MapReduce Program and Deployment in Cloud

Anbuselvan M
Mohamed Muzammil J
Rajailakkiyan I B
Sri Dev S
Vishnu Barath
Vivekanand M U

Outline

1. Introduction to MapReduce
2. Writing a MapReduce Program
3. Hadoop Ecosystem & Running MapReduce Jobs
4. MapReduce Cloud Deployment
5. Deploying MapReduce on Cloud
6. Challenges, Optimization & Future Trends

Introduction to MapReduce

Sri Dev S (22z262)

What is MapReduce?

- A programming model for processing large datasets in parallel.
- Divides tasks into smaller sub-tasks and distributes them across a cluster.
- Consists of **Map**, **Shuffle**, and **Reduce** phases.
- **Working Principle:**
  - **Splits** data into smaller chunks.
  - **Processes** them in parallel.
  - **Combines** results to produce final output.

History & Significance

- **2004:** Developed by Google (Jeffrey Dean & Sanjay Ghemawat).
- Inspired by functional programming concepts (*map* and *reduce*).
- **2006 :**Apache Hadoop adopted MapReduce for large-scale data processing.
- **Key Benefits:**
  - Handles **large-scale data**.
  - **Fault-tolerant** and **scalable**.
  - Runs on **standard hardware**.

Key Concepts – Map, Shuffle, Reduce

Map Phase:

- The input data is **broken into small parts** and sent to multiple computers.
- Each part is processed separately by a **Mapper function**.
- The Mapper converts the data into **key-value pairs**
- Example (Word Count):

  Input : "Hadoop is fast. Hadoop is scalable."

  Output : ("Hadoop", 1), ("is", 1), ("fast.", 1), ("Hadoop", 1), ("is", 1),("scalable.", 1)

Key Concepts – Map, Shuffle, Reduce

Shuffle & Sort Phase:

- The system **groups** all key-value pairs that have the same key.
- It also **sorts** them so they are easy to process in the next step.

Input (Before Shuffle & Sort) :

("Hadoop", 1), ("Hadoop", 1), ("is", 1), ("is", 1), ("fast.", 1), ("scalable.", 1)

Output (After Shuffle & Sort) :

("Hadoop", [1,1]), ("is", [1,1]), ("fast.", [1]), ("scalable.", [1])

Key Concepts – Map, Shuffle, Reduce

Reduce Phase :

- The **Reducer function** adds up all the values for each key.
- It produces the **final result** with the total count for each word.

Input (Grouped Data):

("Hadoop", [1,1]), ("is", [1,1]), ("fast.", [1]), ("scalable.", [1])

Final Output (Word Count):

("Hadoop", 2), ("is", 2), ("fast.", 1), ("scalable.", 1)

Real-World Use Cases

- Search Engines (Google, Bing, Yahoo) - Web page indexing.
- Social Media (Facebook, Twitter) - Trend analysis.
- Fraud Detection (Banks) - Transaction monitoring.
- Healthcare - Medical data analysis.
- Retail (Amazon, Walmart) - Customer behavior insights.

Writing a MapReduce Program

Vishnu Barath K(22z274)

Structure of a MapReduce Job

Reduce Phase :

- The **Reducer function** adds up all the values for each key.
- It produces the **final result** with the total count for each word.

Input (Grouped Data):

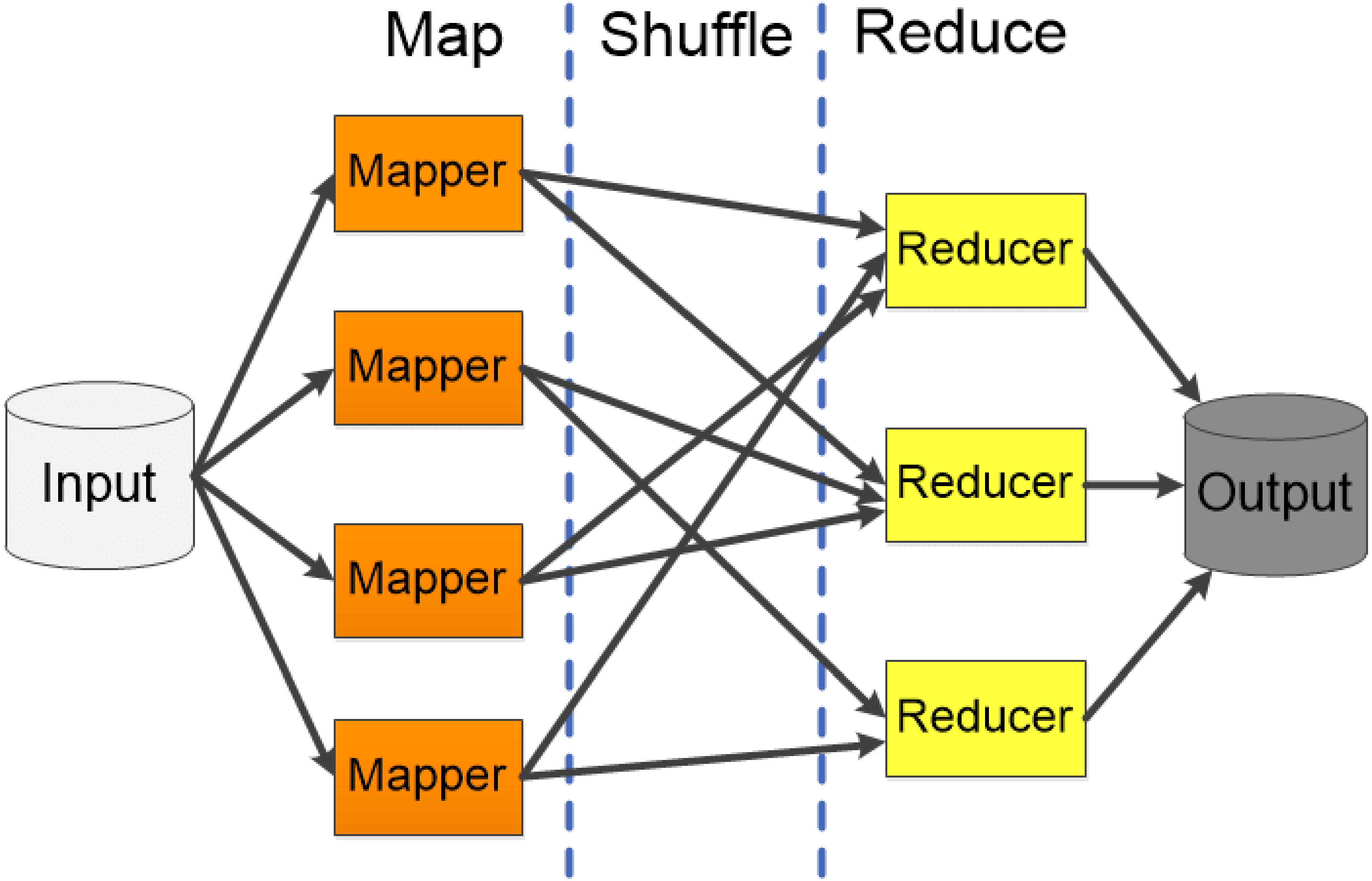
("Hadoop", [1,1]), ("is", [1,1]), ("fast.", [1]), ("scalable.", [1])

Final Output (Word Count):

("Hadoop", 2), ("is", 2), ("fast.", 1), ("scalable.", 1)

Structure of a MapReduce Job

- **Input Data**: Stored in HDFS (Hadoop Distributed File System)
- **Mapper Phase**: Processes input data and generates key-value pairs
- **Shuffle and Sort**: Groups similar keys together
- **Reducer Phase**: Aggregates and processes mapped data
- **Output Data**: Stored back in HDFS


Map
Shuffle
Reduce
Input
Mapper
Mapper
Mapper
Mapper
Reducer
Reducer
Reducer
Output

Writing the Mapper Class

The `Mapper` class processes input and emits key-value pairs

Implementation:

```java
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);
        }
    }
}
```

Writing the Reducer Class

The `Reducer` class processes the key-value pairs and aggregates results

Implementation

```java
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);
    }
}
```

Input and Output Formats

Input Format:

- **TextInputFormat (Default):** Reads text files line by line
- **KeyValueTextInputFormat:** Splits data based on a delimiter
- **SequenceFileInputFormat:** Reads binary key-value pairs

Output Format:

- **TextOutputFormat (Default):** Writes plain text
- **SequenceFileOutputFormat:** Writes binary output

Word Count Example:

```
public class WordCount {
    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);
            }

    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);
        }
```

Word Count Code

```
public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "word count");
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
      }
}
```

Running & Monitoring MapReduce Jobs Hadoop (Single Vs Multi-node cluster)

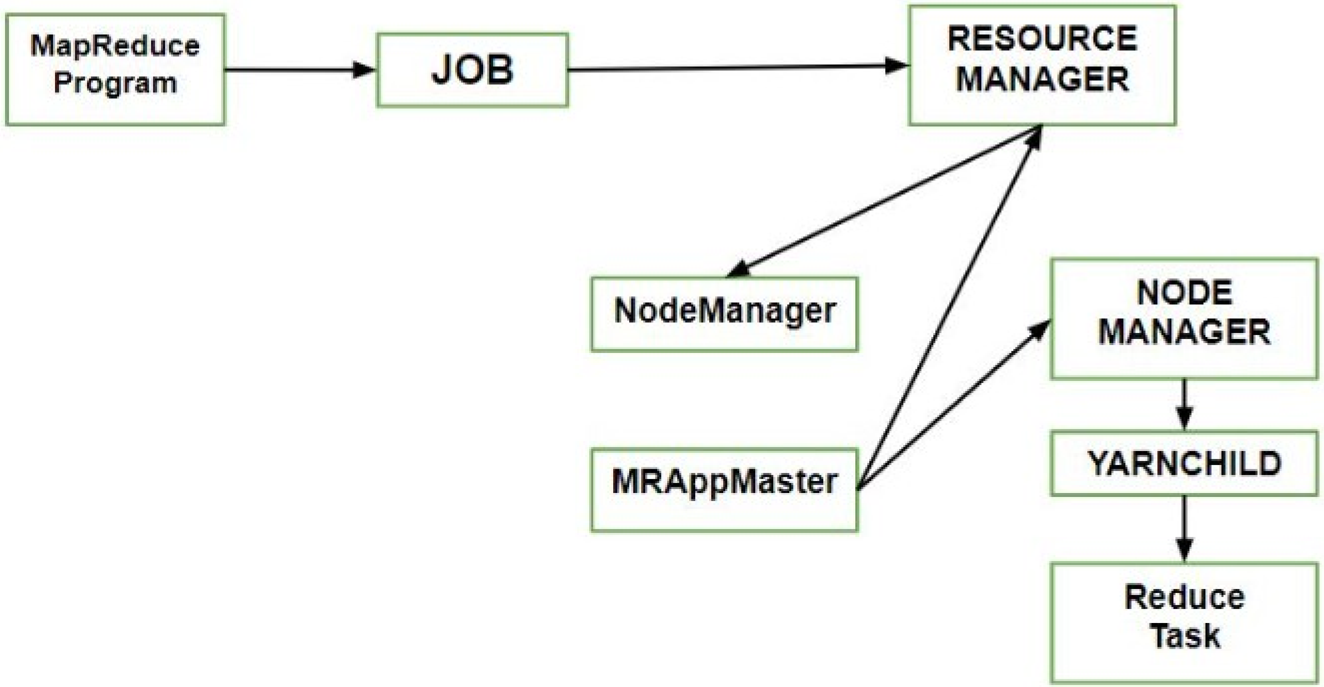
Vivekanand M U (22z275)

Overview of Hadoop and Its Components

- Hadoop is an open-source framework designed for the distributed processing of large datasets across clusters of computers using simple programming models. The **core components of a hadoop ecosystem** include:

    - HDFS NameNode
    - HDFS DataNode
    - YARN NodeManager
    - YARN ResourceManager
    - MapReduce

Running MapReduce Jobs:



Submitting and Monitoring MapReduce Jobs:

Job is the primary interface by which user-job interacts with the ResourceManager.

Job provides facilities to **submit jobs, track their progress, access component-tasks** reports and logs.

The job submission process involves:

1. Checking the **input and output specifications** of the job.
2. Computing the **InputSplit** values for the job.
3. Setting up the requisite accounting information for the **DistributedCache** of the job, if necessary.
4. **Copying the job's jar and configuration** to the MapReduce system directory on the FileSystem.
5. Submitting the job to the ResourceManager and optionally monitoring its status.

History logs: mapred job -history output.jhist

Job.submit() : Submit the job to the cluster and return immediately.
Job.waitForCompletion(boolean) : Submit the job to the cluster and wait for it to finish.

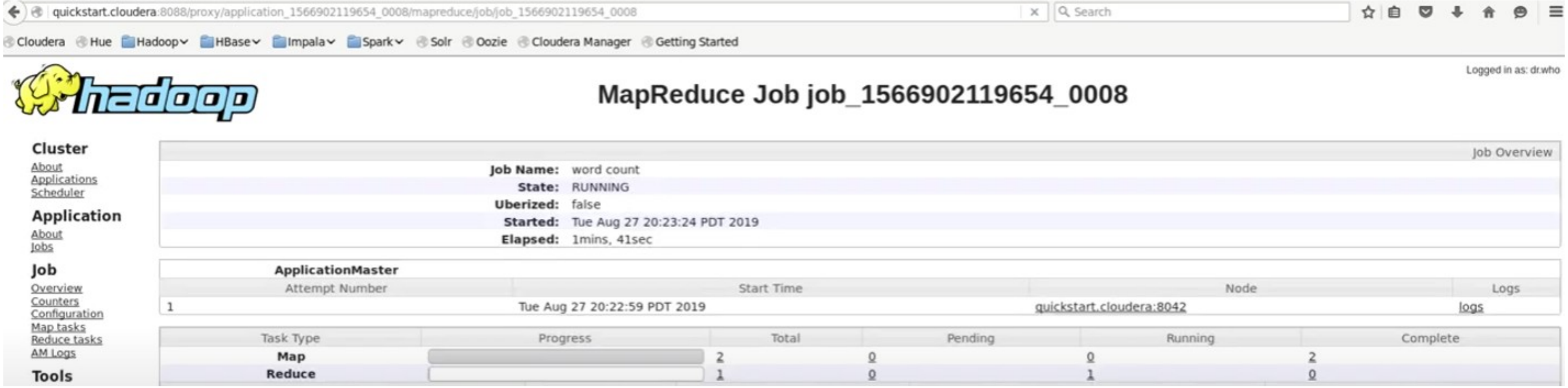
Monitor MapReduce Jobs using Cloudera:

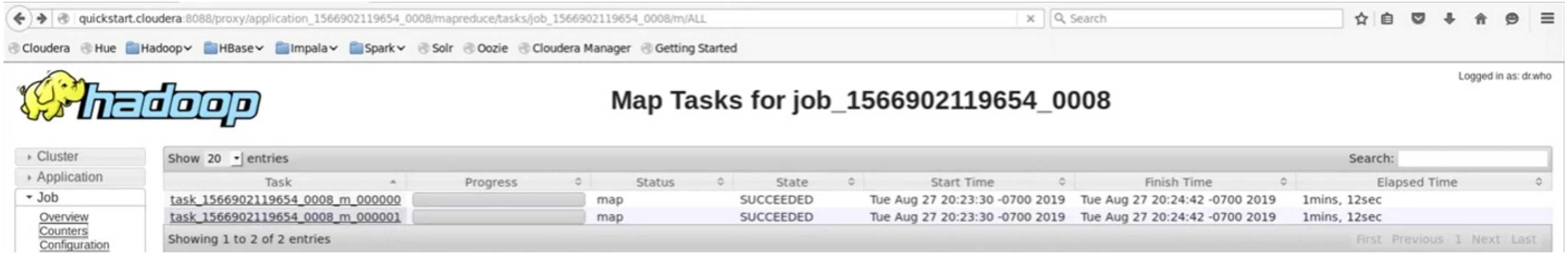
- Cloudera is a company that provides Hadoop-based big data solutions.
- It offers Cloudera Distribution for Hadoop (CDH), a Hadoop ecosystem with tools like HDFS, YARN, Hive, and Spark.
- It simplifies Hadoop cluster management and enhances enterprise security and performance.
- It can be local if installed on a single-node cluster using **Cloudera QuickStart VM**.

```
[cloudera@quickstart hadoop_codes_simple_wc]$ yarn jar wc.jar WordCount /user/cloudera/data/wc_3/input /user/cloudera/hadoop/data/wc_3/output_rec_part_3
19/08/27 20:22:45 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
```

```
19/08/27 20:22:49 INFO mapreduce.JobSubmitter: number of splits:2
19/08/27 20:22:50 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1566902119654_0008
19/08/27 20:22:51 INFO impl.YarnClientImpl: Submitted application application_1566902119654_0008
19/08/27 20:22:51 INFO mapreduce.Job: The url to track the job: http://quickstart.cloudera:8088/proxy/application_1566902119654_0008/
19/08/27 20:22:51 INFO mapreduce.Job: Running job: job_1566902119654_0008
^C[cloudera@quickstart hadoop_codes_simple_wc]$ http://quickstart.cloudera:8088/proxy/application_1566902119654_0008/
```

Monitor MapReduce Jobs using Cloudera:





Configuration (Core XML Files)

Core-site.xml: (Hadoop Configuration)

Specifies the HDFS NameNode location (fs.defaultFS property), which is the entry point for the entire Hadoop filesystem.

```
<configuration>
   <property>
      <name>fs.defaultFS</name>
      <value>hdfs://localhost:9000</value>
   </property>
</configuration>
```

Hdfs-site.xml: (HDFS Configuration)

Defines the replication factor for data blocks, controlling how many copies of your data are stored across the cluster for fault tolerance.

```
<property>
      <name>dfs.replication</name>
      <value>3</value>  <!-- Number of worker nodes –!>
</property>
```

Configuration (Core XML Files)

mapred-site.xml (MapReduce Configuration):
Configures which framework will be used for processing data (typically YARN), determining how MapReduce jobs are executed.

```
<configuration>
    <property>
        <name>mapreduce.framework.name</name>
        <value>yarn</value>
    </property>
</configuration>
```

Yarn-site.xml: (YARN Configuration)
Specifies the auxiliary services needed for applications, particularly the shuffle operation that's critical for MapReduce jobs to function.

```
<property>
        <name>yarn.resourcemanager.hostname</name>
        <value>master</value>
</property>
```

Word Count Example: (Single-node)

- The input file (input.txt) is stored locally on disk (not in HDFS).
- The whole file is processed as one logical block (since there is no HDFS splitting).
- The single node runs the entire **Mapper logic.**

(hadoop, 1)
(mapreduce, 1)
(hadoop, 1)
(spark, 1)
(hadoop, 1)
(hdfs, 1)
(spark, 1)

(hadoop, [1, 1, 1])
(mapreduce, [1])
(hdfs, [1])
(spark, [1, 1])

(hadoop, 3)
(mapreduce, 1)
(hdfs, 1)
(spark, 2)

- The final result is stored in file:///output/part-r-00000.

Word Count Example: (Multi-node)

- The input file (input.txt) is uploaded to HDFS.
- HDFS splits the file into blocks (default 128MB or 256MB).
- Blocks are replicated across 3 worker nodes for fault tolerance.
- Each block is processed independently on different nodes. Each worker node runs its own Mapper tasks.

1. **Block Splitting & Map Phase:**

Node 1 Mapper:

(hadoop, 1)
(mapreduce, 1)

Node 2 Mapper:

(hadoop, 1)
(spark, 1)
(hadoop, 1)

Node 3 Mapper:

(hdfs, 1)
(spark, 1)

2. **Partition Phase:**

partition = key.hashCode() % numReducers

"hadoop" hash → Partition 0 ,
"spark" hash → Partition 2 ,

"mapreduce" hash → Partition 1
"hdfs" hash → Partition 0

Word Count Example: (Multi-node)

3. **Shuffle and Sort:**

Reducer 0 receives:

After sorting: (hadoop, 1), (hadoop, 1), (hadoop, 1), (hdfs, 1)

Reducer 1 receives:

(mapreduce, 1)

Reducer 2 receives:

After sorting: (spark, 1), (spark, 1)

4. **Reducer Execution (Distributed):**

- Each Reducer **runs on a separate node**.
- The final result is written back to HDFS.
- The output is stored in multiple partitioned files (part-r-00000, part-r-00001, etc.).

Hadoop: Single-Node vs Multi-Node Configuration

Configuration File	Single-Node	Multi-Node	Why the Difference?
`core-site.xml`	`fs.defaultFS = hdfs://localhost:9000`	`fs.defaultFS = hdfs://master:9000`	Multi-node uses a remote **master** node instead of **localhost**, allowing distributed access to the filesystem.
`hdfs-site.xml`	`dfs.replication = 1`	`dfs.replication = 3`	Single-node doesn't need data replication, while multi-node maintains **3** copies of data blocks across different nodes for fault tolerance.
`mapred-site.xml`	`mapreduce.framework.name = local`	`mapreduce.framework.name = yarn`	Single-node processes jobs locally, while multi-node distributes computation across the cluster using YARN resource management.
`yarn-site.xml`	*Not required*	`yarn.resourcemanager.hostname = master`	YARN resource manager is essential for multi-node clusters to coordinate job scheduling and resource allocation across nodes.

MapReduce Cloud Deployment

Anbuselvan M (22Z210)

What is Cloud Deployment?

- Cloud deployment means running applications and services on **cloud infrastructure** instead of local servers.
- It provides **on-demand access** to computing resources like storage, processing power, and networking.
- Used for **scalability, cost-efficiency, and flexibility**.
- Cloud providers handle hardware, updates, and security, **reducing operational overhead**.
- Applications can be **accessed from anywhere**, ensuring collaboration and remote work.

Why deploy MapReduce in the Cloud?

MapReduce processes **large datasets**, and cloud platforms help by providing:

- **Scalability** – Handle increasing data volume dynamically.
- **Cost Efficiency** – Pay only for what you use (No upfront investment).
- **Managed Services** – Cloud providers handle infrastructure setup.
- **Faster Processing** – Distributed resources improve performance.

Running Hadoop on the Cloud

What's wrong with on-premise Hadoop?

Rigid and unscalable infrastructure

- Buy too much: waste money & have idle resources lying around
- Buy too little: slow performance

Time-consuming setup & maintenance

- Purchasing and configuring new computers for Hadoop clusters take too long
- Fixing or replacing malfunctioned hardware could impact business operations

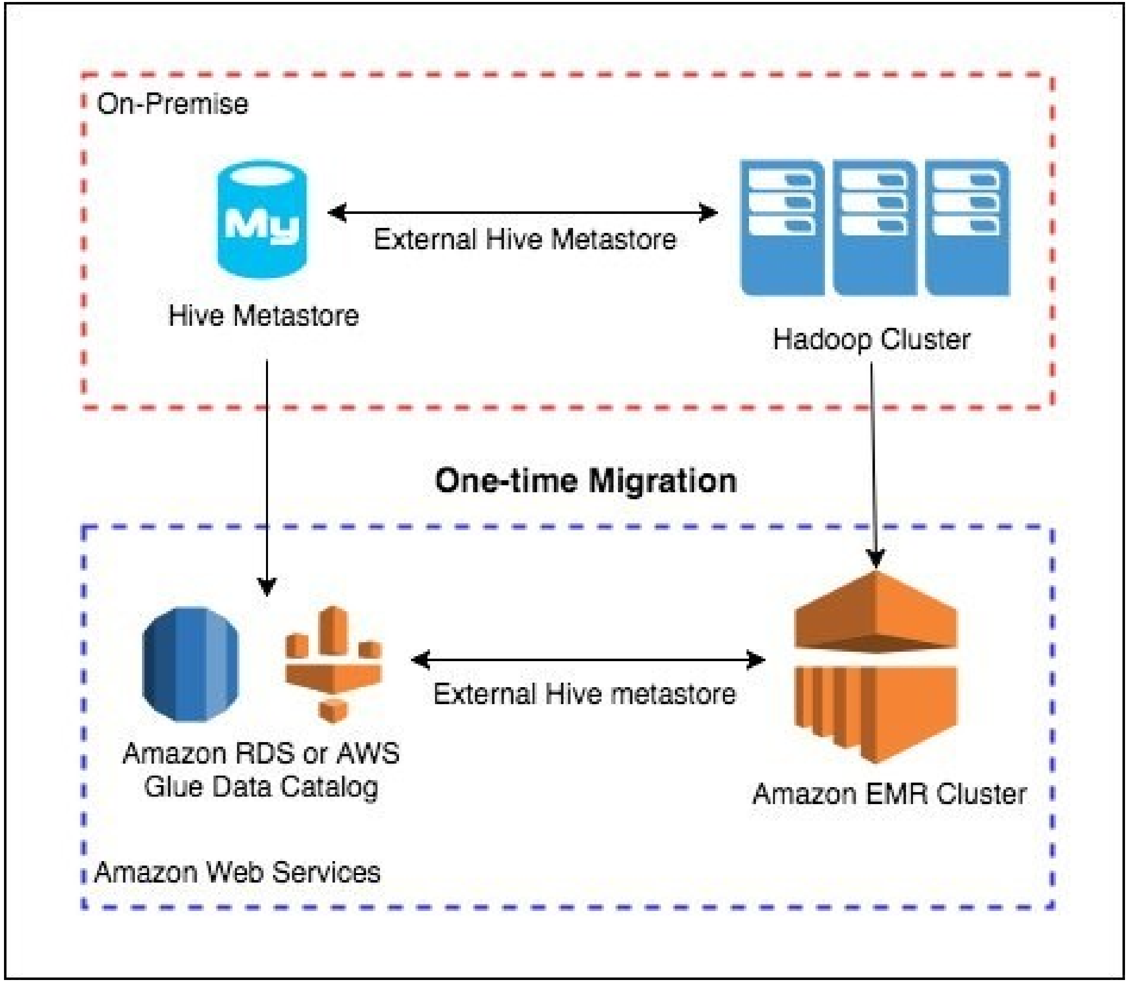
How does running Hadoop on the Cloud help?

Flexible to changes

- Quickly obtain what you need, resize & tear down what you no longer need
- Pay as you go

Managed setup & maintenance

- Launch and allocate new resources for Hadoop needs on the fly
- Hardware failure is managed by cloud providers


On-Premise
Hive Metastore
External Hive Metastore
Hadoop Cluster
One-time Migration
Amazon RDS or AWS
Glue Data Catalog
External Hive metastore
Amazon EMR Cluster
Amazon Web Services

Cloud Platforms Supporting

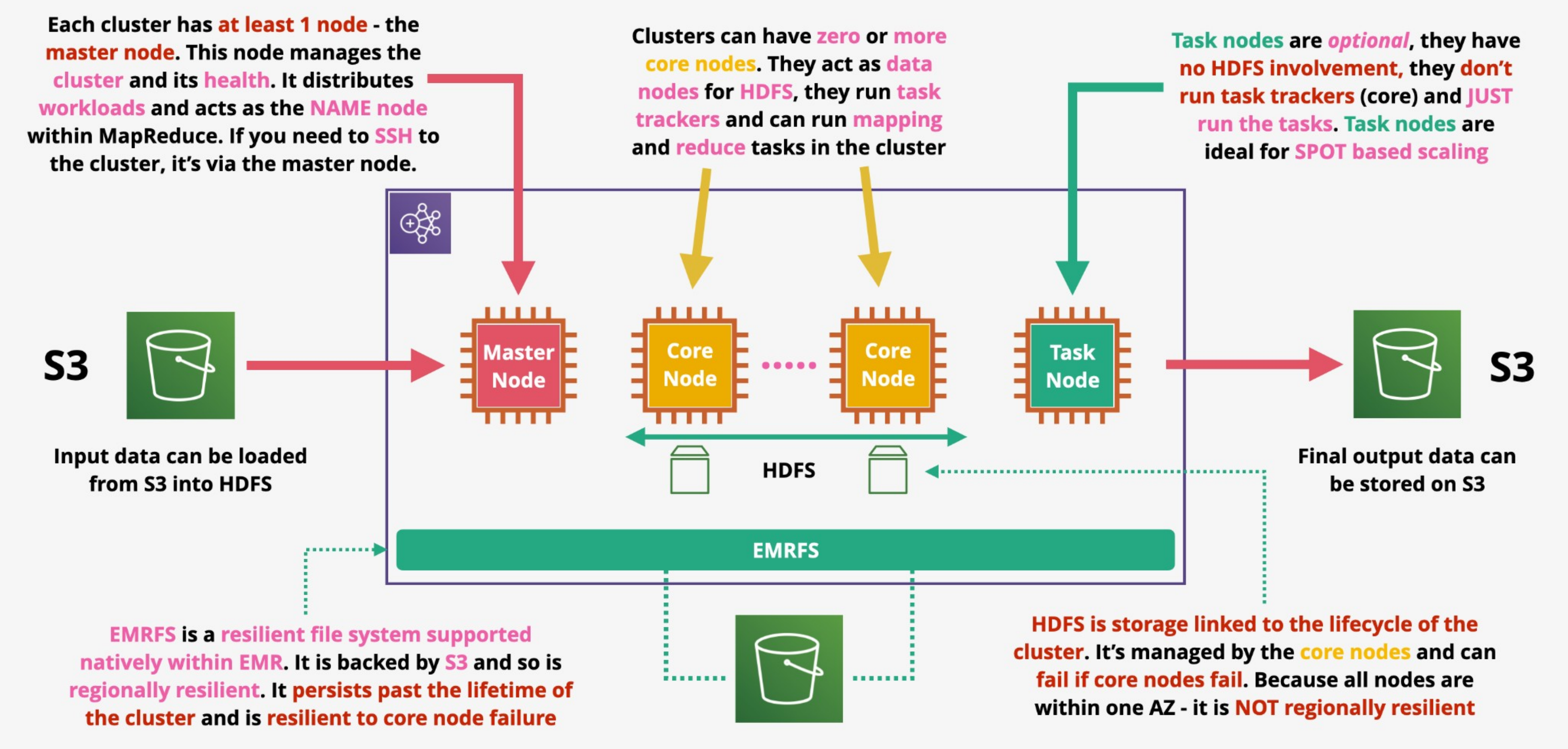
Amazon Web Services	Microsoft Azure	Google Cloud Platform
EMR	HDInsight	Cloud Dataflow
Kinesis	Stream Analytics, Data Lake Analytics, Data Lake Store	Cloud Dataprep (Beta)
Glue	Data Factory, Data Category	Cloud Dataproc
Athena	Data Catalog	Cloud Datalab
QuickSight	Power BI, Power BI Embedded	Genomics
ElasticSearch Service	Marketplace - ElasticSearch	Google Data Studio BETA
Redshift	SQL Data Warehouse	Big Query
AWS Data Pipeline	Azure Bot Service	

AWS EMR

Amazon EMR (Elastic MapReduce) is a **cloud-based big data processing** service.

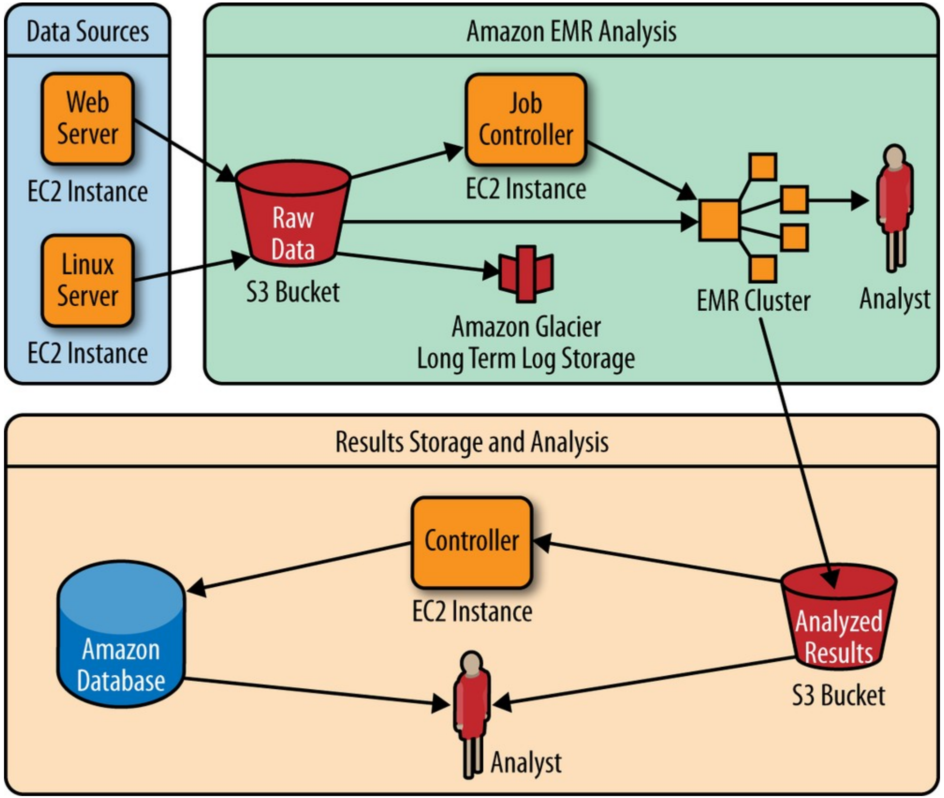
- It helps run **Apache Hadoop, Spark, Presto, and other big data frameworks** efficiently.
- Uses **EC2 instances as nodes** to process large-scale datasets in a distributed manner.
- Stores and retrieves data from **Amazon S3 instead of HDFS** for better scalability.
- Supports **Auto Scaling**, reducing costs by adjusting resources dynamically.
- Ideal for **data analytics, machine learning, and log processing**.

AWS EMR Architecture



Deployment of MapReduce

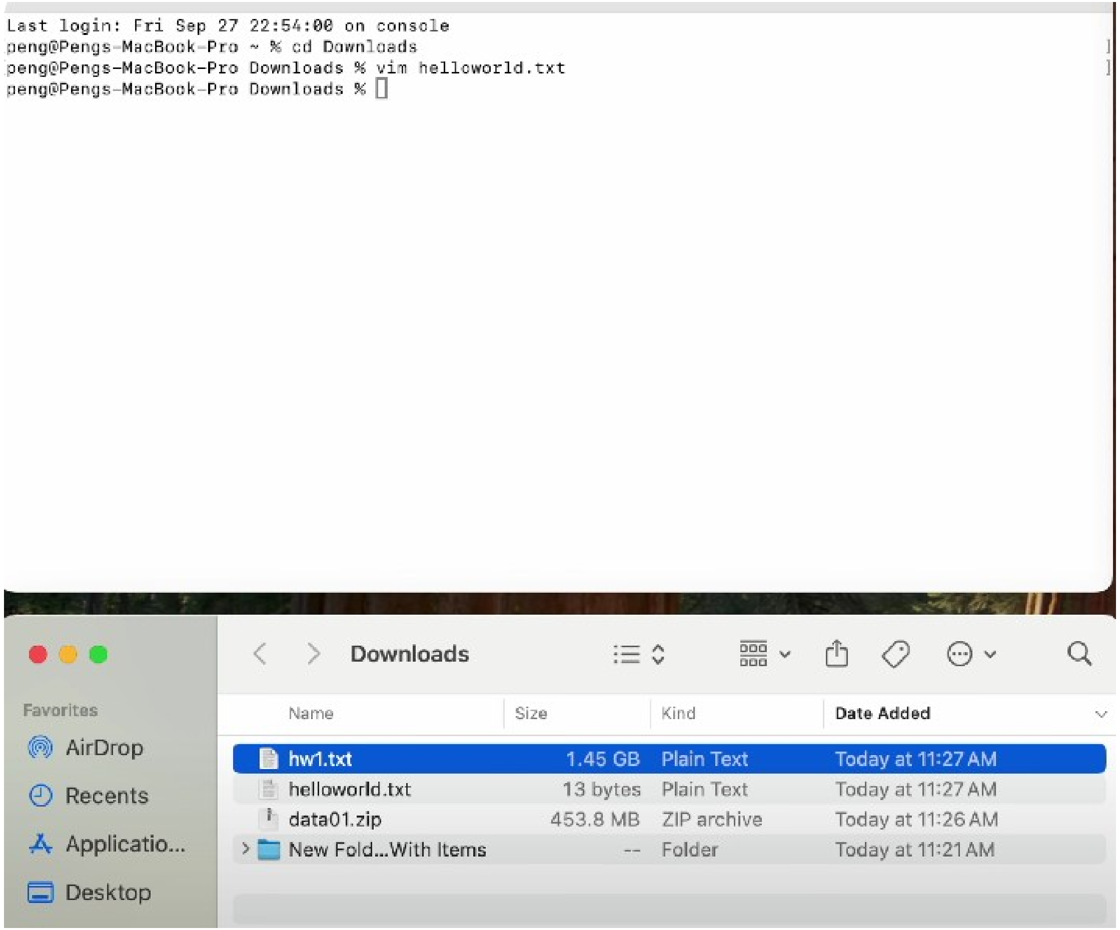
- Amazon EC2 (Elastic Compute Cloud)
- Amazon S3 (Simple Storage Service)
- AWS EMR (Elastic MapReduce)
- Amazon Glacier (Long-Term Storage)
- AWS IAM (Identity & Access Management)
- Amazon Database (RDS/DynamoDB)



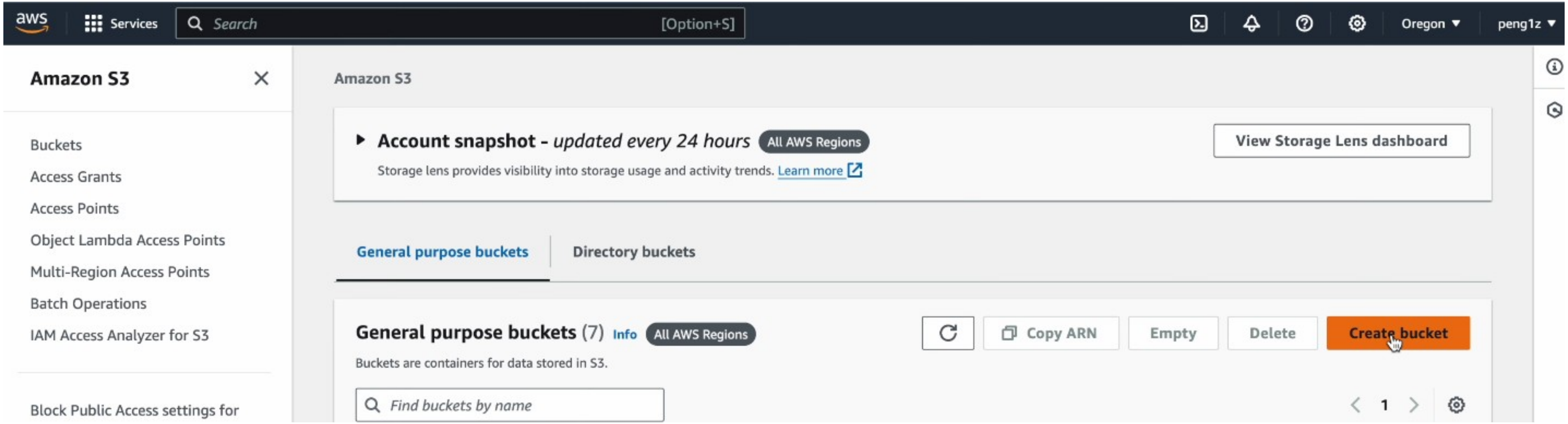
Deploying MapReduce on AWS EMR

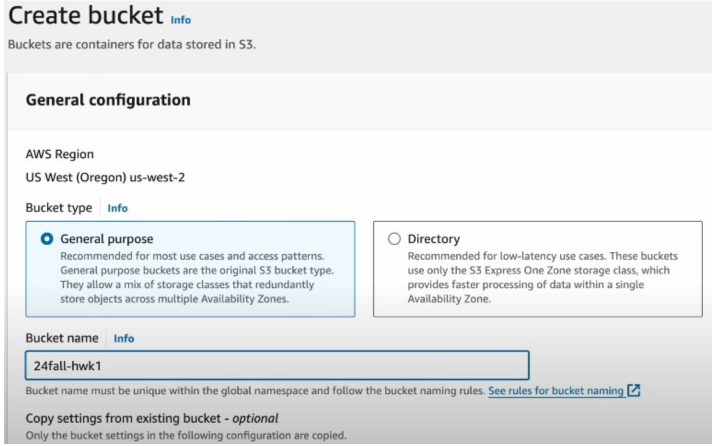
Mohamed Muzammil J (22Z239)

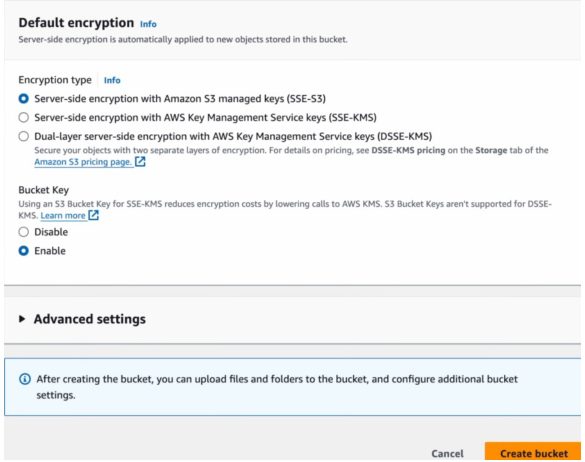
EXAMPLE

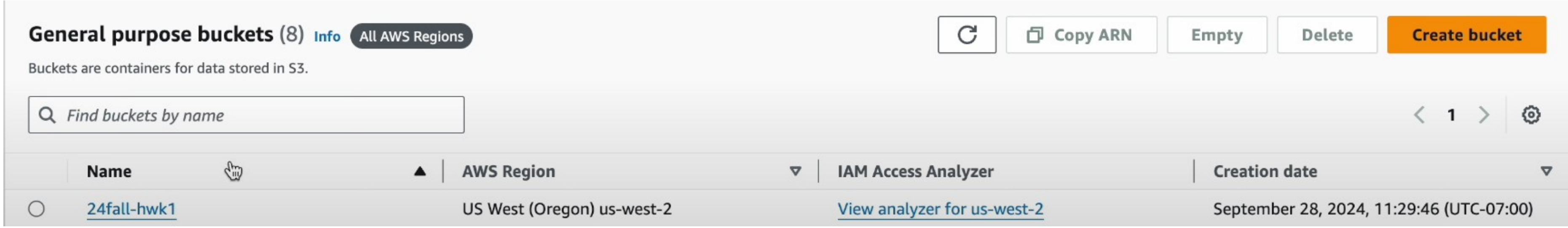


Last login: Fri Sep 27 22:54:00 on console
peng@Pengs-MacBook-Pro ~ % cd Downloads
peng@Pengs-MacBook-Pro Downloads % vim helloworld.txt
peng@Pengs-MacBook-Pro Downloads %
Downloads
Favorites
AirDrop
Recents
Applicatio...
Desktop
Name
Size
Kind
Date Added
hw1.txt 1.45 GB Plain Text Today at 11:27 AM
helloworld.txt 13 bytes Plain Text Today at 11:27 AM
data01.zip 453.8 MB ZIP archive Today at 11:26 AM
New Fold...With Items -- Folder Today at 11:21 AM

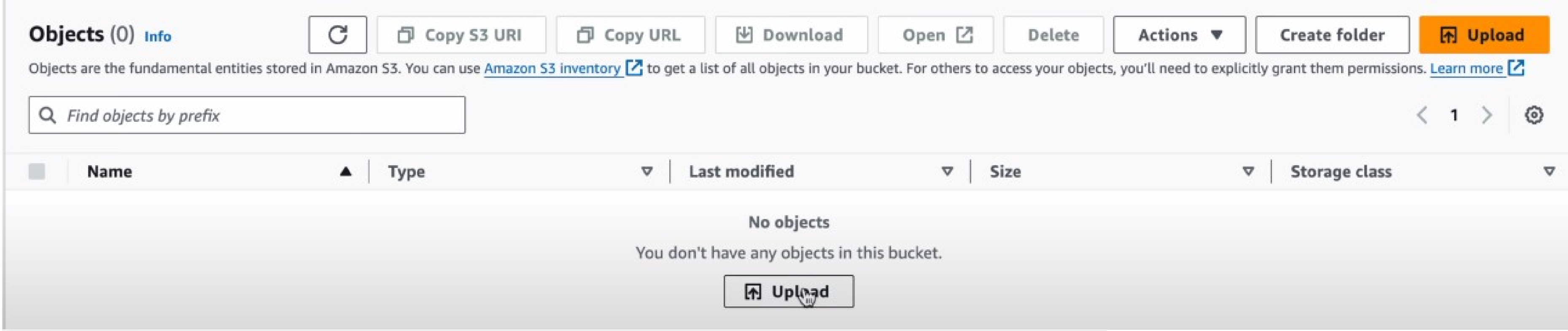

aws
Services
Search
[Option+S]
Oregon
peng1z
Amazon S3
Buckets
Access Grants
Access Points
Object Lambda Access Points
Multi-Region Access Points
Batch Operations
IAM Access Analyzer for S3
Block Public Access settings for
Amazon S3
Account snapshot - updated every 24 hours
All AWS Regions
View Storage Lens dashboard
Storage lens provides visibility into storage usage and activity trends. Learn more
General purpose buckets
Directory buckets
General purpose buckets (7) Info
All AWS Regions
Copy ARN
Empty
Delete
Create bucket
Buckets are containers for data stored in S3.
Find buckets by name
1


Create bucket Info
Buckets are containers for data stored in S3.
General configuration
AWS Region
US West (Oregon) us-west-2
Bucket type Info
General purpose
Recommended for most use cases and access patterns. General purpose buckets are the original S3 bucket type. They allow a mix of storage classes that redundantly store objects across multiple Availability Zones.
Directory
Recommended for low-latency use cases. These buckets use only the S3 Express One Zone storage class, which provides faster processing of data within a single Availability Zone.
Bucket name Info
24fall-hwk1
Bucket name must be unique within the global namespace and follow the bucket naming rules. See rules for bucket naming
Copy settings from existing bucket - optional
Only the bucket settings in the following configuration are copied.

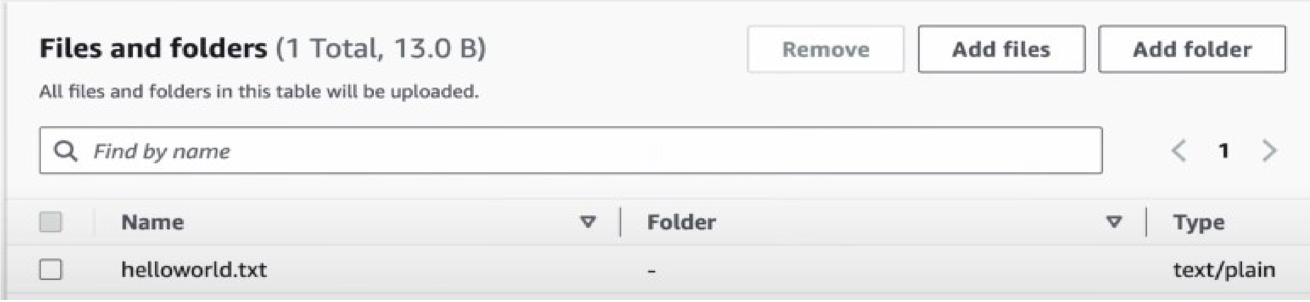

Default encryption Info
Server-side encryption is automatically applied to new objects stored in this bucket.
Encryption type Info
Server-side encryption with Amazon S3 managed keys (SSE-S3)
Server-side encryption with AWS Key Management Service keys (SSE-KMS)
Dual-layer server-side encryption with AWS Key Management Service keys (DSSE-KMS)
Secure your objects with two separate layers of encryption. For details on pricing, see DSSE-KMS pricing on the Storage tab of the Amazon S3 pricing page.
Bucket Key
Using an S3 Bucket Key for SSE-KMS reduces encryption costs by lowering calls to AWS KMS. S3 Bucket Keys aren't supported for DSSE-KMS. Learn more
Disable
Enable
Advanced settings
After creating the bucket, you can upload files and folders to the bucket, and configure additional bucket settings.
Cancel
Create bucket



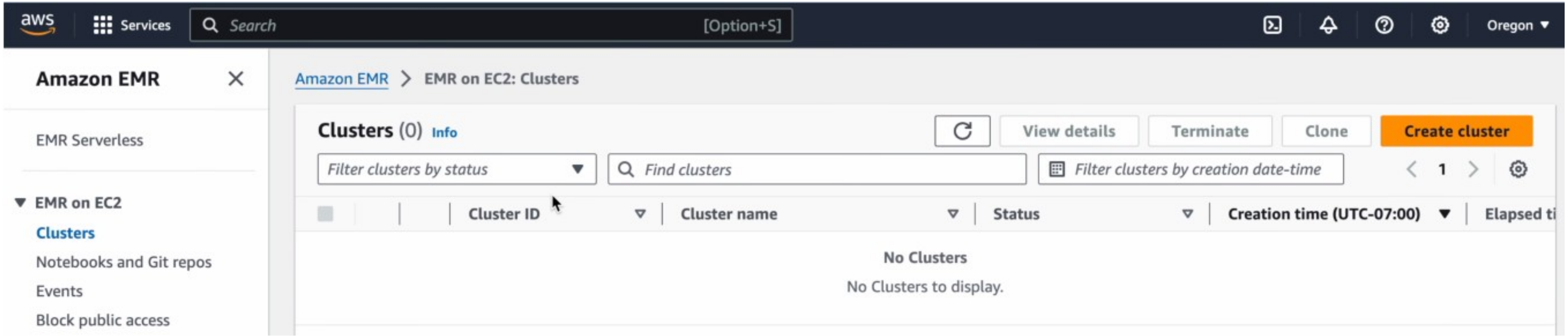
General purpose buckets (8) Info All AWS Regions
Copy ARN
Empty
Delete
Create bucket
Buckets are containers for data stored in S3.
Find buckets by name
1
Name
AWS Region
IAM Access Analyzer
Creation date
24fall-hwk1
US West (Oregon) us-west-2
View analyzer for us-west-2
September 28, 2024, 11:29:46 (UTC-07:00)

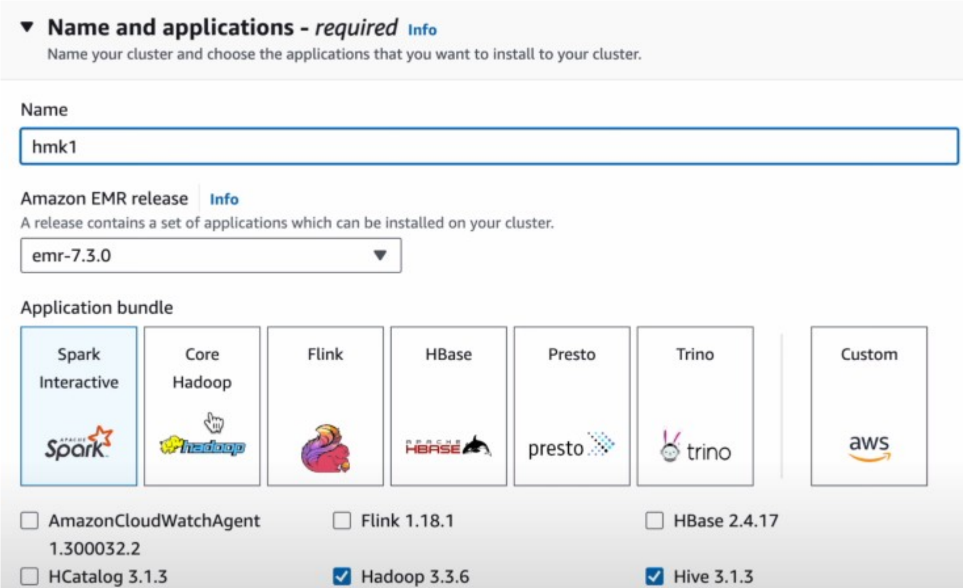


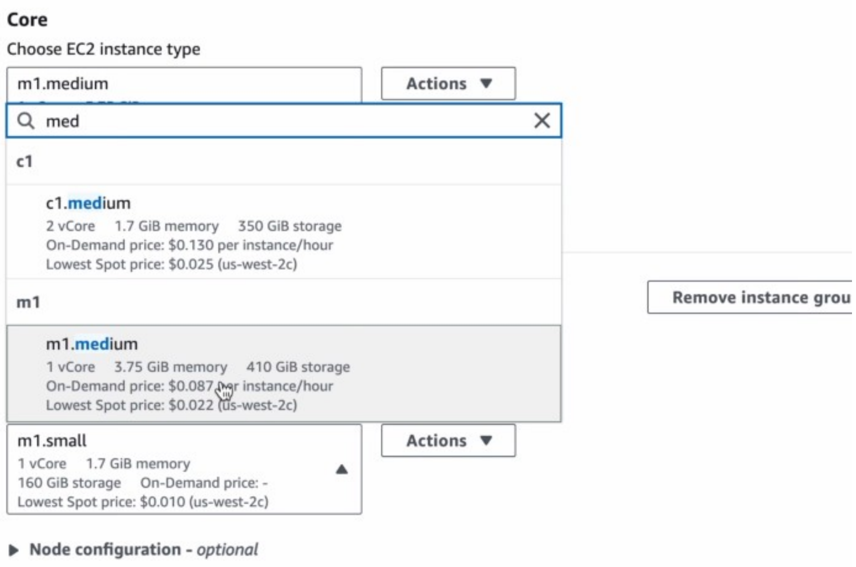
Objects (0) Info
Copy S3 URI
Copy URL
Download
Open
Delete
Actions
Create folder
Upload
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more
Find objects by prefix
1
Name
Type
Last modified
Size
Storage class
No objects
You don't have any objects in this bucket.
Upload

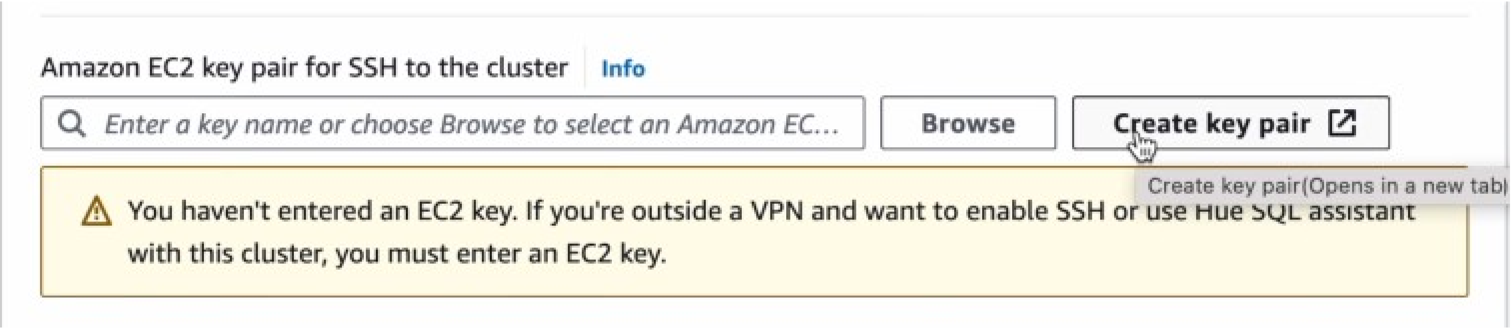


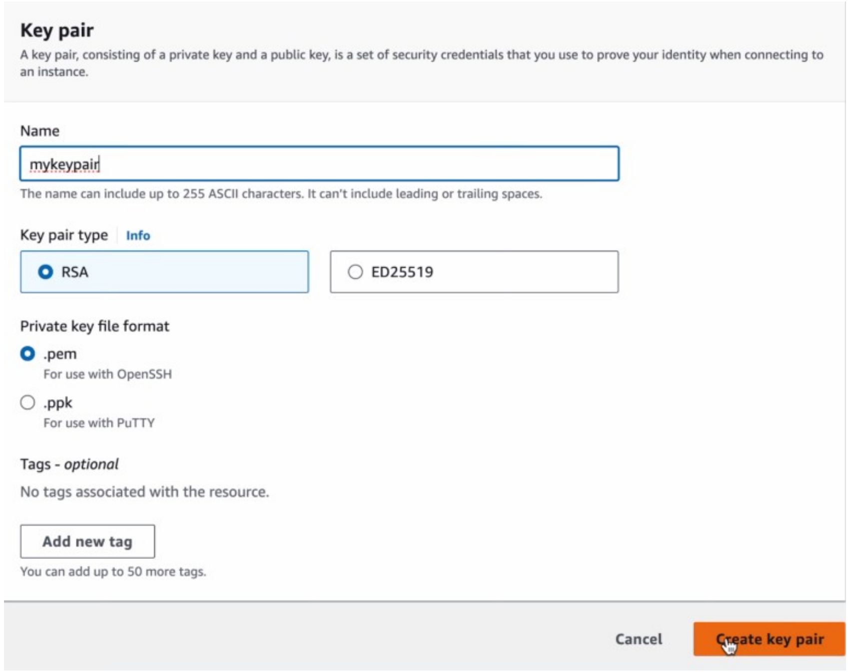
Files and folders (1 Total, 13.0 B)
Remove
Add files
Add folder
All files and folders in this table will be uploaded.
Find by name
1
Name
Folder
Type
helloworld.txt
-
text/plain

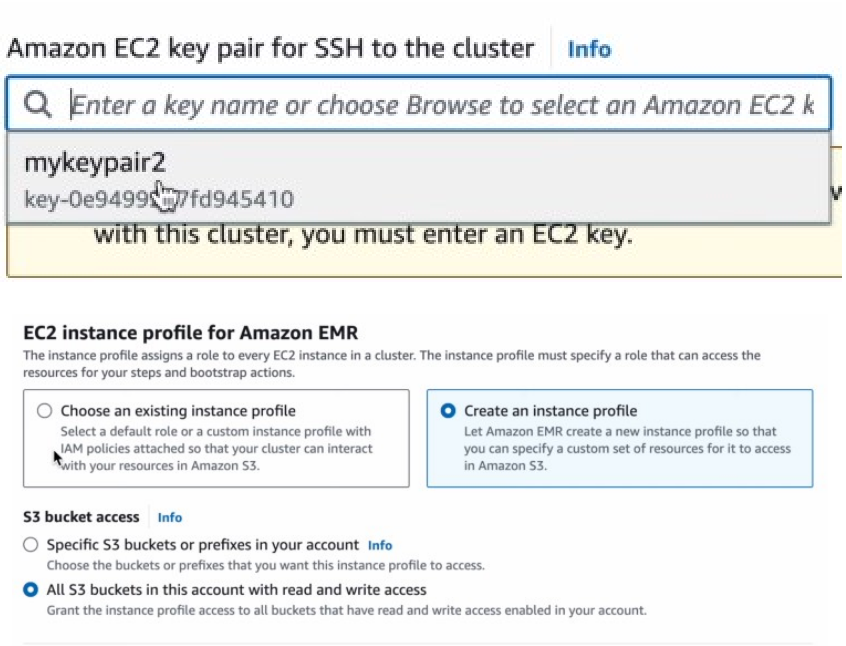

aws
Services
Search
[Option+S]
Oregon
Amazon EMR
EMR Serverless
EMR on EC2
Clusters
Notebooks and Git repos
Events
Block public access
Amazon EMR > EMR on EC2: Clusters
Clusters (0) Info
View details
Terminate
Clone
Create cluster
Filter clusters by status
Find clusters
Filter clusters by creation date-time
1
Cluster ID
Cluster name
Status
Creation time (UTC-07:00)
Elapsed ti
No Clusters
No Clusters to display.

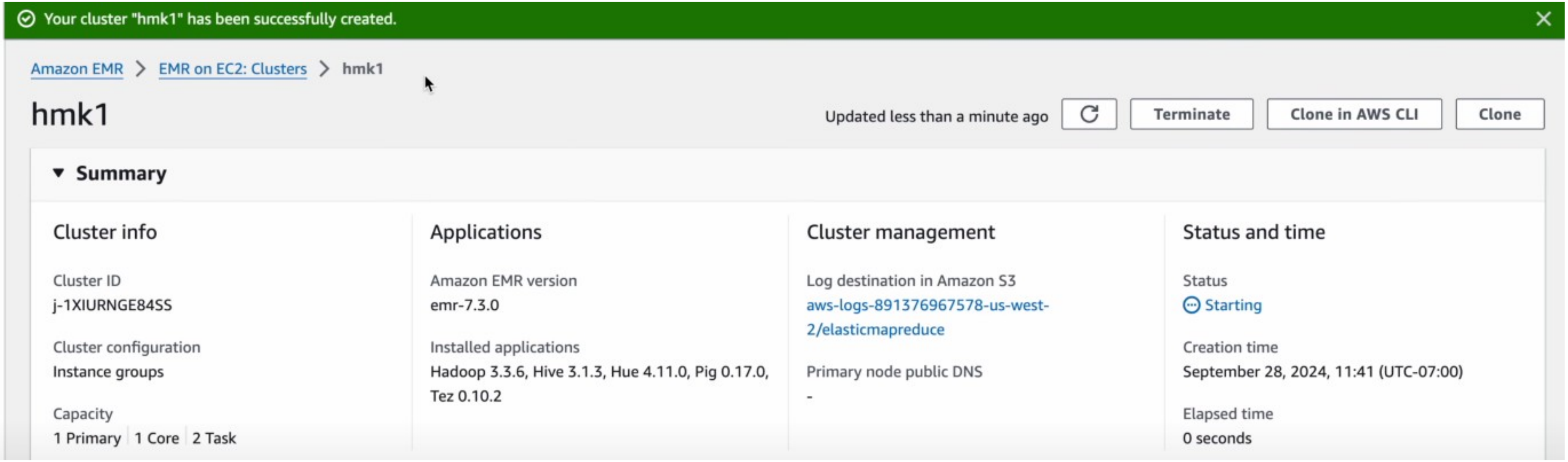

Name and applications - required Info
Name your cluster and choose the applications that you want to install to your cluster.
Name
hmk1
Amazon EMR release Info
A release contains a set of applications which can be installed on your cluster.
emr-7.3.0
Application bundle
Spark Interactive
Core Hadoop
Flink
HBase
Presto
Trino
Custom
AmazonCloudWatchAgent 1.300032.2
Flink 1.18.1
HBase 2.4.17
HCatalog 3.1.3
Hadoop 3.3.6
Hive 3.1.3

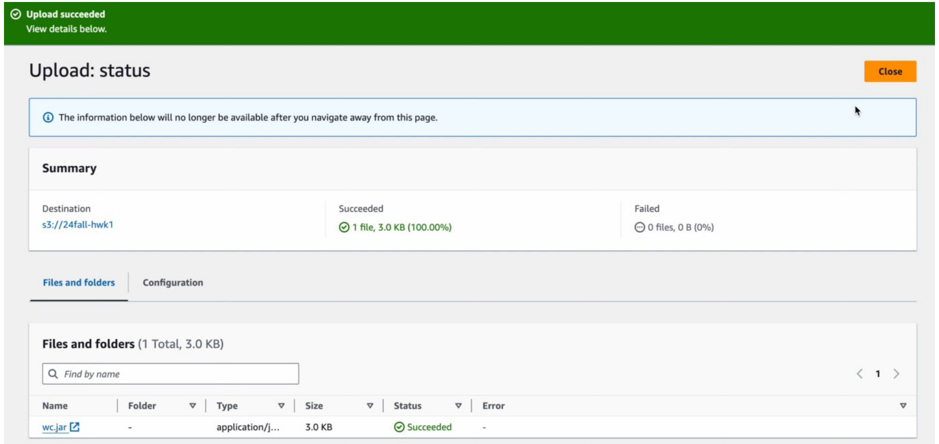

Core
Choose EC2 instance type
m1.medium
Actions
med
c1
c1.medium
2 vCore 1.7 GiB memory 350 GiB storage
On-Demand price: $0.130 per instance/hour
Lowest Spot price: $0.025 (us-west-2c)
m1
m1.medium
1 vCore 3.75 GiB memory 410 GiB storage
On-Demand price: $0.087 per instance/hour
Lowest Spot price: $0.022 (us-west-2c)
Remove instance grou
m1.small
1 vCore 1.7 GiB memory
160 GiB storage On-Demand price: -
Lowest Spot price: $0.010 (us-west-2c)
Actions
Node configuration - optional

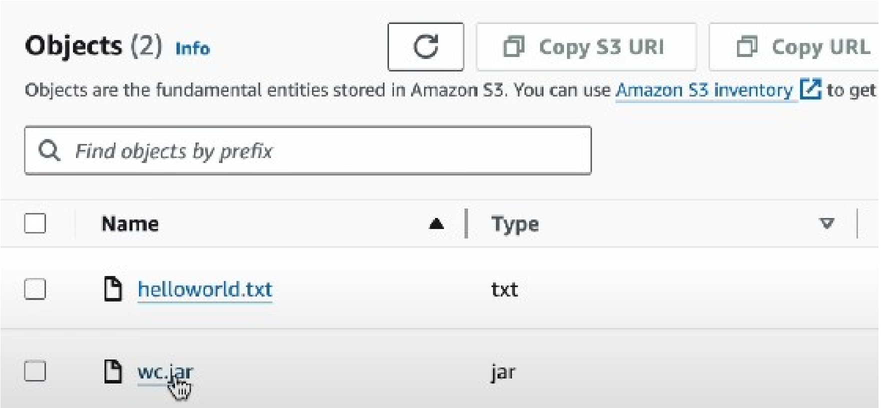

Amazon EC2 key pair for SSH to the cluster
Info
Enter a key name or choose Browse to select an Amazon EC…
Browse
Create key pair
Create key pair(Opens in a new tab)
You haven't entered an EC2 key. If you're outside a VPN and want to enable SSH or use Hue SQL assistant with this cluster, you must enter an EC2 key.

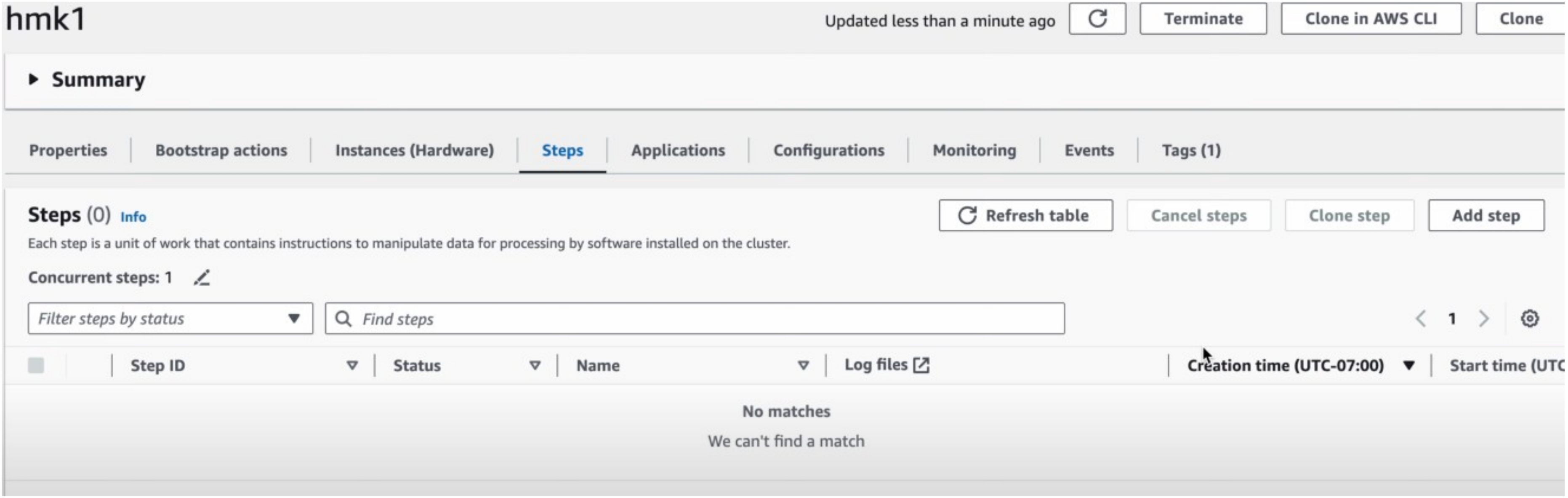

Key pair
A key pair, consisting of a private key and a public key, is a set of security credentials that you use to prove your identity when connecting to an instance.
Name
mykeypair
The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.
Key pair type
Info
RSA
ED25519
Private key file format
.pem
For use with OpenSSH
.ppk
For use with PuTTY
Tags - optional
No tags associated with the resource.
Add new tag
You can add up to 50 more tags.
Cancel
Create key pair

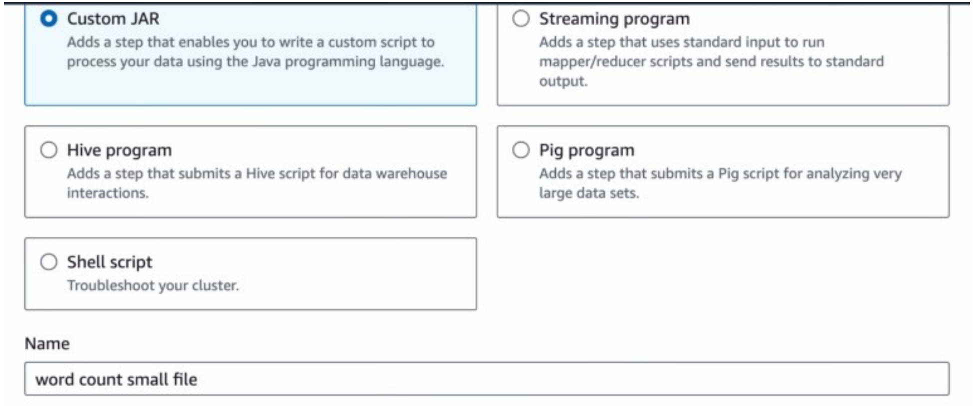

Amazon EC2 key pair for SSH to the cluster
Info
Enter a key name or choose Browse to select an Amazon EC2 k
mykeypair2
with this cluster, you must enter an EC2 key.
EC2 instance profile for Amazon EMR
The instance profile assigns a role to every EC2 instance in a cluster. The instance profile must specify a role that can access the resources for your steps and bootstrap actions.
Choose an existing instance profile
Select a default role or a custom instance profile with IAM policies attached so that your cluster can interact with your resources in Amazon S3.
Create an instance profile
Let Amazon EMR create a new instance profile so that you can specify a custom set of resources for it to access in Amazon S3.
S3 bucket access
Info
Specific S3 buckets or prefixes in your account
Info
Choose the buckets or prefixes that you want this instance profile to access.
All S3 buckets in this account with read and write access
Grant the instance profile access to all buckets that have read and write access enabled in your account.

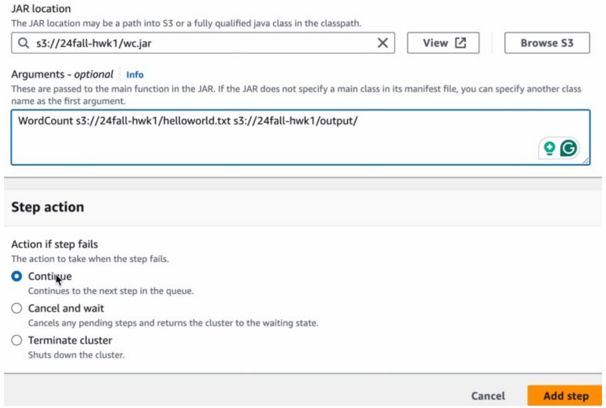

Your cluster "hmk1" has been successfully created.
Amazon EMR > EMR on EC2: Clusters > hmk1
hmk1
Updated less than a minute ago
Terminate
Clone in AWS CLI
Clone
Summary
Cluster info
Cluster ID
j-1XIURNGE84SS
Cluster configuration
Instance groups
Capacity
1 Primary | 1 Core | 2 Task
Applications
Amazon EMR version
emr-7.3.0
Installed applications
Hadoop 3.3.6, Hive 3.1.3, Hue 4.11.0, Pig 0.17.0, Tez 0.10.2
Cluster management
Log destination in Amazon S3
aws-logs-891376967578-us-west-2/elasticmapreduce
Primary node public DNS
-
Status and time
Status
Starting
Creation time
September 28, 2024, 11:41 (UTC-07:00)
Elapsed time
0 seconds

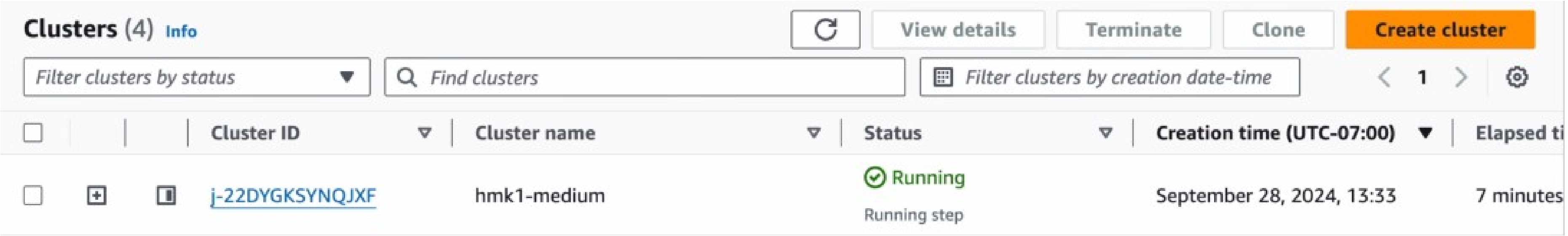

Upload succeeded
View details below.
Upload: status
Close
The information below will no longer be available after you navigate away from this page.
Summary
Destination
s3://24fall-hwk1
Succeeded
1 file, 3.0 KB (100.00%)
Failed
0 files, 0 B (0%)
Files and folders
Configuration
Files and folders (1 Total, 3.0 KB)
Find by name
Name
Folder
Type
Size
Status
Error
wc.jar
-
application/j...
3.0 KB
Succeeded
-

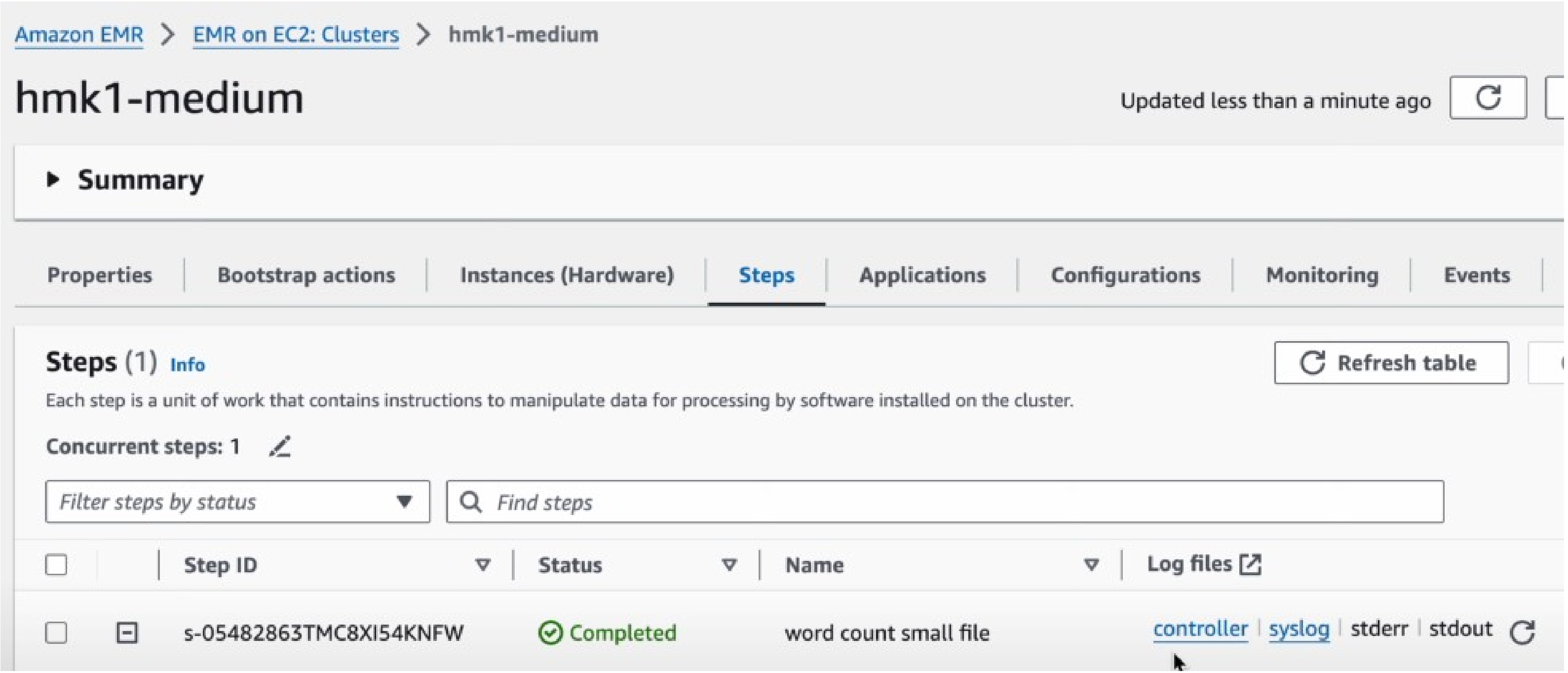

Objects (2) Info
Copy S3 URI
Copy URL
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get
Find objects by prefix
Name
Type
helloworld.txt
txt
wc.jar
jar

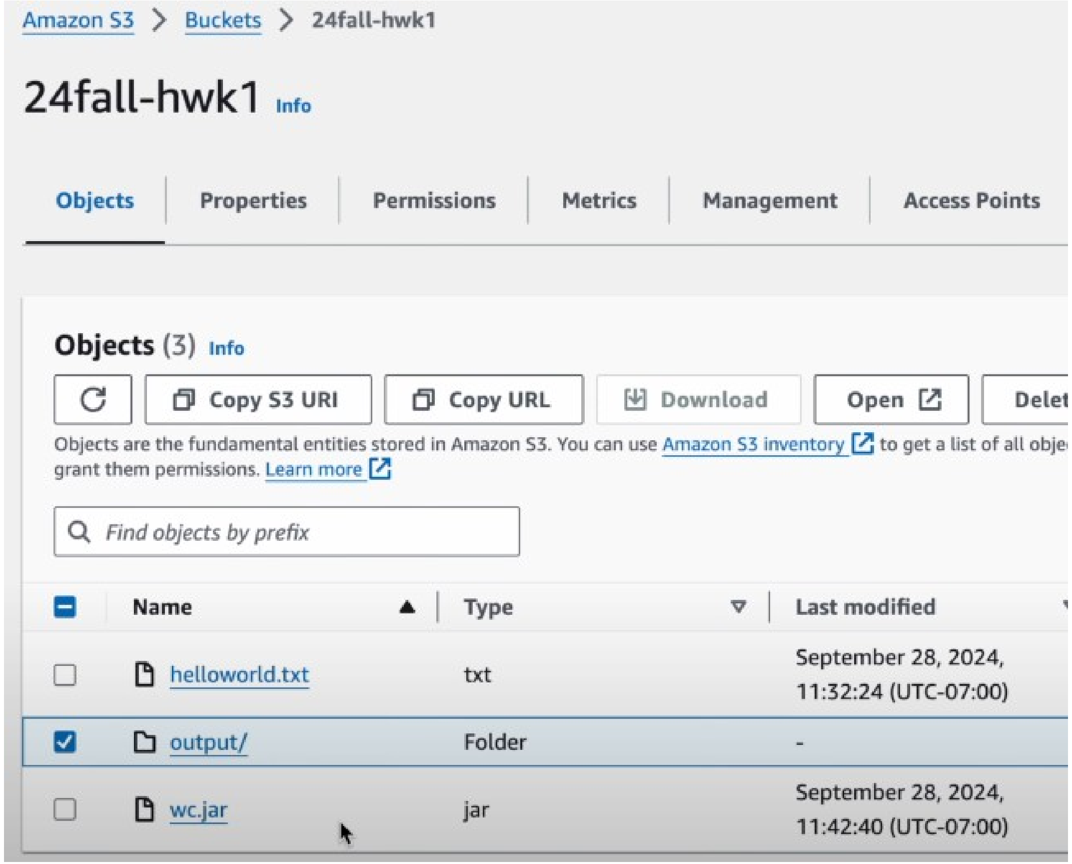

hmk1
Updated less than a minute ago
Terminate
Clone in AWS CLI
Clone
Summary
Properties
Bootstrap actions
Instances (Hardware)
Steps
Applications
Configurations
Monitoring
Events
Tags (1)
Steps (0) Info
Refresh table
Cancel steps
Clone step
Add step
Each step is a unit of work that contains instructions to manipulate data for processing by software installed on the cluster.
Concurrent steps: 1
Filter steps by status
Find steps
1
Step ID
Status
Name
Log files
Creation time (UTC-07:00)
Start time (UTC
No matches
We can't find a match

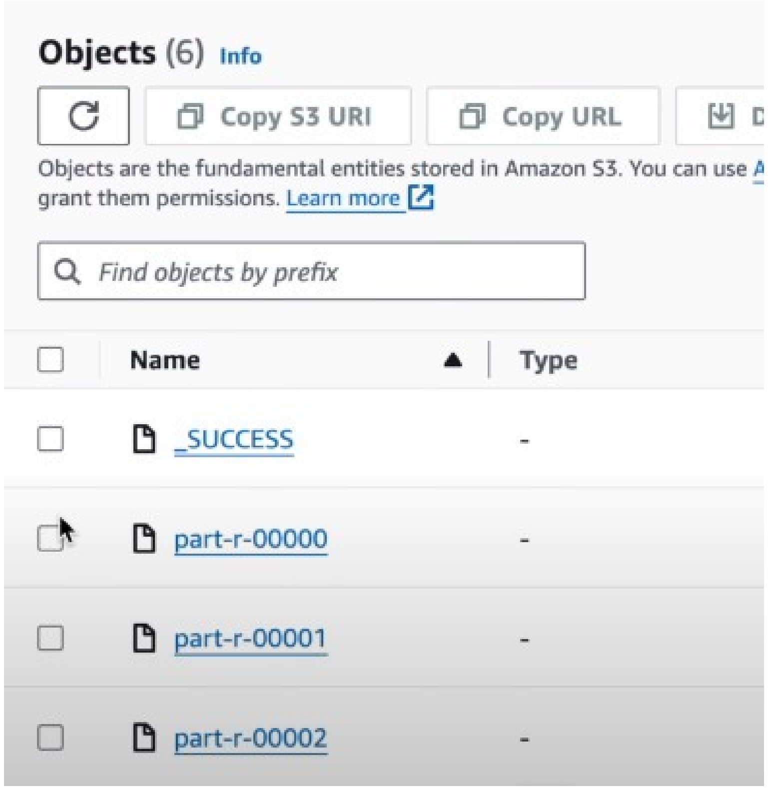

Custom JAR
Adds a step that enables you to write a custom script to process your data using the Java programming language.
Streaming program
Adds a step that uses standard input to run mapper/reducer scripts and send results to standard output.
Hive program
Adds a step that submits a Hive script for data warehouse interactions.
Pig program
Adds a step that submits a Pig script for analyzing very large data sets.
Shell script
Troubleshoot your cluster.
Name
word count small file


JAR location
The JAR location may be a path into S3 or a fully qualified java class in the classpath.
s3://24fall-hwk1/wc.jar
View
Browse S3
Arguments - optional Info
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.
WordCount s3://24fall-hwk1/helloworld.txt s3://24fall-hwk1/output/
Step action
Action if step fails
The action to take when the step fails.
Continue
Continues to the next step in the queue.
Cancel and wait
Cancels any pending steps and returns the cluster to the waiting state.
Terminate cluster
Shuts down the cluster.
Cancel
Add step

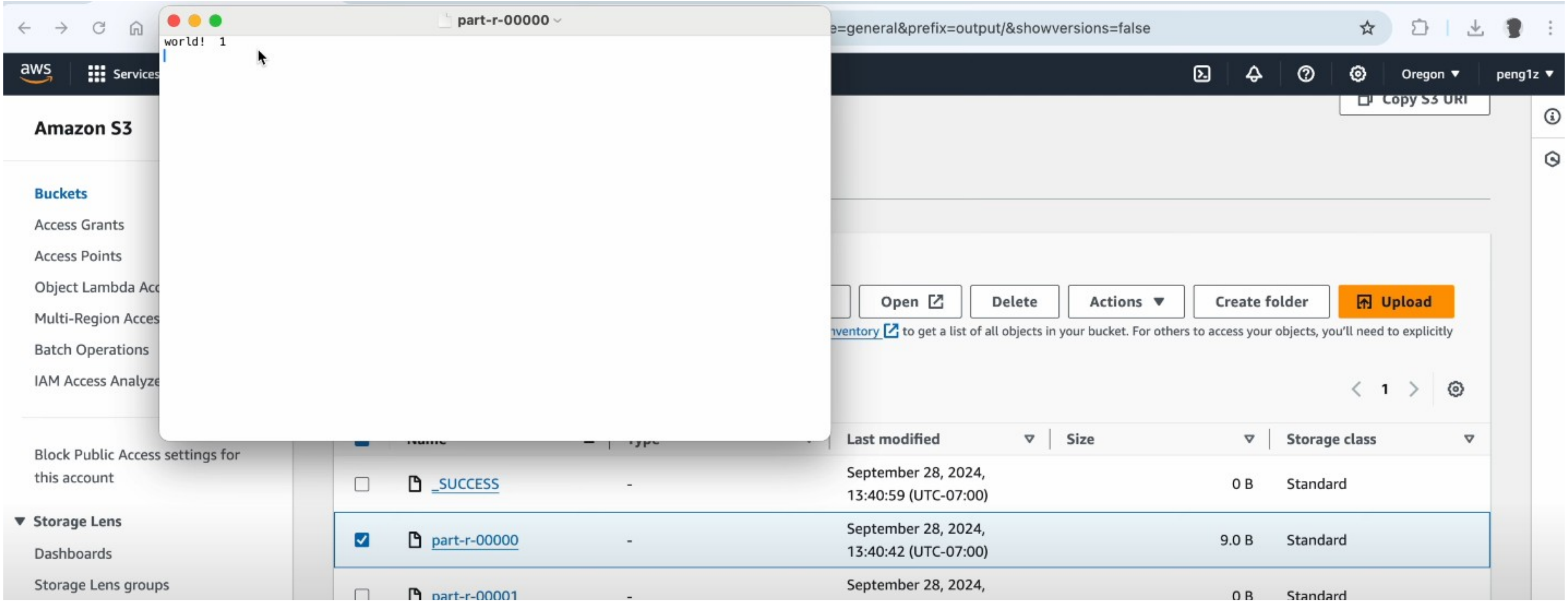

Clusters (4) Info
View details
Terminate
Clone
Create cluster
Filter clusters by status
Find clusters
Filter clusters by creation date-time
1
Cluster ID
Cluster name
Status
Creation time (UTC-07:00)
Elapsed ti
j-22DYGKSYNQJXF
hmk1-medium
Running
Running step
September 28, 2024, 13:33
7 minutes


Amazon EMR > EMR on EC2: Clusters > hmk1-medium
hmk1-medium
Updated less than a minute ago
Summary
Properties
Bootstrap actions
Instances (Hardware)
Steps
Applications
Configurations
Monitoring
Events
Steps (1) Info
Refresh table
Each step is a unit of work that contains instructions to manipulate data for processing by software installed on the cluster.
Concurrent steps: 1
Filter steps by status
Find steps
Step ID
Status
Name
Log files
s-05482863TMC8XI54KNFW
Completed
word count small file
controller
syslog
stderr
stdout


Amazon S3 > Buckets > 24fall-hwk1
24fall-hwk1 Info
Objects
Properties
Permissions
Metrics
Management
Access Points
Objects (3) Info
Copy S3 URI
Copy URL
Download
Open
Delet
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all obje
grant them permissions. Learn more
Find objects by prefix
Name
Type
Last modified
helloworld.txt
txt
September 28, 2024, 11:32:24 (UTC-07:00)
output/
Folder
-
wc.jar
jar
September 28, 2024, 11:42:40 (UTC-07:00)


Objects (6) Info
Copy S3 URI
Copy URL
Objects are the fundamental entities stored in Amazon S3. You can use
grant them permissions. Learn more
Find objects by prefix
Name
Type
_SUCCESS
-
part-r-00000
-
part-r-00001
-
part-r-00002
-

RESULT



part-r-00000
world! 1
e=general&prefix=output/&showversions=false
Services
Oregon
peng1z
Amazon S3
Buckets
Access Grants
Access Points
Object Lambda Acc
Multi-Region Acces
Batch Operations
IAM Access Analyze
Block Public Access settings for this account
Storage Lens
Dashboards
Storage Lens groups
Copy S3 URI
Open
Delete
Actions
Create folder
Upload
nventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly
Last modified
Size
Storage class
_SUCCESS
September 28, 2024, 13:40:59 (UTC-07:00)
0 B
Standard
part-r-00000
September 28, 2024, 13:40:42 (UTC-07:00)
9.0 B
Standard
part-r-00001
September 28, 2024,
0 B
Standard

Common Challenges in MapReduce & Cloud Deployment

Challenges in MapReduce:

- **Scalability Issues** – Handling very large datasets efficiently
- **I/O Bottlenecks** – Disk-based processing slows down execution
- **Fault Tolerance Overhead** – Checkpointing and recovery take time
- **Complex Debugging** – Difficult to troubleshoot errors

Challenges in Cloud Deployment:

- **Security & Data Privacy** – Sensitive data risks in cloud storage
- **Resource Management** – Cost optimization and under-utilization
- **Network Latency** – Delays due to distributed infrastructure
- **Vendor Lock-in** – Dependence on a single cloud provider

Optimization Techniques for Better Performance

MapReduce Optimization:

- **Combiner Functions** – Reduce the amount of data transferred between nodes
- **Data Locality** – Process data closer to where it is stored
- **Efficient Partitioning & Scheduling** – Balance workload distribution
- **Compression & In-Memory Processing** – Reduce storage & improve speed

Cloud Optimization:

- **Auto-scaling** – Dynamically allocate resources based on demand
- **Containerization (Docker, Kubernetes)** – Efficient resource utilization
- **Edge Computing** – Reduce latency by processing data closer to the source
- **Caching & Load Balancing** – Improve response times and availability

Future of Big Data Processing

Spark vs. MapReduce

Feature	MapReduce	Spark
Speed	Slower (Disk-based)	Faster (In-memory)
Ease of Use	Complex (Java, MR jobs)	Simple (Scala, Python)
Performance	Batch Processing	Real-time & Batch