Waterfall Workflow

A picture containing diagram

Description automatically generated

|  |  |
| --- | --- |
| Project timeline |  |
| Plan |  |
| Create |  |
| Review |  |
| Release |  |

TPM Workflow

Diagram

Description automatically generated

|  |  |
| --- | --- |
| Process groups |  |
| Initiating |  |
| Planning |  |
| Executing |  |
| Closing |  |
| Monitoring & controlling |  |

PRINCE2 Structure

A picture containing diagram

Description automatically generated

|  |  |
| --- | --- |
| PRINCE2 processes |  |
| Business case |  |
| Organization |  |
| Quality |  |
| Plan |  |
| Risks |  |
| Change |  |
| Progress |  |
| PRINCE2 themes |  |
| PRINCE2 principles |  |

Agile Workflow

Diagram

Description automatically generated

|  |  |
| --- | --- |
| Sprint |  |
| Requirements understanding |  |
| Design |  |
| Develop |  |
| Test |  |
| Review |  |
| Deliver |  |

Kanban Structure

A picture containing shape

Description automatically generated

|  |  |
| --- | --- |
| Manufacturing |  |
| Distribution |  |
| Consumption |  |
| Kanban card (or label) returns to table |  |
| Kanban table |  |
| Red zone |  |
| If red zone reached, then production (e.g., reconstitution of the consumption) |  |

Online Kanban Board

Graphical user interface, application

Description automatically generated

|  |  |
| --- | --- |
| Online Kanban board |  |
| To do |  |
| In progress |  |
| Done |  |
| Team A |  |
| Modelling T1 |  |
| Presentation T2 |  |
| Validation T3 |  |
| Data collection T1 |  |
| Modelling T3 |  |
| Validation T2 |  |
| Data collection T3 |  |
| Feature Eng. T3 |  |
| Team B |  |
| Validation T4 |  |
| AB Test T4 |  |
| Model 1 Test T4 |  |
| Model 2 Test T4 |  |
| Model 3 Test T4 |  |
| Data collection T4 |  |
| Understanding T4 |  |

Kanban Principles

Diagram, timeline

Description automatically generated

|  |  |
| --- | --- |
| Kanban Principles |  |
| 1. Start with what you do now |  |
| 2. Agree to improve through incremental and evolutional changes |  |
| 3. Encourage or act like a leader at all levels |  |
| 4. Respect the rules and responsibilities of the current process |  |

Kanban Practices

Diagram

Description automatically generated

|  |  |
| --- | --- |
| Kanban practices |  |
| Visualize |  |
| Limit WIP |  |
| Manage the flow |  |
| Evaluate and improve |  |
| Make policies explicit |  |
| Use feedback loops |  |

Scrum in Rugby



Agile Scrum Framework

Diagram, text

Description automatically generated

|  |  |
| --- | --- |
| Stakeholder liaison |  |
| Product owner |  |
| PBIs |  |
| Product backlog |  |
| Development team |  |
| Team forecasts work needed to achieve sprint goal |  |
| Sprint planning |  |
| Topic 1: forecast PBIs |  |
| Topic 2: plan work (e.g., tasks) |  |
| Sprint backlog |  |
| Product backlog refinement |  |
| Daily Scrum |  |
| Sprint (max. 1 month) |  |
| Scrum master |  |
| Iterative-incremental development & delivery |  |
| Potentially releasable increment |  |
| Sprint review |  |
| Sprint retrospective |  |

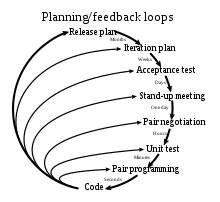
Events in Agile Scrum

A picture containing utensil, table, indoor, sitting

Description automatically generated

|  |  |
| --- | --- |
| Events |  |
| Sprint planning |  |
| Daily Scrum |  |
| Sprint review |  |
| Sprint retrospective |  |
| Product backlog |  |
| Sprint backlog |  |
| Increment |  |
| Sprint |  |

Extreme Programming



|  |  |
| --- | --- |
| Planning/feedback loops |  |
| Release plan |  |
| Months |  |
| Iteration plan |  |
| Weeks |  |
| Acceptance test |  |
| Days |  |
| Stand-up meeting |  |
| One day |  |
| Pair negotiation |  |
| Hours |  |
| Unit test |  |
| Minutes |  |
| Pair programming |  |
| Seconds |  |
| Code |  |

Principles of Lean Management

Diagram

Description automatically generated

|  |  |
| --- | --- |
| Customer needs |  |
| Identify value |  |
| Map the value stream |  |
| Create flow |  |
| Establish pull |  |
| Improve |  |

Comparison of Agile and Traditional Project Management Methods

Table

Description automatically generated

|  |  |
| --- | --- |
| Requirements |  |
| Customer |  |
| Documentation |  |
| Scale of project |  |
| Organization structure |  |
| Model preferences |  |
| Traditional project management |  |
| Clear requirements with low change |  |
| Not involved in the process |  |
| Formal documentation required |  |
| Large-scale |  |
| Linear |  |
| Adaptation to changes |  |
| Agile project management |  |
| Clear requirements with high changes |  |
| Close collaboration |  |
| Implicit knowledge |  |
| Small and medium |  |
| Iterative |  |
| Anticipation of changes |  |

DevOps Pipeline

Diagram, timeline

Description automatically generated

|  |  |
| --- | --- |
| Plan |  |
| Develop |  |
| Build |  |
| Test |  |
| Release |  |
| Deploy |  |
| Operate |  |
| Monitor |  |

DevOps CI/CD

Diagram, timeline

Description automatically generated

|  |  |
| --- | --- |
| Plan |  |
| Develop |  |
| Build |  |
| Test |  |
| Release |  |
| Deploy |  |
| Operate |  |
| Monitor |  |
| Continuous integration |  |
| Continuous delivery |  |
| Continuous deployment |  |

Monolithic vs. Microservices Architecture

Diagram

Description automatically generated

|  |  |
| --- | --- |
| Monolithic architecture |  |
| User interface |  |
| Business layer |  |
| Data interface |  |
| DB |  |
| Microservices architecture |  |
| Microservice UI |  |
| Microservice |  |

CI/CD with Git, Jenkins, Docker, and Kubernetes

Diagram

Description automatically generated

|  |  |
| --- | --- |
| Commit source |  |
| Git |  |
| Build & test |  |
| Jenkins |  |
| Deploy |  |
| Push image |  |
| Docker |  |
| Pull image |  |
| Kubernetes |  |

Docker Components

Diagram

Description automatically generated

|  |  |
| --- | --- |
| Client docker CLI |  |
| REST API |  |
| Server docker daemon |  |
| Docker |  |
| Manages |  |
| Network |  |
| Container |  |
| Image |  |
| Data volumes |  |

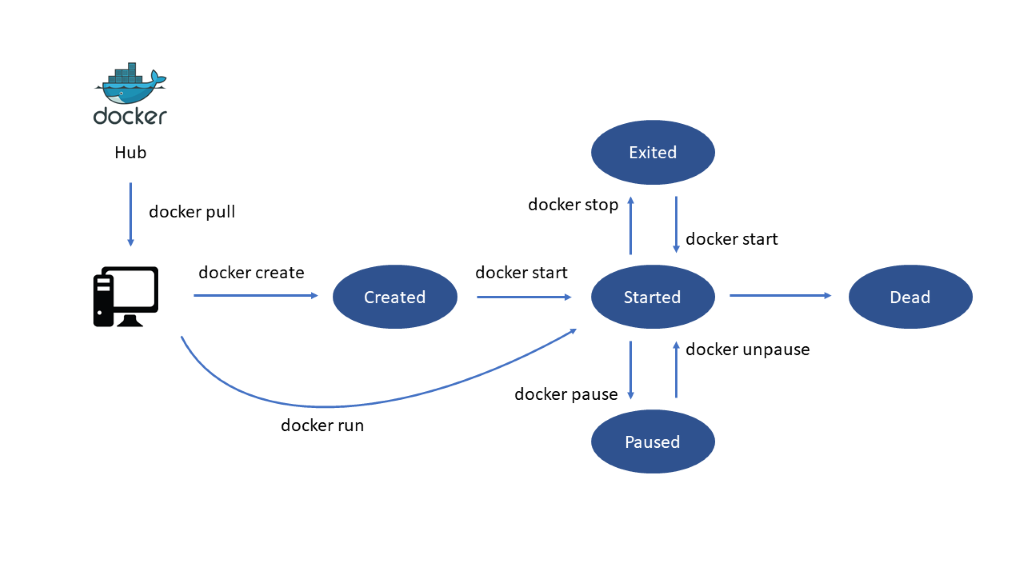
Docker Image vs. Container

Graphical user interface, diagram, application

Description automatically generated

|  |  |
| --- | --- |
| Docker image |  |
| Docker images |  |
| Image registry |  |
| Docker run |  |
| Is instance of |  |
| Docker container |  |
| Docker ps -a |  |
| Host OS |  |

Docker Lifecycle Overview



|  |  |
| --- | --- |
| Docker |  |
| Hub |  |
| Docker pull |  |
| Docker create |  |
| Docker run |  |
| Created |  |
| Docker start |  |
| Exited |  |
| Started |  |
| Paused |  |
| Docker stop |  |
| Docker unpause |  |
| Docker pause |  |
| Dead |  |

CI/CD with Git, Jenkins, Ansible, Docker, and Kubernetes

Diagram

Description automatically generated

|  |  |
| --- | --- |
| Commit source |  |
| Git |  |
| Build & test |  |
| Jenkins |  |
| Deliver |  |
| Ansible |  |
| Push image |  |
| Deploy |  |
| Docker |  |
| Pull image |  |
| Kubernetes |  |

Kubernetes Cluster

Graphical user interface, application, website, Teams

Description automatically generated

|  |  |
| --- | --- |
| Kubelet |  |
| Docker |  |
| Node |  |
| Master |  |
| Kubernetes cluster |  |

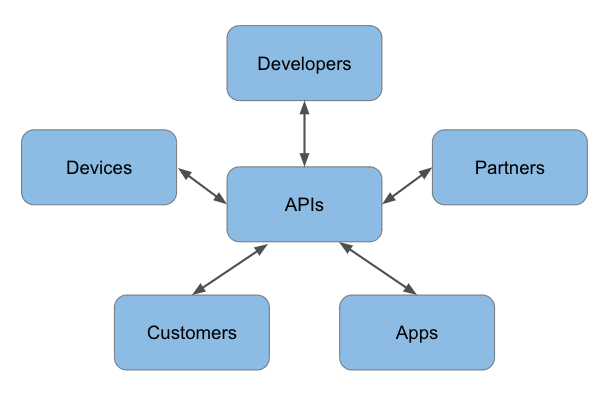
Kubernetes Pods

Chart, bubble chart

Description automatically generated

|  |  |
| --- | --- |
| Kubelet |  |
| Docker |  |
| Pod |  |
| Node |  |

API-Centric View



|  |  |
| --- | --- |
| APIs |  |
| Developers |  |
| Partners |  |
| Apps |  |
| Customers |  |
| Devices |  |

Marick’s Test Quadrant

Technology Facing

Business Facing

Development Support Facing

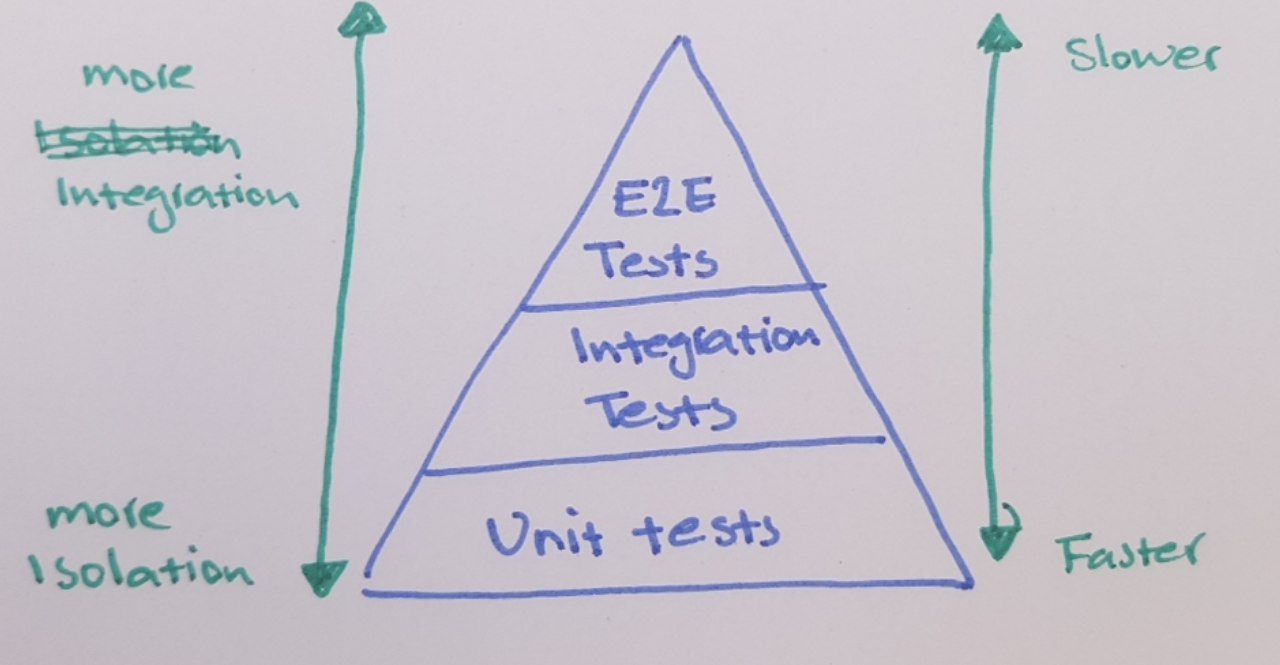
Product Critique

Diagram

Description automatically generated

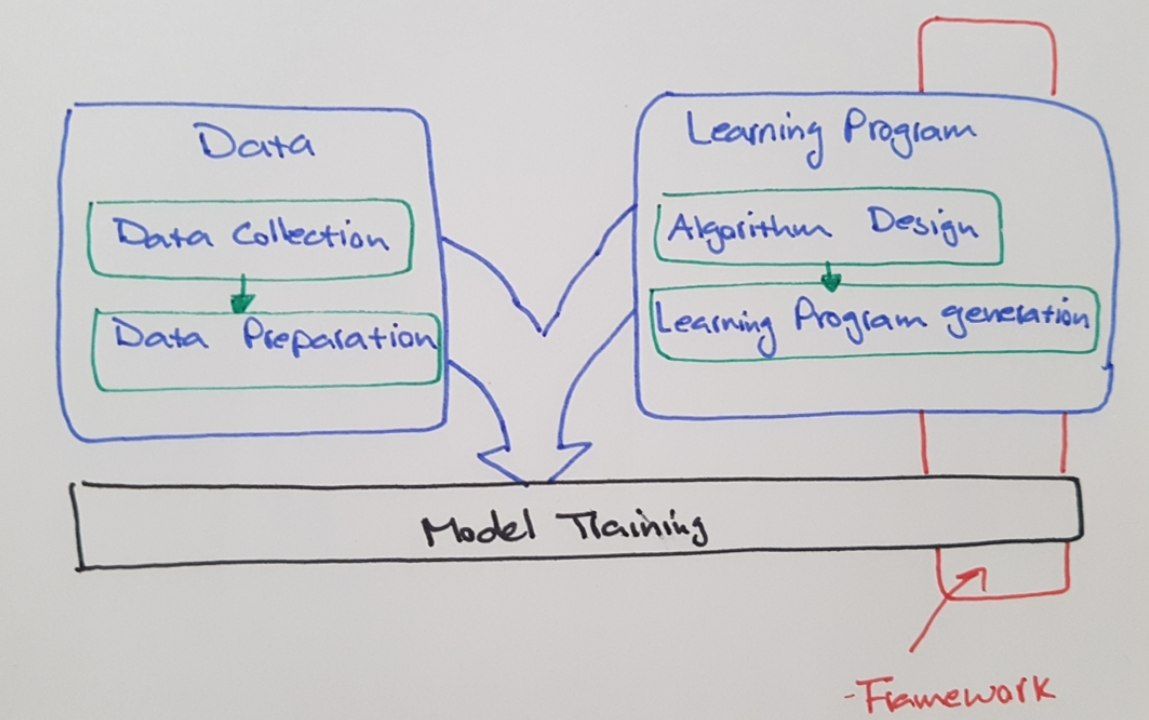
|  |  |
| --- | --- |
| Business facing |  |
| Technology facing |  |
| Development support |  |
| Product critique |  |
| Q2 |  |
| Functional acceptance tests |  |
| (Automated) |  |
| Q3 |  |
| Usability tests |  |
| Exploratory tests |  |
| Showcases |  |
| (Manual) |  |
| Q1 |  |
| Unit tests |  |
| Integration tests |  |
| System tests |  |
| Q4 |  |
| Non-functional acceptance tests |  |
| (Manual/automated) |  |

Test Automation Pyramid



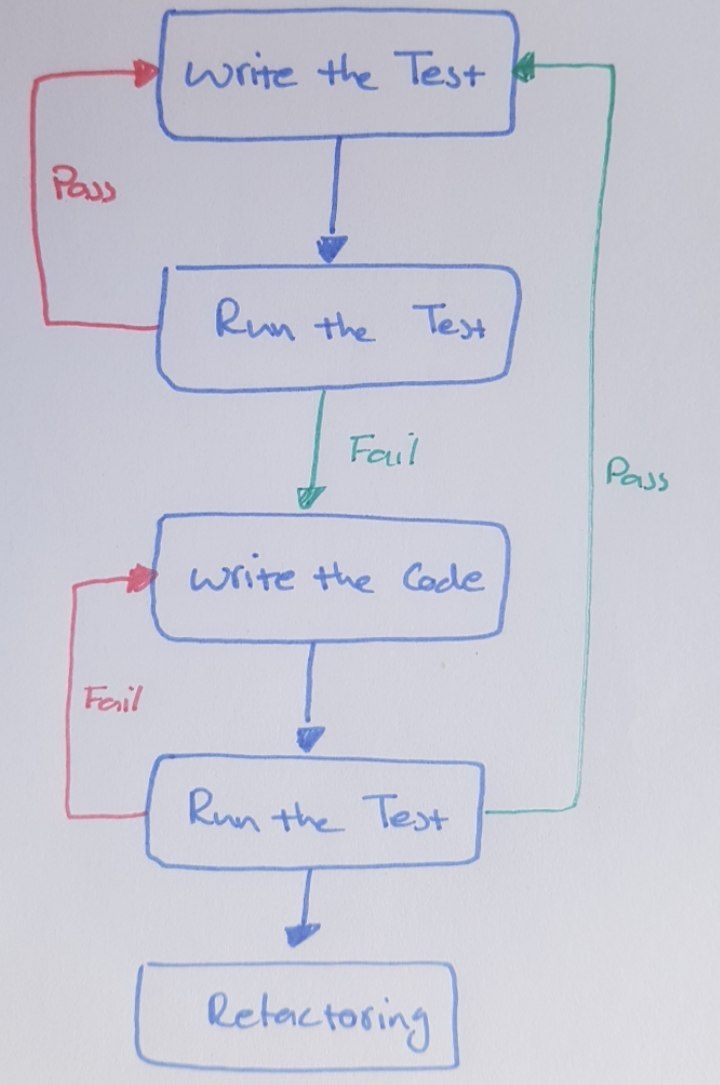
|  |  |
| --- | --- |
| More integration |  |
| More isolation |  |
| E2E tests |  |
| Integration tests |  |
| Unit tests |  |
| Slower |  |
| Faster |  |

Components Involved in the ML Model Building



|  |  |
| --- | --- |
| Data |  |
| Data collection |  |
| Data preparation |  |
| Model training |  |
| Learning program |  |
| Algorithm design |  |
| Learning program generation |  |
| Framework |  |

TDD Lifecycle



|  |  |
| --- | --- |
| Write the test |  |
| Run the test |  |
| Write the code |  |
| Refactoring |  |
| Pass |  |
| Fail |  |

TDD Cycle

Diagram

Description automatically generated

|  |  |
| --- | --- |
| Write a failing test |  |
| Make the test pass |  |
| Refactor |  |

TDD versus BDD versus ATDD

|  |  |  |  |
| --- | --- | --- | --- |
|  | **TDD** | **BDD** | **ATDD** |
| **Definition** | Is a development approach to implement a feature | Is a development approach based on the system behavior | Is a development approach for capturing the requirements (similar to BDD) by writing acceptance tests before implementing the relevant functionality. |
| **Main focus** | Unit tests | Understanding requirements based on system behavior | Writing acceptance tests |
| **Participants** | Developers | Developers, customers, quality assurance engineers | Developers, customers, quality assurance engineers |
| **Language** | Similar to the main code | Plain English | Plain English |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Continuous Delivery (CD) Pipeline

|  |  |
| --- | --- |
| Code commit |  |
| Build |  |
| Acceptance test |  |
| Performance test |  |
| Production |  |

Orchestrated Experiment

|  |  |
| --- | --- |
| Data validation |  |
| Data preparation |  |
| Model training |  |
| Model evaluation |  |
| Model validation |  |

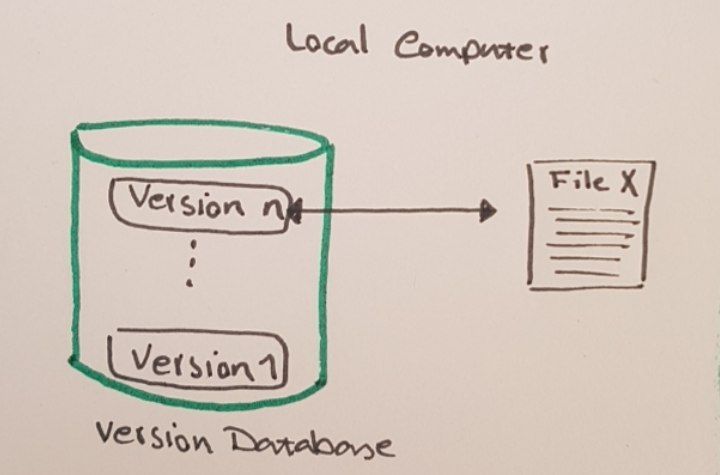
CI/CD for an ML Model

Diagram

Description automatically generated

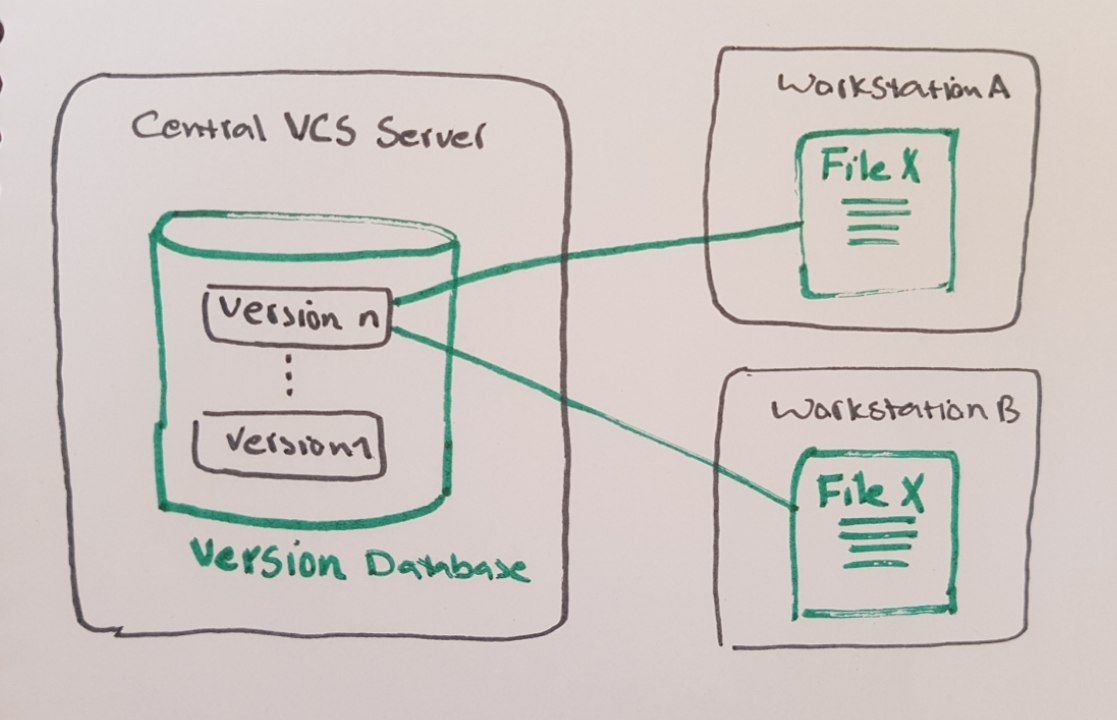
|  |  |
| --- | --- |
| Development & experiment |  |
| Pipeline CI |  |
| Pipeline CD |  |
| Continuous training |  |
| Continuous delivery |  |
| Monitoring |  |

Local Version Control System



|  |  |
| --- | --- |
| Local computer |  |
| Version n |  |
| Version 1 |  |
| Version database |  |
| File X |  |

Centralized Version Control System



|  |  |
| --- | --- |
| Central VCS server |  |
| Version n |  |
| Version 1 |  |
| Version database |  |
| Workstation A |  |
| File X |  |
| Workstation B |  |

Distributed Version Control System

Diagram

Description automatically generated

|  |  |
| --- | --- |
| Server |  |
| Workstation A |  |
| Workstation B |  |
| File X |  |
| Version N |  |
| Version 1 |  |
| Version database |  |

Git Snapshot Approach to Version Control

Diagram

Description automatically generated

|  |  |
| --- | --- |
| Version 1 |  |
| Version 2 |  |
| Version 3 |  |
| Version 4 |  |
| File A |  |
| File B |  |

Sections in a Git Project

Timeline

Description automatically generated

|  |  |
| --- | --- |
| Working directory |  |
| Staging area |  |
| .git directory |  |
| (Repository) |  |
| Checkout the project |  |
| Stage fixes |  |
| Commit |  |

Git Local Repository Trees

A close up of text on a whiteboard

Description automatically generated

|  |  |
| --- | --- |
| Add |  |
| Commit |  |
| Working directory |  |
| Index |  |
| (Stage) |  |
| Head |  |

Branching in GitHub

Diagram

Description automatically generated

|  |  |
| --- | --- |
| ‘Master’ branch |  |
| Create ‘side’ branch |  |
| Commit changes |  |
| Pull in request |  |
| Discuss changes |  |
| Merge ‘side’ to the ‘master’ |  |

IDE Syntax Highlighting

Text

Description automatically generated

|  |  |
| --- | --- |
| import pandas as pd |  |
| import matplotlib.pyploy as plt |  |
| import datetime |  |
| from datetime import datetime |  |
| import plotly as pltly |  |
| import plotly.express as px |  |
| import plotly.graph\_objs as go |  |
| import plotly.io as pio |  |
| from PIL import Image |  |

Eclipse IDE Components

Graphical user interface, application, Word

Description automatically generated

Jupyter Notebook Landing Page

Graphical user interface, application

Description automatically generated

A Jupyter Notebook (.jpynb File)

Graphical user interface, text, application, email

Description automatically generated

Using Jupyter Notebook to Run a Simple Python Code

Graphical user interface, text, application, email

Description automatically generated

Using Markdown to Document the Code in Plain Text in a Jupyter Notebook

Graphical user interface, text

Description automatically generated

In-line Visualization in Jupyter Notebook

Chart, line chart

Description automatically generated

API-Centric View

Diagram

Description automatically generated

|  |  |
| --- | --- |
| Developers |  |
| APIs |  |
| Partners |  |
| Apps |  |
| Customers |  |
| Devices |  |

First Lines of a Dataset With .head()

Table

Description automatically generated

|  |  |
| --- | --- |
| Out[15]: |  |
| SepalLengthCm |  |
| SepalWidthCm |  |
| PetalLengthCm |  |
| PetalWidthCm |  |
| Species |  |
| Iris-setosa |  |

Summary Statistics of a Dataset With .describe()

Table

Description automatically generated

|  |  |
| --- | --- |
| In [16]: 1 iris\_df.describe( ) |  |
| Out[16]: |  |
| SepalLengthCm |  |
| SepalWidthCm |  |
| PetalLengthCm |  |
| PetalWidthCm |  |
| Count |  |
| Mean |  |
| Std |  |
| Min |  |
| Max |  |

Example of a Scatterplot to Visualize Relationships Between Features

Chart, scatter chart

Description automatically generated

|  |  |
| --- | --- |
| Relationship between sepal length and width |  |
| Sepal width |  |
| Sepal length |  |
| Setosa |  |
| Versicolor |  |

Model Development and Production Lifecycle (I)

Diagram

Description automatically generated

|  |  |
| --- | --- |
| Problem understanding |  |
| Data collection |  |
| Cleaning wrangling |  |
| Exploratory data analysis |  |
| Experimentation, feature, and model selection |  |
| Hyperparameters tuning |  |
| Training, final selection |  |
| Initial deployment and integration |  |
| Deployment in production |  |

Model Development and Production Lifecyle (II)

Diagram

Description automatically generated

|  |  |
| --- | --- |
| Experimental phase |  |
| Identify problem and collect and analyze data |  |
| Choose an ML algorithm and code your model |  |
| Experiment with data and model training |  |
| Tune the model hyperparameters |  |
| Iterate tuning and training |  |
| Production phase |  |
| Transform data |  |
| Train model |  |
| Serve the model for online/batch prediction |  |
| Monitor the model’s performance |  |

Model Development and Production With Kubeflow

Graphical user interface

Description automatically generated

|  |  |
| --- | --- |
| Experimental phase with Kubeflow |  |
| Identify problem and collect and analyze data |  |
| Choose an ML algorithm and code your model |  |
| Experiment with data and model training |  |
| Tune the model hyperparameters |  |
| Iterate tuning and training |  |
| PyTorch |  |
| scikit-learn |  |
| TensorFlow |  |
| XGBoost |  |
| Jupyter Notebook |  |
| Fairing |  |
| Pipelines |  |
| Katib |  |
| Production phase with Kubeflow |  |
| Transform data |  |
| Train model |  |
| Serve the model for online/batch prediction |  |
| Monitor the model’s performance |  |
| Chainer |  |
| MPI MXNet |  |
| TFJob |  |
| KFServing |  |
| NVIDIA TensorRT |  |
| PyTorch |  |
| TFServing |  |
| Seldon |  |
| Metadata |  |
| TensorBoard |  |
| Pipelines |  |

How to Approach Powerful Tools

Text

Description automatically generated

|  |  |
| --- | --- |
| Be careful |  |
| This machine has no brain, use your own |  |

SageMaker Administrative Console

Graphical user interface, text, application

Description automatically generated

Granting Access to IAM Roles

Graphical user interface, text, application, email

Description automatically generated