Architecture Design

Amazon Sales Analysis

Written By	Freeva Dsouza
Document Version	0.2
Last Revised Date	15th February 2022

DOCUMENT CONTROL

Change Record:

VERSION	DATE	AUTHOR	COMMENTS
0.1	23rd January 2022	Freeva Dsouza	Introduction and architecture defined
0.2	15th February 2022	Freeva Dsouza	Architecture & Architecture description appended and Updated.

Contents

1. Introduction

1.1 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 :

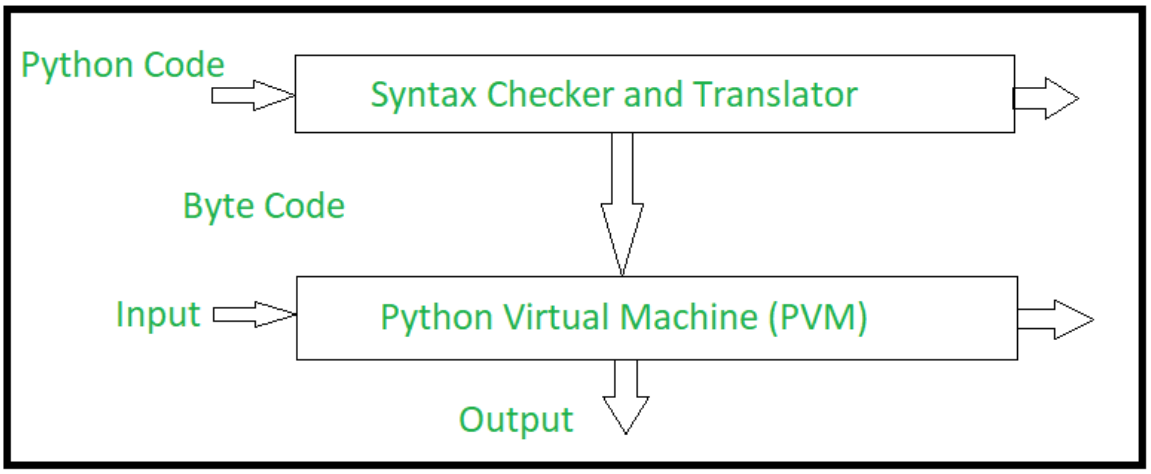
- A set of components (eg: a database, computational modules) that will perform a function required by the system.
- The set of connectors will help in coordination, communication, and cooperation between the components.
- Conditions that how components can be integrated to form the system.
- Semantic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

1.2 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.

2. Architecture

2.1 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 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.

3. Jupyter Notebook Architecture

3.1 Introduction

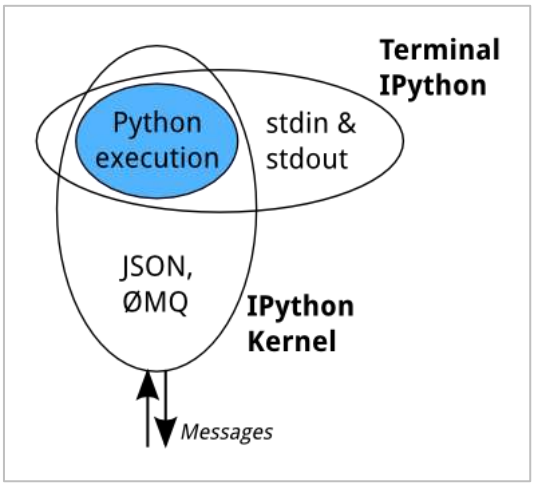
The notebook extends the console-based approach to interactive computing in a qualitatively new direction, providing a web-based application suitable for capturing the whole computation process: developing, documenting, and executing code, as well as communicating the results. The Jupyter notebook combines two components:

A web application: a browser-based tool for interactive authoring of documents which combine explanatory text, mathematics, computations and their rich media output.
Notebook documents: a representation of all content visible in the web application, including inputs and outputs of the computations, explanatory text, mathematics, images, and rich media representations of objects.

3.2 Ipython Kernel

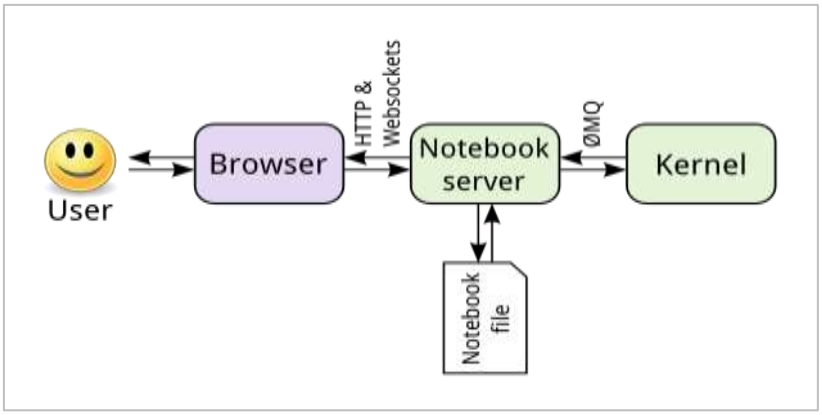
The IPython Kernel is a separate process which is responsible for running user code, and things like computing possible completions. Frontends, like the notebook or the Qt console, communicate with the IPython Kernel using JSON messages sent over ZeroMQ sockets; the protocol used between the frontends and the IPython Kernel is described in Messaging in Jupyter.

The core execution machinery for the kernel is shared with terminal IPython:



3.3 Jupyter Notebook Interface

Jupyter Notebook and its flexible interface extend the notebook beyond code to visualization, multimedia, collaboration, and more. In addition to running your code, it stores code and output, together with markdown notes, in an editable document called a notebook. When you save it, this is sent from your browser to the notebook server, which saves it on disk as a JSON file with a .ipynb extension.



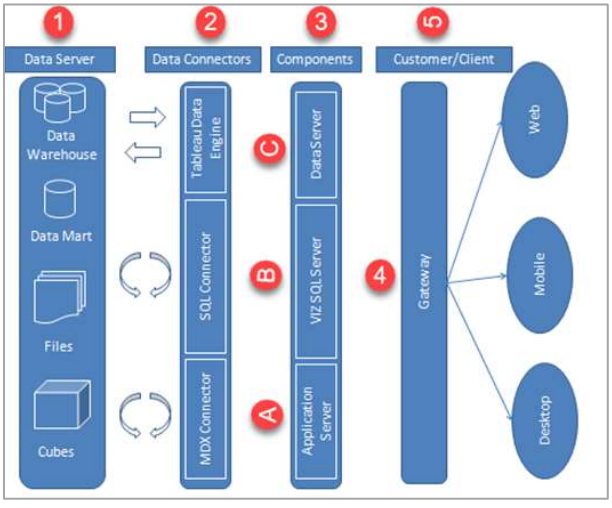
The notebook server, not the kernel, is responsible for saving and loading notebooks, so you can edit notebooks even if you don't have the kernel for that language—you just won't be able to run code. The kernel doesn't know anything about the notebook document: it just gets sent cells of code to execute when the user runs them

4. Tableau Architecture

4.1 Introduction

Tableau Server is designed in a way to connect many data tiers. It can connect clients from desktop, mobile, and web. Tableau Desktop is a robust data visualization tool. It is highly available and secure. It can run on both virtual and physical machines. It is a multi-user, multi-process and multi-threaded system.

4.2 Tableau Server Architecture



Data Server: Tableau can connect to multiple data sources. These data sources can be on-premise or remotely located. It can connect to a database; excel file, and a web application all at the same time. Tableau can connect data from heterogeneous environments. It can blend the data from multiple data sources. It can also make the relationship between various types of data sources.

Data Connectors: The Data Connectors provide an interface to connect external data sources to Tableau Data Server. Tableau has in-built ODBC/SQL connector. This ODBC Connector can connect to any databases without using their native connector. Tableau has an option to select both live and extract data. Based on the usage, one can be easily switched between extracted and live data.

Data Connectors: The Data Connectors provide an interface to connect external data sources to Tableau Data Server. Tableau has in-built ODBC/SQL connector. This ODBC Connector can connect to any databases without using their native connector. Tableau has an option to select both live and extract data. Based on the usage, one can be easily switched between extracted and live data.

Components of Tableau Server:

A) Application Server:

The application server is used to provide the authentications and authorizations. It handles the administration and permission for web and mobile interfaces. It assures security by recording each session id on Tableau Server. The administrator can configure the default timeout of the session in the server.

B) VizQL Server:

VizQL server is used to convert the queries from the data source into visualizations. Once the client request is forwarded to VizQL process, it sends the query directly to data source and retrieves information in the form of images. This image or visualization is presented to the user. Tableau server creates a cache of visualization to reduce the load time. The cache can be shared across many users who have the permission to view the visualization.

C) Data Server:

Data server is used to manage and store the data from external data sources. It is a central data management system. It provides metadata management, data security, data storage, data connection and driver requirements. It stores the relevant details of data set such as metadata, calculated fields, sets, groups, and parameters. The data source could extract data as well make live connections to external data sources.

Gateway:

The gateway channelizes the requests from users to Tableau components. When the client makes a request, it is forwarded to external load balancer for processing. The gateway works as a distributor of processes to various components. In case of absence of external load balancer, gateway also works as a load balancer. For single server configuration, one primary server or gateway manages all the processes. For multiple server configurations, one physical system works as primary server while others are used as worker servers. Only one machine can be used as a primary server in Tableau Server environment.

Clients:

The dashboards and visualizations in Tableau server can be viewed and edited using different clients. The Clients are Tableau Desktop, web browser and mobile applications.