**Project 3: Batch Data Processing Pipeline**

**CS4981 ML Production Systems**

**Instructions**

**Part I: Set up Minio:**

Download and set up the Minio client and server: https://min.io/download

Start Server:

$ ./minio server minio\_data

where minio\_data is a folder you created.

Log into the web interface at http://127.0.0.1:9000 with the default credentials (username: minioadmin, password: minioadmin ). Create a bucket named log-files. Create a user named log-depositor with the read-write policy.

**Part II: Write Log Collection Script**

Implement a script that will run continuously in the background to read the latest entries in the log file and upload the lines as a file to Minio. You can use the [pygtail library](https://pypi.org/project/pygtail/) to emulate the “tail -F” behavior. Your script should run the log collection on a user-specified period (e.g., every 15 min) and sleep until the next run (make sure you set the sleep to sleep the remaining time, so it runs consistently at the provided periods.

You can use the [boto3 library](https://boto3.amazonaws.com/v1/documentation/api/latest/index.html) or [minio library](https://min.io/docs/minio/linux/developers/python/minio-py.html) to upload objects to Minio. You will need to determine a naming scheme for the log file chunks (e.g., log\_file\_YYMMDD-HHMMSS.json). Note: The boto3.client() function accepts several optional keyword parameters for specifying the login details:

aws\_access\_key\_id: The username for Minio

aws\_secret\_access\_key: The password for Minio

endpoint\_url: The URL for Minio (e.g., http://localhost:9000 )

Your script should take the username, password, endpoint URL, bucket and base path, and local log file path as environmental variables. The period can be passed as a command-line argument.

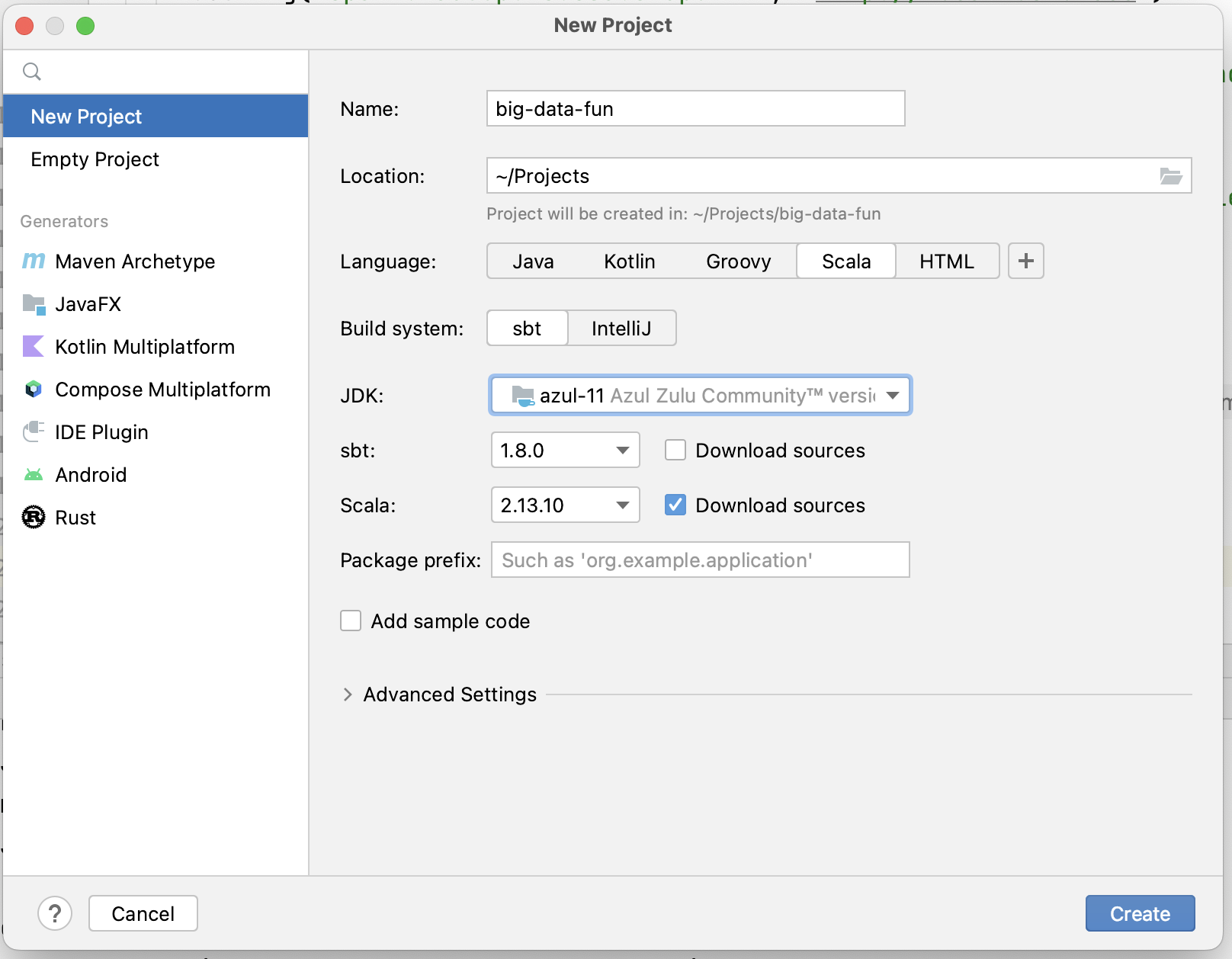
**Part III: Setup a Project for Apache Spark in Intellij**

1. Install the [Scala plugin for IntelliJ](https://www.jetbrains.com/help/idea/discover-intellij-idea-for-scala.html#scala_plugin).

Graphical user interface, text, application, email

Description automatically generated

2. Create a new project. Set the language to Scala, the build system to sbt, and use a version 11 JDK.



3. Once the project is created, modify the build.sbt file to add the following lines at the end:

libraryDependencies += "org.apache.spark" %% "spark-core" % "3.3.1"

libraryDependencies += "org.apache.spark" %% "spark-sql" % "3.3.1"

// for postgres

libraryDependencies += "org.postgresql" % "postgresql" % "42.5.1"

// for minio

libraryDependencies += "org.apache.hadoop" % "hadoop-common" % "3.3.2"

libraryDependencies += "org.apache.hadoop" % "hadoop-client" % "3.3.2"

libraryDependencies += "org.apache.hadoop" % "hadoop-aws" % "3.3.2"

and reload the sbt configuration (there should be a help item pop up in the file editor once you save it).

Text

Description automatically generated

4. Create a Scala object with a main method for your application. You can use these examples as a starting point:

<https://github.com/rnowling/spark-examples>

**Part IV: Implement a Batch Data Processing Pipeline**

The Postgres database contains the received emails, while logs of user actions are stored in the Minio object store. You need to implement a batch processing pipeline that will:

1. Load the logs from Minio
2. Select the events in which emails are classified as spam
3. Retrieve the emails from the Postgres database
4. Join the two data sets to determine which emails are spam or not
5. Store the labeled emails in the object store

**Accessing Postgres**

1. Set up your connection properties (import the Properties class from java.util)

// set the database name as appropriate

val url = "jdbc:postgresql://127.0.0.1:5432/spam\_classification"

val tableName = "labeled\_emails"

val props = new Properties()

// set the following as appropriate

props.setProperty("user", "username")

props.setProperty("password", "password")

2. Use jdbc() read method:

val df = spark.read

.jdbc(url, tableName, props)

**Accessing Files in Minio**

1. Set the following configuration options after creating your SparkSession:

// Replace Key with your minio username

spark.sparkContext

.hadoopConfiguration.set("fs.s3a.access.key", "minioadmin")

// Set your password

spark.sparkContext

.hadoopConfiguration.set("fs.s3a.secret.key", "minioadmin")

// Set your server path

spark.sparkContext

.hadoopConfiguration.set("fs.s3a.endpoint", "http://127.0.0.1:9000")

// Disable SSL

spark.sparkContext

.hadoopConfiguration.set("fs.s3a.connection.ssl.enabled", "false")

2. Now, you can use the usual DataFrame read and write methods for CSV, JSON, etc. Prepend "s3a://" to the start of the file paths:

spark.read.json("s3a://bucket/path/to/file.json")

**Part IV: Run the Pipeline to Process the Data**

Run the simulator to completion to fill your pipeline to process your data and prepare it for developing your machine learning model.

1. Download the email\_json\_dataset1.zip and spam\_simulator.zip files from the shard Box folder and unzip it:

<https://msoe.box.com/s/l9xl4udcwvqzrzwtxwrqw4bejehomk2b>

2. Create a second virtual environment named simulator. Install the updated versions of pip and wheel. Install the simulator's dependencies in your Python environment:

(simulator) $ python install -r requirements

3. Make sure that you have Postgres, Minio, the two REST services, and the log collection service running (set to a 1 minute collection period).

4. Now, run the simulator:

(simulator) $ python spam\_simulator.py simulate-user --email-dir path/to/email\_data --email-url http://127.0.0.1:8888/ --mailbox-url http://127.0.0.1:8889/ --number-emails 100000 --average-events-per-email 5

5. Check that Postgres has entries, the local log file has entries, and that Minio has appropriate log files in it.

6. Run your Spark pipeline. Check the output in Minio.

**Submission Instructions**

Store your code in a private GitHub repository and give read access to the user "rnowling". Make sure to include your group members names in the README.md file at the root of your repository. Submit the URL to Canvas.

**Troubleshooting**

**Error about too many command-line arguments**

You may receive an error indicating that the maximum length of command-line arguments has been exceeded when you try to run your Spark program. Take the following steps to shorten the command-line arguments:

1. Select "Edit Configurations..." from the Run menu

Graphical user interface, text, application, email

Description automatically generated

2. Select the "Modify Options" menu

Graphical user interface, text, application, email

Description automatically generated

3. Select the "Shorten command line" option

Graphical user interface, text, application, email

Description automatically generated

4. One the Run Configuration dialog, select the "@argfile (Java 9+)" option

Graphical user interface, text, application, email

Description automatically generated

5. Click "Apply" and "OK"

**UnsatifisiedLinkError on Windows**

You may receive an UnsatisfiedLinkError that prevents you from running your Spark application on Windows. To resolve this issue:

1. Download winutils.exe and hadoop.dll from here:

<https://github.com/cdarlint/winutils/tree/master/hadoop-3.2.2/bin>

2. Create the directory C:\hadoop\bin and place the files there.

3. Update your environmental PATH variable with the following:

HADOOP\_HOME=C:\hadoop

PATH=%PATH%;C:\hadoop\bin