**Building a CI/CD Pipeline**.

1. General

This project is to build a Github repository and create a scaffolding that will assist in performing both Continuous Integration and Continuous Delivery.

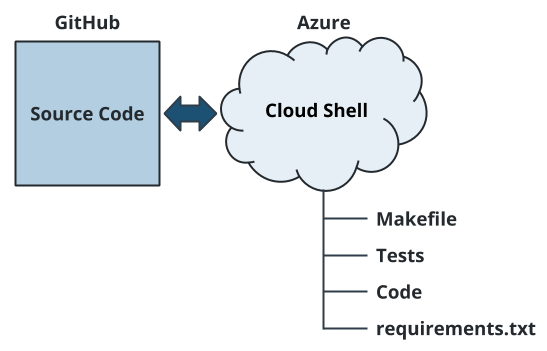
It uses Github Actions along with a Makefile, requirements.txt and application code to perform lint, test, and install.

Then it integrates it with Azure Pipelines to enable Continuous Delivery to Azure App Service.

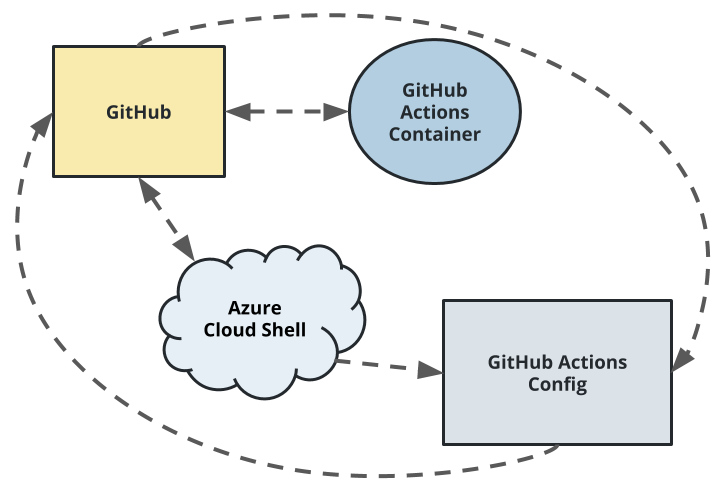
It is to demonstrate how to perform continuous delivery for a Python-based machine learning application using the Flask web framework. We were provided a pre-trained, sklearn model that has been trained to predict housing prices in Boston.

Architecture:

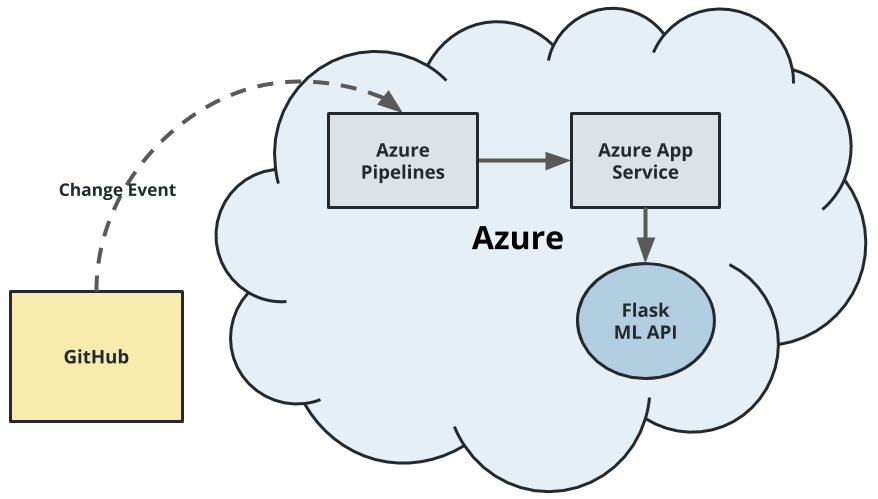
The Azure Cloud Shell that will be used to create an scaffold of tests, source code, Makefile, and requirements.



GitHub Actions - shows how code can be tested automatically by enabling. The push change to GitHub triggers the GitHub Actions container, which in turn runs a series of commands.



CI-continuous delivery through the use of Azure Pipelines and Azure App service.



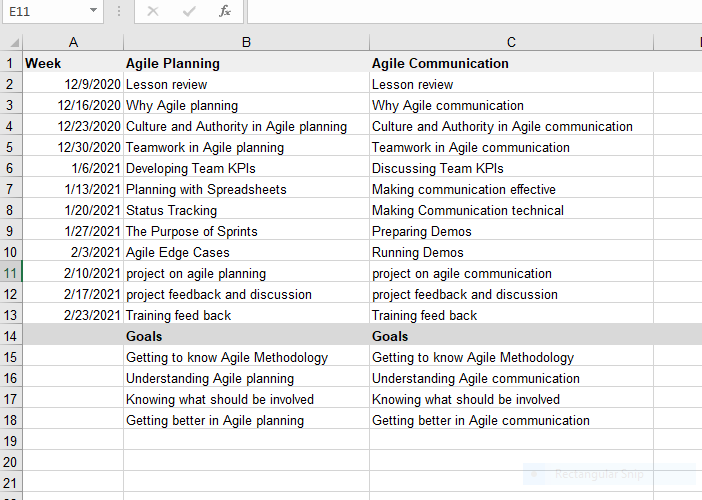
Your setup in the Azure console will involve a similar workflow as the screenshot. You will enable source control integration, select the Azure Pipelines to build provider, and finally configure your App Services permissions.

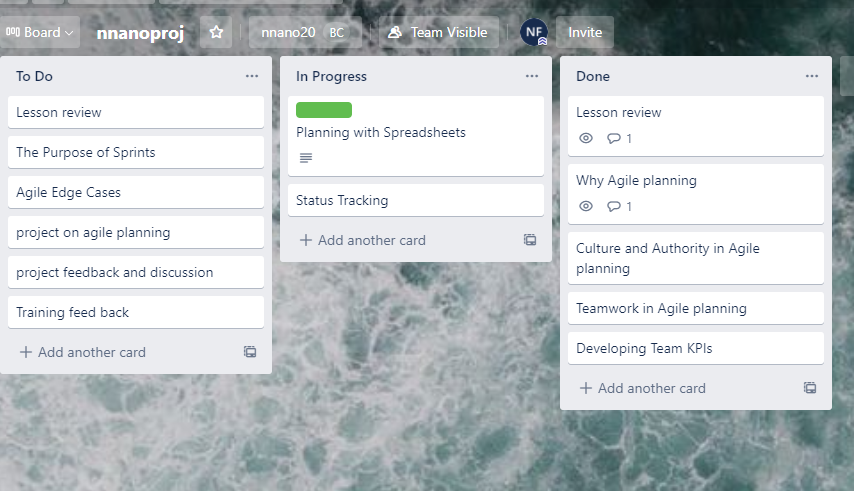
## 1. Create the Cloud-Based Development Environment

It is time to set up an initial project structure in the Azure Cloud Shell environment. First, you'll create a Github repository. Next, you'll launch an Azure Cloud Shell environment and integrate Github repository communication.

2. Project Plan

(With all my apologies, I did not create a project plan for the project, the one I will be showing here is the one I created during the lesson exercise.)





3. CI: set Up Azure Clou Shell

This step sets up a cloud-based development structure using Azure Cloud Shell creating a Makefile, tests, and application scaffolding. A scaffolding code was provided.

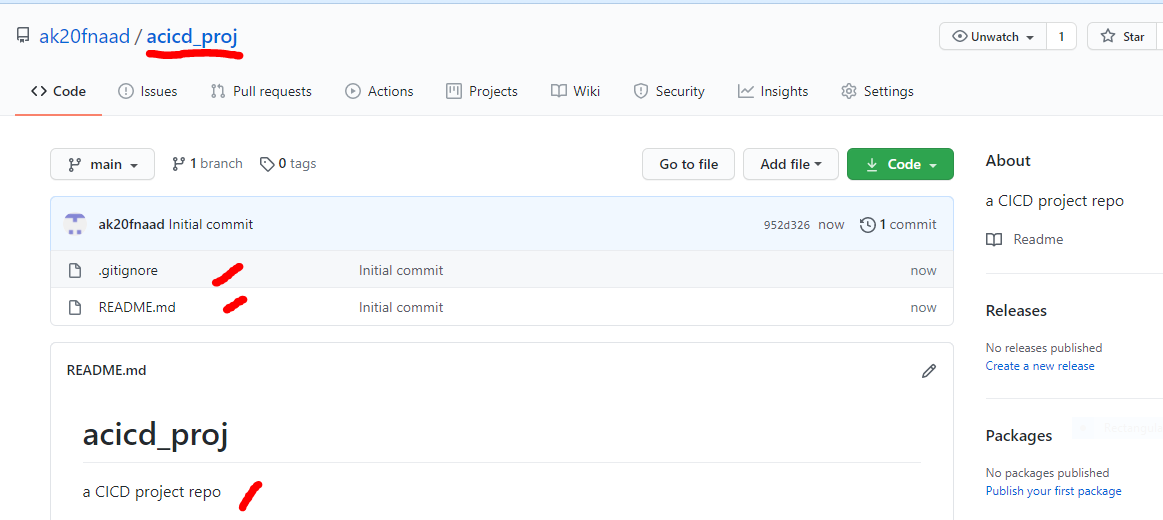
Once you complete this step you will be able to test code locally in the Azure Cloud Shell. This is a local continuous integration step. All Python that is created should have a local continuous integration setup as well as a remote build server that runs the same code.

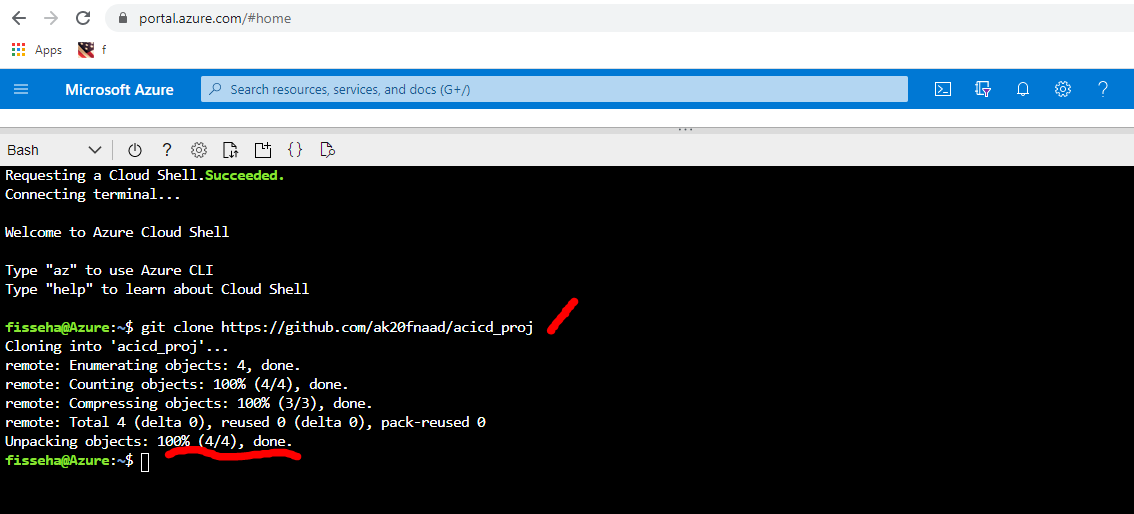
Create the Cloud-Based Development Environment that includes:

The GitHub Repo

Launch Azure Cloud Shell environment

Creating ssh-keys, and uploading them to the GitHub account

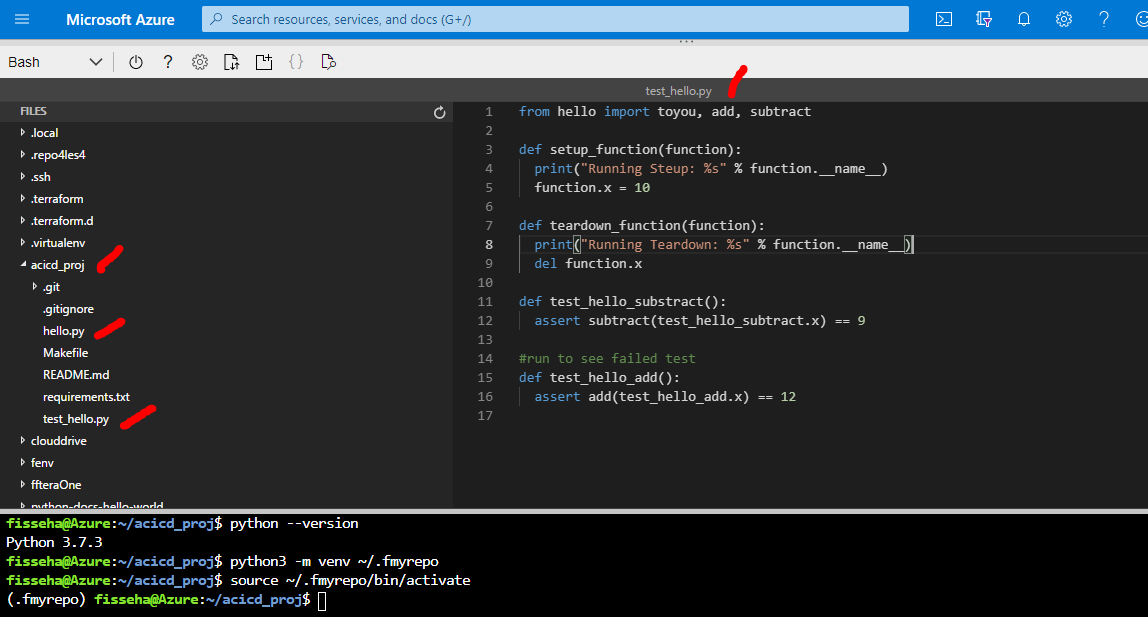




2 Create Project Scaffolding

After the environment was set up the following were created for the test

* Create the Makefile,
* requirements,
* the Python virtual environment
* script and test files



3 Local test

This was to run the make file

Run make all

## 

## 

4 CI: Config GitHub Actions

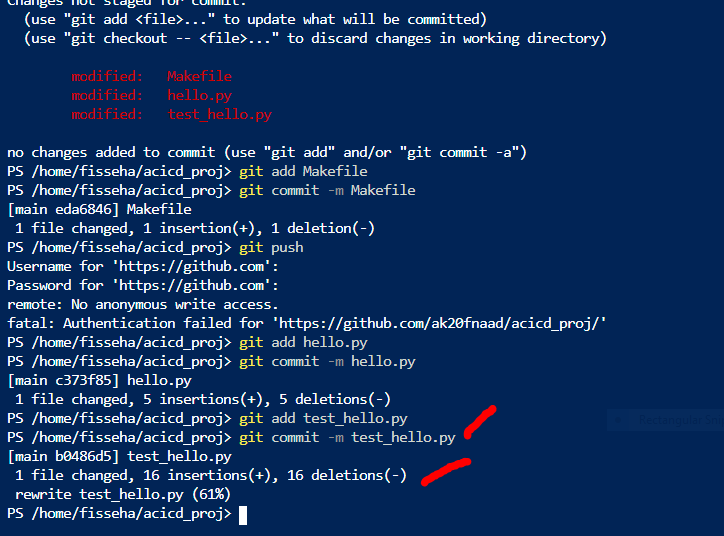
This is for configuring GitHub Actions to test the project upon change events in GitHub. It is a necessary step to perform Continuous Integration remotely.

Enable Github Actions from the account

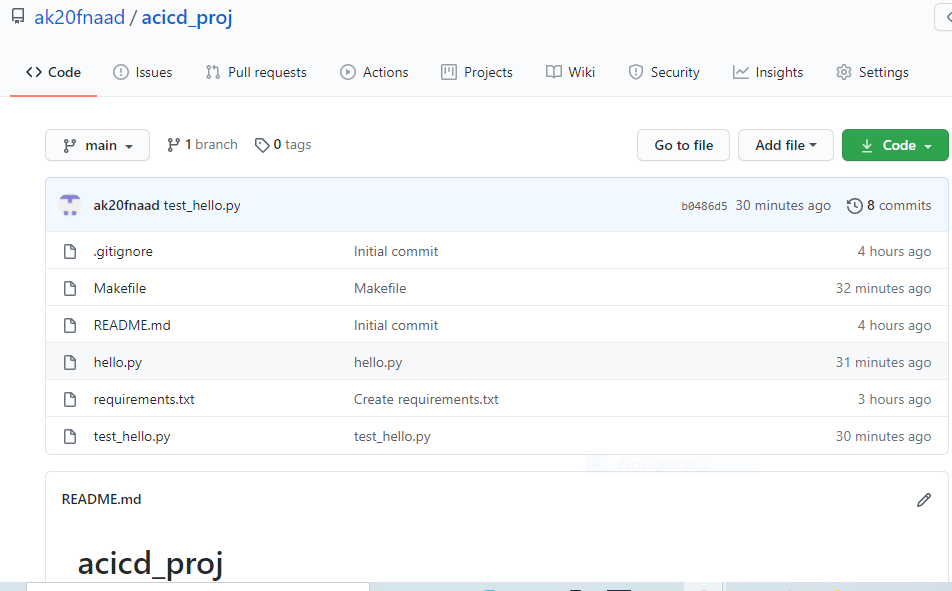
Replace yml code with the provide scaffolding one

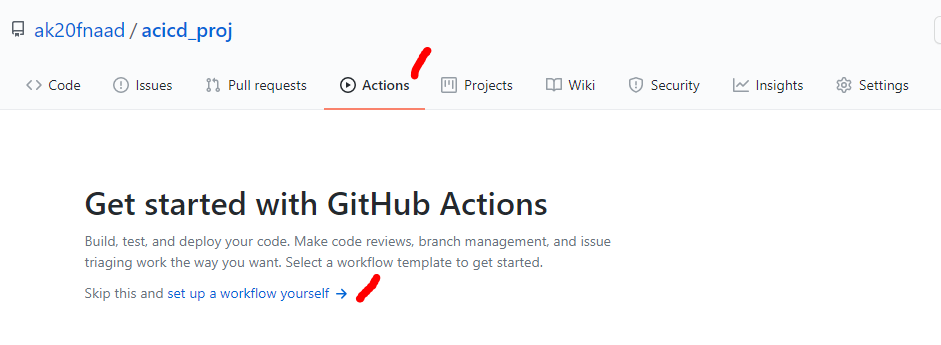
Verify remote test pass in Github Actions UI

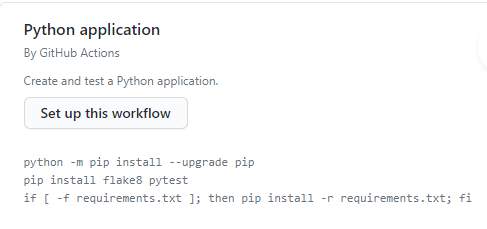
Add bage to README.md

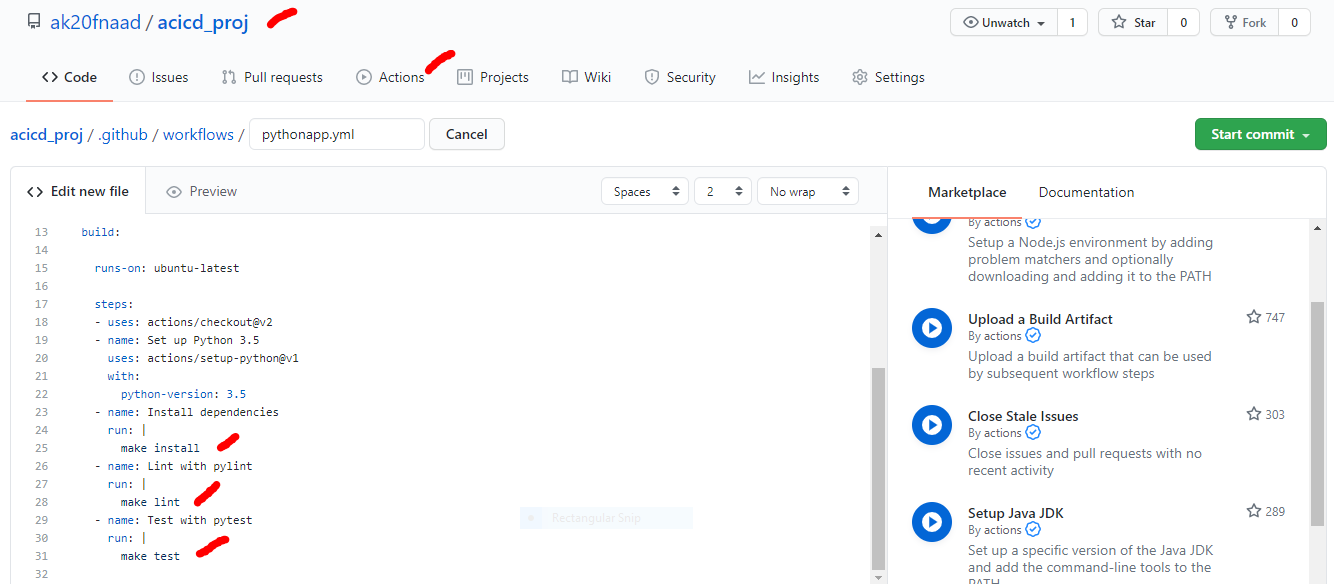


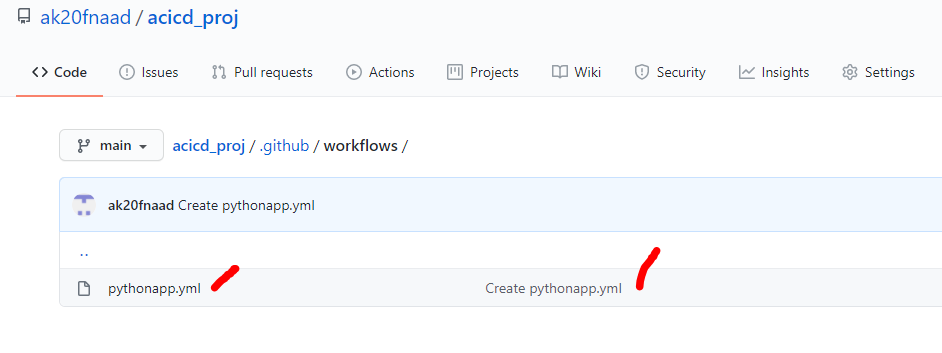


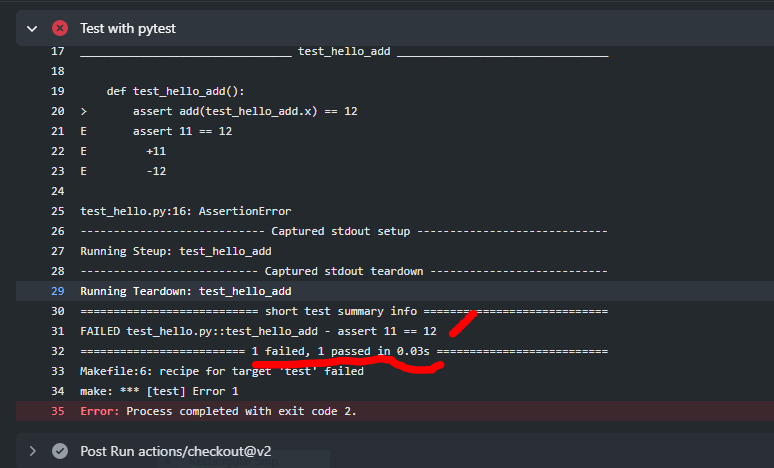






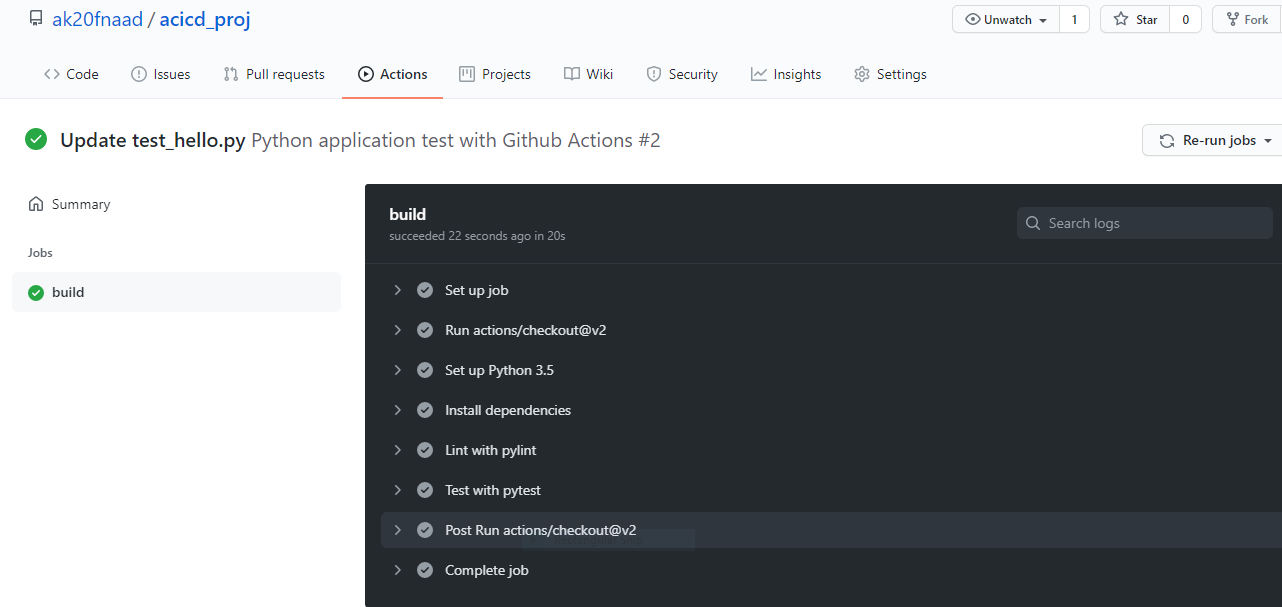


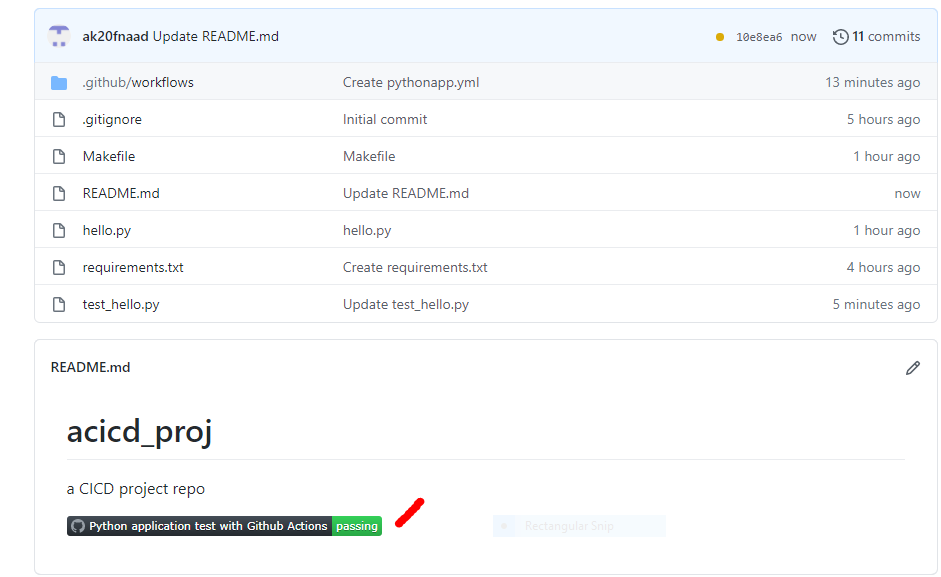




Correcting the error:







5 Continuous Delivery on Azure

This is for setting up Continuous Delivery using Azure technologies. This will involve setting up Azure Pipelines to deploy the [Flask starter code](https://github.com/udacity/nd082-Azure-Cloud-DevOps-Starter-Code/tree/master/C2-AgileDevelopmentwithAzure/project/starter_files)  (provided) to Azure App Services.

Authorize and enable Azure App Service

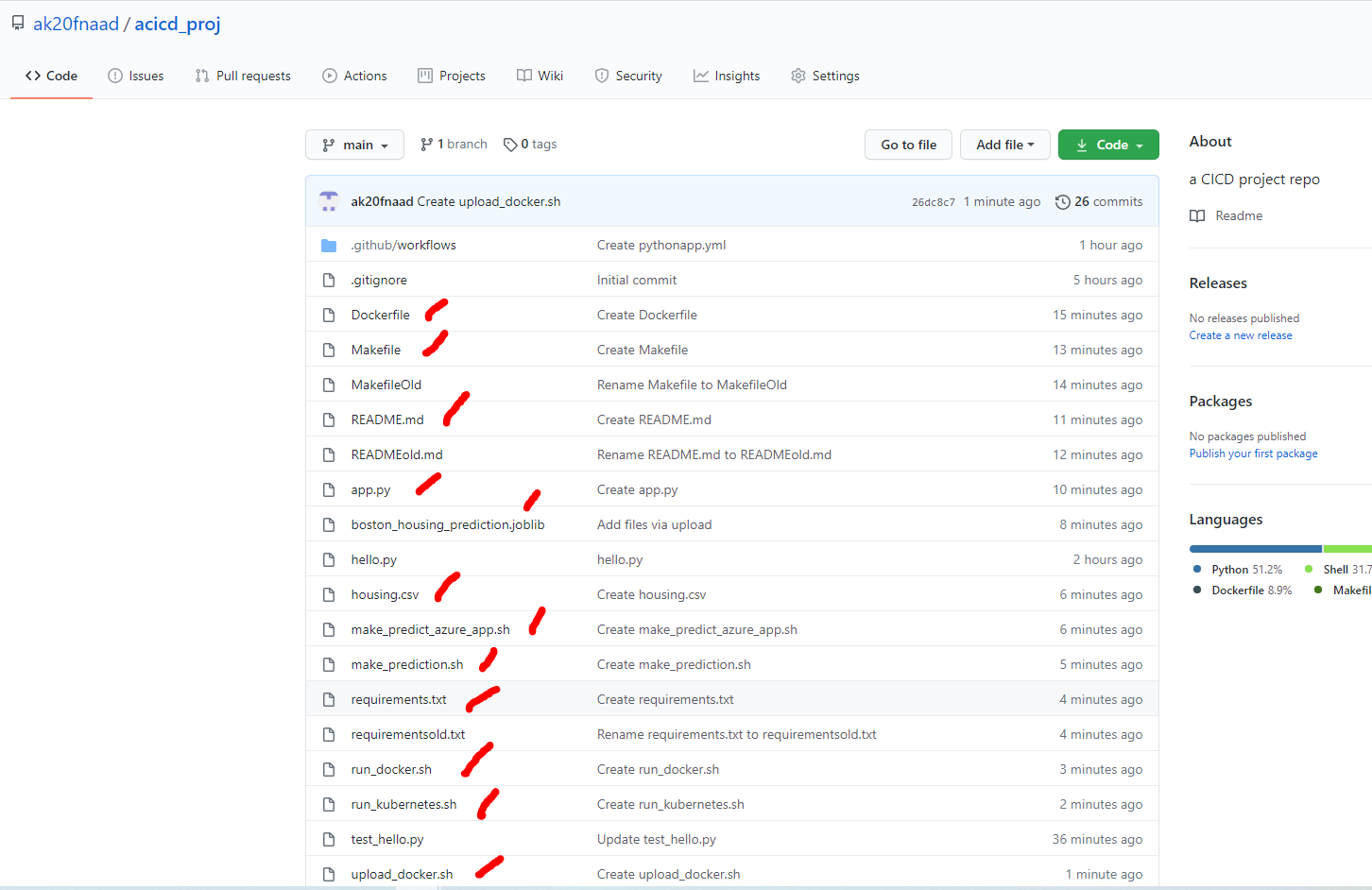
Enable contoinuous deployment with Azure Pipelines

Checking YMAL based config file into Github

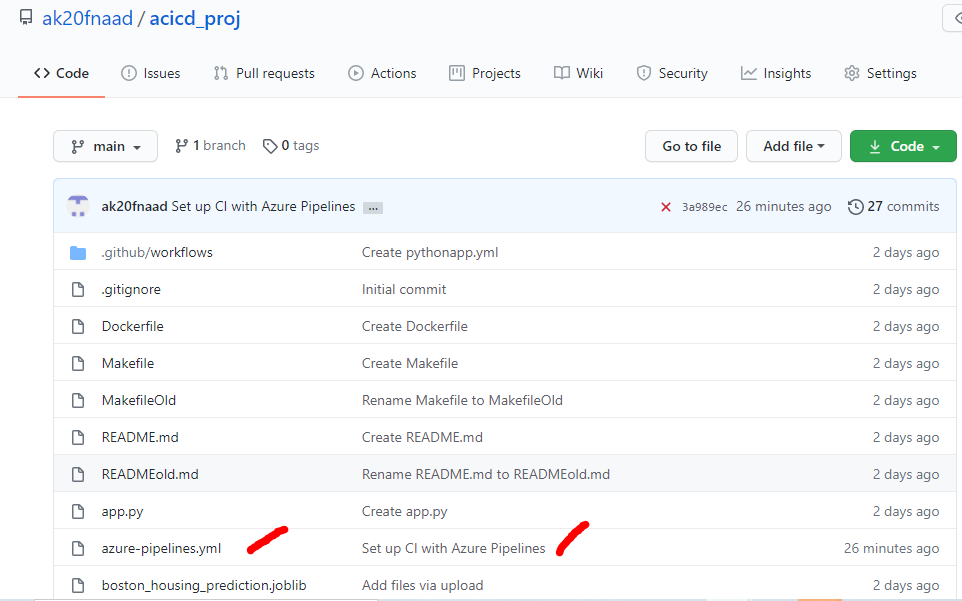
Enable Github + Azure Pipelines

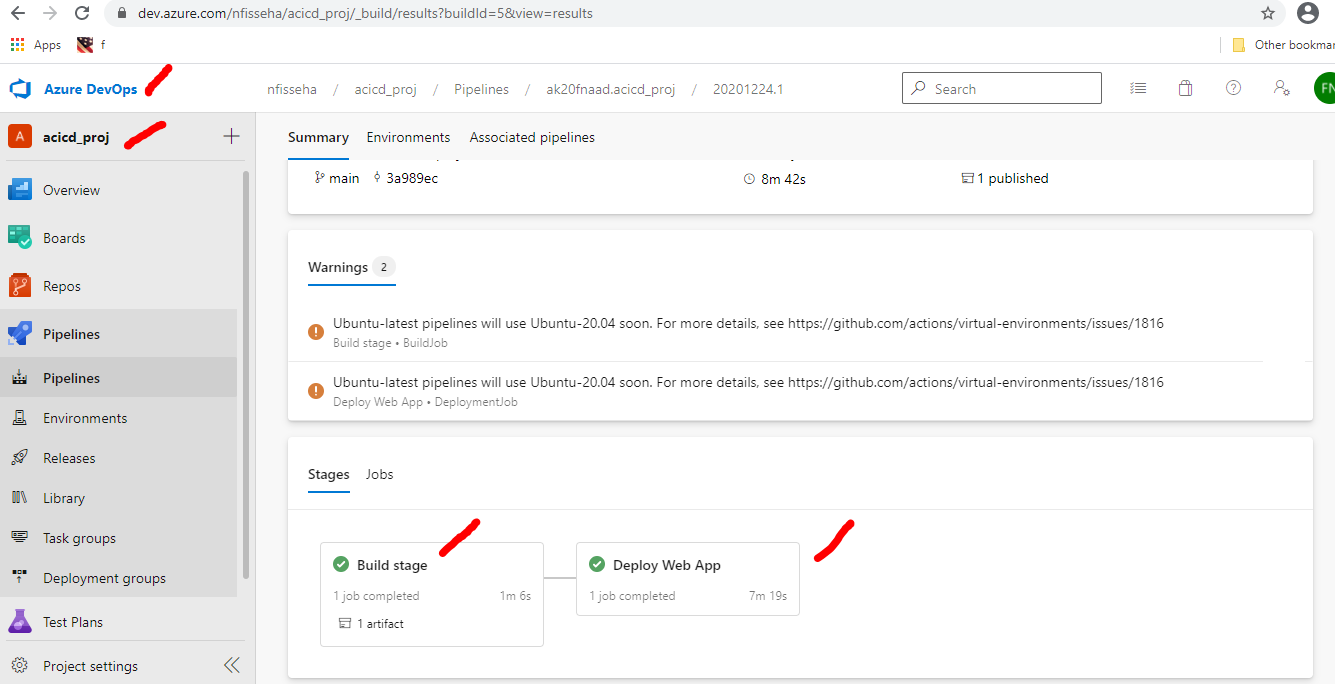
Deploy the project in Azure pipelines

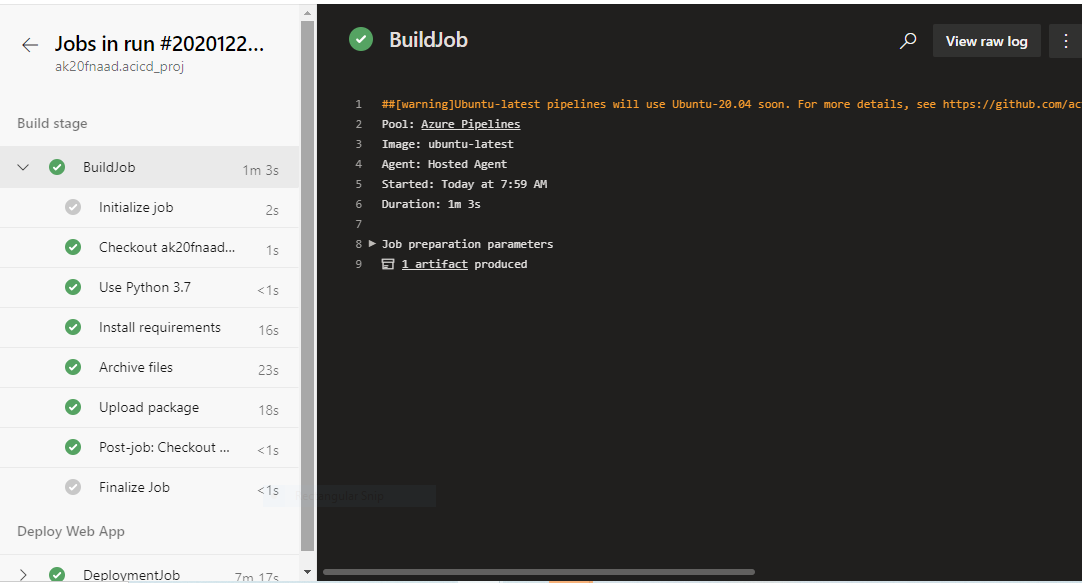
Verify Prediction with starter code

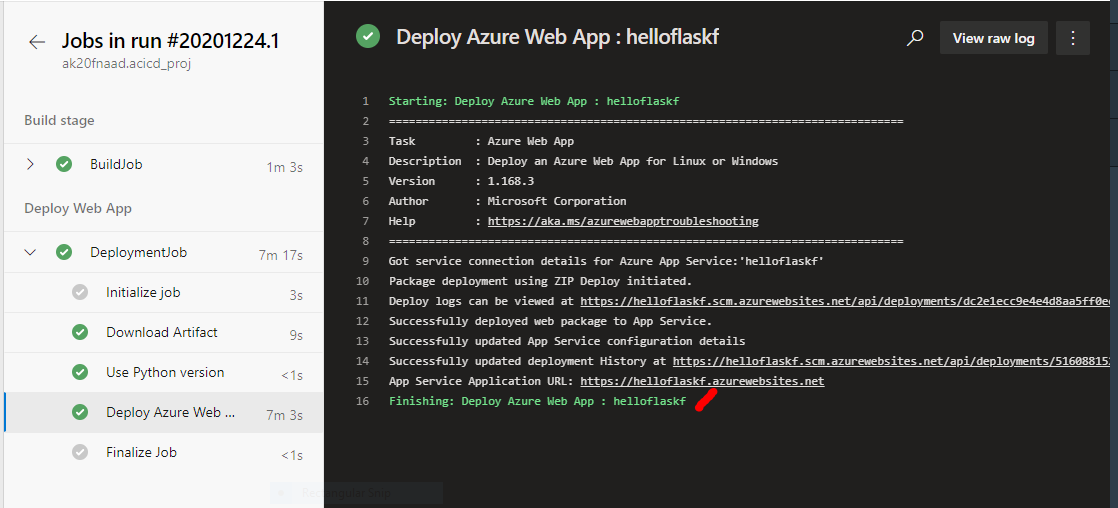


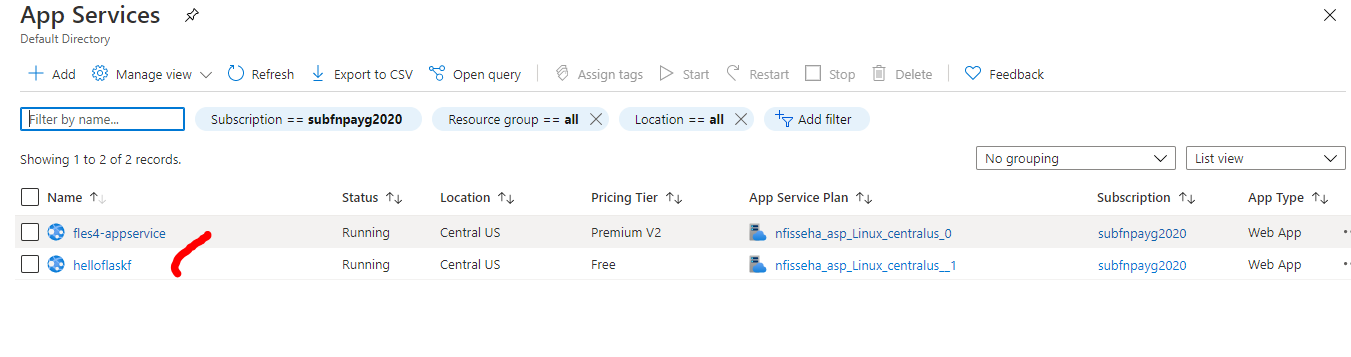
## 

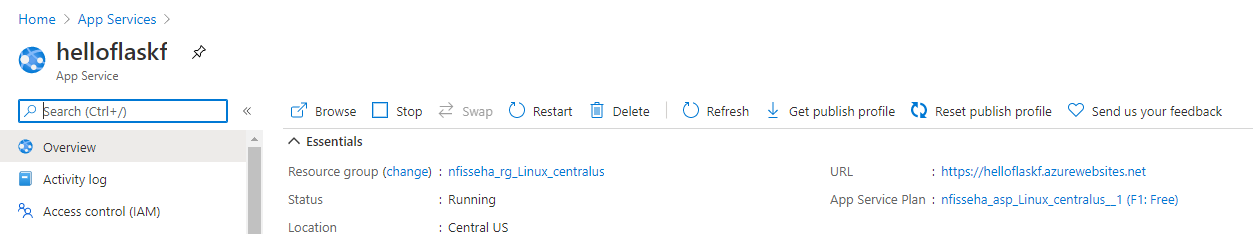


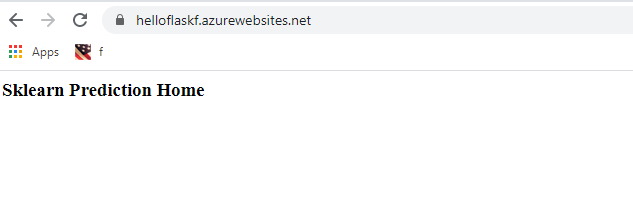












Prediction:

