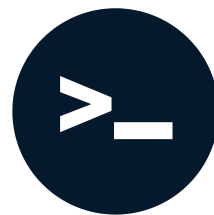


# Introduction to Continuous Integration/Continuous Delivery for Machine Learning

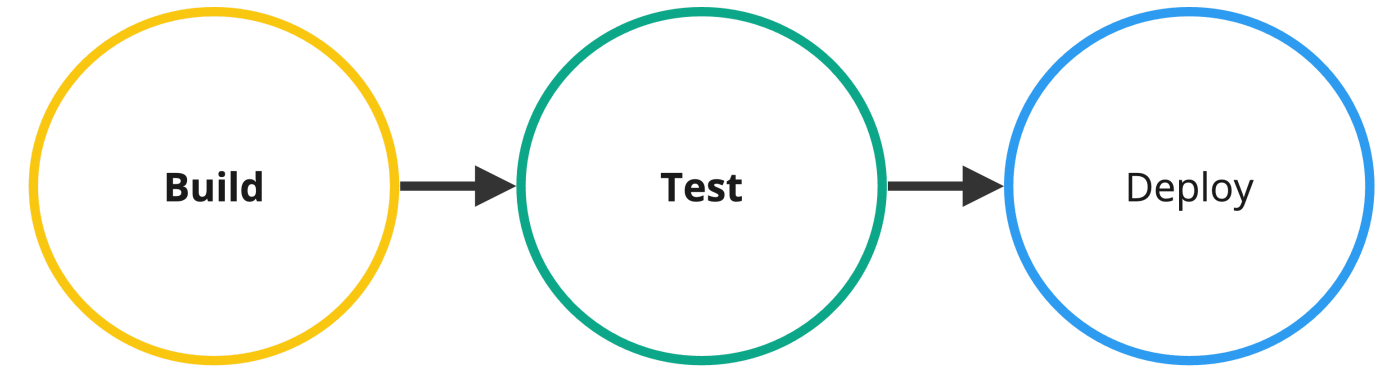
CI/CD FOR MACHINE LEARNING

**Ravi Bhaduria**  
Machine Learning Engineer



# SDLC Overview

- **SDLC: Software Development Life Cycle**
- Systematic approach covering software development from start to finish
- **SDLC workflow** refers to the sequence of steps followed to achieve specific goals:
  - **Build:** compiling and/or packaging code, resolving dependencies
  - **Test:** used to ensure codebase functionality, quality, and reliability
  - **Deploy:** process of making the software available for use in a specific environment



SDLC workflow steps

miro

# SDLC in machine learning

- Machine learning development can be complex and time-consuming
  - Model is an algorithm that evolves dynamically
  - Data engineering is important
- Continuous Integration/Continuous Delivery reduces errors and ensures faster delivery of high-quality ML software
- Essential for efficient machine learning and experimentation

<sup>1</sup> <https://cloud.google.com/blog/products/ai-machine-learning/making-the-machine-the-machine-learning-lifecycle>

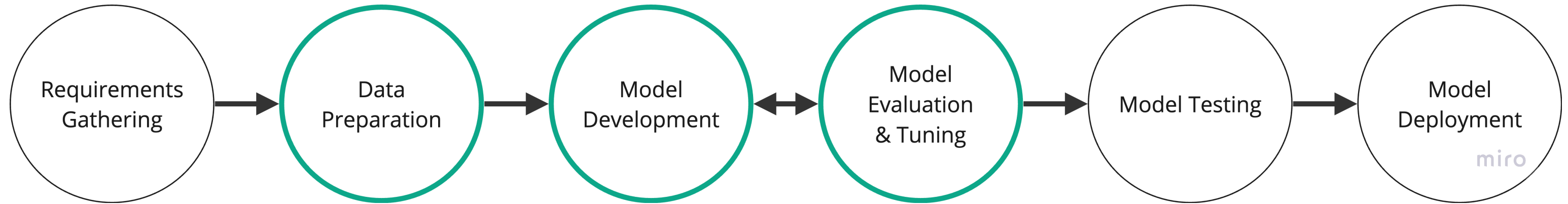
# What is CI/CD?

- **Continuous Integration (CI):** The practice of frequently building, testing, and merging code changes into a shared repository
- Allows developers to detect integration issues early and maintain a consistent codebase
- **Continuous Delivery (CD):** Ensures that code changes can be deployed to production at any time but requires manual approval
- **Continuous Deployment (CD):** Automatically deploys code changes to production without manual intervention

# CI/CD in machine learning

- **Data Dependency:** Data versioning and management strategies
- **Experimentation:** Automating hyperparameter tuning
- **Model Versioning:** Improving collaboration
- **Testing Paradigm:** Goes beyond traditional functional and unit testing
- **Continuous Deployment Challenges:** Complexities in model serving, monitoring, and updates

# Scope of this course



# Summary

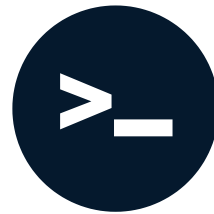
- Software Development Life Cycle workflow involves building, testing, and deploying code
- Continuous Integration (CI) ensures frequent code merging and early issue detection
- Continuous Delivery (CD) allows code changes to be deployed with manual approval
- Continuous Deployment (CD) automates code deployment without manual intervention
- CI/CD in Machine Learning enables
  - Data versioning
  - Building models and model versioning
  - Automating experiments
  - Testing
  - Deployment

**Let's practice!**  
CI/CD FOR MACHINE LEARNING



# Introduction to YAML

CI/CD FOR MACHINE LEARNING



**Ravi Bhaduria**  
Machine Learning Engineer

# What is YAML?

- YAML: YAML Ain't Markup Language
- Used in configuration files, data exchange, and structured data representation
- A data formatting language similar to JSON and XML
- Allows a standard format to transfer data between languages or applications
- Simple and clean format
- Valid file extensions: `.yaml` or `.yml`
- Used to write configuration for variety of CI/CD tools:
  - GitHub Actions
  - Data Version Control (DVC)

# YAML Syntax

- YAML has a hierarchical structure
  - Indentation is meaningful
- **Tabs are not allowed**
- YAML validators  
(<https://www.yamllint.com/>)
- Comments start with #

```
name: Ravi
occupation: Instructor
# This is a valid comment
programming_languages: # and this one too
  python: Advanced
  go: Intermediate
  scala: Beginner
```

# YAML Scalars

- Numbers: Integers or floating-point numbers
- Booleans: `true` or `false`
- Null: keyword `null` or `~`
- Strings: Represented as plain text or enclosed in `' '` or `" "`

```
# Integer
42
# Floating point
3.14
```

```
# Boolean
true
```

```
# Null values
null
```

```
# String value
a: "A string in YAML"
b: 'A string in YAML'
c: A string in YAML
```

# YAML Collections

## Sequences

- Also called lists, arrays, or vectors
- Written in two styles:
  - Block style

```
- first  
- second  
- third
```

- Flow style

```
[first, second, third]
```

## Mappings

- Also called dictionary, key-value pairs, hashes, or objects
- Unique keys, any valid data as values
- Keys and values are separated by `:`

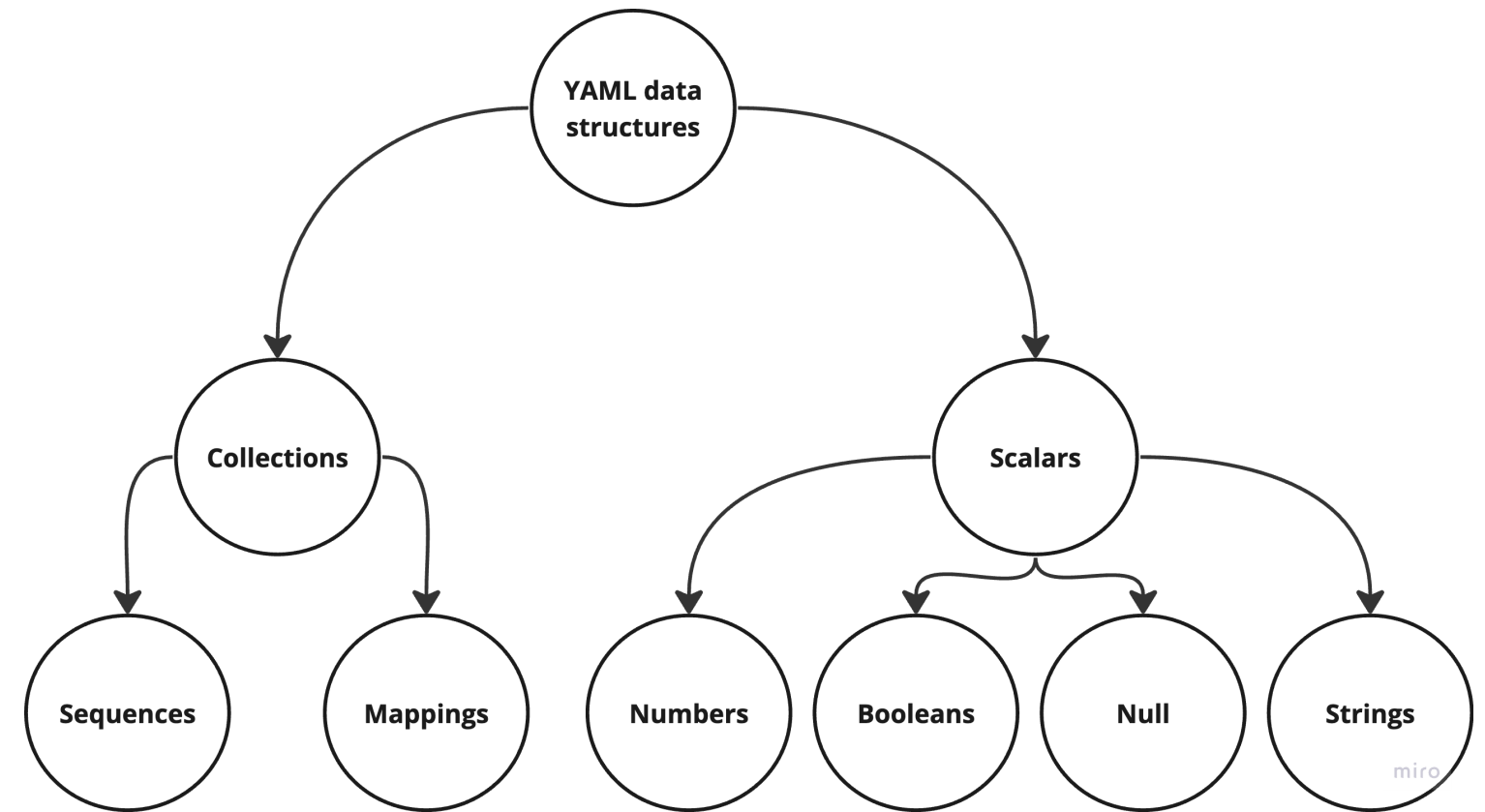
```
key1: value1
```

```
blocklist:  
  - first  
  - second
```

```
flowlist: [1.2, 2, "fifty", true]
```

# Summary

- YAML is a data formatting language
- Useful in writing CI/CD configurations
- Indentation is very important
  - **Tabs are not allowed**
- Mappings, sequences, and scalars are building blocks of YAML



# Editor Exercises Layout

The screenshot displays the DataCamp editor interface for a course titled "CI/CD for Machine Learning". The interface is divided into three main sections:

- Exercise Panel (Left):** Contains the title "YAML mappings and sequences", a description of the task, a list of instructions, and a "Submit Answer" button.
- Code Editor (Center):** Displays a YAML file named "course\_information.yaml" with the following content:

```
1  courses:
2    - name: Machine Learning 101
3      # Complete prerequisites in block format
4      prerequisites:
5        - Linear Algebra
6        - Python Programming
7
8      # Write key for students
9      ____:
10     - name: John Doe
11       # Write midterm scores in flow format
12       midterm_scores: ____
13       final_score: 88
14     - name: Jane Smith
15       # Complete midterm scores in block format
16       midterm_scores:
17         - 78
18         - 84
19
20       ____
21       final_score: 92
```
- Terminal (Bottom Right):** Shows a command prompt with the user's shell prompt and the current directory.

The "Instructions" section lists the following tasks:

- Add the missing prerequisite "Statistics" under the `prerequisites` key.
- Write the key for students, using the keyword `students` as the key.
- Write midterms scores of 85, 92, and 78 for Jon Doe in flow format.
- For Jane Smith's midterm scores, add the missing values `90` under the `midterm_scores` key in block format.

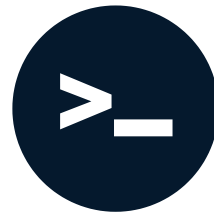
The "Submit Answer" button is located at the bottom of the exercise panel.

**Let's practice!**  
CI/CD FOR MACHINE LEARNING



# Introduction to GitHub Actions

CI/CD FOR MACHINE LEARNING

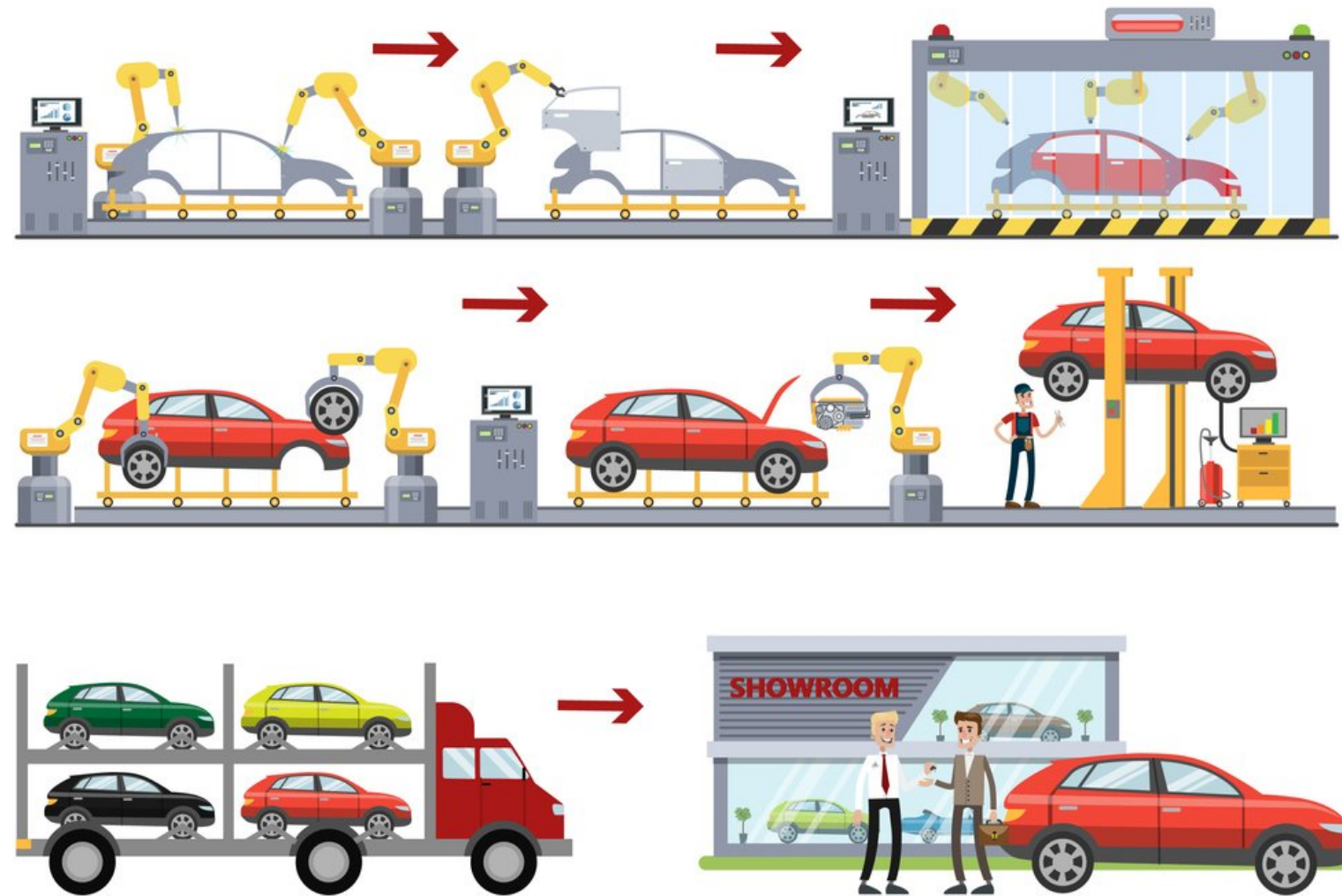


**Ravi Bhaduria**  
Machine Learning Engineer

# What is GitHub Actions?

- **GitHub Actions (GHA):** CI/CD platform to automate *pipelines*
- **Pipeline:** a sequence of steps that represent the flow of work and data

# What is GitHub Actions?



# What is GitHub Actions?



<sup>1</sup> <https://medium.com/empathyco/applying-ci-cd-using-github-actions-for-android-1231e40cc52f>

# GHA Components: Event

- **Event:** is a specific activity in a repository that triggers a workflow run
  - Push
  - Pull Request
  - Opening an issue

# GHA Components: Workflow

- **Workflow:** automated process that will run one or more jobs
  - Defined in YAML files
  - Triggered automatically by **event**
    - Manual run possible
  - Housed in `.github/workflows` directory in the repository
  - Multiple workflows can be defined

# GHA Components: Steps and Actions

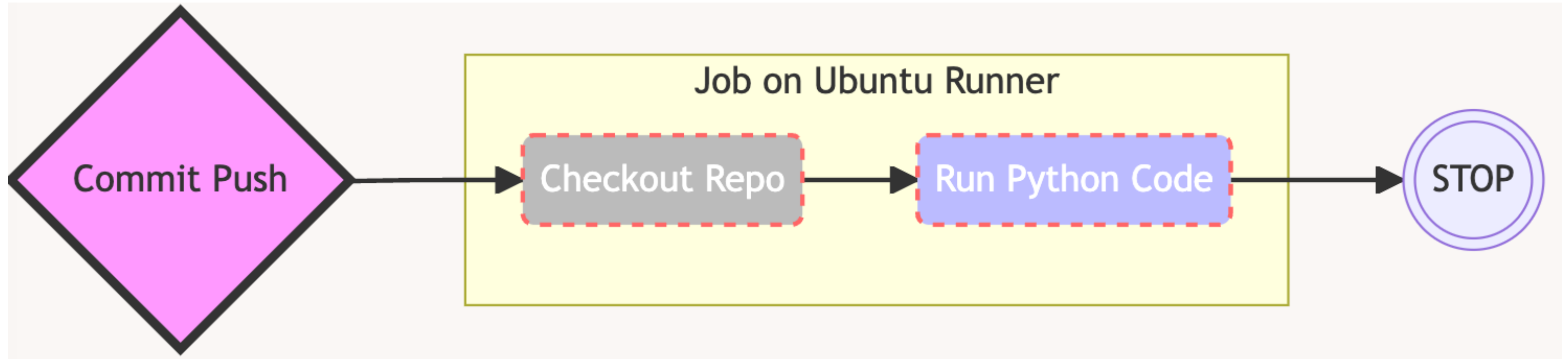
- **Steps:** individual units of work
  - Executed in order, depends on previous step
  - Run on the same machine, so data can be shared
  - Unit of work examples
    - Compiled code application, shell script
    - **Action:** GHA platform specific application
      - E.g. checkout repo, comment on PR

# GHA Components: Jobs and Runners

- **Job:** set of *steps*
  - Each job is independent
  - Parallel execution is possible
  - Executed on the compute machine called **runners**

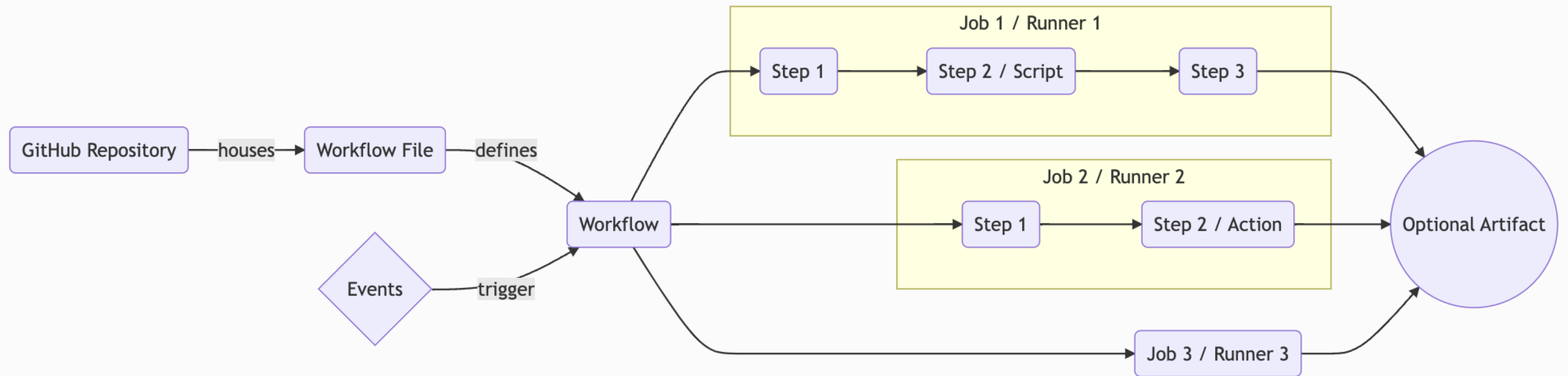


# A simple GHA workflow



- **Event:** Push
- **Job:** runs on *Ubuntu* runner, has two steps
  - **Action:** Checkout Repo
  - Run Python Code

# Putting it all together



**Let's practice!**  
CI/CD FOR MACHINE LEARNING