
National University
of computer and emerging sciences

# Assignment No 1

# MLOps

**Names:**

**Abdullah Shakeel (i19-1717)**

**Haris Usman (i19-1965)**

**Section:**

**B**

## GitHub repository link:

https://github.com/NUCES-ISB/i191717_i191695_polymath

## Description:

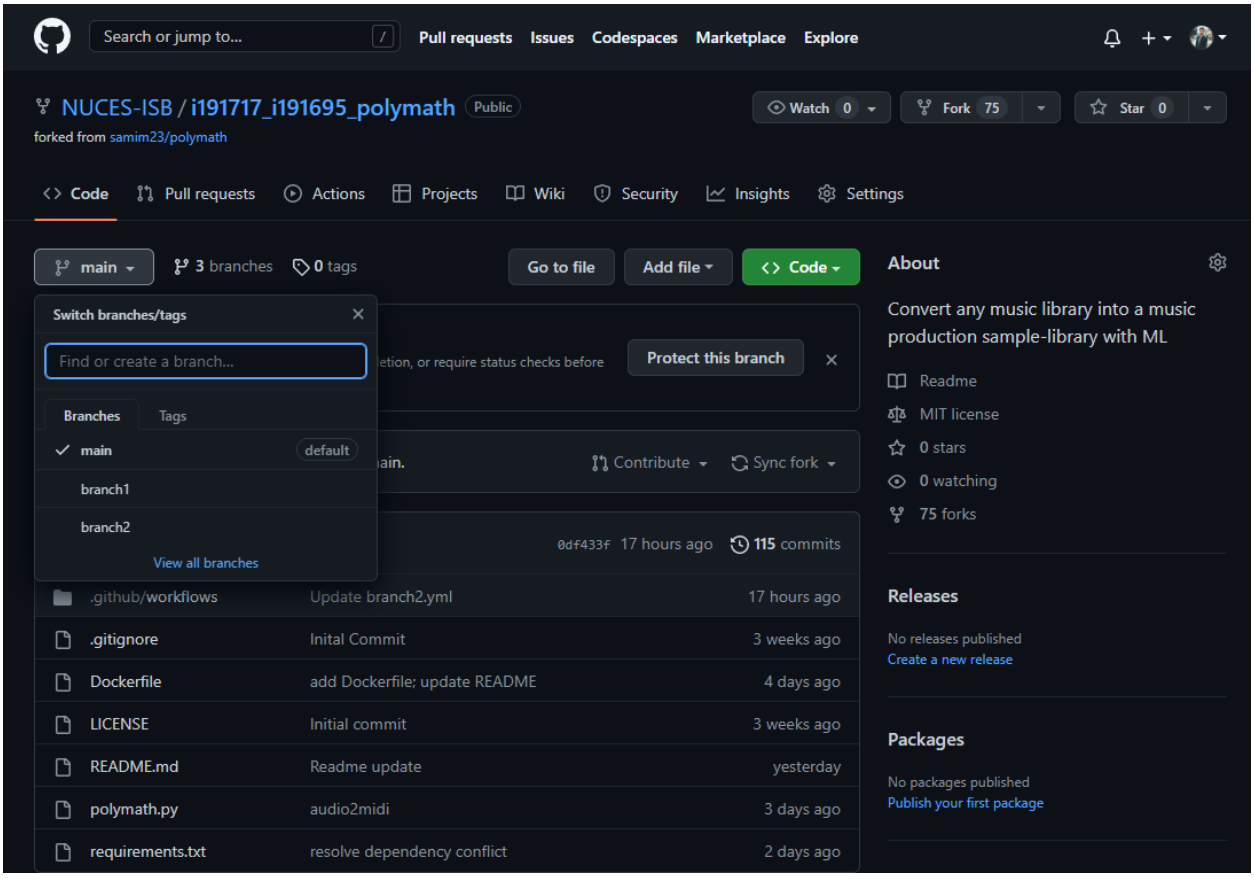
## Repository selection:

We forked the following repository and according to requirement we do not own this repository directly or indirectly, the selected repository is in trending and workflow does not exist for the selected repository.

**Repo link**: https://github.com/samim23/polymath

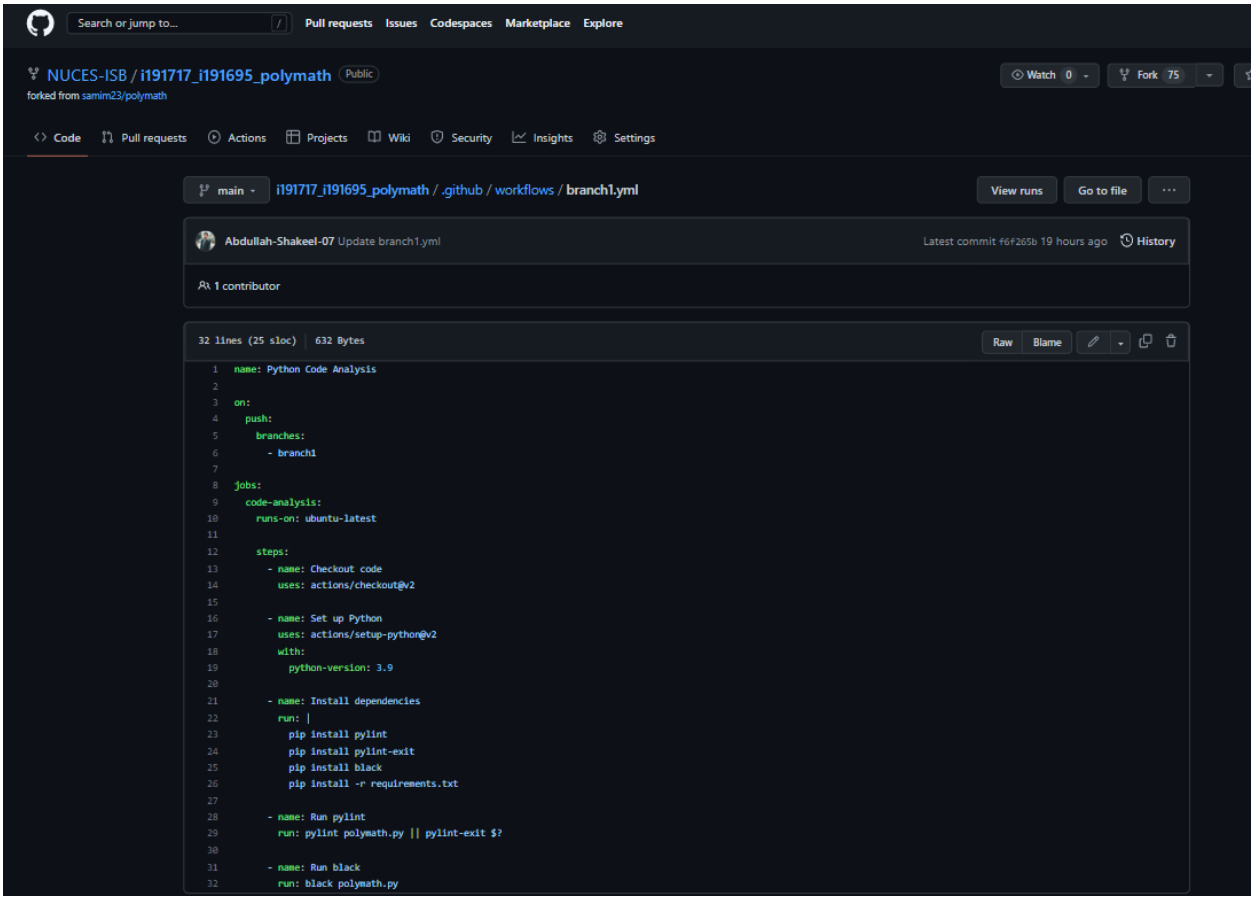
## Workflow creation:

First of all, we created two branches one for each group member.




As you can see in the above picture, there are two branches named branch 1 and branch 2.

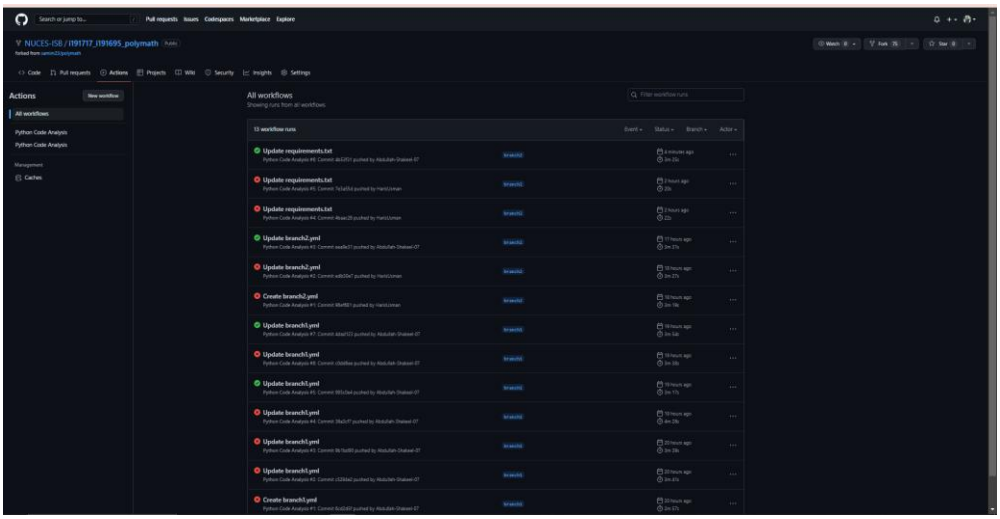
Then we created workflows for each branch and run the test (pylint and black as our repo contain python code). The structure of our yml file is as follows




**Description of above code:**

The above code defines a job named "code-analysis" that runs on the “branch1” (“brach2” uses the same code) branch of a repository whenever a push event occurs. The job uses the "ubuntu-latest" runner environment and has four steps.

The first step checks out the source code of the repository using the "actions/checkout" action. The second step sets up the Python environment using the "actions/setup-python" action and specifies python version 3.9. The third step installs multiple packages, including pylint, pylint-exit, black, and the packages listed in the "requirements.txt" file. The fourth step runs two commands: pylint and black. The pylint command checks the quality of the Python code in a file named "polymath.py" using the pylint package, and if there are any errors or warnings, it exits with a non-zero status code. The pylint-exit command is used to fail the build if pylint has detected any issues in the code. The fifth step formats the code using the black package.



As you can, after a couple of tries we have successfully tested both branches.

## Result of pylint

```
polymath.py:527:12: W0612: Unused variable 'comp_intensity' (unused-variable)
polymath.py:553:0: C0116: Missing function or method docstring (missing-function-docstring)
polymath.py:553:0: C0103: Function name "getNearest" doesn't conform to snake_case naming style (invalid-name)
polymath.py:560:0: C0116: Missing function or method docstring (missing-function-docstring)
polymath.py:560:0: R0914: Too many local variables (26/15) (too-many-locals)
polymath.py:591:12: C0103: Variable name "v" doesn't conform to snake_case naming style (invalid-name)
polymath.py:630:8: C0103: Constant name "basic_pitch_model" doesn't conform to UPPER_CASE naming style (invalid-name)
polymath.py:630:8: W0603: Using the global statement (global-statement)
polymath.py:637:4: C0103: Variable name "keepOriginalBpm" doesn't conform to snake_case naming style (invalid-name)
polymath.py:640:4: C0103: Variable name "pitchShiftFirst" doesn't conform to snake_case naming style (invalid-name)
polymath.py:652:45: C0103: Variable name "f" doesn't conform to snake_case naming style (invalid-name)
polymath.py:671:45: C0103: Variable name "f" doesn't conform to snake_case naming style (invalid-name)
polymath.py:560:0: R0912: Too many branches (31/12) (too-many-branches)
polymath.py:560:0: R0915: Too many statements (100/50) (too-many-statements)
polymath.py:24:0: W0611: Unused predict imported from basic_pitch.inference (unused-import)

-----------------------------------
Your code has been rated at 6.02/10
```

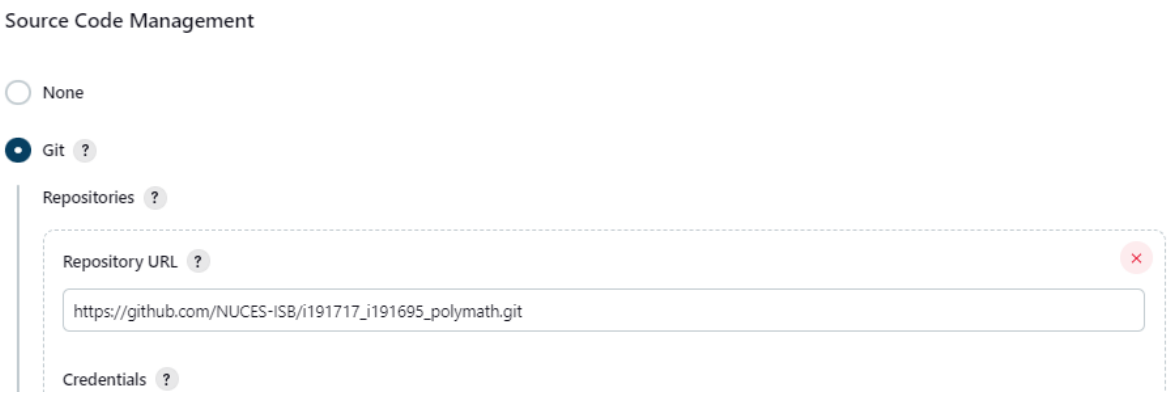
## Result of black

```
Run black
▶ Run black polymath.py
reformatted polymath.py

All done! ✨ 🍰 ✨
1 file reformatted.
```

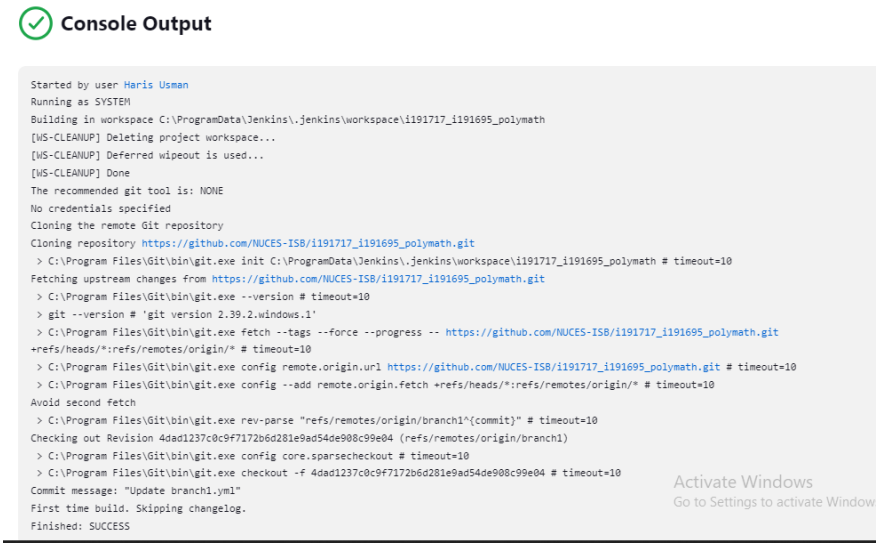
# Jenkins

For building a Jenkins job we ran Jenkins on localhost 8080 and used following GitHub link as shown in the image.

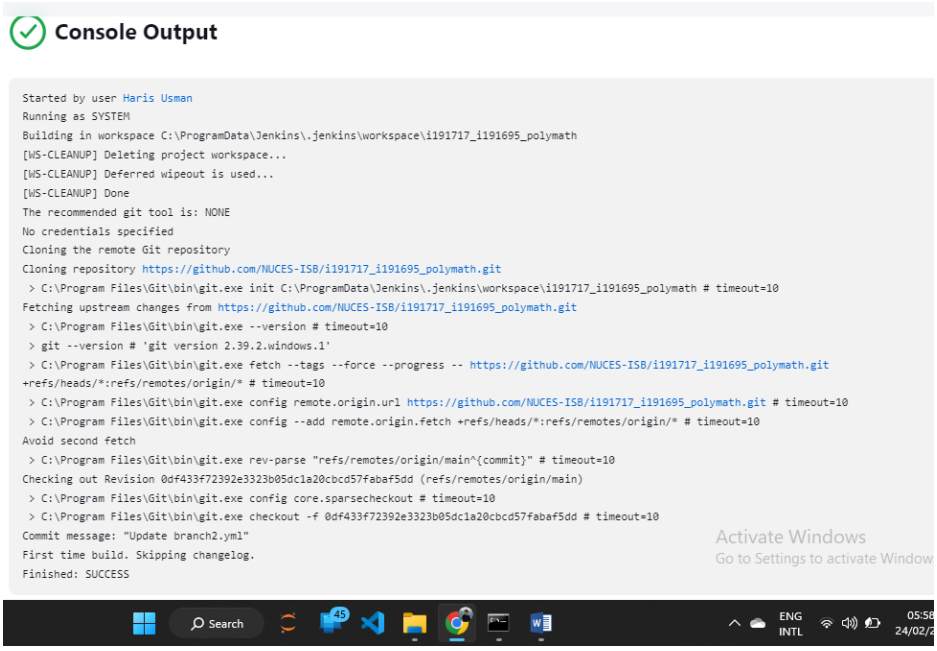



Then we executed build commands on both branches and master that where successful and their prof are attached in the following images

# Build success for Branch1

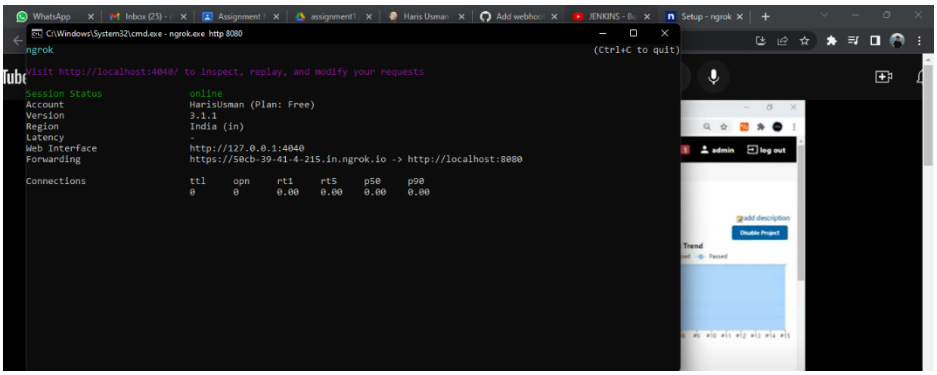



# Build success for Branch2

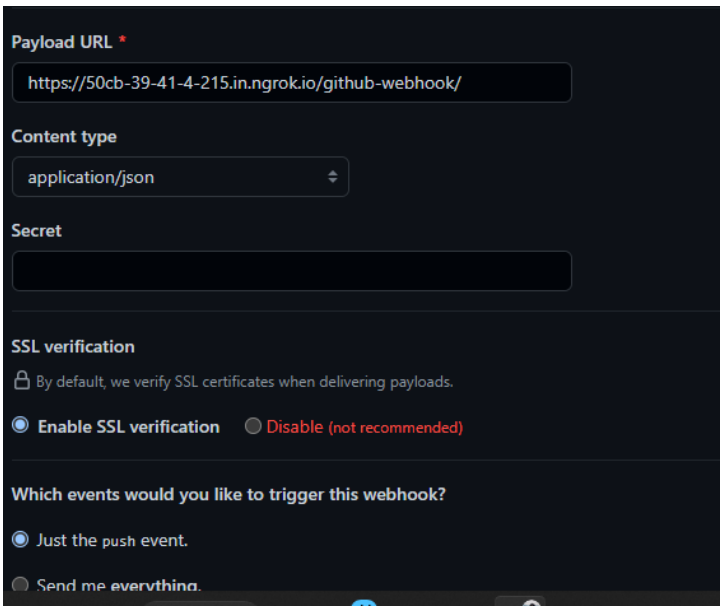



# Deploying on Ngrok

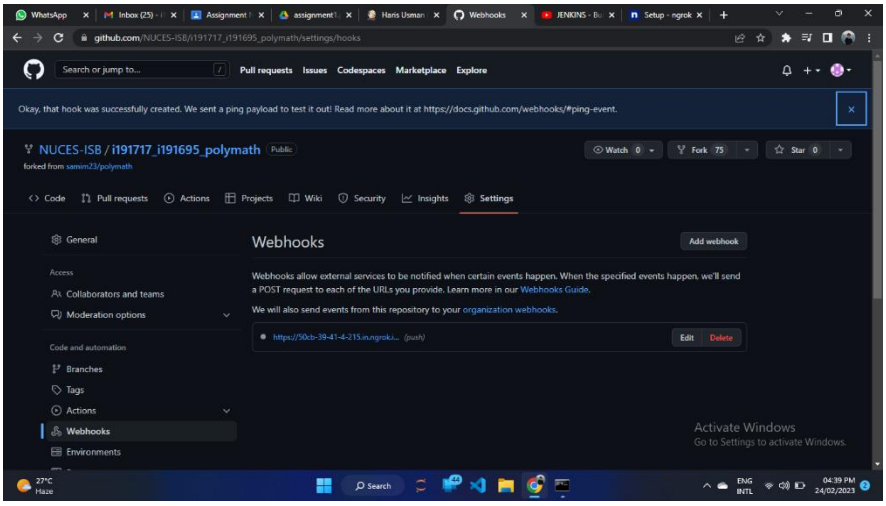
For Deployment on ngrok we configured authtoken that was successful. Then we ran ngrok server which was successful.



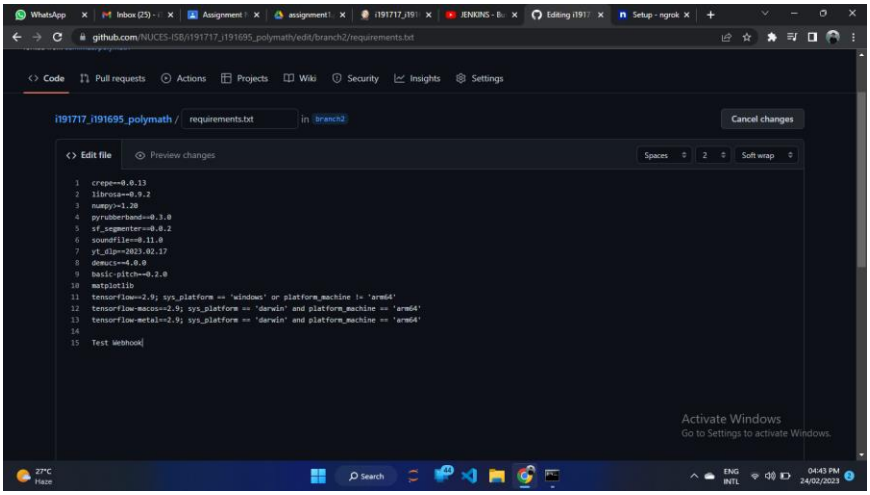

For successful implementation of automatic build, we created a webhook in GitHub settings with following configuration.



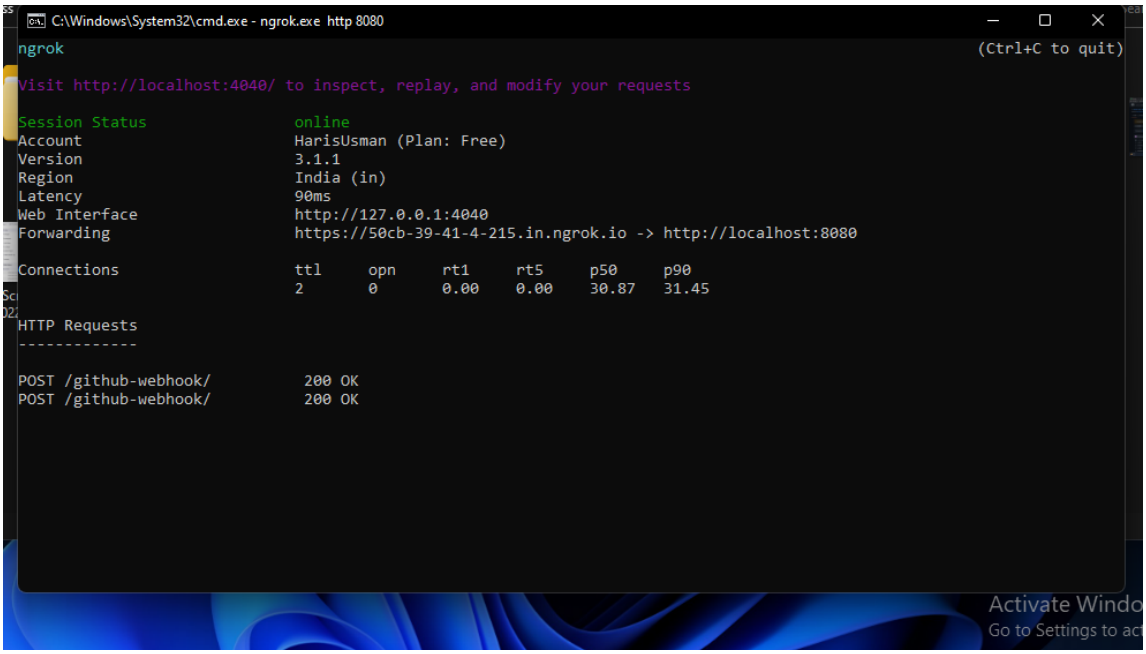

With above configuration webhook was created successfully as shown in following image.



To check if project would built automatically I edited a file and committed from branch2.




Then immediately response was generated from the server that is attached below



Then Jenkins job was executed automatically ones the change was committed. Build success image is attached below.