Managing the Cloud Project: Analyzing Crime Data Using Cloud Services



Arizona State University

Department of Information Technology

Alok Sahu

Ramya Chandrashekar

Venkata Siva Abhishek Munukutla

Mohan Suraj Lakshmi Narayana

Supervisor: Thomas Pearson

Abstract

Controlling crime is a challenging task. It is hard to predict the areas need increased security and the type of attacks that might take place. Historic crime data can be helpful in getting a better picture of the frequency and nature of crime in an area. In this project, we use historic crime data of Los Angeles and analyze them to build reports. These reports use attributes like, nature of crime, area, weapon used and time to build comprehensive dashboards which can be used to manage resources and provide better security. We ran map reduce on the data for a set of use cases on Amazon EMR cloud service and generated dashboards using Tableau Desktop. This document describes the different steps involved in building the project.

Managing the Cloud Project: Analyzing Crime Data Using Cloud Services

## 1. Downloading a large amount of data

1.1. We chose the dataset “Crime Data from 2010 to Present” from the website data.gov. It reflects incidents of crime in the City of Los Angeles dating back to 2010. The dataset contains columns that describe the crime and its nature and details about the individuals involved in these crimes. The size of the file is about 260 Mb. We plan to analyze the data to help us extract information about the following:

* Number of crimes based on location and time of the day
* Number of victims of crimes based on age range and sex
* Number of crimes based on the type of premise
* Number of crimes committed based on weapons used
* Number of crimes based on nature of crime and location

Based on the analysis, we can detect the pattern of crimes across Los Angeles and give few workable suggestions to improve the prevailing order of the city.

1.2. To download the dataset, visit the website [www.data.gov](http://www.data.gov) and search for the dataset “Crime data 2010”.



Figure View of the Data.gov website

Then, click on the link to the dataset and download it in .CSV format from its page.

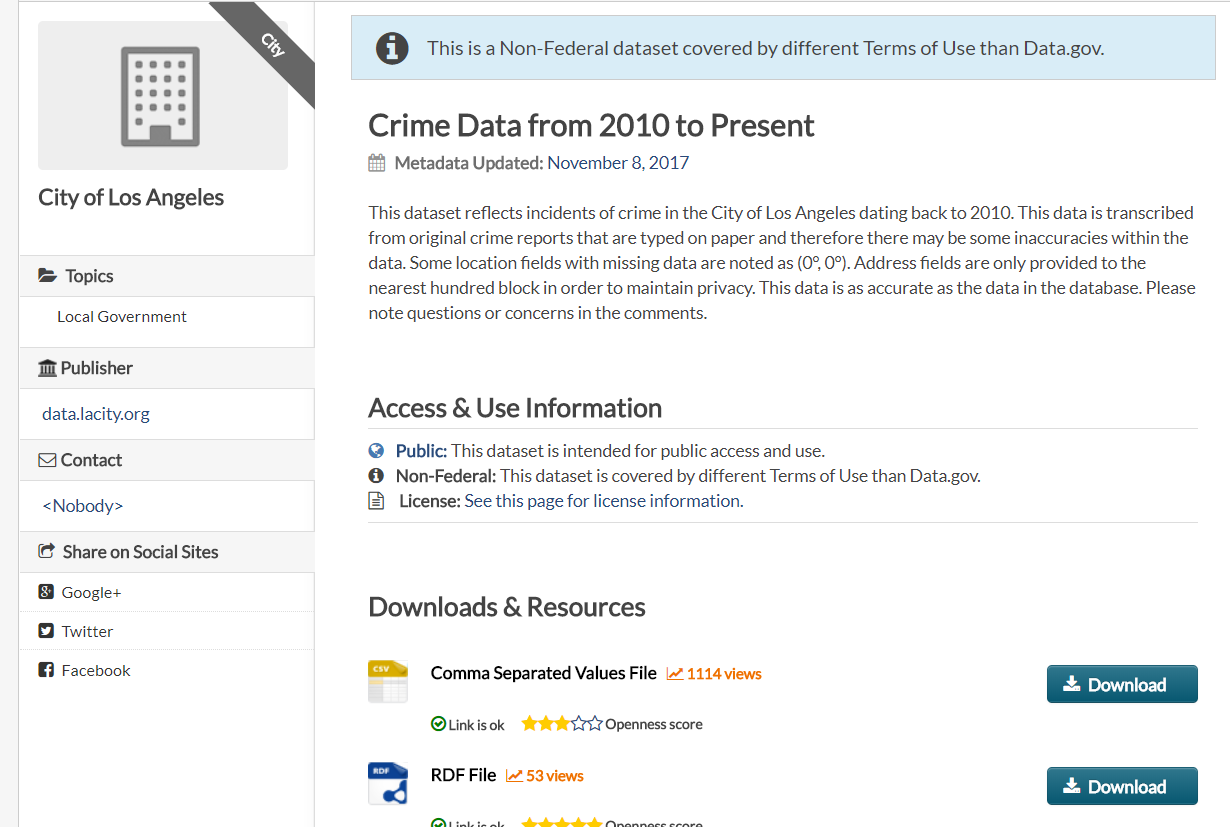


Figure View of the page on" Crime Data from 2010 to Present" dataset

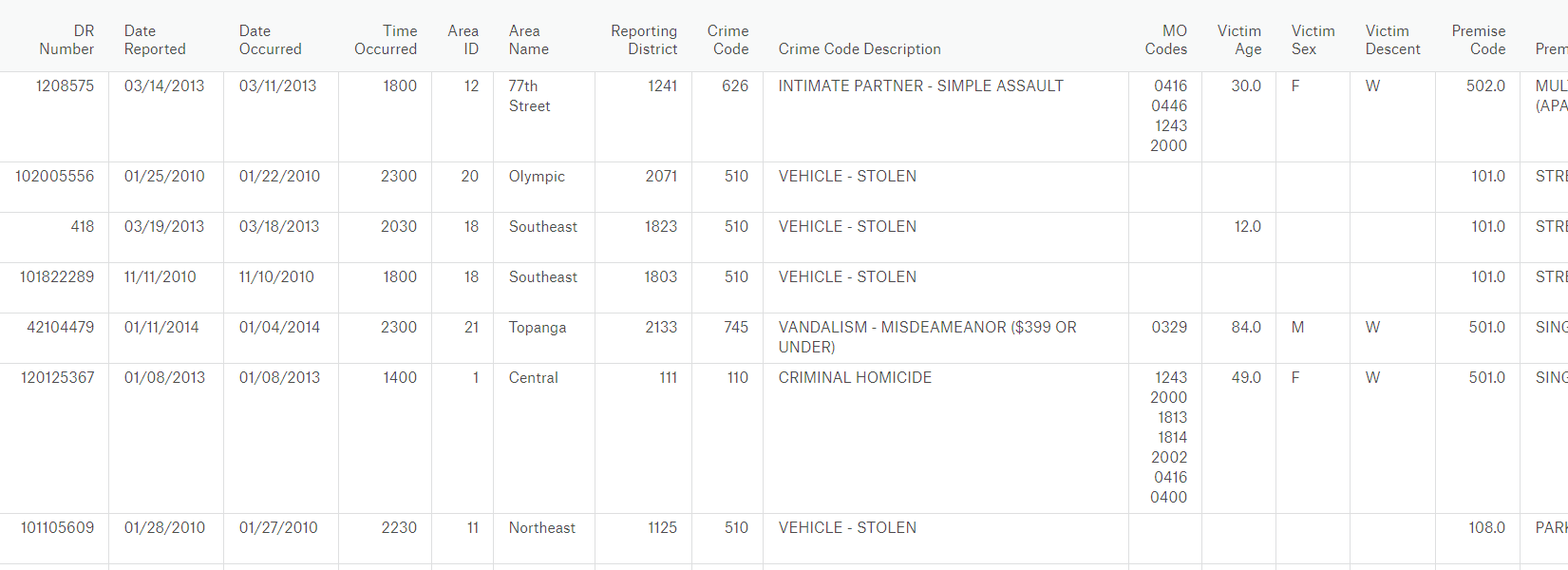


Figure View of the first few records in the dataset

## 2. Storing the data on the cloud

2.1. To store the data, we used Amazon S3. We uploaded the .txt version of the dataset to an S3 bucket and used the built-in features of Amazon EMR to load the data onto our cluster.

* By using Amazon S3 as a data store we can separate our cluster’s storage and compute nodes. This enables us to save costs by sizing our cluster for compute requirements instead of paying to store our entire dataset with 3x replication in the on-cluster Hadoop Distributed File System (HDFS).

2.2. To store the dataset in the cloud, i.e., Amazon S3, follow the steps mentioned below:

* Go to aws.amazon.com and log into your account
* In the ***Services*** tab, click on S3 available under *Storage* menu
* In the S3 Console, click on ***Create bucket***

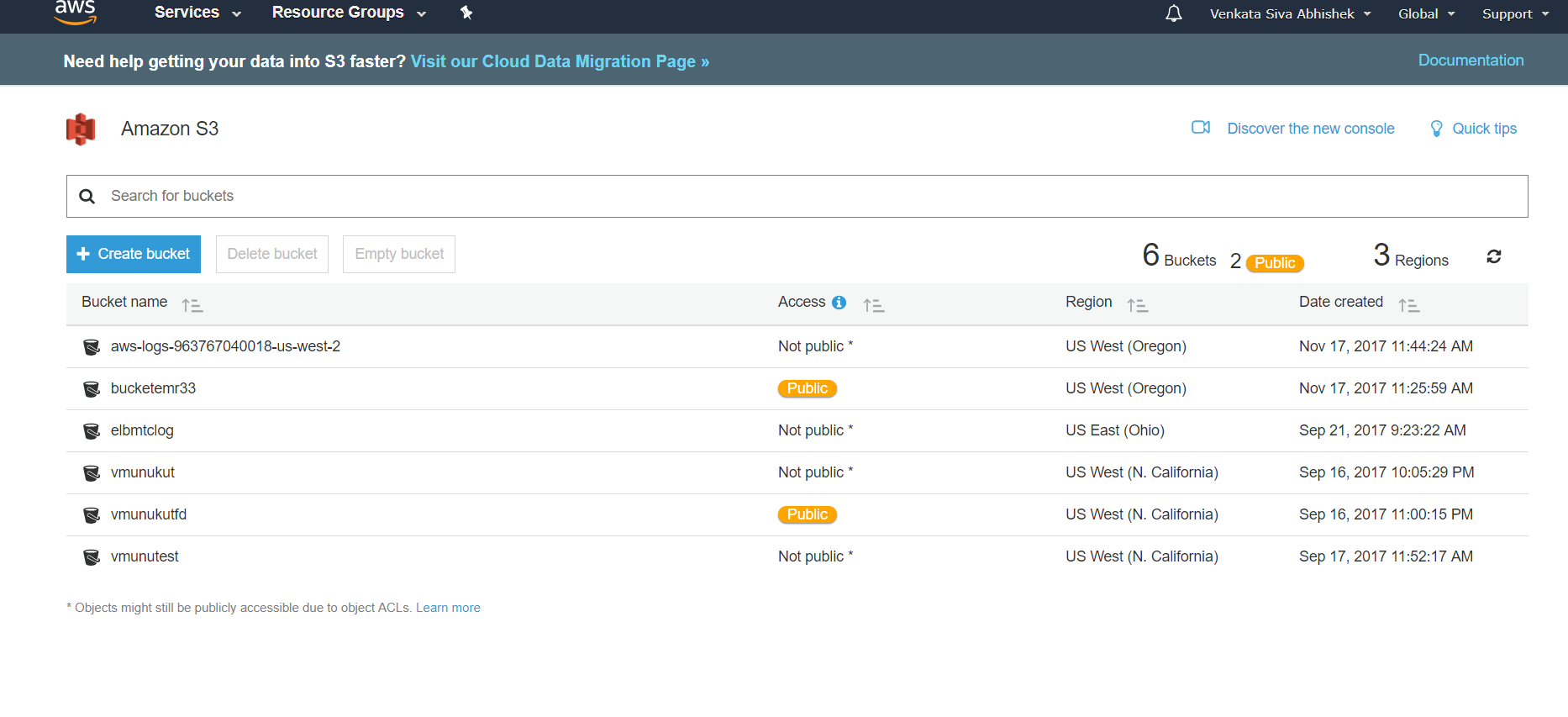


Figure S3 Dashboard

* Name your S3 bucket and select a region of your choice
* In the ***set permission*** tab, select *Grant public read access to this bucket* option under Manage Public Permissions to enable others to read the output files or set a bucket policy to give access to specific AWS users.

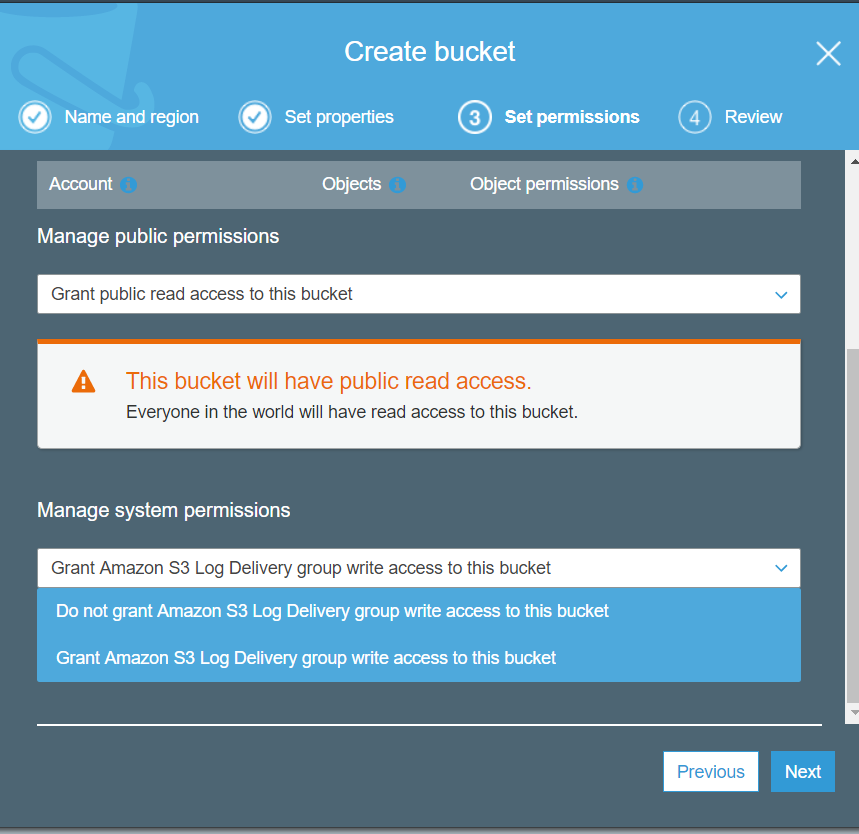


Figure Setting Permissions for S3 bucket

* Click on next and then create. Find your bucket on your S3 Console.
* Click on the bucket. Click on ***Create Folder*** to create folder structures for input, output and job files.

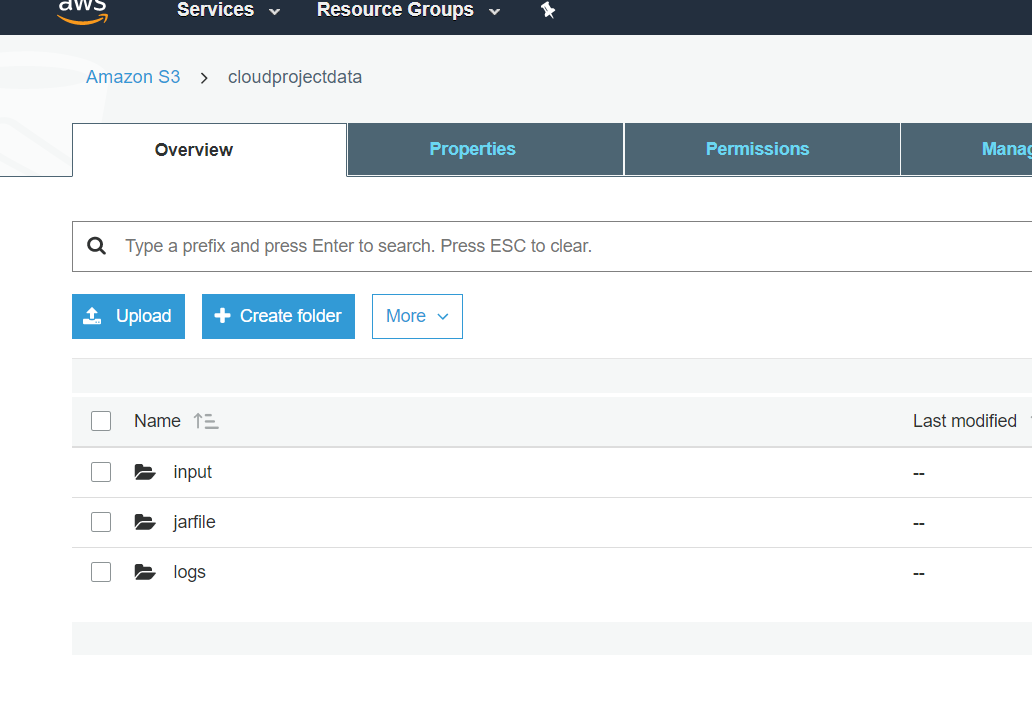


Figure S3 Bucket Folder Structure

* To upload, go to the specific folder you want to upload the file into and click on ***Upload***. Select ***Add Files*** and select your dataset or JAR file on your PC.

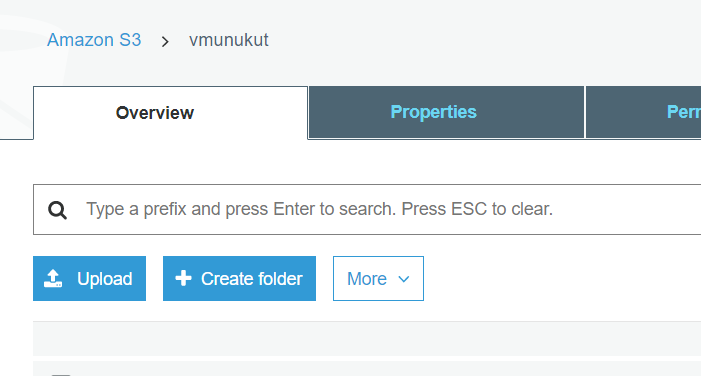


Figure Upload option in S3

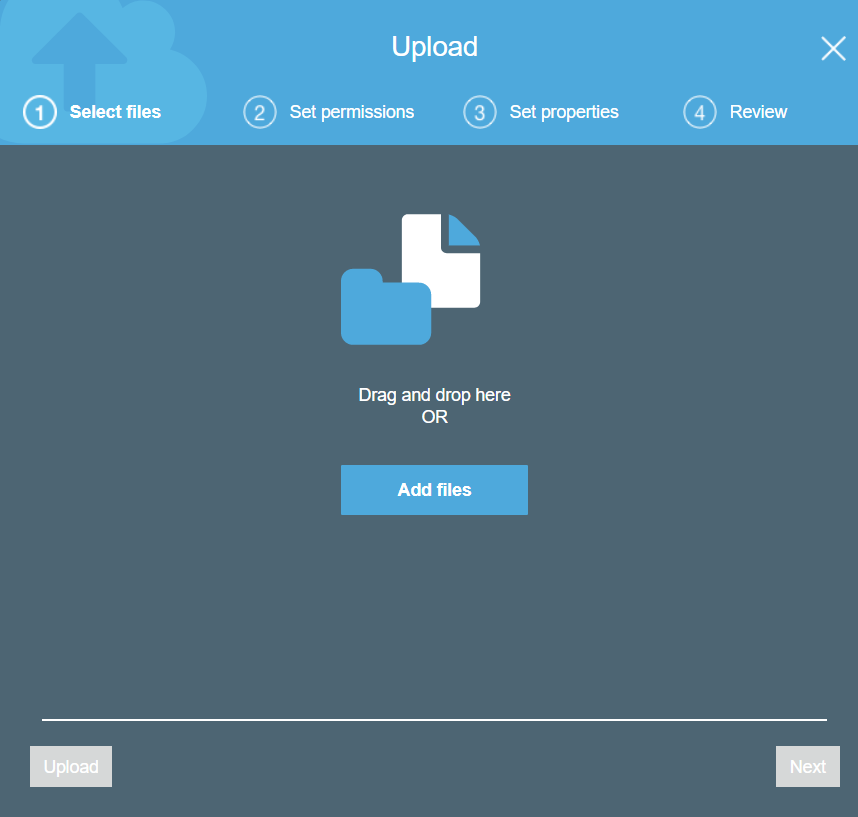


Figure Upload screen in S3

* Click on ***Upload.*** Wait for your file to upload. Find the uploaded file in the folder once the upload is successful.
* The input file, job file and the location for the output files are successfully created.

## 3. Building a distributed data processing system in the cloud

3.1 To analyze data, we used Elastic Map Reduce on Amazon Web Service. Amazon EMR uses Apache Hadoop, a Java-based programming framework, to distribute data and processing across a resizable cluster of Amazon EC2 instances. Amazon EMR can process unstructured data and is typically used for processing Big Data quickly and cost-effectively.

We chose Amazon EMR because of the following **reasons**:

* Auto-scaling Cluster – EMR segregates slave nodes into Master Nodes and Core Nodes. The Master Node acts both as data node and worker node and the Core Node only acts as worker node. This type of segregation allows the users to remove task nodes (Scale-in) without losing HDFS data and making usage cost-effective.
* The Cluster can be automatically terminated after successful completion of the jobs to improve utilization and reduce costs. EMR clusters can be terminated but not shut down.
* AWS EMR monitors the slave nodes continuously and replaces unhealthy nodes with new ones.
* EMR is available on AWS as Software as a Service(SaaS).

3.2 Amazon EMR is a paid service and does not qualify for the free tier. But students and trainers can request for an Educate account which provides them with a credit of 100$. This credit would be sufficient to deploy an academic project on EMR and learn about using the service.

* Goto the link : <https://aws.amazon.com/education/awseducate/>
* Click on “Join AWS Educate Today” button.
* Click on student which will take you to a web form that requires the details of the student.
* We are required to use the university email id while registering.
* Click on submit. The request is submitted and will be reviewed by the AWS team.
* On successful approval, an email mentioning the details of the educate account along with the 100$ credits coupon will be sent to the university id.

3.3 Steps to setup Amazon EMR:

* Go to aws.amazon.com and log into your account
* In the ***Services*** tab, click on *EMR* available under Analytics menu.
* In the Clusters tab, click on ***Create Cluster***.

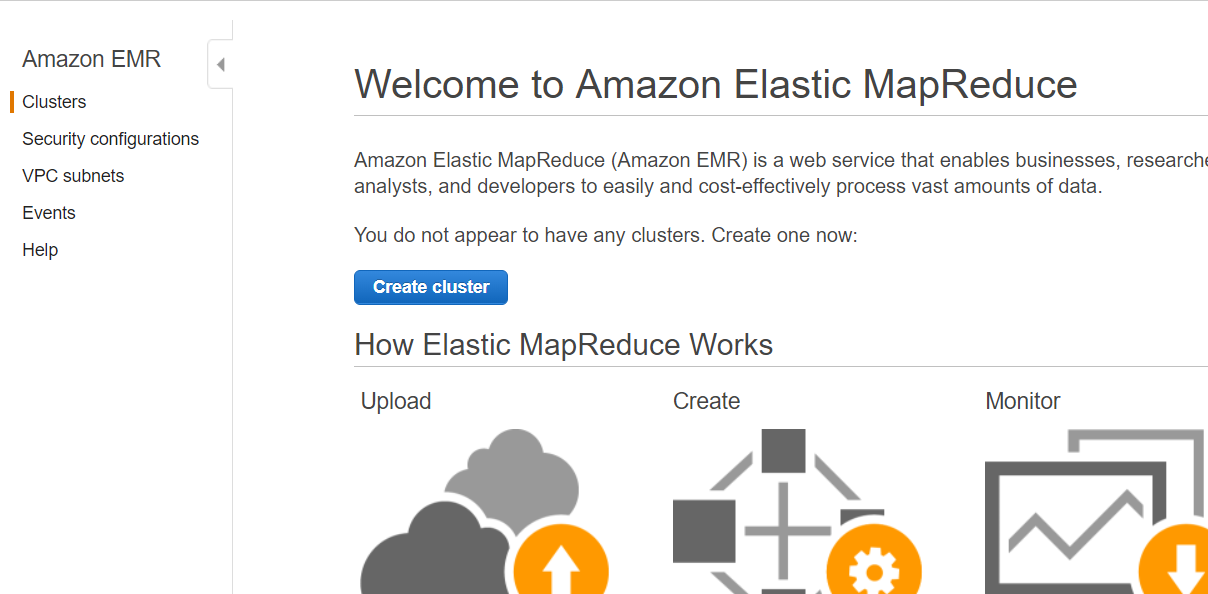


Figure EMR Clusters Tab

* Name the cluster.
* Check the option Logging (checked by default).
* In the S3 Folder option in General Configuration give the path of Logs folder in your S3 bucket.

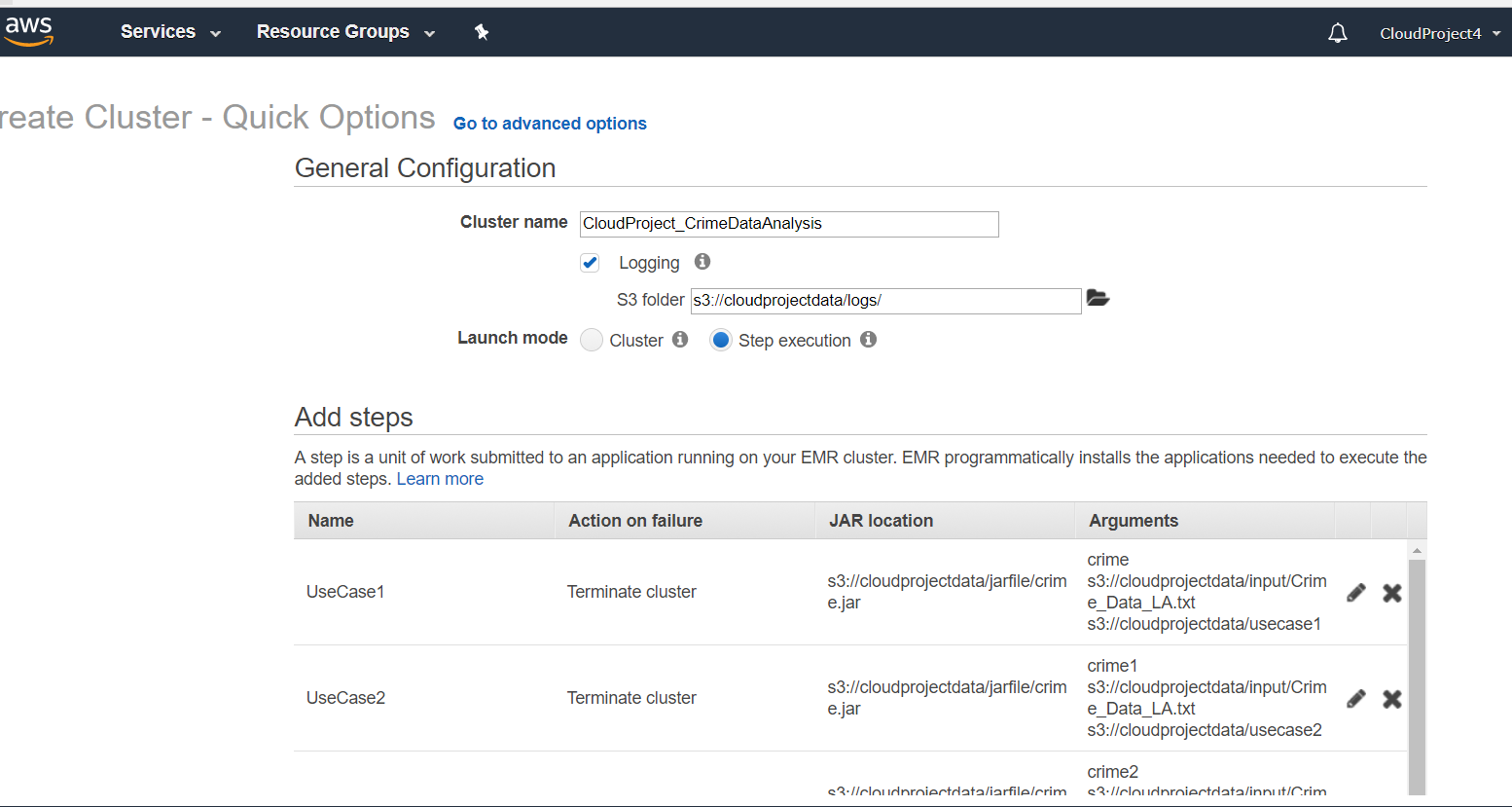


Figure Creating EMR Cluster

* Select ***Step Execution*** as Launch Mode.
* Select ***Custom JAR*** in ***Step Type*** dropdown options.

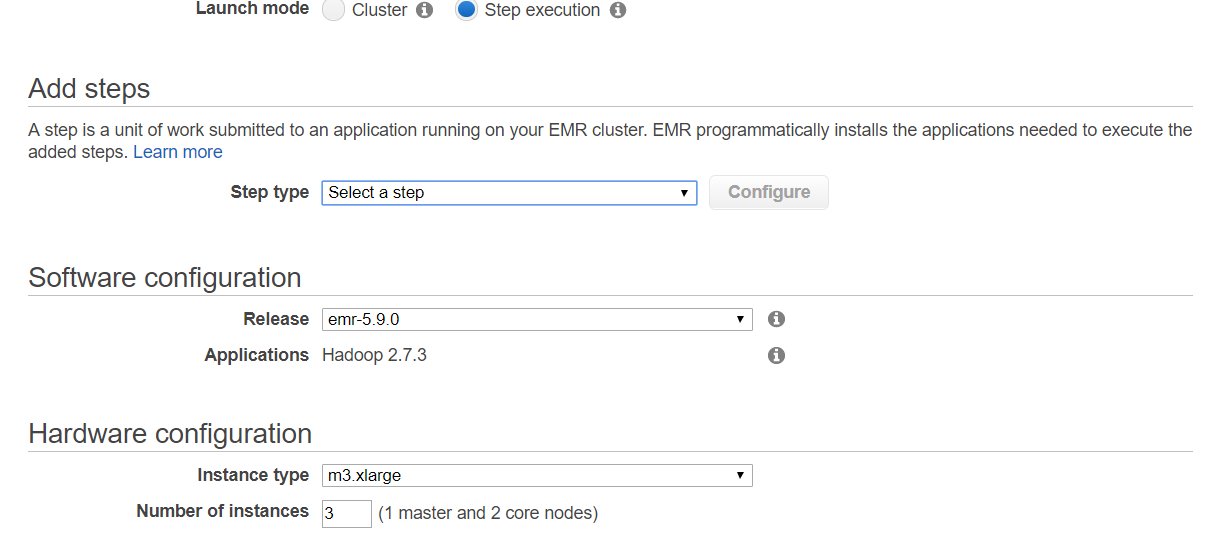


Figure Add Step in EMR

* Click on ***Configure***beside Step Type to configure the Custom JAR.

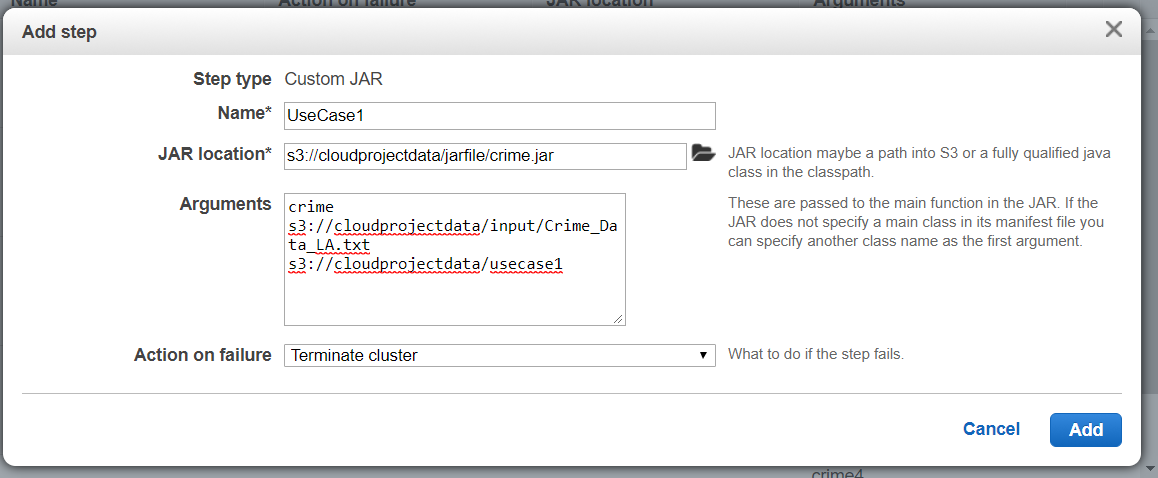


Figure Configure JAR in EMR

* Name the Step and select the location of the JAR File in the S3 bucket.
* Write the arguments for the Custom JAR in this order:
  + Name of the class (of Map Reduce program; must contain main method)
  + Path of the input file in S3 bucket. Example: s3://bucketname/input/data.txt
  + Path to the output folder. Example: s3://bucketname/outputs/output1
* Select ***Terminate Cluster*** option in ***Action on failure*** dropdown to terminate the cluster immediately if failure occurs during computation.
* Click on ***Add***. EMR lets us add multiple steps.
* In our project, we added 5 Steps, each with a different class

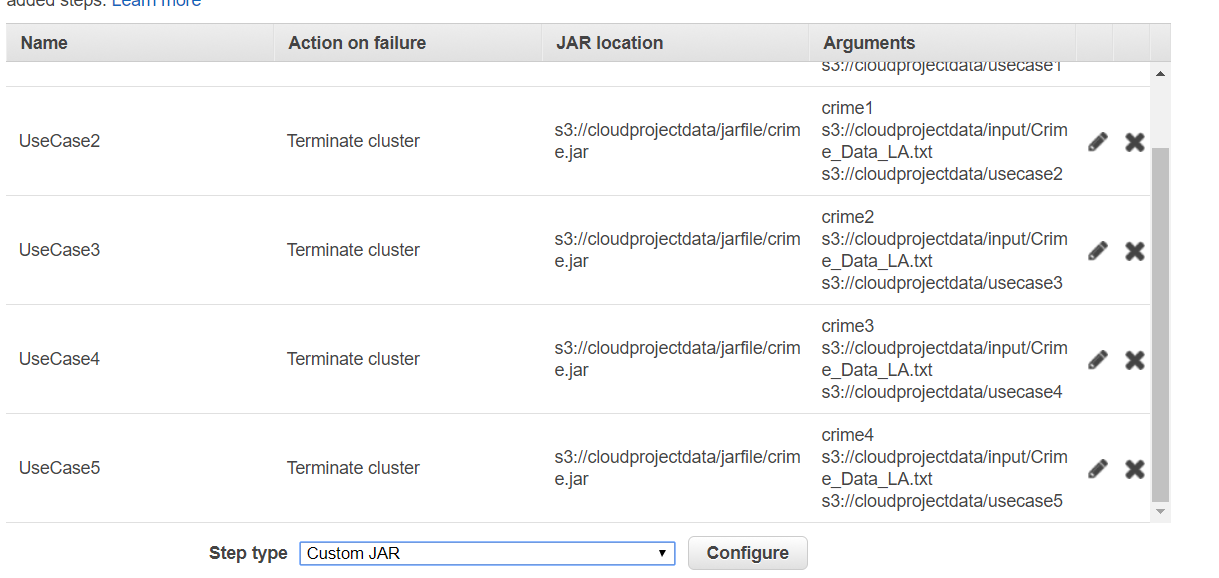


Figure Multiple Steps

* Select appropriate Software and Hardware Configurations. We used the default configurations as they suited our requirement.
  + Release: emr-5.9.0
  + Applications: Hadoop 2.7.3
  + Instance Type: m3.xlarge
  + Number of instances: 3 (1 master node and 2 core nodes)
* In the Security and access section, Select ***Default*** option for Permissions. We can add Custom EMR role and EC2 instance profile if we plan to use custom roles tailored to the cluster. By default, EMR\_DefaultRole and EMR\_EC2\_DefaultRole are set to EMR role and EC2 instance role respectively.
* Click on ***Create Cluster***. AWS will redirect you to the Cluster’s Console.

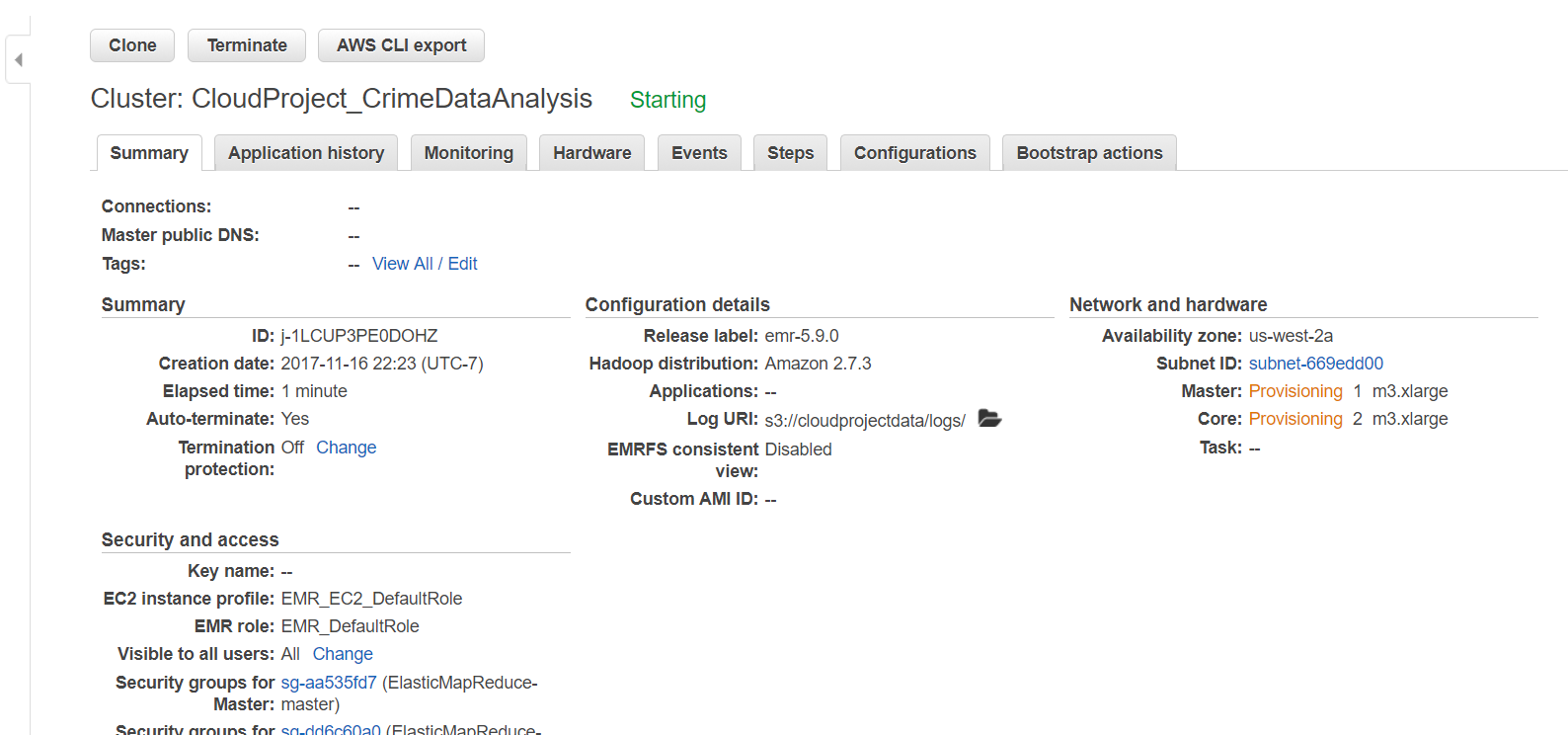


Figure Cluster Console

* AWS shows the status of the Master Node and Core Nodes in the Summary tab of the cluster.
* The status of the Cluster is displayed beside the title of the cluster in the Cluster Tab.
* Once the Nodes are provisioned, the jobs are executed, i.e., the analysis begins.

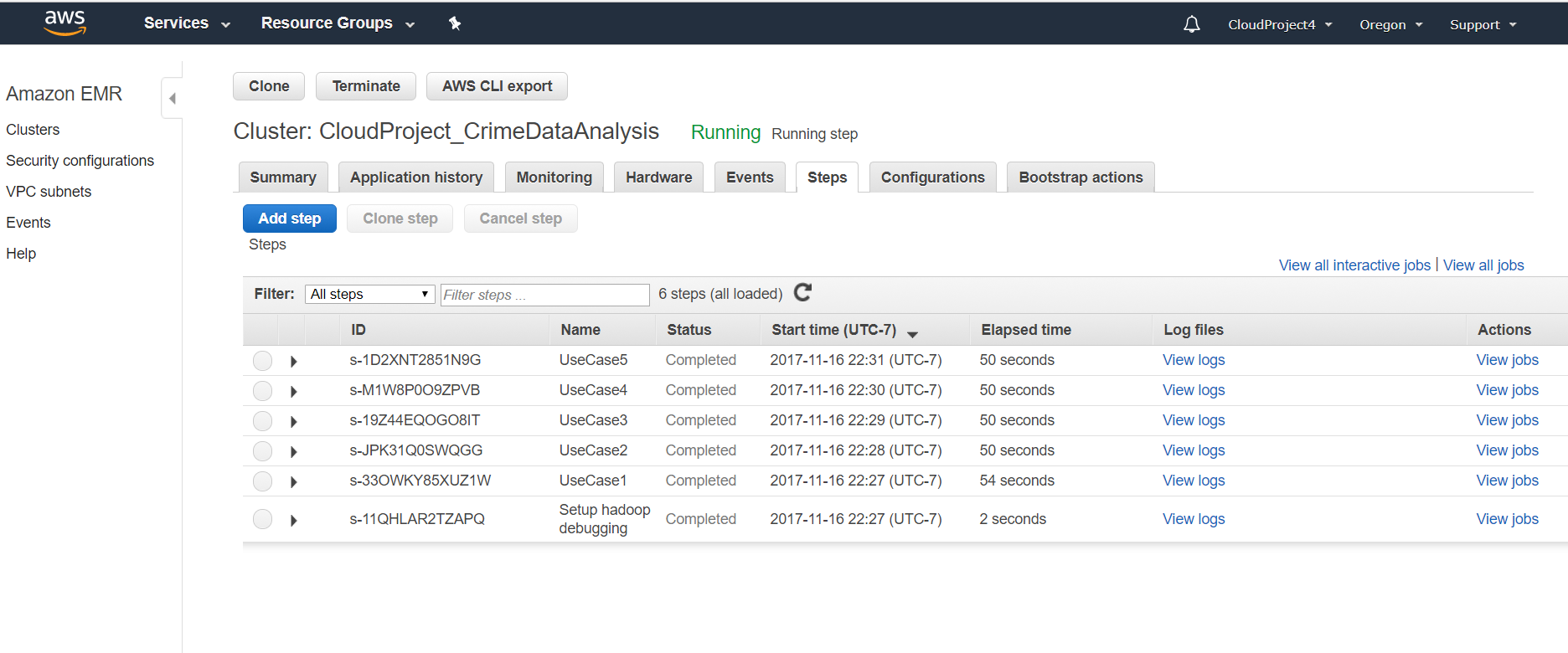


Figure Steps Tab of Cluster

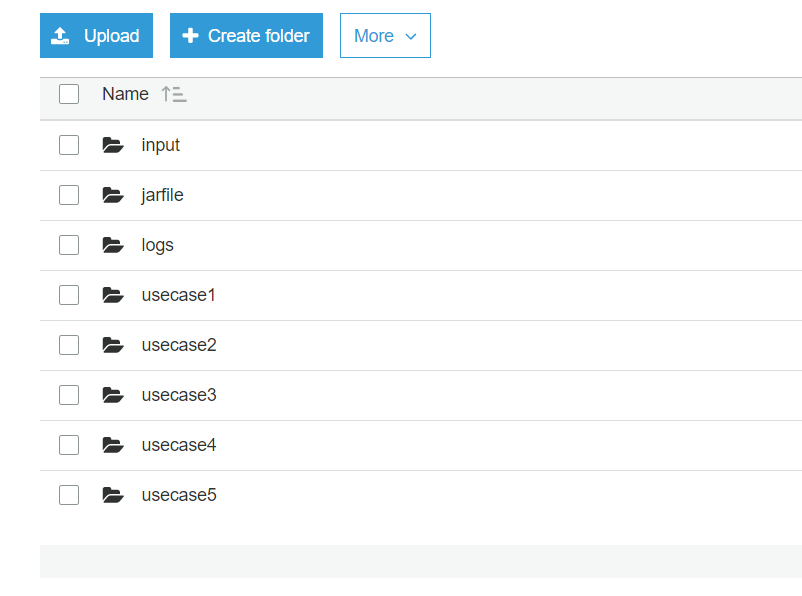


Figure View of S3 Bucket

* Once the EMR instance completes execution, it terminates based on our settings.
* The output folders in the S3 bucket contain multiple output files for each use case. These files are created by multiple nodes of the cluster and each of these nodes will process a part of the data.
* The entire result is formed by the combination of all the different partial output files.

## 4. Analyzing the data

4.1 In our project, we are analyzing the data to extract the following information.

* Number of crimes based on location and time of the day
* Number of victims of crimes based on age range and sex
* Number of crimes based on the type of premises
* Number of crimes committed based on weapons used
* Number of crimes based on nature and location

Based on the analysis, we can detect patterns of crimes across Los Angeles and give few workable suggestions to improve the prevailing order of the city such as focusing on hotspots (locations), changing policies, studying the nature of crimes, etc. For example, weapons used, and number of crimes committed with each type of weapon can help us understand if the city needs stricter gun control laws.

4.2 In our project, we are using Map Reduce program to analyze data.

Hadoop MapReduce has the following advantages:

* Map reduce is one of the earliest framework written in Java to process high volume of data for search processing. Below are the few advantages of Map Reduce.
* Simple Coding Model: With MapReduce, the programmer does not have to implement parallelism, distributed data passing, or any of the complexities that they would otherwise be faced with. This greatly simplifies the coding task and reduces the amount of time required to create analytical routines.
* Fault Tolerance: Because of its highly distributed nature, MapReduce is very fault tolerant. Typically the distributed file systems that MapReduce support, along with the master process, enable MapReduce jobs to survive hardware failures
* Supports Unstructured Data: Unstructured data is data that does not follow a specified format for big data. If 20 percent of the data available to enterprises is structured data, the other 80 percent is unstructured. Unstructured data is really most of the data that you will encounter.

We thought it’s important to learn and implement the HDFS file system and Map Reduce architecture before embracing latest frameworks such as Spark and Storm, as it'll provide strong foundation for future learning.

We are making use of 5 java classes to extract information mentioned in 4.1. We pack the 5 java classes into a JAR file to run jobs on Amazon EMR. The details of each java class is mentioned below. Each class consists of two sub-classes and a main method. One of the sub-classes extends Mapper Class while the other extends Reducer Class from Hadoop distribution. The main method drives the job.

To run the job, the main method is called. The main method contains the driver code. The main method consists of the following sequence:

* Create an object called job of class Job.
* Configure the job by calling functions such as *setJarbyClass, setOutputKeyClass, setMapperClass*, etc. as shown in *Figure 15.*
  + *setJarbyClass*: Used to assign a specific class to the job from the JAR file.
  + *setOutputKeyClass*: Used to set the types expected as output from both the map and reduce phases.
  + *setOutputValueClass*: If the expected output is different for map and reduce, use this method to set the type.
  + *setMapperClass:* Used to set the map class.
  + *setReducerClass:* Used to set the reduce class.
  + *setCombinerClass:* Used to set the combiner class. Reducer Class is the default argument.
* Set the input format as file and add input path (given as arguments to the main method).
  + *FileInputFormat:* It is the base class for all file-based InputFormats.
  + *FileOutputFormat:* It is the base class for all file-based OutputFormats.
* Set the output format as file and add output path (given as arguments to the main method).
* Submit the job, then poll for progress until the job is complete.
* The driver code is similar for all the classes. Change the attributes based on the class.

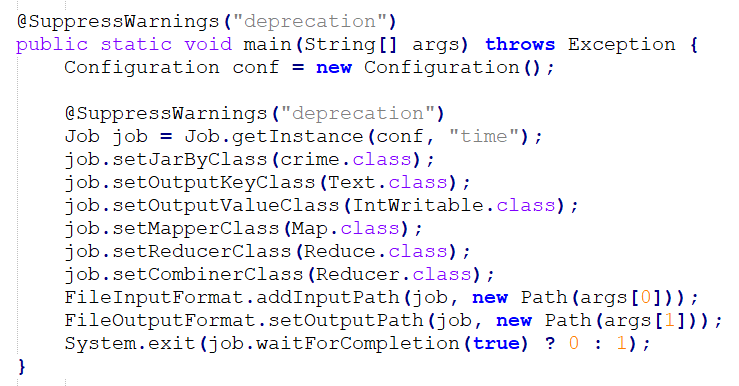


Figure Map Reduce Driver Code

**Mapper Class:**

* The Map class extends Mapper class available in Hadoop distribution.
* Create a new Text object to generate a key-value output.
* Override the default map method.
* The map method splits the string based on comma.
* Write the logic to obtain the required output and create a key-value pair.
* Write the key-value pairs to output using *context.write*() where context is an object of class Context.

For example:

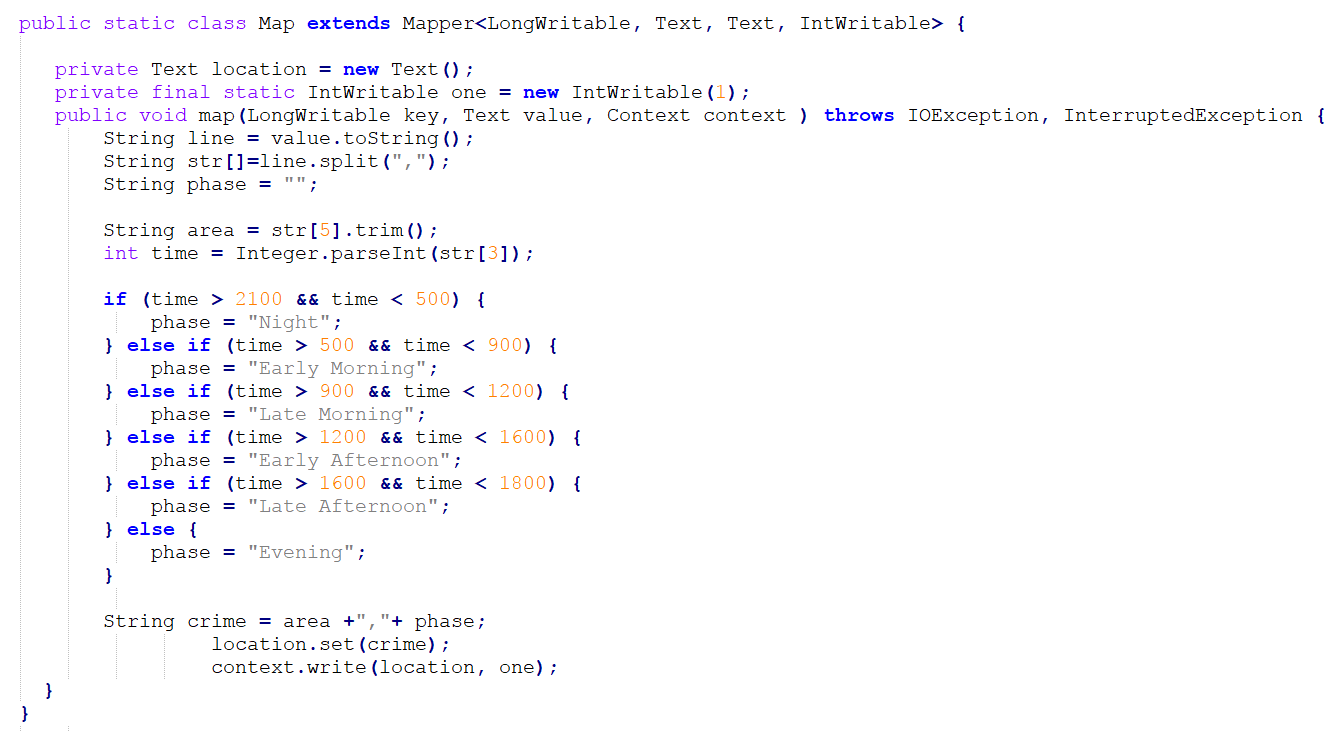


Figure Map Class (Crime class)

**Reducer Class:**

* The Reduce class extends Reducer Class available in Hadoop Distribution.
* Override the default reduce method.
* The reduce method combines similar output key-value pairs generated by mapper tasks.
* Write the logic to combine similar key-value pairs and generate output values.
* Write the key-value pairs to output using *context.write*() where context is an object of class Context.

For example:

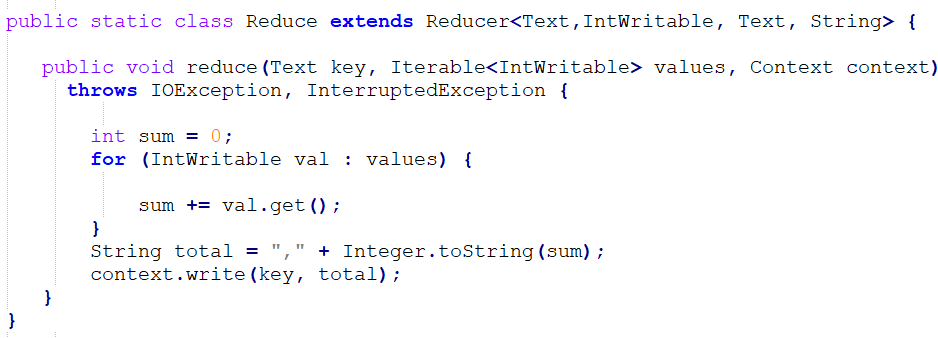


Figure Reducer Class (Crime class)

The following table contains the list of classes used in our project and the objective for each of it.

|  |  |  |  |
| --- | --- | --- | --- |
| **Sno.** | **Class Name** | **Job Name** | **Objective** |
| 1 | Crime | Usecase1 | To obtain number of crimes based on location and time of the day |
| 2 | Crime1 | Usecase2 | To obtain number of victims of crimes based on age range and sex |
| 3 | Crime2 | Usecase3 | To obtain number of crimes based on the type of premises |
| 4 | Crime3 | Usecase4 | To obtain list of weapons used and number of crimes committed with each type of weapon |
| 5 | Crime4 | Usecase5 | To obtain number of crimes based on nature and location |

Table Class and its Objective

## 5. Displaying the results of the big data analysis

5.1 In our project, we used Tableau to visualize the data obtained from analysis. To do so, we connected Tableau to Amazon Athena, a server-less interactive query service that makes it easy to analyze big data in S3 using standard SQL. Tableau desktop then connects directly to the data source stored as data tables on Amazon Athena. This data is used to create dashboards that comprehensively display the results of the analysis.

5.2 We created a database called db\_CrimeData to hold the output data. Different tables are created for each use case. The table tb\_UseCase1Results is created for use case 1. Similarly, tables are created for the other use cases.

* Click on Create table in the Database window.

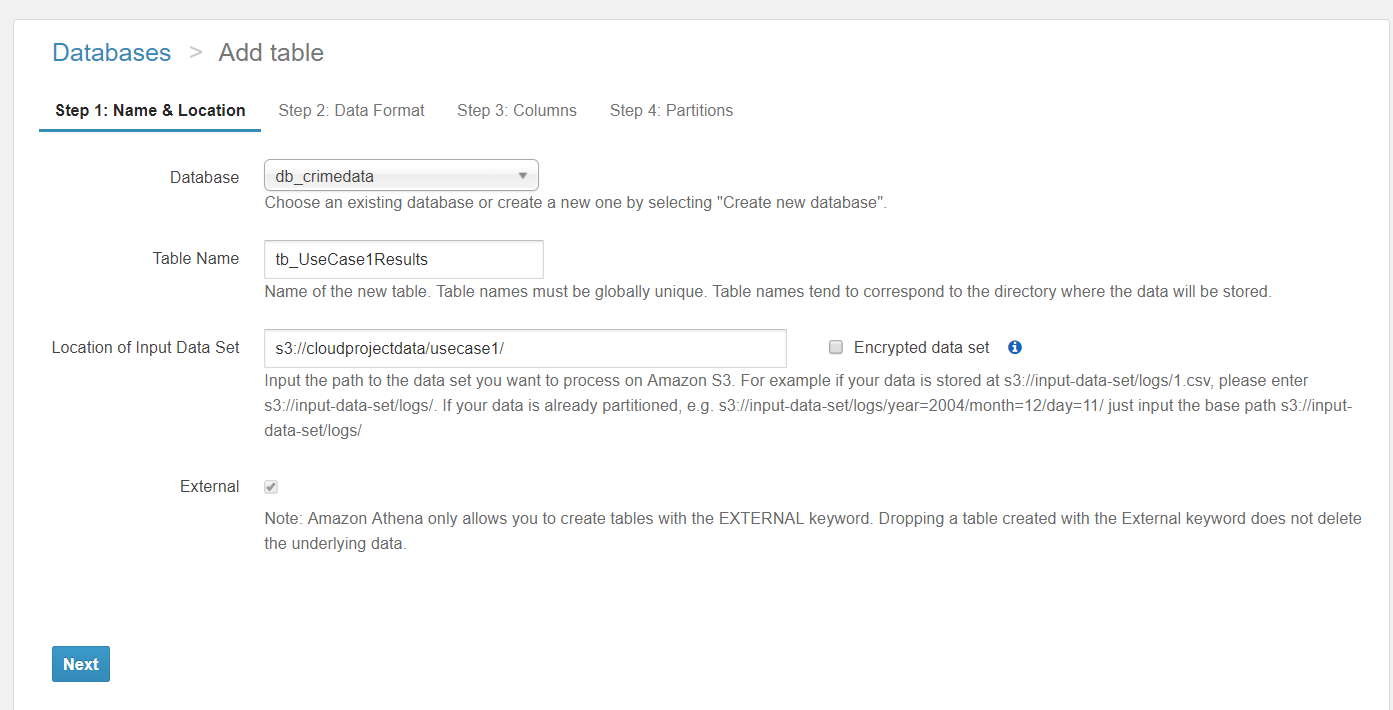


Figure Table creation for use case 1 on Amazon Athena

* Enter the table name and the data source location which is the output files in the S3 bucket.
* Choose text files as data format in step 2.
* Enter the datatype and name of each columns in the Columns tab.
* Create the table. Repeat the process for each use case.

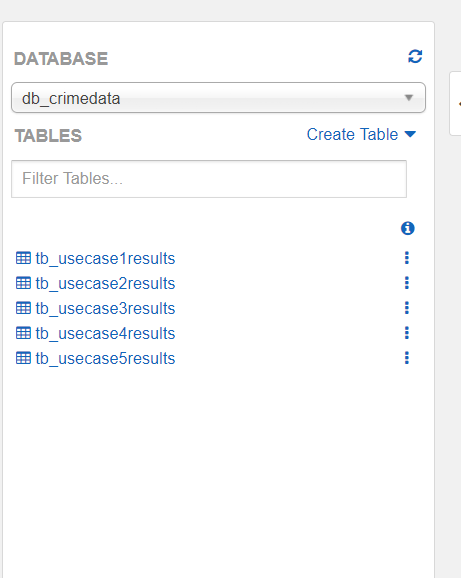


Figure View of the list of all the tables created

We use this data source to create dashboards on tableau desktop.

* Open tableau desktop. Click on Amazon Athena under connect to a server.
* Enter the details of the AWS account which holds the data tables and click on sign-in.

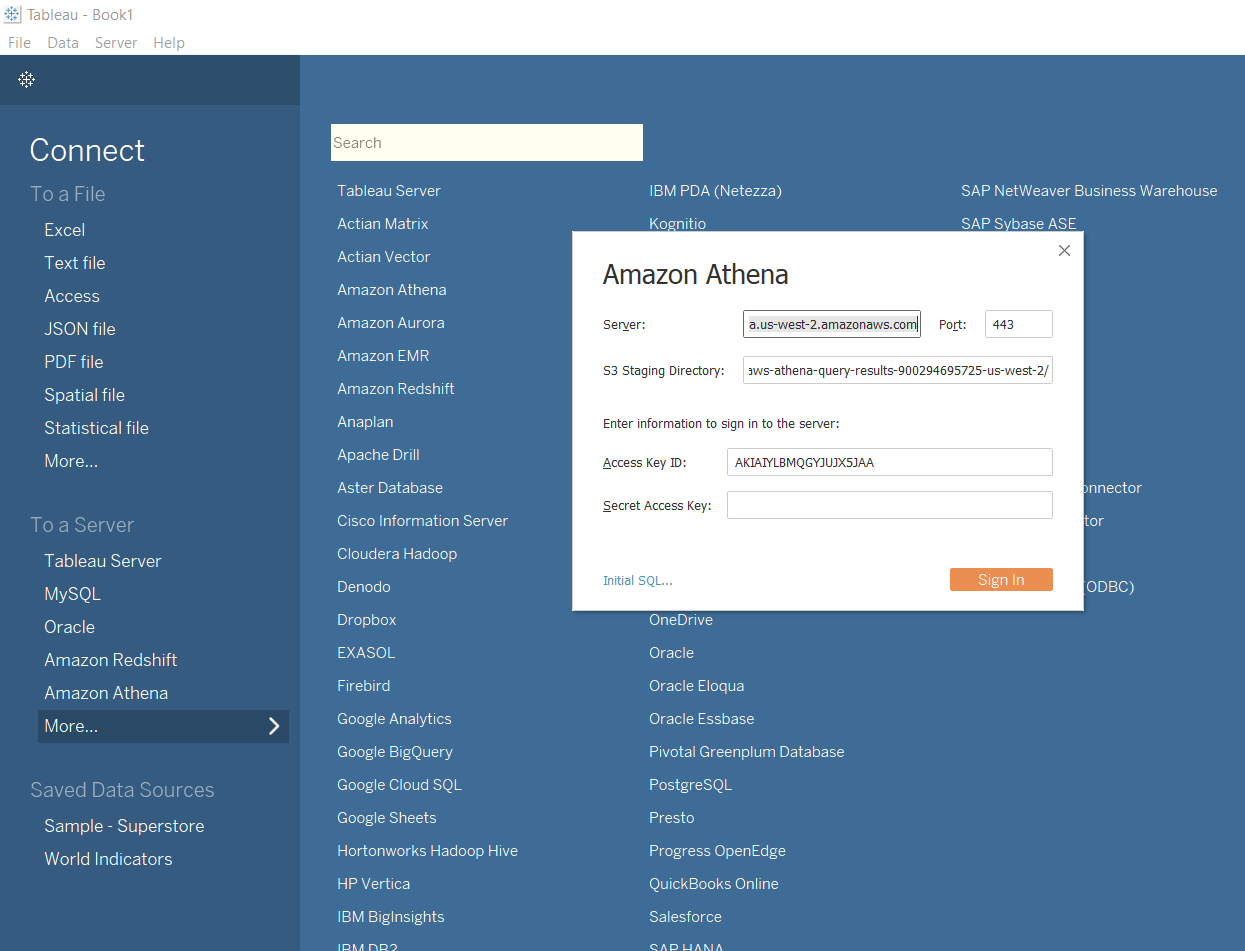


Figure View of Access to Athena

* The table that is needed for each use case is chosen.

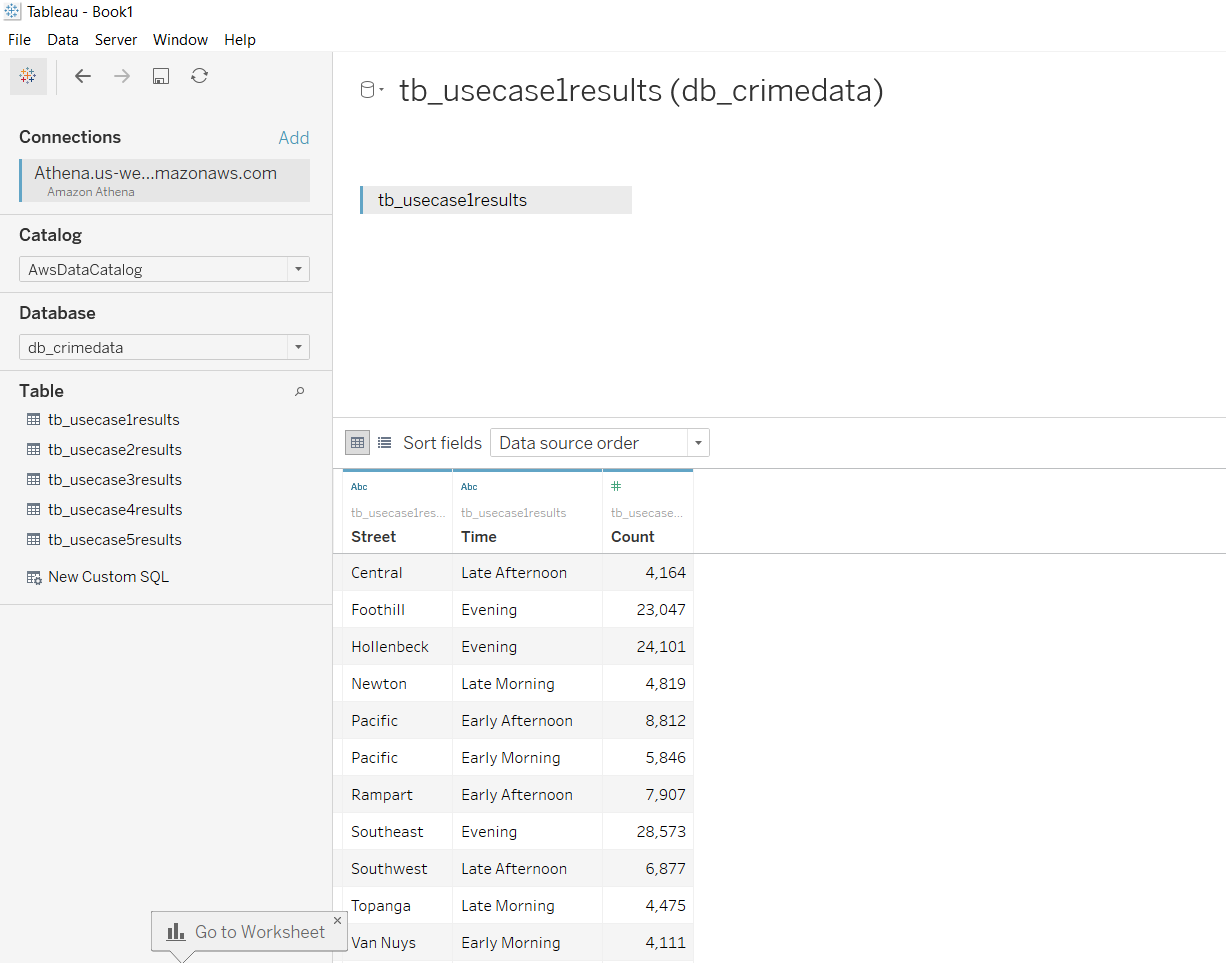


Figure Athena - Choosing Table

* The report is built in the worksheet tab using this data source.
* The data table columns are populated in the dimensions window. These dimensions are dragged and dropped on to the worksheet to create tables and charts.

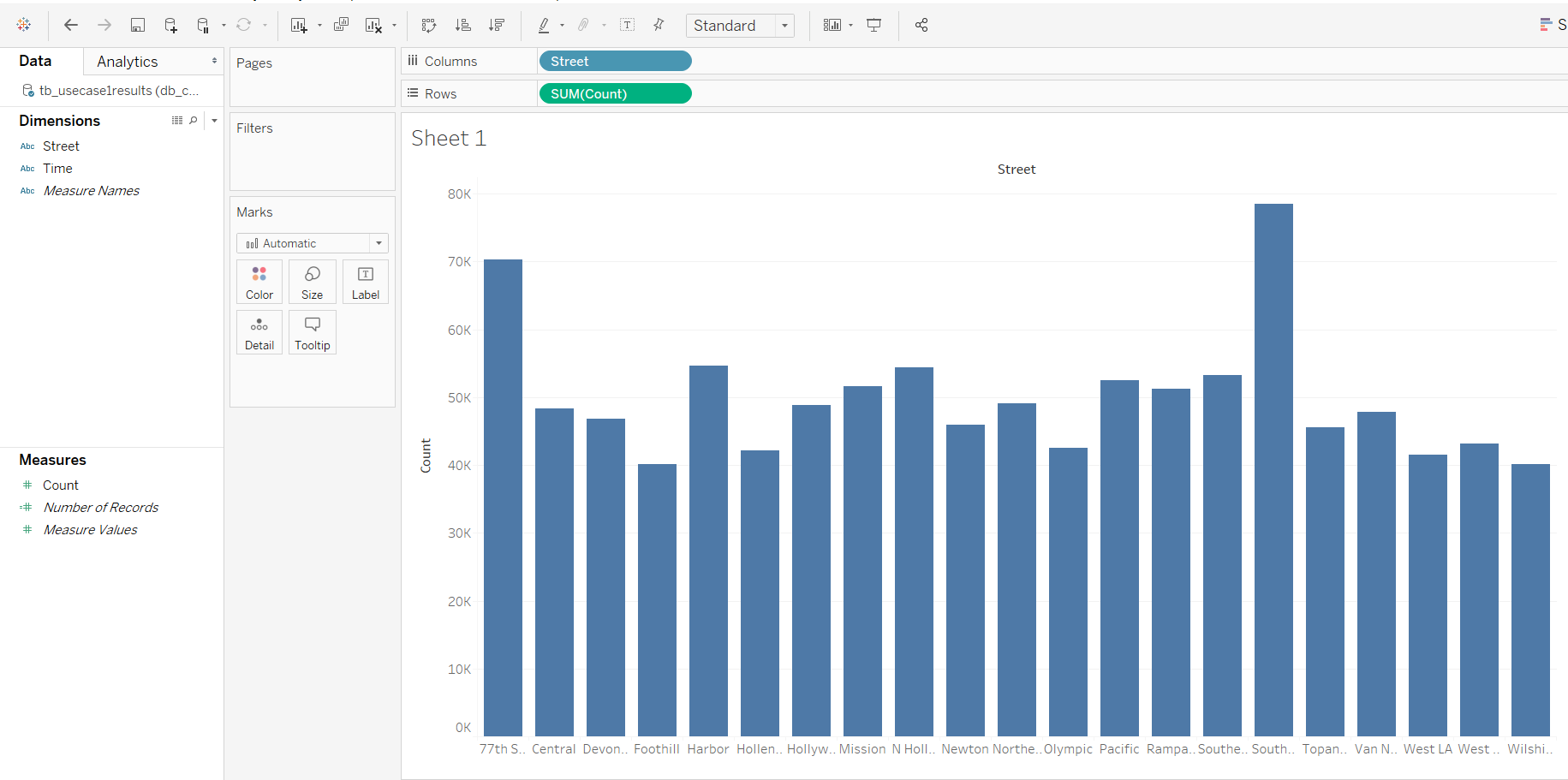


Figure Tableau Dashboard Sample

* We created dashboards using worksheets for each use case.
* We used these dashboards to create a story which holds all the use cases as a single unit.
* We then publish the workbook on Tableau Public web server which can be accessed with the following link:

<https://public.tableau.com/profile/ramya.chandrashekar#!/vizhome/Final_CloudProject/CloudProject-AnalysingCrimeData>

5.3 Description of the reports:

The final workbook contains the following 5 use cases

1. Use case 1: We created a bar graph representing the crime rate based on time for the streets in Los Angeles. The time and sheet filters can be used to view and compare data for a particular street and time.

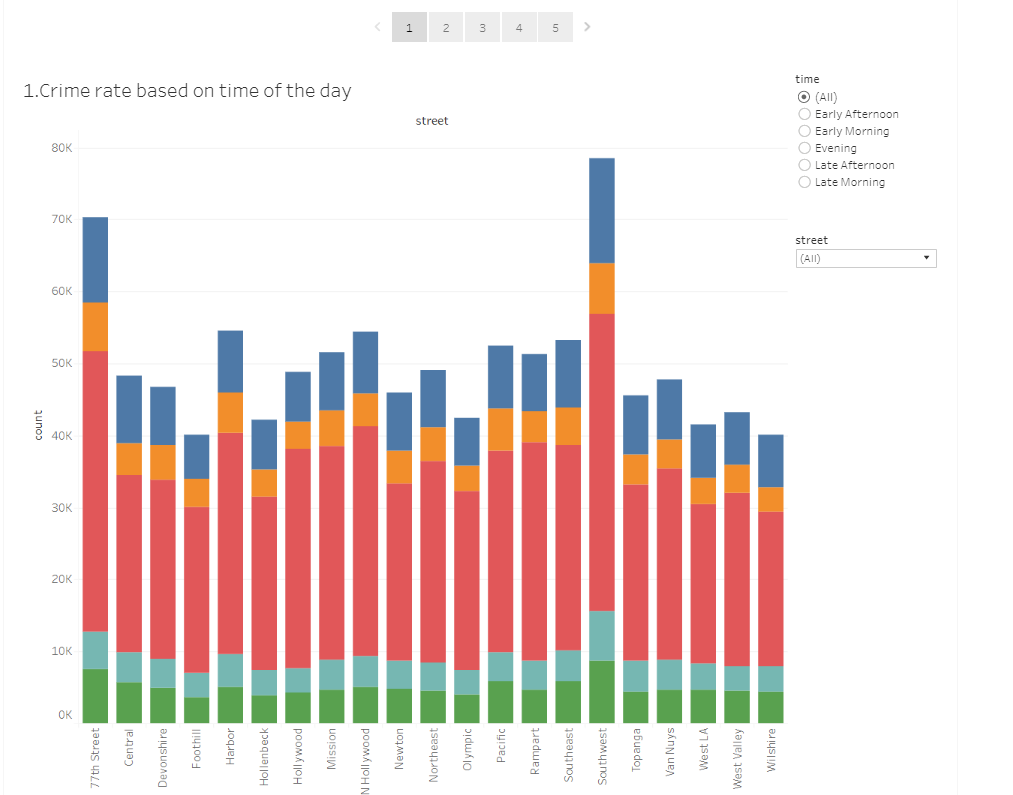


Figure Tableau Dashboard for Usecase 1

1. Use case 2: We created a nested bar graph representing crime rate based on sex and age.

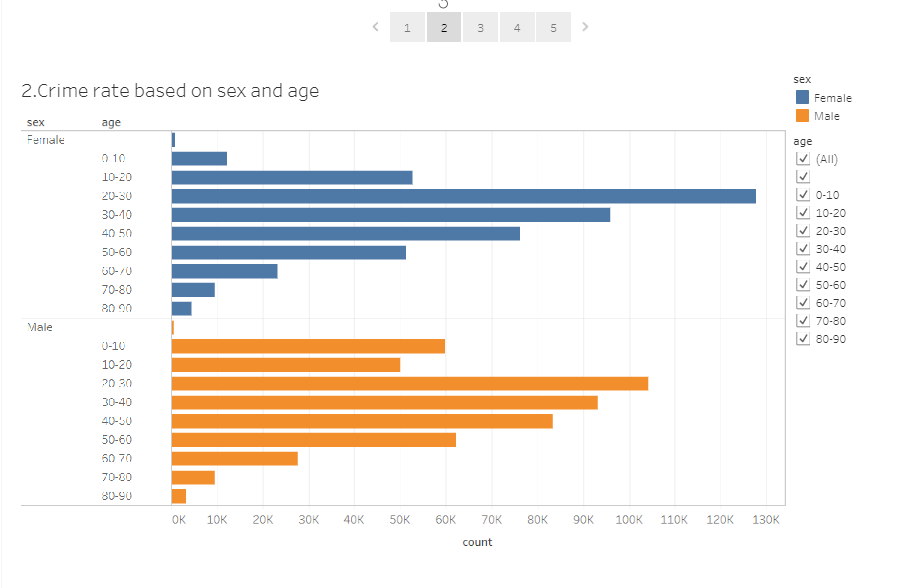


Figure Tableau Dashboard for Usecase 2

1. Use case 3: We created a highlights graph that represents the crime rate based on location. The location filter allows to filter and compare crime at particular locations.

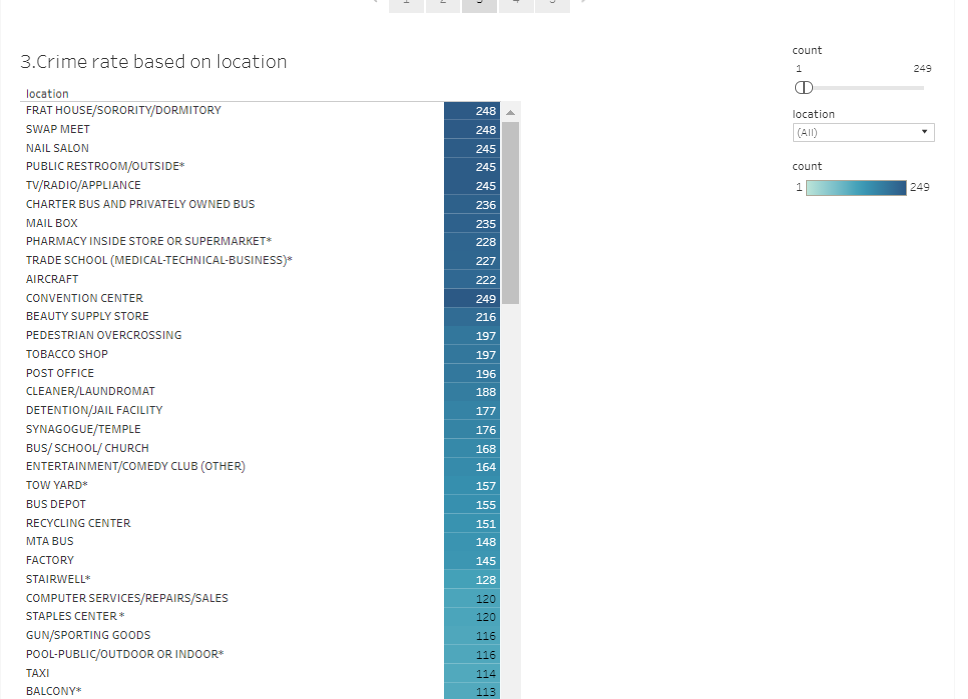


Figure Tableau Dashboard for Usecase 3

1. Use case 4: We created a bar graph that represents the crime rate based on type of weapon. Below is a packed bubbles chart that represents the top 6 lethal weapons used and the count of crime.

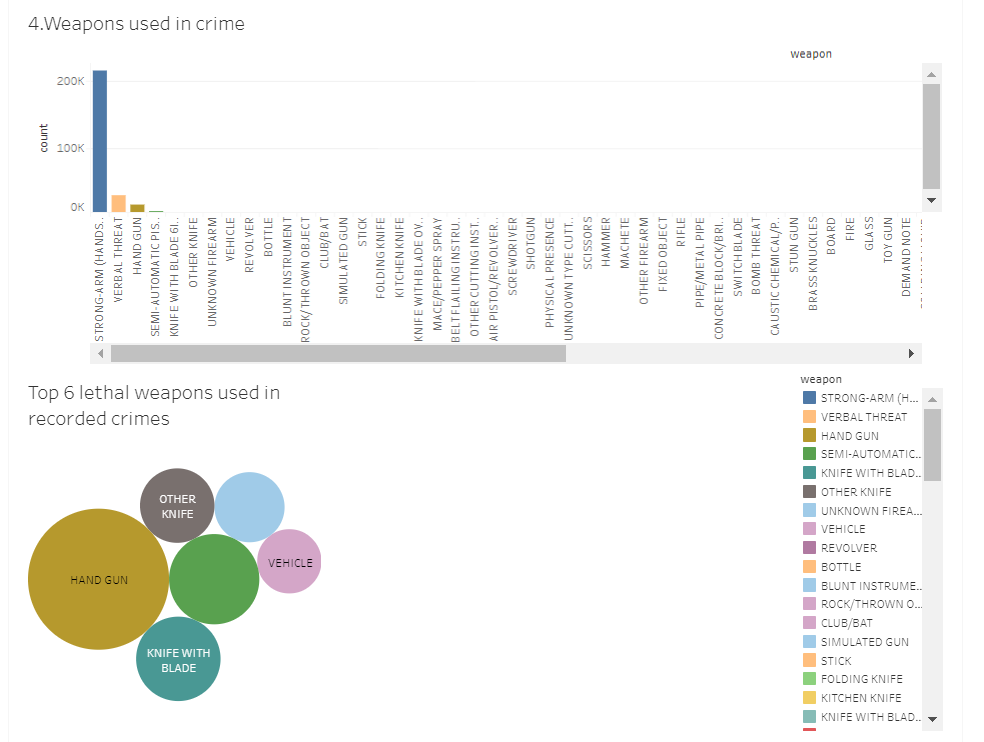


Figure Tableau Dashboard for Usecase 4

1. Use case 5: We created a treemap that represents the rate of crime based on street. And on click of the street, the highlights table below filters out the crime rate in that street based on type of crime.

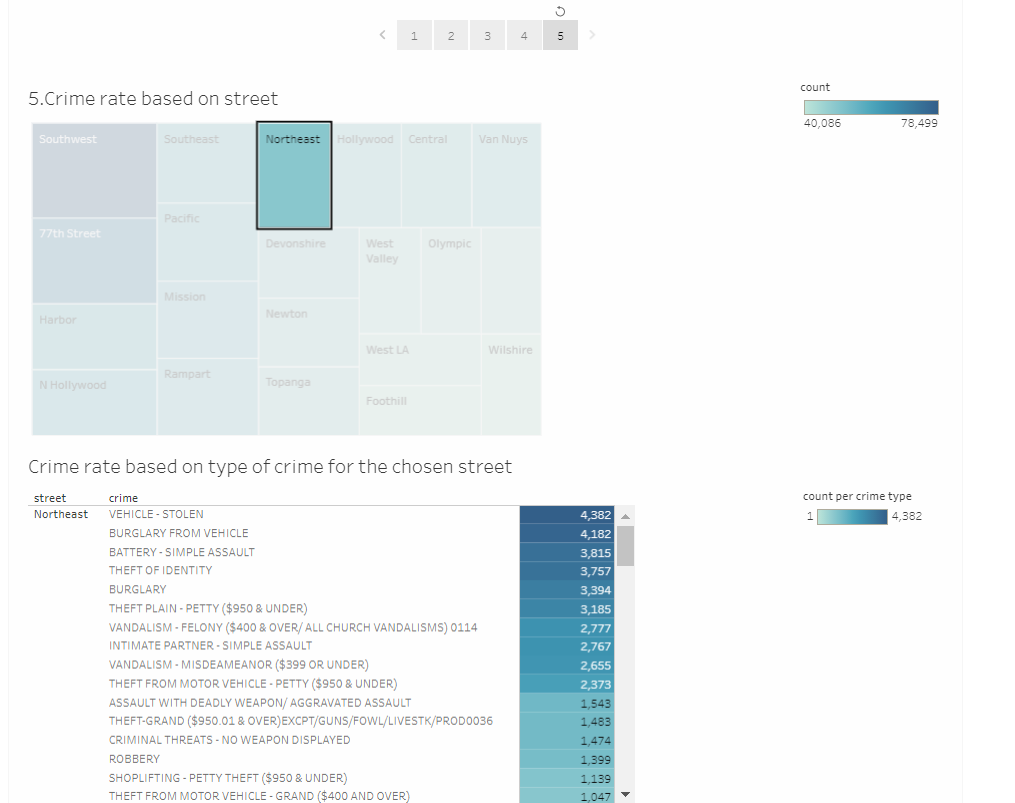


Figure Tableau Dashboard for Usecase 5

Summary

The objective of the project was to analyze historic crime data of the city of Los Angeles to understand the patterns of crimes using cloud services. The project was divided various stages. Firstly, the dataset of historic crime data of the city of Los Angeles was downloaded from *data.gov* website. Secondly, the dataset was stored on Amazon S3. Thirdly, Mapper, Reducer and Driver classes were written in Java to analyze the data in the dataset. Fourthly, Amazon EMR, a distributed data processing system in cloud, was setup. The EMR was configured with the location of output and the dataset on Amazon S3. The location of the JAR file on Amazon S3 containing all the Mapper, Reducer and Driver classes was added to the EMR during configuration. The EMR was setup in step execution mode so that creating a new instance of EC2 for each use case could be avoided. Lastly, the output from the EMR was visualized using Tableau. To do so, a database was created on top of the Amazon S3 bucket using Amazon Athena. Then, Tableau desktop was connected directly to the data source stored as data tables on Amazon Athena. The data was used to create dashboards that comprehensively displayed the results of the analysis.

To achieve the objective, various cloud technologies such as Amazon S3, Amazon EMR, and Amazon Athena, and essential technologies for data analysis such as Map Reduce using Java and Apache Hadoop were learnt.

References

1. Crime data from 2010 to present - <https://catalog.data.gov/dataset/crime-data-from-2010-to-present>
2. Amazon S3 - <https://aws.amazon.com/s3/getting-started/>
3. Amazon EMR - <http://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-what-is-emr.html>
4. Running EMR Job using custom JAR - <https://muhammadkhojaye.blogspot.com/2012/04/how-to-run-amazon-elastic-mapreduce-job.html>
5. Amazon Athena - <http://docs.aws.amazon.com/athena/latest/ug/what-is.html>
6. Introduction to Apache Hadoop - <https://opensource.com/life/14/8/intro-apache-hadoop-big-data>
7. Tableau - <https://www.tableau.com/learn>
8. Connecting Tableau and Athena - <https://www.tableau.com/about/blog/2017/5/connect-your-s3-data-amazon-athena-connector-tableau-103-71105>
9. Hadoop MapReduce Tutorial - <https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html>
10. Hadoop vs Spark - <https://www.scnsoft.com/blog/spark-vs-hadoop-mapreduce>