**Milestone 1**

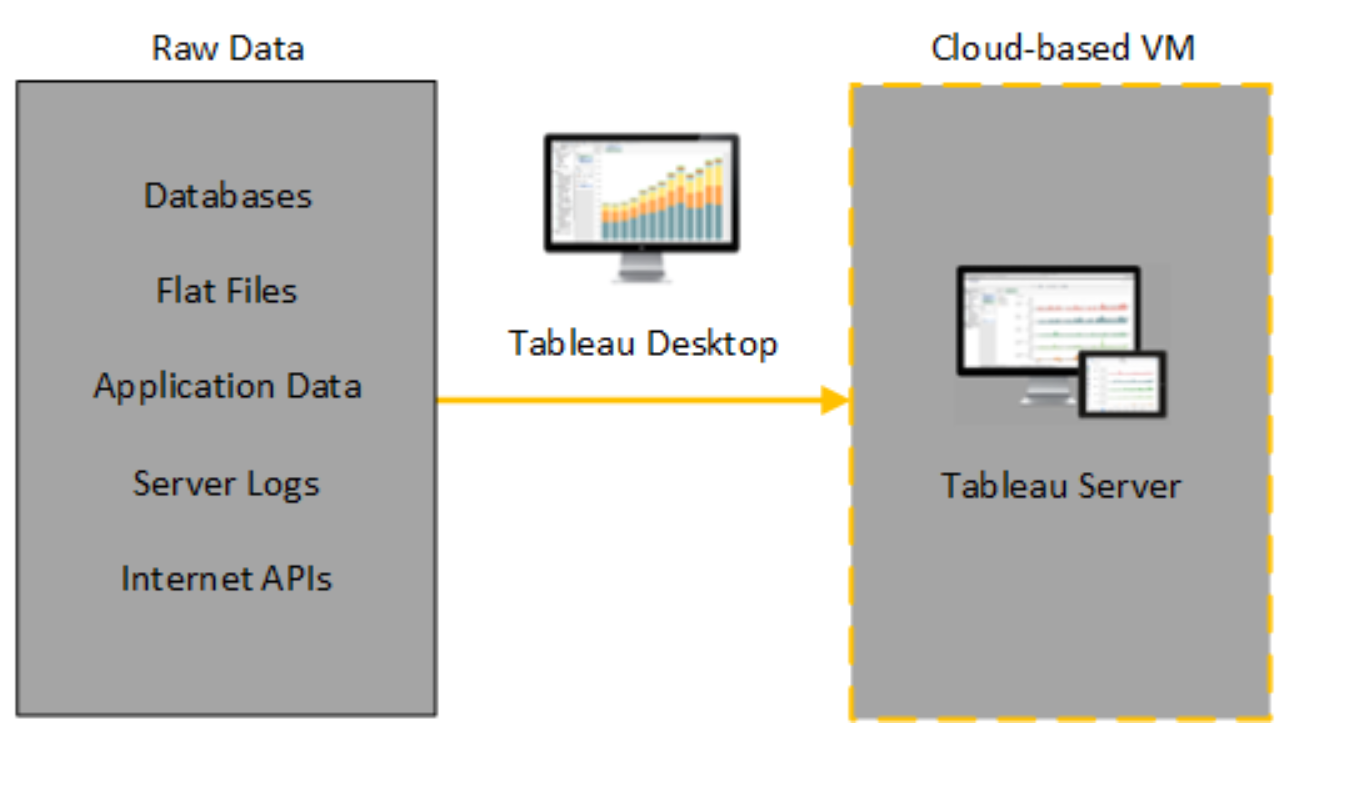
**Report of project cloud visualization**

A log file is a type of documentation automatically generated by the system, and it records significant changes like a file creation or its modification time, user name access, adjustments, and others. These files produced contain critical information for companies or organizations. Examples of log files are error logs, authentication logs, audit logs, or system logs.

Data exploration is the first step of data analysis used to explore and visualize data to uncover insights from the start or identify areas or patterns to dig into more. Using interactive dashboards and point-and-click data exploration, users can better understand the bigger picture and get insights faster.

To download the CSV file from the cloud we need an AWS to be installed in the system and the following commands are used to download the CSV file from the cloud.

Users can investigate log data through the user interface from various perspectives, spatiotemporal regions, and query criteria. In the timeline view, which includes the scalable Theme River and other elements Arc diagrams are used to visualize the distributions of RAS logs. The first layer depicts machine job allocations as semitransparent rectangles. And the second layer is about the tableau desktop and the third view is about the data processing in the server.



The data processing cycle consists of a series of steps where raw data (input) is fed into a system to produce actionable insights When data is processed, it is collected and converted into usable information. Data processing, which is typically performed by a data scientist or team of data scientists, must be done correctly so that the end product, or data output.

**Sample Table to Complete Data Visualization**

Let's create a Tableau table to help with a map visualization. We want to show the breakdown of trade categories as well as the corresponding values. To create a new Tableau sheet, click the + icon located next to the current tab. As a result, your Tableau workspace should now be empty. You should now see a Tableau table with the Country names listed to the left, followed by Export, Import and Turnover columns. To rearrange the columns’ order in Tableau, click and drag the Turnover header and then move it to the left.

Commands to download CSV files from the cloud using AWS:

* C:\> msiexec.exe /i <https://awscli.amazonaws.com/AWSCLIV2.msi>
* C:\> AWS –version

aws-cli/2.7.24 Python/3.8.8 Windows/10 exe/AMD64 prompt/off

* aws s3 ls s3://mit-supercloud-dataset/datacenter-challenge/202201/ --no-sign-request

A directory listing can be obtained using the following command :

* aws s3 cp s3://mit-supercloud-dataset/datacenter-challenge/202201/labelled\_job\_stats.csv. --no-sign-request

{339 labelled\_job\_stats.csv will be downloaded }

* aws s3 cp s3://mit-supercloud-dataset/datacenter-challenge/202201/labelled\_jobids.csv . --no-sign-request

      82332 labelled\_jobids.csv will be downloaded using the following command

Tableau helps people and organizations be more data-driven You can export directly to Excel the data used to generate the view formatted as a crosstab. When you export your view as a crosstab, Tableau automatically opens the Excel application and pastes a crosstab version of the current view into a new Excel workbook.

To import the CSV file in tableau these are the following steps :

* Create a Data Source or Add a New Connection with Clipboard Data.
* Select the data you want and copy it to the clipboard.
* Open Tableau Desktop and do one of the following:

On the data source page, select Data > Paste Data as Connection or Paste Data as Data Source. ...Select File > Save to save the data source. We also trying to implement the D3js framework to show the visualization the D3.js is an open-source library and the source code of the library is freely available on the web at https://d3js.org/ website.

To implement this framework for visualization we need to write the code in python using the frontend JavaScript and HTML.

d3.csv () function is asynchronous and code after for(s in sources){ starts before the outdated array is filled up.

That's why you see it as undefined and as valid inside d3.csv() accessor functions.

You have to use outdated only when all files are read.

One possible solution:

d3.csv(‘your CSV.csv').

then(function(data)

{ // data is now whole data set // draw chart in here! }) .catch(function(error){ // handle error })

//sample page

**<script src="https://d3js.org/d3.v5.min.js"></script>**

**<html>  
<head>  
<title>My sample HTML page</title>  
</head>  
<body>  
<h1>D3.js</h3>   
<p>I'm using D3.js here !</p>  
<p>We all are using D3.js here !</p>  
<script src="https://d3js.org/d3.v5.min.js"></script>  
<script>  
d3.select('h1').style('color', 'green');  
d3.selectAll('p').style('font-size', '30px');  
</script>  
</body>  
</html>**

The above sample code is only to run one single CSV file for visualization

Dynamic properties and Data binding: Another main concept of D3 is mapping a set of data to the DOM elements in a dynamic manner. Here we can introduce datasets and then we can update, append and display the DOM elements using those datasets, in real time.

**let dataset = [1,2,3,4,5]   
d3.selectAll(‘p’) //Select 'p' element  
.data(dataset) //data()puts data into waiting for processing  
.enter() //take data elements one by one  
.append(‘p’) //for each data item appending <p>  
.text(‘Sample text’); //add sample text to each**