

Table of Contents

Background	1
Establishing the goals of your project	2
What is your project fundamentally about?	2
What are you intending to design/build/investigate?	2
What do you intend to deliver as the project results?	2
What would constitute, in your own and your supervisor's eyes, a 100% satisfactory solution?	3
Design	3
Front-end	3
Back-end.....	3
Testing	4
In the worst case what is the minimum that needs to be completed to achieve a pass?	4
Planning.....	4
Front-end	4
Back-end.....	4
Testing.....	4
External Libraries	5
What are your personal aims that you hope to achieve?	5
Design	5
Technical	5
Testing.....	5
Error calculation	5
Planning.....	5
Optimisation.....	5
Figures	7
Sources	9

Background

There has been existing work on automating tests e.g., Git [\[Figure 6\]](#) and Jenkins. While the existing tools are great at tracking and managing source code change and enabling teamwork on a project, they aren't effective at setting benchmarks and conduct tests for the Machine Learning code. Machine Learning has additional data requirements because it consists of training and testing code.

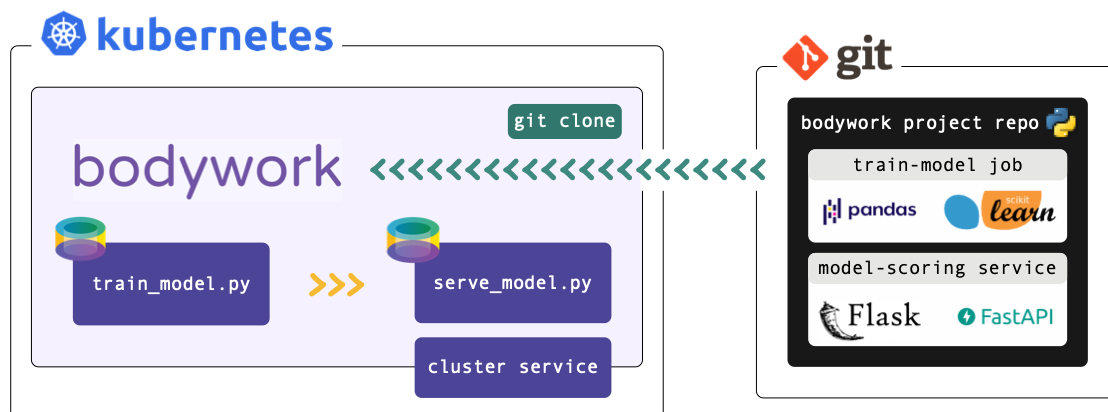
When applying the project, the system should be able to train the datasets based on the training and the benchmarks before moving onto the testing data to determine the model's performance.

Establishing the goals of your project

What is your project fundamentally about?

When building machine learning projects, additional functionalities of training and benchmarking are needed for testing during the code changes. In my project, the goal is to automate the training of the ML code to select the best ML model [1]. My project is divided into two sections: the input containing the ML (training and benchmark) code, and the data. Whenever there is a change made to the repository, the ML system will be retrained to compare with the previous tests. The comparison will help to determine which version of code to become the updated version. Through automation, the project aims to create an iterative process to improve the input machine learning model.

Implement a pipeline to train and test the model with predictions.



What are you intending to design/build/investigate?

The target is to build a machine learning continuous integration system. The project input will be divided into the code (training and benchmark) and the data sections. I will fit the data onto the model to determine its performance. Then I will produce the visualised performance data e.g., model accuracy, sensitivity, and whether they follow the user's requirements.

Whenever there are changes within the dataset or the ML (training and benchmark code), the program must retrain the existing ML model. Consequently, the project simplifies the testing process and ensure that the code will take less time to reach the deployment stage. [Figure 5](#) visualises the MLCI repository branch when two users attempt concurrent edits. After the project has met its objectives, the code will be moved onto the stage and production stage where the code will be deployed.

What do you intend to deliver as the project results?


[Figure 4](#) is the project architectural diagram providing an overview of the requirements. Using the figure, I aim to combine existing version control system with automated training on the machine learning system. In addition to the backend programming, I intend to host the programs on a browser platform, which will allow the user to conduct the testing from different platforms. The browser will contain a user-friendly menu that leads to training and displaying the model score, which will help the user to effectively apply new changes to the ML repository. If the program is examining large datasets, then I will develop or use a suitable existing data version control system to track the changes in the repository.

What would constitute, in your own and your supervisor's eyes, a 100% satisfactory solution?




A perfect solution would be a web-based platform running the machine learning continuous integration tool. Web browser is portable and can be applied for development on different devices from computer to tablet.

Resolve commit conflicts made by different users as show in [Figure 1](#).






Design

Objective	Stage
Outline the software development cycle.	




Front-end

Objective	Stage
Produce a menu for the different functionalities	
For each function, produce a separate webpage for displaying the outputs.	
Develop a UI for configuring the code for testing.	



Back-end

Objective	Stage
Develop version control and suitably handle the commit conflicts.	
Merge function runs tests on the training and the datasets.	
Include external servers to run and test the program in different environment. [Figure 3]	
Create an iterative process to train and benchmark the ML code.	
Calculate a prediction accuracy score on the training and testing datasets.	


Model

Objective	Stage
Automate pushing code	
Automate the ML models and the data	
Train the Machine learning system in a way that the code ca be run on different machines.	


Data Control

Objective	Stage
Design my own data version control system if there are no suitable existing data version control systems.	
Allowing the user to determining the training and validation sample size.	


Display

Objective	Stage
Using the merge results, generate and display performance statistics.	

Testing

Objective	Stage
Set up a test system to compare the new model with the existing model using a common test score.	


Benchmarking

Objective	Stage
Use the benchmark to perform other operations e.g., push code, model, and data onto the repository.	


In the worst case what is the minimum that needs to be completed to achieve a pass?

My goal is developing the basic frameworks for the program as the minimal requirement. The minimal program will be a working version of the testing and running of the machine learning code on a command prompt system. My further purpose would be to produce a visual framework to run the code and will be finished when the time constraints permit the addition of the supplementary functions.

Planning



Objective	Stage
Build the diagram outlining the program structure.	

Front-end


Objective	Stage
Create a web interface for the continuous integration platform	

Back-end




Data

Objective	Stage
Load the datasets into the repository.	
Apply existing data version control system to keep the data up to date.	

Data Processing






Objective	Stage
Split the data into training and testing groups.	

Code function



Objective	Stage
Detect changes within the repository.	
Link existing source control systems with the developed code.	
Enables the ML code to run different versions of local code.	

Testing


Objective	Stage
-----------	-------

For each failed build and test, identify the bugs for the users.	
Provide an outcome for each training process.	
Identify and select a version control system to ensure that the program is up to date.	
I can successfully run the back-end code using the command prompt platform.	
The user can run the ML code (training and benchmarking) as inputs to the software.	

Display

Objective	Stage
Present the benchmark statistics in a visually aesthetic format.	
Generate a probabilistic result (probability in which a result is valid). [3]	

External Libraries




Connect the ML-CI tool with an existing source control (e.g., GitHub or Jenkins)	
--	---

What are your personal aims that you hope to achieve?



Design

1. Learn and apply the ML development cycle into the current CI system. 



Technical

1. Improve and demonstrate Python skills at creating the coded solutions. 
2. Advance understanding in continuous integration for helping with future teamwork. 
2. Improve upon the existing automated testing strategies and improving automated training and benchmarking mechanisms for the ML system. 

Testing

1. Automated testing aims to improve the understanding of the software development cycle [\[Figure 2\]](#). Within the cycle, improve the training for the testing stage and identify the differences between the traditional software testing with testing ML software. 
2. Determine a score (probability) for the validity of the test and a confidence interval. 


Error calculation

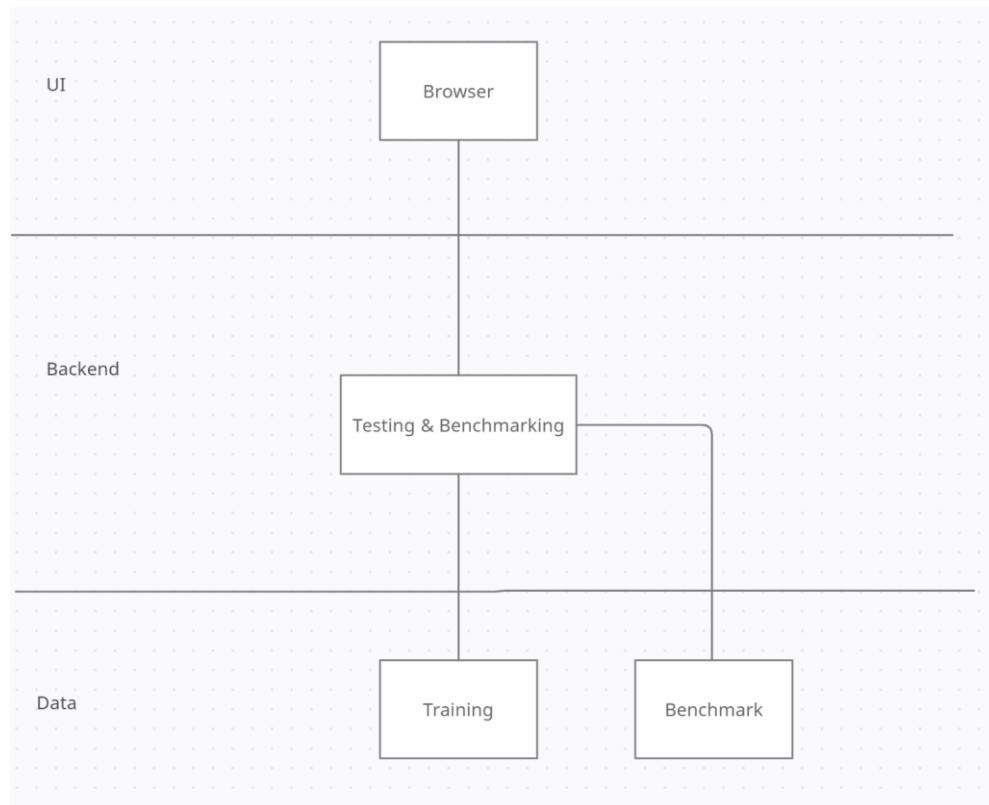
1. Demonstrate building machine learning to minimize the errors from overfitting. 
2. Make some progress within the CI development for the machine learning type of code. 

Planning

1. Apply my research skill into planning and coding the project. 

Optimisation

1. Perform optimisation operations on the machine learning code testing conditions. 



Figures

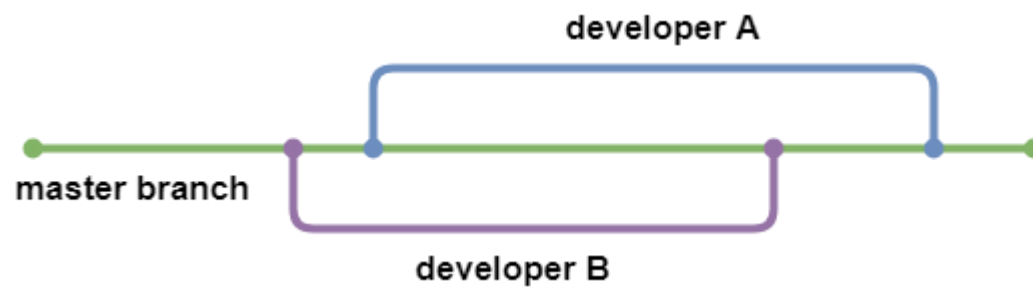


Figure 1

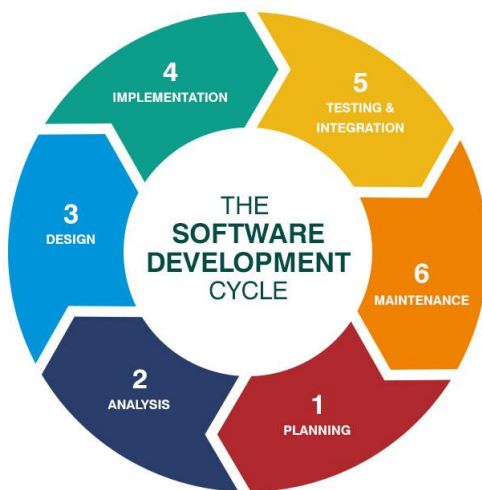


Figure 2

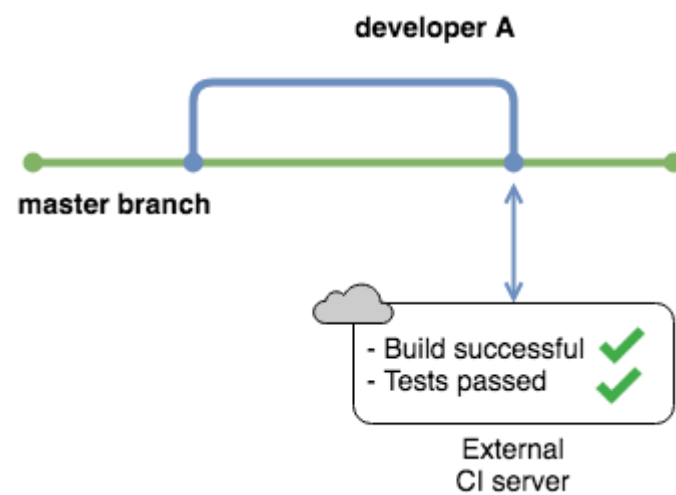


Figure 3

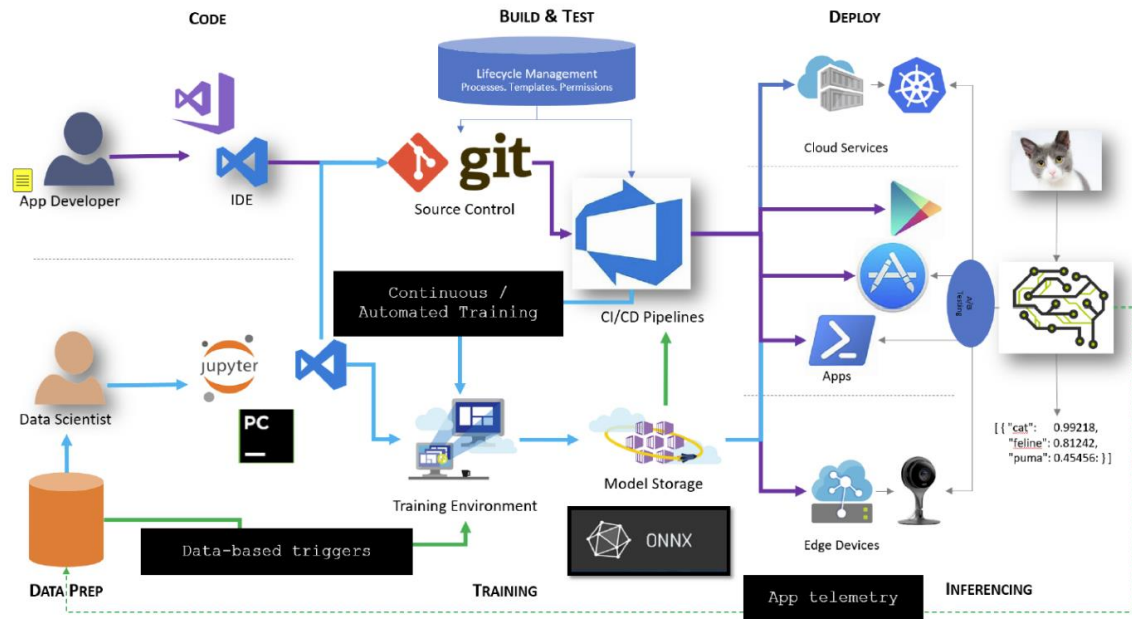


Figure 4

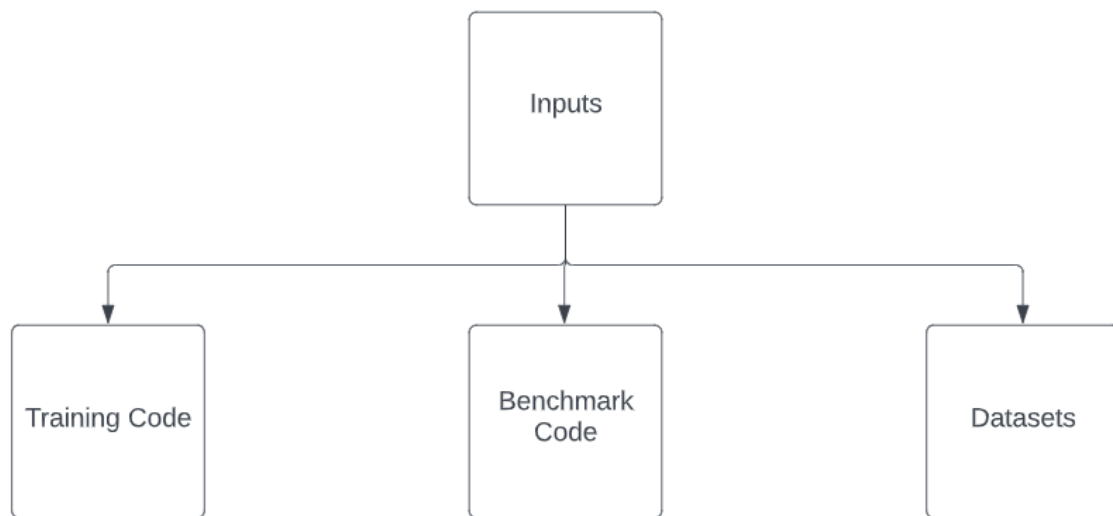


Figure 5

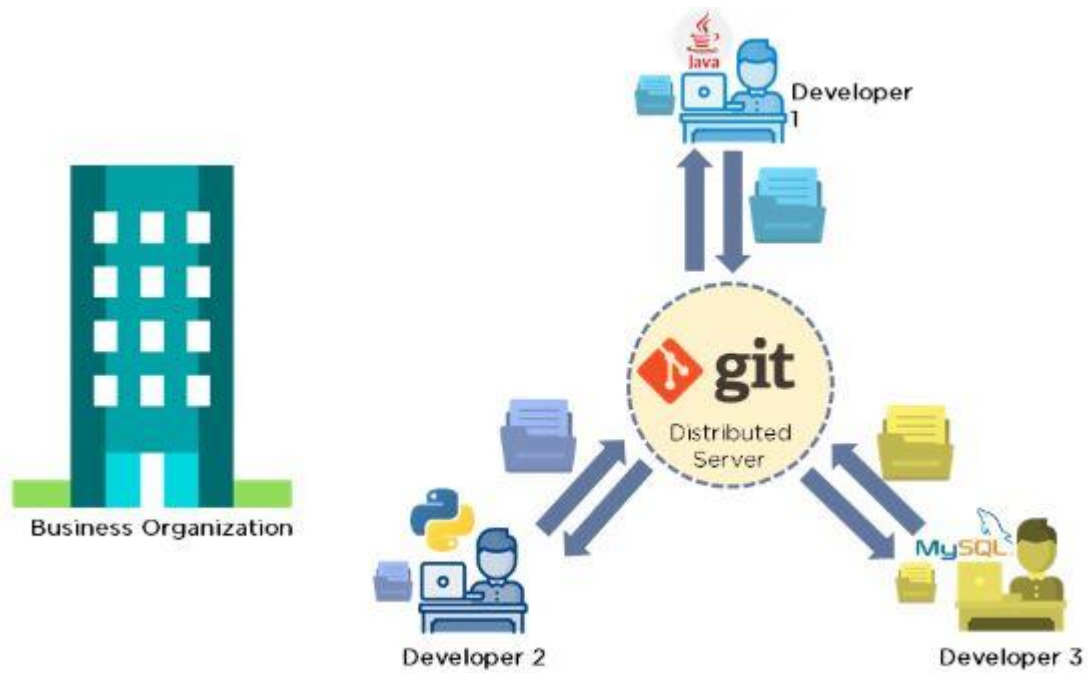


Figure 6

Sources

1. Karlaš, Bojan, et al. "Building continuous integration services for machine learning." Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2020.
2. Danglot, Benjamin, et al. "An approach and benchmark to detect behavioral changes of commits in continuous integration." Empirical Software Engineering 25.4 (2020): 2379-2415.
3. Renggli, Cedric, et al. "Continuous integration of machine learning models with ease. ml/ci: Towards a rigorous yet practical treatment." Proceedings of Machine Learning and Systems 1 (2019): 322-333.

"Continuous delivery is the ability to get changes of all types – including new features, configuration changes, bug fixes, and experiments – into production, or into the hands of users, safely and quickly, in a sustainable way."

Jez Humble

Traditional Delivery Process

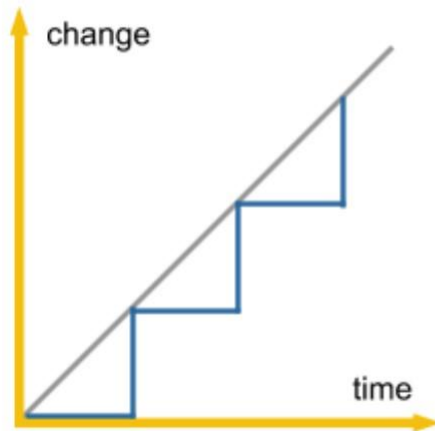
Traditional code development divides the responsibilities into three teams: the development 🖥️, quality assurance 🔧, and release teams ✅. The development team focuses on coding 🐱💻 for passing the code onto the quality assurance team. Using the technical documentation, the quality team conducts user acceptance tests 🔧 on the code from debugging to checking whether the code is following the requirements. After passing the quality tests, the production team determines whether it is possible to stage the updates 👍.

The traditional release cycle can take up to several months to ensure that the code is in production 🏢. The delay ⌚ may result in the code being not up to date 📅 and not fitting the purpose of the customer. If the customer discovered a bug, it has been too long for the development team to recall and identify the problem. Consequently, this makes debugging more challenging after deploying the program. With individuals working on the different code sections, it weakens the communication 🗣️ between the different teams, which makes it challenging to form more coherent and consistent code 🖥️. Therefore, we need an alternative approach to improve the software delivery experiences.

Continuous Delivery

Continuous Delivery uses a pipeline to automate the changes to ensure a faster deployment. A case study examines the differences between Yahoo! and Flickr's code management 🧑💼 culture in 2005. While Yahoo focused on traditional coding approaches. Flickr implemented continuous delivery to produce regular updates each day 📅.

Yahoo!



Flickr

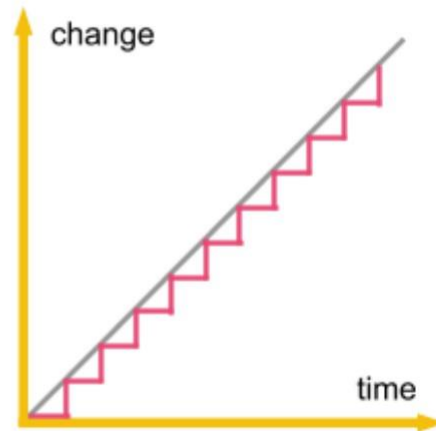


Figure 1.2 – Comparison of the release cycles of Yahoo! and Flickr

The exchange allowed Yahoo! to realise the benefits of continuous integration, reducing downtime. The minor changes in code are reversible and have less risk than implementing a lot of changes in one update. In each update, there will be fewer bugs for the programmer to go through and the client will experience fewer changes.

Today's focus has been on how to construct the program's main structure. I decided to explore the foundations for the program through the Machine Learning Life Cycle.

Define the input for the program.

Previous work has been focused on establishing a pipeline and implementing a function to track the changes. The next step will be to evaluate upon whether the current procedures are in the correct direction and to implement code for tracking the repository into the integration system.

Client working on machine learning repository.

Watches the repo for changes: GitHub

GitHub: check the last time of the updates. Use the Id to check whether there have been any changes.

Tracking code: examine the commit ids.

Create local copies of the data for comparison.

Git computes differences and stores the code.

How to focus on tracking the data in the repository?

Timestamp to track the data changes (simple)

Examine the changes on the single repository.

Version Control : use the existing version control on the sources.

Optional: Design a source control for the data.

Installed git-python to use Python to track the Git repository versions.

I have added comments for the files pipeline.py and pipeline_example.py.

Career: PHD research project for funding and advertised PHD project roles.

Application process: interview.

Institutes with programs: Biology and Wellcome Trust Program: Interview + Selection Process.

<https://wellcome.org/grant-funding/schemes/four-year-phd-programmes-studentships-basic-scientists>

Key identification: Department + Research Topics.

Topics: science and machine learning.

<https://learn.microsoft.com/en-us/azure/devops/pipelines/customize-pipeline?view=azure-devops>

Website to determine the platform for the YAML file.

<https://www.youtube.com/watch?v=9BgIDqAzfuA&list=PL7WG7YrwYcnDBDuCkFbcyjnZQrdskFsBz&index=4>

Tutorial series for implementing a machine learning continuous integration system.

Apply DVC to help with managing the larger datasets.

I've followed the tutorial and completed the steps from the first two videos.

1. Changed load data methods to loading the data from the csv and splitting them into training and the testing sets.
2. Next step: follow the 3rd step video tutorial to extend the application functionalities.
3. I added the wget library to the requirements because it is needed for downloading a zip file.
4. I created the farmers.csv file and now am encountering some difficulties loading the data from the csv file. I managed to load the data headings and am uncertain about the next steps.
5. Identified the useless columns in the table for the future operation to drop the useless columns from the table.

Monthly updates on the application.

Datasets: background using various sources.

Is it possible to automate each part in the server?

Create small versions of the application

Basic: appearance – divide the program into the front end and the back end.

[https://neptune.ai/blog/ways-ml-teams-use-ci-cd-in-production#:~:text=Continuous%20integration%20\(CI\)%20is%20the,\(to%20build%20the%20application\).](https://neptune.ai/blog/ways-ml-teams-use-ci-cd-in-production#:~:text=Continuous%20integration%20(CI)%20is%20the,(to%20build%20the%20application).)

Backend: run preliminary work before the machine learning program.

Design an automated system: run a sequence of instructions.

Predictor: job

Each job is a workflow with a configured engine to run the workflows.

Input: machine, types, and outputs.

Executable: the directory to run the application.

Example:

Job name and runnable determines whether it is possible to run.

Custom data validation methods, there is drop-down.

An administrator can write down the drop-down items.

Set the tasks into order and save the tasks.

Is there going to be overwrite conflicts between the standard outputs and the outputs?

If there are some commands, files outputs are written on system, standard output, and standard error. Outputs are sent to standard out and written to others e.g. png.

Standard output -> file

Out glob: check whether the file has been examined.

https://www.commonwl.org/user_guide/topics/yaml-guide.html

To save work from writing a workflow language

User configures and determines the benchmarks for the machine learning algorithm

Gain performance statistics on the changes

Record the previous running statistics to make decisions.

Writing own or pre-existing data storage systems.

Example input: mnist, consider the scale of the project for the engineering

Example output: performance number from the classification.

Congressional maps

: clustering for grouping different districts

Continuous integration

Adds model to the repository and tests the program functionalities.

Source control

Examine continuous integration and data tracking tools: GitHub, Alien Brain

Other source control systems (GitHub, alienbrain). Source control for data sets?

What other CI tools are out there (Jenkins)

Automated build systems for software.

User interfaces for automation and build systems (i.e., Jenkins)

What languages or technologies to use to complete the project?

Protein Web Server

Strength

1. It will be an excellent exercise for building a new web project.
2. A challenge to learn and apply interfaculty skills, gain a better understanding of protein structures and new algorithms.

World Conqueror 4 mod development

Strengths

1. The application is fun to play and has excellent layout and strategies.

Limitations

1. There are inadequate number of tutorials for how to implement the ai algorithms and creating new levels.

Machine Learning Continuous integration tool

Strengths

1. Plenty of potentials because it is helpful for programmers to improve their code.
2. Python is a common program language.
3. Python's libraries and functionalities made it easier for processing the machine learning functionalities.

Redistricting Tool

Python : Programming language because it has the geopandas library and the plotlines for plotting the data from each county.

Establishing the goals of your project

What is your project fundamentally about?

Recently, there have been an increase in the machine learning development. When building machine learning projects, the common challenge is overfitting or underfitting the model. In our project, the goal is to automate the building, testing, and the deploying of the new ML code [1]. Through the automation, the project aims to create an iterative process to improve the existing machine learning model. Consequently, through the programs, a working machine learning code will be built.

What are you intending to design/build/investigate?

The target is to build a machine learning continuous integration system. While there are existing systems to automate tests on repository code, there remains plenty of areas of development for the machine learning coding aspects.

The project will be divided into the training and the data sections. Based on the training and the testing data, the program automates error analysis, which allows the program to continue the iterations. Using the tests, the project aims to simplify the process to validate the tests and ensure that the product transitions to the deployment stage within a shorter timeframe.

Figure 5 provides a visual representation for the inputs. The inputs will be divided into two sub sections: the code (ML training and benchmark) and the datasets to train the ML algorithms.

What do you intend to deliver as the project results?

Figure 4 provides an overview of the project diagram, in which we allow connections with the existing source control systems. Using the figure, I aim to deliver an interactive training program on ML code. Using each test result and the performance statistics, we aim to retrain the model through automation.


In ML testing, compare the previous and the current test accuracies to determine which version of code to become the updated version. Applying the methods help to consolidate knowledge of the cost of testing and apply the tests on different data sample sizes.

Use the difference between the test set results and the expected results to calculate the test score. Test score will help to compare the machine learning model with the expected result. After a series of continuous iteration testing the code and assess the model with the desired model, the test score will be expected to converge.










What would constitute, in your own and your supervisor's eyes, a 100% satisfactory solution?

Resolve commit conflicts made by different users as show in Figure 1.



Front-end

Objective	Stage
Provide different interfaces with each functionality.	


Back-end

Objective	Stage
Enables the ML code to run different versions of local code.	
Develop version control and suitably handle the commit conflicts.	
Provide a platform to build and run the code.	
Merge function runs tests on the training and the datasets.	
Using the merge results, generate performance statistics.	
Include external servers to run and test the program in different environment. [Figure 3]	
Create an iterative process to train and benchmark the ML code.	
Create a benchmark. The benchmark will provide a report on the repository stats e.g., accuracy and sensitivity.	
Calculate a prediction accuracy score on the training and testing datasets.	

Data Control


Objective	Stage
For large datasets, conduct research into different data version controls and select a suitable method to process the datasets.	
Allowing the user to determining the training and validation sample size.	

Testing


Objective	Stage
Set up a test system to compare the new model with the existing model using a common test score.	

In the worst case what is the minimum that needs to be completed to achieve a pass?










Planning

Objective	Stage
Build the diagram outlining the program structure.	

Front-end

Objective	Stage
Create a web interface for the continuous integration platform	

Back-end

Objective	Stage
Provide unit testing for each functionality	
Split the data into training and testing groups.	
Present the benchmark statistics in a visually aesthetic format.	
For each failed build and test, identify the bugs for the users.	
Provide an outcome for each training process.	
Identify and select a version control system to ensure that the program is up to date.	
Connect the ML-CI tool with an existing source control (e.g., GitHub or Jenkins)	
I can successfully run the back-end code using the command prompt platform.	
Integrate the system with an existing ML service.	

What are your personal aims that you hope to achieve?

1. Improve and demonstrate Python skills at creating the coded solutions. 🎯
2. Automated testing aims to improve the understanding of the software development cycle [Figure 2]. Within the cycle, improve the training for the testing stage and identify the differences between the traditional software testing with testing ML software. 🎯
3. Learn and apply the ML development cycle into the current CI system. 🎯
4. Demonstrate building machine learning to minimize the errors from overfitting. 🎯
5. Improve upon the existing automated testing strategies and improving automated training and benchmarking mechanisms for the ML system. 🎯
6. Apply my research skill into planning and coding the project. 🎯
7. Make some progress within the CI development for the machine learning type of code. 🎯

Figures

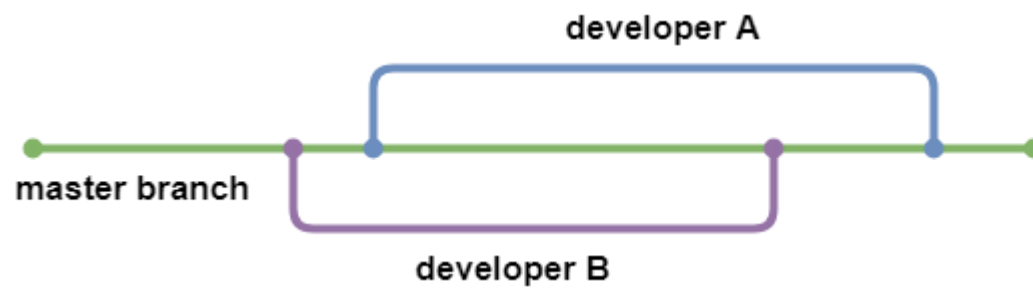


Figure 1

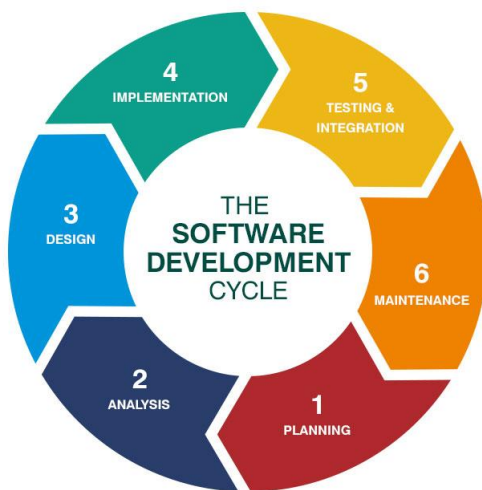


Figure 2

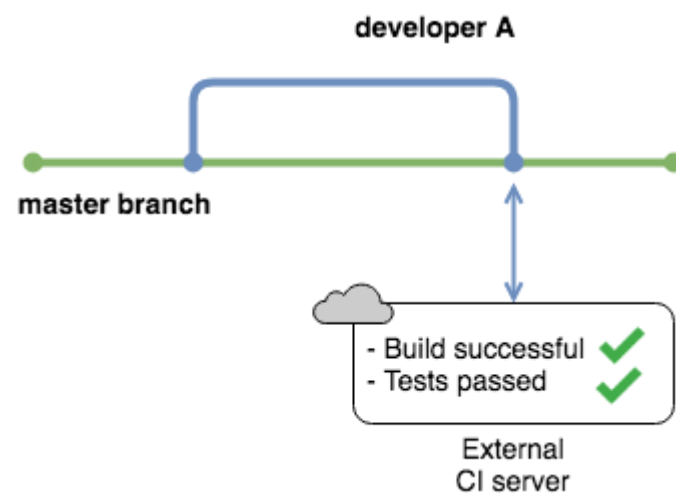


Figure 3

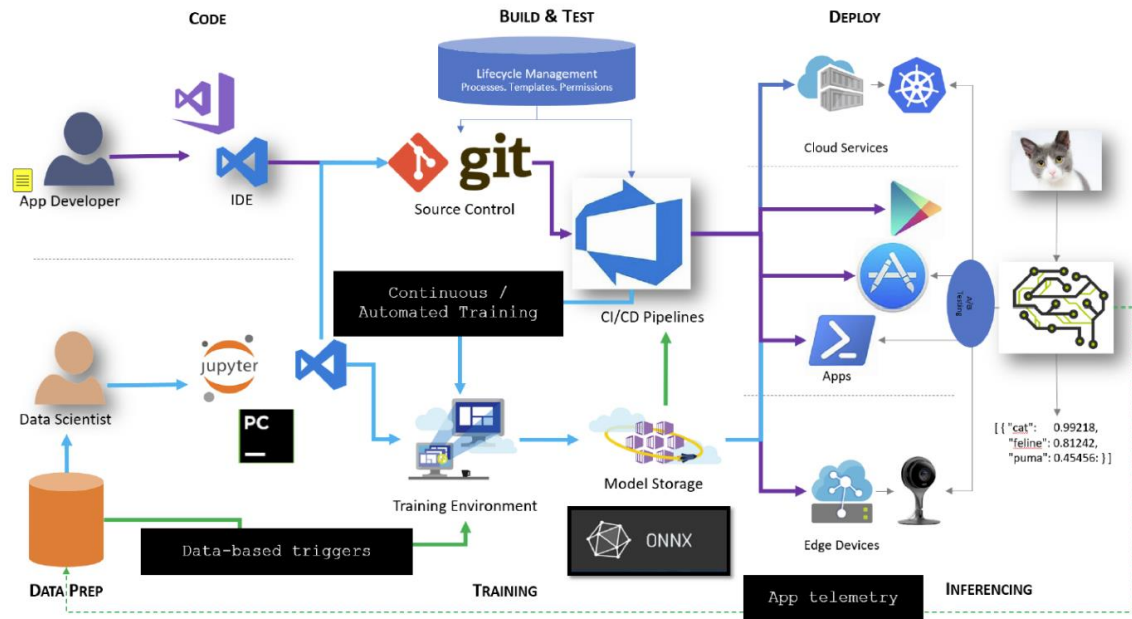


Figure 4

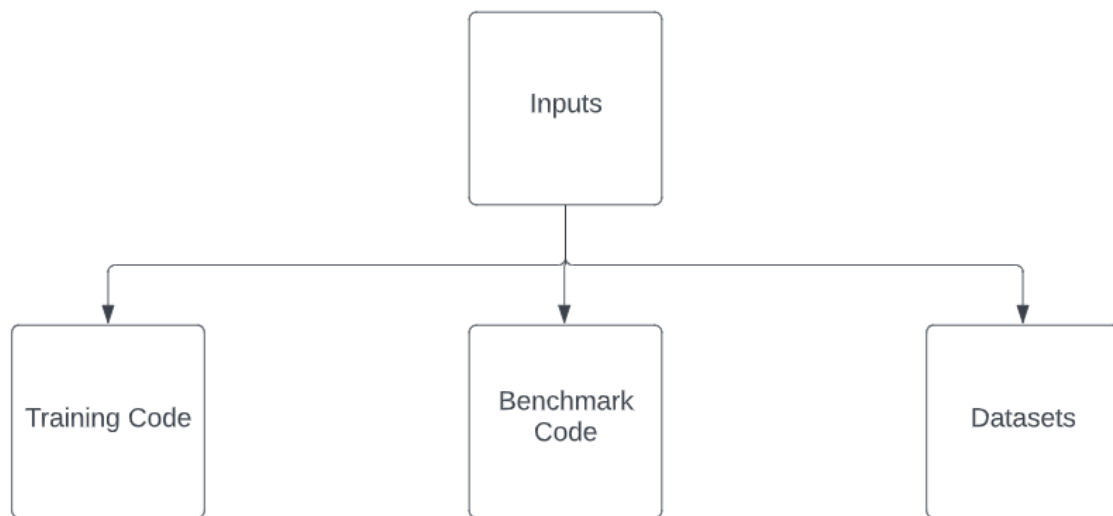


Figure 5

Sources

1. Karlaš, Bojan, et al. "Building continuous integration services for machine learning." Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2020.
2. Danglot, Benjamin, et al. "An approach and benchmark to detect behavioral changes of commits in continuous integration." Empirical Software Engineering 25.4 (2020): 2379-2415.



ML-CI System Specification

Table of Contents

Background	1
Establishing the goals of your project	1
What is your project fundamentally about?1	
What are you intending to design/build/investigate? 1	
What do you intend to deliver as the project results? 2	
What would constitute, in your own and your supervisor's eyes, a 100% satisfactory solution? 2	
Design	2
Front-end	2
Back-end.....	2
In the worst case what is the minimum that needs to be completed to achieve a pass? 3	
Planning.....	3
Front-end	3
Back-end.....	3
Figures	0
Sources	3

Background

There has been existing work on automating tests e.g., Git [\[Figure 6\]](#) and Jenkins. While the existing tools are great at tracking and managing source code change and enabling teamwork on a project, they aren't effective at setting benchmarks and conduct tests for the Machine Learning code. Machine Learning has additional data requirements because it consists of training and testing code.

When applying the project, the system should be able to train the datasets based on the training and the benchmarks before moving onto the testing data to determine the model's performance.

Establishing the goals of your project

What is your project fundamentally about?

When building machine learning projects, the common challenge is overfitting or underfitting the model. In our project, the goal is to automate the building, testing, and the deploying of the new ML code [\[1\]](#). My project is divided into two sections: the input containing the ML training and benchmark code, and the data. Through the automation, the project aims to create an iterative process to improve the existing machine learning model. Consequently, through the programs, a working machine learning code will be built.

What are you intending to design/build/investigate?

The target is to build a machine learning continuous integration system. While there are existing systems to automate tests on repository code, there remains plenty of areas of development for the machine learning coding aspects.

The project will be divided into the training and the data sections. Based on the training and the testing data, the program automates error analysis, which allows the program to continue the iterations. Using the tests, the project aims to simplify the process to validate the tests and ensure that the product transitions to the deployment stage within a shorter timeframe.

Figure 5 provides a visual representation for the inputs. The inputs will be divided into two sub sections: the code (ML training and benchmark) and the datasets to train the ML algorithms.

What do you intend to deliver as the project results?

Figure 4 provides an overview of the project diagram, in which we allow connections with the existing source control systems. Using the figure, I aim to deliver an interactive training program on ML code. Using each test result and the performance statistics, we aim to retrain the model through automation.


In ML testing, compare the previous and the current test accuracies to determine which version of code to become the updated version. Applying the methods help to consolidate knowledge of the cost of testing and apply the tests on different data sample sizes.

Use the difference between the test set results and the expected results to calculate the test score. Test score will help to compare the machine learning model with the expected result. After a series of continuous iteration testing the code and assess the model with the desired model, the test score will be expected to converge.


What would constitute, in your own and your supervisor's eyes, a 100% satisfactory solution?

Resolve commit conflicts made by different users as show in Figure 1.



Design

Objective	Stage
Implement a full software development cycle.	






Front-end

Objective	Stage
Provide different interfaces with working functionalities.	



Model

Objective	Stage
Automate pushing code	
Automate the ML models and the data	


Back-end

Objective	Stage
Develop version control and suitably handle the commit conflicts.	
Merge function runs tests on the training and the datasets.	
Include external servers to run and test the program in different environment. [Figure 3]	
Create an iterative process to train and benchmark the ML code.	
Calculate a prediction accuracy score on the training and testing datasets.	


Data Control

Objective	Stage
For large datasets, conduct research into different data version controls and select a suitable method to process the datasets.	
Allowing the user to determining the training and validation sample size.	


Testing

Objective	Stage
Set up a test system to compare the new model with the existing model using a common test score.	

Display


Objective	Stage
Using the merge results, generate and display performance statistics.	

Benchmarking


Objective	Stage
Use the benchmark to perform other operations e.g., push code, model, and data onto the repository.	

In the worst case what is the minimum that needs to be completed to achieve a pass?

Planning


Objective	Stage
Build the diagram outlining the program structure.	

Front-end


Objective	Stage
Create a web interface for the continuous integration platform	

Back-end



Data


Objective	Stage
Load the datasets into the repository.	

Data Processing






Objective	Stage
Split the data into training and testing groups.	

Code function



Objective	Stage
Detect changes within the repository.	
Link existing source control systems with the developed code.	

Enables the ML code to run different versions of local code.	
--	---


Testing

Objective	Stage
For each failed build and test, identify the bugs for the users.	
Provide an outcome for each training process.	
Identify and select a version control system to ensure that the program is up to date.	
I can successfully run the back-end code using the command prompt platform.	
Integrate the system with an existing ML service.	









Display

Objective	Stage
Present the benchmark statistics in a visually aesthetic format.	
Generate a probabilistic result (probability in which a result is valid). [3]	

External Libraries

Connect the ML-CI tool with an existing source control (e.g., GitHub or Jenkins)	
--	---

What are your personal aims that you hope to achieve?

1. Improve and demonstrate Python skills at creating the coded solutions. 
2. Automated testing aims to improve the understanding of the software development cycle [Figure 2]. Within the cycle, improve the training for the testing stage and identify the differences between the traditional software testing with testing ML software. 
3. Learn and apply the ML development cycle into the current CI system. 
4. Demonstrate building machine learning to minimize the errors from overfitting. 
5. Improve upon the existing automated testing strategies and improving automated training and benchmarking mechanisms for the ML system. 
6. Apply my research skill into planning and coding the project. 
7. Make some progress within the CI development for the machine learning type of code. 
8. Determine a score (probability) for the validity of the test and a confidence interval. 

Figures

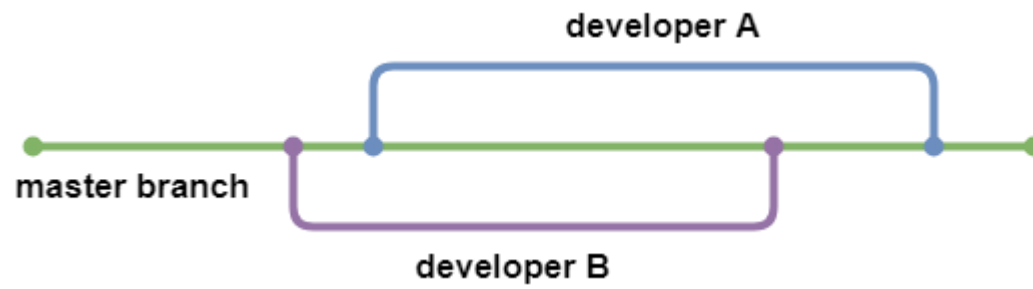


Figure 1

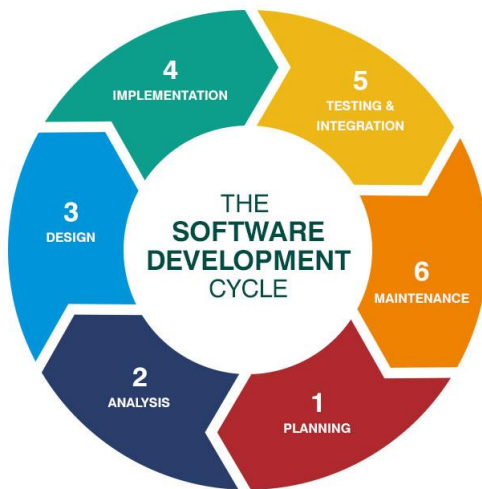


Figure 2

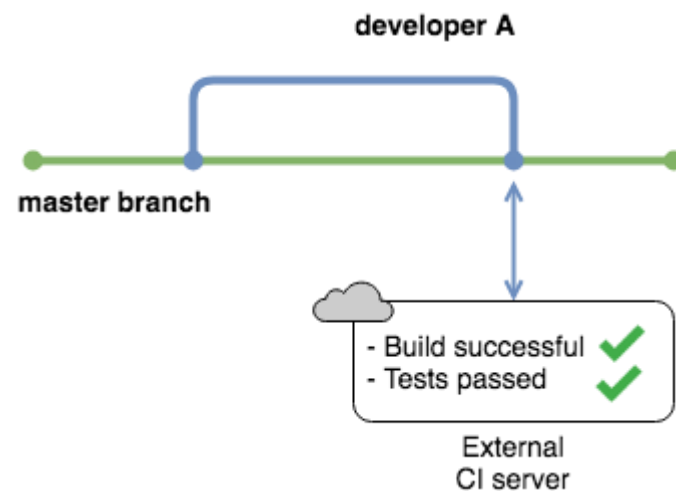


Figure 3

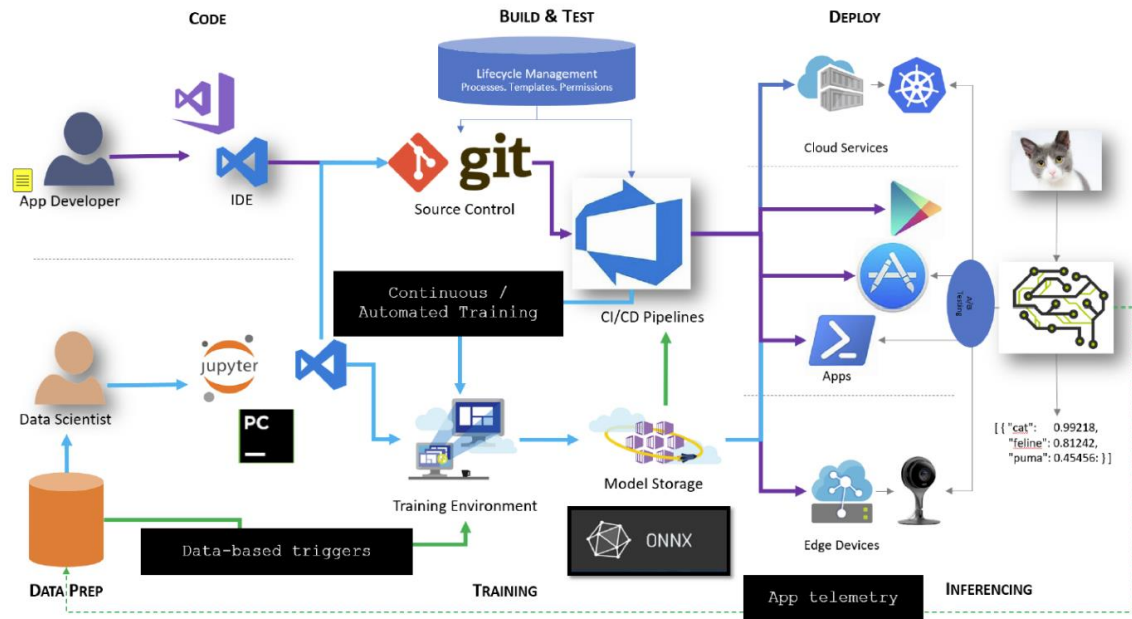


Figure 4

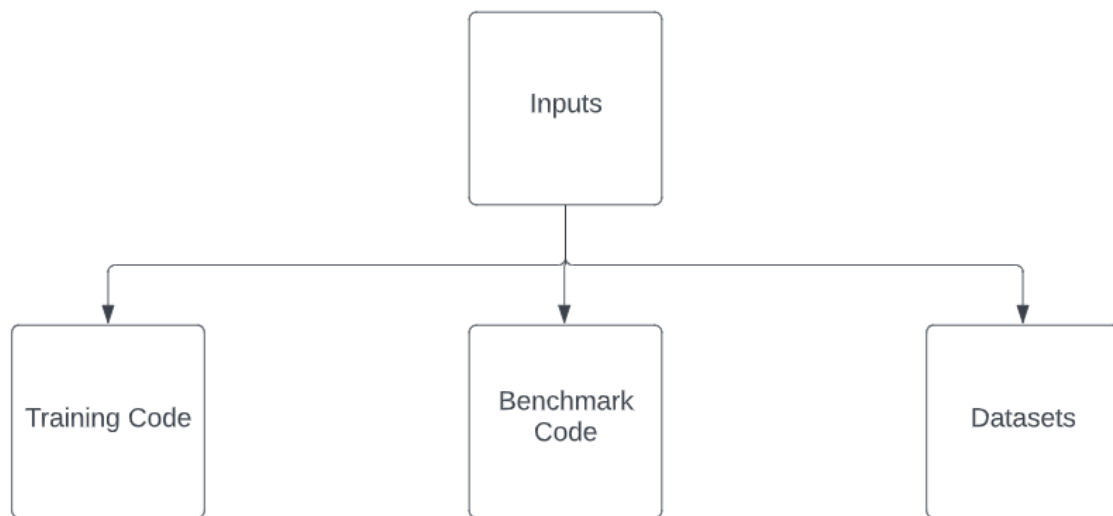


Figure 5

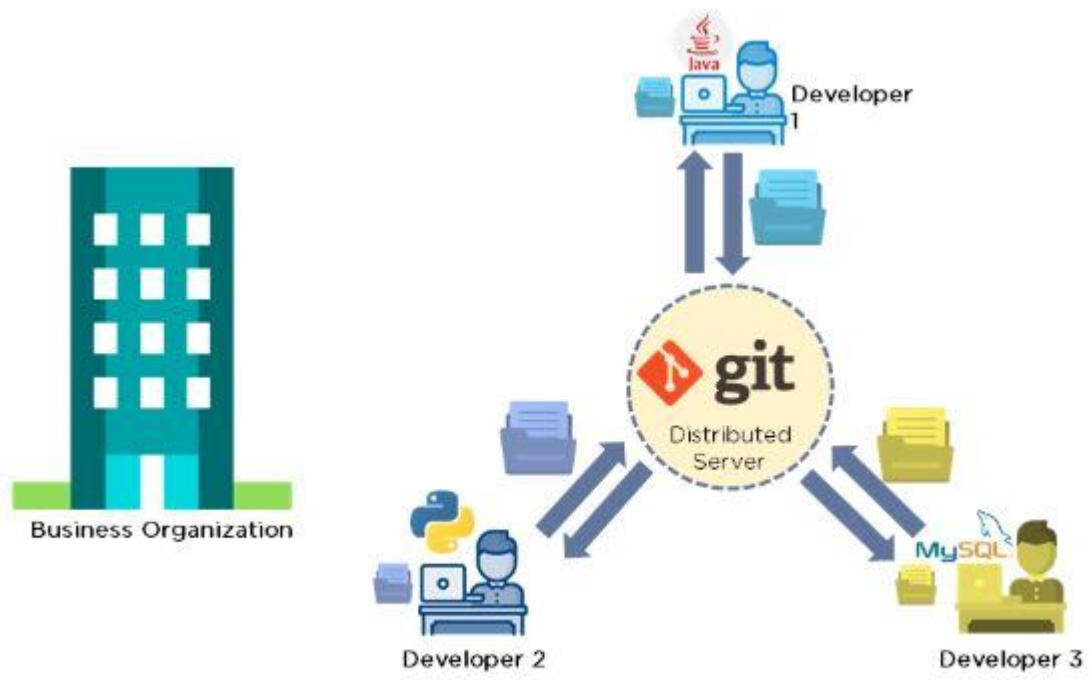


Figure 6

Sources

1. Karlaš, Bojan, et al. "Building continuous integration services for machine learning." Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2020.
2. Danglot, Benjamin, et al. "An approach and benchmark to detect behavioral changes of commits in continuous integration." Empirical Software Engineering 25.4 (2020): 2379-2415.
3. Renggli, Cedric, et al. "Continuous integration of machine learning models with ease. ml/ci: Towards a rigorous yet practical treatment." *Proceedings of Machine Learning and Systems* 1 (2019): 322-333.