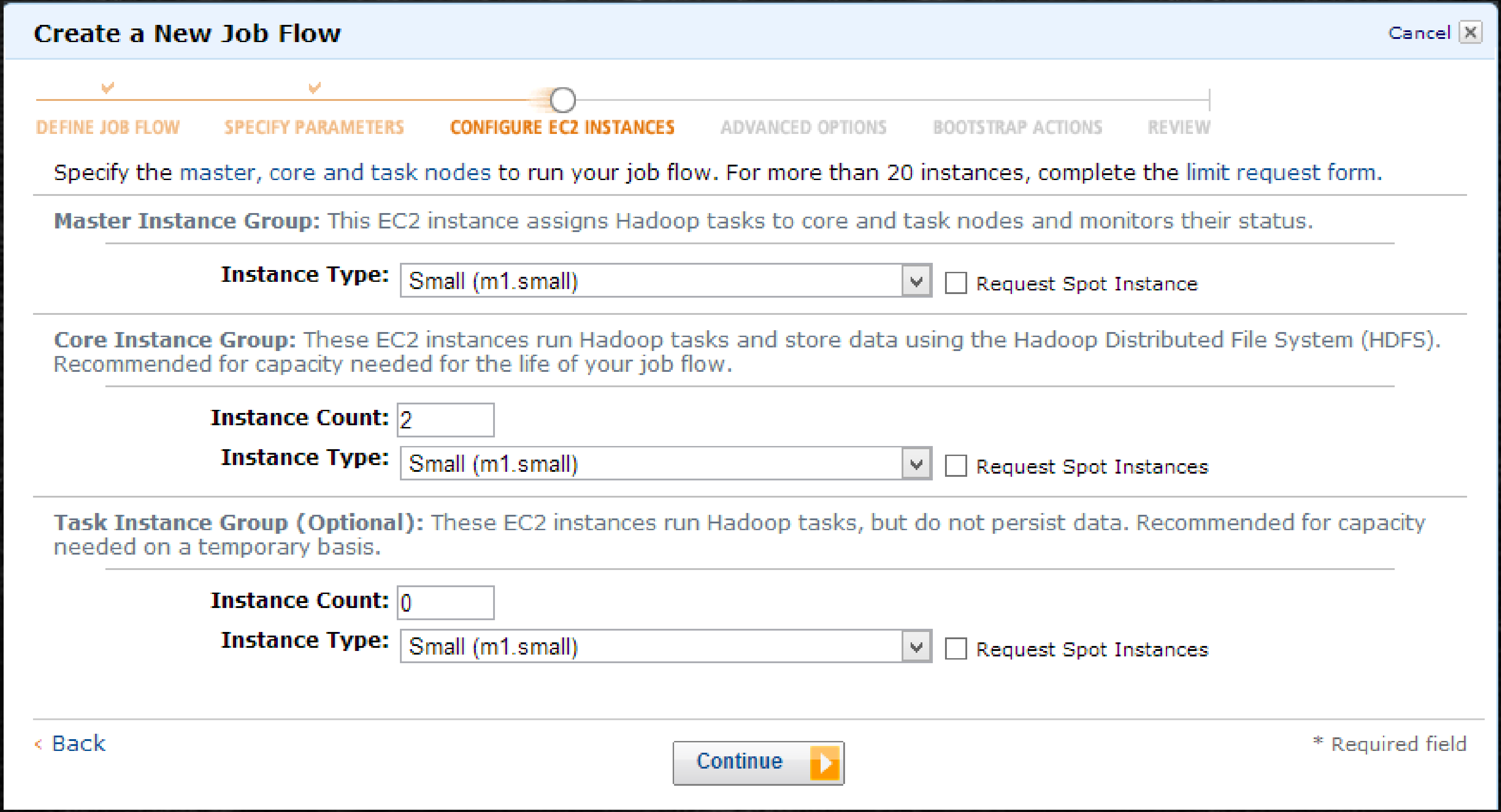
JAR Location*: s3n://shadihadoop/JoinHadoop.jar

JAR Arguments*:
```
s3n://shadihadoop/UserDetails.txt
s3n://shadihadoop/DeliveryDetails.txt
s3n://shadihadoop/out
```

* Required field

Using Amazon EMR

Set the number and type of the EC2 instances used to process your application



Create a New Job Flow

Specify the master, core and task nodes to run your job flow. For more than 20 instances, complete the limit request form.

Master Instance Group: This EC2 instance assigns Hadoop tasks to core and task nodes and monitors their status.

Instance Type: Small (m1.small) ☐ Request Spot Instance

Core Instance Group: These EC2 instances run Hadoop tasks and store data using the Hadoop Distributed File System (HDFS). Recommended for capacity needed for the life of your job flow.

Instance Count: 2

Instance Type: Small (m1.small) ☐ Request Spot Instances

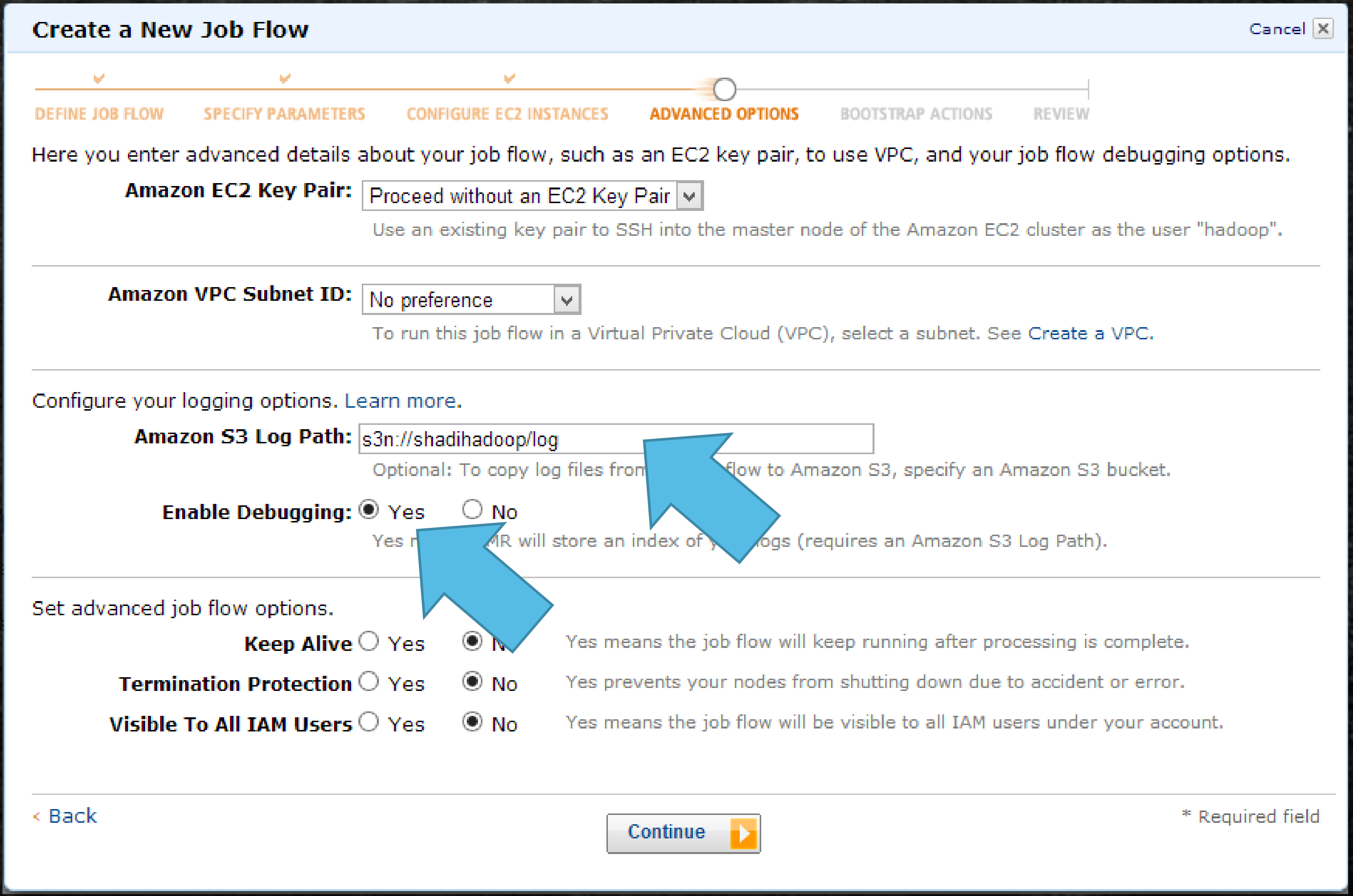
Task Instance Group (Optional): These EC2 instances run Hadoop tasks, but do not persist data. Recommended for capacity needed on a temporary basis.

Instance Count: 0

Instance Type: Small (m1.small) ☐ Request Spot Instances

Using Amazon EMR

Remember to ENABLE the Debugging and provide the log Path



Create a New Job Flow

Here you enter advanced details about your job flow, such as an EC2 key pair, to use VPC, and your job flow debugging options.

Amazon EC2 Key Pair: Proceed without an EC2 Key Pair

Use an existing key pair to SSH into the master node of the Amazon EC2 cluster as the user "hadoop".

Amazon VPC Subnet ID: No preference

To run this job flow in a Virtual Private Cloud (VPC), select a subnet. See Create a VPC.

Configure your logging options. Learn more.

Amazon S3 Log Path: s3n://shadihadoop/log

Optional: To copy log files fro flow to Amazon S3, specify an Amazon S3 bucket.

Enable Debugging: [✓ Yes] No

Yes MR will store an index of ogs (requires an Amazon S3 Log Path).

Set advanced job flow options.

Keep Alive Yes [✓ No] Yes means the job flow will keep running after processing is complete.

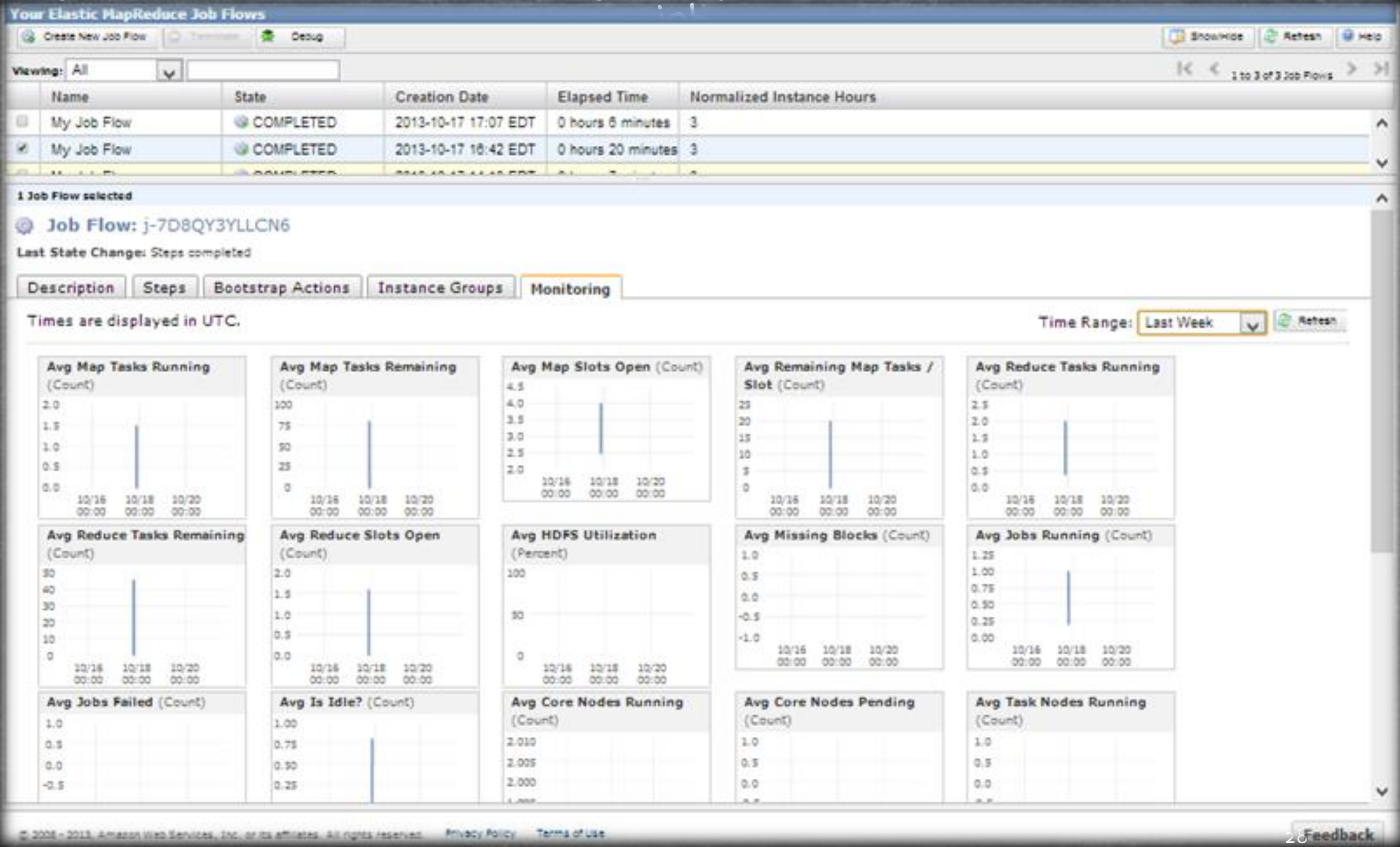
Termination Protection Yes [✓ No] Yes prevents your nodes from shutting down due to accident or error.

Visible To All IAM Users Yes [✓ No] Yes means the job flow will be visible to all IAM users under your account.

* Required field

Performance

Using Amazon cloudwatch (Monitoring Tab) you can check the performance of the



Your Elastic MapReduce Job Flows

Viewing: All

	Name	State	Creation Date	Elapsed Time	Normalized Instance Hours
	My Job Flow	COMPLETED	2013-10-17 17:07 EDT	0 hours 6 minutes	3
✓	My Job Flow	COMPLETED	2013-10-17 16:42 EDT	0 hours 20 minutes	3

1 Job Flow selected

Job Flow: j-7D8QY3YLLCN6

Last State Change: Steps completed

Times are displayed in UTC.

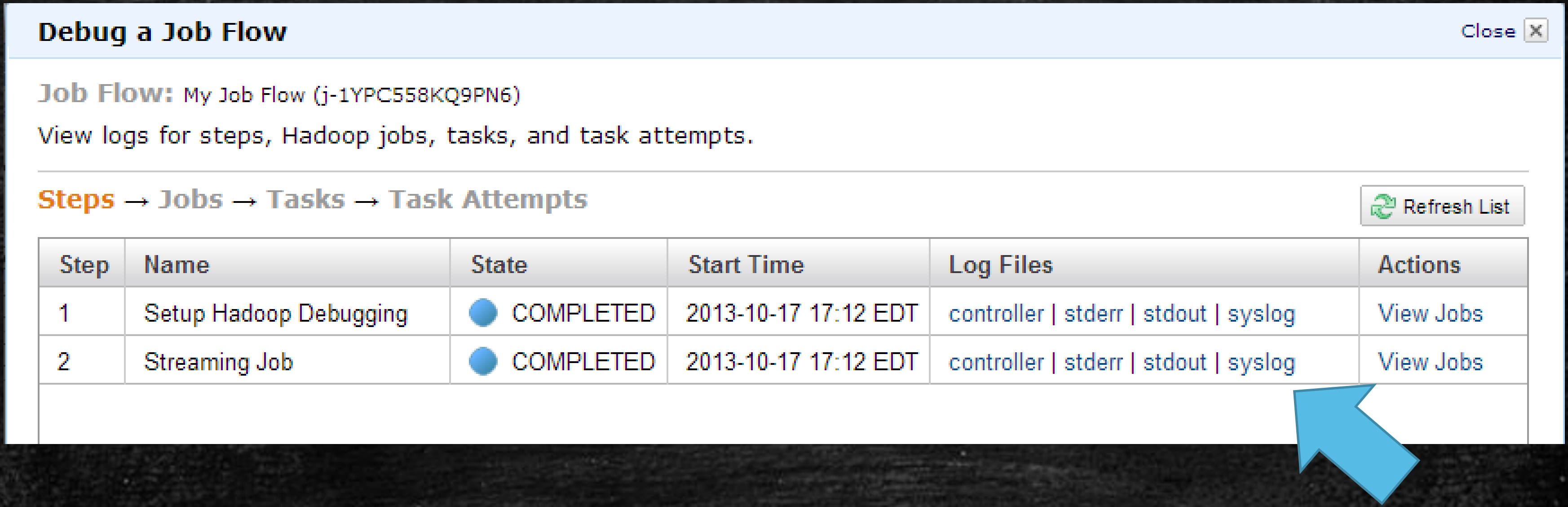
Time Range: Last Week

Create New Job Flow
Terminate
Debug
Show/Hide
Refresh
Help
1 to 3 of 3 Job Flows
Description
Steps
Bootstrap Actions
Instance Groups
Monitoring
Refresh
Avg Map Tasks Running (Count)
Avg Map Tasks Remaining (Count)
Avg Map Slots Open (Count)
Avg Remaining Map Tasks / Slot (Count)
Avg Reduce Tasks Running (Count)
Avg Reduce Tasks Remaining (Count)
Avg Reduce Slots Open (Count)
Avg HDFS Utilization (Percent)
Avg Missing Blocks (Count)
Avg Jobs Running (Count)
Avg Jobs Failed (Count)
Avg Is Idle? (Count)
Avg Core Nodes Running (Count)
Avg Core Nodes Pending (Count)
Avg Task Nodes Running (Count)
10/16 00:00
10/18 00:00
10/20 00:00
Feedback

© 2008 - 2013, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

Debugging the Job

For detailed information on the MapReduce progress, click on the syslog link



Debug a Job Flow Close

Job Flow: My Job Flow (j-1YPC558KQ9PN6)

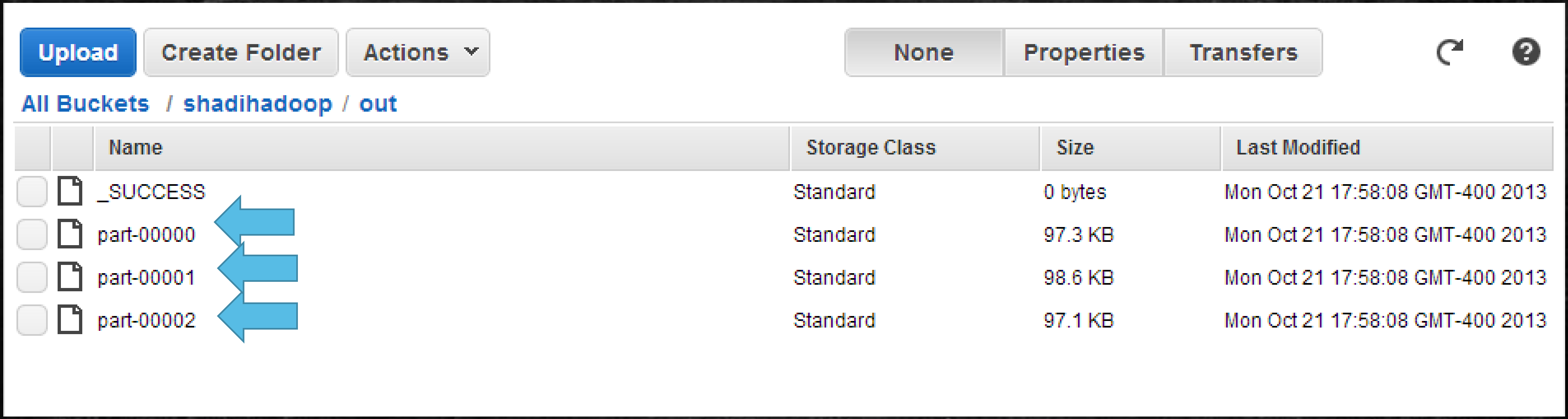
View logs for steps, Hadoop jobs, tasks, and task attempts.

Steps → Jobs → Tasks → Task Attempts

Refresh List

Step	Name	State	Start Time	Log Files	Actions
1	Setup Hadoop Debugging	● COMPLETED	2013-10-17 17:12 EDT	controller | stderr | stdout | syslog	View Jobs
2	Streaming Job	● COMPLETED	2013-10-17 17:12 EDT	controller | stderr | stdout | syslog	View Jobs

Accessing the Results on S3



All Buckets / shadihadoop / out

Name	Storage Class	Size	Last Modified
_SUCCESS	Standard	0 bytes	Mon Oct 21 17:58:08 GMT-400 2013
part-00000	Standard	97.3 KB	Mon Oct 21 17:58:08 GMT-400 2013
part-00001	Standard	98.6 KB	Mon Oct 21 17:58:08 GMT-400 2013
part-00002	Standard	97.1 KB	Mon Oct 21 17:58:08 GMT-400 2013

Elastic Map Reduce

Questions?