🦿 Deployment

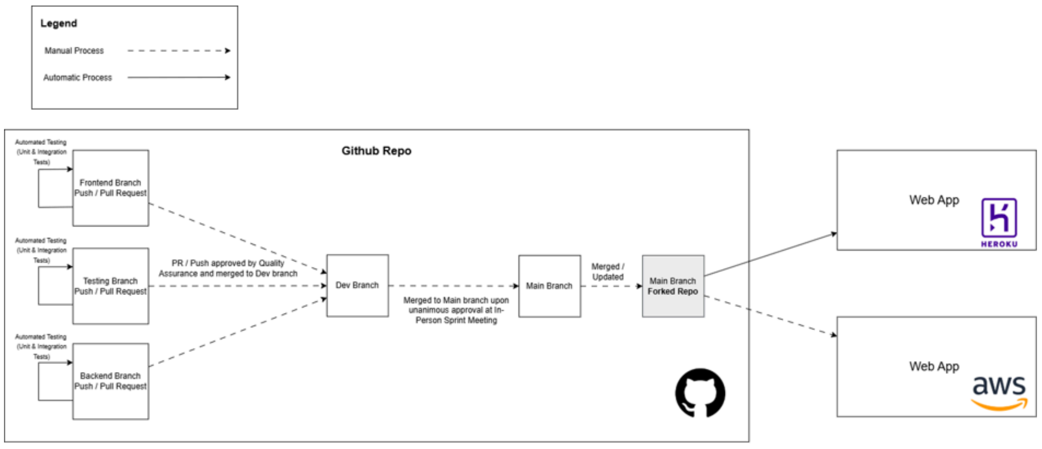
Background Info - Branch Hierarchy (High to Low)

Main → Dev → Frontend | Backend | Testing

Continuous Integration

- To fully grasp our Continuous Deployment pipeline, read through the following workflow with CI
- ⬇️
- Developers from different branches (Frontend, Backend, Testing) will work individually or collaboratively in their respective branches on their tasks/features/updates
- When they are done with their updates, the developers will push their updates into their respective work branches
    - In Example: Federico, the testing lead, will be working in testingBranch and/or test-to-dev branch
- The updates will go through a series of test suites, currently consisting of **23** test suites, and a total of **54** tests
    - Unit Tests (Components), Integration Tests (Templates/Pages)
    - The tests will run on *every branch* on *pushes* and *pull requests*
- The developers will then contact the Quality Assurance developer to check the code, and from here there are two scenarios:
    - ① All tests have passed ✅
        - the Quality Assurance developer will give another quick review of the update and then push it to the **Dev** branch
    - ② Some tests have failed ❌
        - the Quality Assurance developer will go through the error logs in Github Actions (where the automated tests are) and identify the problems
            - there is the possibility that there are changes in the base code that is not reflected in the testing code that is causing the ❌ and not any bugs

        - once confirmed, the Quality Assurance developer will either contact the developer responsible of that update if unfixable directly || the Quality Assurance developer will do a quick fix confirmed by the responsible developer || the Quality Assurance developer will note down the fixes needed in the *testing* cod
        - once resorted, the Quality Assurance developer will push the updates into the **Dev** branch
- Unless crowded with waves of updates from different developers from different branches, the Quality Assurance developer will update all other branches with the latest Dev branch updates and ensure that everything is in sync
- If there are any merge conflicts when updating other branches with the latest changes, the Quality Assurance developer will fix it
    - will contact the lead of the respective branch where the conflicts occur if further assistance is needed
- ***Main*** branch is updated on a fortnightly basis at the *end-of-sprint* meetings upon unanimous vote from all the team members
    - or on rarer occasions, when a prototype is ready to go AND the update is needed, the Quality Assurance developer has the permission to update the Main branch when appropriate
    - after being approved by the Scrum Master


Legend
Manual Process
Automatic Process
Github Repo
Automated Testing (Unit & Integration Tests)
Frontend Branch Push / Pull Request
Automated Testing (Unit & Integration Tests)
Testing Branch Push / Pull Request
Automated Testing (Unit & Integration Tests)
Backend Branch Push / Pull Request
PR / Push approved by Quality Assurance and merged to Dev branch
Dev Branch
Merged to Main branch upon unanimous approval at In-Person Sprint Meeting
Main Branch
Merged / Updated
Main Branch Forked Repo
Web App
HEROKU
Web App
aws

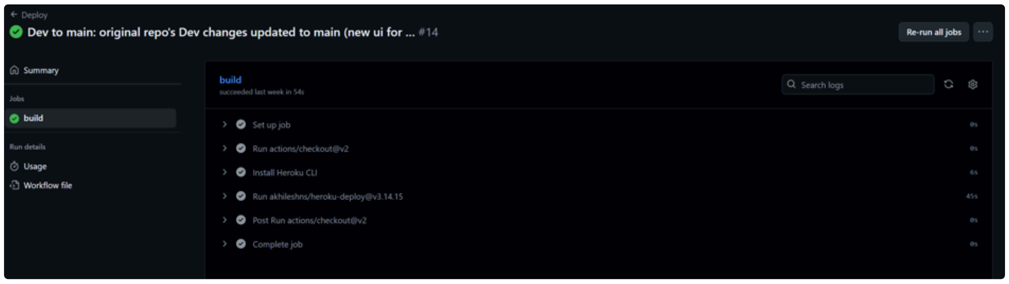
Deployment Pipeline that demonstrates workflow

Note: Dev branch also undergoes automated testing

Continuous Deployment

on *Heroku*

- The ***Main*** branch updates happen in a two step process:
  - First, the latest prototype / version the team agrees on is merged/pushed/PR into the ***Main*** branch
    - (*Manual Process*, as expected)
  - Then, the Quality Assurance developer updates a forked repository (always up to date with original repository's Main branch)
    - (a small, effortless, extra *Manual Process*)
      - that keeps everything safe and clean
- Once both ***Main*** branch have been updated, the automated deployment workflow will run for that it is triggered by any push or pull request from the ***Main*** branch
- Once the green tick in Github Actions pops out ✓ , the build would have been a success, and the configured (connected) Heroku app will be automated instantly



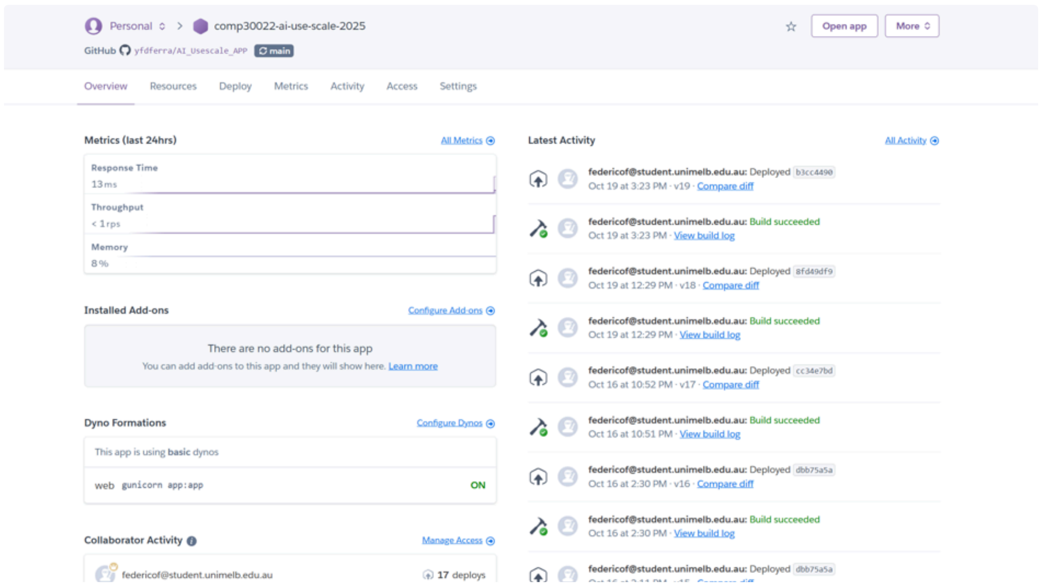
Dev to main: original repo's Dev changes updated to main (new ui for ... #14

build
build
succeeded last week in 54s

```
Set up job                                  0s
Run actions/checkout@v2                     0s
Install Heroku CLI                          6s
Run akhileshns/heroku-deploy@v3.14.15       45s
Post Run actions/checkout@v2                0s
Complete job                                0s
```

the build in github actions

- As seen in the image above, the workflow installs Heroku CLI and runs the deployment in Github Actions altogether



GitHub yfdferra/AI_Usescale_APP main

Metrics (last 24hrs)

Response Time 13 ms
Throughput < 1 rps
Memory 8 %

Installed Add-ons

There are no add-ons for this app
You can add add-ons to this app and they will show here. Learn more

Dyno Formations

This app is using **basic** dynos

web `gunicorn app:app` ON

Collaborator Activity

federicof@student.unimelb.edu.au 17 deploys

Latest Activity

federicof@student.unimelb.edu.au: Deployed b3cc4490
Oct 19 at 3:23 PM · v19 · Compare diff

federicof@student.unimelb.edu.au: Build succeeded
Oct 19 at 3:23 PM · View build log

federicof@student.unimelb.edu.au: Deployed 8fd49df9
Oct 19 at 12:29 PM · v18 · Compare diff

federicof@student.unimelb.edu.au: Build succeeded
Oct 19 at 12:29 PM · View build log

federicof@student.unimelb.edu.au: Deployed cc34e7bd
Oct 16 at 10:52 PM · v17 · Compare diff

federicof@student.unimelb.edu.au: Build succeeded
Oct 16 at 10:51 PM · View build log

federicof@student.unimelb.edu.au: Deployed dbb75a5a
Oct 16 at 2:30 PM · v16 · Compare diff

federicof@student.unimelb.edu.au: Build succeeded
Oct 16 at 2:30 PM · View build log

federicof@student.unimelb.edu.au: Deployed dbb75a5a

Overview page of the connected Heroku App

- It was challenging to configure and deploy on Heroku
- For that our project is a hybrid project with JavaScript in the Frontend and Python in the Backend
- That causes Heroku to be unable to automatically build and deploy
- It took many attempts to successfully configure the builds and workflow for Heroku to finally be able to deploy using Python (with a Frontend build stored into it)
  - First, use Node.js buildpack to build react Frontend

    -
      ```
      cd client
      npm run build
      ```
  - Then, use Python buildpack to run the Flask backend

    -
      ```
      gunicorn app:app
      ```
- When it finally did deploy successfully, the deployed version originally only had the Backend, with no UI (only a static view that was originally hardcoded for project start up)
  - the routing had to be adjusted again
  - the constant that stored the base url was updated from:
    - our localhost: http://localhost:5000
    - to
    - a nested check to see if the process.env.NODE_ENV environment variable is changed to "production"
- See ⬇️ here for the deployed app (version not entirely up to date yet, will be updated again at end of sprint)

AI Use-scales
comp30022-ai-use-scale-2025-3b2430121090.herokuapp.com

on AWS

http://catspin.fun:5173/

- An alternative to the Heroku app is the AmazonWebServices hosted web app
- The group kept both to allow a freedom of choice, and for each method to serve as a backup for one another
- The AWS method is simpler to set up and to use because it can be done directly via its CLI
  - simple deployment
- But it is not automated, all manual process