CLOUD COMPUTING

Amazon Elastic Compute Cloud – EC2
Amazon Elastic MapReduce – EMR

Leonidas Akritidis – lakritidis@ihu.gr

Introduction to Amazon EC2

- **EC2 stands for Elastic Compute Cloud.**
- A cloud service that provides **secure** and **resizable compute capacity** in the cloud.
  - In simple words: flexible processing power on demand.
- Designed to make web-scale cloud computing easier for developers.
- SLA commitment of 99.99% availability for each Amazon EC2 region. Each region consists of at least 3 availability zones.

EC2 Instances

- The services of EC2 are provided through the creation, deployment, and usage of **EC2 instances.**
- An instance is simply a **virtual computing environment** created with the aim of executing of a particular job.
- An instance type is a **configuration of CPU, memory, storage, and networking capacity** of an instance.
- There are 275 such instance types to help optimize the cost and performance of workloads.
- Available with choice of CPU, storage and networking options, operating system, and purchase model.

EC2 Instance Categories

- *General Purpose:* Ideal for business critical applications, small and mid-sized databases, web tier apps, etc.
- *Compute Optimized*: Ideal for high performance computing, batch processing, video encoding, and more.
- *Memory Optimized*: Ideal for high performance databases, distributed web scale in-memory caches, real time big data analytics, and more.
- *Storage Optimized*: Ideal for NoSQL databases, data warehousing, distributed file systems, and more.
- *Accelerated Computing*: Ideal for machine learning, graphic intensive applications, gaming, and more.

EC2 Instance Types and support

- More information and full details on the provided hardware per instance type https://aws.amazon.com/ec2/instance-types/
- **Note**: Some instance types are not supported by some availability zones (e.g. slide 14).
- If an instance type is not supported by the current availability zone of the user, then we should either:
  - Change the availability zone/region, or
  - Select another instance type.

EMR: Amazon Elastic MapReduce

- Amazon offers the ability to create MapReduce clusters and deploy standard MapReduce jobs on these clusters, through its EC2 infrastructure.
- This service is called **Elastic MapReduce (EMR)**.
- It resides inside the "*Analytics*" group of services, within the main menu of AWS management console.
- EMR launches EC2 instances that serve as MapReduce processing nodes and HDFS data nodes.
- Supported instance types for EMR.

EMR Work flow (Step 2)

1. Create a bucket in Amazon S3. 
2. **Create an Amazon EC2 key pair for securely accessing the master node of the EMR cluster.**
3. Create and configure an EMR cluster in EC2.
4. Study the problem and design a MapReduce algorithm.
5. Upload the input data to be processed and the executable code in a S3 bucket.
6. Deploy the job across the cluster.
7. Retrieve the output of the application in a S3 bucket.

EC2 Key Pairs (1)

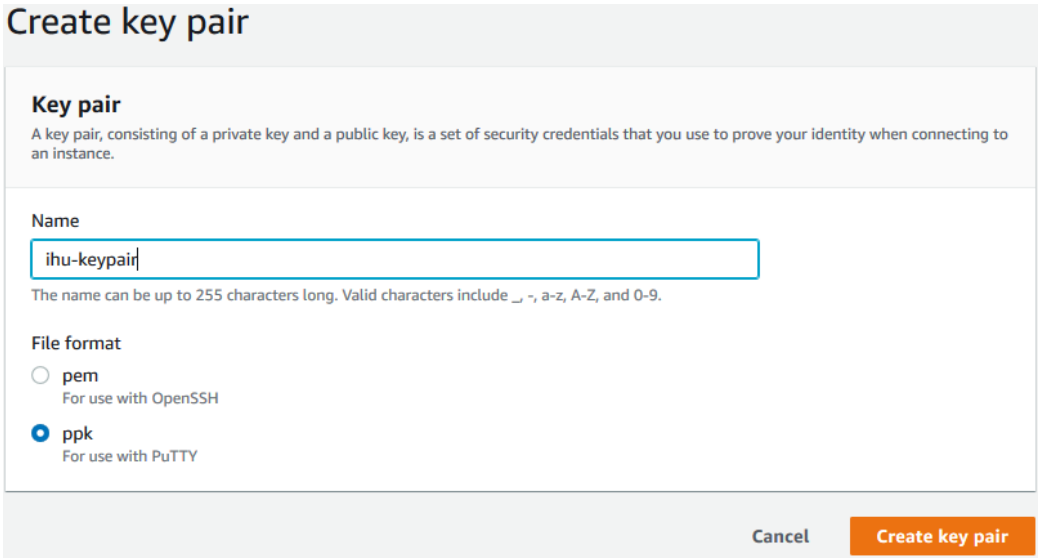
- Key pairs constitute a strategy for **securely accessing** and **managing** EC2 resources (in our case, the EMR cluster).
- The are used in replacement of the conventional "log-in" procedure through usernames and passwords.
- **Key pair = Two keys: one held by Amazon (public key) and one held by the user (private key).**
- The communication between the two parts (Amazon and the user) is performed by exchanging encrypted messages.

Public/Private keys – PEM/PPK files

- The public and private keys are used to encrypt a message upon transmission, and decrypt the cipher upon receipt.
  - A sniffer who steals an encrypted message cannot decrypt it. Its contents are inaccessible without the private key.
  - A sniffer cannot pretend that he/she is any of the two communicating parts.
- A PEM or a PPK file will be created and associated with every key pair that is created by the user.
- (One of) These files are required to contact directly the master node of an EMR cluster via SSH.

Creating an EC2 Key Pair

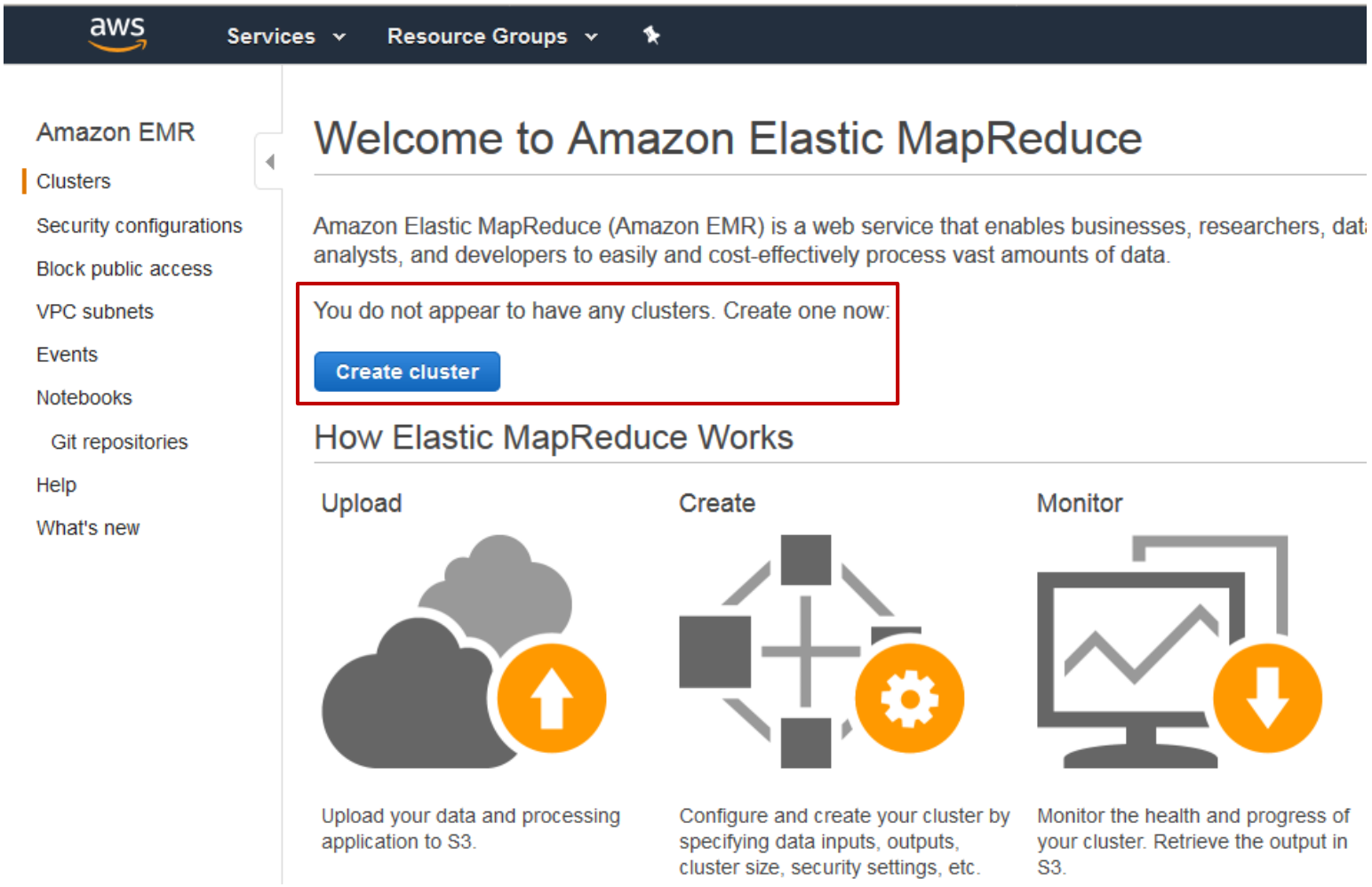
- Key pairs are created from AWS management console.
- From the left-handed pane, select "*NETWORK & SECURITY → Key Pairs*".
- Then, press the "*Create Key Pair*" button.
- Set a name for the key-pair.
- Select ppk File format.
- **Download and store the generated ppk file.**
- This file will later grant SSH access to the EMR cluster via PuTTY.



EMR Work flow (Step 3)

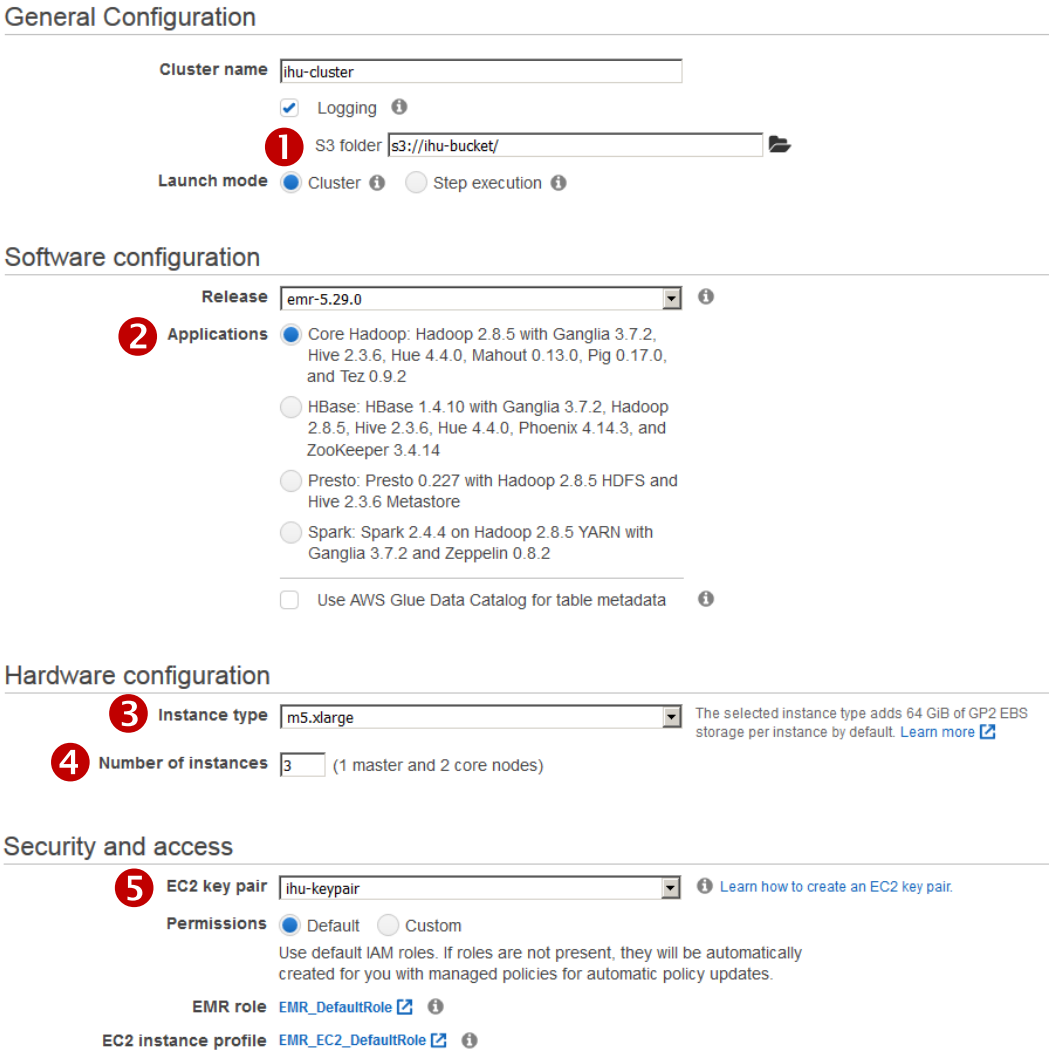
1. Create a bucket in Amazon S3. ✅
2. Create an Amazon EC2 key pair for securely accessing the master node of the EMR cluster. ✅
3. **Create and configure an EMR cluster in EC2.**
4. Study the problem and design a MapReduce algorithm.
5. Upload the input data to be processed and the executable code in a S3 bucket.
6. Deploy the job across the cluster.
7. Retrieve the output of the application in a S3 bucket.

EMR & AWS Management Console



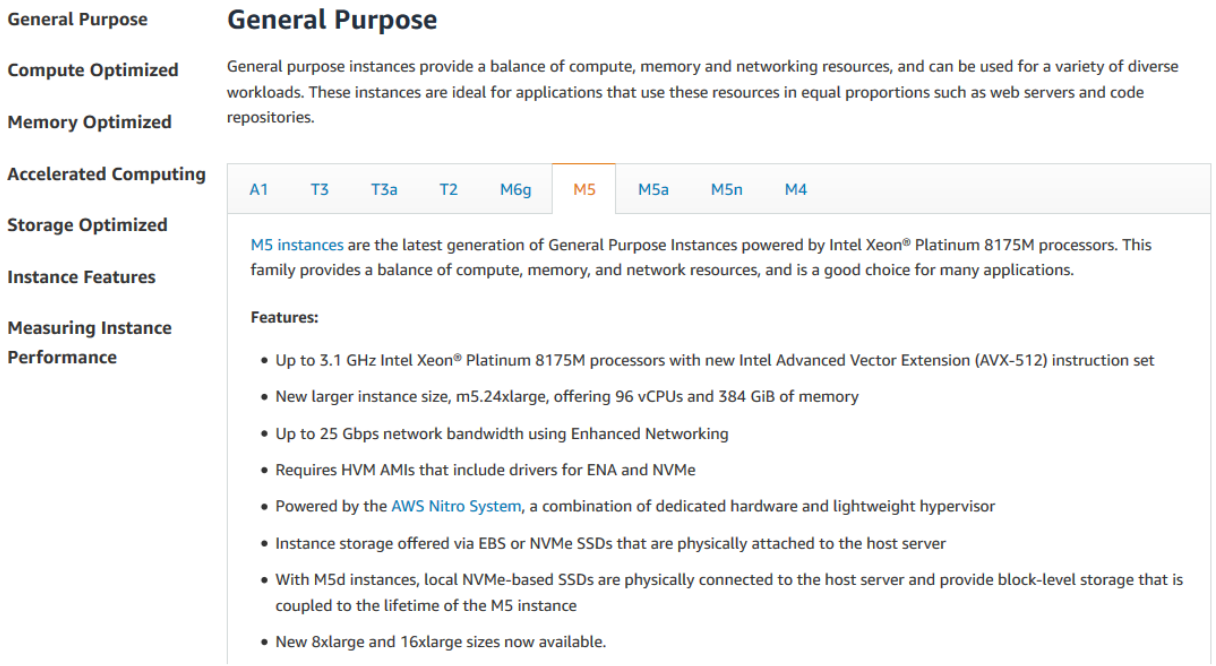
EMR: Creating a MapReduce Cluster

1. S3 bucket for MapReduce I/O.
2. Hadoop MapReduce and accompanying applications (Apache Hive data warehouse storage system, Mahout distributed linear algebra framework, etc).
3. The instance type determines the hardware specifications of the machines of the cluster.
4. Number of instances represent the number of machines in EMR Cluster (must be > 2, since 1 is reserved for the Master).
5. EC2 key pair for accessing the EMR cluster via SSH.



EMR Cluster: Instance types

- Default instance type: m5.xlarge.
- A general purpose instance "*provides a balance of compute, memory and networking resources, and can be used for a variety of diverse workloads.*"



EMR Cluster: Instance type hardware

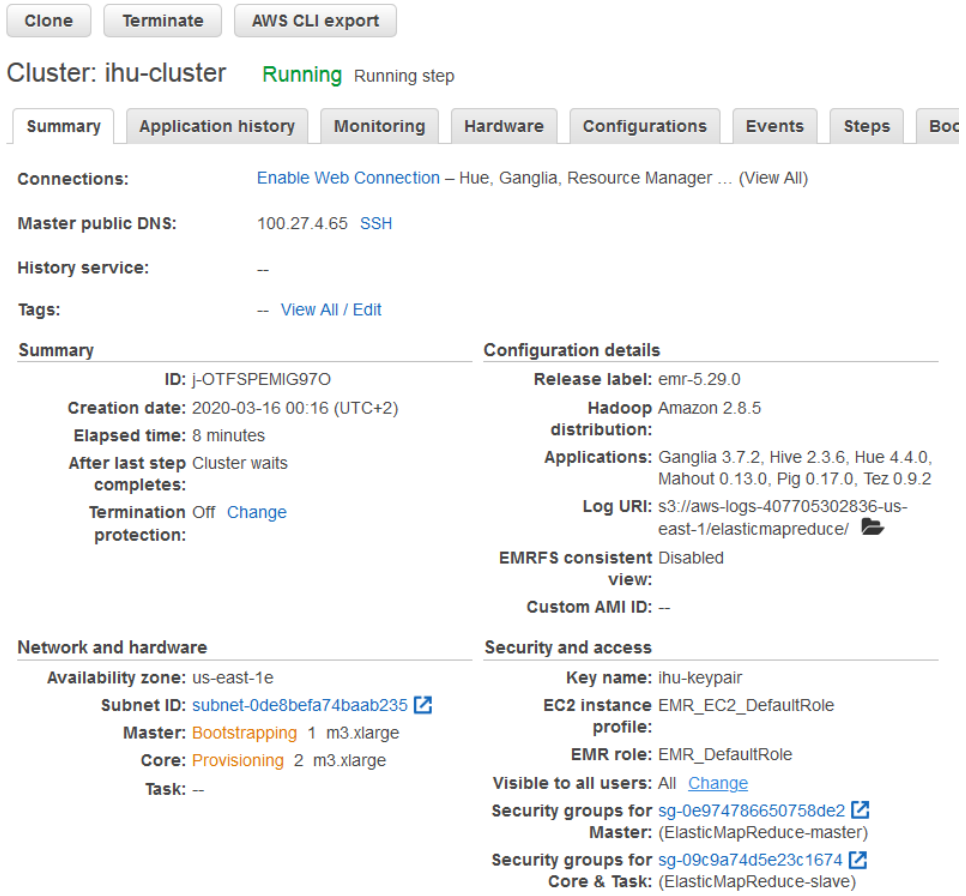
- Details for the hardware specifications of the m5.xlarge instances can be found under the *M5* tab.

Instance Size	vCPU	Memory (GiB)	Instance Storage (GiB)	Network Bandwidth (Gbps)	EBS Bandwidth (Mbps)
m5.large	2	8	EBS-Only	Up to 10	Up to 4,750
m5.xlarge	4	16	EBS-Only	Up to 10	Up to 4,750
m5.2xlarge	8	32	EBS-Only	Up to 10	Up to 4,750
m5.4xlarge	16	64	EBS-Only	Up to 10	4,750
m5.8xlarge	32	128	EBS Only	10	6,800
m5.12xlarge	48	192	EBS-Only	10	9,500
m5.16xlarge	64	256	EBS Only	20	13,600
m5.24xlarge	96	384	EBS-Only	25	19,000
m5.metal	96*	384	EBS-Only	25	19,000
m5d.large	2	8	1 x 75 NVMe SSD	Up to 10	Up to 4,750
m5d.xlarge	4	16	1 x 150 NVMe SSD	Up to 10	Up to 4,750
m5d.2xlarge	8	32	1 x 300 NVMe SSD	Up to 10	Up to 4,750

- **Note:** m5.xlarge may not be supported by the current availability zone (in this case, see again slide 5).

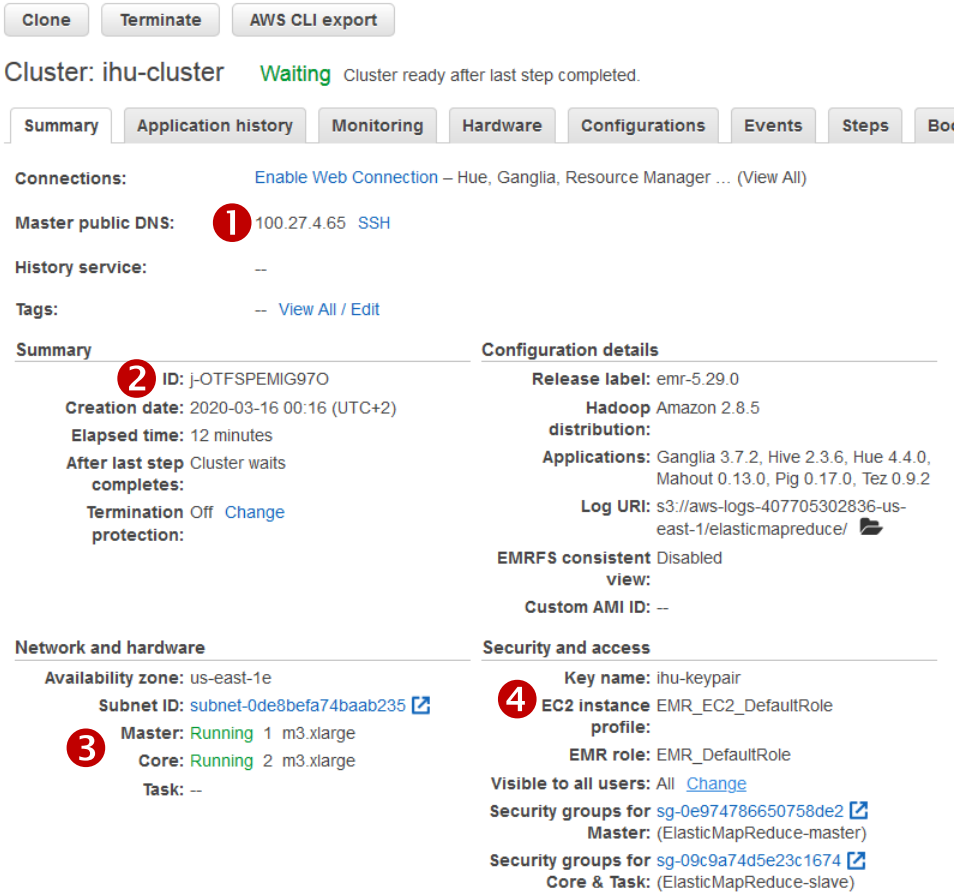
Managing the EMR Cluster

- With these simple actions, AWS will create and start an EMR cluster with 3 nodes (that is, 3 EC2 instances).
- The cluster can be **cloned** or **terminated** at any time from the buttons at the top.
- The management and monitoring of the cluster are performed from the tab headers at the top of the screen.



Managing the EMR Cluster (2)

1. IP address to connect with SSH.
   Note: SSH Connection to the Master node of the cluster **is not** allowed by default. Several configuration options must be set before SSH connection is possible. See next.
2. EMR Cluster ID, creation date and uptime.
3. Master and Worker nodes general status. Subnet ID is crucial to access the VPC which hosts the cluster.
4. Security settings: active key pair, security groups and permissions for the Master and Worker nodes of the cluster.



Connecting to EMR Cluster via SSH (1)

- Slides 18 – 25 present the necessary configuration settings for accessing our EMR cluster via SSH.
- **SSH (Secure Shell)** is a protocol that establishes secure network connections over an encrypted channel (in contrast to Telnet where the connections are unencrypted).
- It is a command line tool that allows secure remote access to a network resource, by adopting the well-known client-server architecture.
- We shall utilize PuTTY, a popular, open-source, SSH/Telnet client.

Connecting to EMR Cluster via SSH (2)

- **For each one of the nodes of the EMR cluster, AWS automatically creates and launches an EC2 instance.**
- These instances in turn, reside into **Virtual Private Clouds (VPCs).**
- To allow access to the Master of the EMR cluster, both the VPC and the security group of the respective EC2 instance, must be granted the appropriate permissions.
- The checklist in the following slide contains the necessary steps to achieve this goal.

Configurations for SSH Connection

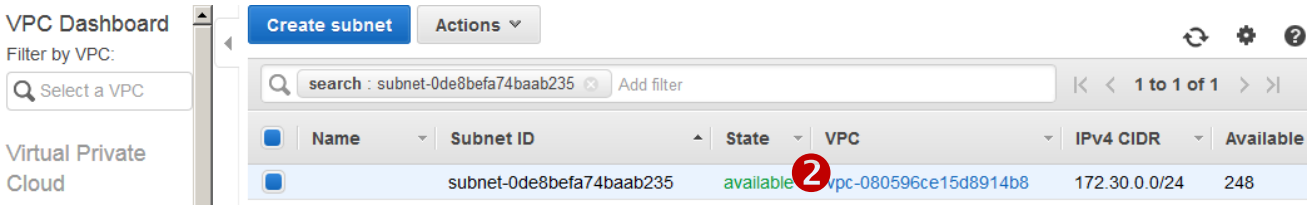
To establish a SSH connection with the EMR cluster, the steps of the following checklist must be completed.

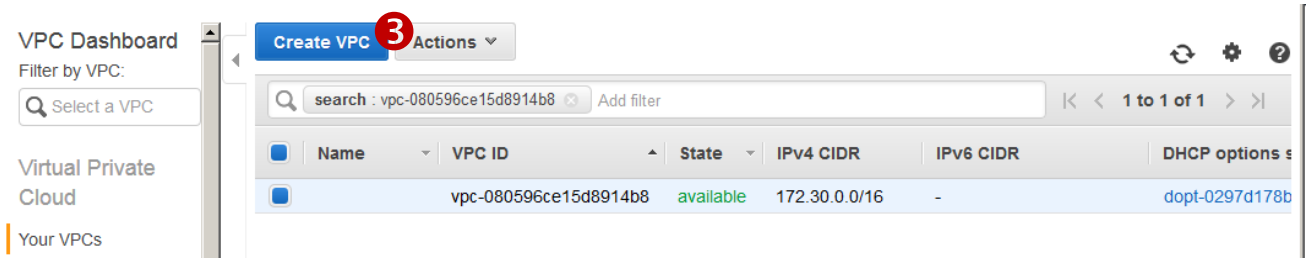
1. Configure the VPC (Virtual Private Cloud) which hosts the EMR cluster.
2. Locate the EC2 instance which hosts EMR Master.
3. Grant the appropriate permissions for SSH access to the "*ElasticMapReduce-master*" security group.
4. Configuring PuTTY for establishing a SSH connection with the EMR Cluster.

Configuring the VPC

Step 1: configure the VPC (Virtual Private Cloud) which hosts the EMR cluster.

1. From the previous slide, on ❸, click on the Subnet ID.
2. Click on the VPC of the Subnet.
3. From the Actions button, make sure that "*DNS Resolution*", "*DNS Hostnames*", and "*ClassicLink DNS Support*" are all enabled.

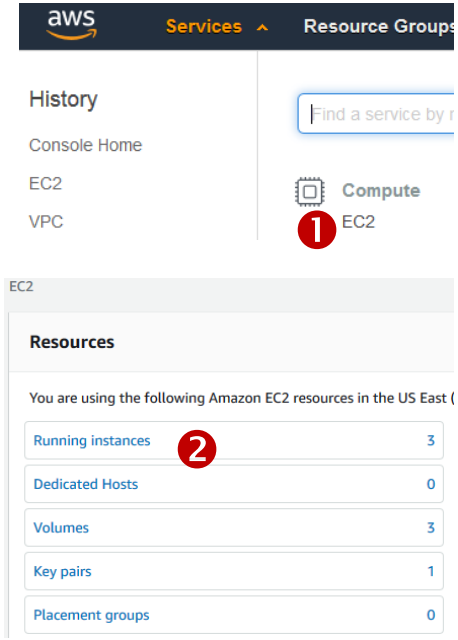


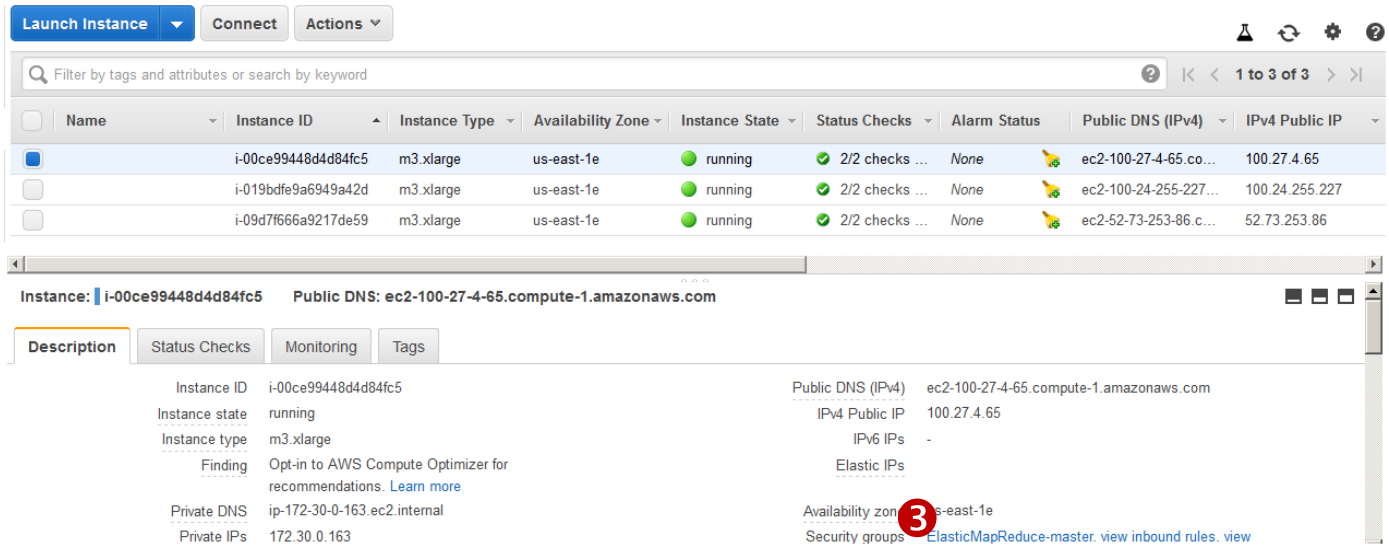


Configuring the EC2 instance

Step 2: locate the EC2 instance which hosts EMR Master.

1. From AWS Management Console, select "*EC2*".
2. In the EC2 Dashboard, click on the "Running Instances: 3" link.
3. On the list with the three EC2, instances, locate the one which has the security group "*ElasticMapReduce-master*". Click on the respective link.

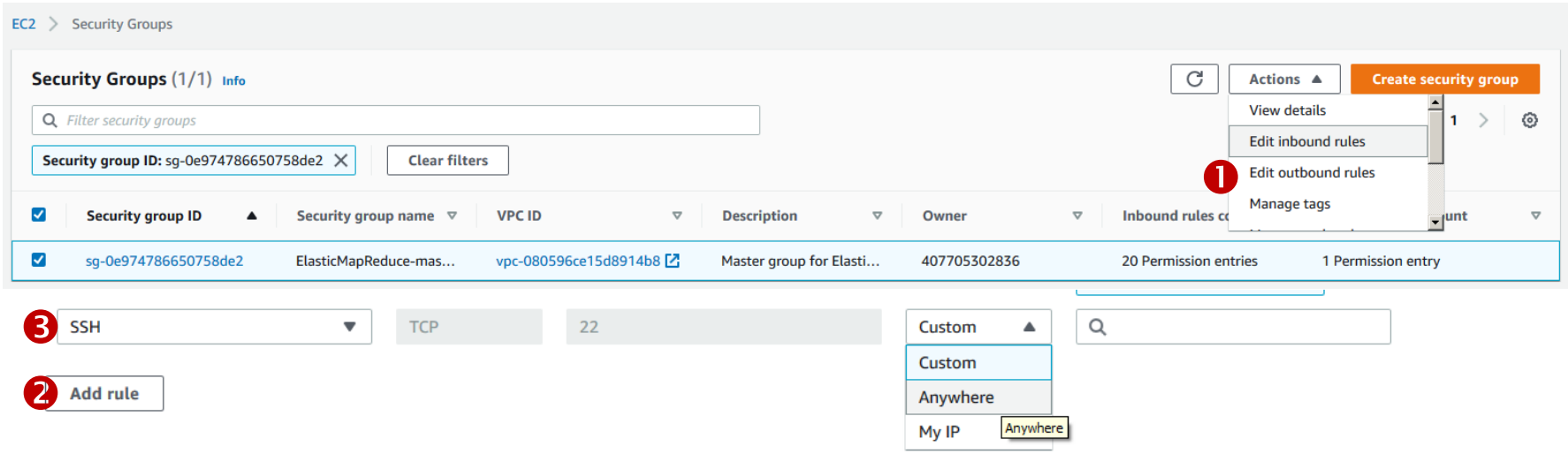




Configuring the Security Group

Step 3: grant the permissions for SSH access to the "*ElasticMapReduce-master*" security group.

1. From the "*Actions*" button, select "*Edit inbound rules*".
2. In the list of rules that opens, click on the "*Add rule*" button at the bottom of the list.
3. Add a rule for SSH access to anybody, as shown on the figure.
4. Click on the "*Save rules*" button.

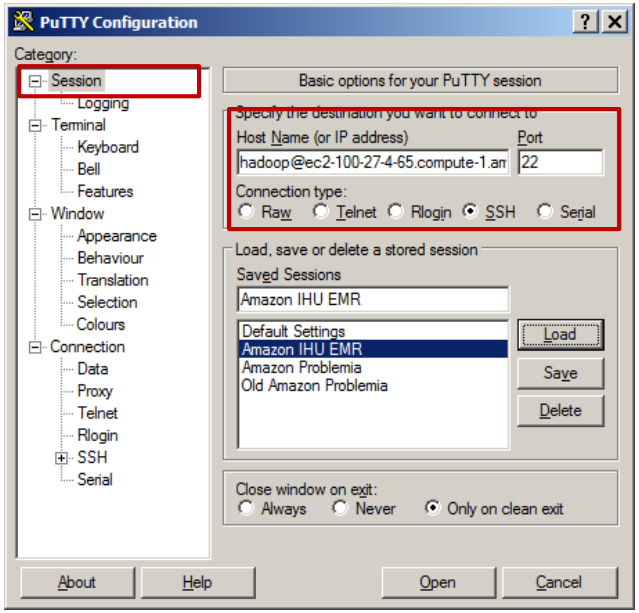


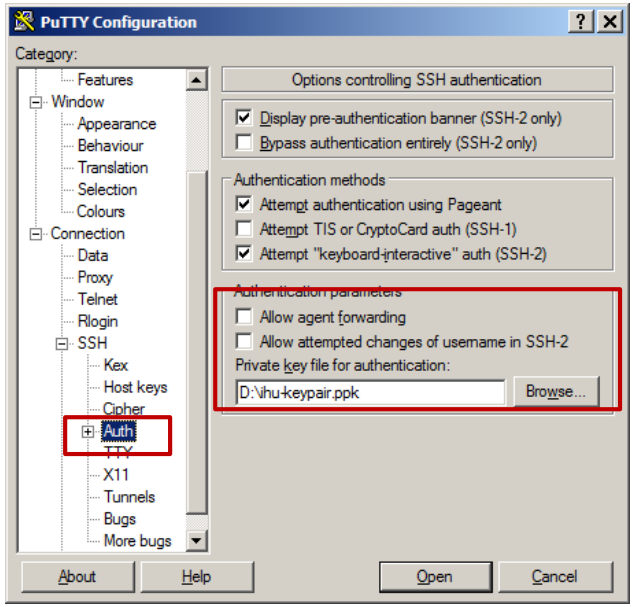
Connecting to the EMR Cluster via SSH with PuTTY

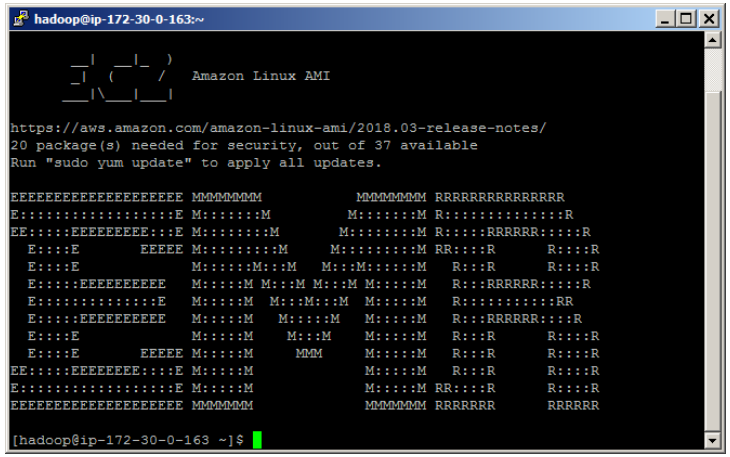
Step 4: Now we are ready to use the popular, freeware, third party software, named PuTTY, to establish a SSH connection to the cluster.

1. Download PuTTY.exe to your computer from: http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html
2. Start PuTTY.
3. In the Category list, click Session.
4. In the Host Name field, type the correct hostname (slide 16, field 1).
5. In the Category list, expand "*Connection → SSH*", and then click "*Auth*".
6. For Private key file for authentication, click Browse and select the private key file (**ihu-keypair.ppk**) used to launch the cluster.
7. Click Open.
8. Click Yes to dismiss the security alert.

Connecting to the EMR Cluster via SSH with PuTTY







EMR Work flow (Step 4)

1. Create a bucket in Amazon S3. ✅
2. Create an Amazon EC2 key pair for securely accessing the master node of the EMR cluster. ✅
3. Create and configure an EMR cluster in EC2. ✅
4. **Study the problem and design a MapReduce algorithm.**
5. Upload the input data to be processed and the executable code in a S3 bucket.
6. Deploy the job across the cluster.
7. Retrieve the output of the application in a S3 bucket.

Preparing the MapReduce job

- At this point the problem to be solved and the algorithm to be executed become the most important elements.
- We will compute Scientometrics in parallel by using MapReduce.
- Relevant article: L. Akritidis, P. Bozanis, "Computing Scientometrics in Large-Scale Academic Search Engines with MapReduce", In Proceedings of the 13th International Conference on Web Information System Engineering (WISE), Lecture Notes in Computer Science (LLNCS), vol. 7651, pp. 609-623, 2012.

Scientometrics

- Metrics evaluating the research work of a scientist by assigning impact scores to his/her articles.
- Usually expressed as definitions of the form:
  - A scientist a is of value V, if at least V of his articles have been assigned a score $S \geq V$.
- A researcher must author numerous qualitative and influential articles.
- Most popular metric: h-index, defined as,
  - A scientist a has h-index h, if at least h of his articles have received h citations (i.e., a score $S \geq h$).

h-index example

TITLE	CITED BY
Identifying the productive and influential bloggers in a community L Akritidis, D Katsaros, P Bozanis IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and ...	61
Identifying influential bloggers: Time does matter L Akritidis, D Katsaros, P Bozanis 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and ...	59
The *f* index: Quantifying the impact of coterminal citations on scientists' ranking D Katsaros, L Akritidis, P Bozanis Journal of the American Society for Information Science and Technology 60 (5 ...	40
Effective rank aggregation for metasearching L Akritidis, D Katsaros, P Bozanis Journal of Systems and Software 84 (1), 130-143	36
Effective ranking fusion methods for personalized metasearch engines L Akritidis, D Katsaros, P Bozanis 2008 Panhellenic Conference on Informatics, 39-43	13
A supervised machine learning classification algorithm for research articles L Akritidis, P Bozanis Proceedings of the 28th Annual ACM Symposium on Applied Computing, 115-120	11
Improved retrieval effectiveness by efficient combination of term proximity and zone scoring: A simulation-based evaluation L Akritidis, D Katsaros, P Bozanis Simulation Modelling Practice and Theory 22, 74-91	11
Improving opinionated blog retrieval effectiveness with quality measures and temporal features L Akritidis, P Bozanis World Wide Web 17 (4), 777-798	10
Identifying attractive research fields for new scientists L Akritidis, D Katsaros, P Bozanis Scientometrics 91 (3), 869-894	10
Computing scientometrics in large-scale academic search engines with mapreduce L Akritidis, P Bozanis International Conference on Web Information Systems Engineering, 609-623	6
Positional data organization and compression in web inverted indexes L Akritidis, P Bozanis International Conference on Database and Expert Systems Applications, 422-429	6
QuadSearch: A novel metasearch engine L Akritidis, G Voutsakelis, D Katsaros, P Bozanis	5

Article #	Citation Count	h-index
1	$61 \geq 1$	1
2	$59 \geq 2$	2
3	$40 \geq 3$	3
4	$36 \geq 4$	4
5	$13 \geq 5$	5
6	$11 \geq 6$	6
7	$11 \geq 7$	7
8	$10 \geq 8$	8
9	$10 \geq 9$	9
10	**$6 \leq 10$**	**9**

h-index in large-scale data

- To compute h-index, it is required that for each scientist we maintain a list of all of his/her articles sorted in decreasing order of citations.
- In large-scale data, there are numerous authors and numerous such lists.
  - CiteSeerX dataset: 10 million articles.
  - Microsoft Academic Graph: 209M articles, 253M authors.
- The required data does not fit into the main memory of a single workstation.
- h-index calculation must be performed in parallel.

Parallelizing the problem

- Goal: Compute Scientometrics in parallel
- Input: $(p_i, C_{pi}) \rightarrow (paperID, paperContent)$
- Output: $(a, h_a) \rightarrow (author, hindex)$
- To reach our goal, we have to construct for each author, a list of his/her articles sorted by decreasing number of citations:

$$\left(a, SortedList\left[\left(p_1, S_x^{p_1}\right), \left(p_2, S_x^{p_2}\right), \ldots, \left(p_N, S_x^{p_N}\right)\right]\right)$$

- Then, we just iterate through the list and we compute the desired h-index value.

More resources

- Download a draft preprint of the article.
- A presentation of the article in WISE 2012.
- A public GitHub repository with:
  - The source code of the MapReduce algorithms in Java, and
  - a toy dataset with 100 articles from CiteSeerX.
- The full CiteSeerX dataset.

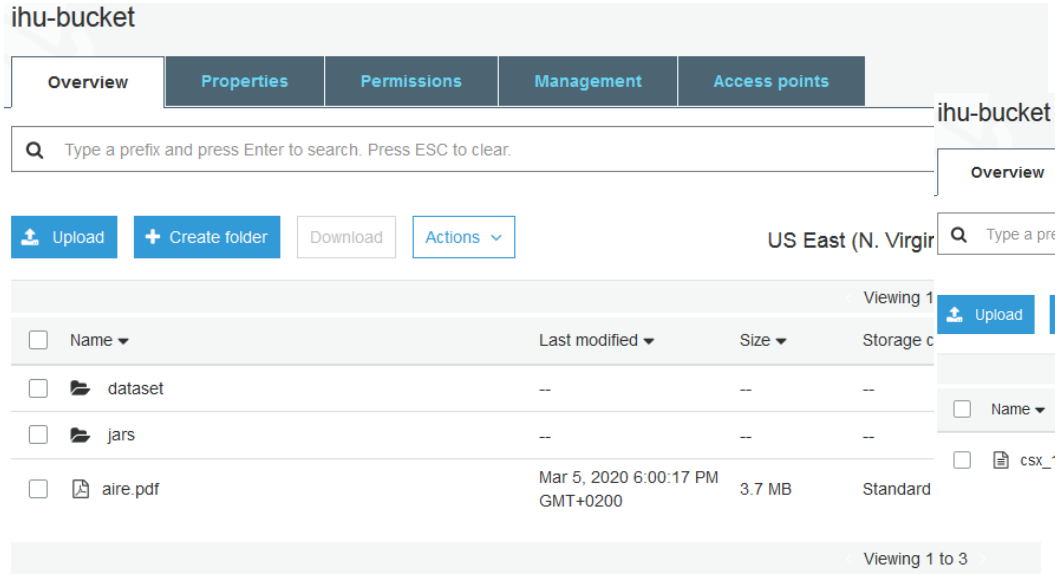
EMR Work flow (Step 5)

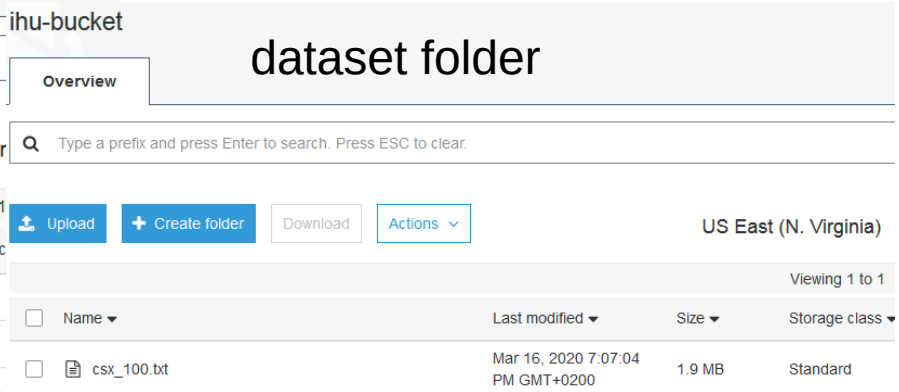
1. Create a bucket in Amazon S3. ✅
2. Create an Amazon EC2 key pair for securely accessing the master node of the EMR cluster. ✅
3. Create and configure an EMR cluster in EC2. ✅
4. Study the problem and design a MapReduce algorithm. ✅
5. **Upload the input data to be processed and the executable code in a S3 bucket.**
6. Deploy the job across the cluster.
7. Retrieve the output of the application in a S3 bucket.

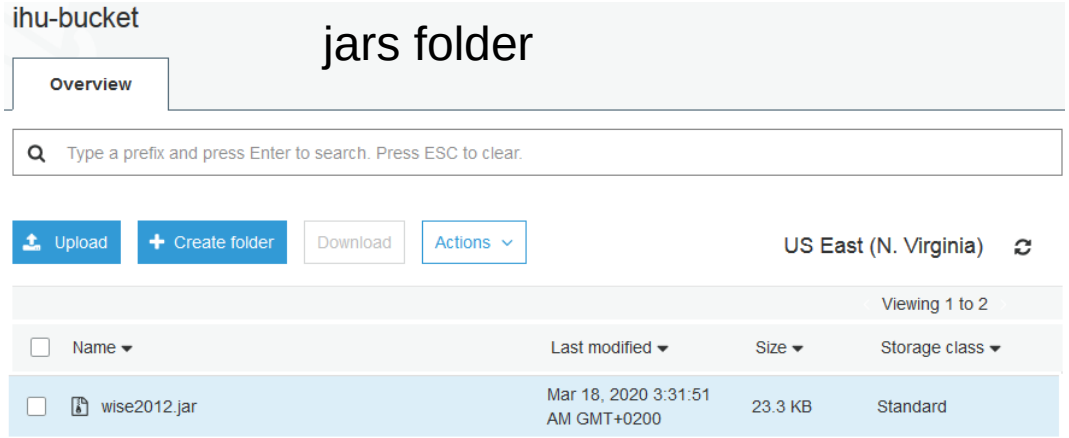
Code & dataset

- Download the code from the aforementioned GitHub repository.
- Build, compile and generate an executable (binary) JAR file from the downloaded code.
  - Some MapReduce, HDFS, YARN and other JARS maybe required on the build path of JAR.
  - Alternatively, advanced users can compile the code and generate an executable binary JAR through SSH connection with the Master machine of their EMR cluster.
- Download the toy dataset again from the aforementioned GitHub repository.

S3 ihu-bucket snapshots







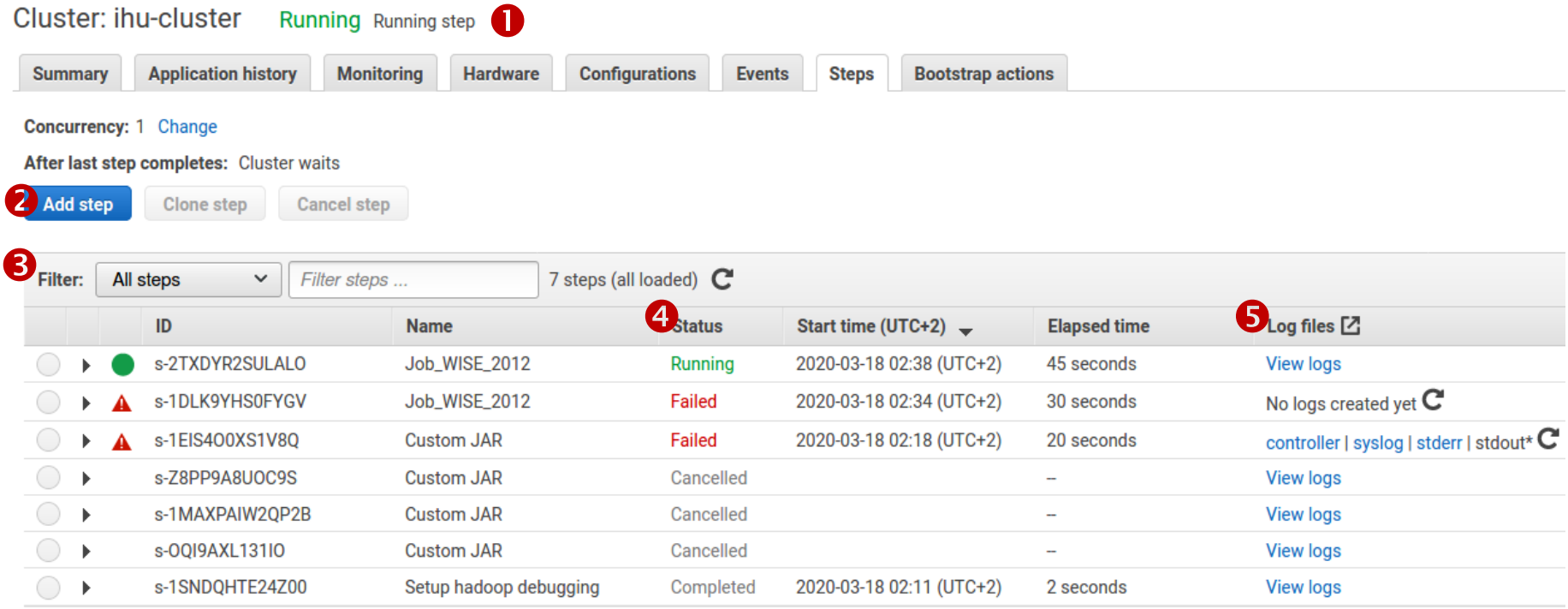
EMR Work flow (Step 6)

1. Create a bucket in Amazon S3. ✅
2. Create an Amazon EC2 key pair for securely accessing the master node of the EMR cluster. ✅
3. Create and configure an EMR cluster in EC2. ✅
4. Study the problem and design a MapReduce algorithm. ✅
5. Upload the input data to be processed and the executable code in a S3 bucket. ✅
6. **Deploy the job across the cluster.**
7. Retrieve the output of the application in a S3 bucket.

EMR Steps

- Open the management console and navigate to the created EMR cluster.
- The "*Steps*" tab accommodates the interface for creating, cancelling and monitoring the MapReduce jobs that have been scheduled on this cluster.
- AWS includes several types of such steps.
  - Running a standard & simplistic Word Count example job.
  - Running a custom JAR file,
  - etc.

EMR Steps management (1)



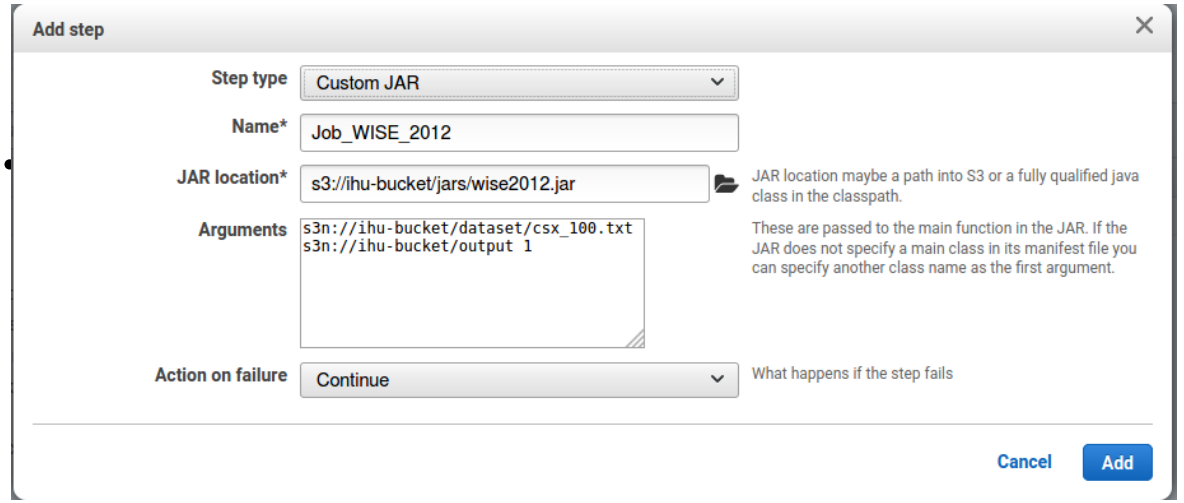
EMR Steps management (2)

1. Cluster Status: "*Waiting*" (cluster is ready to accept new jobs), "*Running*" (cluster is running the scheduled step/s), "*Terminated*".
2. Add a new step button.
3. List of scheduled steps. The pending steps will be run sequentially unless the user cancels some of them.
4. Step status (*pending*, *running*, *failed*, *completed*, *success*).
5. Log files. In most cases *stderr* is the log file of interest. It enlists Java & MapReduce runtime errors.

- Click on "*Add Step*".

Creating a Step in EMR cluster (1)

- A dialog box for the creation of a new step appears.
- Step type: Custom JAR.
- Name: A custom name.
- JAR location: the location of the executable JAR file within a valid S3 bucket.
- The arguments are application-specific.
- In our case, they reflect the input & folders in S3.



EMR Steps management (2)

- Click on the "*Add*" button.
- The new job is appended in the "*Steps*" list with status "*Pending*".
- The step will be executed automatically and its status will change according to its success/failure.
- On the latter case, the log files provide valuable information (especially the *stderr* one).

EMR Work flow (Step 6)

1. Create a bucket in Amazon S3. ✅
2. Create an Amazon EC2 key pair for securely accessing the master node of the EMR cluster. ✅
3. Create and configure an EMR cluster in EC2. ✅
4. Study the problem and design a MapReduce algorithm. ✅
5. Upload the input data to be processed and the executable code in a S3 bucket. ✅
6. Deploy the job across the cluster. ✅
7. **Retrieve the output of the application in a S3 bucket.**

Collecting the job's output

- In the case of successful completion, EMR will automatically create a folder named "*output*" inside our `ihu-bucket`.
- This folder contains the files with the output of the job.
- In our example, only one such file exists, `part-r-00000`.
- For larger jobs with multiple Reducers, the output may span multiple files.

Collecting the job's output

part-r-00000

Parsed author name → A A DICK 1 ← h-index

```
A A Abrikosov   1
A A Aho 1
A A Argyros 1
A A Beilinson   2
A A Caudy   1
A A Chien   1
A A DICK    1
A A Desrochers  1
A A Dumbreck    1
A A Fel'dbaum   1
A A Ferri   2
A A Fr&amp;amp;k    1
A A Frank   2
A A Freitas 1
A A Gaertner    1
A A Holzbacher  1
A A J Marley    1
A A Kherani 1
A A Khokhar 2
A A Kurzhanskiy 1
A A Lazar   2
A A Lumsdaine   1
A A Maciejewski 2
A A McRae   1
A A Medovikov   1
A A Porter  1
A A Ranicki 1
A A Rescigno    1
```

Thank you! Any Questions?