

## Big Data Wrangling With Google Books Ngrams

### **Big Data Wrangling With Google Books Ngrams**

Author: Ja`Mone Bridges

Class: Data Science Bootcamp

Institution: BrainStation

## Table of Contents

Introduction .....	4
Instructions .....	5
• Access AWS.....	5
• Access EMR.....	5
• Setup Cluster .....	5
• Cluster Configuration.....	5
• Cluster CPU Core Selection .....	5
• Cluster scaling and provisioning .....	5
• Cluster Security .....	6
• Cluster Identity and Access Management (IAM) roles.....	6
• EMR and Spark Cluster .....	6
• Connect to Hadoop with SSH.....	6
• Large File Transfer to Hadoop .....	7
• Setup JupyterHub Secure Port for Localhost.....	7
• Connect to JupyterHub in Browser .....	7
• PySpark.....	7
• Retrieve the Data from Hadoop to the local machine .....	8
• Local Jupyter Notebook.....	8
• Store the data locally .....	9
• Terminate the Cluster .....	9
Findings .....	10
Summary.....	11
Appendix.....	12
Appendix A .....	12
Appendix B.....	13
Appendix C.....	14
Appendix D.....	15
Appendix E .....	16
Appendix F .....	17
Appendix G.....	18
Appendix H.....	19

---

Big Data Wrangling With Google Books Ngrams

Appendix I.....	20
Appendix J.....	21
Appendix K.....	22
Appendix L.....	23
Appendix M.....	24
Appendix N.....	25

## Introduction

The purpose of this report is to instruct the user how to use AWS (Amazon Web Services) cloud services to process large datasets and databases. We will spin up (startup) a cluster with three primary services. The first service, Hadoop, allows for large volumes of data to be stored and retrieved at high speed. The second service, PySpark and JupyterHub allows users to perform analysis with RDD (Resilient Distributed Dataset). The third service, S3, is AWS secure cloud data storage that can be public or private. S3 allows access from the web to read or write data.

This report will refer to AWS setup pages and CLI (Command Line Interface) commands. The images of the AWS and CLI are in the appendix. The report will be referenced in the appendix by letters.

---

Big Data Wrangling With Google Books Ngrams

## Instructions

- **Access AWS:** Goto URL <https://aws.amazon.com/> and click on “Sign in to the Console”.  
If you need to create a new account, please do so now.
- **Access EMR:** After logging on to AWS, in the upper left is a search bar. See appendix A. Search for EMR and select it.
- **Setup Cluster:** Make sure that Clusters are selected under “EMR on EC2” on the left side panel. Then click on “Create cluster”. Refer to appendix B for display.
- **Cluster Configuration:** In the “Name and Applications” field, give the cluster a name. The name can be anything suitable. Under “Amazon EMR release” select “emr-6.10.0”. For the Application bundle select Custom. All of the following checkboxes must be selected, “Hue – Livy – Spark – Hadoop – Hive – JupyterHub”. The “Operating system option” has “Amazon Linux release”. Please, refer to appendix C.
- **Cluster CPU Core Selection:** For “Cluster configuration” select the radio button “Uniform instance groups”. Under the “Primary” header and “Choose EC2 instance type” select “m5.xlarge”. Under “Core” and “Choose EC2 instance type” select "m5.xlarge". Please, refer to appendix D for a visual.
- **Cluster scaling and provisioning:** Select option “Set cluster size manually”. For “Provisioning configuration” type 2 under “Instance(s) size. Under header “Networking” pick options for “Virtual private cloud” and “Subnet”. If no options are available, then click on the respective “Create” button on the right. Please, refer to appendix E if a visual guide is desired.

- **Cluster Security:** Under “Security configuration and EC2 key pair” look for heading “Amazon EC2 key pair for SSH to the cluster”. If a key pair does not exist, then click on the “Create key pair” button to make one. Be sure to select the \*.pem option when making and new key pair. NOTE: Please, be aware that the key pair will only work in the AWS region that it is created for. Store the downloaded key \*.pem file in a location to be used later to connect to the cluster. Next, browse and select the key pair under “Amazon EC2 key pair for SSH to the cluster”. For a visual aid please refer to appendix F.
- **Cluster Identity and Access Management (IAM) roles:** If there is an existing role select it for “Service Role” and “Instance profile”, then select “Create cluster” button. Otherwise, click on “Create a service role” and follow the instructions. Once the service role has been created select and perform the same process for “Instance profile“. Now click on the “Create cluster” button. Please refer to append F for a visual aid.
- **EMR and Spark Cluster:** The cluster has now been created and the system is now installing the software and booting the system. It will take several minutes for the process to complete. Under “Status and time” on the right side of the window “Status” will change to “Waiting” when the cluster is ready to be used.
- **Connect to Hadoop with SSH:** Once the status changes to “Waiting”, under “Cluster management” click on “Connect to the Primary node using SSH” and copy the connection link. Open a CLI or terminal with ssh access and paste the connection link. The address of the key pair (\*.pem file) will likely require changing. Once the address to the key pair is correct press enter. The window will appear the same as appendix G.

## Big Data Wrangling With Google Books Ngrams

- Large File Transfer to Hadoop:** On the CLI input command “hadoop distcp s3://brainstation-dsft/eng\_1M\_1gram/eng\_1M\_1gram.csv”. Now the data will be downloaded and stored directly to the child nodes of Hadoop. For a visual please refer to appendix I and J.
- Setup JupyterHub Secure Port for Localhost:** Open a new CLI or terminal. Type in command “ssh -I /Path/To/Key.pem -L 9995:localhost:9443 [hadoop@xxxxxxxxxxxxxxxx.compute.amazonaws.com](mailto:hadoop@xxxxxxxxxxxxxxxx.compute.amazonaws.com)”. Replace the x’s with the connect link portion used to connect to Hadoop via SSH. The window will be like appendix K.
- Connect to JupyterHub in Browser:** Open a web browser and type in the following URL <https://localhost:9995>. A warning may appear regarding the expiration of the certificate, go to advance, and proceed to the site. At the logon page enter username “jovyan” and password “jupyter”. A successful login will similar to appendix N. Click new in the upper right corner, then select ”PySpark”.
- PySpark:** Please refer to appendix L and M. Type spark in the first line and run to start the spark session. Follow the command in appendix L and M to open the data file on the Hadoop server. Then perform a sanity check on the data to ensure that it’s loaded. Now, check the number of rows and columns. The shape of the dataset is (261,823,225, 5). The column name are “token, year, frequency, pages, and books”. To run a SQL statement on a PySpark dataframe it must be converted to a view. See appendix M In [8] for the command. A sql statement of “SELECT \* FROM eng\_table WHERE token LIKE ‘%data%’ is ran in order to get all the tokens with the work data. The data is stored in a new PySpark dataframe. The count of the new dataframe is 24,642. A sanity check is also performed on the new dataframe. The new dataframe is then written back to the

## Big Data Wrangling With Google Books Ngrams

Hadoop child nodes with command “`newdf.write.csv(path, header=True)`”. See appendix M for the path of how to store the data to Hadoop. PySpark is then stopped with command `spark.stop()`.

- Retrieve the Data from Hadoop to the local machine:** In the CLI that is connected to Hadoop type the command “`hadoop fs -getmerge /user/hadoop/eng_1M_1gram/eng_token_data.csv`”. This will combine all the files into a single file and store it on the cluster Linux file system. To gain access to the file for the local machine, it is stored in an S3 bucket. IAM with a user setup with a key and secret must be used to access the S3 bucket from the local machine. Search the internet on how to setup the IAM account. Store the dataset on the S3 bucket with command “`s3-dist-cp eng_token_data.csv – dest s3://bucketname`”.
- Local Jupyter Notebook:** To access the dataset with python verify that the boto3 package is installed. If not, you can use the command “`! pip install boto3`” in python to install the package. To use the boto3 package “`import boto3`” and then set up the connection to the S3 bucket. The access key, secret key, and the name of the region the S3 bucket is hosted is required for the next command. “`S3 = boto3.client('s3', aws_access_key_id='', aws_secret_access_key='', region_name='')`”. The following commands will load the dataset into a pandas dataframe. Import pandas first with command “`import pandas as pd`”. Access the dataset with command “`obj = s3.get_object(Bucket='BucketName', Key='eng_token_data.csv')`”. Store the dataset in a pandas dataframe with “`df = pd.read_csv(obj['Body'])`”.



---

Big Data Wrangling With Google Books Ngrams

- **Store the data locally:** Perform a sanity check on the data with ‘df.head(10)’ and “df.shape”. If the data is present, then store the data with “df.to\_csv(‘path/to/store/file/filename.csv’).
- **Terminate the Cluster:** Go to the browser with the cluster console. Click on “Clusters” on the left-hand panel. View appendix B to verify the correct page. Click on the active cluster. On the next page click “terminate”. This will stop any additional charges now that the cluster is no longer required. The account can now be logged off and all the related windows closed.

### Findings

The dataset with tokens of 'data' is a subset of eng\_1M\_1gram. Hadoop and PySpark were used to analyze and filter the eng\_1M\_1gram dataset. Once the dataset was filtered, it was saved to Hadoop. The file on the Hadoop file system was split into 40 files, one for each partition of PySpark's RDDs. To consolidate the files into a single file and store it on the file system of the server "hadoop fs -getmerge" command was used. Then the file was transferred to an S3 bucket so it could be accessed by my local machine. The file was downloaded in a python notebook using the boto3 package. The cleaning of the data revealed that there were years missing information amounting to 0.47%. Because of the small number of missing values, the rows were removed instead of imputed. The final data shape of the subset was (24,642, 5). The bar graph shows exponential growth in the token 'data' from the 1500's to current times. A logged scale was used to make the graph readable. The number of tokens with 'data' was very sparse from 1500 to 1700 and then the token took off with exponential growth. The likely reason for the initial growth is that the number of published books grew at the same rate. An analysis of the original dataset would have to be conducted to test this hypothesis. After 1970 the growth is likely due to the rise of computers in the mainstream.

### Summary

The Hadoop HDFS stores data through distribution to many Child Nodes. The Head Node breaks up an incoming file up into equal size pieces. The size of the pieces of the file are such that it requires a low amount of memory in both the Head Node and the Child Nodes. Then, the Head Node assigns the Child Nodes of each piece of data that will be stored and transfers the data to those Child Nodes. There are multiple copies of each piece of data that are stored on different Child Nodes. In case of the event that a Child Node fails other Child Nodes have a copy of the same data.

Hadoop and Spark are different systems with different purposes. Hadoop is made to store and retrieve large volumes of data at high speed. It has redundancies (duplicated data) on separate nodes so if a small number of nodes fail the system is still functional. Hadoop was designed for commodity hardware to keep the cost of hardware and maintenance down. It was also designed so that it scales horizontally requiring minimal setup to expand the system. Spark is pandas for very large datasets. Spark runs on clusters allowing it to divide the amount of work and use the servers in the cluster. It accomplishes this by splitting the data into separate groups for each CPU core to process. The system the does this in Spark is called RDD (Resilient Distributed Dataset). Spark also allows users to run SQL queries directly on Spark dataframes. This removes the need to learn all the differences between Spark and pandas commands. Together, Hadoop and Spark are a very powerful combination for Data Scientist working on large dataset. They offer scalable storage and computational power.

## Big Data Wrangling With Google Books Ngrams

## Appendix

## Appendix A

Console Home | Console Home x +

us-east-2.console.aws.amazon.com/console/home?region=us-east-2

Services Search [Alt+S]

Ohio JBAWS

### Console Home

Reset to default layout + Add widgets

#### Recently visited

- EMR
- Billing and Cost Management
- EC2
- S3
- IAM

[View all services](#)

#### Applications (0)

Region: US East (Ohio)

us-east-2 (Current Region) Find applications

< 1 >

Name	Description	Region	Originating account
No applications			
Get started by creating an application to view your application cost, security findings, and metrics all in one place.			

[Create application](#)

[Go to myApplications](#)

#### Welcome to AWS

[Getting started with AWS](#)

Learn the fundamentals and find valuable information to get the most out of AWS.

[Training and certification](#)

Learn from AWS experts and advance your skills and knowledge.

#### AWS Health

Open issues: 0 (Past 7 days)

Scheduled changes: 0 (Upcoming and past 7 days)

Other notifications: 0 (Past 7 days)

#### Cost and usage

Last month costs: **\$0.00**

Average month costs: **\$0.56**

Top costs for current month

Amazon Elastic Comput...	\$2.47
Amazon Elastic MapRed...	\$0.52

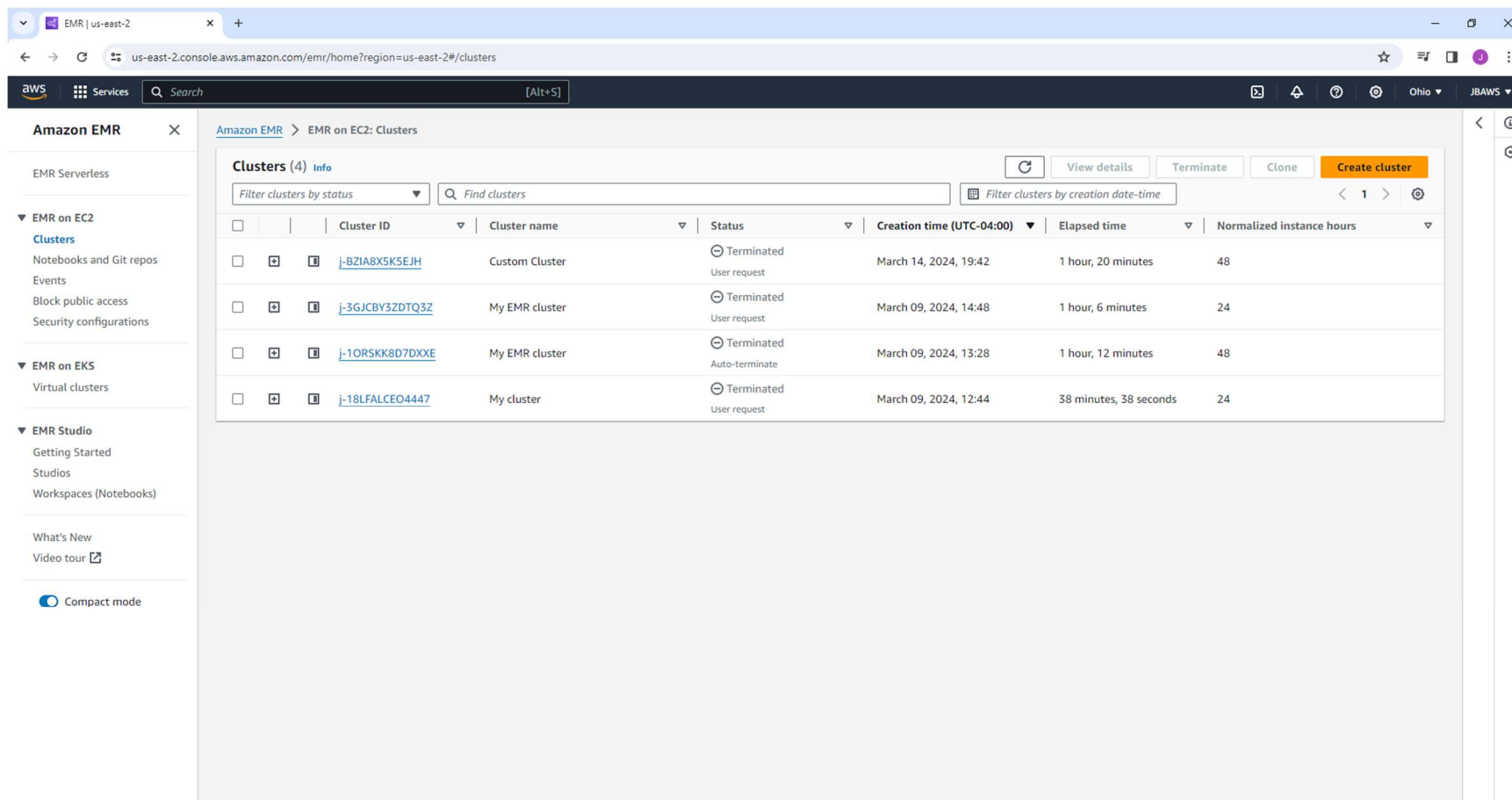
Month (Year): Oct 23, Nov 23, Dec 23, Jan 24, Feb 24, Mar 24

Mar 24 breakdown:

EC2 - Compute	\$2.47
Elastic MapReduce	\$0.52
Tax	\$0.39
EC2 - Other	\$0.00

## Big Data Wrangling With Google Books Ngrams

## Appendix B



The screenshot displays the Amazon EMR console interface. The left sidebar shows the navigation menu with options like EMR Serverless, EMR on EC2 (selected), EMR on EKS, and EMR Studio. The main content area is titled "Clusters (4) info" and shows a table of four terminated EMR clusters. The table columns include Cluster ID, Cluster name, Status, Creation time (UTC-04:00), Elapsed time, and Normalized instance hours. The clusters are all in a "Terminated" state, with reasons like "User request" or "Auto-terminate".

	Cluster ID	Cluster name	Status	Creation time (UTC-04:00)	Elapsed time	Normalized instance hours
<input type="checkbox"/>	<a href="#">j-BZIA8X5K5EJH</a>	Custom Cluster	Terminated User request	March 14, 2024, 19:42	1 hour, 20 minutes	48
<input type="checkbox"/>	<a href="#">j-3GJCBY3ZDTQ3Z</a>	My EMR cluster	Terminated User request	March 09, 2024, 14:48	1 hour, 6 minutes	24
<input type="checkbox"/>	<a href="#">j-1ORSKK8D7DXXE</a>	My EMR cluster	Terminated Auto-terminate	March 09, 2024, 13:28	1 hour, 12 minutes	48
<input type="checkbox"/>	<a href="#">j-18LFALCEQ4447</a>	My cluster	Terminated User request	March 09, 2024, 12:44	38 minutes, 38 seconds	24

## Big Data Wrangling With Google Books Ngrams

## Appendix C

**Create cluster** [Info](#)

▼ **Name and applications - required** [Info](#)  
Name your cluster and choose the applications that you want to install on your cluster.

Name  
EMR and Spark cluster

Amazon EMR release [Info](#)  
A release contains a set of applications which can be installed on your cluster.  
emr-6.10.0

Application bundle

Spark	Core Hadoop	HBase	Presto	Trino	Custom

☐ Flink 1.16.0  
☐ HCatalog 3.1.3  
☒ Hue 4.10.0  
☒ Livy 0.7.1  
☐ Phoenix 5.1.2  
☒ Spark 3.3.1  
☐ Tez 0.10.2  
☐ ZooKeeper 3.5.10

☐ Ganglia 3.7.2  
☒ Hadoop 3.3.3  
☐ JupyterEnterpriseGateway 2.6.0  
☐ MXNet 1.9.1  
☐ Pig 0.17.0  
☐ Sqoop 1.4.7  
☐ Trino 403

☐ HBase 2.4.15  
☒ Hive 3.13  
☒ JupyterHub 1.5.0  
☐ Oozie 5.2.1  
☐ Presto 0.278  
☐ TensorFlow 2.11.0  
☐ Zeppelin 0.10.1

**AWS Glue Data Catalog settings**  
Use the AWS Glue Data Catalog to provide an external metastore for your application.  
☐ Use for Hive table metadata  
☐ Use for Spark table metadata

**Operating system options** [Info](#)  
☒ Amazon Linux release  
☐ Custom Amazon Machine Image (AMI)

**Summary** [Info](#)

**Name and applications - required**

Name  
EMR and Spark Cluster

Amazon EMR release  
emr-6.10.0

Application bundle  
Custom (Hadoop 3.3.3, Hive 3.1.3, Hue 4.10.0, JupyterHub 1.5.0, Livy 0.7.1, Spark 3.3.1)

**Cluster configuration - required**

Uniform instance groups  
Primary (m5.xlarge), Core (m5.xlarge), Task (m5.xlarge)

**Cluster scaling and provisioning - required**

Provisioning configuration  
Core size: 1 instance

**Configure IAM roles**  
You must choose a service role and instance profile before you create this cluster.  
[Choose IAM roles](#)

Cancel [Create cluster](#)

CloudShell Feedback

© 2024, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

## Big Data Wrangling With Google Books Ngrams

## Appendix D

The screenshot shows the AWS Management Console interface for creating an EMR cluster. The browser address bar indicates the URL: `us-east-2.console.aws.amazon.com/emr/home?region=us-east-2#/createCluster`. The console header includes the AWS logo, a search bar, and navigation links for Services, Ohio, and JBAWS.

The main content area is divided into two columns. The left column, titled "Cluster configuration - required", contains the following sections:

- Cluster configuration - required** (Info icon): Choose a configuration method for the primary, core, and task node groups for your cluster.
  - ☒ **Uniform instance groups**: Choose the same EC2 instance type and purchasing option (On-Demand or Spot) for all nodes in your node group. [Learn more](#)
  - ☐ **Flexible instance fleets**: Choose from the widest variety of provisioning options for the EC2 instances in your cluster. Diversify instance types and purchasing options, and use an allocation strategy. [Learn more](#)
- Uniform instance groups**
  - Primary**: Choose EC2 instance type. A dropdown menu shows `m5.xlarge` with details: 4 vCore, 16 GiB memory, EBS only storage, On-Demand price: \$0.192 per instance/hour, Lowest Spot price: \$0.069 (us-east-2a). An "Actions" button is next to it.
  - ☐ **Use high availability**: Launch highly available, more resilient cluster with three primary nodes on On-Demand Instances. This configuration applies for the lifetime of your cluster. [Learn more](#)
  - Node configuration - optional**: Expandable section.
- Core**: Choose EC2 instance type. A dropdown menu shows `m5.xlarge` with details: 4 vCore, 16 GiB memory, EBS only storage, On-Demand price: \$0.192 per instance/hour, Lowest Spot price: \$0.069 (us-east-2a). An "Actions" button is next to it. A "Remove instance group" button is also present.
- Node configuration - optional**: Expandable section.
- Add task instance group**: Button. Below it, text says: "You can add up to 48 more task instance groups."

The right column, titled "Summary", contains the following sections:

- Summary** (Info icon):
  - Name and applications - required**:
    - Name: EMR and Spark Cluster
    - Amazon EMR release: emr-6.10.0
    - Application bundle: Custom (Hadoop 3.3.3, Hive 3.1.3, Hue 4.10.0, JupyterHub 1.5.0, Livy 0.7.1, Spark 3.3.1)
  - Cluster configuration - required**:
    - Uniform instance groups
    - Primary (m5.xlarge), Core (m5.xlarge)
  - Cluster scaling and provisioning - required**:
    - Provisioning configuration
    - Core size: 1 instance
- Configure IAM roles** (Info icon): You must choose a service role and instance profile before you create this cluster. A "Choose IAM roles" button is present.
- At the bottom of the summary are "Cancel" and "Create cluster" buttons.

The footer of the console includes "CloudShell" and "Feedback" links on the left, and copyright information "© 2024, Amazon Web Services, Inc. or its affiliates." along with "Privacy", "Terms", and "Cookie preferences" links on the right.

## Big Data Wrangling With Google Books Ngrams

## Appendix E

Create cluster > EMR on EC2: C

us-east-2.console.aws.amazon.com/emr/home?region=us-east-2#/createCluster

Services Search [Alt+S]

15 - 100 GiB per volume General Purpose SSD (gp2)

### Cluster scaling and provisioning - required

Choose how Amazon EMR should size your cluster.

Choose an option

- ☒ Set cluster size manually  
Use this option if you know your workload patterns in advance.
- ☐ Use EMR-managed scaling  
Monitor key workload metrics so that EMR can optimize the cluster size and resource utilization.
- ☐ Use custom automatic scaling  
To programmatically scale core and task nodes, create custom automatic scaling policies.

### Provisioning configuration

Set the size of your core instance group. Amazon EMR attempts to provision this capacity when you launch your cluster.

Name	Instance type	Instance(s) size	Use Spot purchasing option
Core	m5.xlarge	2	<input type="checkbox"/>

### Networking - required

Choose the network settings that determine how you and other entities communicate with your cluster.

Virtual private cloud (VPC) [Info](#)

vpc-0025e415b8fd0f86a [Browse](#) [Create VPC](#)

Subnet [Info](#)

subnet-07c2a63baf5363e58 [Browse](#) [Create subnet](#)

► EC2 security groups (firewall)

### Steps (0)

Use commands and scripts to tell your cluster where to find and how to process your data. Steps run consecutively unless you enable the Concurrency option.

[Remove](#) [Edit](#) [Add](#)

### Cluster termination and node replacement

### Summary

#### Name and applications - required

Name  
EMR and Spark Cluster

Amazon EMR release  
emr-6.10.0

Application bundle  
Custom (Hadoop 3.3.3, Hive 3.1.3, Hue 4.10.0, JupyterHub 1.5.0, Livy 0.7.1, Spark 3.3.1)

#### Cluster configuration - required

Uniform instance groups  
Primary (m5.xlarge), Core (m5.xlarge)

#### Cluster scaling and provisioning - required

Provisioning configuration  
Core size: 2 instances

**Configure IAM roles**  
You must choose a service role and instance profile before you create this cluster.

[Choose IAM roles](#)

[Cancel](#) [Create cluster](#)

CloudShell Feedback

© 2024, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)



## Big Data Wrangling With Google Books Ngrams

## Appendix F

**Security configuration and EC2 key pair** [Info](#)  
Choose a security configuration or create a new one that you can reuse with other clusters.

**Security configuration**  
Select your cluster encryption, authentication, authorization, and instance metadata service settings.

**Amazon EC2 key pair for SSH to the cluster** [Info](#)

**Identity and Access Management (IAM) roles - required** [Info](#)  
Choose or create a service role and instance profile for the EC2 instances in your cluster.

**Amazon EMR service role** [Info](#)  
The service role is an IAM role that Amazon EMR assumes to provision resources and perform service-level actions with other AWS services.

☒ **Choose an existing service role**  
Select a default service role or a custom role with IAM policies attached so that your cluster can interact with other AWS services.

☐ **Create a service role**  
Let Amazon EMR create a new service role so that you can grant and restrict access to resources in other AWS services.

**Service role**

**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.

**Instance profile**

**Summary** [Info](#)

**Name and applications - required**

Name  
EMR and Spark Cluster

Amazon EMR release  
emr-6.10.0

Application bundle  
Custom (Hadoop 3.3.3, Hive 3.1.3, Hue 4.10.0, JupyterHub 1.5.0, Livy 0.7.1, Spark 3.3.1)

**Cluster configuration - required**

Uniform instance groups  
Primary (m5.xlarge), Core (m5.xlarge)

**Cluster scaling and provisioning - required**

Provisioning configuration  
Core size: 2 instances

© 2024, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

## Big Data Wrangling With Google Books Ngrams

## Appendix G

The screenshot displays the Amazon EMR console interface for a cluster named "EMR and Spark Cluster". The breadcrumb navigation shows the path: Amazon EMR > EMR on EC2: Clusters > EMR and Spark Cluster. The cluster ID is j-2VX93C9VJ4PF0. The status is "Waiting", and it was created on March 18, 2024, at 13:17 UTC-04:00. The cluster has 1 Primary node, 2 Core nodes, and 0 Task nodes. The console is organized into several sections: Summary, Properties, Bootstrap actions, Instances (Hardware), Steps, Applications, Configurations, Monitoring, Events, and Tags (0). The Summary section includes Cluster info, Applications, Cluster management, and Status and time. The Properties section includes Operating system, Cluster logs, and Cluster termination and node replacement. The Network and security section includes Network, Security configuration, and Permissions. The footer shows the AWS logo, CloudShell, Feedback, and copyright information for Amazon Web Services, Inc. or its affiliates.

Properties > EMR and Spark Cl... x EMR on EC2: Security configur... x +

us-east-2.console.aws.amazon.com/emr/home?region=us-east-2#/clusterDetails/j-2VX93C9VJ4PF0

aws Services Search [Alt+S]

Amazon EMR > EMR on EC2: Clusters > EMR and Spark Cluster

### EMR and Spark Cluster

Updated less than a minute ago [Refresh](#) [Terminate](#) [Clone in AWS CLI](#) [Clone](#)

#### ▼ Summary

Cluster info	Applications	Cluster management	Status and time
<p>Cluster ID</p> <p>j-2VX93C9VJ4PF0</p> <p>Cluster configuration</p> <p>Instance groups</p> <p>Capacity</p> <p>1 Primary 2 Core 0 Task</p>	<p>Amazon EMR version</p> <p>emr-6.10.0</p> <p>Installed applications</p> <p>Hadoop 3.3.3, Hive 3.1.3, Hue 4.10.0, JupyterHub 1.5.0, Livy 0.7.1, Spark 3.3.1</p>	<p>Log destination in Amazon S3</p> <p><a href="#">aws-logs-975050206953-us-east-2/elasticmapreduce</a></p> <p>Persistent application UIs</p> <p><a href="#">Spark History Server</a></p> <p><a href="#">YARN timeline server</a></p> <p><a href="#">Tez UI</a></p> <p>Primary node public DNS</p> <p><a href="#">ec2-3-147-13-110.us-east-2.compute.amazonaws.com</a></p> <p><a href="#">Connect to the Primary node using SSH</a></p> <p><a href="#">Connect to the Primary node using SSM</a></p>	<p>Status</p> <p><span>Waiting</span></p> <p>Creation time</p> <p>March 18, 2024, 13:17 (UTC-04:00)</p> <p>Elapsed time</p> <p>12 minutes, 28 seconds</p>

#### Properties

Bootstrap actions | Instances (Hardware) | Steps | Applications | Configurations | Monitoring | Events | Tags (0)

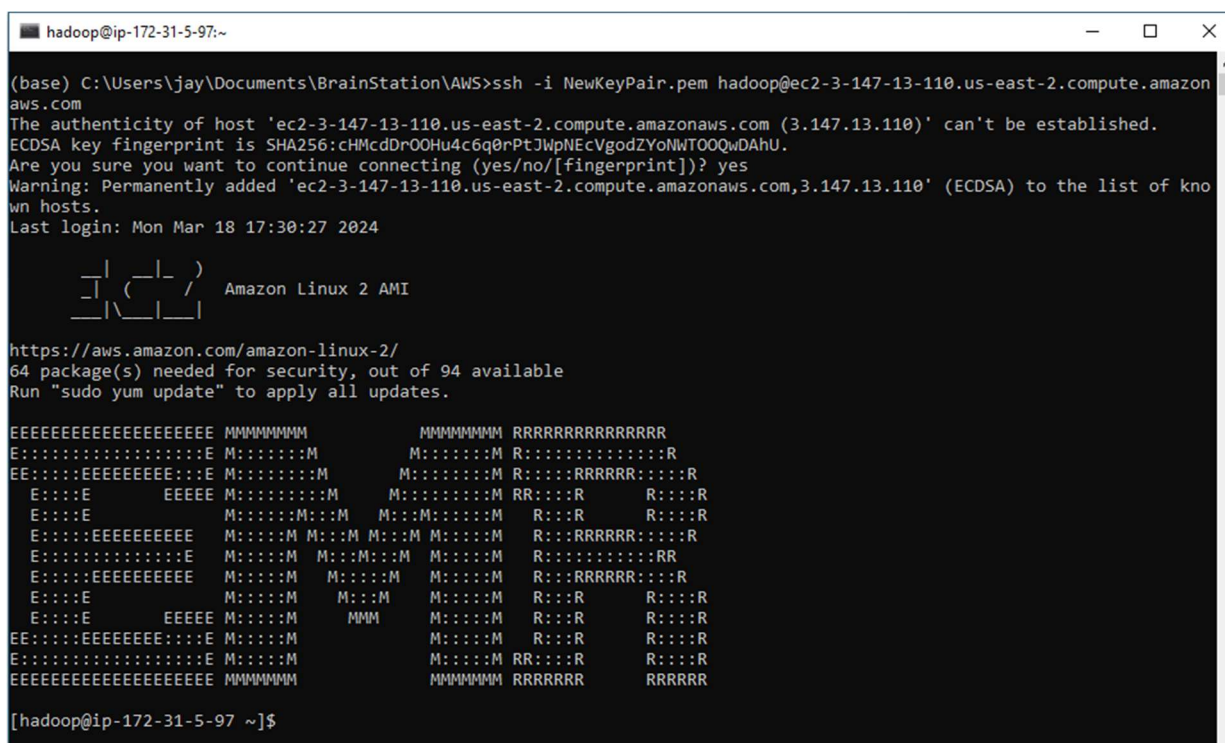
Operating system <a href="#">Info</a>	Cluster logs <a href="#">Info</a>	Cluster termination and node replacement <a href="#">Info</a> <a href="#">Edit</a>
<p>Amazon Linux release</p> <p>2.0.20230808.0</p>	<p>Archive log files to Amazon S3</p> <p>Turned on</p> <p>Amazon S3 location</p> <p><a href="#">s3://aws-logs-975050206953-us-east-2/elasticmapreduce/</a></p> <p>Encryption for logs</p> <p>Turned off</p>	<p>Termination option</p> <p>Automatically terminate cluster after idle time</p> <p>Termination protection</p> <p>Off</p> <p>Idle time</p> <p>1 hour</p> <p>Unhealthy node replacement</p> <p>On</p>

#### Network and security [Info](#)

Network	Security configuration	Permissions
---------	------------------------	-------------

CloudShell Feedback

© 2024, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)



## Big Data Wrangling With Google Books Ngrams

## Appendix I

```

hadoop@ip-172-31-5-97:~
(base) C:\Users\jay\Documents\BrainStation\AWS>ssh -i NewKeyPair.pem hadoop@ec2-3-147-13-110.us-east-2.compute.amazonaws.com
Last login: Mon Mar 18 18:38:25 2024

 _ | ( _ | )
 _|/  _|/  Amazon Linux 2 AMI

https://aws.amazon.com/amazon-linux-2/
64 package(s) needed for security, out of 94 available
Run "sudo yum update" to apply all updates.

EEEEEEEEEEEEEEEEEEEE MMMMMMMM MMMMMMMM RRRRRRRRRRRRRRRR
E:EEEEEEEEEEEEEEEEEEEE M:EEEEEEEE M:EEEEEEEE R:EEEEEEEEEEEE
EE:EEEEEEEEEEEEEEEEEEEE M:EEEEEEEE M:EEEEEEEE R:RRRRRRRRRRR
E:EE:EEEEEEEEEEEEEEEE M:EEEEEEEE M:EEEEEEEE RR:RR R:RR
E:EE:EEEEEEEEEEEEEEEE M:EEEEEEEE M:EEEEEEEE R:RR R:RR
E:EEEEEEEEEEEEEEEEEEEE M:EEEEEEEE M:EEEEEEEE R:RRRRRRRRRR
E:EEEEEEEEEEEEEEEEEEEE M:EEEEEEEE M:EEEEEEEE R:RRRRRRRRRR
E:EE:EEEEEEEEEEEEEEEE M:EEEEEEEE M:EEEEEEEE R:RR R:RR
E:EE:EEEEEEEEEEEEEEEE M:EEEEEEEE M:EEEEEEEE R:RR R:RR
EE:EEEEEEEEEEEEEEEEEEEE M:EEEEEEEE M:EEEEEEEE R:RR R:RR
E:EEEEEEEEEEEEEEEEEEEE M:EEEEEEEE M:EEEEEEEE R:RR R:RR
EEEEEEEEEEEEEEEEEEEEEEEE MMMMMMMM MMMMMMMM RRRRRRRRRRRR

[hadoop@ip-172-31-5-97 ~]$ hadoop distcp s3://brainstation-dsft/eng_1M_1gram.csv /user/hadoop/eng_1M_1gram/eng_1M_1gram.csv
2024-03-18 18:42:24,529 INFO tools.DistCp: Input Options: DistCpOptions{atomicCommit=false, syncFolder=false, deleteMissing=false, ignoreFailures=false, overwrite=false, append=false, useDiff=false, useRdiff=false, fromSnapshot=null, toSnapshot=null, skipCRC=false, blocking=true, numListStatusThreads=0, maxMaps=20, mapBandwidth=0.0, copyStrategy='uniformsize', preserveStatus=[], atomicWorkPath=null, logPath=null, sourceFileListing=null, sourcePaths=[s3://brainstation-dsft/eng_1M_1gram.csv], targetPath=/user/hadoop/eng_1M_1gram/eng_1M_1gram.csv, filtersFile='null', blocksPerChunk=0, copyBufferSize=8192, verboseLog=false, directWrite=false, useIterator=false}, sourcePaths=[s3://brainstation-dsft/eng_1M_1gram.csv], targetPathExists=false, preserveRawXattrs=false
2024-03-18 18:42:24,775 INFO client.DefaultNoHARMFaloverProxyProvider: Connecting to ResourceManager at ip-172-31-5-97.us-east-2.compute.internal/172.31.5.97:8032
2024-03-18 18:42:24,919 INFO client.AHSPProxy: Connecting to Application History server at ip-172-31-5-97.us-east-2.compute.internal/172.31.5.97:10200
2024-03-18 18:42:27,640 INFO tools.SimpleCopyListing: Starting: Building listing using multi threaded approach for s3://brainstation-dsft/eng_1M_1gram.csv
2024-03-18 18:42:27,642 INFO tools.SimpleCopyListing: Building listing using multi threaded approach for s3://brainstation-dsft/eng_1M_1gram.csv: duration 0:00.002s
2024-03-18 18:42:27,773 INFO tools.SimpleCopyListing: Paths (files+dirs) cnt = 1; dirCnt = 0
2024-03-18 18:42:27,773 INFO tools.SimpleCopyListing: Build file listing completed.
2024-03-18 18:42:27,774 INFO Configuration.deprecation: io.sort.mb is deprecated. Instead, use mapreduce.task.io.sort.mb
2024-03-18 18:42:27,775 INFO Configuration.deprecation: io.sort.factor is deprecated. Instead, use mapreduce.task.io.sort.factor
2024-03-18 18:42:27,886 INFO tools.DistCp: Number of paths in the copy list: 1
2024-03-18 18:42:27,911 INFO tools.DistCp: Number of paths in the copy list: 1
2024-03-18 18:42:27,936 INFO client.DefaultNoHARMFaloverProxyProvider: Connecting to ResourceManager at ip-172-31-5-97.us-east-2.compute.internal/172.31.5.97:8032
2024-03-18 18:42:27,936 INFO client.AHSPProxy: Connecting to Application History server at ip-172-31-5-97.us-east-2.compute.internal/172.31.5.97:10200
2024-03-18 18:42:28,022 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/hadoop/.staging/job_1710782882902_0010
2024-03-18 18:42:28,132 INFO mapreduce.JobSubmitter: number of splits:1
2024-03-18 18:42:28,292 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1710782882902_0010
2024-03-18 18:42:28,292 INFO mapreduce.JobSubmitter: Executing with tokens: []
2024-03-18 18:42:28,468 INFO conf.Configuration: resource-types.xml not found
2024-03-18 18:42:28,468 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.

```



## Big Data Wrangling With Google Books Ngrams

## Appendix J

```

hadoop@ip-172-31-5-97:~
2024-03-18 18:42:28,528 INFO impl.YarnClientImpl: Submitted application application_1710782882902_0010
2024-03-18 18:42:28,575 INFO mapreduce.Job: The url to track the job: http://ip-172-31-5-97.us-east-2.compute.interna
l:20888/proxy/application_1710782882902_0010/
2024-03-18 18:42:28,575 INFO tools.DistCp: DistCp job-id: job_1710782882902_0010
2024-03-18 18:42:28,576 INFO mapreduce.Job: Running job: job_1710782882902_0010
2024-03-18 18:42:34,632 INFO mapreduce.Job: Job job_1710782882902_0010 running in uber mode : false
2024-03-18 18:42:34,633 INFO mapreduce.Job: map 0% reduce 0%
2024-03-18 18:42:50,713 INFO mapreduce.Job: map 100% reduce 0%
2024-03-18 18:43:37,893 INFO mapreduce.Job: Job job_1710782882902_0010 completed successfully
2024-03-18 18:43:37,977 INFO mapreduce.Job: Counters: 42
  File System Counters
    FILE: Number of bytes read=0
    FILE: Number of bytes written=294915
    FILE: Number of read operations=0
    FILE: Number of large read operations=0
    FILE: Number of write operations=0
    HDFS: Number of bytes read=376
    HDFS: Number of bytes written=5292105197
    HDFS: Number of read operations=12
    HDFS: Number of large read operations=0
    HDFS: Number of write operations=5
    HDFS: Number of bytes read erasure-coded=0
    S3: Number of bytes read=5292105197
    S3: Number of bytes written=0
    S3: Number of read operations=0
    S3: Number of large read operations=0
    S3: Number of write operations=0
  Job Counters
    Launched map tasks=1
    Other local map tasks=1
    Total time spent by all maps in occupied slots (ms)=5858592
    Total time spent by all reduces in occupied slots (ms)=0
    Total time spent by all map tasks (ms)=61027
    Total vcore-milliseconds taken by all map tasks=61027
    Total megabyte-milliseconds taken by all map tasks=187474944
  Map-Reduce Framework
    Map input records=1
    Map output records=0
    Input split bytes=137
    Spilled Records=0
    Failed Shuffles=0
    Merged Map outputs=0
    GC time elapsed (ms)=216
    CPU time spent (ms)=59680
    Physical memory (bytes) snapshot=1036165120
    Virtual memory (bytes) snapshot=4434186240
    Total committed heap usage (bytes)=617611264
    Peak Map Physical memory (bytes)=1068023808
    Peak Map Virtual memory (bytes)=4434186240
  File Input Format Counters
    Bytes Read=239
  File Output Format Counters
    Bytes Written=0
  DistCp Counters
    Bandwidth in Bytes=91243193
    Bytes Copied=5292105197
    Bytes Expected=5292105197
    Files Copied=1
[hadoop@ip-172-31-5-97 ~]$

```

```
(base) C:\Users\jay\Documents\BrainStation\AWS>ssh -i NewKeyPair.pem -L 9995:localhost:9443 hadoop@ec2-3-147-13-110.us-east-2.compute.amazonaws.com
Last login: Mon Mar 18 17:32:48 2024 from d24-150-232-237.home.cgocable.net

 _| ( _| )
 _| \ _| _|   Amazon Linux 2 AMI

https://aws.amazon.com/amazon-linux-2/
64 package(s) needed for security, out of 94 available
Run "sudo yum update" to apply all updates.
Last login: Mon Mar 18 17:32:48 2024 from d24-150-232-237.home.cgocable.net

 _| ( _| )
 _| \ _| _|   Amazon Linux 2 AMI

https://aws.amazon.com/amazon-linux-2/
64 package(s) needed for security, out of 94 available
Run "sudo yum update" to apply all updates.

EEEEEEEEEEEEEEEEEEEE MMMMMMMM MMMMMMMM RRRRRRRRRRRRRR
E::::::::::::::::::::E M::::::::M M::::::::M R::::::::::::R
EE::::::::EEEEEEEE::::E M::::::::M M::::::::M R::::RRRRR::::R
E::::E EEEEE M::::::::M M::::::::M RR::::R R::::R
E::::E M::::::::M M::::::::M M::::::::M R::::R R::::R
E::::EEEEEEEEEE M::::::::M M::::::::M M::::::::M R::::RRRRR::::R
E::::::::::::E M::::::::M M::::::::M M::::::::M R::::::::::::RR
E::::EEEEEEEEEE M::::::::M M::::::::M M::::::::M R::::RRRRR::::R
E::::E M::::::::M M::::::::M M::::::::M R::::R R::::R
E::::E EEEEE M::::::::M MMM M::::::::M R::::R R::::R
EE::::::::EEEEEEEE::::E M::::::::M M::::::::M R::::R R::::R
E::::::::::::E M::::::::M M::::::::M RR::::R R::::R
EEEEEEEEEEEEEEEEEEEE MMMMMMMM MMMMMMMM RRRRRRR RRRRRR

[hadoop@ip-172-31-5-97 ~]$
```

# Big Data Wrangling With Google Books Ngrams

## Appendix L

The screenshot shows a Jupyter Notebook titled "Big Data Wrangling With Google Books Ngrams" running on a JupyterHub instance. The browser address bar shows a local URL: `https://localhost:9995/user/jovyan/notebooks/Big%20Data%20Wrangling%20With%20Google%20Books%20Ngrams.ipynb`. The notebook interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations, running cells, and code execution. The notebook content consists of several code cells:

```
In [1]: spark

Starting Spark application

ID      YARN Application ID  Kind  State  Spark UI  Driver log  User  Current session?
12  application_1710782882902_0016  pyspark  idle  Link  Link  None  ✓

SparkSession available as 'spark'.
<pyspark.sql.session.SparkSession object at 0x7f940bc47c50>

4. Read in csv file from Hadoop into a pySpark dataframe.

In [2]: hdfs_path = "hdfs://ec2-3-147-13-110.us-east-2.compute.amazonaws.com/user/hadoop/eng_1M_1gram/eng_1M_1gram.csv"

In [3]: # Read the CSV file into a data file
df = spark.read.csv(hdfs_path, header=True)

In [4]: df

DataFrame[token: string, year: string, frequency: string, pages: string, books: string]

In [5]: # Rows
df.count()

261823225

4a.

Shape is (261,823,225, 5)

In [6]: df.columns

['token', 'year', 'frequency', 'pages', 'books']

4a. Schema

In [7]: df.printSchema()

root
```

# Big Data Wrangling With Google Books Ngrams

## Appendix M

The screenshot shows a Jupyter Notebook interface with the following content:

```
-- token: string (nullable = true)
-- year: string (nullable = true)
-- frequency: string (nullable = true)
-- pages: string (nullable = true)
-- books: string (nullable = true)
```

**4b. Make a new dataframe from a sql query with column token containing the word 'data'.**

```
In [8]: # Register dataframe as a view
df.createOrReplaceTempView("eng_table")

# Run sql Query
newdf = spark.sql("SELECT * FROM eng_table WHERE token LIKE '%data%'")
```

**4b. Describe the new dataset.**

```
In [9]: newdf.count()
24642
```

```
In [10]: newdf.columns
['token', 'year', 'frequency', 'pages', 'books']
```

```
In [11]: newdf.head(5)
[Row(token='laticaudata', year='1800', frequency='1', pages='1', books='1'), Row(token='laticaudata', year='1823', frequency='2', pages='2', books='2'), Row(token='laticaudata', year='1827', frequency='1', pages='1', books='1'), Row(token='laticaudata', year='1843', frequency='2', pages='2', books='1'), Row(token='laticaudata', year='1844', frequency='6', pages='6', books='4')]
```

```
In [12]: newdf.printSchema()
root
|-- token: string (nullable = true)
|-- year: string (nullable = true)
|-- frequency: string (nullable = true)
|-- pages: string (nullable = true)
|-- books: string (nullable = true)
```

**4c. Write new dataframe to HDFS**

```
In [13]: hdfs_write_path = "hdfs://ec2-3-147-13-110.us-east-2.compute.amazonaws.com/user/hadoop/eng_1M_igram/eng_token_data.csv"

In [14]: newdf.write.csv(hdfs_write_path, header=True)

In [15]: spark.stop()
```



## Big Data Wrangling With Google Books Ngrams

## Appendix N

The screenshot displays the JupyterHub web interface in a browser window. The address bar shows the URL `https://localhost:9995/user/jovyan/tree?`. The page header includes the JupyterHub logo, a "Logout" button, and a "Control Panel" button. Below the header, there are tabs for "Files", "Running", and "Clusters". The "Files" tab is active, showing a file browser view. At the top of the file browser, there is a prompt "Select items to perform actions on them." and buttons for "Upload", "New", and a refresh icon. A table lists the files in the current directory:

	Name	Last Modified	File size
<input type="checkbox"/>	/		
<input type="checkbox"/>	jupyterhub-proxy.pid	13 minutes ago	2 B
<input type="checkbox"/>	jupyterhub.sqlite	seconds ago	102 kB
<input type="checkbox"/>	jupyterhub_cookie_secret	13 minutes ago	65 B