

# Architecture Design

## NBA DRAFT COMBINE MEASUREMENT

<b>Written By</b>	Author 1
<b>Version</b>	0.1
<b>Last Revised Date</b>	

Rudransh Srivastava

## DOCUMENT CONTROL

### Change Record:

VERSION	DATE	AUTHOR	COMMENTS
0.1	16-Apr-2022	Author 1	Introduction and Architecture defined

### Approval Status:

VERSION	REVIEW DATE	REVIEWED DATE	APPROVED BY	COMMENTS

## Contents

<b>1. Introduction.....</b>	<b>04</b>
<b>1.1 What is Architecture Design Document?.....</b>	<b>04</b>
<b>1.2 Scope.....</b>	<b>04</b>
<b>2. Architecture .....</b>	<b>05</b>
<b>2.1 Plotly Server Architecture .....</b>	<b>05</b>
<b>2.2 Plotly Express Architecture .....</b>	<b>08</b>
<b>3. Deployment.....</b>	<b>10</b>
<b>3.1 Deployment options for Plotly .....</b>	<b>10</b>
<b>3.2 Deployment using Heroku.....</b>	<b>10</b>

## 1. Introduction

### 1.1 What is Architecture design document?

Any software needs the architectural design to represent 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.

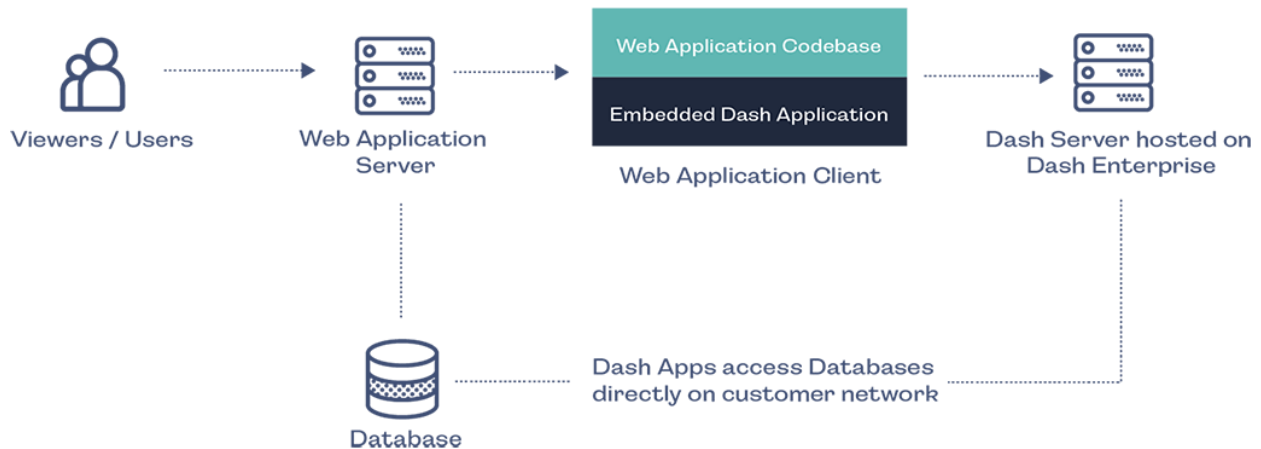
Each style will describe a system category that consists of:

- 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.

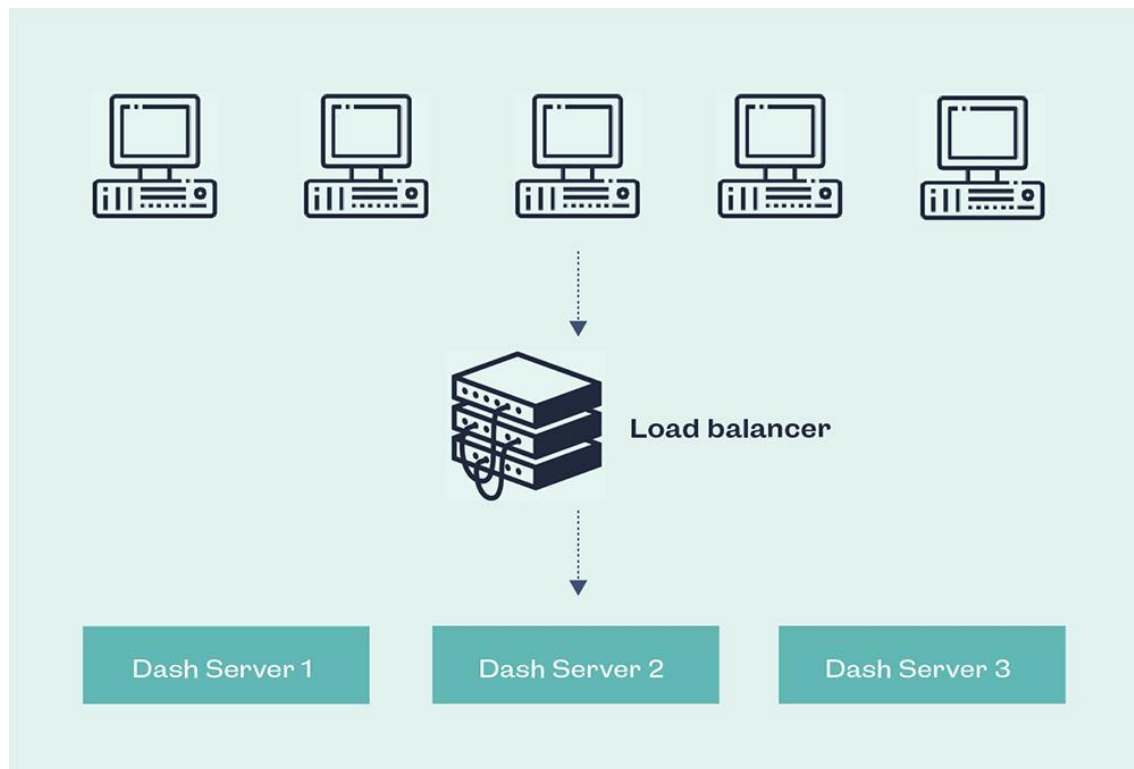
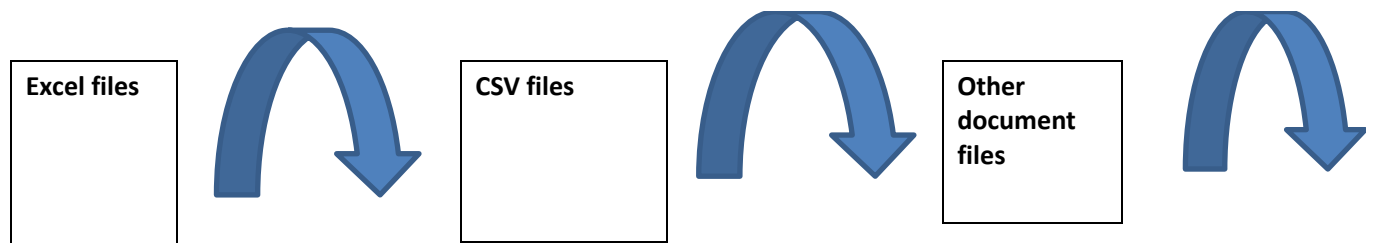
## 2. Architecture



### 2.1 Plotly Server Architecture

The following diagram shows Plotly server architecture

6 ARCHITECTURE DESIGN



## 1. Gateway/Load Balancer

### 7 ARCHITECTURE DESIGN



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 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:-

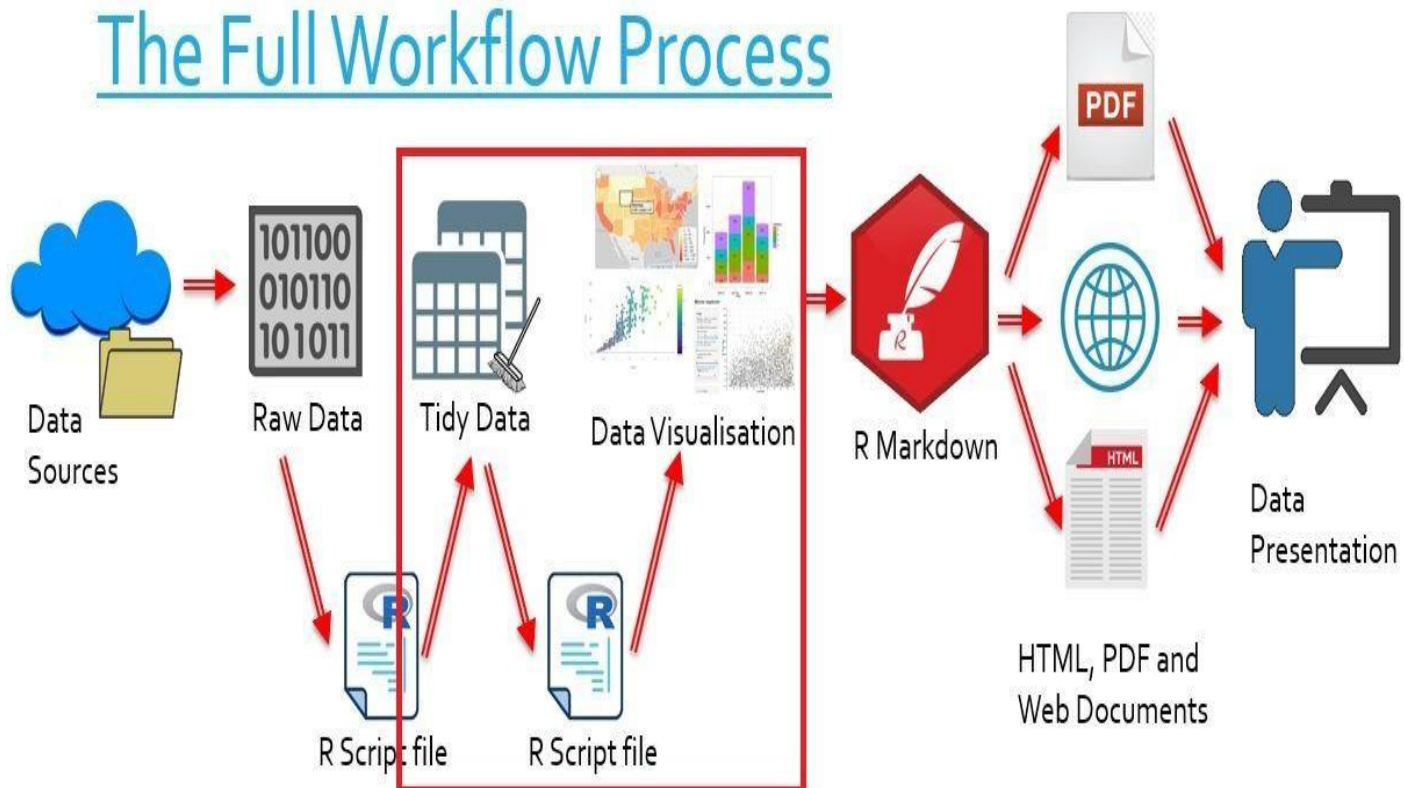
Data Server Manages connections to Plotly Server data sources It also maintains metadata from Plotly Desktop, such as calculations, definitions, and groups.

## 6 Plotly Communication Flow:-

### 8 ARCHITECTURE DESIGN



## The Full Workflow Process



## 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 px` and 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

9

ARCHITECTURE DESIGN



**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 `numpy arrays` and `xarrays` to `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' `legendgroup` and `showlegend` attributes 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 in the input `DataFrame` or `xarray`, and provide extra control with the `labels` argument.

**Automatic Hover Labels:** PX functions populate the hover-label using the labels mentioned above, and provide extra control with the `hover_name` and `hover_data` arguments.

**Styling Control:** PX functions read styling information from the default figure template, and support commonly-needed cosmetic controls like `category_orders` and `color_discrete_map` to precisely control categorical variables.

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

**Faceting:** the 2D-cartesian plotting functions support row, column and wrapped faceting with `facet_row`, `facet_col` and `facet_col_wrap` arguments.

**Marginal Plots:** the 2D-cartesian plotting functions support marginal distribution plots with the `marginal`, `marginal_x` and `marginal_y` arguments.

**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.scatter` supports 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 deployment 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 own 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 own license or purchase on your preferred marketplace.

## Deploying Dash/Plotly App

11

ARCHITECTURE DESIGN



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

Our recommend method for securely deploying Dash applications is Dash Enterprise.

Dash Enterprise can be installed on the Kubernetes services of AWS, Azure, GCP, or an on-premise Linux Server.

### 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

## 12 ARCHITECTURE DESIGN



- . Dashboard Toolkit
- . Embedding Dash apps in Existing websites or Salesforce
- . AI App Catalog
- . Big Data Best Practices
- . GPU support

13.

**ARCHITECTURE DESIGN**



Dash Open Source	<b>&gt; 2M users</b> <i>The largest open-source community for building ML apps</i>	Open Source Features
	<b>Python, R, &amp; Julia support</b> <i>Build Dash apps in your language of choice</i>	
	<b>Extensive docs &amp; gallery</b> <i>+ 60 gallery apps with code, +2k forum posts monthly</i>	
Dash Enterprise	<b>App Manager</b> <i>Deploy, share, and manage your Dash apps</i>	ML Ops Features
	<b>Kubernetes</b> <i>Horizontal scaling &amp; zero downtime deployment</i>	
	<b>No-Code Authentication</b> <i>Login to Dash apps with SSO, SAML, AD, LDAP, &amp; more</i>	
	<b>Job Queue</b> <i>Queueing is key to building scalable ML apps</i>	
	<b>Password Vault</b> <i>Secure access to tokens and passwords in your Dash app code</i>	
	<b>Design Kit</b> <i>Professional grade styling &amp; branding</i>	Low-Code Features
	<b>Snapshot Engine</b> <i>Create, archive, &amp; share Dash app views as links</i>	
	<b>Reporting</b> <i>Programmatically generate pixel-perfect PDFs &amp; reports</i>	
	<b>Dashboard Toolkit</b> <i>Drag &amp; drop, crossfilter, &amp; chart editing for Dash apps</i>	
	<b>Embedding</b> <i>Embed Dash apps inside web apps or Salesforce</i>	
	<b>ML &amp; AI Templates</b> <i>Copy-paste Dash templates for PyTorch, Tensorflow, Keras, etc.</i>	Enterprise AI Features
	<b>User Analytics</b> <i>Built-in user database &amp; usage dashboard</i>	
	<b>GPU &amp; Dask Acceleration</b> <i>Parallelize your Python code or run in GPU memory</i>	
	<b>Data Science Workspaces</b> <i>Write Dash &amp; Jupyter code in the Workspaces editor</i>	
	<b>Big Data for Python</b> <i>Connect Dash apps to Databricks, Snowflake, Dask, and more</i>	