Python AI assistant project

SE489

Prof. Mamdouh Alenezi

Prepared by	
Mohammed Ibnouf	219110250
Almuhannad Alotaibi	219211206
Ali Basodan	218110378

Version History			
Date	Version	Description	Supervisor
3/18/23	1	-	-
1/5/23	2	-	-

Table of contents:

Summary

Milestones

Phase 1

In this phase the brainstorming of the whole system where implemented and the flow of the system is being initiated and enhancement has been decided along with the pipeline

Phase 2

In this phase the pipeline we created as well as proper testing and the code has been tested to be automated pushed to production

Phase 3

This phase where the projects come to its conclusion, another cycle has been generated to form away of assurance that everything is working fine

Notes

No notes have been added this far!

Phase 1

Introduction of Python Ai assistant project

Why this project

AI assistants have revolutionized the way we interact with technology. These intelligent digital helpers use natural language processing, machine learning, and other sophisticated algorithms to understand and respond to our requests. They can help us with a range of tasks, from scheduling appointments and managing our to-do lists to answering our questions and even writing content. With their ability to learn and adapt to our preferences, AI assistants are becoming an increasingly valuable tool for enhancing our productivity, efficiency, and overall quality of life in the modern era

About

Jarvis is a voice commanding assistant service in Python 3.8 It can recognize human speech, talk to users and execute basic commands.

The Requirements to run the project :
Operation system: Ubuntu 20.04 (Focal Fossa)
Python Version: 3.8.x

>>*Along with other dependencies are provided in the file of the repository*

The application provides skills that can be edited and added from the source code, the skills or its called the “assistant skills” can help now the limitations of “Jarvis” and what can and can’t you ask it. Example of such:

- Opens a web page (e.g 'Jarvis open youtube')
- Play music in Youtube (e.g 'Jarvis play mozart')
- Increase/decrease the speakers master volume (also can set max/mute speakers volume) ** (e.g 'Jarvis volume up!')
- Opens libreoffice suite applications (calc, writer, impress) (e.g 'Jarvis open calc')

- Tells about something, by searching on the internet (e.g 'Jarvis tells me about oranges')
- Tells the weather for a place (e.g 'Jarvis tell_the_skills me the weather in London')

Github link

<https://github.com/OUF5/Python-ai-assistant/tree/DevOps_Test>

Change History

The changing monitoring is done using the Google Docs History
And monitoring the version control using Git

Team Structure:

- Software developer/tester: Mohammed Ibnouf.
- Quality Assurance: Ali Basodan.
- Release Manager: Almuhannad Alotaibi.

Overview

Jarvis is a command assistant service in Python that executes commands given by the user. It uses third-party APIs for web information search, weather forecasting, etc....

Assistant Features

- Asynchronous command execution.
- Answers general questions (via call Wolfram API), e.g ('Jarvis tell me the highest building').
- Configurable assistant name.
- Log preview in the console.
- Keeps commands history and learned skills in MongoDB.'.

Enhancements and Future Plans

In the description of the project in GitHub, it says that it supports two user input modes (texts and speech). But when we went through the code, we found out that the speech functionality is not completed. So as an enhancement to the assistant we will modify the input speech functionality, this might disturb the current development as it requires modification insight in the source code.

Communication and Proposed Pipeline

Communication Medium Used

Our group decided to use ***WhatsApp*** group chats as the main communication medium since it provides the fastest and easiest way for instant messaging and for general availability. Additionally, ***Google Meet*** will be the designated online video conference platform used in case group members needed to showcase or present work for feedback and general discussions.

Planned Pipeline

The following are the tools that will be used for the project's pipeline:

Code phase:

Visual Studio Code

Visual Studio Code is lightweight and versatile and will provide major extensibility for our use case. Furthermore, VCS comes integrated with Git as well which will make it easier to use our tools of version control in the same environment.

Version Control:

Git and Github

Our group will be using Git and Github for version control and collaboration. Since the project requires group work, Git will help us to track changes that each member has made, and review work done and eventually commit to our central repository. Github will be used to store our work online and facilitate collaboration by sharing the same codebase.

Testing phase:

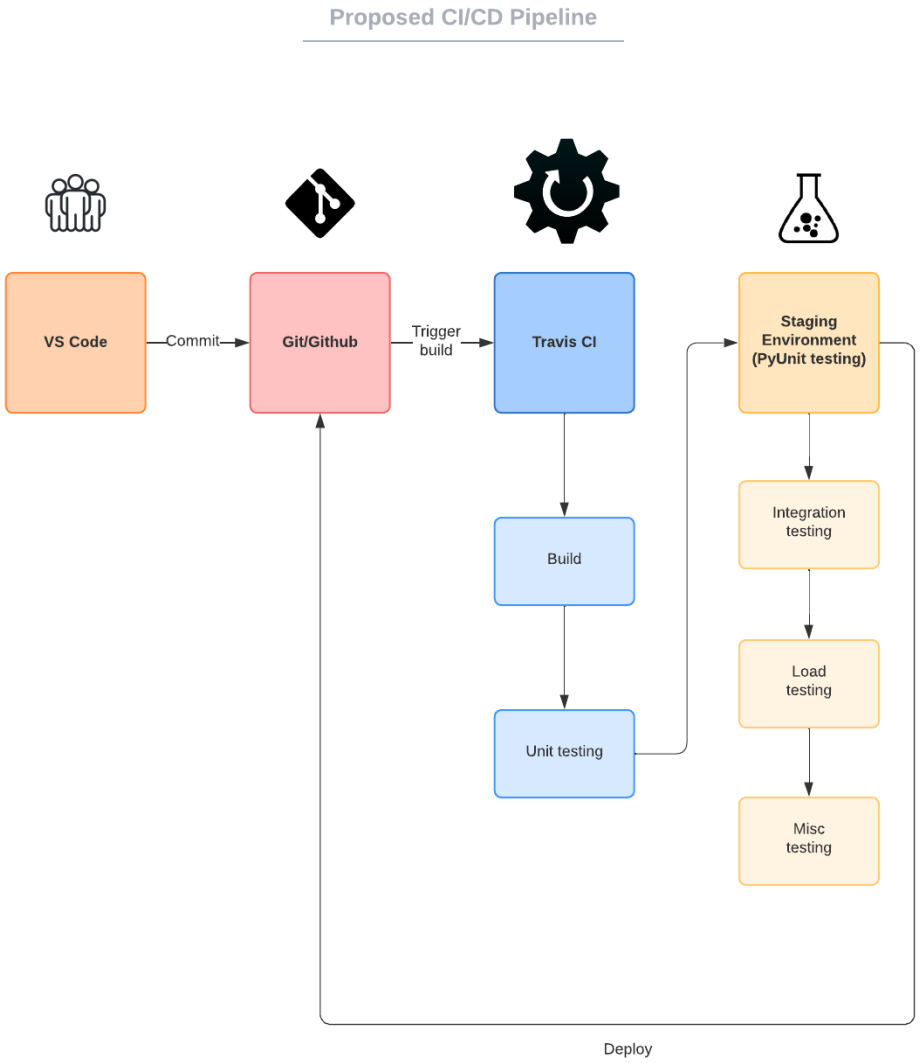
PyUnit

The PyUnit framework will be used in our project for testing since it provides extensive utilities that will help us detect and minimize code bugs and errors.

Continuous Integration and Continuous Deployment (CI/CD):

Travis

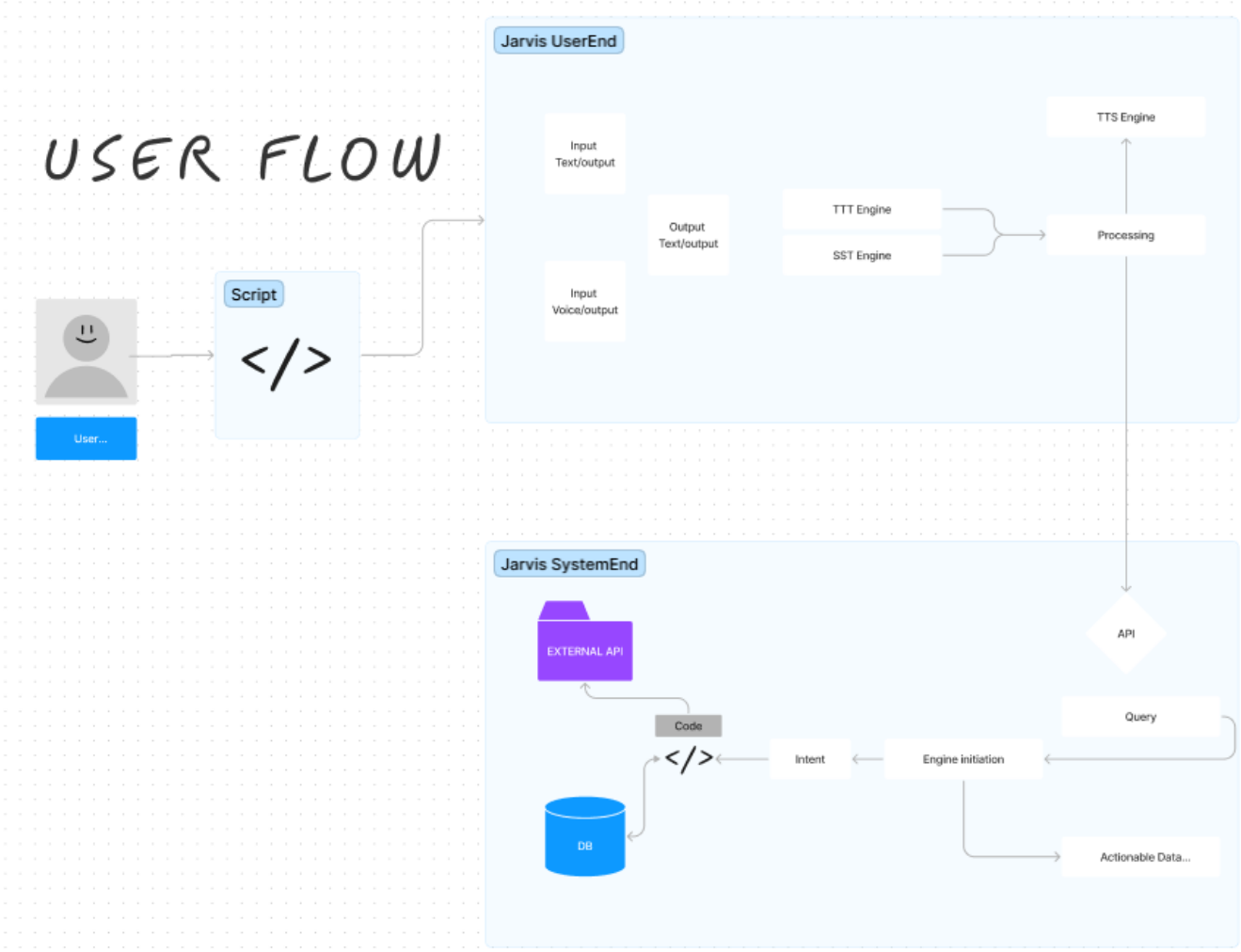
Travis is already being used by the original project, so we decided to keep the same tool and not force other similar tools such as Jenkins. Travis provides similar capabilities to Jenkins and can be integrated with other tools for flexibility. We will be using Travis for our CI/CD needs as it allows us to customize our workflow for automated testing, building, and delivery. Additionally, Travis CI offers continuous deployment into GitHub for our case.



CI/CD Pipeline diagram

Architecture and Design

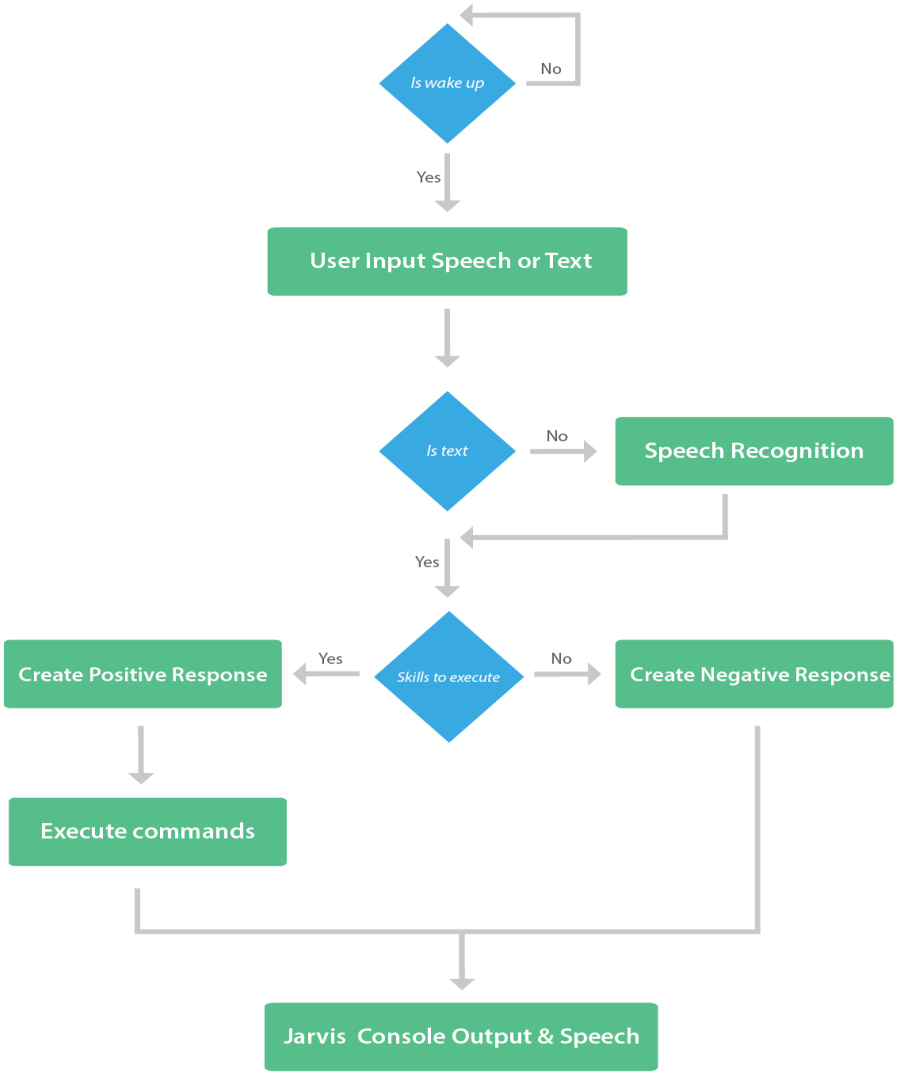
User Flow



As it shows in this user flow diagram the user will input a command <Text,voice> the output will generated using engines <TEXT TO TEXT, SPEECH TO TEXT, TEXT TO

SPEECH> using provided API's like Wolfram API to identify of command in the Database.

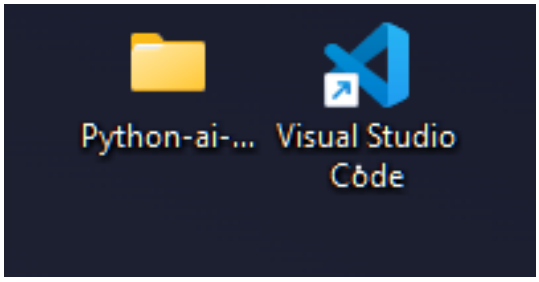
Decision model



This is the decision model which represents the full overview of how the application works and with respect to the commands how it does function.

Program Execution

Screenshots of running version



The project has been cloned from the GitHub link to our GitHub repository and local device to complete the project using VScode

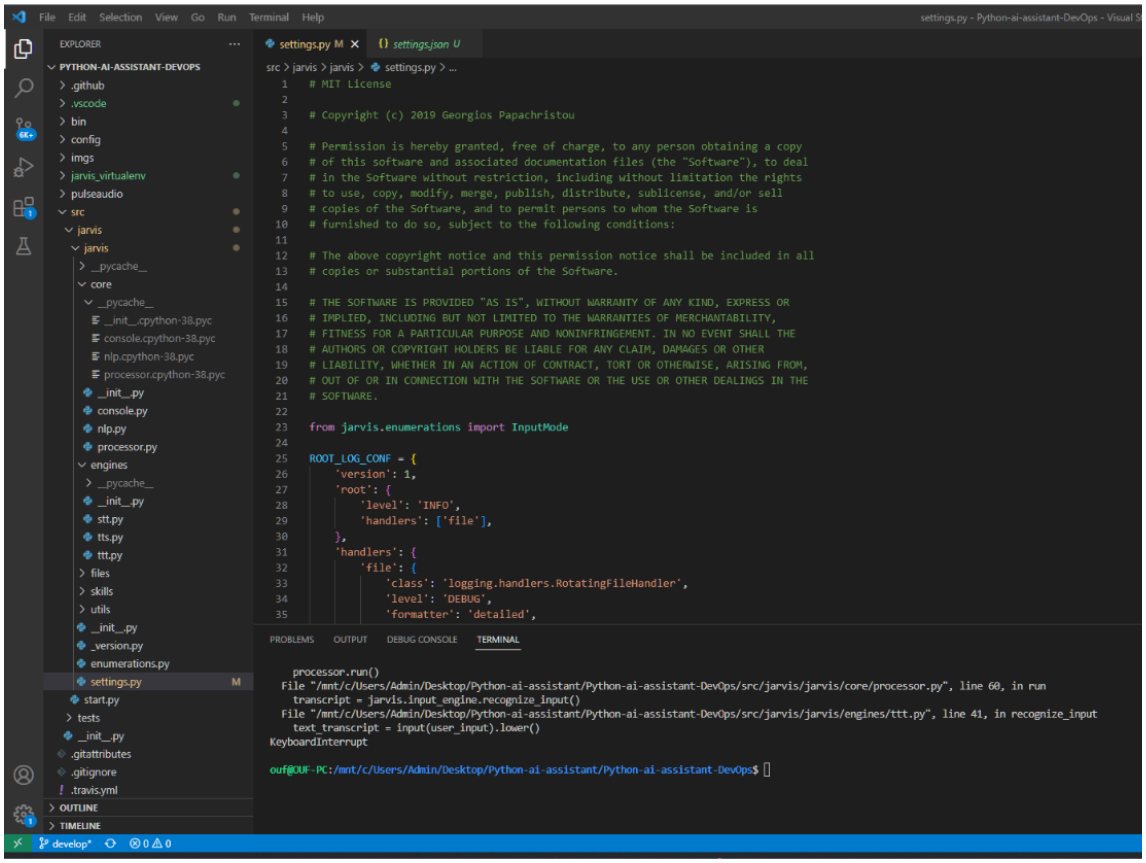
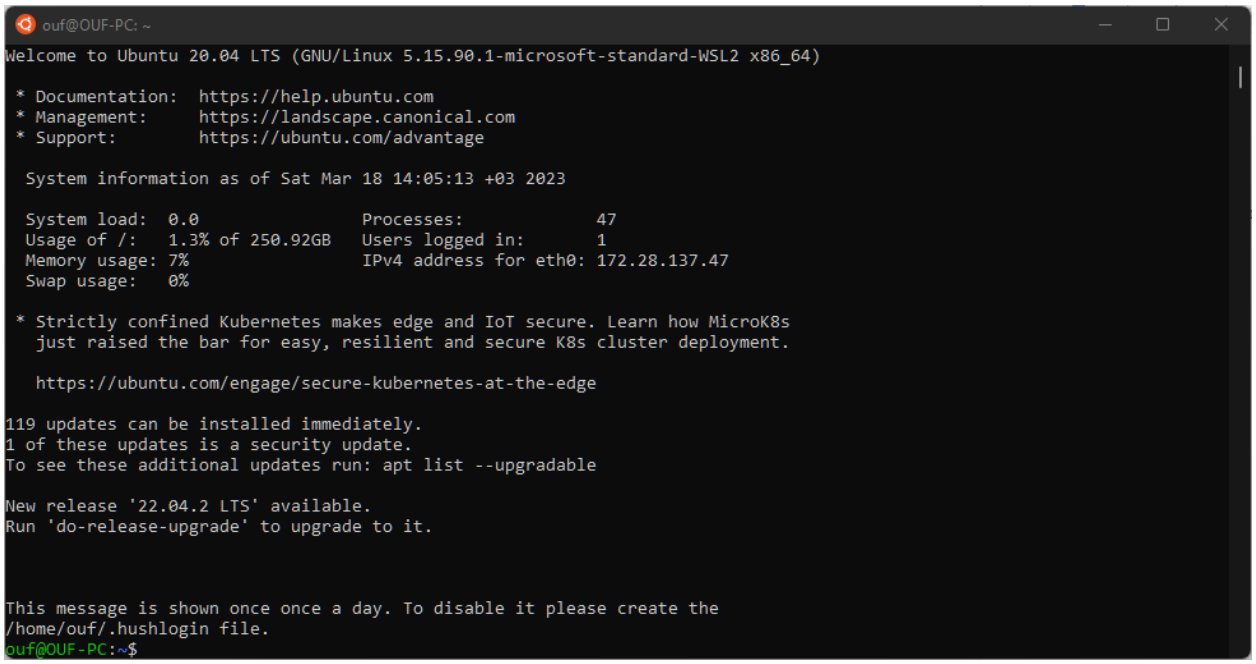
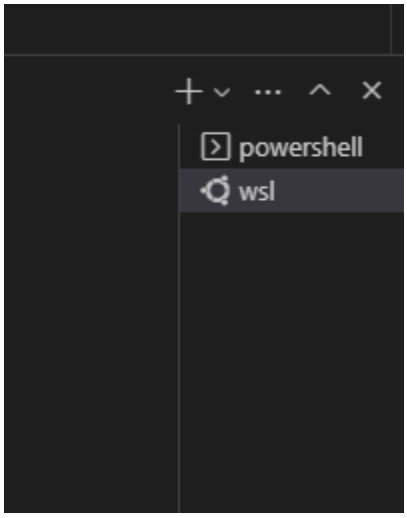


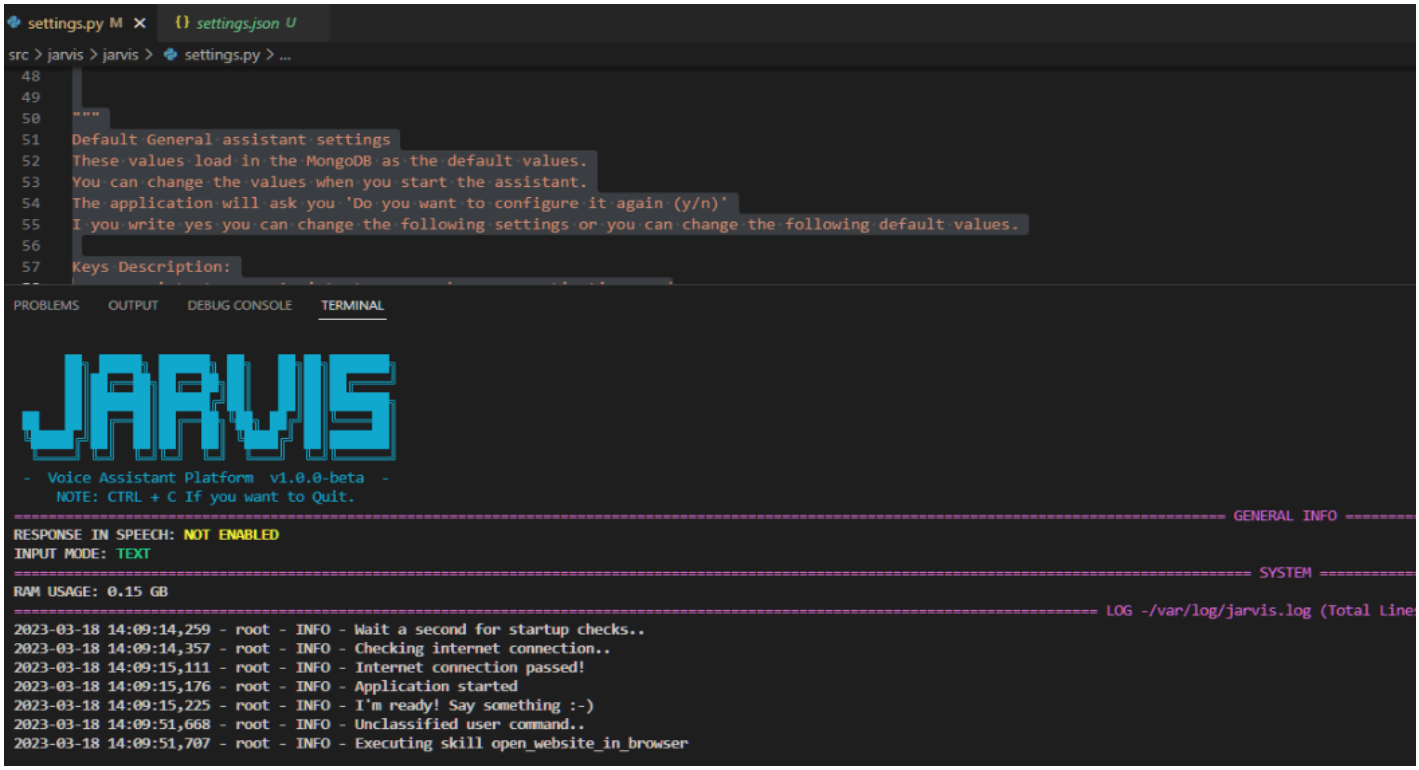
Photo of the running version

Consideration point: the program only runs in Ubuntu so the virtual machine had to be implemented but due to its hardware usage for unnecessary purposes, we decided to use WSL second version to run bash on.

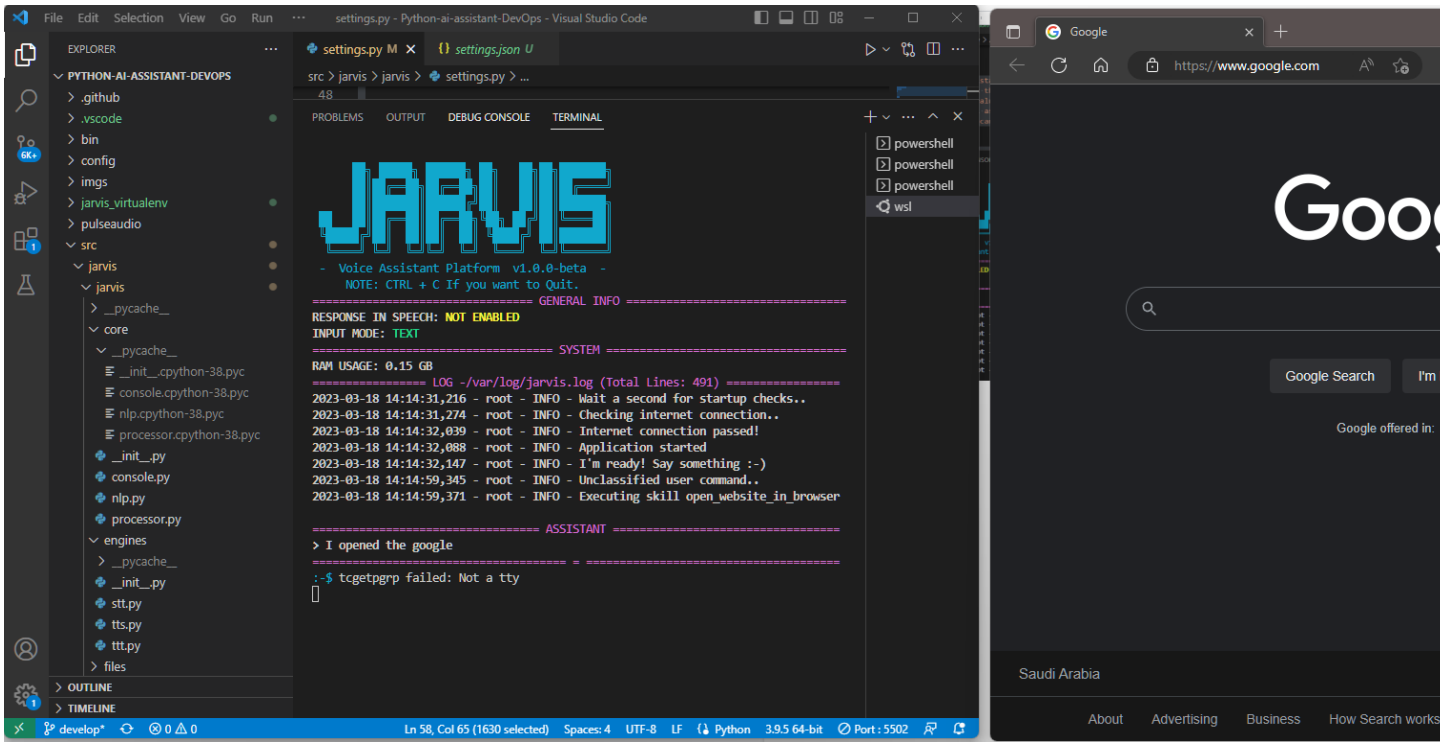




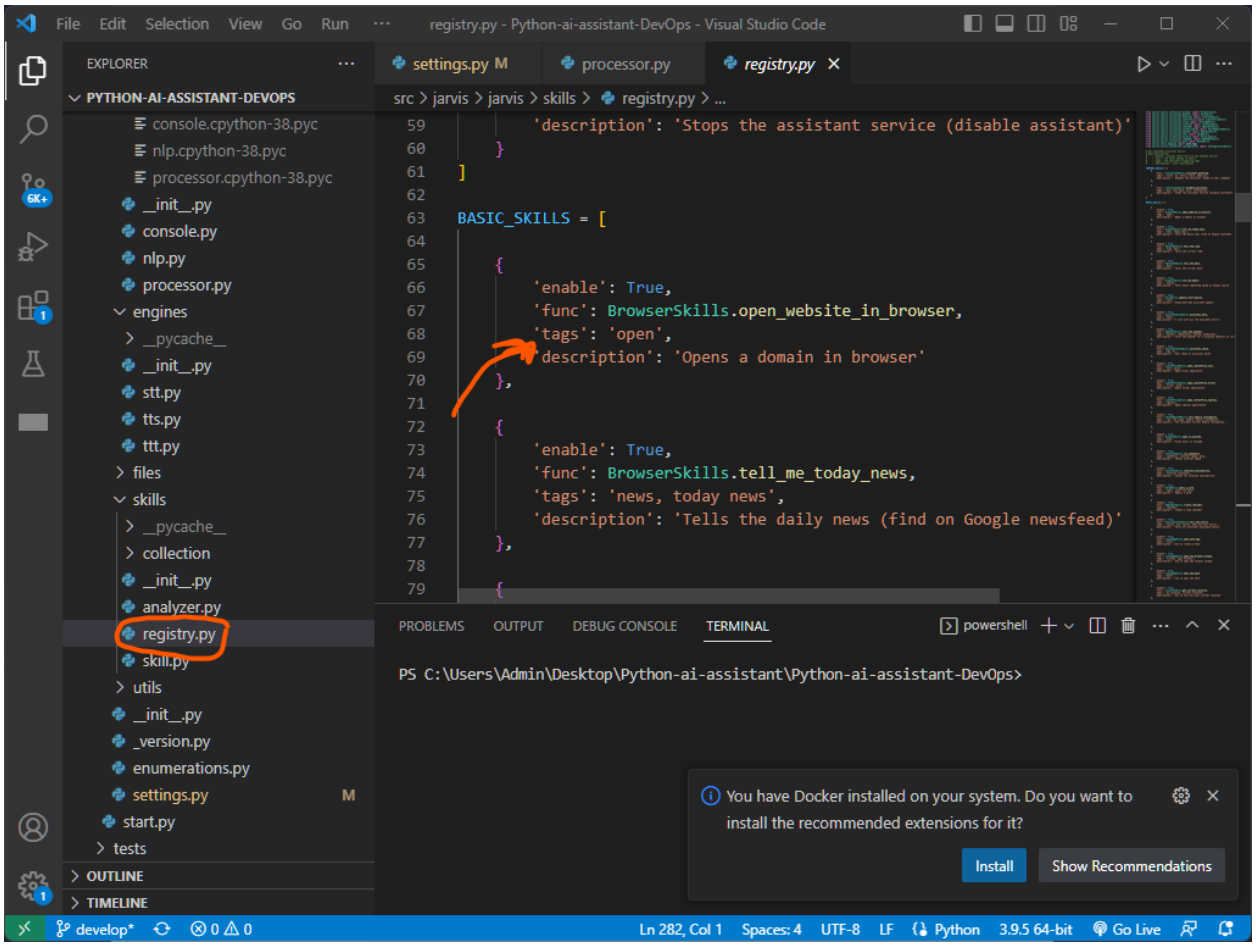
Running Terminal in VsCode <WSL>



Running Version of the software: the program is running and taking input as a text using the `Text To Text Engine (TTT)`

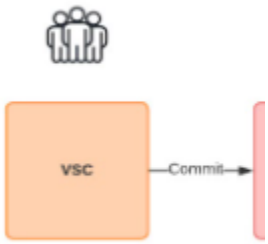


Running Command “Open Google”: as you can see that Jarvis used the <basic> registry skills as the following picture demonstrates, as this project allows the administrator to add a skill fo Jarvis

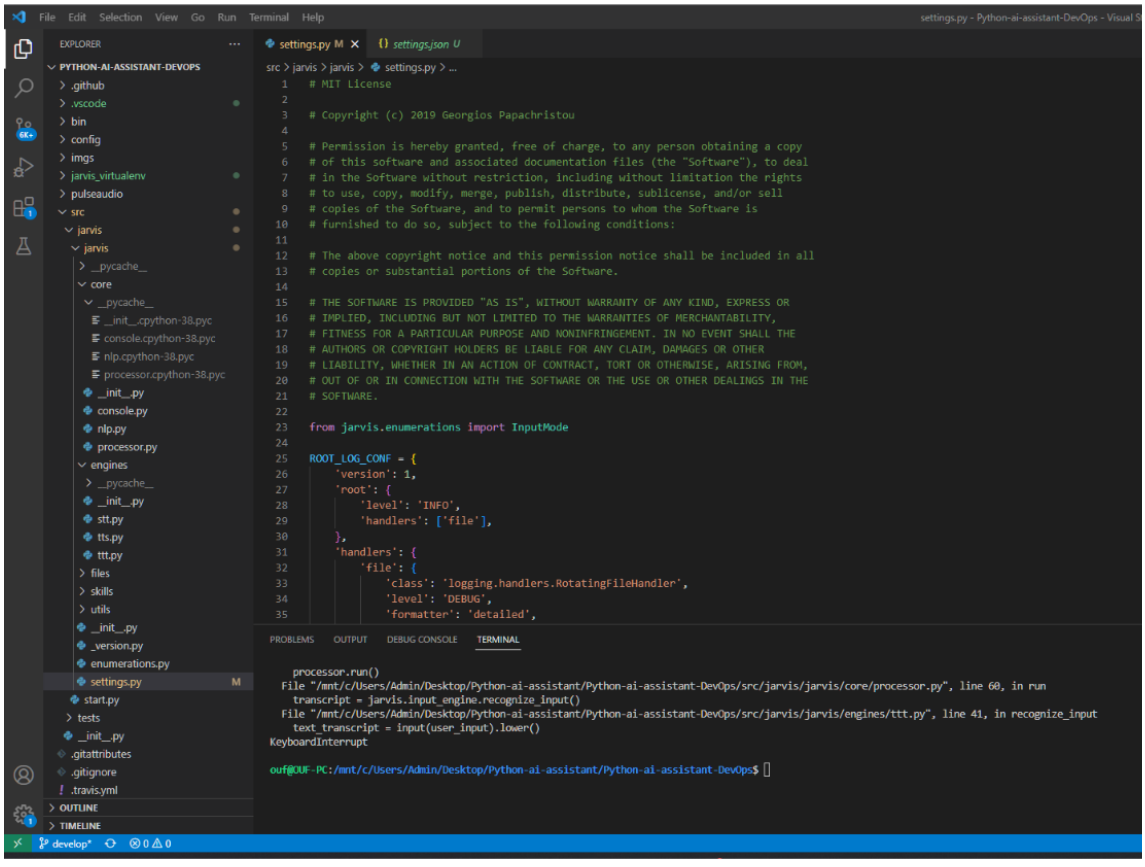


Phase 2

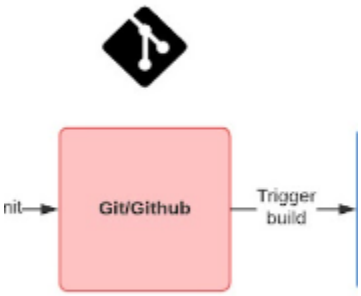
CI/CD Pipeline



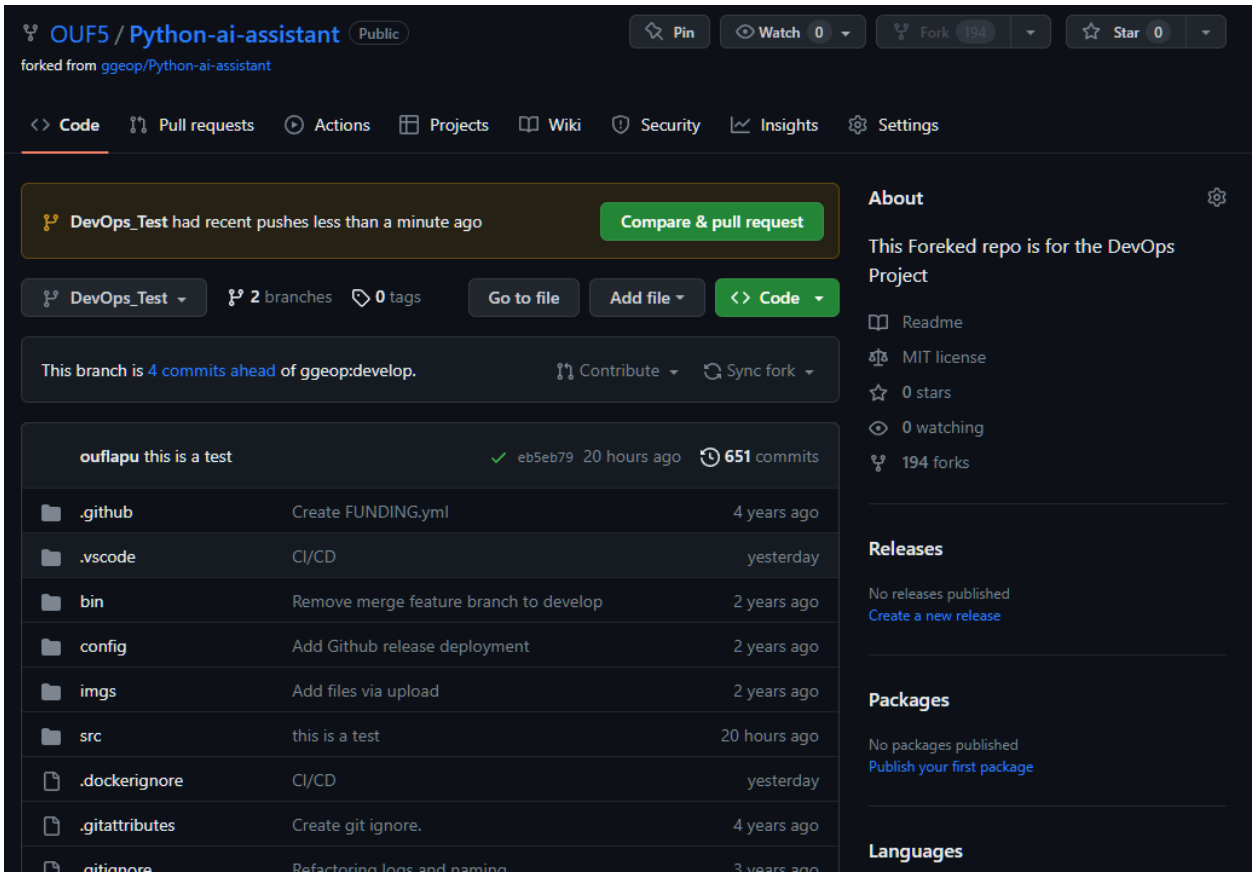
The first stage of our planned pipeline is to use VsCode and Wsl V.2 to **clone** the repository from github as well as modify the code and run several tests. As Vscode helps us push and pull with syncing option with respect to **Github**



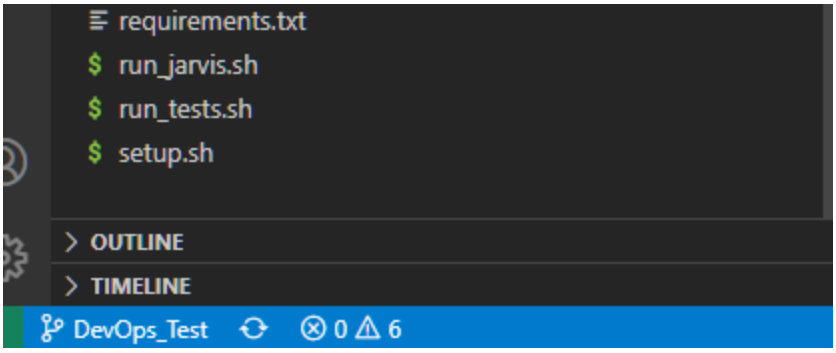
<The project>



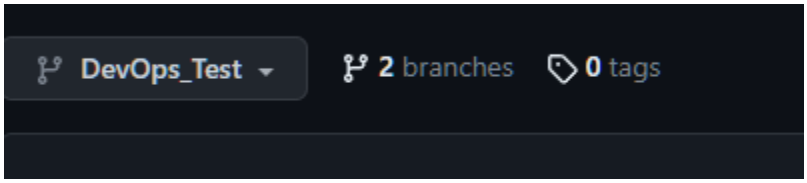
After Cloning using **git**, The second stage is to Fork the repository and add it into our repository as it will allow us to add our own branch as well as help with the second stage of the pipeline which is **CI/CD** and this stage requires the authority of the owned repository.



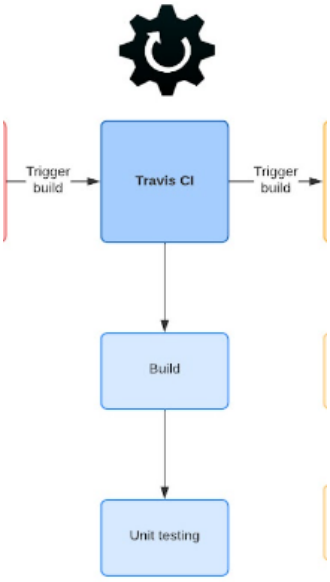
As its shown that the repository has been forked



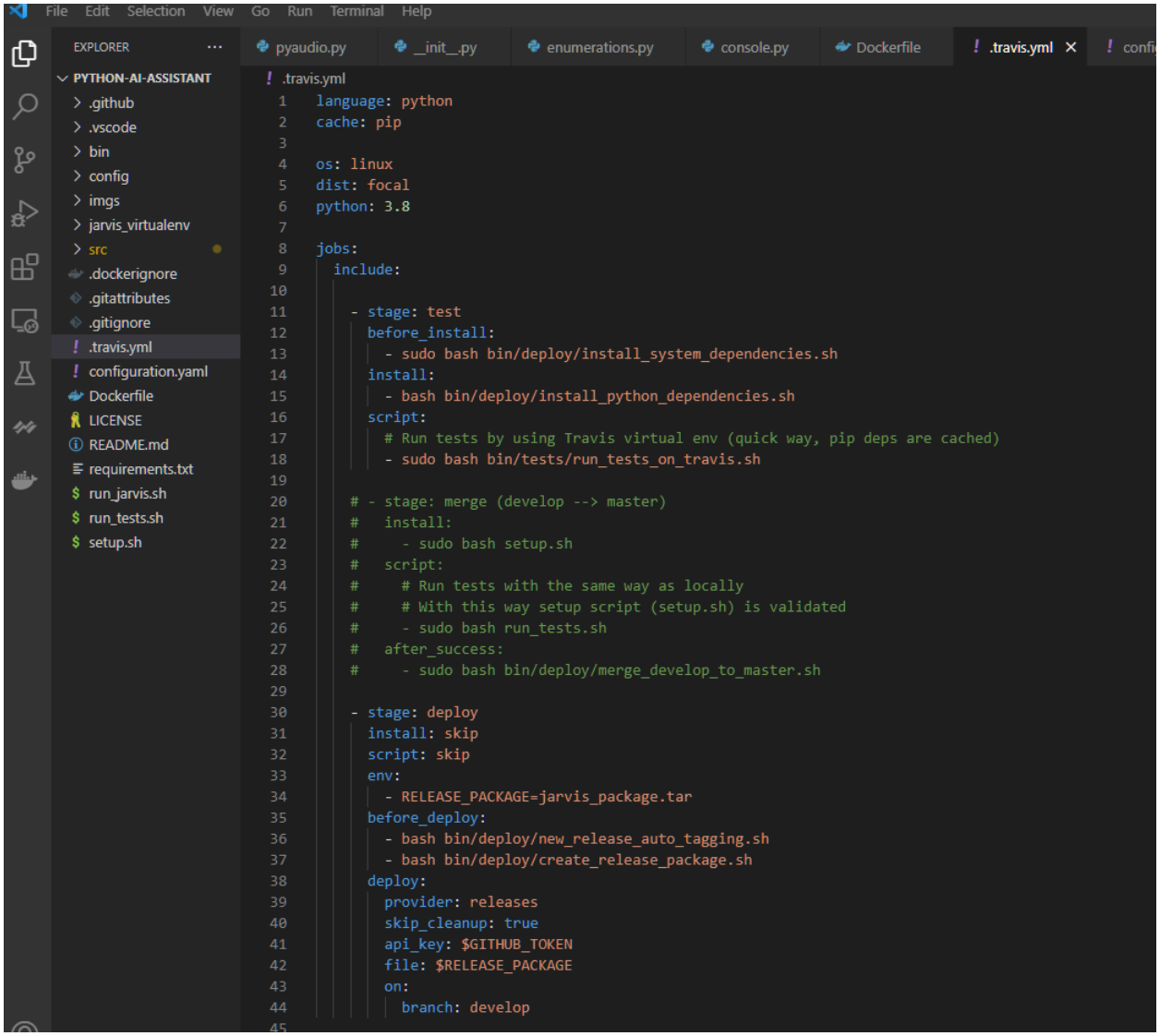
In VsCode the newly generated branch has been created and chosen to be committed in.



in the Github repository

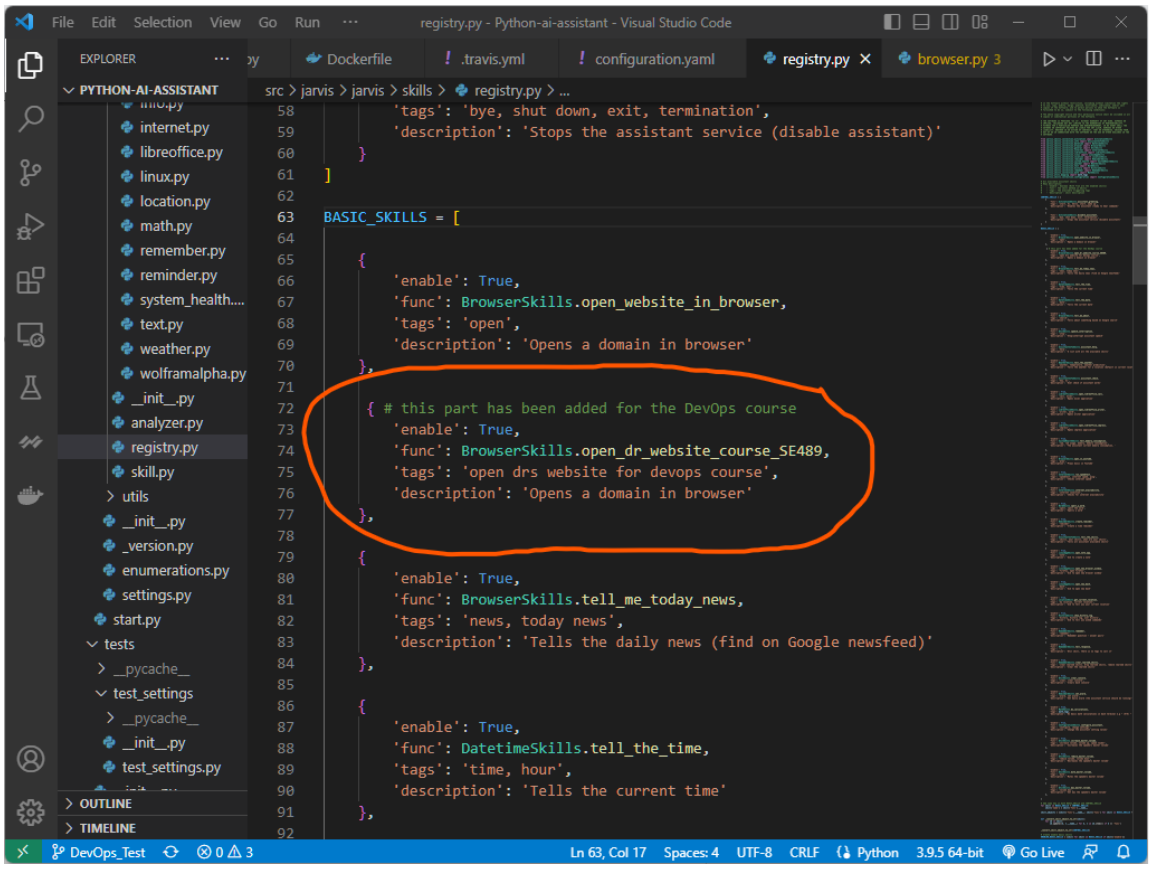


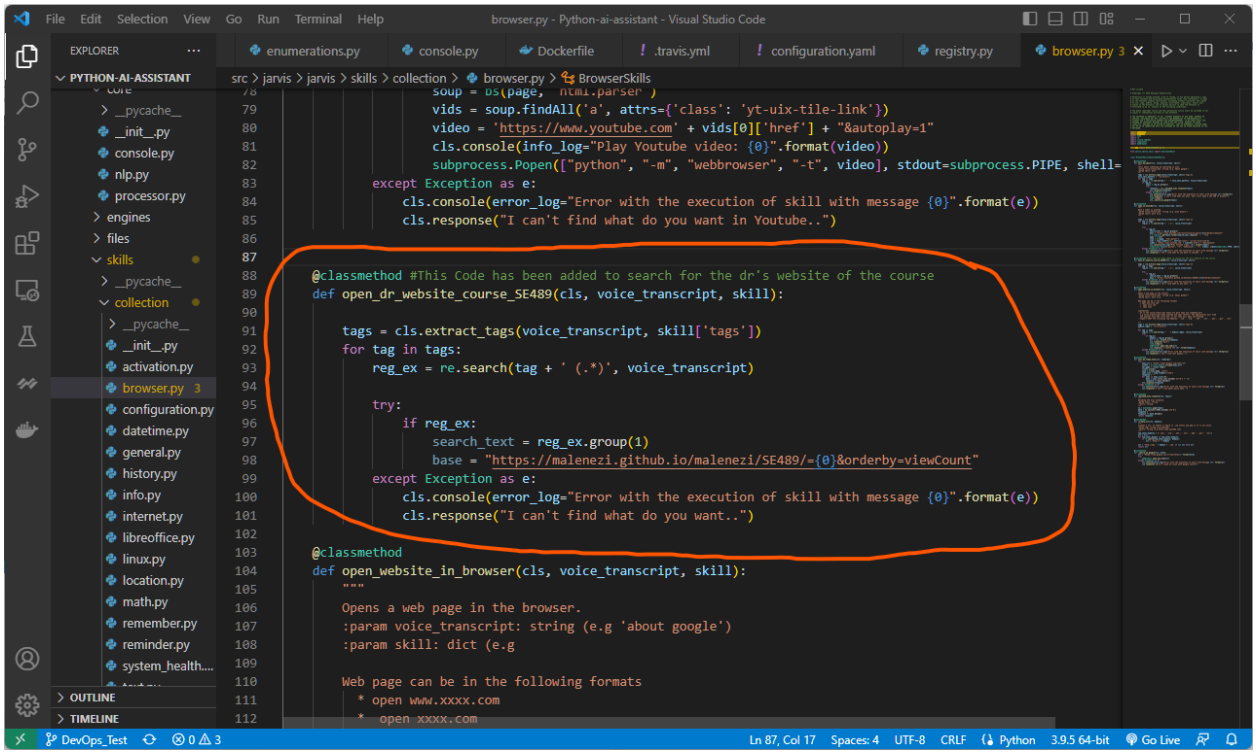
In this stage github will take an important rule into triggering the building phase as we integrated **Travis CI** to handle the triggering part by building and testing the commit<changes> then deploy it in github after deep checking, below the process of how **Travis CI** works will displayed:



The yml file of travis

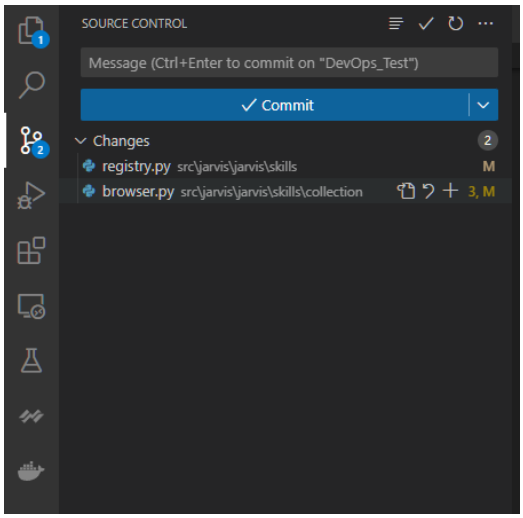
#1 commit any changes in the Code



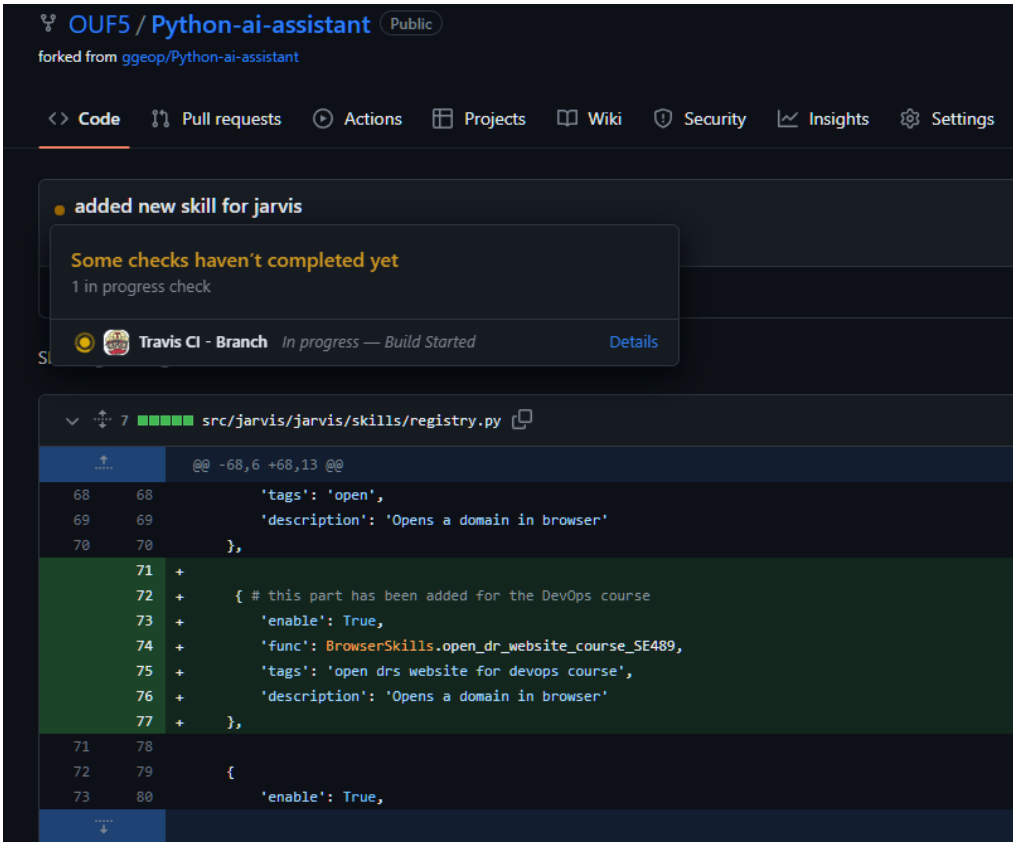


We decided to add new skills or " assistant skills" to jarvis the skill enable the user to command Jarvis to open the Course website <https://malenezi.github.io/malenezi/SE489/> by telling him the following command :

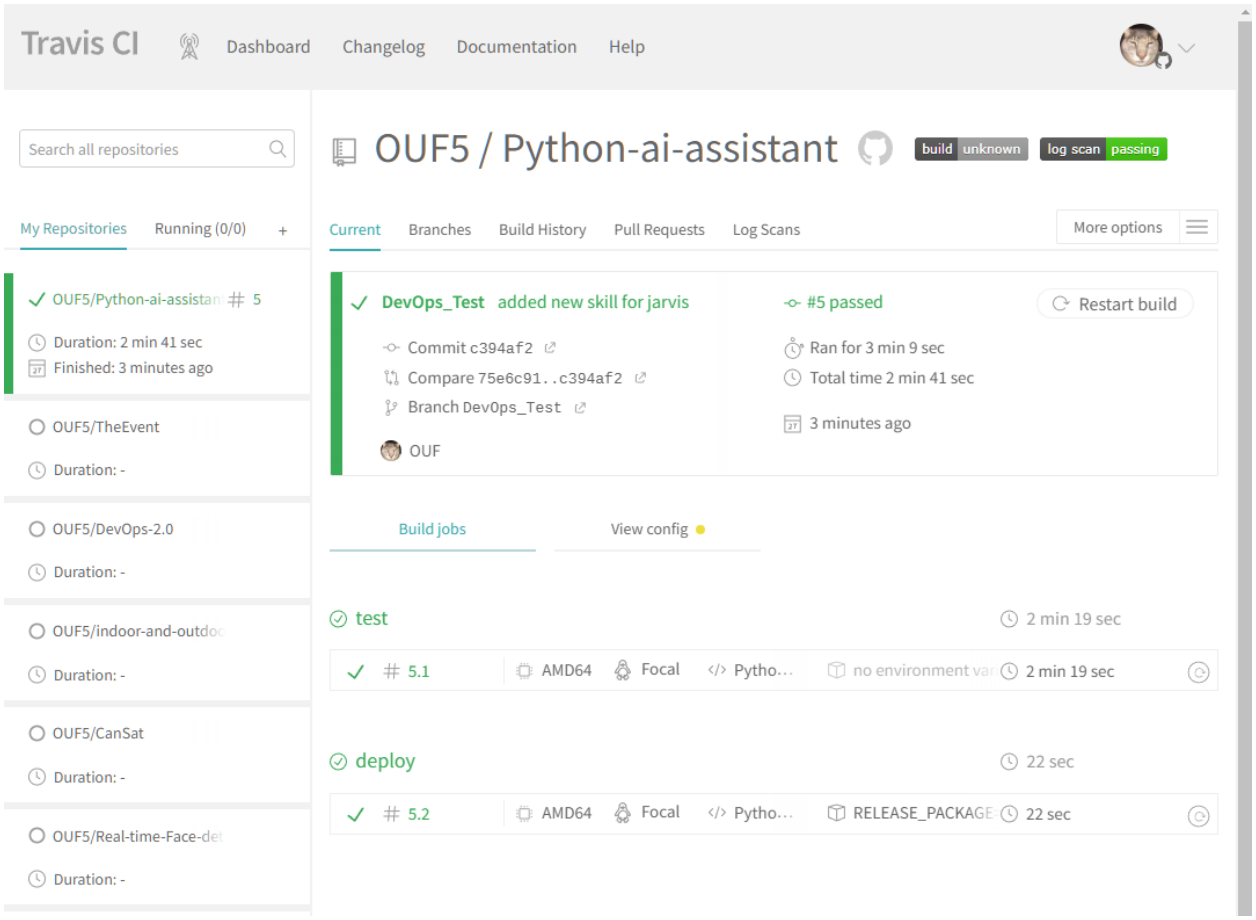
```
open drs website for devops course
```



We committed the changes with a comment "added new skill for jarvis" as to heads of see what is going on in github repository



Looking at github commits we can see that the Travis CI are pending the commit to be built and tested.



This the Travis page to report the Building that happened after our commit and as it shows the build with test and deploy has been done with travis.

CI/CD Testing

Continuous testing is a key component of continuous integration / continuous delivery/deployment (CI/CD) pipelines. It provides timely feedback to help improve code quality, avoid bottlenecks, and expedite DevOps processes.

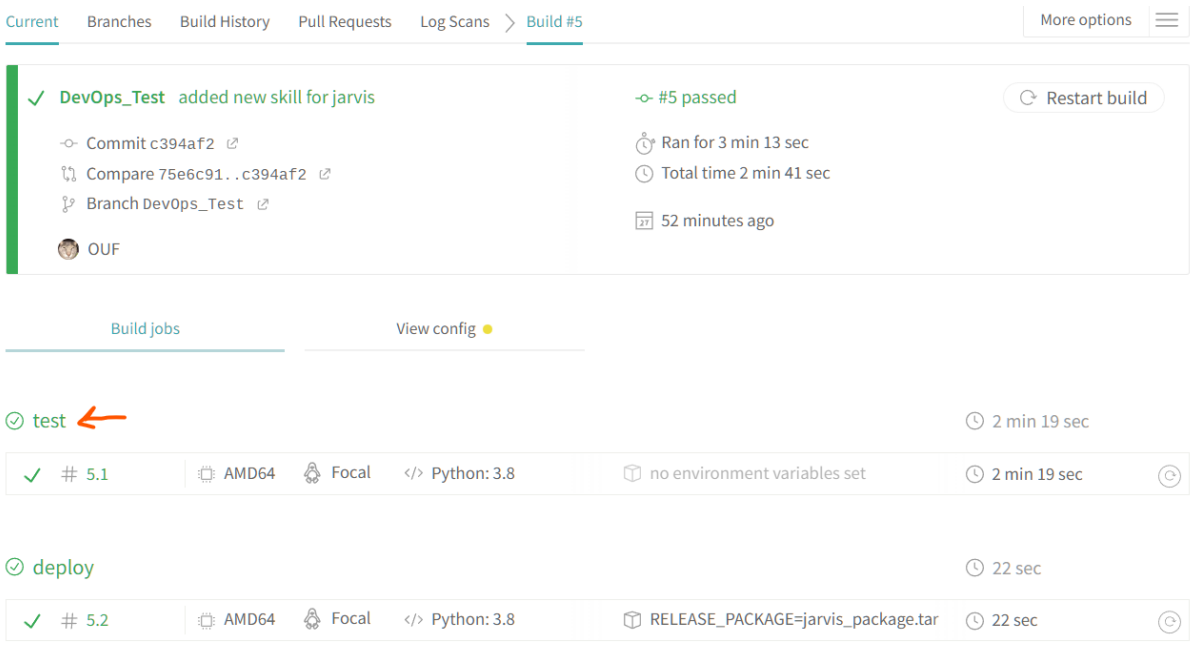
skill_analyzer_tests.py

```python
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity

from jarvis import settings
from jarvis.skills.registry import CONTROL_SKILLS, BASIC_SKILLS, ENABLED_BASIC_SKILLS
from jarvis.enumerations import MongoCollections
from jarvis.skills.analyzer import SkillAnalyzer
from jarvis.utils.mongoDB import db


class TestSkillMatching(unittest.TestCase):

    def setUp(self):

        all_skills = {
            MongoCollections.CONTROL_SKILLS.value: CONTROL_SKILLS,
            MongoCollections.ENABLED_BASIC_SKILLS.value: ENABLED_BASIC_SKILLS,
        }
        for collection, documents in all_skills.items():
            db.update_collection(collection, documents)

        default_assistant_name = settings.DEFAULT_GENERAL_SETTINGS['assistant_name']
        default_input_mode = settings.DEFAULT_GENERAL_SETTINGS['input_mode']
        default_response_in_speech = settings.DEFAULT_GENERAL_SETTINGS['response_in_speech']

        default_settings = {
            'assistant_name': default_assistant_name,
            'input_mode': default_input_mode,
            'response_in_speech': default_response_in_speech,
        }

        db.update_collection(collection=MongoCollections.GENERAL_SETTINGS.value, documents=[default_settings])

        self.skill_analyzer = SkillAnalyzer(
                                            weight_measure=TfidfVectorizer,
                                            similarity_measure=cosine_similarity,
                                            args=settings.SKILL_ANALYZER.get('args'),
                                            sensitivity=settings.SKILL_ANALYZER.get('sensitivity'),
                                            )

```

Unit Test Code #1

```
41    def test_all_skill_matches(self):
42        """
43        In this test we examine the matches or  ALL skill tags with the functions
44        If all skills matched right then the test passes otherwise not.
45
46        At the end we print a report with all the not matched cases.
47
48        """
49
50        verifications_errors = []
51
52        for basic_skill in BASIC_SKILLS:
53            print('-------------------------------------------------------------------------------------------')
54            print('Examine skill: {0}'.format(basic_skill.get('name')))
55            for tag in basic_skill.get('tags',).split(','):
56                # If the skill has matching tags
57                if tag:
58                    expected_skill = basic_skill.get('name')
59                    actual_skill = self.skill_analyzer.extract(tag).get('name')
60                    try:
61                        self.assertEqual(expected_skill, actual_skill)
62                    except AssertionError as e:
63                        verifications_errors.append({'tag': tag, 'error': e})
64
65        print('------------------------------------ SKILLS MATCHING REPORT ------------------------------
  -----')
66        if verifications_errors:
67            for increment, e in enumerate(verifications_errors):
68                print('{0})'.format(increment))
69                print('Not correct match with tag: {0}'.format(e.get('tag')))
70                print('Assertion values (expected != actual): {0}'.format(e.get('error')))
71            raise AssertionError
72        else:
73            print('All skills matched correctly!')e
```

Unit Test Code #2



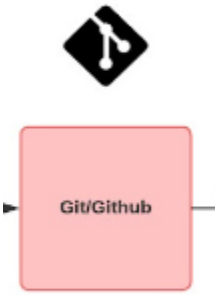
We used Travis CI to do the Continuous Testing

Job log View config

Remove log Raw log

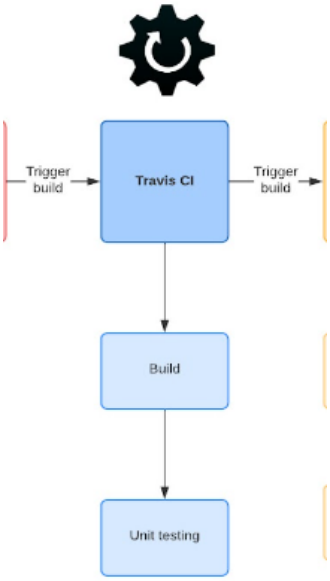
```
0.06s
▶ 1  Worker information                                                    worker_info
6                                                                                       0.01s
▶ 7  Build system information                                              system_info
166
167                                                                                     2.02s
                                                          docker_mtu_and_registry_mirrors
                                                                          resolvconf
▶ 168 $ git clone --depth=50 --branch=DevOps_Test https://github.com/OUF5/Python-ai-   git.checkout  0.75s
178
179 $ source ~/virtualenv/python3.8/bin/activate                                        0.01s
▶ 180 Setting up build cache                                                cache.1
186
▶ 187                                                                       cache.pip
189 $ python --version
190 Python 3.8.13
191 $ pip --version
192 pip 22.1.2 from /home/travis/virtualenv/python3.8.13/lib/python3.8/site-packages/pip (python 3.8)
▶ 193 $ sudo bash bin/deploy/install_system_dependencies.sh                  before_install  81.62s
▶1131 $ bash bin/deploy/install_python_dependencies.sh                      install         21.39s
1288 $ sudo bash bin/tests/run_tests_on_travis.sh                                       5.01s
1289 ========================================= = =======================================
1290 First time configuration..
1291 ========================================= = =======================================
1292 ================================ Assistant Name ================================
1293 Assistant name- jarvis configured successfully!
1294 Input mode - text configured successfully!
1295 Speech response output- False configured successfully!
1296 ------------------------------------------------------------------------------------
1297 Examine skill: open_website_in_browser
1298 ------------------------------------------------------------------------------------
1299 Examine skill: open_dr_website_course_SE489
1300 ------------------------------------------------------------------------------------
1301 Examine skill: tell_me_today_news
1302 ------------------------------------------------------------------------------------
```

Travis CI provides Job log that shows the status of the every skill and all of them passed including the new skill that we add

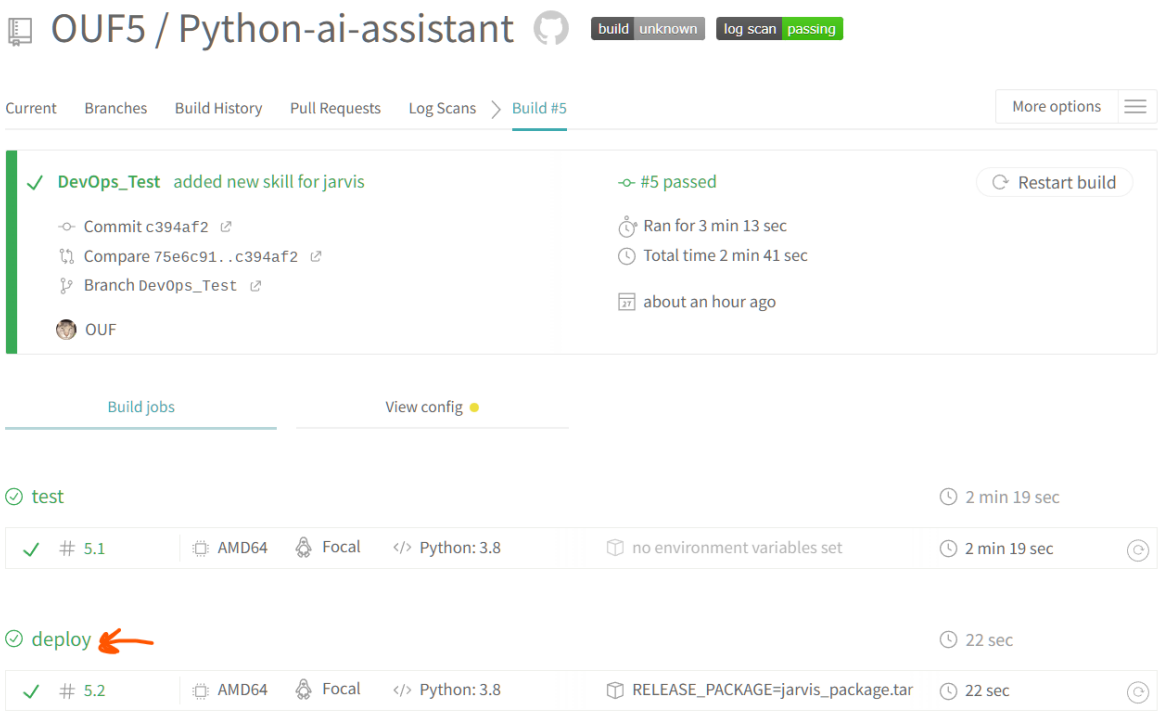


In this stage the deployment is continues as it uses Travis CI for this job and GitHub enable the option of deployment from Travis, and Travis CI process for the deployment is the following :

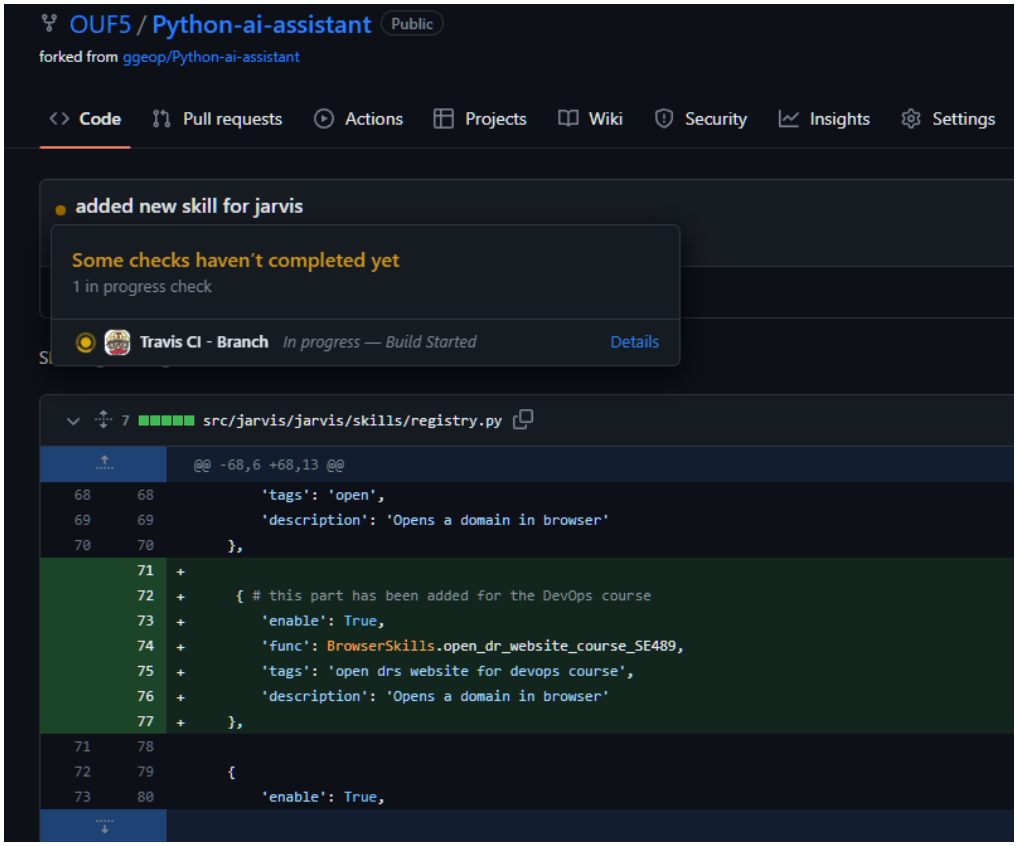
1- Trigger Build
2- Test out the build using pyUnit test
3- pending the commit until the success of the build and test
4- deploy the commit and the changes into GitHub



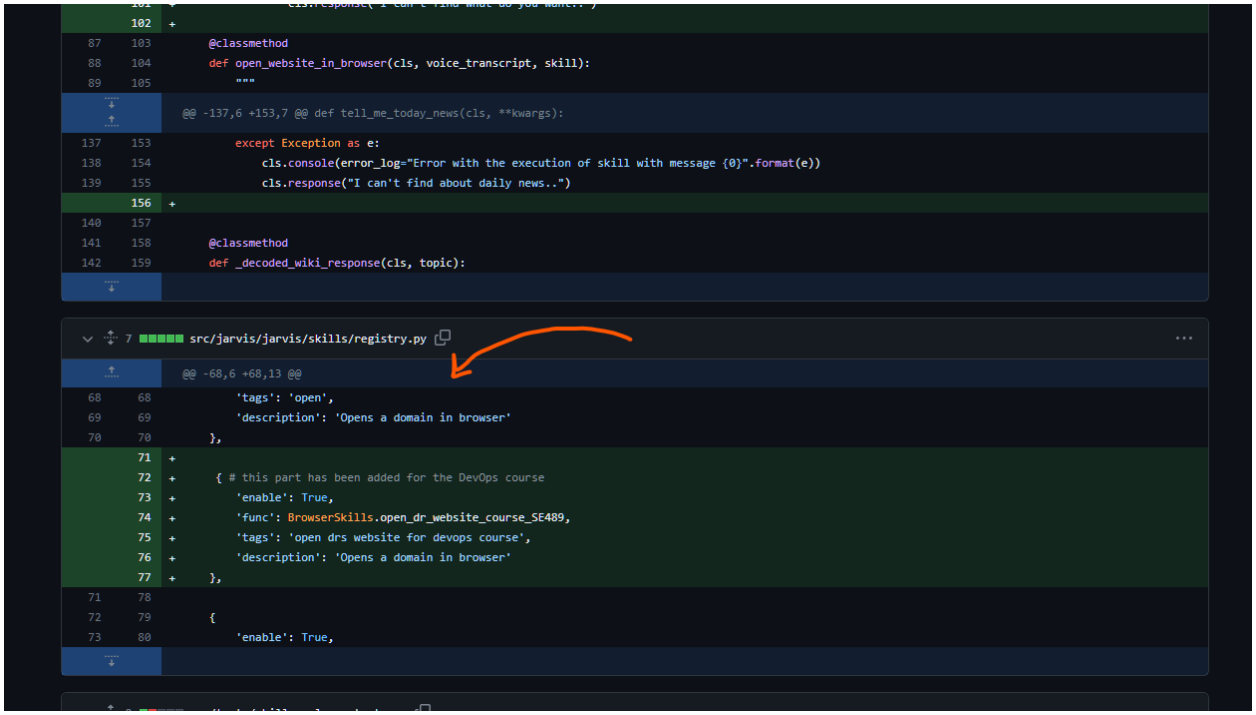
Here is some detailed pictures of how the deployment works in Travis and GitHub:



The deploy in the build jobs



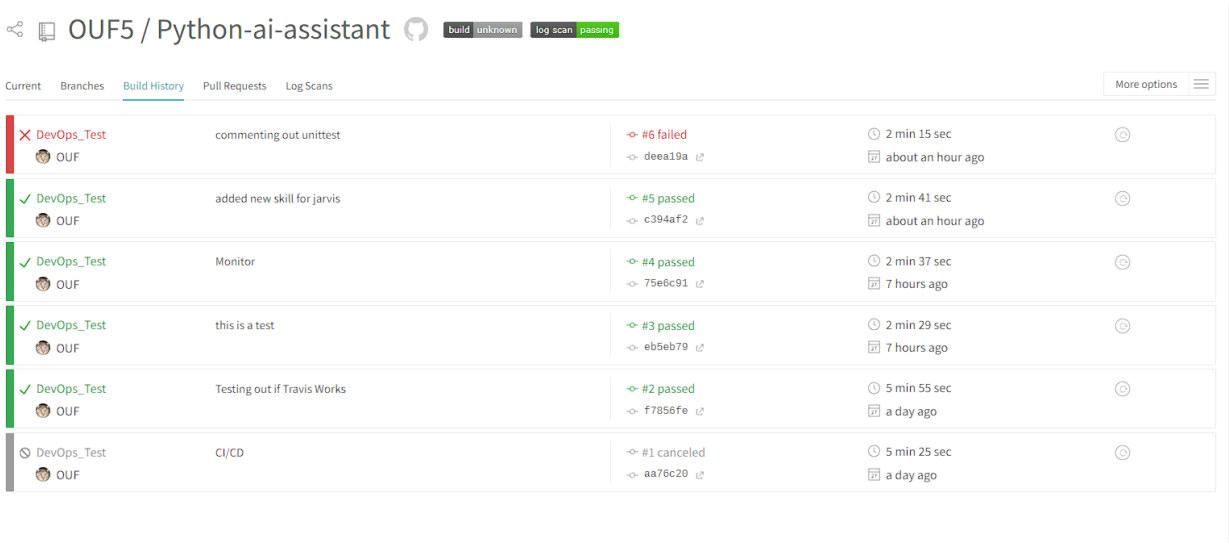
As this indicates that commit is waiting for travis to deploy it



As here shows that the commit has been deployed after the build and test is done

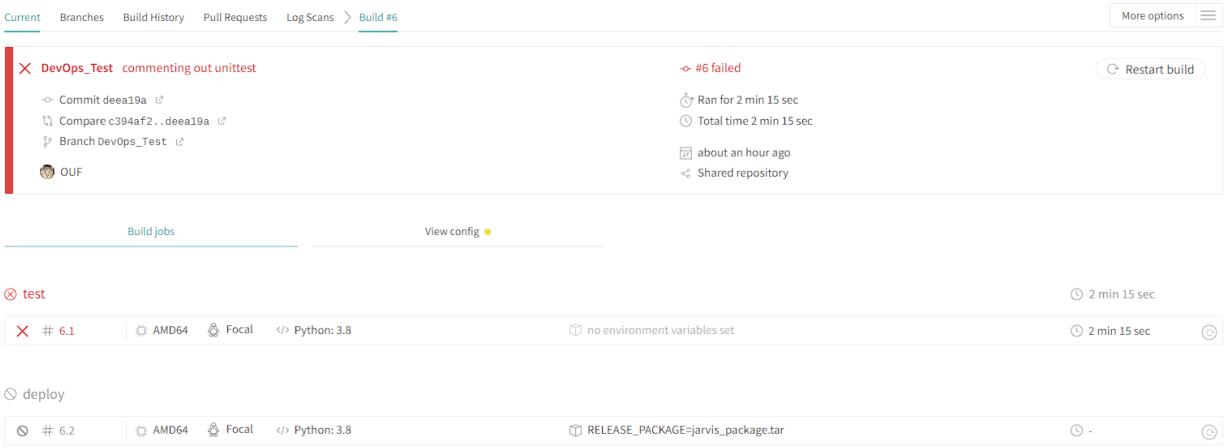
CI/CD Monitoring

Utilizing a monitoring mechanism is essential to increase productivity and maintain overall quality of our workflow. One of the main reasons why we chose Travis CI as a tool is that it provides monitoring capabilities for the CI/CD pipeline. Travis CI provides a dashboard that displays the status of each build and test in real-time. We are able to see which builds passed and which failed, and we can get an overview about logs and output from each build and test.



A dashboard showcasing the status of each build

As here shows that the commit has been deployed after the build and test is done

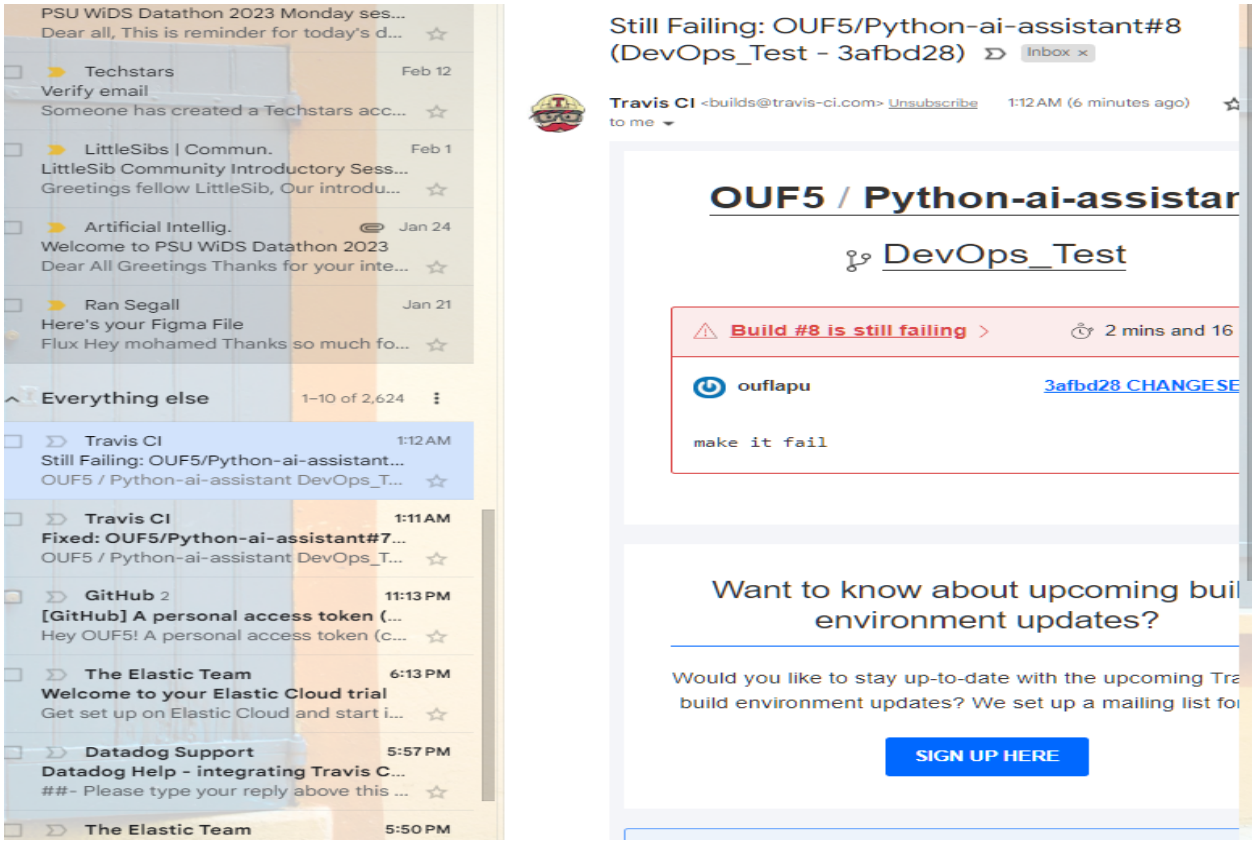


Providing metrics and general information about successful/failed builds

Travis CI can also send notifications to team members via email when a build or test fails or succeeds which will allow us to react quickly for feedback and troubleshooting.

```
43        on:
44          branch: develop
45
46    # stages:
47    #   - name: test
48    #     if: NOT (branch IN (develop,master)) AND (NOT(branch==^Release.*))
49    #   - name: merge (develop --> master)
50    #     if: branch==develop AND NOT(type==pull_request)
51    #   - name: deploy
52    #     if: branch==master
53
54    notifications:
55     email:
56         - mohammed5ibnouf@gmail.com
57         - 219211206@psu.edu.sa
58         - 218110378@psu.edu.sa
59
60
```

Travis configuration file with email notification option



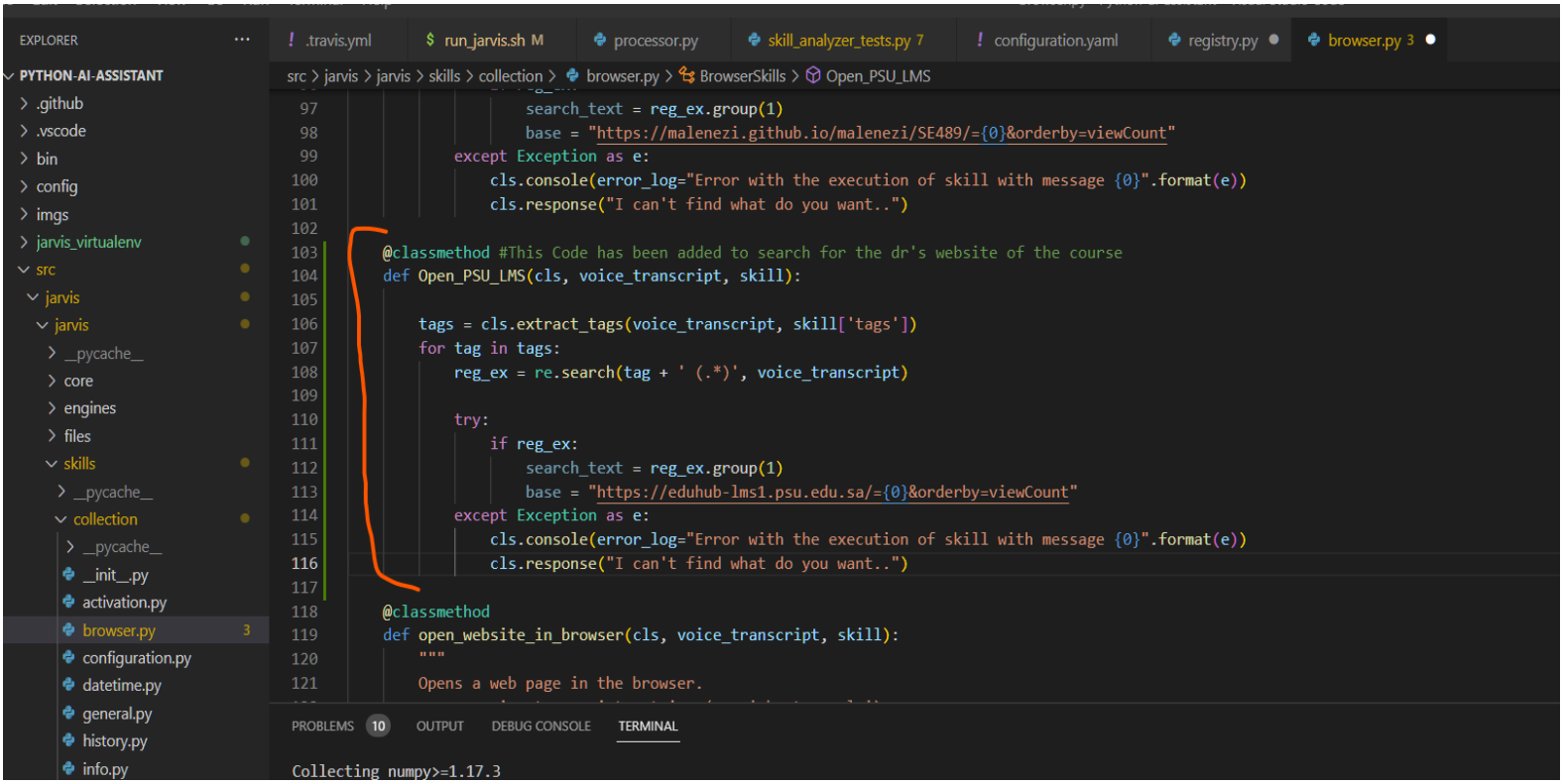
Email notification received

For these reasons, we have decided to use Travis CI's monitoring capabilities to help ensure that any issues or errors in the pipeline are caught early on and can be addressed quickly and that would help us to prevent code regressions and maintain code quality over time.

Phase 3

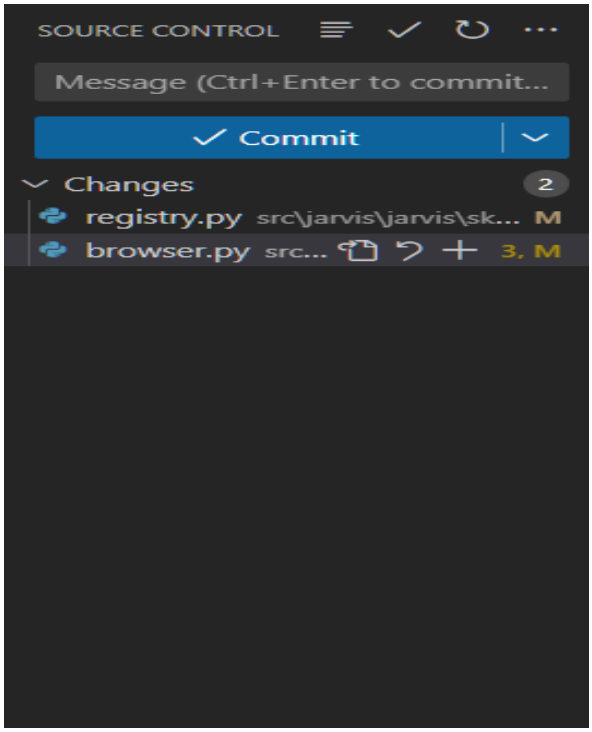
Second Cycle of the CI/CD Pipeline

We decided to add new skill for this phase also which will allow our AI assistant to open PSU's LMS system <eduhub-lms1.psu.edu.sa>



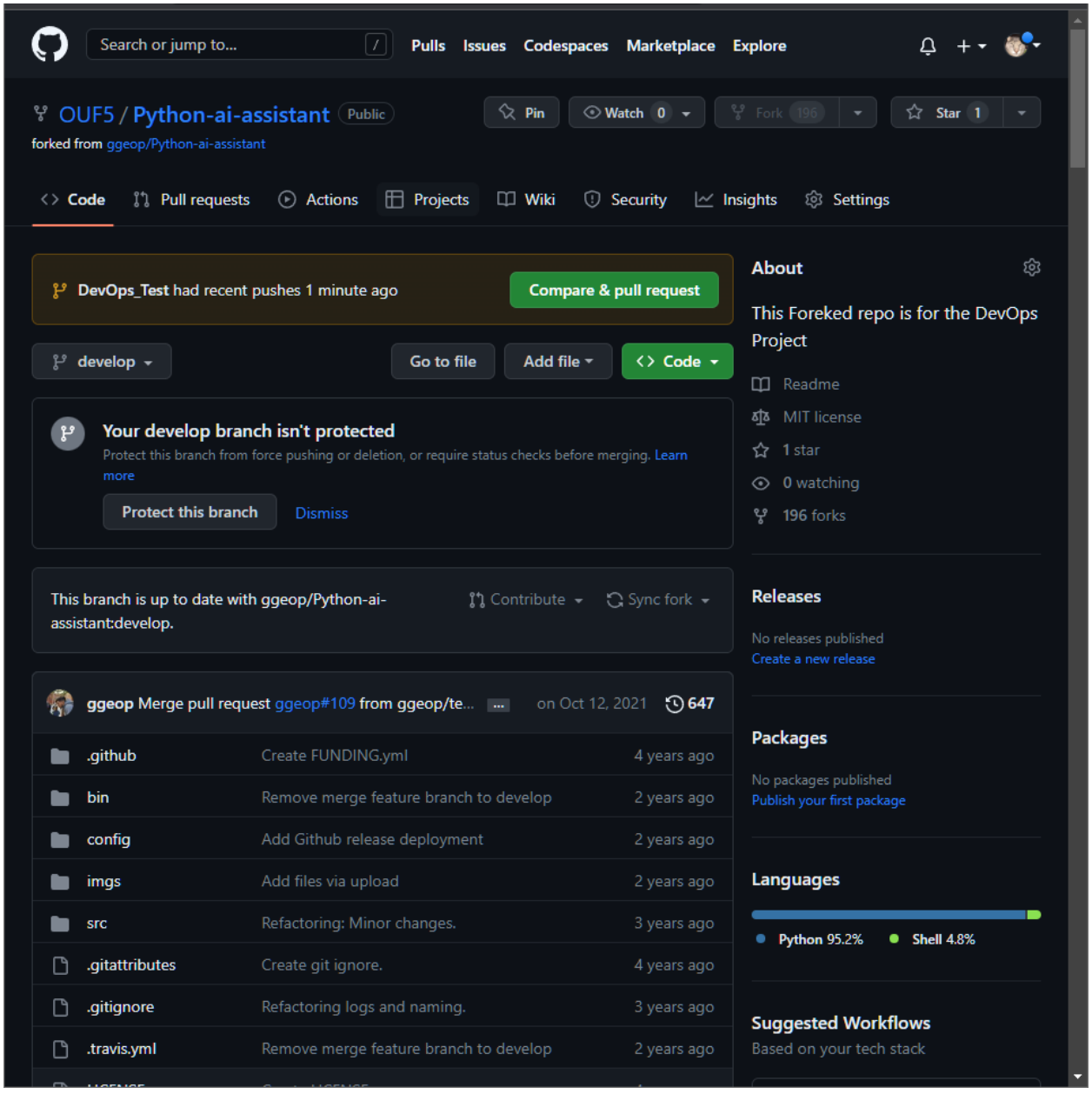
The code to open PSU LMS website

```
        'func': BrowserSkills.open_dr_website_course_SE489,
        'tags': 'open drs website for devops course',
        'description': 'Opens a domain in browser'
    },

    { # this part has been added for the DevOps course
        'enable': True,
        'func': BrowserSkills.Open_PSU_LMS,
        'tags': 'open the psu lms',
        'description': 'Opens a domain in browser'
    },

    {
        'enable': True,
        'func': BrowserSkills.tell_me_today_news,
        'tags': 'news, today news',
        'description': 'Tells the daily news (find on Google newsfeed)'
    },

    {
        'enable': True,
        'func': DatetimeSkills.tell_the_time,
        'tags': 'time, hour',
        'description': 'Tells the current time'
    },
```

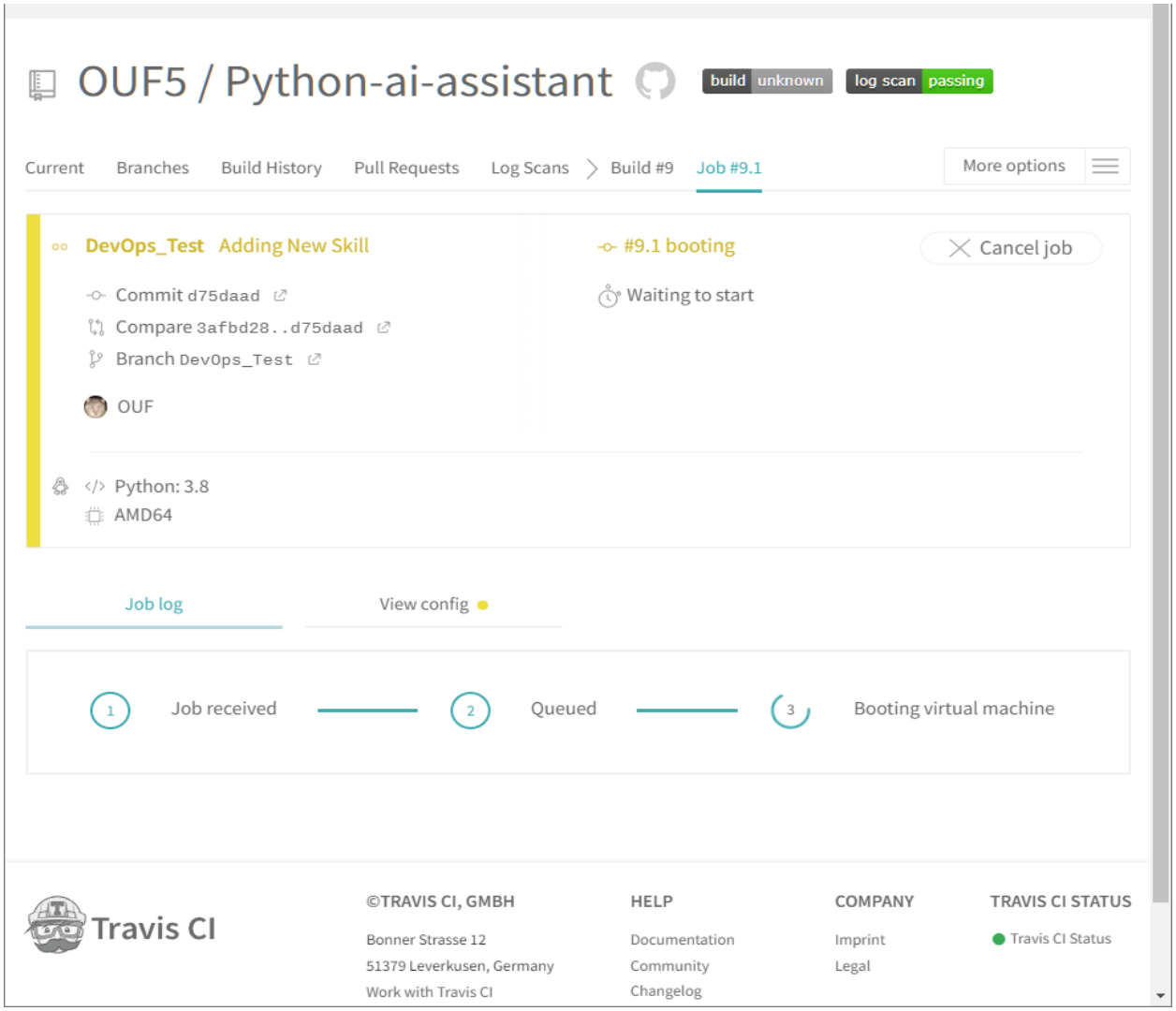


The commits to our github repository through VSCode

The changes that we have committed is pushed to the main repository to initiate the auto building by Travis CI which will start the build and testing processes

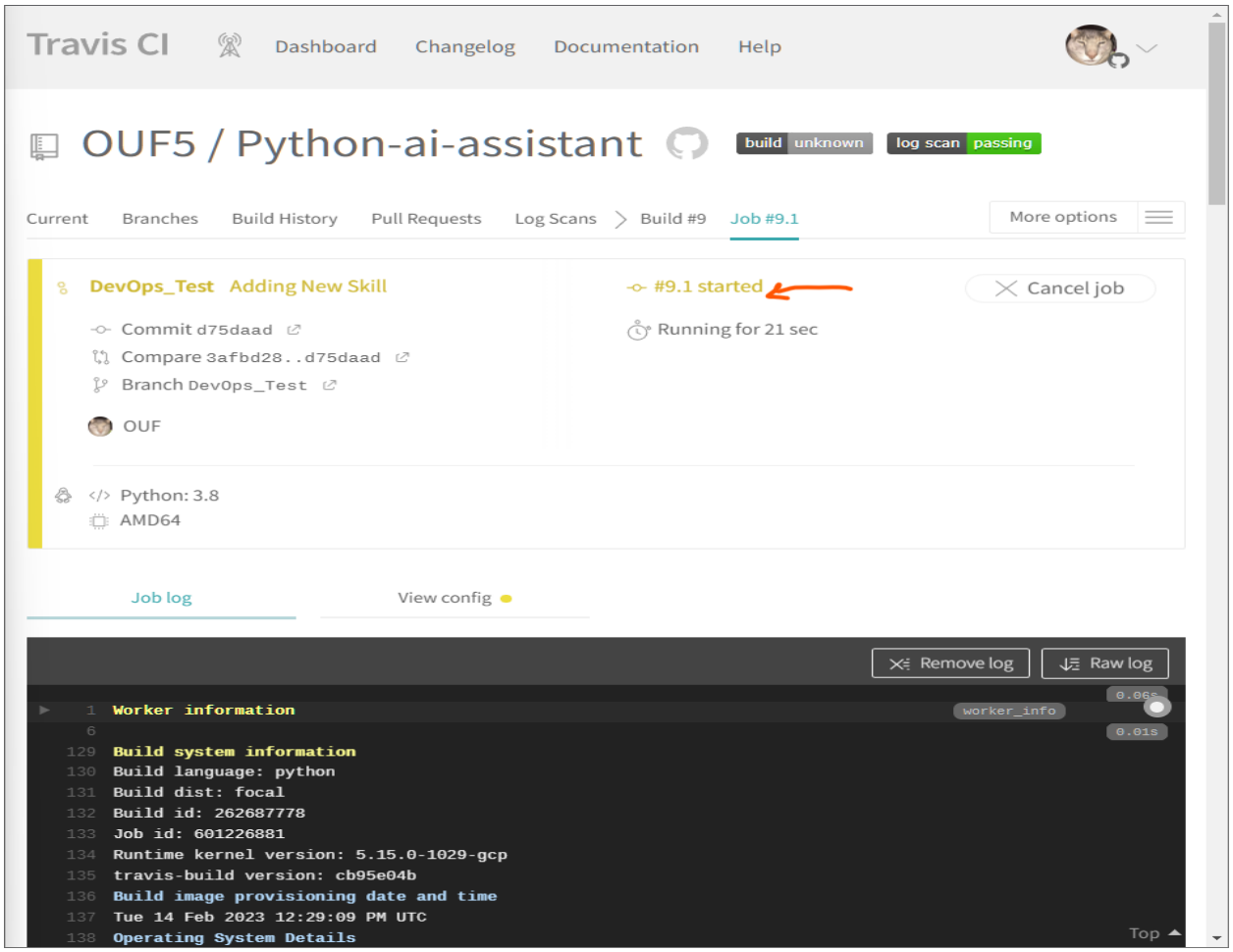


The push and commit in GitHub



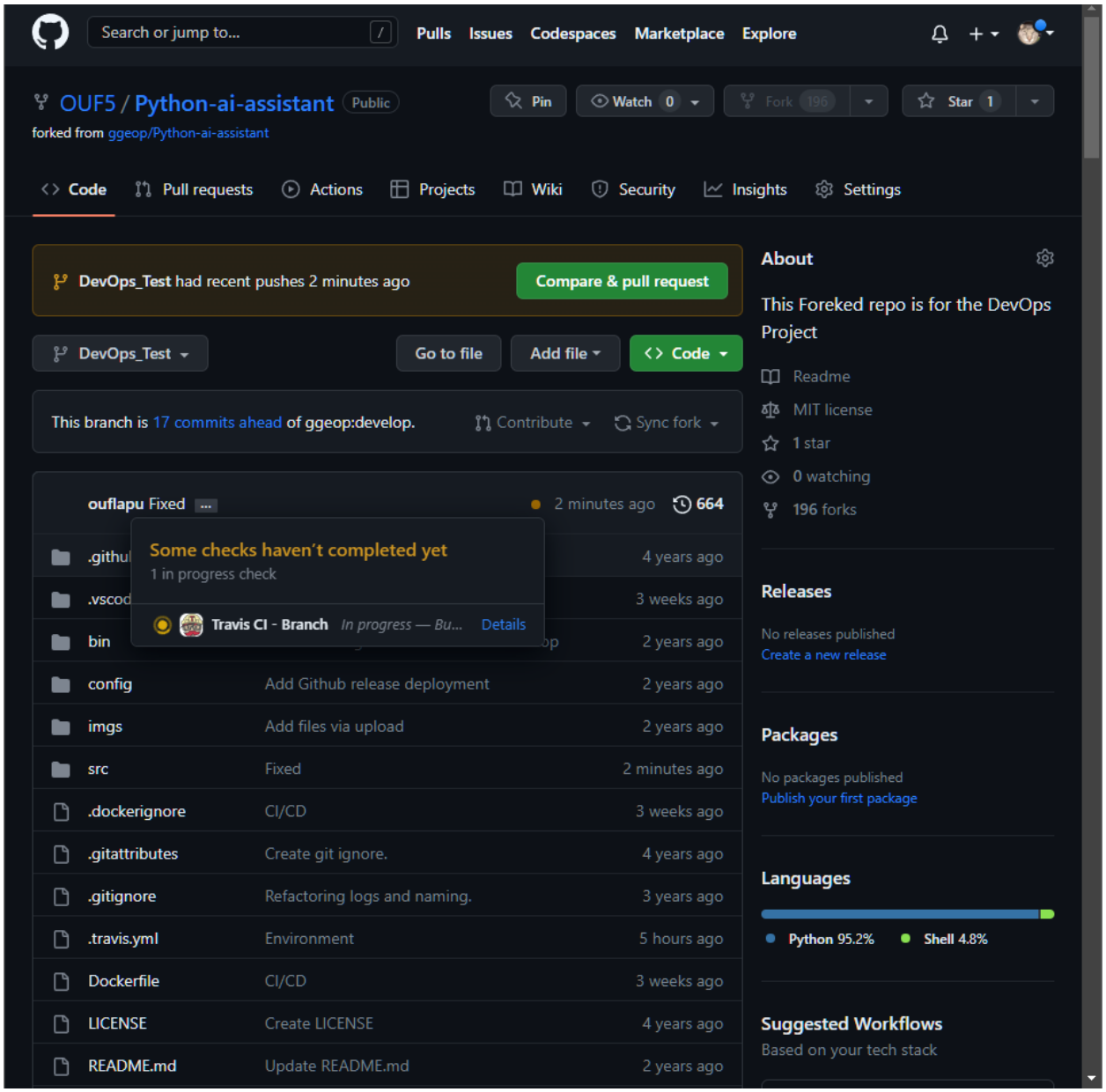
Booting status involves setting up the necessary software, libraries, and dependencies required for the build.

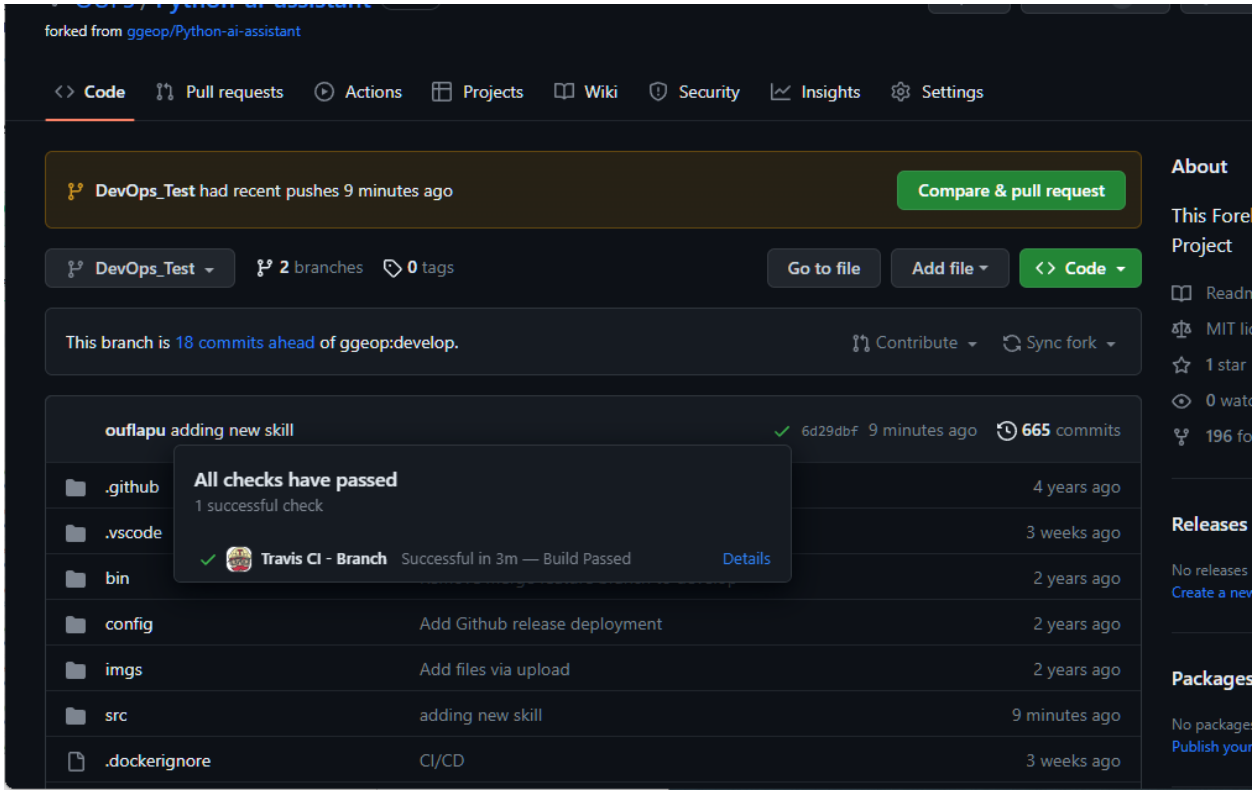
Here is the starting process of running the build itself. This involves running the build script, which typically includes steps like compiling code, running tests, and packaging artifacts.

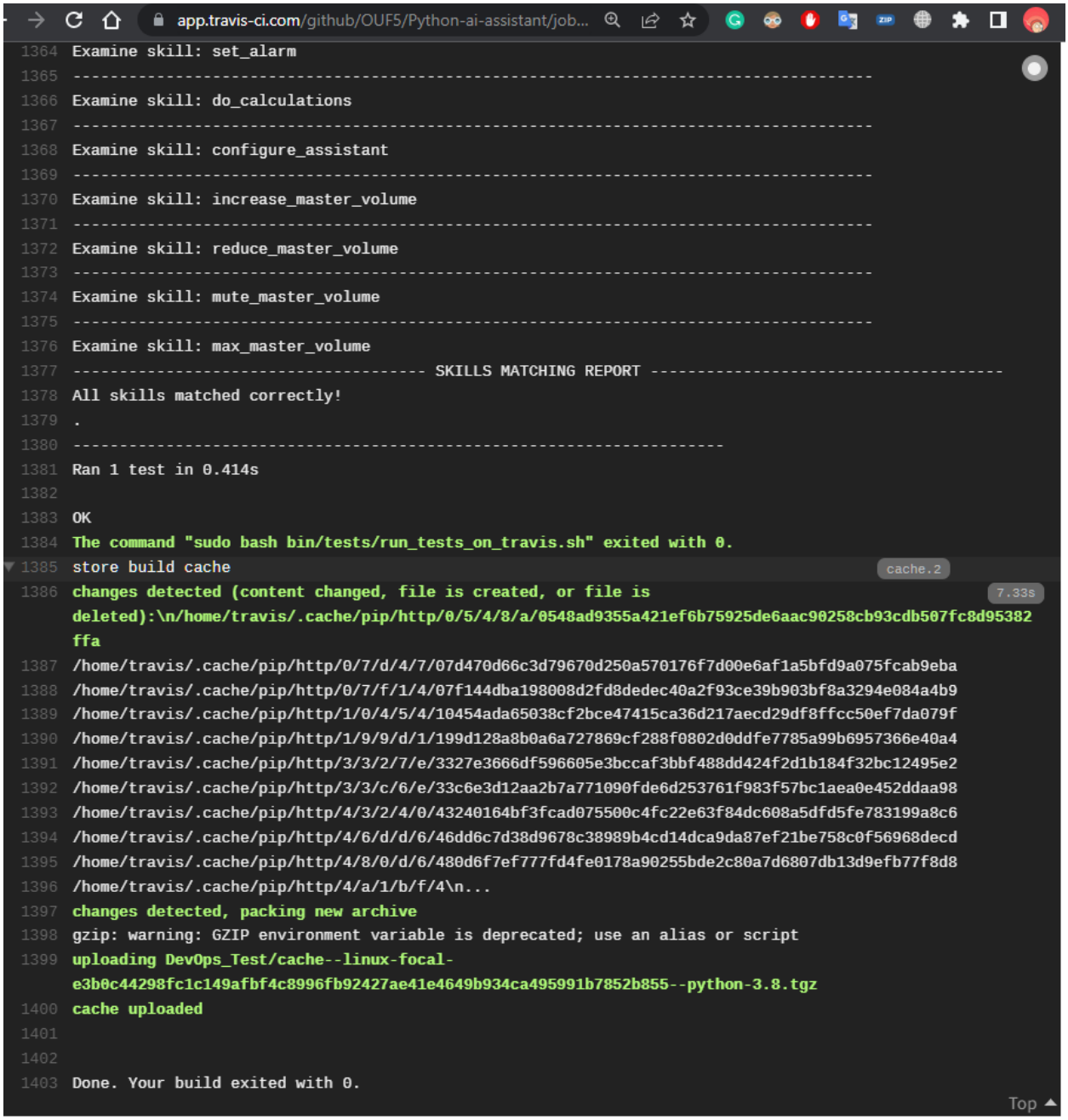


Started status

Travis CI is providing a notice on the main repository in github of the building process. The build is successful at the end of the new skill addition.

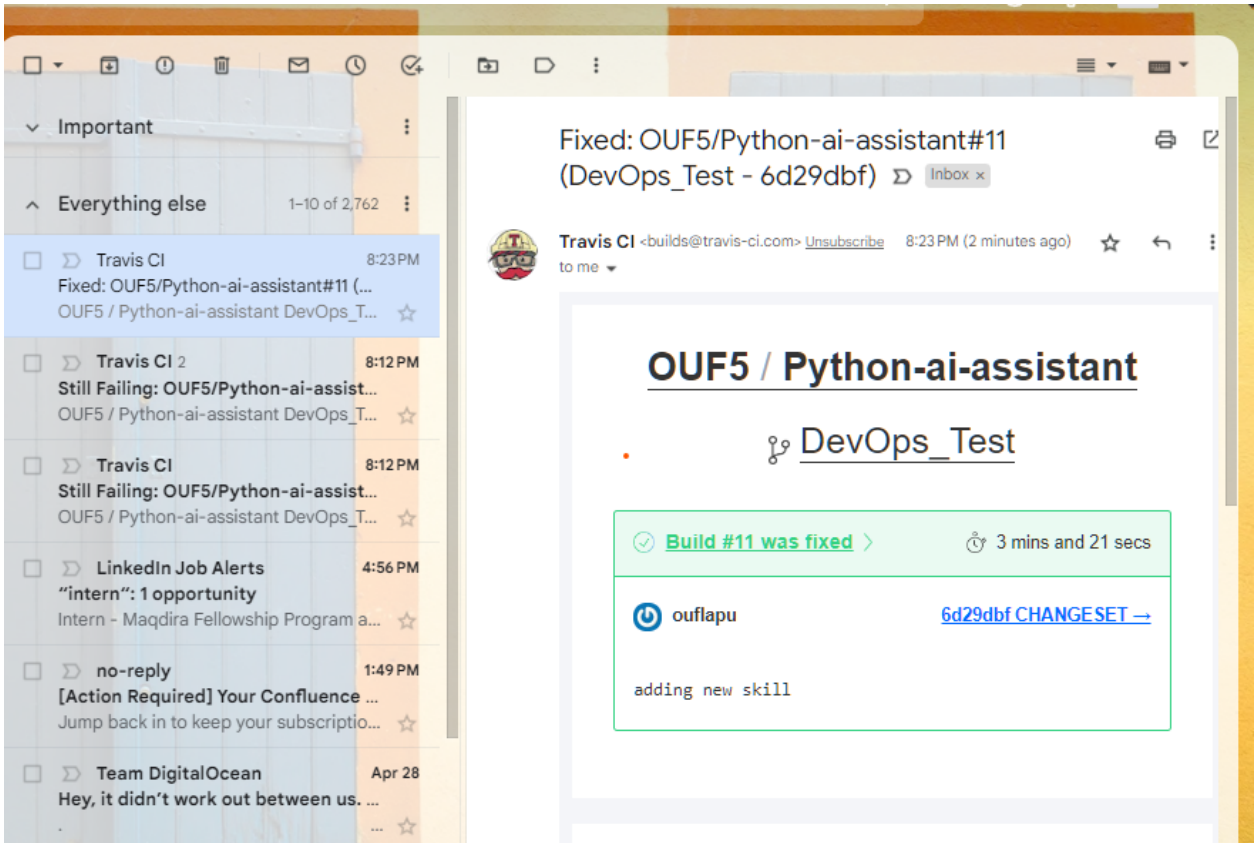



forked from ggeop/Python-ai-assistant
Code
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
DevOps_Test had recent pushes 9 minutes ago
Compare & pull request
DevOps_Test
2 branches
0 tags
Go to file
Add file
Code
This branch is 18 commits ahead of ggeop:develop.
Contribute
Sync fork
ouflapu adding new skill
6d29dbf 9 minutes ago
665 commits
All checks have passed
1 successful check
Travis CI - Branch Successful in 3m — Build Passed
Details
.github 4 years ago
.vscode 3 weeks ago
bin 2 years ago
config Add Github release deployment 2 years ago
imgs Add files via upload 2 years ago
src adding new skill 9 minutes ago
.dockerignore CI/CD 3 weeks ago
About
Releases
Packages



Travis CI Job Log shows that all the skills have been passed the testing.

Part of the monitoring capabilities in Travis CI which we had relied on in the previous phase. Monitoring is done via Travis email notifications and a report on the main dashboard.



Email notification received

A dashboard showcasing the status of each build