Architecture Design

# Amazon Sales Data Analysis

|  |  |
| --- | --- |
| **Written By** | Sanjay B.Y ,  Pavan Kumar Shetty  Umashankar K.H  Revan Siddha |
| **Document Version** | 0.1 |
| **Last Revised Date** |  |

**DOCUMENT CONTROL**

## Change Record:

|  |  |  |  |
| --- | --- | --- | --- |
| **VERSION** | **DATE** | **AUTHOR** | **COMMENTS** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Reviews:**

|  |  |  |  |
| --- | --- | --- | --- |
| **VERSION** | **DATE** | **REVIEWER** | **COMMENTS** |
|  |  |  |  |

**Approval Status:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **VERSION** | **REVIEW**  **DATE** | **REVIEWED BY** |  | **APPROVED BY** | **COMMENTS** |
|  |  |  |  |  |  |

# Contents

1. [Introduction 04](#_TOC_250005)
   1. [What is Architecture Design Document? 04](#_TOC_250004)
   2. [Scope 04](#_TOC_250003)
2. [Architecture 05](#_TOC_250002)
   1. Tableau Architecture 05
   2. Tableau Server Architecture 05
   3. Gateway/Load Balancer 06
   4. Application Server 06
   5. VIZQL Server 07
   6. Data Engine 07
   7. Backgrounder 07
   8. Data Server 07
   9. Tableau Communication Flow 07
3. Deployment 08
   1. [Deployment Options in Tableau 09](#_TOC_250001)
4. Python………………………………………………………………………………………………………………….10
   1. Python Architecture………………………………………………………………………………….11
5. Matplotlib……………………………………………………………………………………………………………..12
   1. Data Visualization……………………………………………………………………………………..13
   2. Matplotlib Architecture…………………………………………………………………………….14

# Introduction

## What is Architecture design document?

Any software needs the architectural design to represents the design of software. IEEE defines architectural design as “the process of defining a collection of hardware and software components and their interfaces to establish the framework for the development of a computer system.” The software that is built for computer-based systems can exhibit one of these many architectures.

Each style will describe a system category that consists of :

* A set of components (eg: a database, computational modules) that will perform a function required by the system.
* The set of connectors will help in coordination, communication, and cooperation between the components.
* Conditions that how components can be integrated to form the system.
* Semantic models that help the designer to understand the overall properties of the system.

## Scope

Architecture Design Document (ADD) is an architecture design process that follows a step-by-step refinement process. The process can be used for designing data structures, required software architecture, source code and ultimately, performance algorithms. Overall, the design principles may be defined during requirement analysis and then refined during architectural design work.

# Architecture



**Tableau Server Architecture**

Tableau has a highly scalable, n-tier client-server architecture that serves mobile clients, web clients and desktop-installed software. Tableau Server architecture supports fast and flexible deployments.

### ARCHITECTURE DESIGN

**6**

The following diagram shows Tableau Server’s architecture:

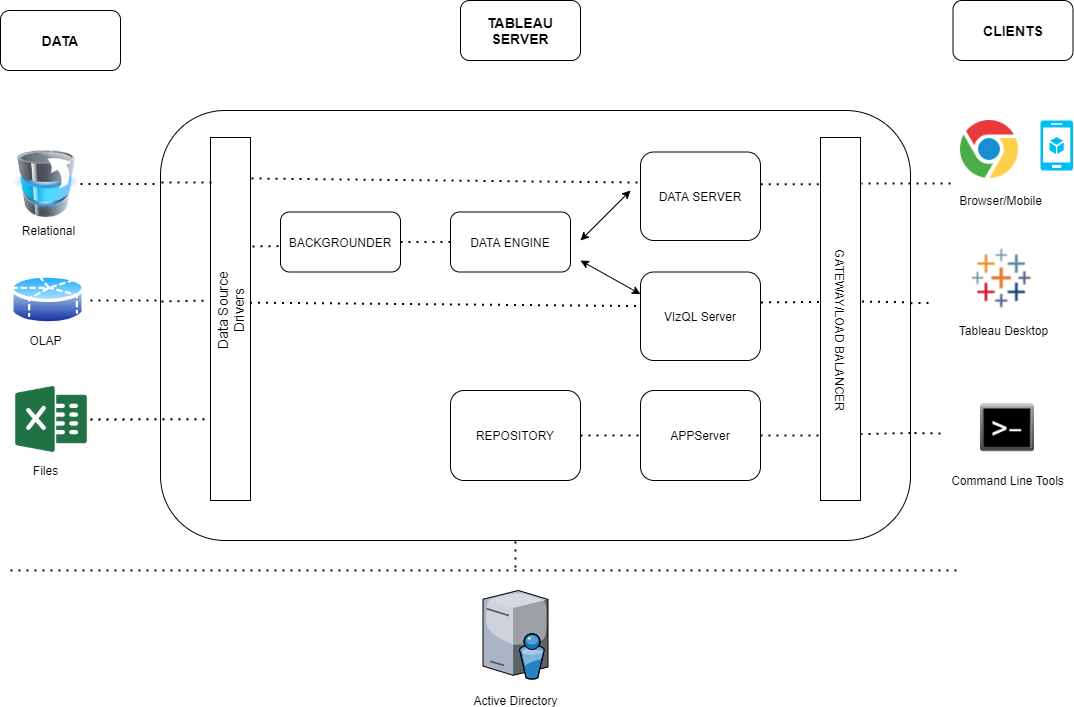


Tableau Server is internally managed by the multiple server processes.

**1. Gateway/Load Balancer**

It acts as an Entry gate to the Tableau Server and also balances the load to the Server if multiple Processes are configured.

**2) Application Server:-**

Application Server processes (wgserver.exe) handle browsing and permissions for the Tableau Server web and mobile interfaces. When a user opens a view in a client device, that user starts a session on Tableau Server. This means that an Application Server thread starts and checks the permissions for that user and that view.

1. **Repository:-**

Tableau Server Repository is a PostgreSQL database that stores server data. This data includes information about Tableau Server users, groups and group assignments, permissions, projects, data sources, and extract metadata and refresh information.

1. **VIZQL Server:-**

Once a view is opened, the client sends a request to the VizQL process (vizqlserver.exe). The VizQL process then sends queries directly to the data source, returning a result set that is rendered as images and presented to the user. Each VizQL Server has its own cache that can be shared across multiple users

1. **Data Engine:-**

It Stores data extracts and answers queries.

1. **Backgrounder:-**

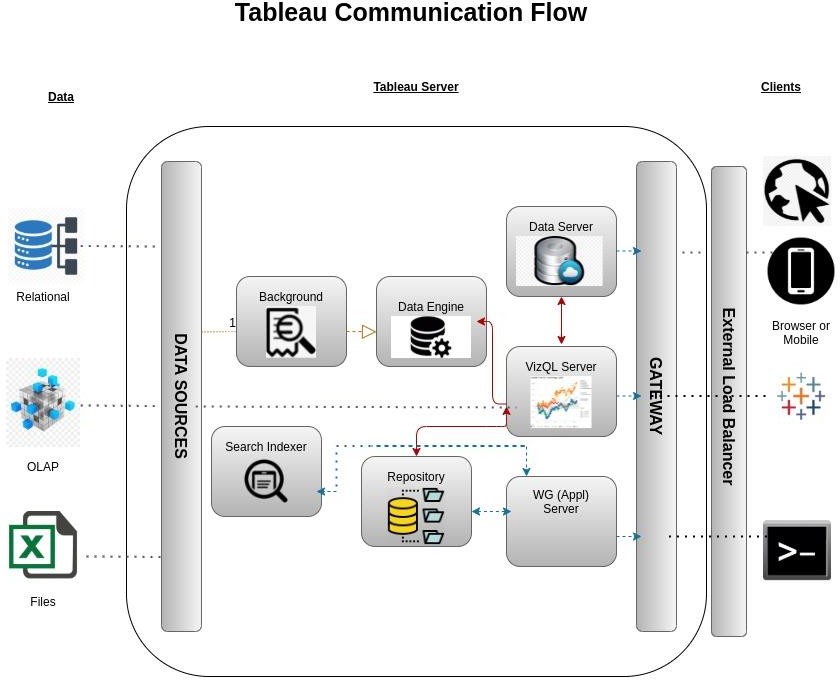
The backgrounder Executes server tasks which includes refreshes scheduled extracts, tasks initiated from tabcmd and manages other background tasks.

1. **Data Server:-**

Data Server Manages connections to Tableau Server data sources

It also maintains metadata from Tableau Desktop, such as calculations, definitions, and groups.

**8) Tableau Communication Flow**

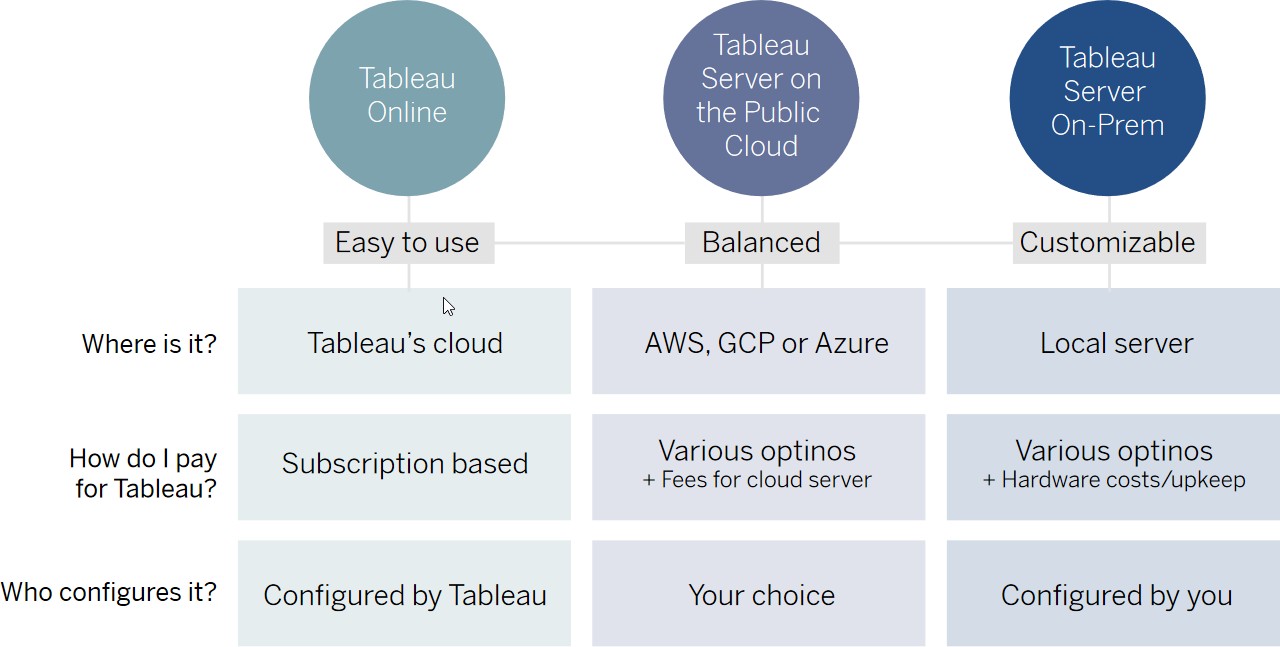


# Deployment Description

## Deployment options in Tableau

Tableau’s analytics platform offers three different deployment options depending on your

environment and needs. The below graphic shows each option at a glance:



1. **Tableau Online** Get up and running quickly with no hardware required. Tableau Online is fully hosted by Tableau so all upgrades and maintenance are automatically managed for you.
2. **Tableau Server** deployed on public cloud: Leverage the flexibility and scalability of cloud infrastructure without giving up control. Deploy to Amazon Web Services, Google Cloud Platform, or Microsoft Azure infrastructure to quickly get started with Tableau Server (on your choice of Windows or Linux). Bring your own license or purchase on your preferred marketplace.
3. **Tableau Server deployed on-premises**: Manage and scale your own hardware and software (whether Windows or Linux) as needed. Customize your deployment as you see fit.

## Single Node Architecture



This architecture is a single node architecture. This is the most simple deployment topology.

1. **Python Architecture**

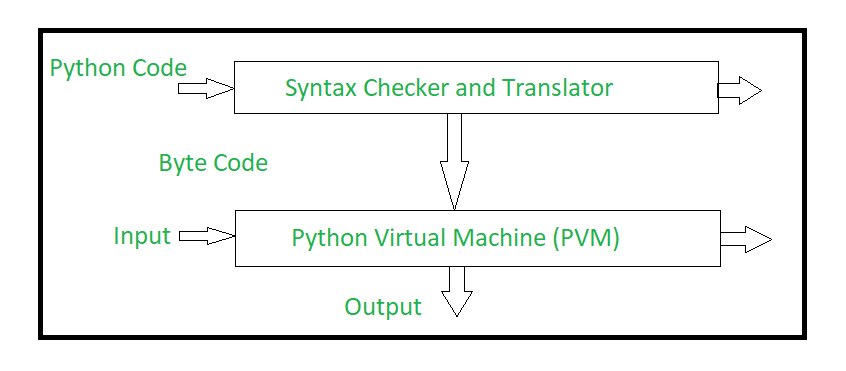
Python is an object-oriented programming language like Java. Python is called an interpreted language.

Python uses code modules that are interchangeable instead of a single long list of instructions that was standard for functional programming languages.The standard implementation of python is called “cpython”. It is the default and widely used implementation of Python.

Python doesn’t convert its code into machine code, something that hardware can understand.

It actually converts it into something called byte code. So within python, compilation happens, but it’s just not into a machine language.

It is into byte code (.pyc or .pyo) and this byte code can’t be understood by the CPU. So we need an interpreter called the python virtual machine to execute the byte codes..



The Python source code goes through the following to generate an executable code :

Step 1: The python compiler reads a python source code or instruction. Then it verifies that the instruction is well-formatted, i.e. it checks the syntax of each line. If it encounters an error, it immediately halts the translation and shows an error message.

Step 2: If there is no error, i.e. if the python instruction or source code is well-formatted then the compiler translates it into its equivalent form in an intermediate language called “Byte code”.

Step 3: Byte code is then sent to the Python Virtual Machine (PVM) which is the python interpreter. PVM converts the python byte code into machine-executable code. If an error occurs during this interpretation then the conversion is halted with an error message.

5 Matplotlib Architecture

[**Data visualization**](https://en.wikipedia.org/wiki/Data_visualization) is the graphic representation of the data.

It is a way of showing complex data in a graphical and easy to understand way, which is very useful when trying to explain the insight obtained from the analysis of increasingly large datasets.

Plots and graphs can be very effective in conveying a clear description of the data especially when disclosing findings to an audience or sharing the data with other peer data scientists.

Also, they can be very valuable when it comes to supporting decision making.

Data visualization involves specific terminology, some of which are derived from statistics. For example, author [Stephen Few](http://www.perceptualedge.com/about.php) defines two types of data, which are used in combination to support a meaningful analysis or visualization:

Categorical: Text labels describing the nature of the data, such as “Name” or “Age”. This term also covers qualitative (non-numerical) data.

Quantitative: Numerical measures, such as “25” to represent the age in years.

Two primary types of information displays are tables and graphs.

A table contains quantitative data organized into rows and columns with categorical labels. It is primarily used to look up specific values. In the example above, the table might have categorical column labels representing the name (a qualitative variable) and age (a quantitative variable), with each row of data representing one person (the sampled experimental unit or category subdivision).

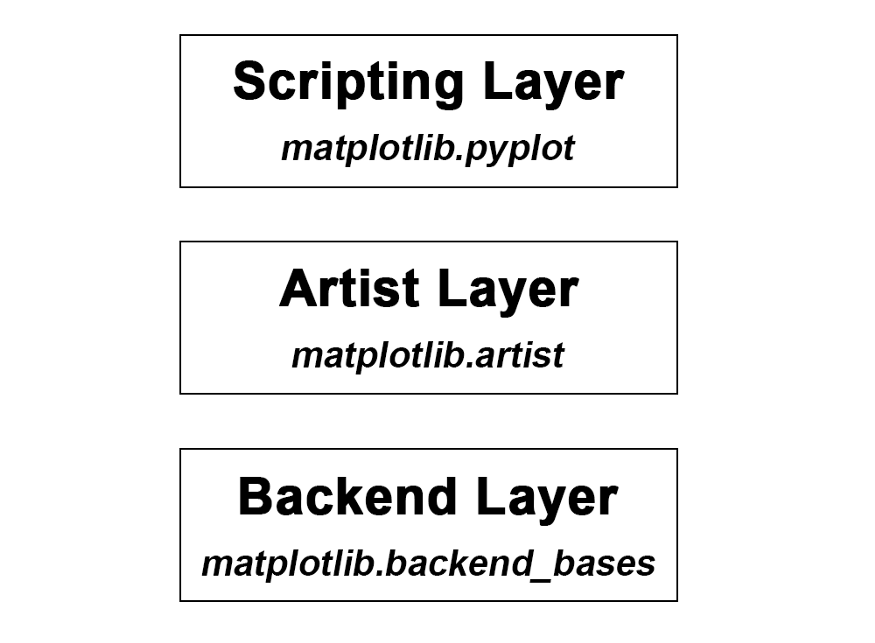
A graph is primarily used to show relationships among data and portrays values encoded as visual objects (e.g., lines, bars, or points). Numerical values are displayed within an area delineated by one or more axes. These axes provide scales (quantitative and categorical) used to label and assign values to the visual objects. Many graphs are also referred to as charts.

**What is Matplotlib**



[Matplotlib](https://matplotlib.org/) is the oldest Python plotting library, and it’s still the most popular.It was created in 2003 as part of the [SciPy Stack](https://www.scipy.org/about.html), an open-source scientific computing library similar to [Matlab](https://www.mathworks.com/products/matlab.html).Matplotlib is designed with the philosophy that you should be able to create simple plots with just a few commands.

Matplotlib Architecture



The Matplotlib architecture is composed of three main layers:

* **Backend Layer**— Handles all the heavy works via communicating to the drawing toolkits in your machine. It is the most complex layer.
* **Artist Layer** — Allows full control and fine-tuning of the Matplotlibfigure — the top-level container for all plot elements.
* **Scripting Layer** — The lightest scripting interface among the three layers, designed to make Matplotlib work like MATLAB script.

## Backend Layer

## The backend layer handles all the heavy works via communicating to the toolkits like [wxPython](https://wxpython.org/" \t "_blank) or drawing languages like [PostScript](https://en.wikipedia.org/wiki/PostScript) in your machine. It is the most complex layer of the Matplotlib library.

It has three main built-in abstract interface classes:

* **FigureCanvas —**[matplotlib.backend\_bases.FigureCanvasBase](https://matplotlib.org/api/backend_bases_api.html#matplotlib.backend_bases.FigureCanvasBase)The canvas the figure renders into.
* **Renderer —**[matplotlib.backend\_bases.RendererBase](https://matplotlib.org/api/backend_bases_api.html" \l "matplotlib.backend_bases.RendererBase" \t "_blank)An abstract base class to handle drawing/rendering operations. Responsible to draw in the FigureCanvas.
* **Event —**[matplotlib.backend\_bases.Event](https://matplotlib.org/api/backend_bases_api.html" \l "matplotlib.backend_bases.RendererBase" \t "_blank) Handles user inputs such as keyboard and mouse clicks.

A regular user hardly has to deal with this layer, which is why we won’t go into more detail.

## Artist Layer

It allows control and fine-tune as many elements as possible in the figure just like an artist paints on the canvas. This layer is comprised of one main object, that uses the Renderer to draw on the canvas. It allows you to do more customization compare to the **Scripting layer** and is more convenient for advanced plots. Especially when handling multiple figures/axes, you will not get confused as to which one is currently active since every subplot is assigned to an object.That’s why the Artist Layer is sometimes called **object-based plotting**. It is used more often when writing a web application, or a UI application, or perhaps a script to be shared with other developers.

Everything visible on a Matplotlib figure is an artist instance: The title, the lines, the tick labels, the images, and so on, all correspond to an individual artist.

There are two types of Artist objects. The first type is the **primitive type**, such as a [Line2D](https://matplotlib.org/api/_as_gen/matplotlib.lines.Line2D.html), [Rectangle](https://matplotlib.org/api/_as_gen/matplotlib.patches.Rectangle.html), [Circle](https://matplotlib.org/api/_as_gen/matplotlib.patches.Circle.html), and [Text](https://matplotlib.org/api/text_api.html). And the second type is the **composite type**, such as the [Axis](https://matplotlib.org/api/axis_api.html?highlight=axis#module-matplotlib.axis), [Tick](https://matplotlib.org/api/axis_api.html#tick-objects), [Axes](https://matplotlib.org/api/axes_api.html#the-axes-class) and [Figure](https://matplotlib.org/api/_as_gen/matplotlib.figure.Figure.html#matplotlib.figure.Figure).

It is important to notice that each composite artist may contain other composite artists as well as primitive artists. For example a figure artist can contain an axis artist as well as a rectangle or text artist.

## Scripting Layer

It is the top layer, designed to make Matplotlib work like MATLAB script. It is a collection of command style functions and is therefore considered the easiest layer to use.

The artistic layer is syntactically heavy as it is meant for developers and not for individuals whose goal is to perform quick exploratory analysis of some data.

This is why many Matplotlib tutorials prefer to introduce from this layer. It is the easiest part to start with and use, you add up objects — e.g. line, text, rectangle- on top of the figure.

Scripting layer plotting is sometimes also called **procedural plotting**.

Matplotlib’s scripting layer is essentially the matplotlib.pyplot interface, which automates the process of defining a canvas and defining a figure artist instance and connecting them.