Lab 16. ETL Best Practices for Tableau

Tableau is a leader among BI tools and it brings great power to any organization. But with great power, there is great responsibility. Tableau is a tool that does what the end user wants. However, Tableau is at the top of the iceberg and it works with other tools, such as Data Warehouse and ETL (ETL allows us to load data into Data Warehouse or Data Platform). The key aspect of any Tableau implementation is quality analytics and actionable business insights that will drive business decisions and help the business grow. As a result, it is important to integrate Tableau with ETL and Data Warehouse.

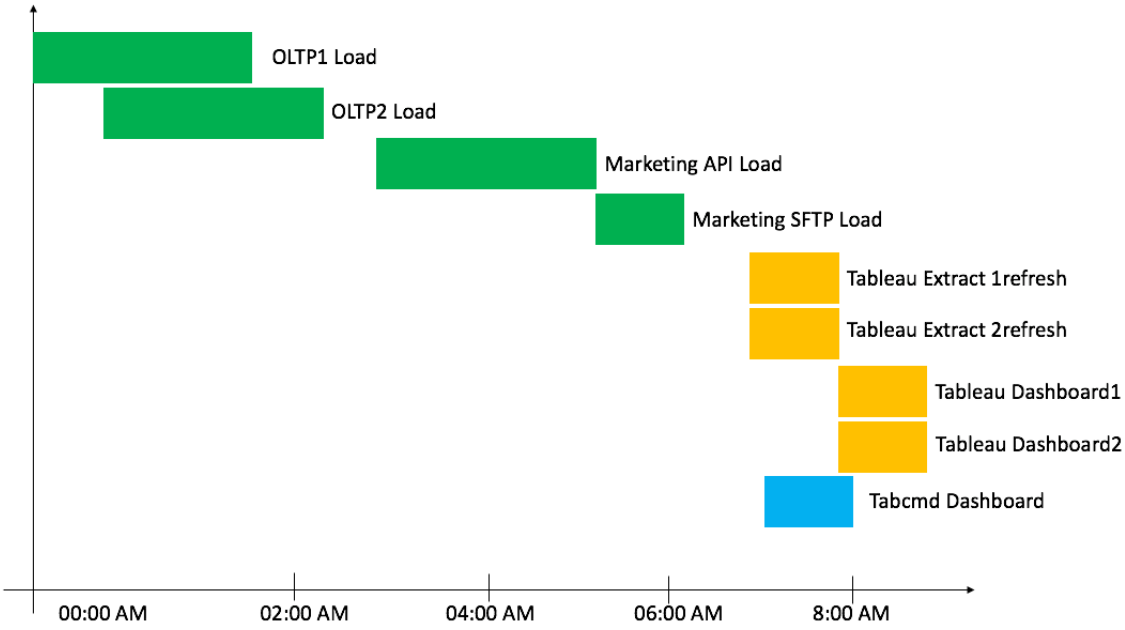
In this lab, we want to go through one of the most common problems that we spot on every project. Usually, organizations tend to use multiple independent vendors for their BI solution. As a result, there could be a gap, especially between a Data Warehouse and BI tool. We will cover the following topics:

- Getting started with Matillion ETL
- Deploying Tabcmd on Linux
- Creating Matillion Shared Jobs

Introduction

Let's understand this situation better through the following example. In one of our projects, a customer had a Cloud Data Warehouse and uses Matillion to load it. The customer was using Tableau as a primary BI tool.

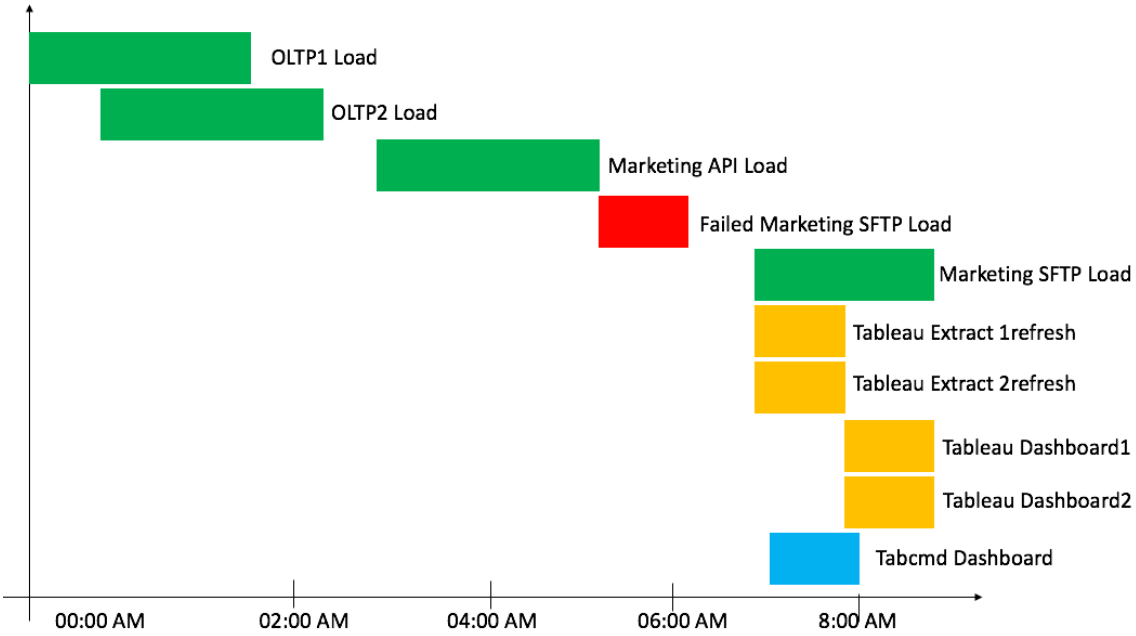
Let's look at the ideal ELT, which is shown in the following screenshot:



There are two individual processes here. The green one is Matillion, that is scheduled via Matillion Scheduler. The orange one is Tableau and it is scheduled via Tableau Server. Usually, we assume that ETL is done at 6 am and we scheduled Tableau extracts and dashboards a bit later. In our example, it is 7 am. In addition, we are using `tabcmd` and schedule Tableau Reports via [**Windows Task Scheduler**].

As you might guess, the marketing data source isn't the most reliable. Let's consider a scenario, where SFTP was delayed files delivery.

As a result, the ELT job was failed and automatically restarted later at 7 a.m., as shown in the following screenshot:



In our scenario, the ELT process was finished around 9 am. This should be the time of triggering BI reports and refreshing exports.

As a result, business users got their dashboards with inconsistent data and they usually send all ELT/DW emails into the spam folder. Based on our experience, around noon, users will realize that they spent half of the workday for nothing by working with inconsistent data.

We will simulate this issue and learn best practices for ETL and Tableau using Matillion ETL. You will learn the following topics:

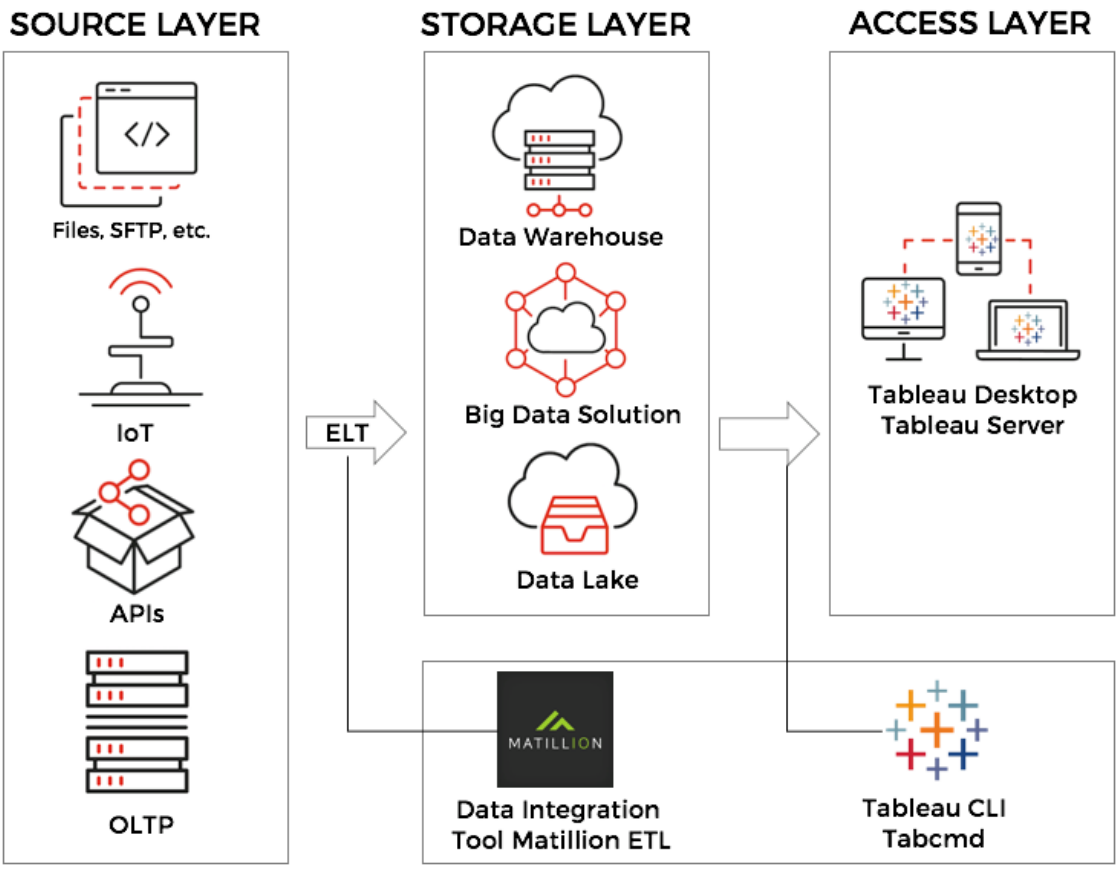
- How to install Matillion ETL
- How to install Tabcmd for Linux
- How to create custom Tableau component for Matillion

Technical requirements

You may use Redshift or Snowflake from *Tableau for Big Data*. We also need Matillion ETL from the AWS Marketplace.

Getting started with Matillion ETL

According to best practices, we should integrate our data pipelines with Tableau Server using `tabcmd`. The following diagram will show us the common architecture of a modern data warehouse project:



We will install the Linux version of Tabcmd on a Matillion EC2 instance and integrate with Tableau Server. We should be sure that our ETL and Tableau Server can talk with each other. In our case, we will use the same AWS account and region in order to use the same private network. In case you have Tableau Server on-premise, you should configure your firewall. We choose Matillion ETL because this is the leader among modern cloud ETL solutions and works with Redshift, Snowflake, and BigQuery.

How to do it...

Since we have already trialed Snowflake and Redshift, we should get Matillion ETL. In our example we will use Matillion ETL for Redshift. We will use free trial and get it from the AWS Market Place. There is detail information on how to launch Matillion for Redshift at:

https://redshiftsupport.matillion.com/customer/en/portal/articles/2487672-launching-detailed-instructions.

Moreover, you may contact Matillion support and they will help you. They have awesome support and are happy to give a hand. We won't get into details about how to install Matillion ETL because it is quite simple and straightforward.

How it works...

By the end of the preparation, you should have running an Matillion ETL EC2 instance within your account. In addition, you should configure it to the Redshift instance, which we used in *Tableau for Big Data*. Tableau will connect to the Redshift. We will use `tabcmd` for the following tasks:

- Triggering a refresh of Tableau Extract at the end of ETL

- Triggering an export of Tableau Workbook in order to generate a PDF and export to S3

There's more...

In our case, we were using Matillion ETL for Redshift, but you may use Matillion ETL for Snowflake as well as for Google BigQuery. You might apply these principles to any of data platforms.

Deploying Tabcmd on Linux

Since Tableau released Tableau on Linux, we don't need to spend any more time on converting Windows Tabcmd for Linux. We can simply download `tabcmd` from Tableau website and install it to the Linux box with Matillion ETL. We are going to use 2018.2 version for this recipe but you can use 2019.x `tabcmd` distributive.

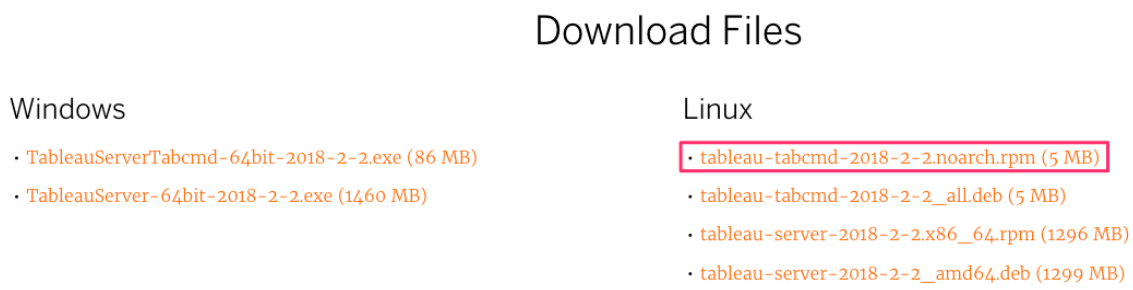
Note

It is important that the `tabcmd` version was aligned with Tableau Server version. For example, if you have Tableau Server 2019.1 then you need to download Tabcmd 2019.1.

How to do it..

Let's download `tabcmd` and install it at Linux box:

1. First, we will go to the Tableau Releases website (https://www.tableau.com/support/releases) and download Tabcmd for Linux. We should download the same version as our Tableau Server. In our case, it is 2018.2, as shown in the following screenshot:



2. We will download RMP archive because Amazon Linux has lots in common with Red Hat.
3. Then we should, upload this into the EC2 instance with Matillion. There are multiple ways to do this. For example, the fastest way for us is to use AWS CLI S3. We will upload the file into the S3 bucket and then download it from EC2 instance.
4. Next, we should install this archive on the EC2. Go to the location of archive and execute the following command or specify the full path to the file:

```
sudo rpm -Uvh tableau-tabcmd-2018-2-2.noarch.rpm
```

5. As a result, we will have to install Tabcmd for our Linux system. Now, we want to make sure, that everything works as expected.

Note

It is important that Tableau and Matillion can see each other from the network point of view. We recommend that you deploy your data analytics solution using the same AWS account and the same region. In case you have to use other topology, you might need to configure access.

6. In order to test, we can do the following commands---login to Tableau Server and trigger extract. Also, you might to execute any other `tabcmd` command, as follows:

```
#matillion is running under tomcat user and we will switch to this user
sudo -su tomcat
#go to tabcmd location
cd /opt/tableau/tabcmd/bin
#login tableau server
./tabcmd  login -u Admin -p 'p@ssword' -s https://myserver:443 --no-certcheck  --
accepteula
#refresh extract
./tabcmd refreshextracts --datasource "My Sexy Data Source" --project "My project" --
no-certcheck -synchronous
```

In the preceding command lines, we are using the following Tabcmd parameters:

- `--no-certcheck` : We need this in case of SSL.
- `--accepteula` : This is new parameter, that was introduced recently.
- `-u` : Tableau user name that has permissions to perform desired action.
- `-p` : This parameter represents the password.
- `-s` : Tableau host or load balancer endpoint.
- `--datasource` : Tableau data source.
- `--project` : Project where the data source is stored.
- `--synchronous` : This parameter will await feedback from the Tableau Server about the end of the Tableau Extract refresh. This allows us to execute jobs in chain. As a result, we can trigger Tableau from Matillion EC2. We can even copy this logic into the Matillion Bash component, but it will be hard for business users to go through it and self-serve.

How it works...

Tabcmd is a command-line utility that automates site administration tasks on your Tableau Server site. In our case, we used the functionality of `tabcmd` to trigger Tableau Extracts.

There's more...

You can find the `tabcmd` list of commands at Tableau official documentation at https://onlinehelp.tableau.com/current/server/en-us/tabcmd_cmd.htm.

Creating Matillion Shared Jobs

In order to simplify the job of end users, we will leverage Matillion Shared Job (https://redshiftsupport.matillion.com/customer/portal/articles/2942889-shared-jobs) and Matillion variables (https://redshiftsupport.matillion.com/customer/portal/articles/2037630-using-variables).

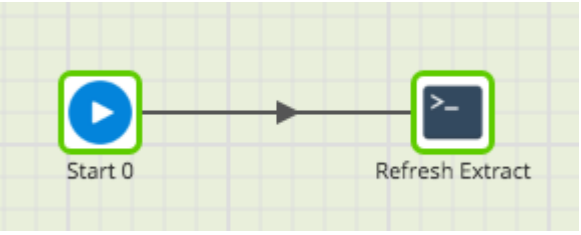
The main purpose of Shared Jobs is to bundle entire workflows into a single custom component. We will create the following two custom components:

- Refresh Tableau Extract
- Export PDF dashboard to S3 bucket

How to do it..

Before we start, we should create a new Orchestration Job for each use case and then we can insert Matillion variables and create Shared Jobs.

1. Create new Orchestration Job with bash component and name it `Refresh Extract`, as follows



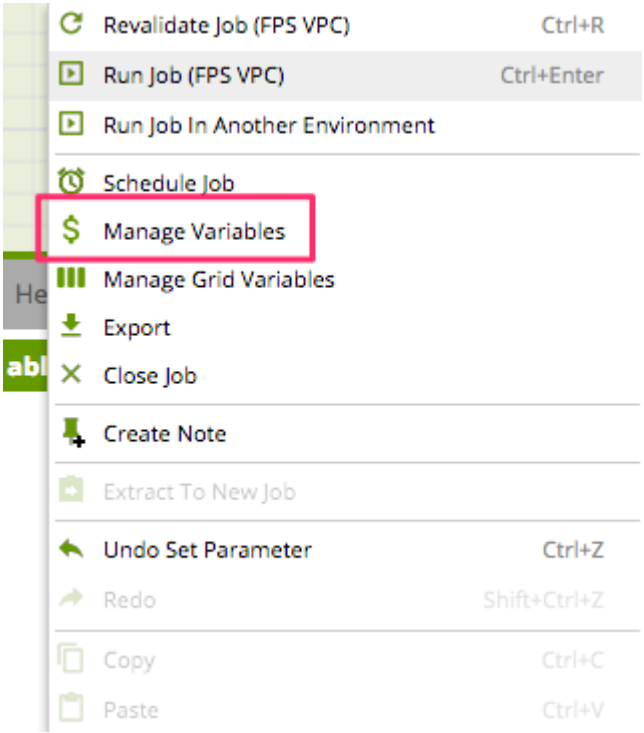
Bash component has the `timeout` parameter. By default, it is 1,000 seconds. For extract, we might increase this in order to wait while your biggest extract refresh.

2. Then paste the code that we tested already and replace Tableau objects with the Matillion parameters, as follows:

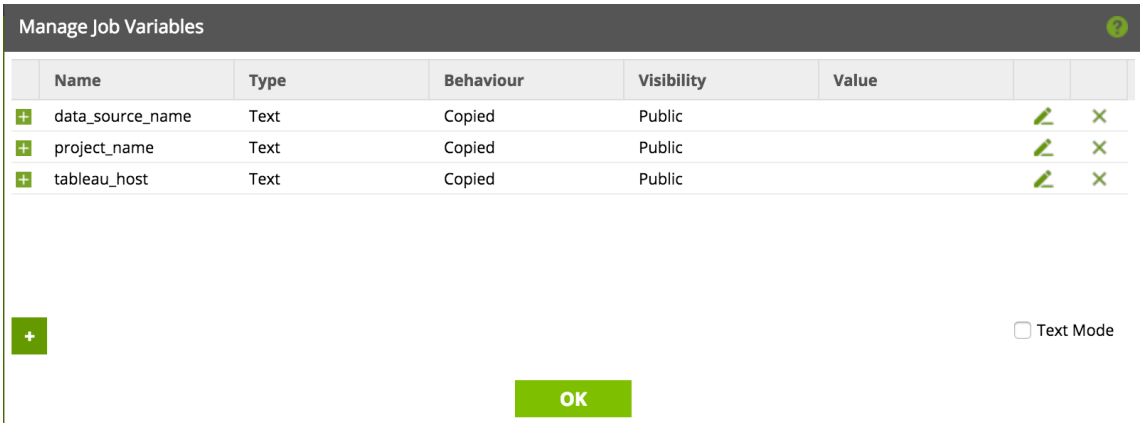
```
#go to tabcmd location
cd /opt/tableau/tabcmd/bin
#login tableau server
./tabcmd  login -u Admin -p '${password}' -s ${tableau_host} --no-certcheck  --
accepteula
#refresh extract
./tabcmd refreshextracts --datasource "${data_source_name}" --project
"${project_name}" --no-certcheck --synchronous
```

As a result, this component will refresh Tableau Extract based on value for variable.

3. In addition, we should create Matillion variable for our parameter. Click on the right button on canvas and choose **`Manage Variables`** as shown in the following screenshot:



4. Then add three new variables that have to be public, as follows:



5. Create one more job or duplicate an existing one and name it `Tableau Export PDF`. Enter the following code with the Matillion parameters:

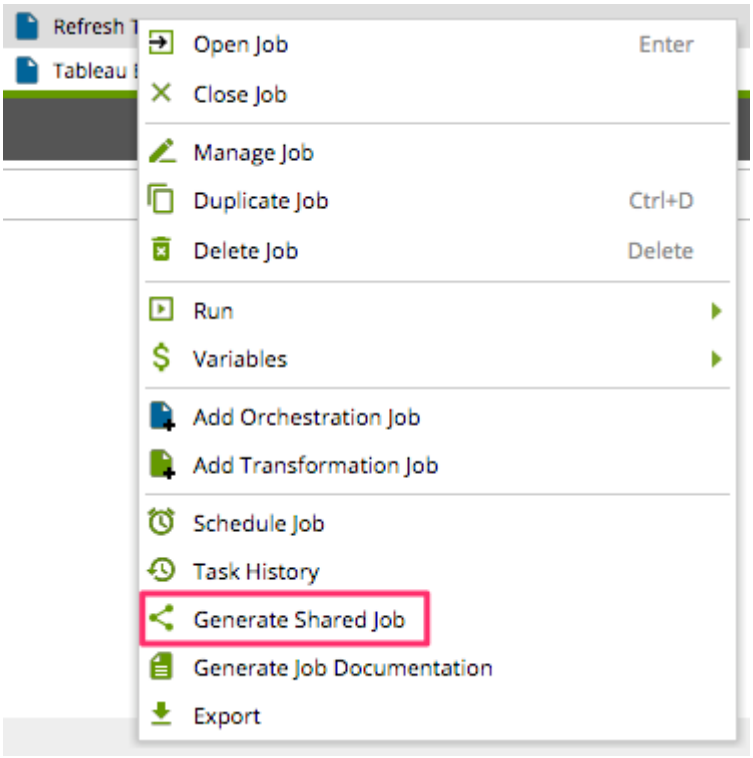
```
#go to tabcmd location
cd /opt/tableau/tabcmd/bin
#login tableau server
./tabcmd  login -u Admin -p 'p@ssword' -s ${tableau_host} --no-certcheck  --accepteula
#export pdf from Tabelau Server
./tabcmd export "${tableau_view_name}" --pdf --pagelayout landscape -f "/tmp/$(date
+%Y%m%d)_${tableau_report_name}.pdf" --no-certcheck
#upload pdf to the S3
aws s3 cp /tmp/$(date +%Y%m%d)_${tableau_report_name}.pdf s3://${bucket_name}/$(date
+%Y%m%d)/$(date +%Y%m%d)_${tableau_report_name}.pdf
#clean out
rm /tmp/$(date +%Y%m%d)_${tableau_report_name}.pdf
```

This script will export Tableau View into the `/tmp` location on our EC2 and then will upload to the Reporting Bucket via AWS CLI. Moreover, it will automatically create a folder in bucket with date. In addition, we specify the file name according with our naming convention.

6. Moreover, you should create the following variables in the same way as in step #1:

- `tableau_report_name`
- `tableau_view_name`
- `tableau_host`
- `bucket_name` You can see how this solution is flexible and you can achieve many different use cases.

7. Now, we can create the Shared Jobs and wrap our Orchestration Jobs. Click on the right button on job name and choose **`Generate Shared Job`** , as follows:



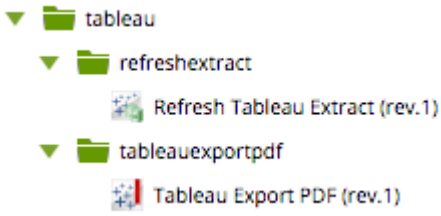
8. Then, we should fill in the form and choose the following mentioned options:

`Package`	`Packt.tableau.refreshextract`
`Name`	`Refresh Tableau Extract`
`Description`	`This component refresh Tableau Extract`

9. Click on **`Next`** and you will see the **`Parameter Configuration`** step. Then, click on **`OK`**.
10. Then, do the same for the second job **`Tableau Export to PDF`**, as follows:

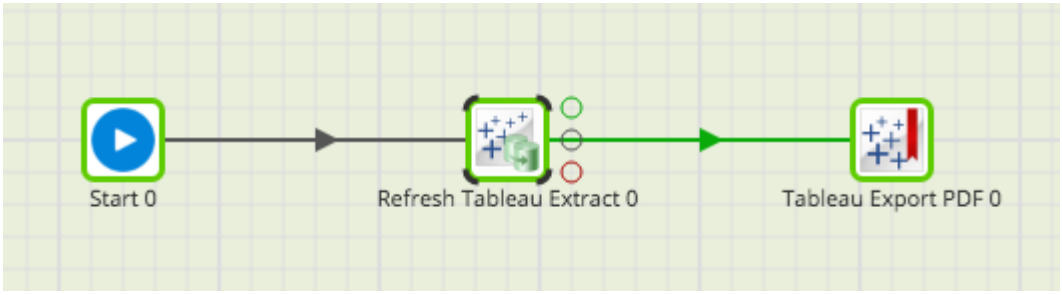
`Package`	`Packt.tableau.exporttopdf`
`Name`	`Tableau Export to PDF`
`Description`	`This component will export PDF report to S3 Bucket`

11. Click on **`Next`**, fill in the **`Parameter Configuration`** page and click on **`OK`**. These parameters will be used for the data entry later.
12. Let's check out the jobs. Navigate to **`Shared Jobs Pane`** | **`User Defined`** | **`Fenago`**, as follows:



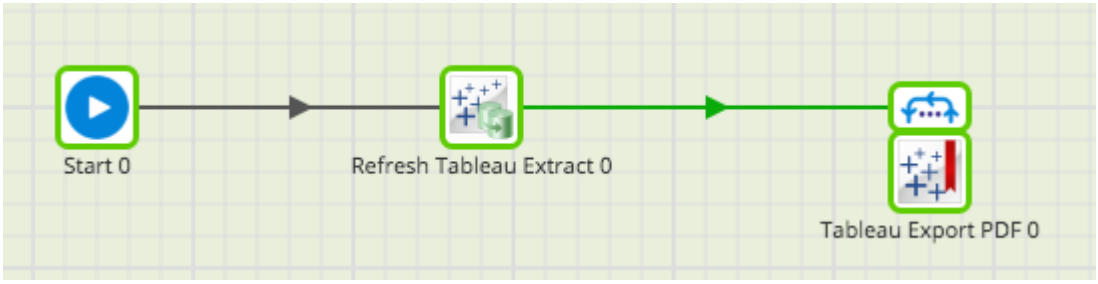
As a result, you'll see our new jobs.

13. Let's put it all together. Create a new Orchestration Job and drag and drop our new shared components, as follows:



Usually, one extract can source many different workbooks. As a result, we can use another powerful feature of [*Matillion---Fixed Iterator.*] More information can be found here: https://redshiftsupport.matillion.com/customer/en/portal/articles/2235536-fixed-iterator.

14. Let's add a fixed iterator on top of the job as it will allow us to specify multiple reports at once, as follows:



As a result, we have created new custom components that look very friendly and familiar for our end users. Using this approach, we can leverage any Tabcmd command and create a custom component for it.

How it works...

We used Tabcmd to manage Tableau Server and integrate it with data. Adding the Linux Tableau CLI `tabcmd` tool to EC2 Linux allows us to use Matillion components to plug Tableau into the data pipeline. And in case of a successful execution of ETL, we can easily trigger Tableau Extracts or perform any other actions.