# Introduction to CML

Continuous Machine Learning

# References

- **Continuous Machine Learning (CML) is CI/CD for Machine Learning Projects**
  - **Cml.dev**
- **Configuring Github Workflows**
  - https://docs.github.com/en/actions/configuring-and-managing-workflows/configuring-a-workflow
  - https://docs.github.com/en/actions/reference/workflow-syntax-for-github-actions#jobsjob_idsteps

# Tools

- Linode.com - $5 a month Linux instance
- Codeanywhere.com - Cloud Integrated Development Environment

# What is CML?

- Continuous Machine Learning (CML) is an open-source library for implementing continuous integration & delivery (CI/CD) in machine learning projects.
- Use CML to automate parts of your development workflow, including model training and evaluation, comparing ML experiments across your project history, and monitoring changing datasets.
- On every pull request, CML helps you automatically train and evaluate models, then generates a visual report with results and metrics.

# CML in Use

Workspace vs. Master

Training metrics

| Path | Param | Old | New |
|---|---|---|---|
| params.yaml | iter | 1 | 30 |

GPU info

| | Value |
|---|---|
| index | 0 |
| type | Tesla M60 |
| uuid | GPU-ed845738-2b56-4a07-e18a-cc4181e25fcb |
| mem_used | 7422 |

On every pull request, CML helps you automatically train and evaluate models, then generates a visual report with results and metrics.

To the left, an example report for a neural style transfer model.

# CML Principles

- GitFlow for data science. Use GitLab or GitHub to manage ML experiments, track who trained ML models or modified data and when. Codify data and models with DVC instead of pushing to a Git repo.
- Auto reports for ML experiments. Auto-generate reports with metrics and plots in each Git Pull Request. Rigorous engineering practices help your team make informed, data-driven decisions.
- No additional services. Build your own ML platform using just GitHub or GitLab and your favorite cloud services: AWS, Azure, GCP. No databases, services or complex setup needed.

# Workflow Actions and GitHub

- Workflows are custom automated processes that you can set up in your repository to build, test, package, release, or deploy any project on GitHub.
- With workflows you can automate your software development life cycle with a wide range of tools and services.
- Workflow files use YAML syntax, and must have either a .yml or .yaml file extension.

# YAML

- YAML is an acronym for 'Yet Another Markup Language', but is now more commonly referred to as '*YAML* Ain't Markup Language' … think GNU is not Unix.
- YAML is a data-orientated, human readable, serializable language.
- YAML borrows from other languages.
  - Scalars, lists, and associative arrays are based on Perl.
  - The document separator "---" is based on MIME.
  - Escape sequences are based on C.
  - Whitespace wrapping is based on HTML.
- YAML relies on indentation to group common elements and provides data hierarchy.
  - NOTE: YAML indentation uses [spaces], not [tabs].
  - NOTE: Spaces must be placed between all keys and values as well.

# YAML Sample

```
--- !clarkevans.com/^invoice
invoice: 34843
date   : 2001-01-23
bill-to: &id001
    given  : Chris
    family : Dumars
    address:
        lines: |
            458 Walkman Dr.
            Suite #292
        city    : Royal Oak
        state   : MI
        postal  : 48046
ship-to: *id001
product:
    - sku         : BL394D
      quantity    : 4
      description : Basketball
      price       : 450.00
    - sku         : BL4438H
      quantity    : 1
      description : Super Hoop
      price       : 2392.00
tax  : 251.42
total: 4443.52
comments: >
    Late afternoon is best.
    Backup contact is Nancy
    Billsmer @ 338-4338.
```

SCALAR

COLLECTIONS

MULTI-LINE COLLECTIONS

LISTS/DICTIONARIES

MULTI-LINE FORMATTING

https://www.codeproject.com/Articles/1214409/Learn-YAML-in-five-minutes

# Scalers

- Scalars, or variables, are defined using a colon and a space.

```
integer: 25
string: "25"
float: 25.0
boolean: Yes
```

# Lists

- Associative arrays and lists can be defined using a conventional block format (or an inline format that is similar to JSON).
- --- # Shopping List in Block Format
  - milk
  - eggs
  - Juice
- --- # Shopping List in Inline Format

  [milk, eggs, juice]

# CML Workflow Sample

- This is the gist of the CML workflow: when you push changes to your GitHub repository, the workflow in your `.github/workflows/cml.yaml` file gets run and a report generated.
- The CML functions let you display relevant results from the workflow, like model performance metrics and vizualizations, in GitHub checks and comments. What kind of workflow you want to run, and want to put in your CML report, is up to you.

# Workflow Action

```yaml
name: model-training
on: [push]
jobs:
  run:
    runs-on: [ubuntu-latest]
    container: docker://dvcorg/cml-py3:latest
    steps:
      - uses: actions/checkout@v2
      - name: cml_run
        env:
          repo_token: ${{ secrets.GITHUB_TOKEN }}
        run: |
          pip install -r requirements.txt
          python train.py

          cat metrics.txt >> report.md
          cml-publish confusion_matrix.png --md >> report.md
          cml-send-comment report.md
```

# Line-by-Line

- Next, we will break the CML sample script down, line by line.

# Name

```
name: model-training
```

- The name of your workflow.
- GitHub displays the names of your workflows on your repository's actions page.
- If you omit `name`, GitHub sets it to the workflow file path relative to the root of the repository.

# On <triggering event>

```
on: [push]
```

- The name of the GitHub event that triggers the workflow.
- You can provide a single event `string`, `array` of events, `array` of event `types`, or an event configuration `map` that schedules a workflow or restricts the execution of a workflow to specific files, tags, or branch changes.
- This is a required field and there are many events that can trigger a workflow.
- Trigger the workflow on push or pull request
- on: [push, pull_request]

# Workflow Events

- check_run
- check_suite
- create
- delete
- deployment
- deployment_status
- fork
- gollum
- issue_comment
- issues
- label
- milestone
- page_build
- project

project_card
project_column
public
pull_request
pull_request_review
pull_request_review_comment
pull_request_target
push
registry_package
release
status
watch
workflow_run

# Jobs

```
jobs:
```

- A workflow run is made up of one or more jobs. Jobs run in parallel by default. To run jobs sequentially, you can define dependencies on other jobs using the `jobs.<job_id>.needs` keyword.
- Each job runs in an environment specified by `runs-on`.
- You can run an unlimited number of jobs as long as you are within the workflow usage limits.

# Run

`run:`

- Runs command-line programs using the operating system's shell. If you do not provide a `name`, the step name will default to the text specified in the `run` command.
- Commands run using non-login shells by default. You can choose a different shell and customize the shell used to run commands.
- Each `run` keyword represents a new process and shell in the runner environment. When you provide multi-line commands, each line runs in the same shell.

# Runs-on

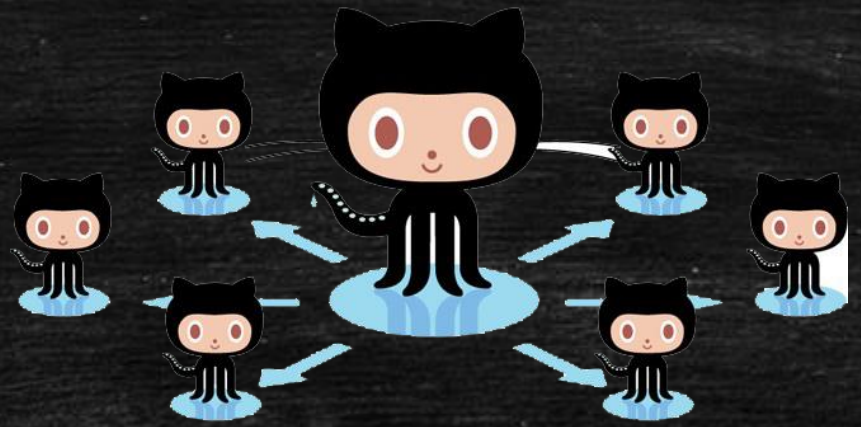
```
runs-on: [ubuntu-latest]
```

- The underlying virtual machine to run the job on. The machine can be either a GitHub-hosted runner, or a self-hosted runner. This machine may also host a (Docker) container if specified.

| Virtual environment | YAML workflow label |
| --- | --- |
| Windows Server 2019 | windows-latest or windows-2019 |
| Ubuntu 20.04 | ubuntu-20.04 |
| Ubuntu 18.04 | ubuntu-latest or ubuntu-18.04 |
| Ubuntu 16.04 | ubuntu-16.04 |
| macOS Catalina 10.15 | macos-latest or macos-10.15 |

# Runners!

- The runner is the application that runs a job from a GitHub Actions workflow. The runner can run on the hosted machine pools or run on self-hosted environments.



# GitHub Hosted Runners

- Receive automatic updates for the operating system, preinstalled packages and tools, and the self-hosted runner application.
- Are managed and maintained by GitHub.
- Provide a clean instance for every job execution.
- Use free minutes on your GitHub plan, with per-minute rates applied after surpassing the free minutes.

# Self-Hosted Runners

- Receive automatic updates for the self-hosted runner application only. You are responsible updating the operating system and all other software.
- Can use cloud services or local machines that you already pay for.
- Are customizable to your hardware, operating system, software, and security requirements.
- Don't need to have a clean instance for every job execution.
- Are free to use with GitHub Actions, but you are responsible for the cost of maintaining your runner machines.

# Self-Hosted Runners

- Let's dive into self-hosted runners a bit more...

# Self-Hosted Runner Requirements

- You can install and run the self-hosted runner application on the machine.
- The machine can communicate with GitHub Actions.
- The machine has enough hardware resources for the type of workflows you plan to run. The self-hosted runner application itself only requires minimal resources.
- **Note:** If you want to run workflows that use Docker container actions or service containers, you must use a Linux machine and Docker must be installed.

# Self-Hosted Supported Operating Systems

- Linux
  - Red Hat Enterprise Linux 7
  - CentOS 7
  - Oracle Linux 7
  - Fedora 29 or later
  - Debian 9 or later
  - Ubuntu 16.04 or later
  - Linux Mint 18 or later
  - openSUSE 15 or later
  - SUSE Enterprise Linux (SLES) 12 SP2 or later
- Windows
  - Windows 7 64-bit
  - Windows 8.1 64-bit
  - Windows 10 64-bit
  - Windows Server 2012 R2 64-bit
  - Windows Server 2016 64-bit
  - Windows Server 2019 64-bit
- MacOS
  - macOS 10.13 (High Sierra) or later

# Self-Hosted Runner Communication

- The self-hosted runner polls GitHub to retrieve application updates and to check if any jobs are queued for processing.
- The self-hosted runner uses a HTTPS *long poll* that opens a connection to GitHub for 50 seconds, and if no response is received, it then times out and creates a new long poll.
- The application must be running on the machine to accept and run GitHub Actions jobs.

# container

```
container: docker://dvcorg/cml-py3:latest
```

- The Docker image to use as the container to run the action. The value can be the Docker Hub image name or a public docker registry name.
- If you do not set a `container`, all steps will run directly on the host specified by `runs-on` unless a step refers to an action configured to run in a container.
- We will look at the docker container in the next slide…

# Docker Hub

- Docker Hub is the world's largest library and community for container images
- Browse over 100,000 container images from software vendors, open-source projects, and the community.
- Let's look at Docker Hub...
  - https://hub.docker.com/search?q=dvcorg%2Fcml-py3&type=image

# Review

- The *runs-on* statement defines the runner.
- The runner may be self-hosted or hosted (typically a virtual machine).
- The runner hosts a *container.*

```
runs-on: [ubuntu-latest]
container: docker://dvcorg/cml-py3:latest
```

# steps

`steps:`

- A job contains a sequence of tasks called `steps`.
- Steps can run commands, run setup tasks, or run an action in your repository, a public repository, or an action published in a Docker registry.
- Not all steps run actions, but all actions run as a step. Each step runs in its own process in the runner environment and has access to the workspace and filesystem.
- Because steps run in their own process, changes to environment variables are not preserved between steps. GitHub provides built-in steps to set up and complete a job.

# uses

```
- uses: actions/checkout@v2
```

- Selects an action to run as part of a step in your job. An action is a reusable unit of code. You can use an action defined in the same repository as the workflow, a public repository, or in a published Docker container image.
- The *checkout* action is specified in this script:
  - https://github.com/marketplace/actions/checkout

# name

```
- name: cml_run
```

- The name of the job displayed on GitHub.

# env

```
env:
```

- Identifies a section in the script which is a `map` of environment variables that are available to all jobs and steps in the workflow.
- The term *map* is typical when describing key/value pairs. (Think Javascript objects)

# Repro_token

```
repo_token: ${{ secrets.GITHUB_TOKEN }}
```

- GitHub provides a token that you can use to authenticate on behalf of GitHub Actions.
- GitHub automatically creates a `GITHUB_TOKEN` secret to use in your workflow. You can use the `GITHUB_TOKEN` to authenticate in a workflow run.
- Before each job begins, GitHub fetches an installation access token for the job. The token expires when the job is finished.
- To use the `GITHUB_TOKEN` secret, you must reference it in your workflow file. Using a token might include passing the token as an input to an action that requires it, or making authenticated GitHub API calls.

# run

```
run: |
```

- Runs command-line programs using the operating system's shell.
- Run followed by a pipe '|' designates a multi-line command.

```
- name: Clean install dependencies and build
  run: |
    npm ci
    npm run build
```

# pip

```
pip install -r requirements.txt
```

- `pip install -r requirements.txt`
  - installs the libraries listed in the `requirements.txt` file

3 lines (3 sloc) | 30 Bytes

```
1  setuptools
2  sklearn
3  matplotlib
```

# python

```
python train.py
```

- This command runs the training code contained in train.py

# cat

```
cat metrics.txt >> report.md
```

- On Unix-like operating systems, the cat command reads data from files, and outputs their contents.
- It is the simplest way to display the contents of a file at the command line.
- The ">>" operator appends the output to the "report.md" file.

# Cml-publish

```
cml-publish confusion_matrix.png --md >> report.md
```

- Publish an image for writing to CML report.
- The library comes pre-installed on the CML docker image.
- In the above example, note the field `container: docker://dvcorg/cml-py3:latest` specifies the CML Docker image with Python 3 will be pulled by the GitHub Actions runner.
- The next slide lists all of the CML functions.

# CML Functions

- CML provides a number of helper functions to help package outputs from ML workflows, such as numeric data and data vizualizations about model performance, into a CML report.

| Function | Description | Inputs |
|---|---|---|
| cml-send-comment | Return CML report as a comment in your GitHub/GitLab workflow. | <path to report> --head-sha <sha> |
| cml-send-github-check | Return CML report as a check in GitHub | <path to report> --head-sha <sha> |
| cml-publish | Publish an image for writing to CML report. | <path to image> --title <image title> --md |
| cml-tensorboard-dev | Return a link to a Tensorboard.dev page | --logdir <path to logs> --title <experiment title> --md |

# Cml-send-comment

```
cml-send-comment report.md
```

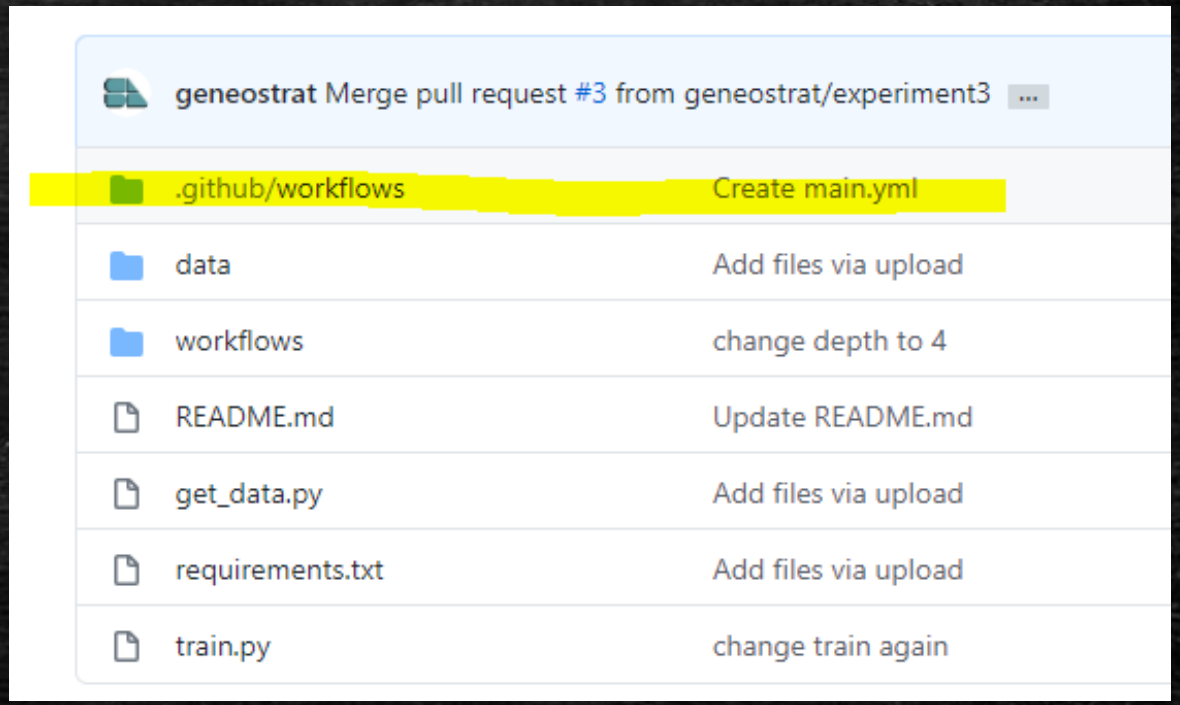
- Sends the resulting report in email.

# Let's Configure the Workflow!

- The following slides step through the process of installing the workflow action.

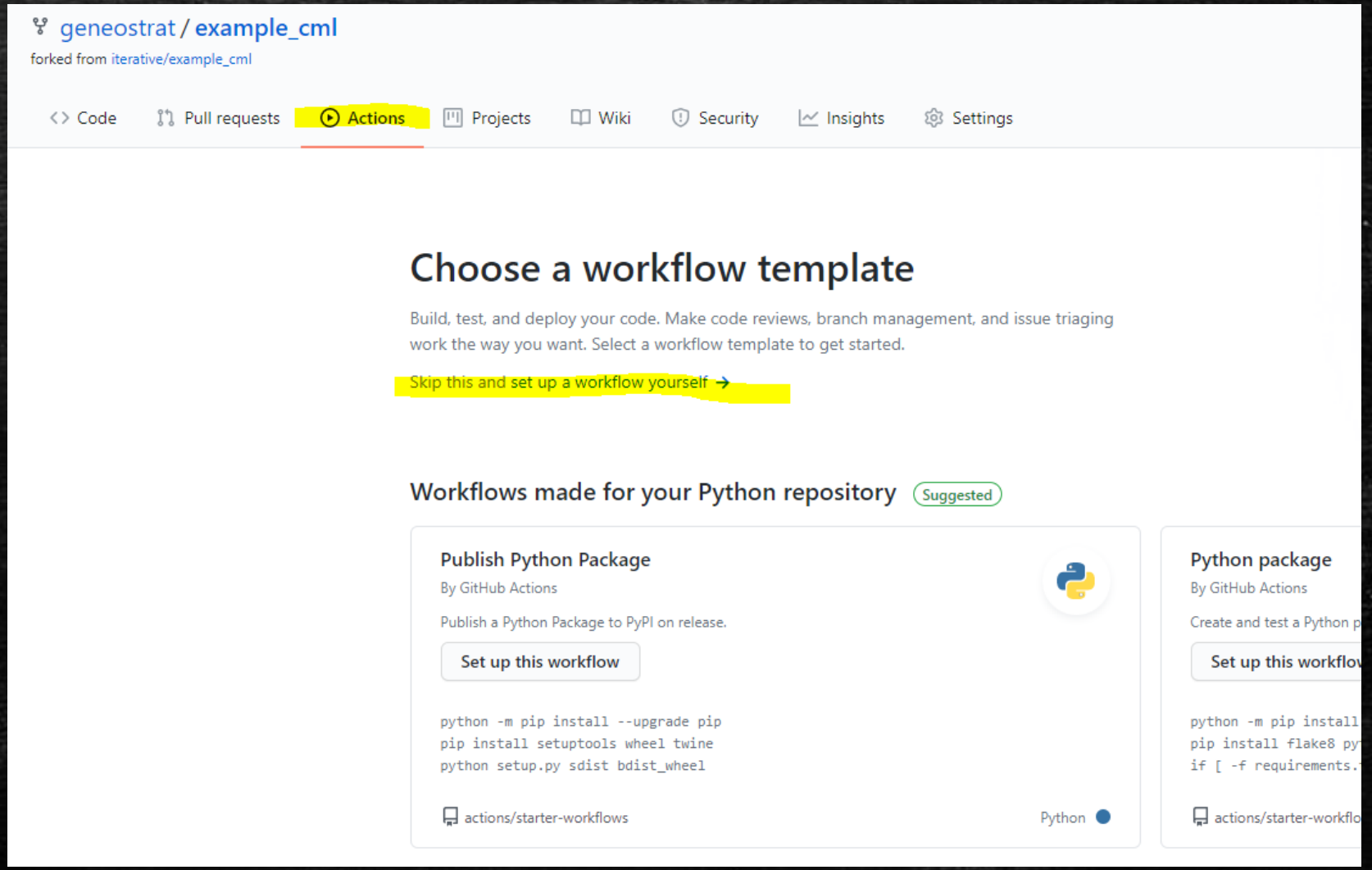
# Where Workflows Live

- You can create more than one workflow in a repository. You must store workflows in the `.github/workflows` directory in the root of your repository.



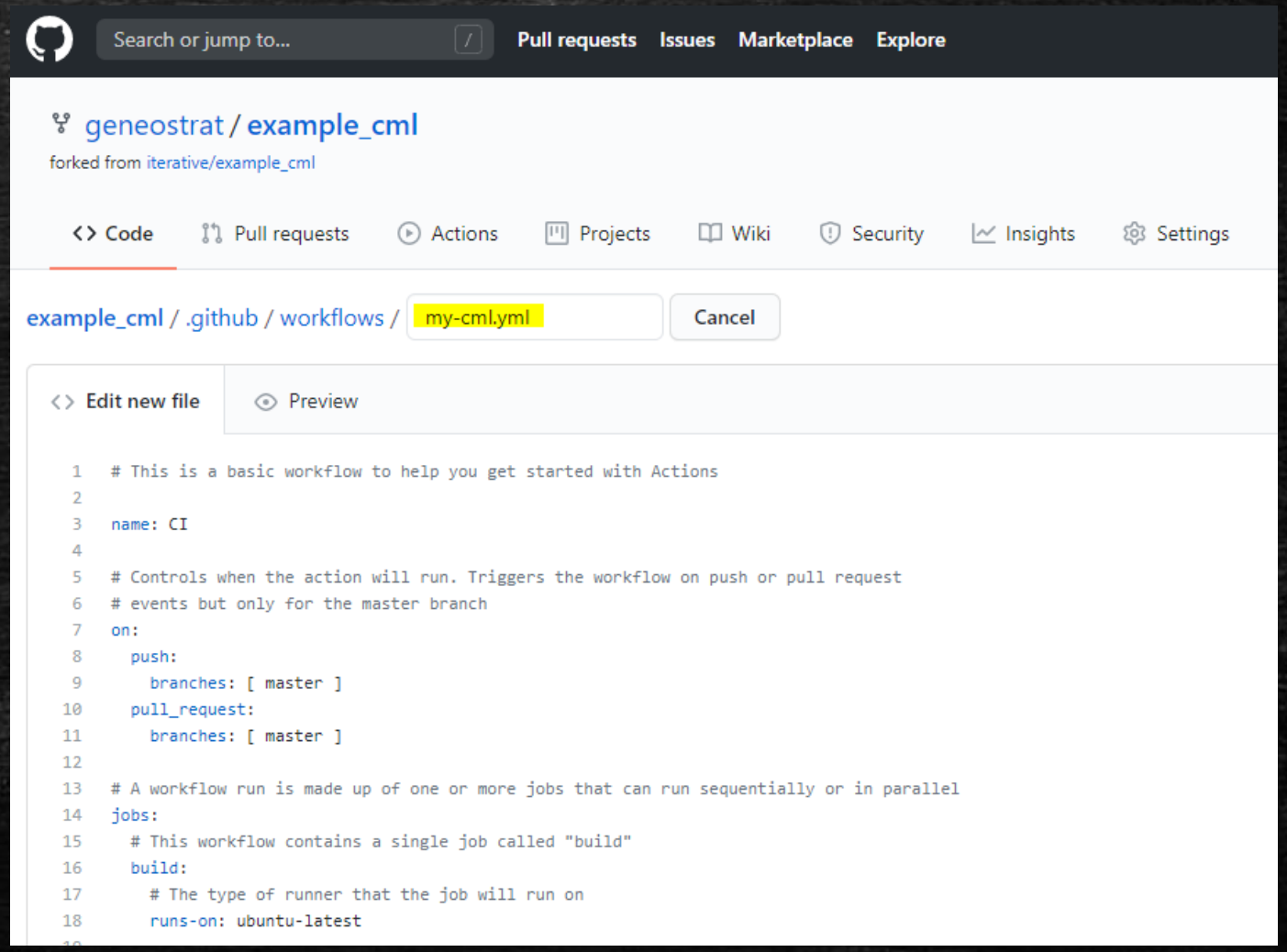
# Choose Workflow Template

- Setup the workflow template yourself.



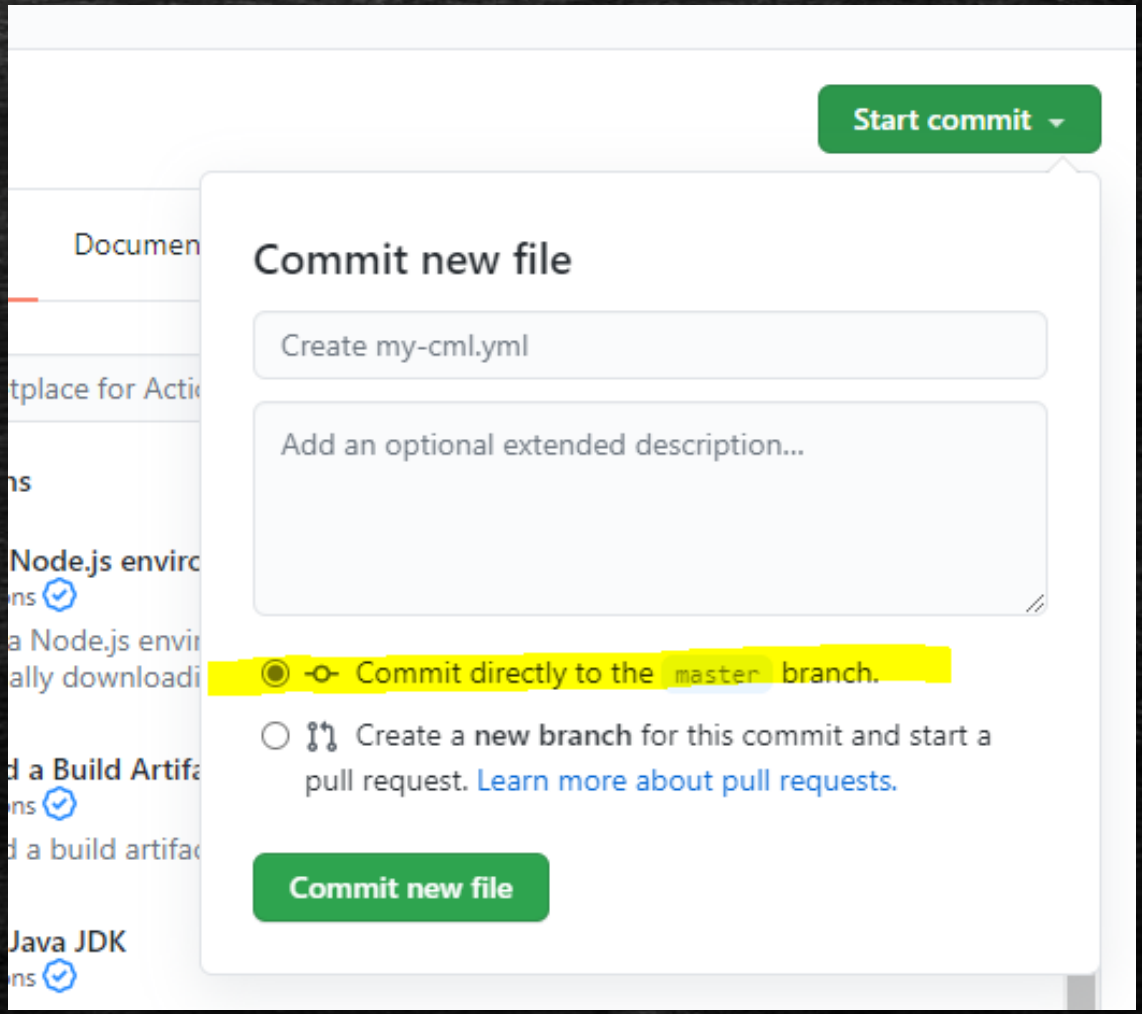
# Create the Action File

- Provide a name for your YML file.
- Copy the sample CML action script or write your own.



# Commit the Action File

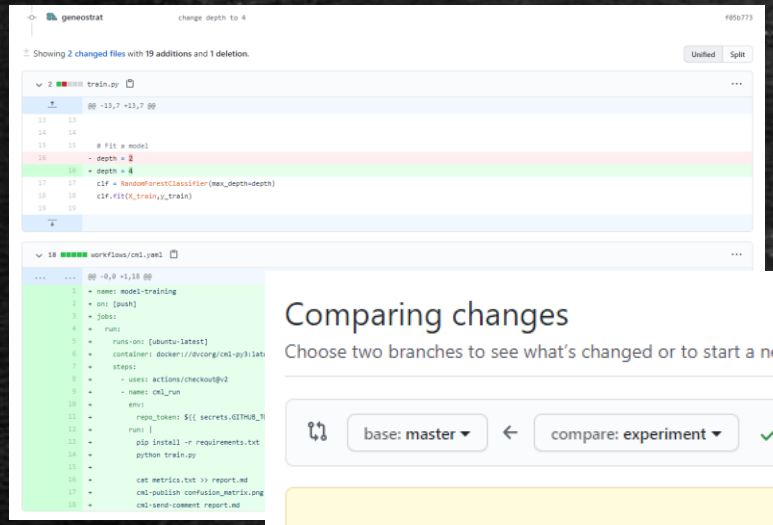
- **Note**: You have to commit the action file to the [master] branch or the action file will not participate in the GitHub workflow system.

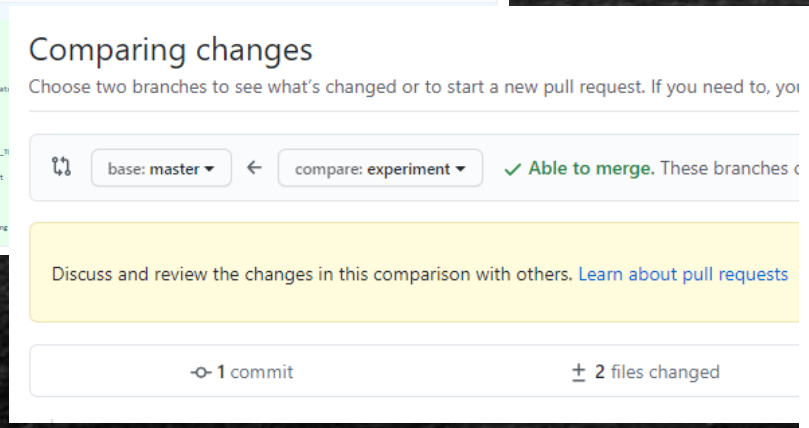


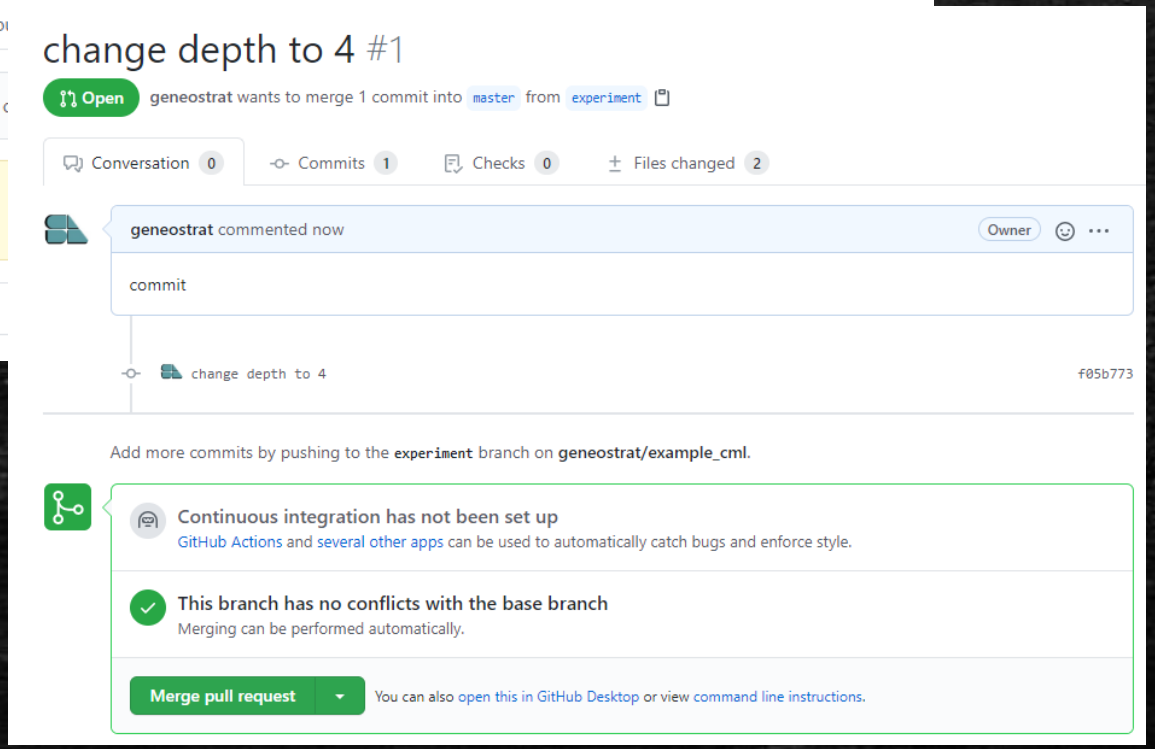
# No Action Taken

- I created the action file in the same branch that I edited the training file. I didn't expect the action to be invoked on this initial check in.
- A subsequent check in did not invoke the workflow either.

# Not Committed to Master Branch



# Successul Commit and Action

Conversation 0 · Commits 1 · Checks 1 · Files changed 1

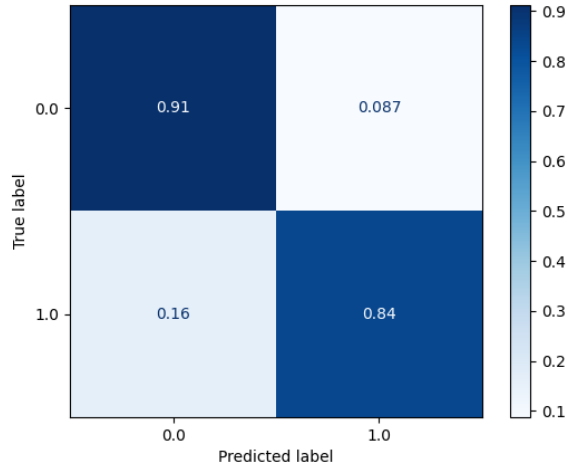
**geneostrat** commented 3 days ago — Owner

*No description provided.*

change train again — 229f961

**github-actions** bot commented on 229f961 3 days ago

Accuracy: 0.872




**geneostrat** merged commit 2d8adcf into master 3 days ago

1 check passed

View details · Revert

# Live Demo!

- Let's make a change and check it in!

# What’s Next?

- Now that we've been introduced to CML, it may be time to take a look at Data Version Control (DVC).