# Intermediate YAML

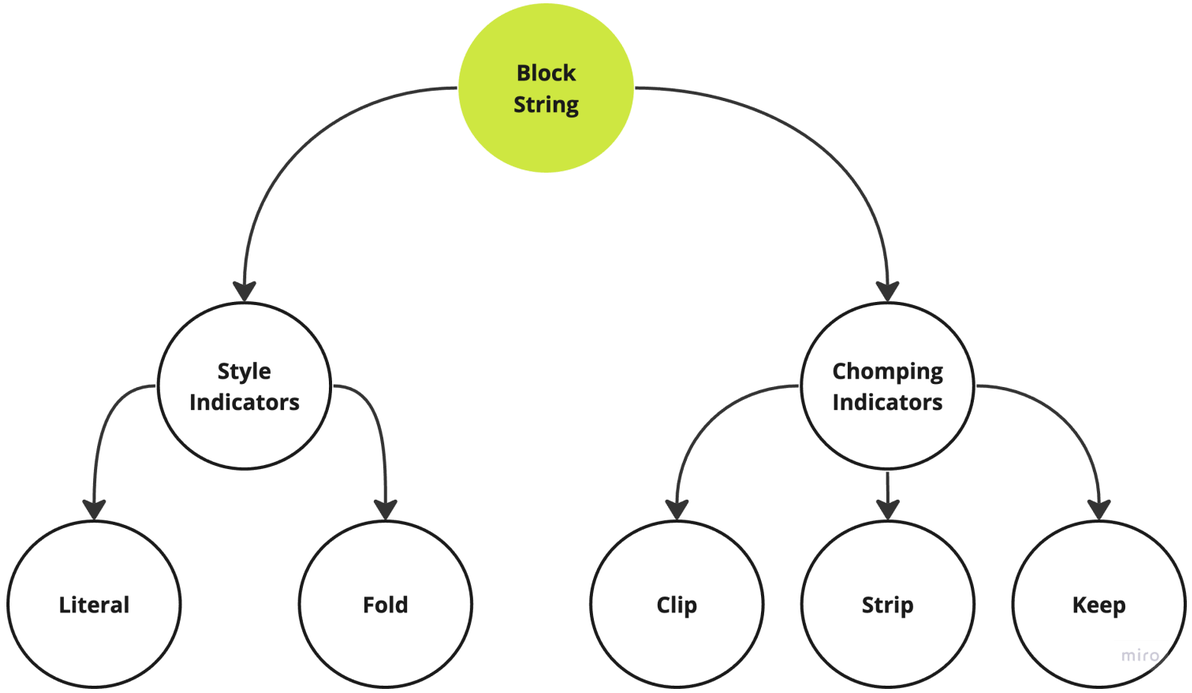
## CI/CD FOR MACHINE LEARNING

**Ravi Bhadauria**
Machine Learning Engineer

datacamp

# Multiline strings: Block scalar format

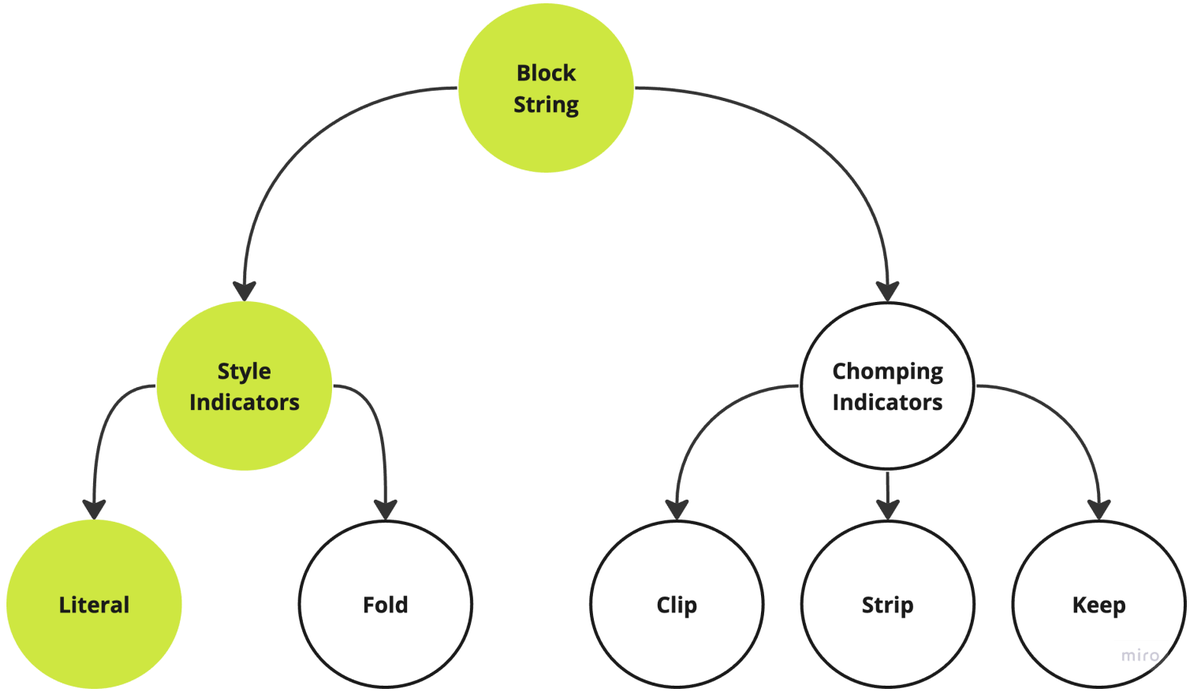
- Represent multiline strings or blocks of text
  - Code snippets (shell commands)
  - Paragraphs (logs)
  - Configuration files



# Style indicators - literal

- Literal style ( `|` ): preserves line break and indentation
- Useful for writing shell commands

```
commands: |
  echo "Running python script..."
  python3 script.py
```



# Literal style example

YAML

```yaml
example: |\n
  Several lines of text,\n
  with some "quotes" of various 'types',\n
  and also a blank line:\n
  \n
  and some text with\n
    extra indentation\n
  on the next line,\n
  plus another line at the end.\n
  \n
  \n
```

Result

```
Several lines of text,\n
with some "quotes" of various 'types',\n
and also a blank line:\n
\n
and some text with\n
  extra indentation\n
on the next line,\n
plus another line at the end.\n
```

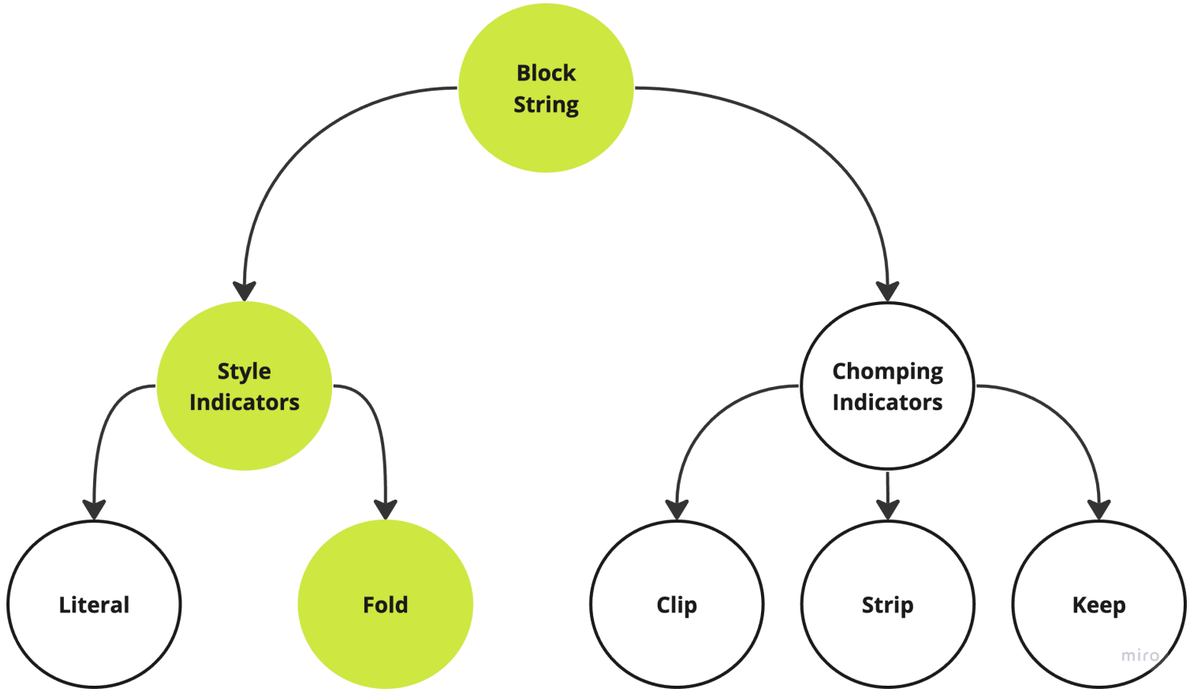
miro

[1] https://yaml-multiline.info/

# Style indicators - fold

- Fold style ( `>` ): removes line breaks
- Useful for writing wrapped messages like paragraphs

```
message: >
  Successfully executed the code!
  All processes completed
  without any errors.
  The results have been saved
  in the designated
  output directory.
```



# Fold style example

YAML

```yaml
example: >\n
  Several lines of text,\n
  with some "quotes" of various 'types',\n
  and also a blank line:\n
  \n
  and some text with\n
    extra indentation\n
  on the next line,\n
  plus another line at the end.\n
  \n
  \n
```

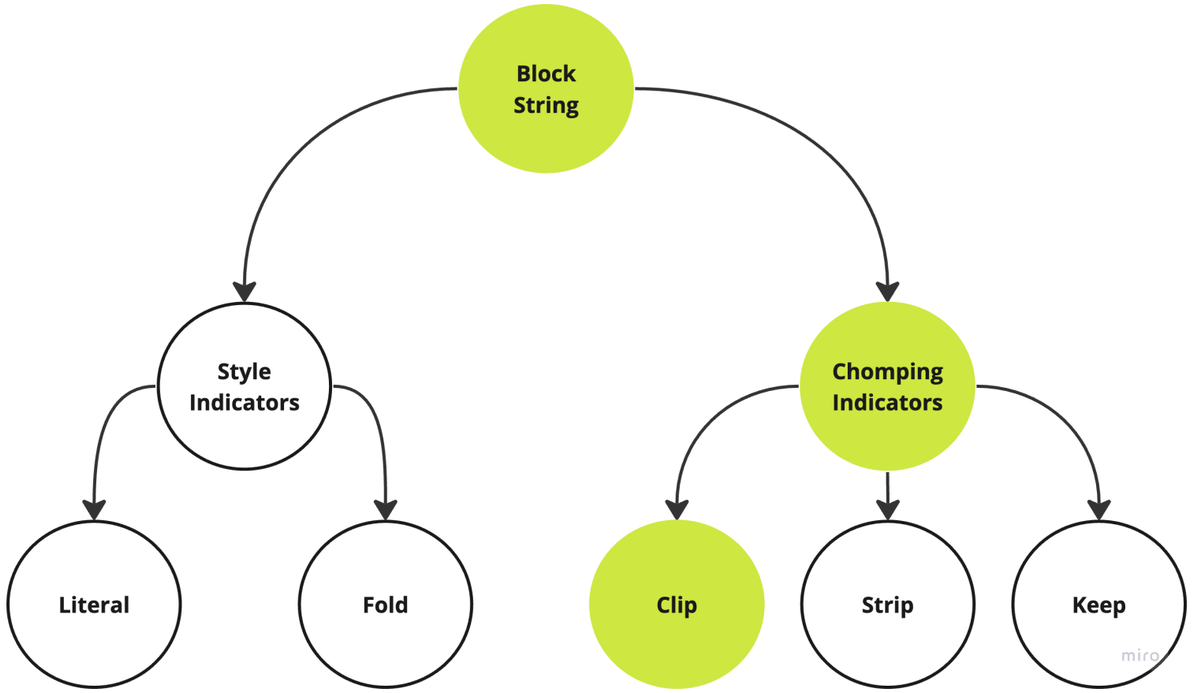
Result

```
Several lines of text, with some "quotes"
of various 'types', and also a blank
line:\n
and some text with\n
  extra indentation\n
on the next line, plus another line at the
end.\n
```

[1] https://yaml-multiline.info/

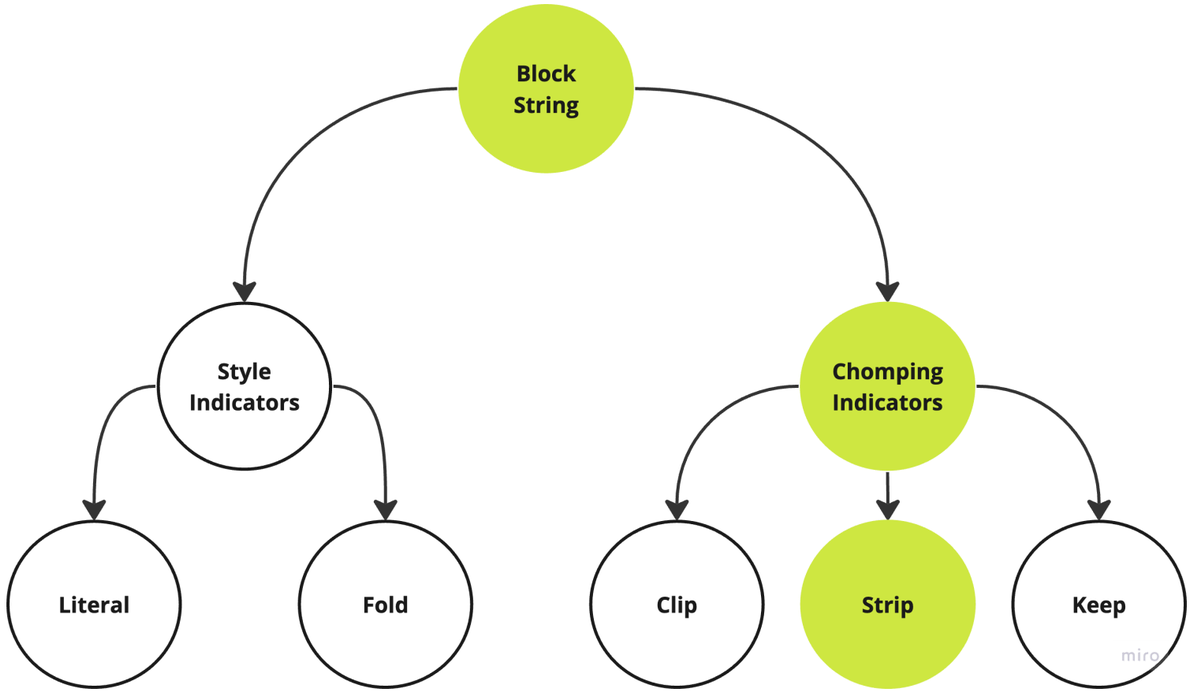
# Chomping indicators

- Control the behavior of newlines at the *end* of the string
- Added after the style indicators
- **clip**: default mode, single newline at the end
  - No additional symbol needed for representation



# Chomping indicators - strip

- **strip** ( `-` ): removes all newlines at the end



# Strip chomping example

YAML

```
example: |-\n
··Several lines of text,\n
··with some "quotes" of various 'types',\n
··and also a blank line:\n
··\n
··and some text with\n
····extra indentation\n
··on the next line,\n
··plus another line at the end.\n
··\n
··\n
```

Result

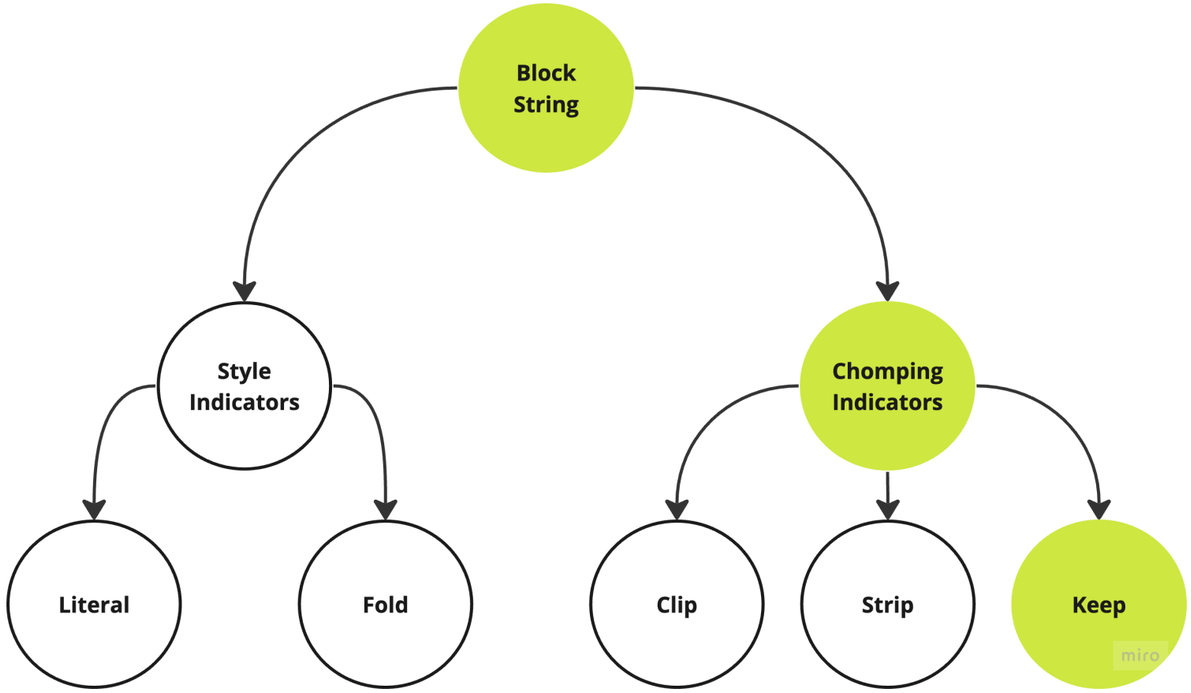
```
Several lines of text,\n
with some "quotes" of various 'types',\n
and also a blank line:\n
\n
and some text with\n
  extra indentation\n
on the next line,\n
plus another line at the end.
```



[1] https://yaml-multiline.info/

# Chomping indicators - keep

- **keep** ( `+` ): retains all newlines at the end



# Keep chomping example

YAML

```yaml
example: |+\n
··Several lines of text,\n
··with some "quotes" of various 'types',\n
··and also a blank line:\n
··\n
··and some text with\n
····extra indentation\n
··on the next line,\n
··plus another line at the end.\n
··\n
··\n
```

Result

```
Several lines of text,\n
with some "quotes" of various 'types',\n
and also a blank line:\n
\n
and some text with\n
  extra indentation\n
on the next line,\n
plus another line at the end.\n
\n
\n
```



[1] https://yaml-multiline.info/

# Dynamic value injection

- **Expressions** allow parsers to dynamically substitute values
- Usage:
  - Environment variables
  - References to other parts of YAML
- Typically represented as `${{ ... }}`

```
database:
  host: ${{ config.database.host }}
  port: ${{ config.database.port }}
  username: ${{ config.database.username }}
  password: ${{ config.database.password }}
```

# Multi-document YAML

- Multiple separate YAML documents within a single file
- Each document contains valid YAML content
- Easy to organize by related documents
- Separate documents by three dashes `---`

```
---
- name: John
  age: 30
---
- name: Jane
  age: 28
---
- name: Bob
  age: 35
  occupation: Developer
```

# Summary

| Name | YAML Symbol |
| --- | --- |
| Literal Style Indicator | \| |
| Fold Style Indicator | > |
| Strip Chomping Indicator | - |
| Keep Chomping Indicator | + |
| Expressions | `${{ ... }}` |
| Multi-document YAML | --- |

# Let's practice!

CI/CD FOR MACHINE LEARNING

# Setting a basic CI pipeline

CI/CD FOR MACHINE LEARNING

**Ravi Bhadauria**
Machine Learning Engineer

# Anatomy of GitHub Actions workflow

```
# Name of the workflow
name: CI
# Events that trigger workflow
on:
  push:
    branches: [ "main" ]
```

# Anatomy of GitHub Actions workflow

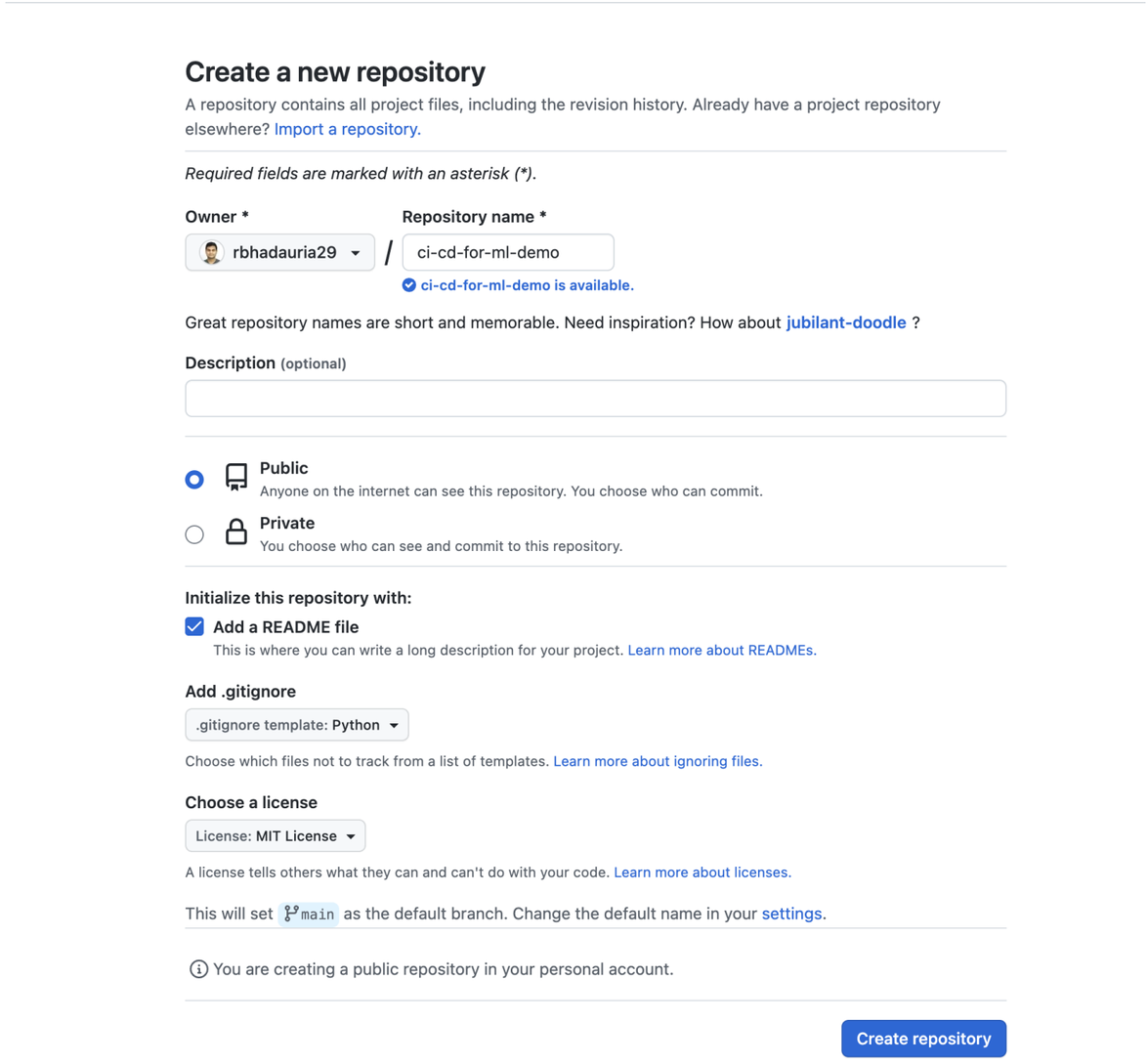
```
# Define jobs and runners
jobs:
  build:
    runs-on: ubuntu-latest
# Define steps
    steps:
      - name: Run a multi-line script
        run: |
          echo Hello, world!
          echo Add other actions to build,
          echo test, and deploy your project.
```

# Anatomy of GitHub Actions workflow

```
name: CI

on:
  push:
    branches: [ "main" ]

jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - name: Run a multi-line script
        run: |
          echo Hello, world!
          echo Add other actions to build,
          echo test, and deploy your project.
```

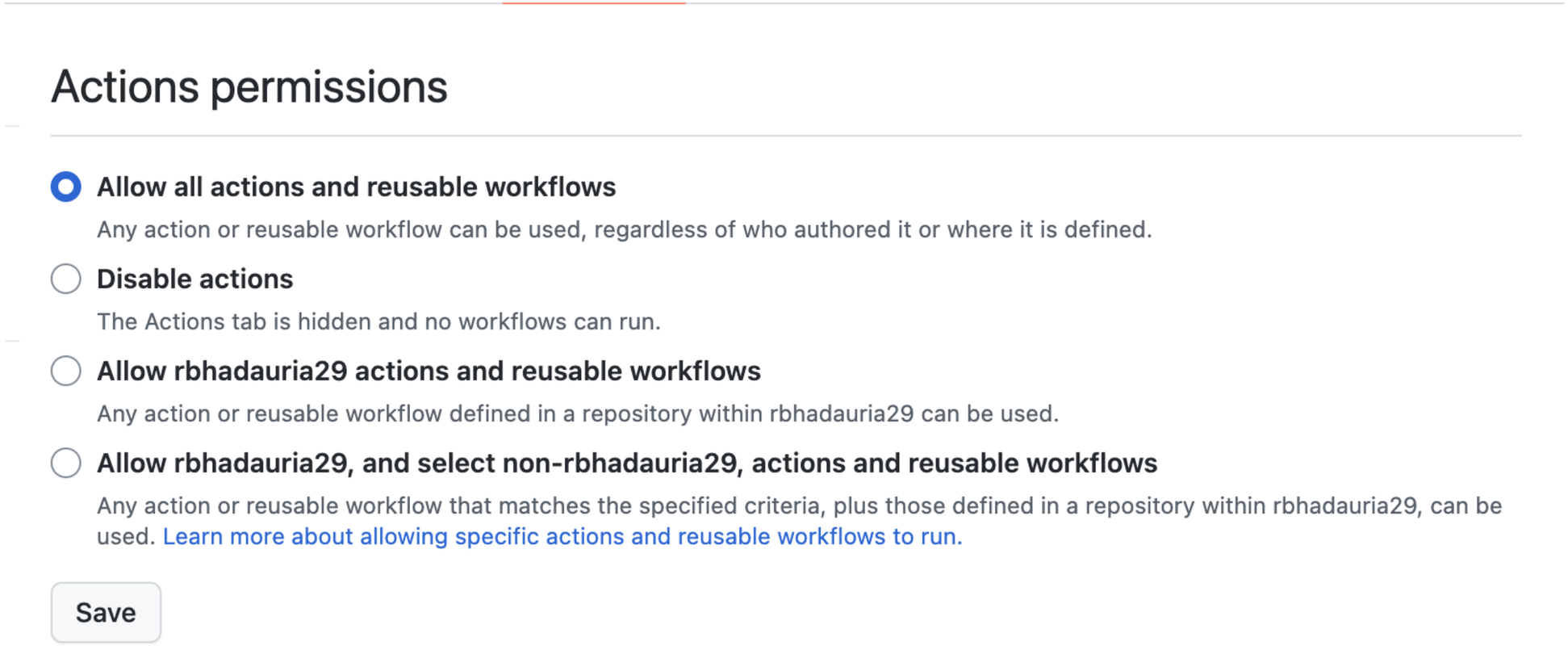
# Create a GitHub repository

- https://github.com/new

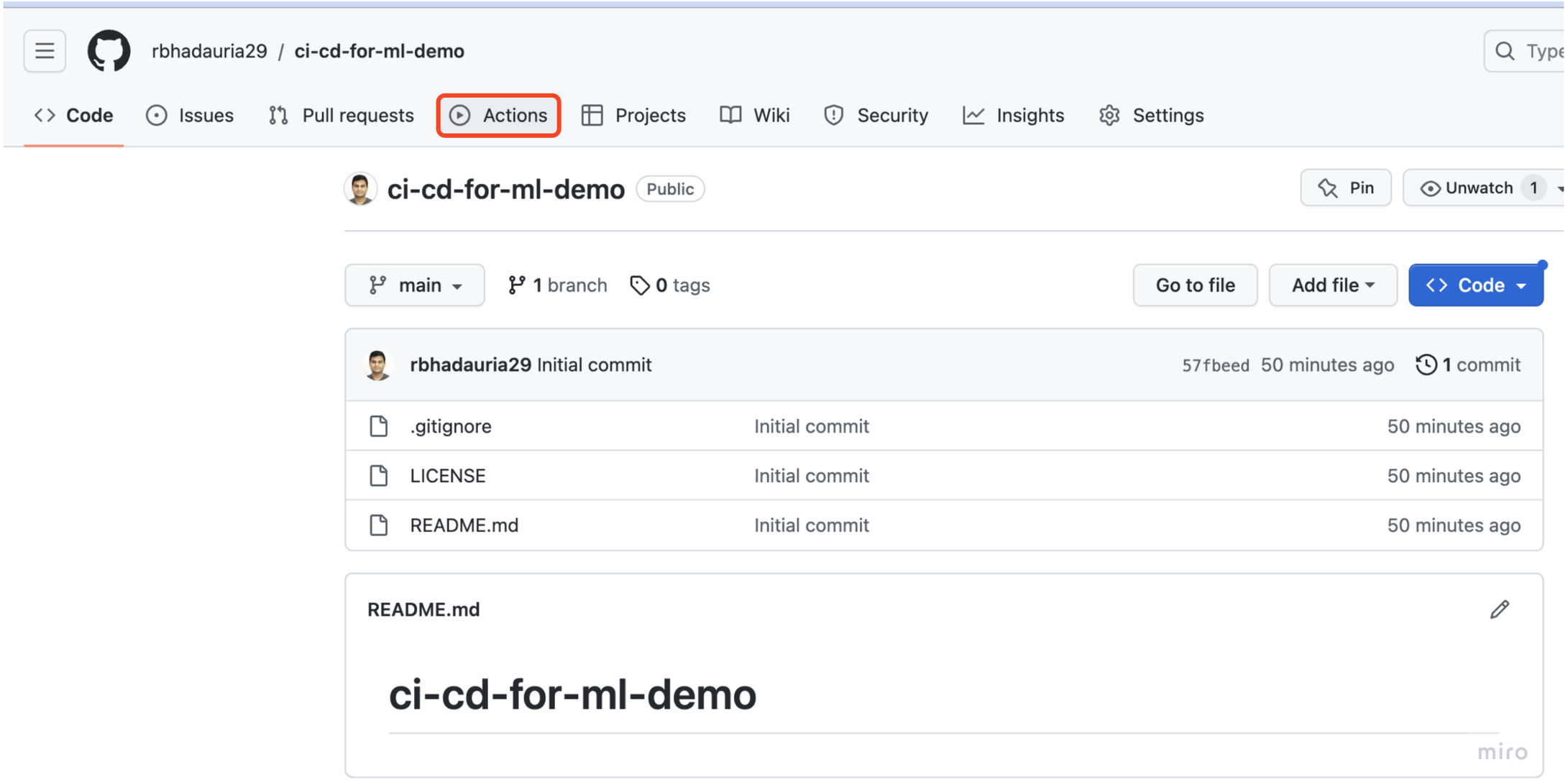


# Set up GitHub Actions

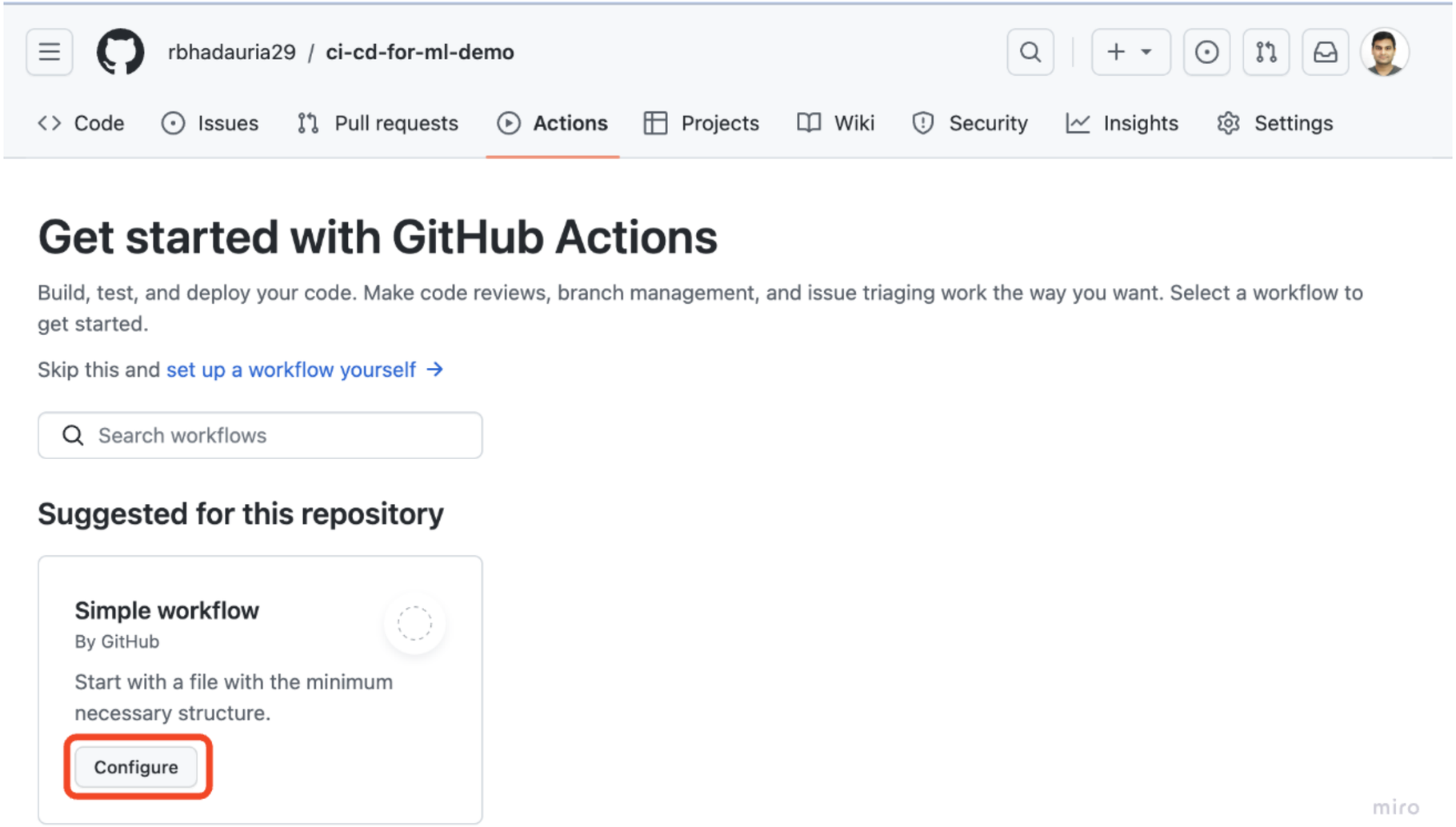
- https://github.com/rbhadauria29/ci-cd-for-ml-demo/settings/actions



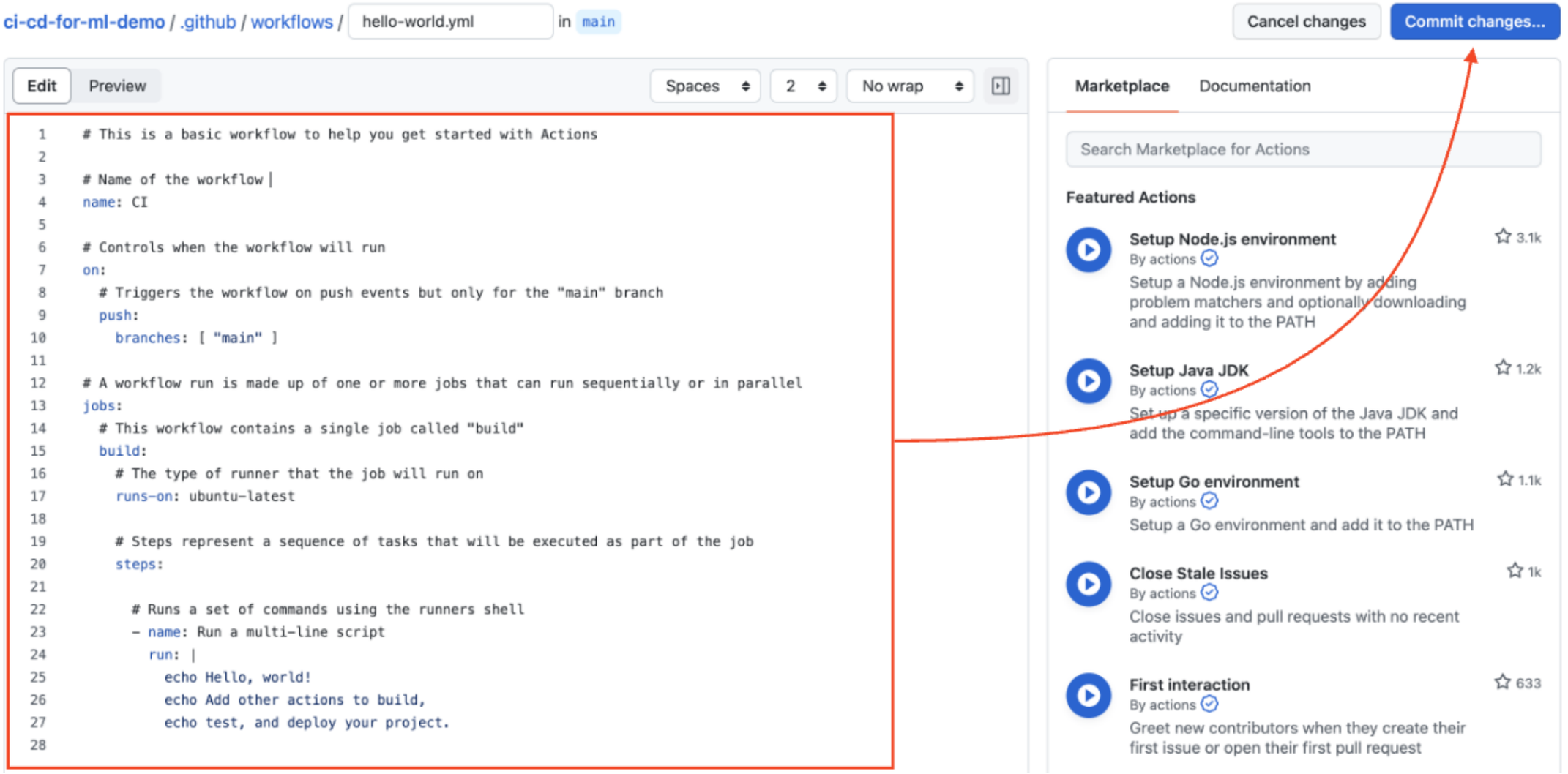
# Setup a "Hello, World!" GitHub Action



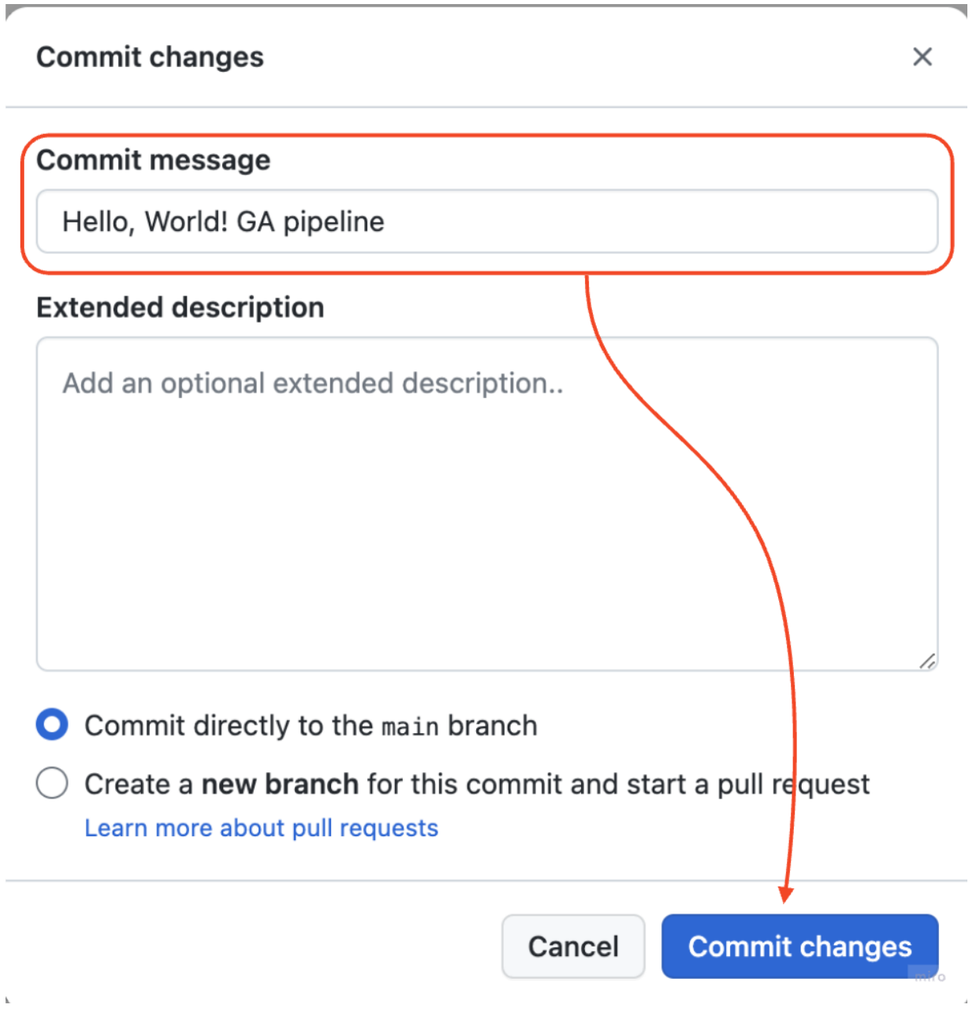
# Setup a "Hello, World!" GitHub Action



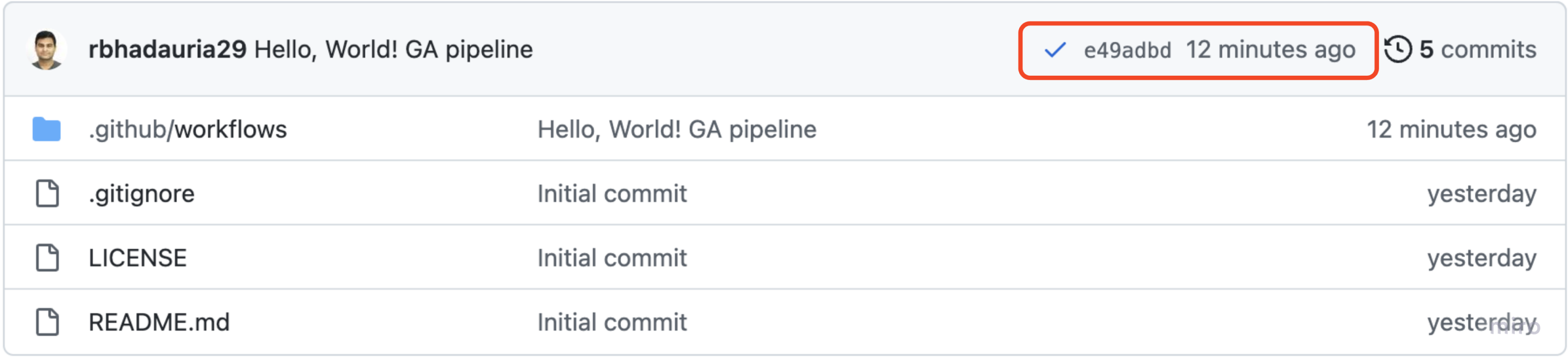
# Setup a "Hello, World!" GitHub Action



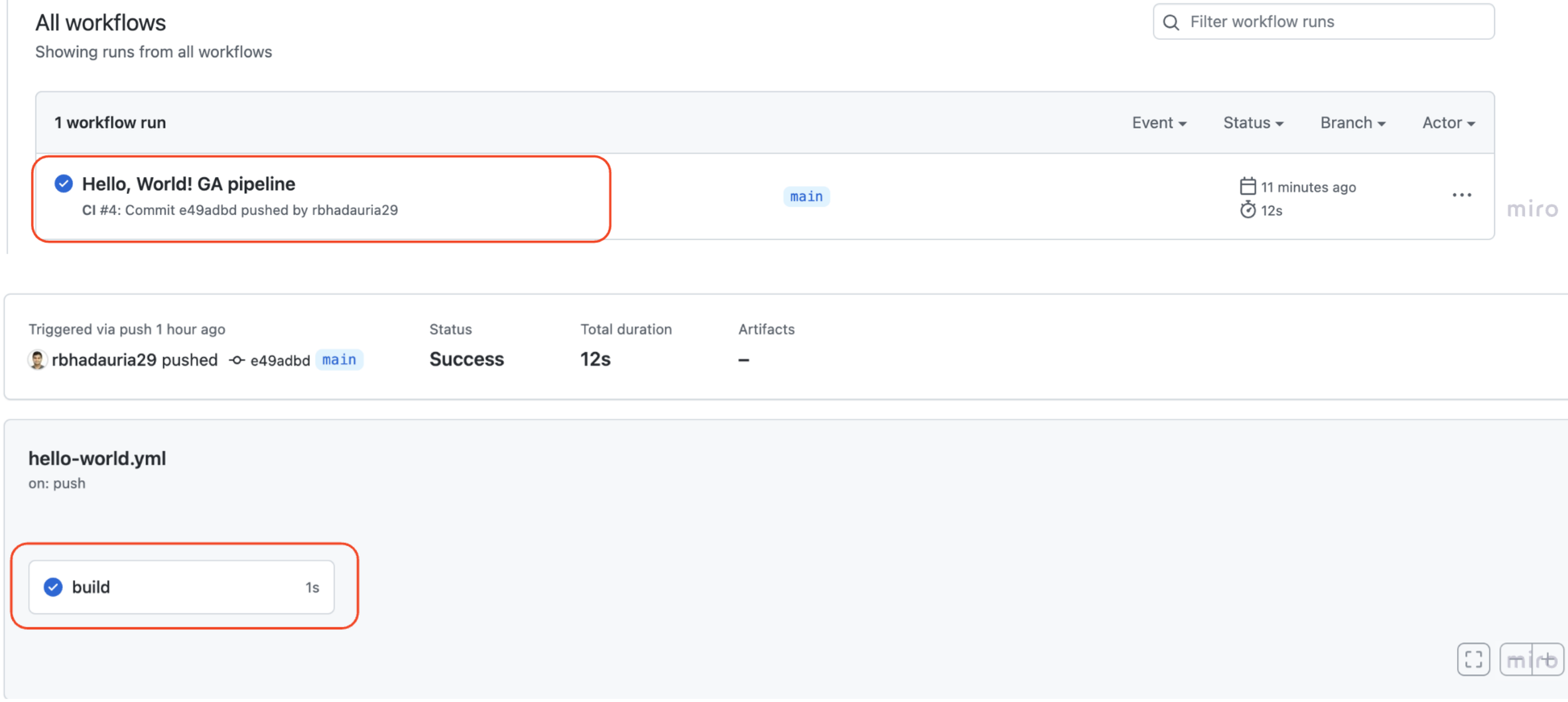
# Setup a "Hello, World!" GitHub Action



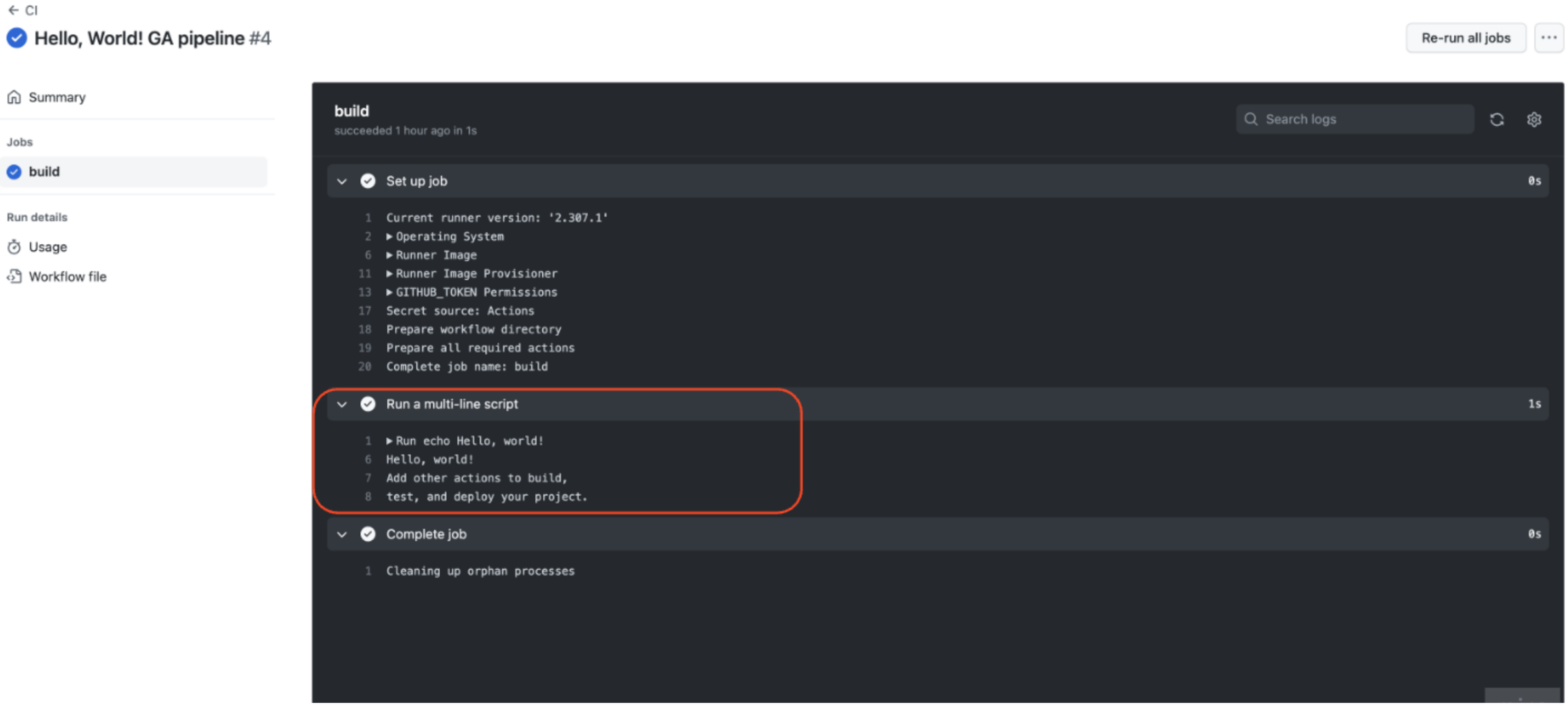
# Inspect the GitHub Actions Run



# Inspect the GitHub Actions Run



# Inspecting the Logs



# Let's practice!

CI/CD FOR MACHINE LEARNING

# Running repository code

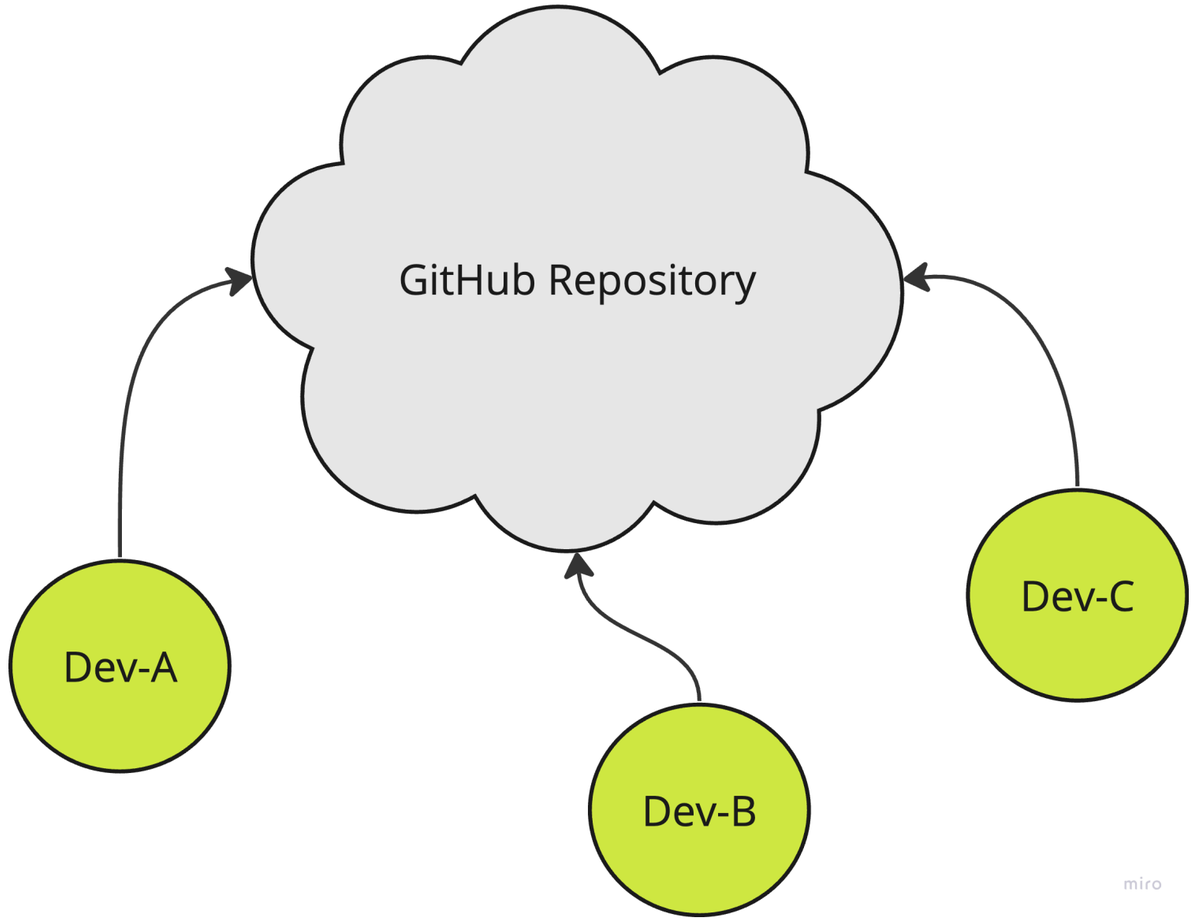
CI/CD FOR MACHINE LEARNING

**Ravi Bhadauria**
Machine Learning Engineer

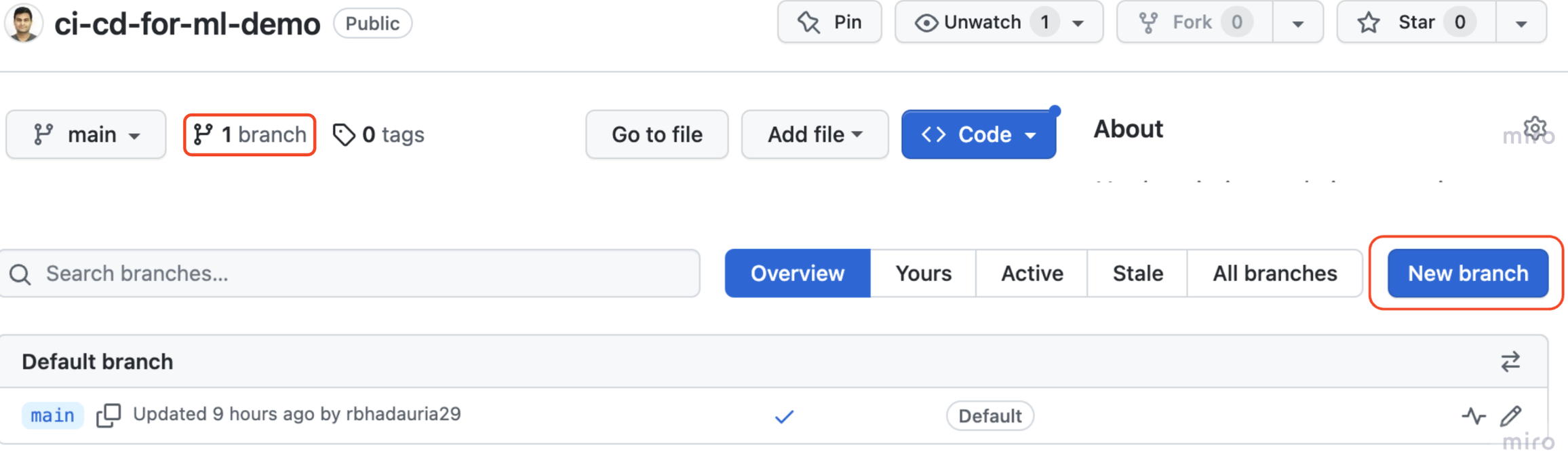
datacamp

# Shared repository model

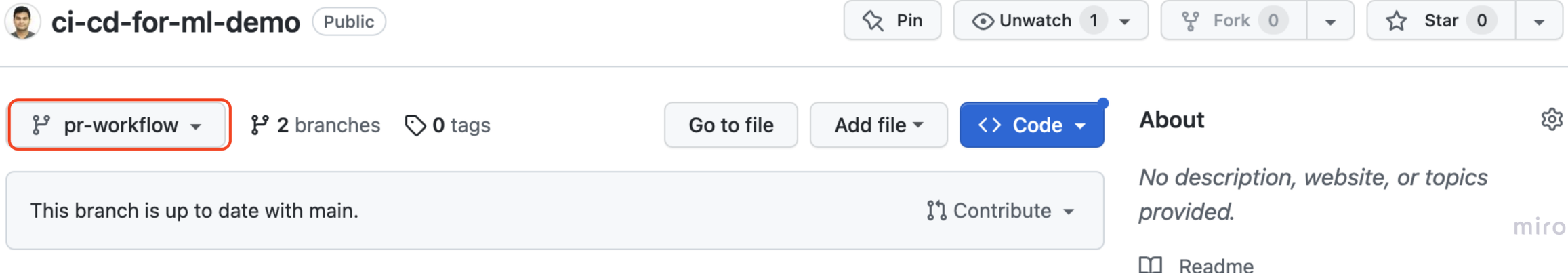
- Multiple collaborators work on same repository
- Feature/Topic branches are created for related work
  - Pull requests are opened
  - Code is reviewed and changes are suggested
  - Branch code is merged into `main` branch
- CI/CD enabled checks improve code quality



# Create a feature branch



# Create a feature branch



# Add repository code

Save as `hello_world.py` in the branch

```
import datetime

def main():
    print("Hello, World!")

    # Get the current time and print it
    current_time = datetime.datetime.now()
    print("Current time:", current_time)

if __name__ == "__main__":
    main()
```

# Configure workflow event

- Specify target branch for pull request event

```
on:
  pull_request:
    branches: [ "main" ]
```

# Actions syntax

```
steps:
  - name: My Action
    uses: <org_or_user_name>/<reponame>@<version_number>
    with:
      action-argument: value
```

- Action repository https://github.com/<org_or_user_name>/<reponame>
  - E.g. https://github.com/actions/checkout
- Arguments can be provided using *with* key
- GitHub marketplace actions at https://github.com/actions

# Configure workflow steps and actions

- Checking out repository in job step

```
steps:
  - name: Checkout
    uses: actions/checkout@v3
```

- Setting up python environment

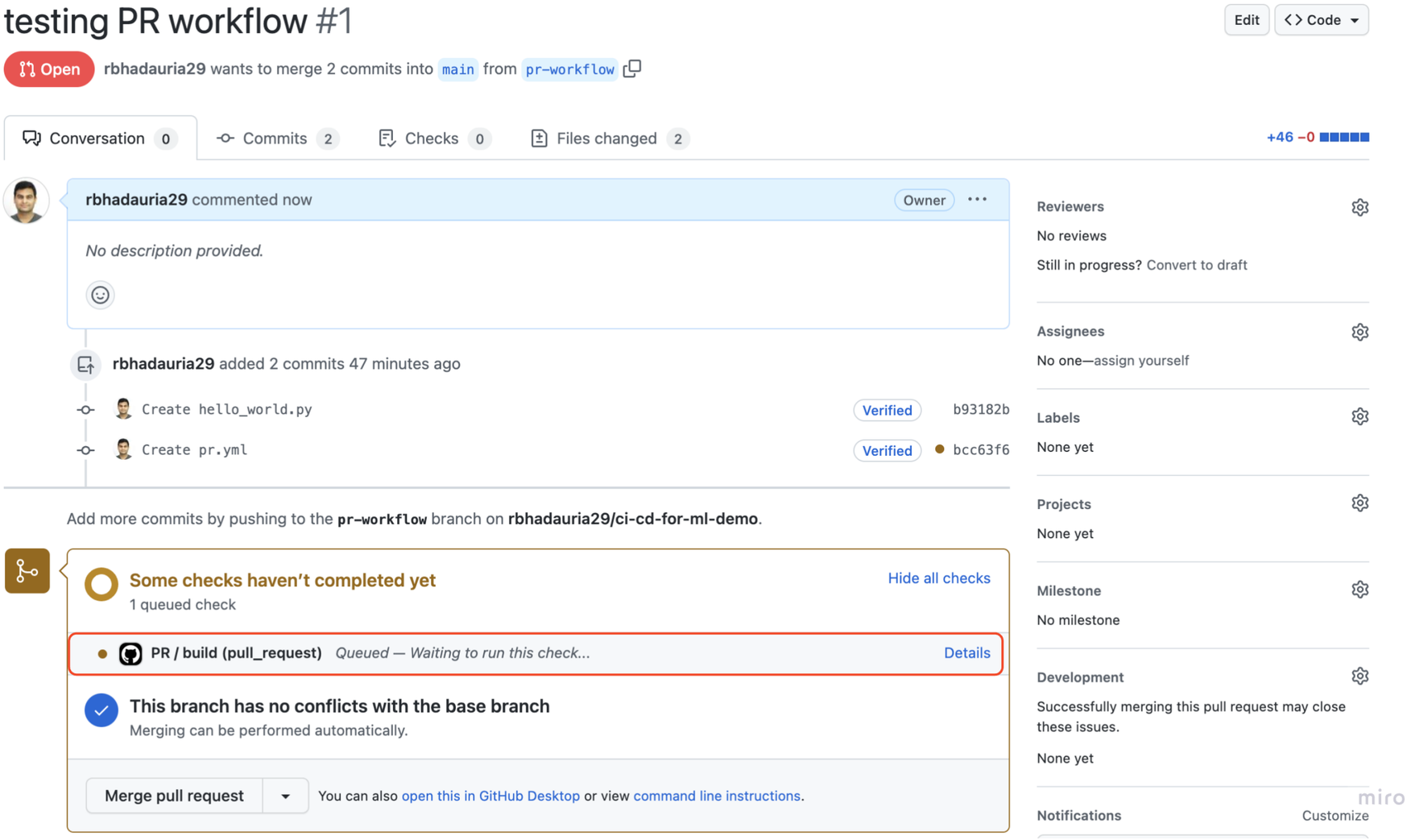
```
steps:
  - name: Setup Python
    uses: actions/setup-python@v4
    with:
      python-version: 3.9
```

# Putting it together

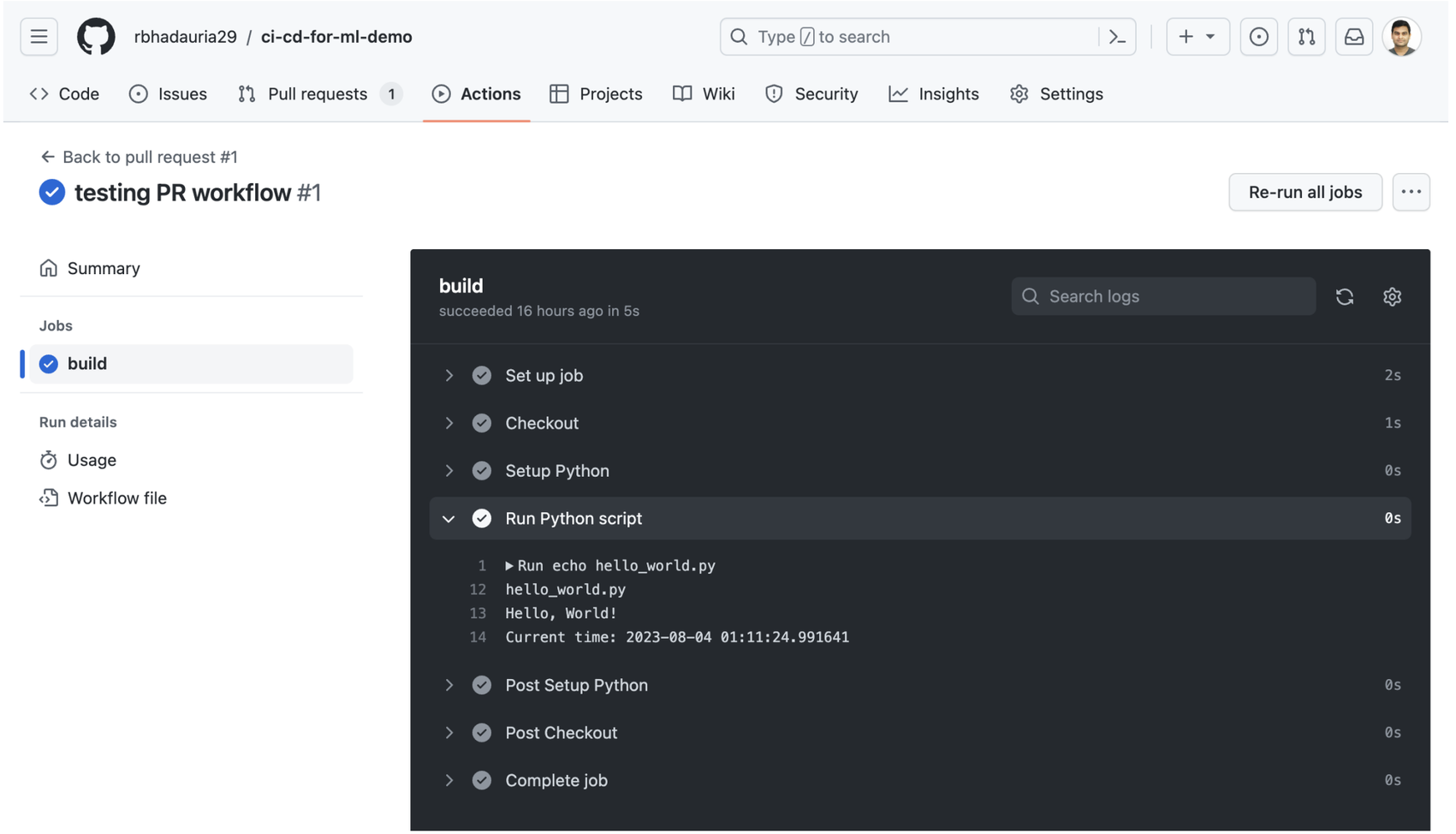
- Workflow triggers on pull request to `main`
- `build` job contains three steps
  - `Checkout` checks out the repository
  - `Setup Python` sets up the Python environment
  - `Run Python script` runs the Python file

```
name: PR

on:
  pull_request:
    branches: [ "main" ]

jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout
        uses: actions/checkout@v3
      - name: Setup Python
        uses: actions/setup-python@v4
        with:
          python-version: 3.9
      - name: Run Python script
        run: |
          echo hello_world.py
          python hello_world.py
```

# Create PR and trigger workflow



# Inspect workflow logs



# Let's practice!

CI/CD FOR MACHINE LEARNING

# Environment Variables and Secrets

CI/CD FOR MACHINE LEARNING

**Ravi Bhadauria**
Machine Learning Engineer

datacamp

# Contexts

- Access information about predefined variables and data
  - workflow runs
  - variables
  - runner environments
  - jobs and steps
- Access contexts using the expression syntax `${{ context.XXX }}`

- Contexts used in this course
  - `github` : information about the workflow run
  - `env` : variables set in the workflow
  - `secrets` : names and values that are available to workflow
  - `job` : info about the current job
  - `runner` : info about the machine

[1] https://docs.github.com/en/actions/learn-github-actions/contexts

# Variables

- Store non-sensitive information in plain text
  - compiler flags, usernames, file paths
- Declared as value for `env` key
- Global/local scope is controlled by the level where defined
- Accessed from the `env` context as `${{ env.ENV_VAR }}`

```
name: Greeting on variable day
# Global env
env:
  Greeting: Hello
jobs:
  greeting_job:
    runs-on: ubuntu-latest
    # Local env: scoped to greeting_job
    env:
      First_Name: Ravi
    steps:
      - run: |
          echo "${{ env.Greeting }} \
          ${{ env.First_Name }}."
```

# Secrets

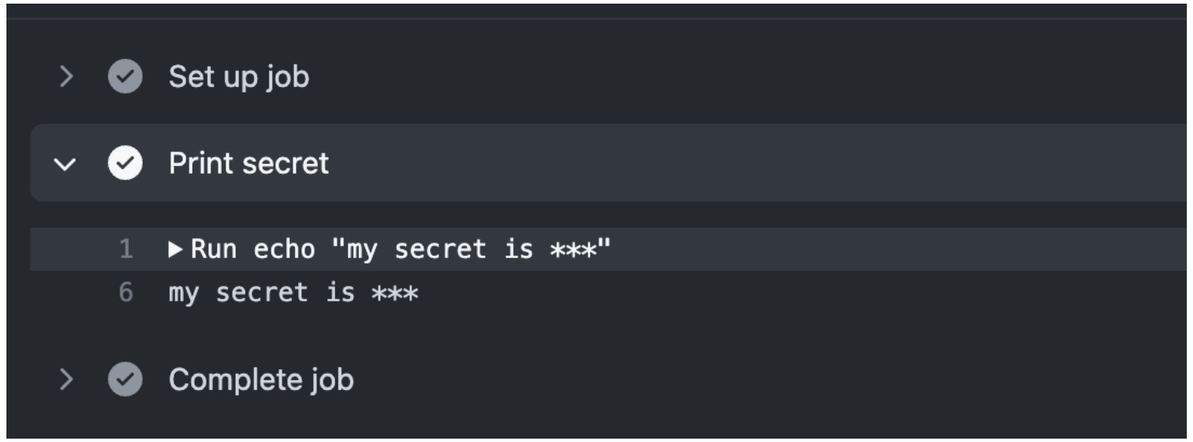
- Store sensitive information in encrypted manner
  - passwords, API keys
- Access values via `secrets` context
  - `${{ secrets.SuperSecret }}`
- Can store them as input or environment variable

```
steps:
  - name: Hello world action
    env: # Set the secret as an env var
      super_secret: ${{ secrets.SuperSecret }}
    with: # Or as an input
      super_secret: ${{ secrets.SuperSecret }}
```

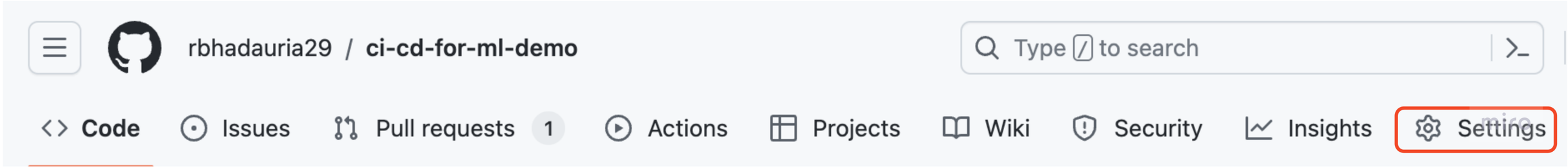
- Printing secret

```
steps:
- name: Print secret
  run: |
    echo "my secret is \
    ${{ secrets.SuperSecret }}"
```

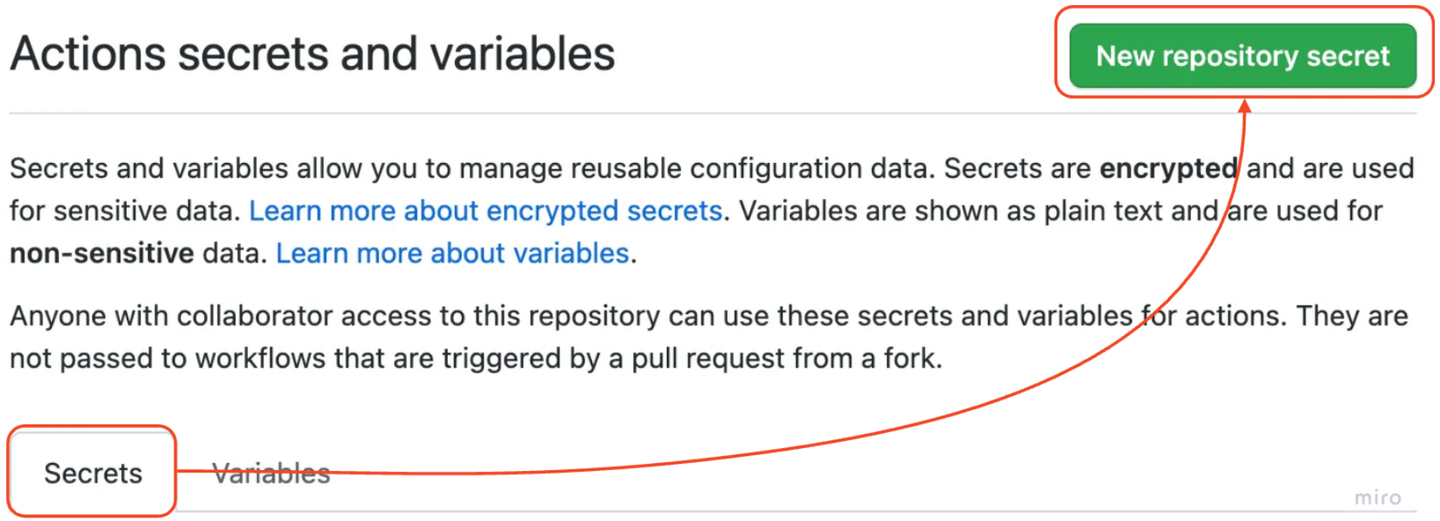
- Output



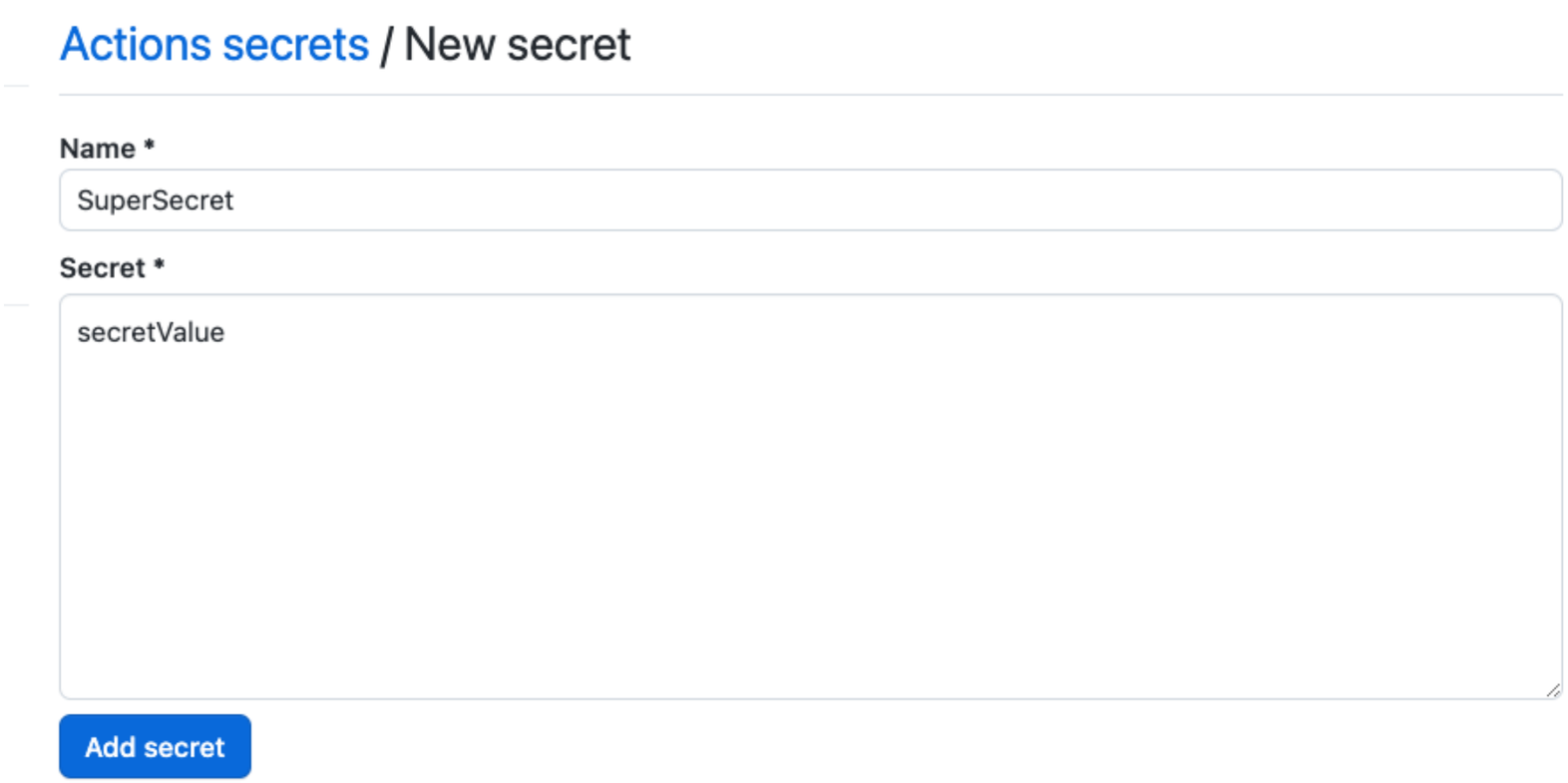
# Setting secrets




- Go to **Security > Secrets and Variables > Actions**



# Setting secrets



# GITHUB_TOKEN secret

- Built in secret provided by GitHub Actions
- Used to perform workflow actions
  - Cloning the repository and fetching code
  - Opening and closing issues and pull requests
  - Commenting on issues and pull requests
- Automatically available in every GitHub Actions workflow
  - Accessed via `${{ secrets.GITHUB_TOKEN }}`
- Permissions can be tuned to the right degree

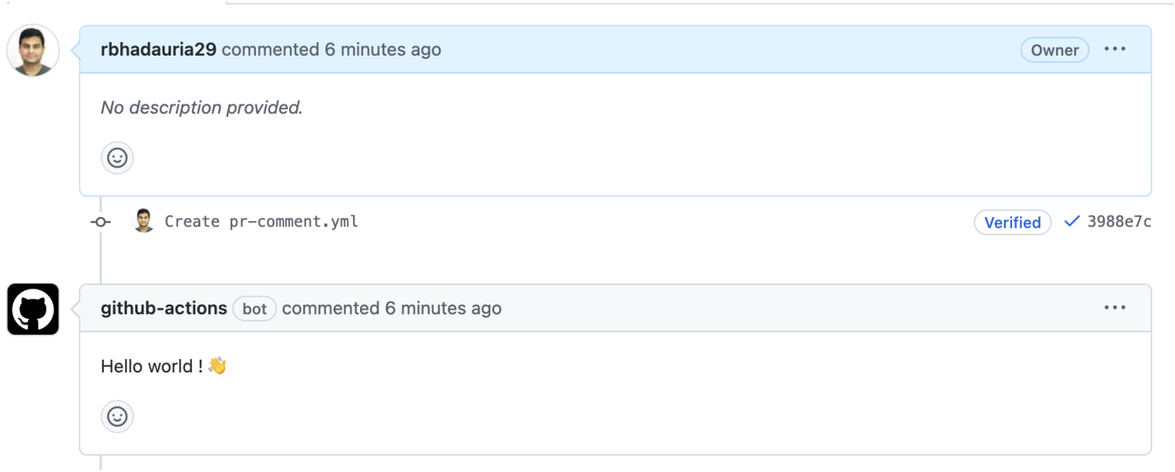
# Example: commenting on a pull request

- Grant permissions to write comments in PR

```
permissions:
  pull-requests: write
```

- Use GITHUB_TOKEN to authorize

```
permissions:
  pull-requests: write
steps:
  - name: Comment PR
    uses: thollander/actions-comment-pull-request@v2
    with:
      GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
      message: |
        Hello world ! :wave:
```




[1] https://gist.github.com/rbhadauria29/6d7fc51944b4fb48425c3c307fec77c6

# Let's practice!

CI/CD FOR MACHINE LEARNING