

Architecture Design

NBA Data Analysis

**Written By** Author 1

**Document -**v**ersion** 0.1

**Last Revised Date**

Vamshi Chowdary



**DOCUMENT CONTROL**

**Change Record:**

**VERSION DATE AUTHOR COMMENTS**

0.1 03-july-2023 Author 1 Introduction and Architecture Defined

**Approval Status:**

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



**Contents**

**1. Introduction………………………………………………………………………...04**

**1.1 What is Architecture Design Document? … ……….…….…………. 04**

**1.2 Scope………….…………………………………………………………... 04**

**2. Architecture …………………………………………………………………….. 05**

**2.1 Plotly Server Architecture ……………………………………………. 05**

**2.2 Plotly Express Architecture …………………………………………. 08**

**3. Deployment….………………………………………………………………………10**

**3.1 Deployment options for Plotly ……………………..………………..…………………….10**

**3.2 Deployment using Heroku…………………………………………………………………….10**



**1. Introduction**

**1.1 What is an Architecture design document?**

Any software needs an architectural design to represent the design of the 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.

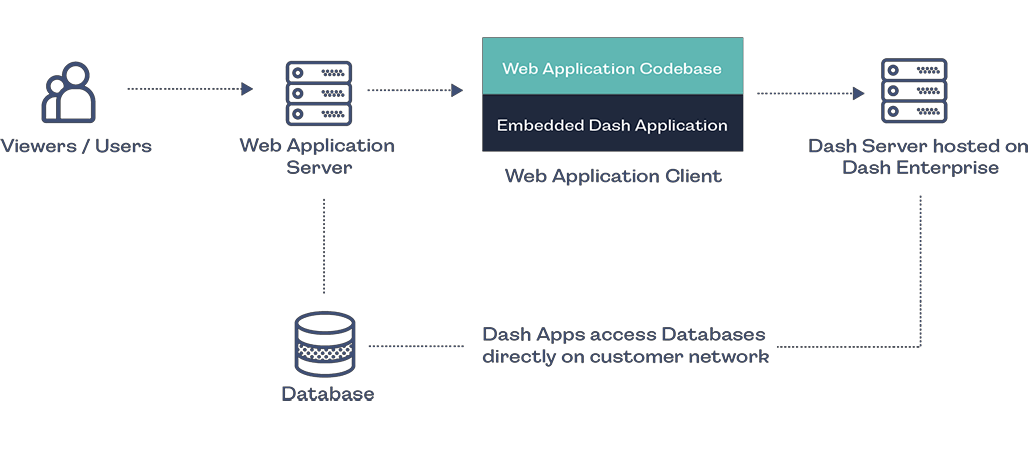
**1.2 Scope**



Architecture Design Document (ADD) is an architectural 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**

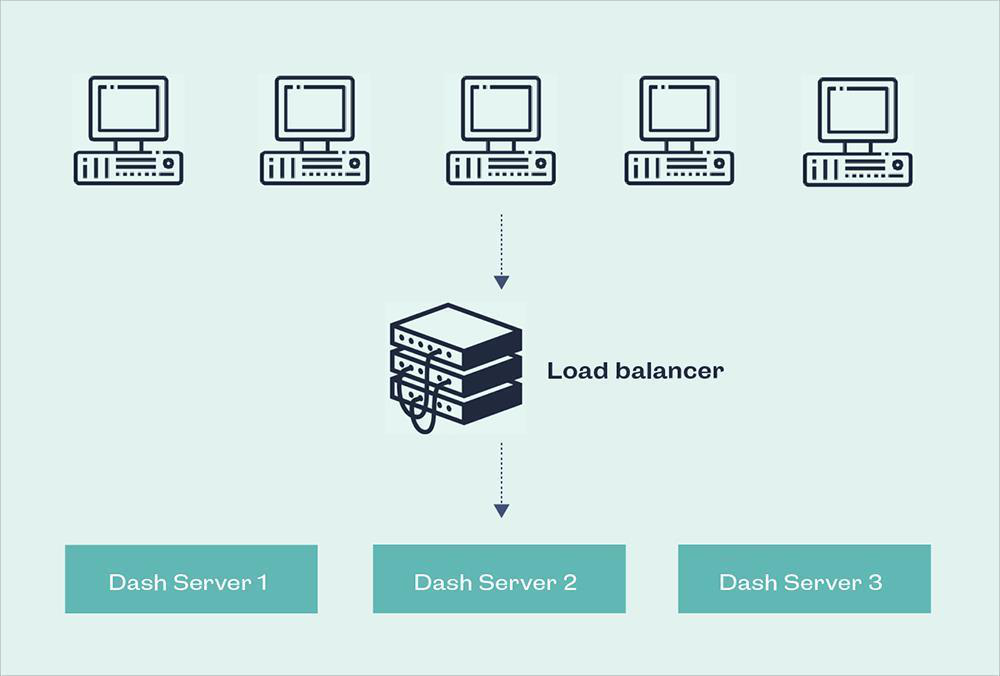
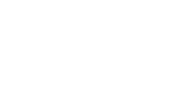


**2.1 Plotly Server Architecture**

**The following diagram shows the Plotly server architecture**



**Excel files CSV files Other document files**



**1. Gateway/Load Balancer**



It acts as an Entry gate to the Plotly Server and also balances the load to the

Server if multiple Processes are configured.

**2 Application Server**

Application Server processes handle browsing and permissions for the Plotly Server web and mobile interfaces. When a user opens a view in a client device, that user starts a session on Plotly Server. This means that an

The application Server thread starts and checks the permissions for that user and that view.

**3 Repository**

Plotly Server Repository is a Dash database that stores server data.

Dash is ideal for building and deploying data apps with customized user interfaces. It’s particularly suited for anyone who works with data.

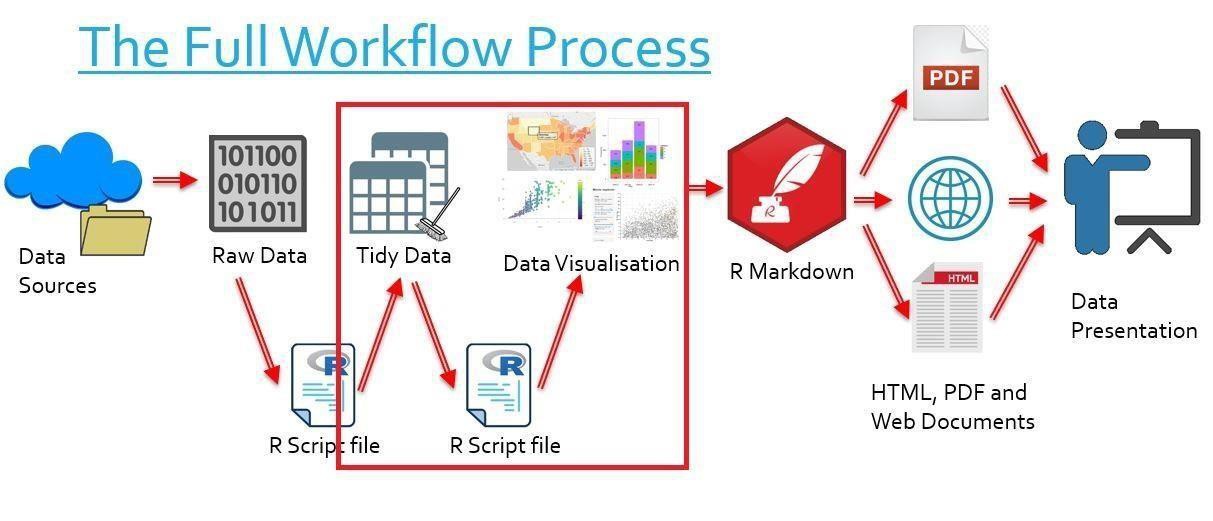
**4. Data Engine**

It Stores data extracts and answers queries

**5. Data Server**

Data Server Manages connections to Plotly Server data sources It also maintains metadata from Plotly Desktop, such as calculations, definitions, and groups.

**6. Plotly Communication Flow**



**2.2. Plotly Express Architecture**

High-Level Features

The Plotly Express API in general offers the following features:

A single-entry point into plotly: just import plotly. express as pxand get access to all the plotting functions, plus built-in demo datasets

under px.data and built-in color scales and sequences under px.color. Every PX



**Sensible, Overridable Defaults**: PX functions will infer sensible defaults wherever possible, and will always let you override them.

**Flexible Input Formats**: PX functions accept input in a variety of formats, from lists and dicts to long-form or wide-form Pandas DataFrames to NumPyarrays and arraysto GeoPandas GeoDataFrames.

**Automatic Trace and Layout configuration**: PX functions will create one trace per animation frame for each unique combination of data values mapped to discrete color, symbol, line-dash, facet-row, and/or facet-column. Traces' legend groupand showlegendattributes are set such that only one legend item appears per unique combination of discrete color, symbol, and/or line-dash. Traces are automatically linked to a correctly-configured subplot of the appropriate type.

**Automatic Figure Labelling**: PX functions label axes, legends, and colorbars based on the input

DataFrameor xarray, and provide extra control with the labelsargument.

**Automatic Hover Labels**: PX functions populate the hover label using the labels mentioned above, and

provide extra control with the hover\_nameand hover\_dataarguments.

**Styling Control**: PX functions read styling information from the default figure template, and support commonly-needed cosmetic controls like category\_ordersand color\_discrete\_mapto precisely control categorical variables.

**Uniform Color Handling**: PX functions automatically switch between continuous and categorical colors based on the input type.

**Faceting**: the 2D-cartesian plotting functions support row, column, and wrapped facetting with

facet\_row, facet\_col,and facet\_col\_wraparguments.

**Marginal Plots**: the 2D-cartesian plotting functions support marginal distribution plots with the

marginal, marginal\_x,and marginal\_yarguments.

**A Pandas backend**: the 2D-cartesian plotting functions are available as a Pandas plotting backend so you can call them via df.plot().

**Trendlines**: px.scattersupports built-in trendlines with accessible model output.

**Animations**: many PX functions support simple animation support via the animation\_frame and animation\_group arguments.



**Automatic WebGL switching**: for sufficiently large scatter plots, PX will automatically use WebGL for hardware-accelerated rendering.

**3. Deployment Description**

**3.1 Deployment options in Plotly**

.Plotly contains many deployment options

.Some of them are deployed using Heroku, Docker, etc.

**Plotly Online**

Get up and running quickly with no hardware required. Plotly Online is fully hosted by Plotly, so all upgrades and maintenance are automatically managed for you.

**Plotly Server deployed on-premises**

Manage and scale your hardware and software (whether Windows or Linux)

as needed. Customize your deployment as you see fit.

**Plotly 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 Plotly Server (on your choice of Windows or Linux). Bring your license or purchase on your preferred marketplace.

**Deploying Dash/Plotly App**



By default, Dash/Plotly apps run on **localhost**- you can only access them on your machine. To share a Dash/Plotly app, you need to "deploy" it to a server.

Our recommended method for securely deploying Dash applications is Dash

[Enterprise.](https://plotly.com/dash)

Dash Enterprise can be installed on the Kubernetes services of AWS, Azure, GCP, [or an on-premise Linux Server.](https://plotly.com/dash/on-premises-linux/?utm_source=docs&utm_medium=workspace&utm_campaign=nov&utm_content=linux)

**Dash Enterprise Deployment**

Dash Enterprise is Plotly's commercial product for developing & deploying Dash Apps on your company's on-premises Linux servers or VPC (AWS, Google Cloud, or Azure).

In addition to easy, git-based deployment, the Dash Enterprise platform provides a complete Analytical App Stack. This includes:

. LDAP & SAML Authentication Middleware

. Data Science Workspaces

. High Availability & Horizontal Scaling

. Job Queue Support

. Enterprise-Wide Dash App Portal

. Design Kit

. Reporting, Alerting, Saved Views, and PDF Reports



. Dashboard Toolkit

. Embedding Dash apps in Existing websites or Salesforce

. AI App Catalog

. Big Data Best Practices

. GPU support

13.

