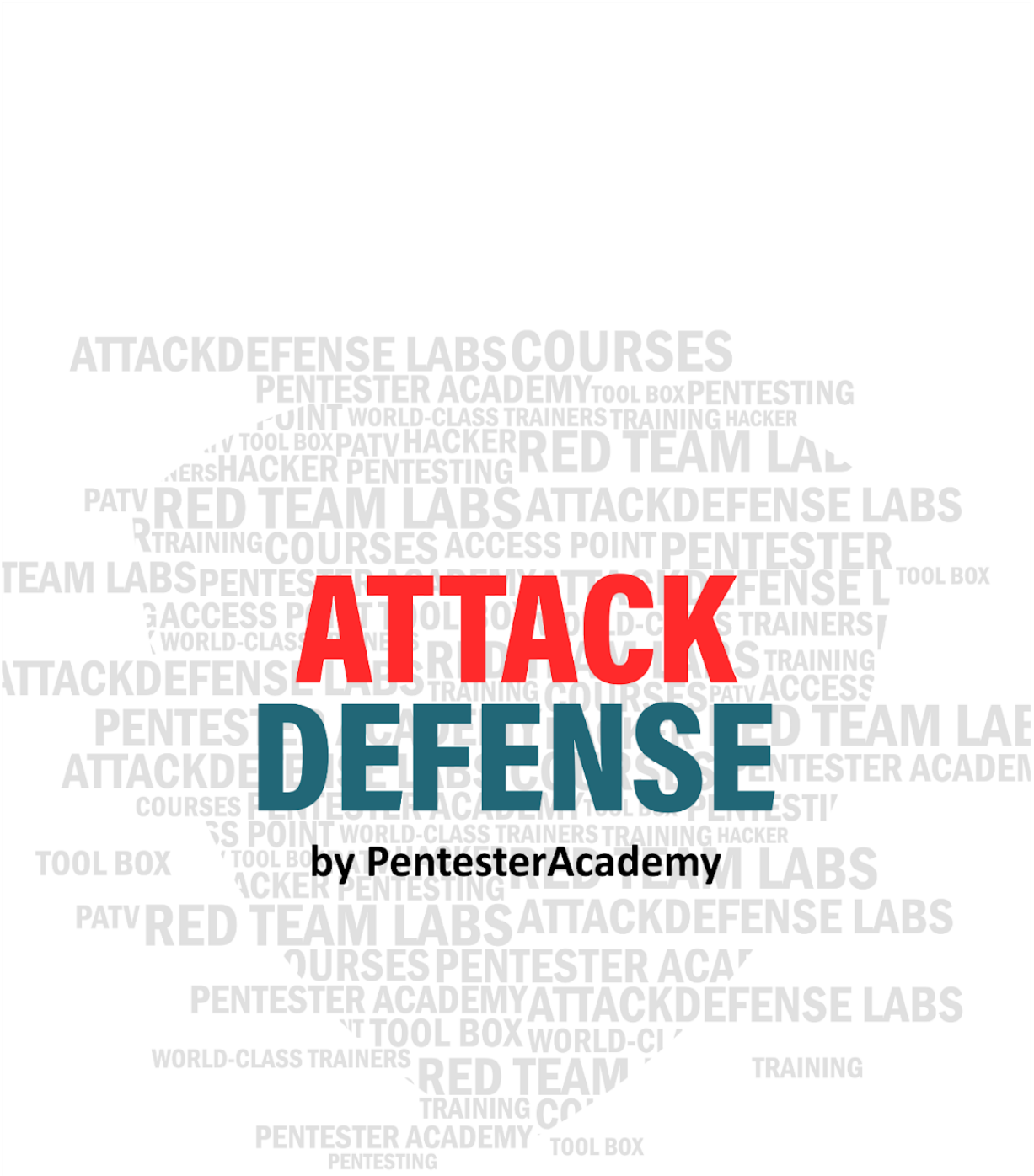

ATTACK
DEFENSE
by PentesterAcademy

Name	Fix the App: Django WebApp
URL	https://attackdefense.com/challengedetails?cid=2174
Type	Fix The Code: Web Applications

Important Note: This document illustrates all the important steps required to complete this lab. This is by no means a comprehensive step-by-step solution for this exercise. This is only provided as a reference to various commands needed to complete this exercise and for your further research on this topic. Also, note that the IP addresses and domain names might be different in your lab.

Challenge Description

DevOps practices are to combine software development (Dev) and IT operations (Ops) in order to improve the delivery process. DevOps pipelines are chained tasks and components that run in a sequence to cover different phases of software compilation, packaging, automated testing, and test deployment.

In this lab, we have a DevSecOps pipeline for a Django web application. The pipeline consists of the following components (and tasks):

- VS Code Server (For pulling, modifying, and pushing the code)
- GitLab server (For hosting code)
- Jenkins server (For integrating). Different phases and components used:
  - Build: Django
  - Code testing: Django
  - Test Deployment: Ansible
  - Dynamic Testing: Selenium
- Test server (For test deployment)
- Archery Sec server (For Vulnerability management)

It is suggested to play our DevOps focused labs before playing this lab.

DevSecOps refer to introducing security in different stages of the DevOps process. This is done to catch the vulnerabilities/insecurities as soon as possible in the pipeline. In this lab, the pipeline consists of the following components (and tasks):

- Automated Code Review: DevSkim
- Sensitive Information Scan phase: Truffle Hog
- Software Component Analysis: Safety
- Static Code Analysis: Bandit
- Dynamic Application Security Testing: OWASP ZAP
- Compliance as Code: Inspec

Objective: Fix the Issues in the stages of the pipeline and Find the flags!

Instructions:

- The GitLab server is reachable with the name 'gitlab'
- Gitlab credentials:

Username	Password
root	welcome123

- The Archery server is reachable by the name "archerysec"
- ArcherySec credentials:

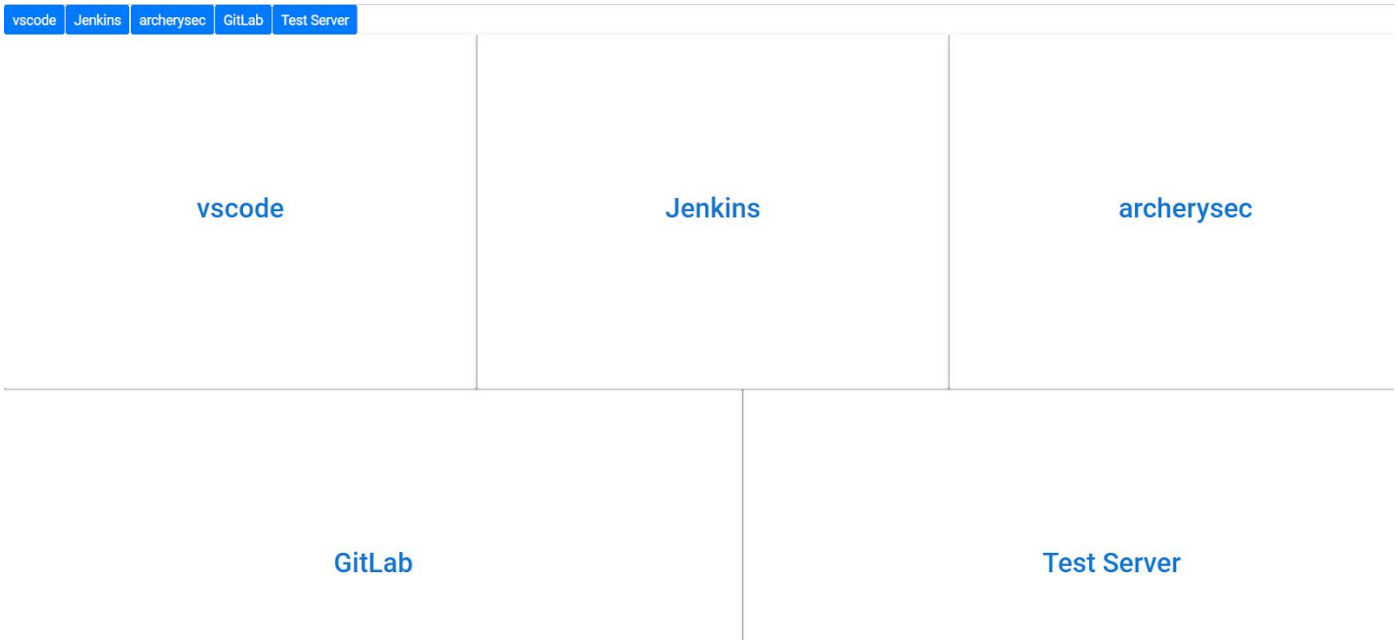
Username	Password
admin	admin

References:

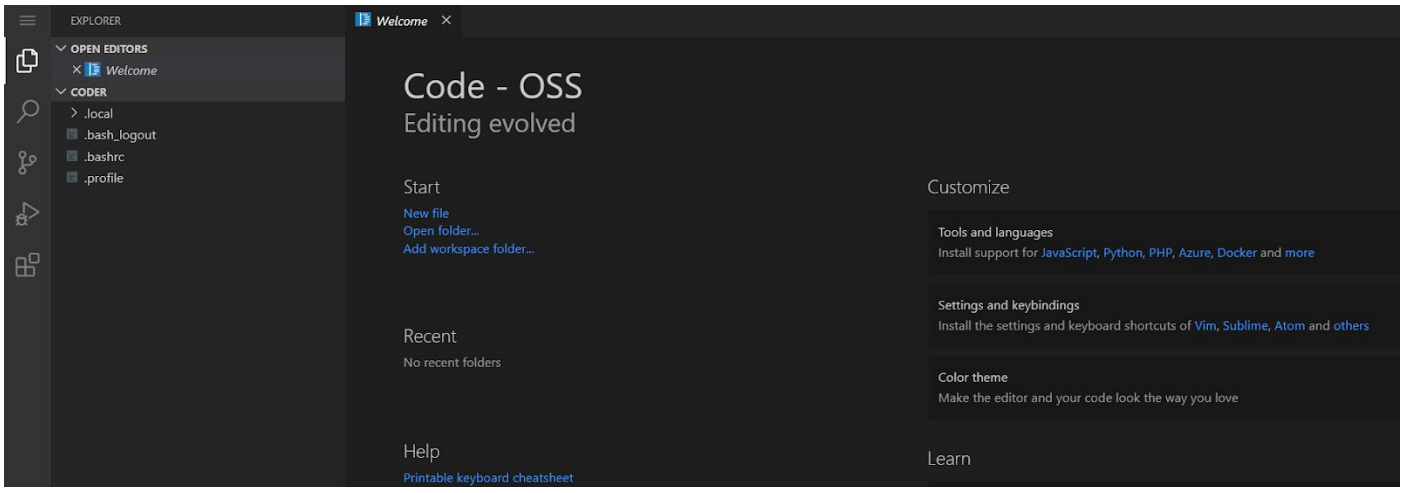
- Django-blog (https://github.com/AmirAhrari/django-blog)

Lab Setup

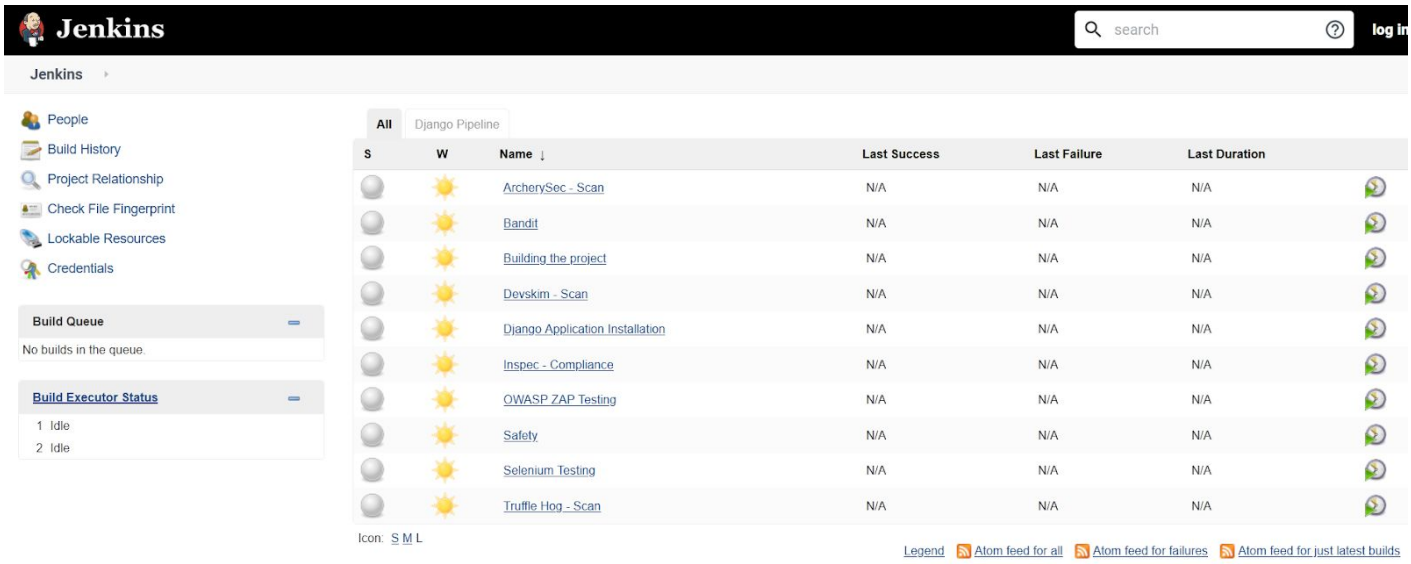
On starting the lab, the following interface will be accessible to the user.



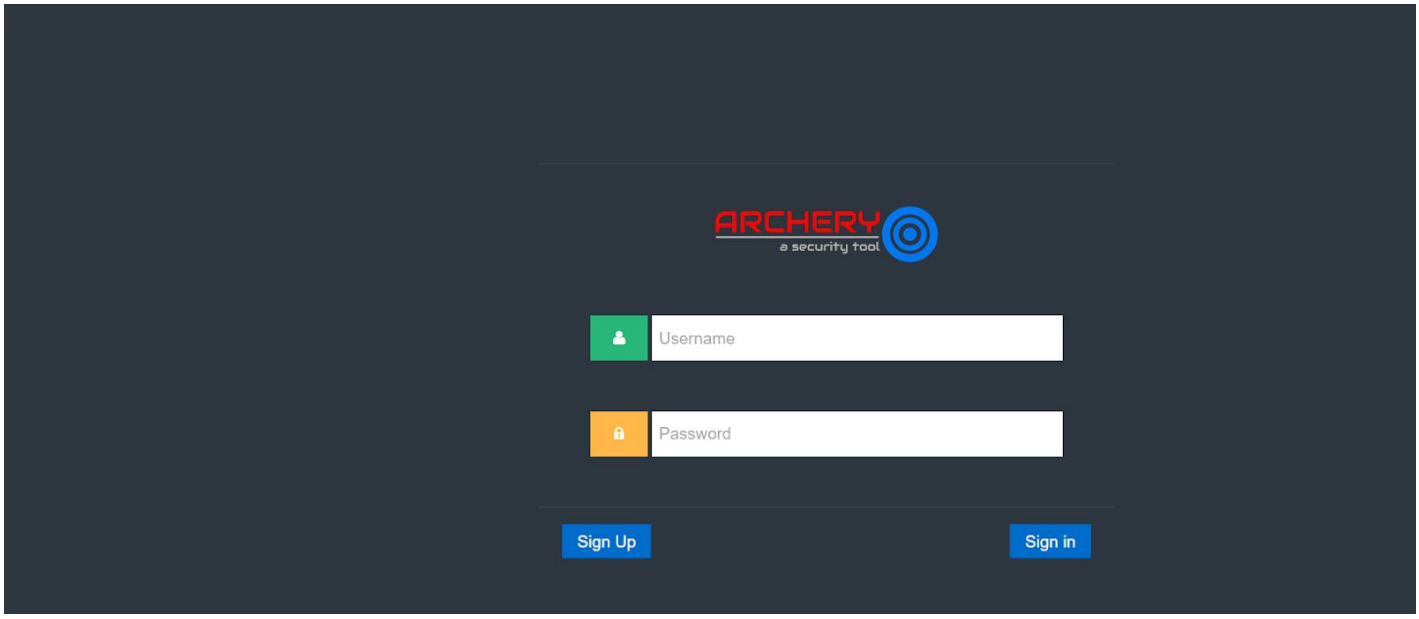
On choosing (clicking the text in the center) top left panel, **vscode** will open in a new tab



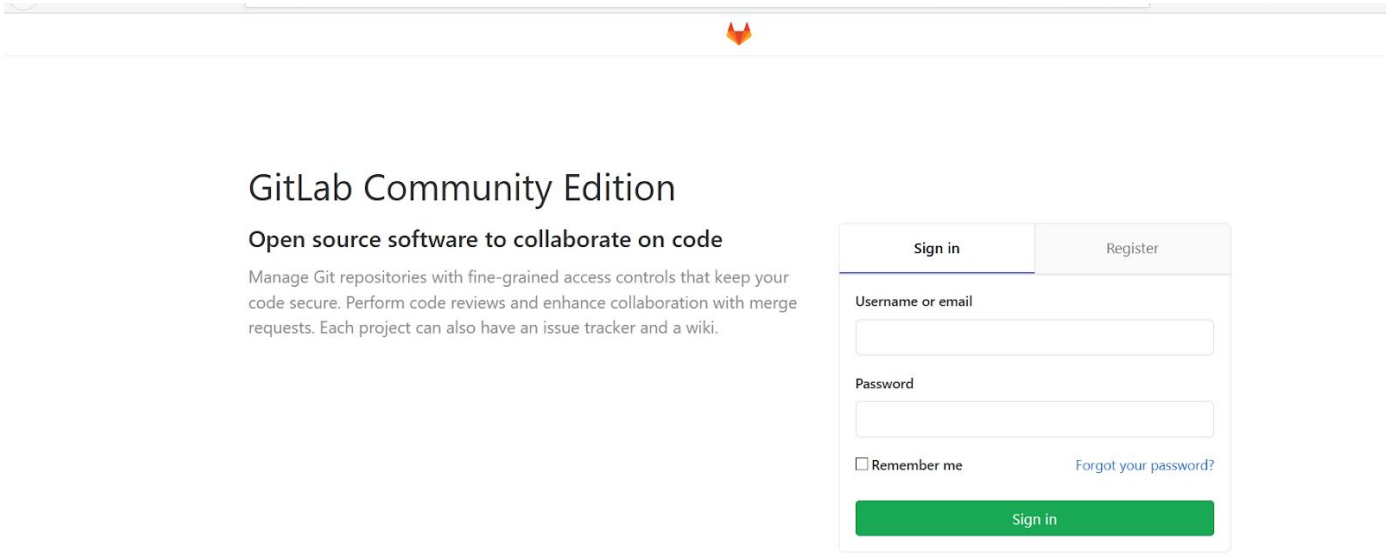
Similarly on selecting the top middle panel, a web UI of **Jenkins** will open in a new tab.



On selecting the top right panel, a web UI of **ArcherySec** will open in a new tab.



On selecting the bottom left panel, a web UI of **Gitlab** will open in a new tab.



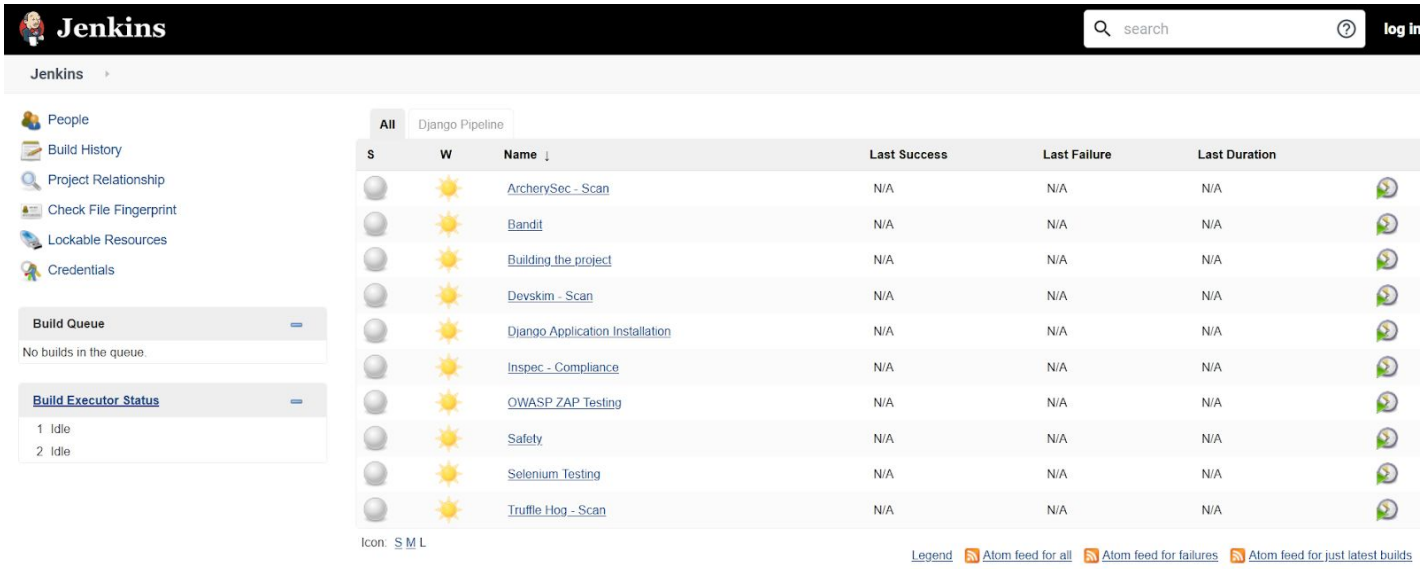
And on selecting the bottom right panel, a web UI of **Test Server** will open in a new tab.

```
Bad Gateway
```

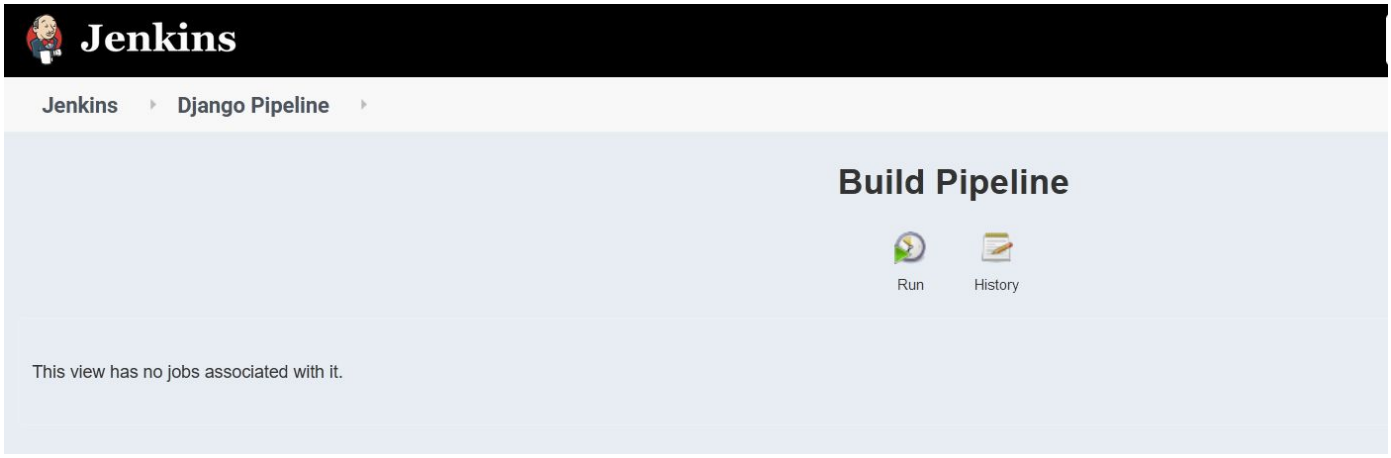
The page will reload until the test-server has started running the web service at port 8000

Solution

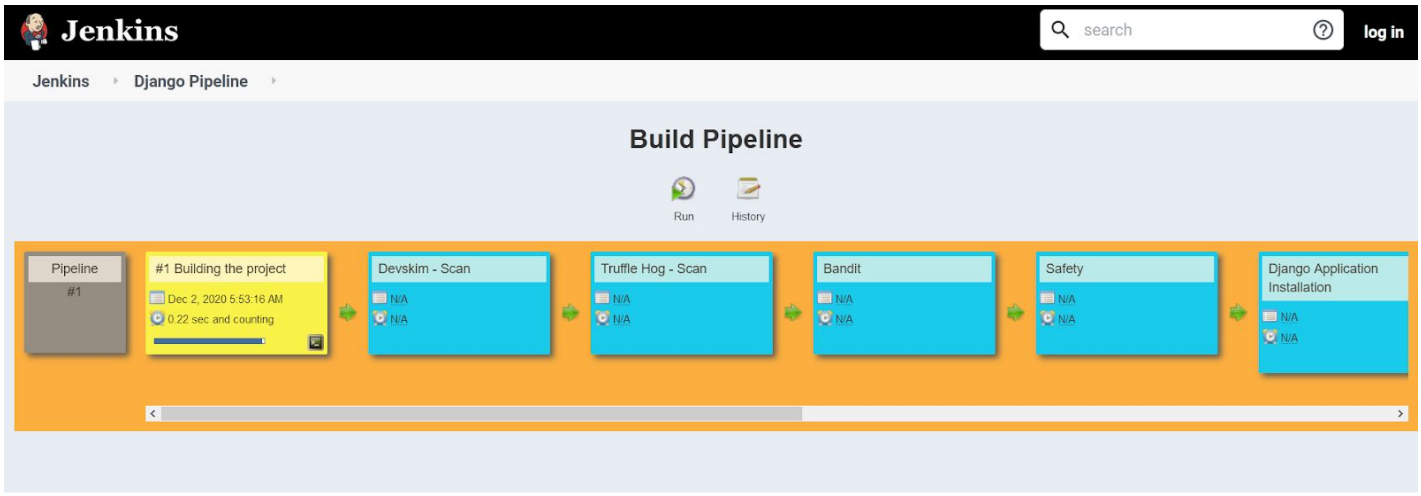
Step 1: Open the Jenkins page



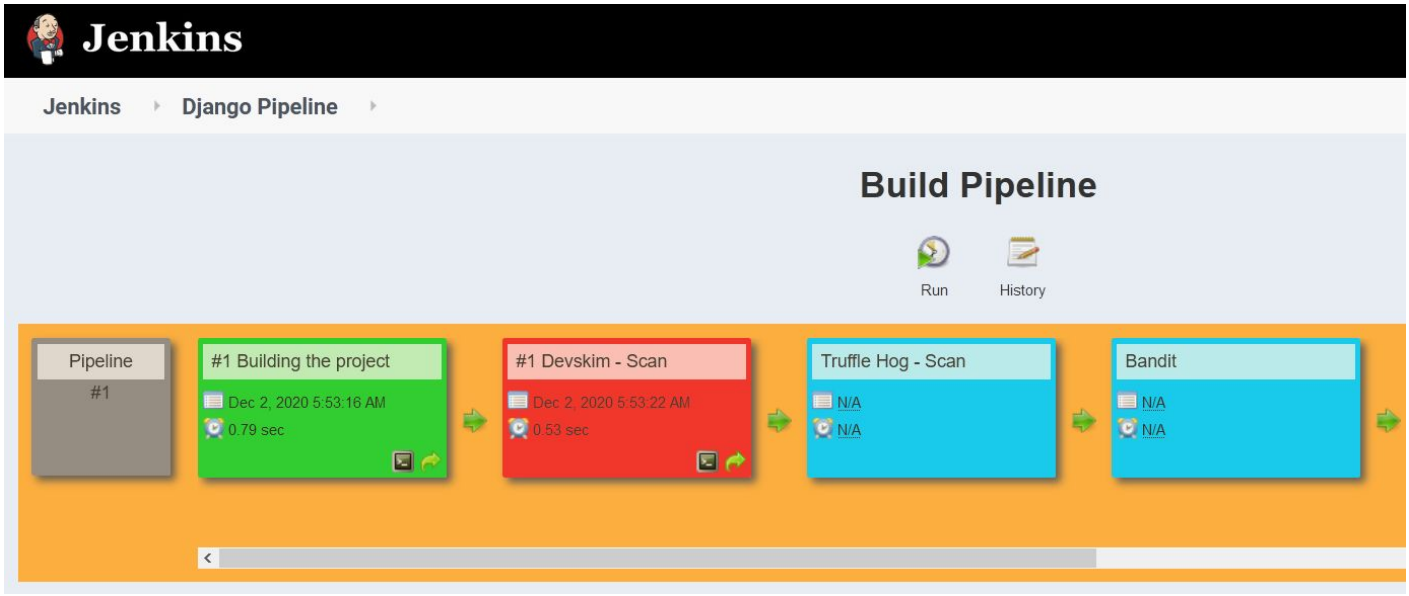
There are 10 jobs present in the Jenkins interface. Navigate to the Django Pipeline view section.



Click on the Run button to start building the pipeline.



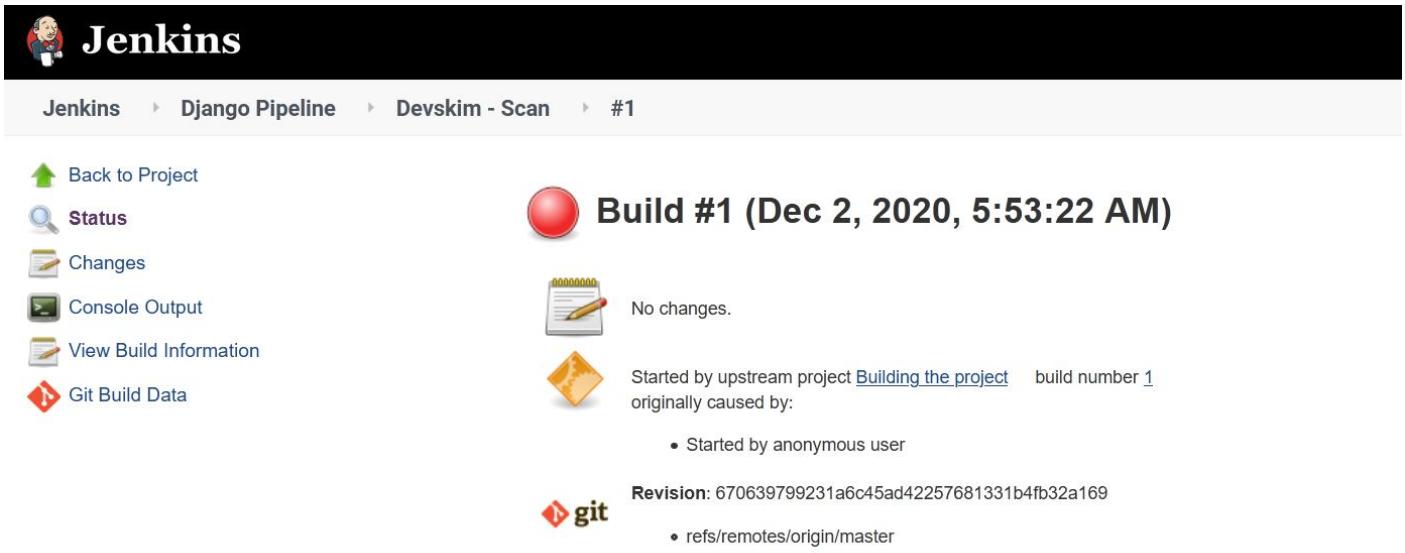
The projects will start building one by one. Keep reloading the page in intervals to check the changes on the page.



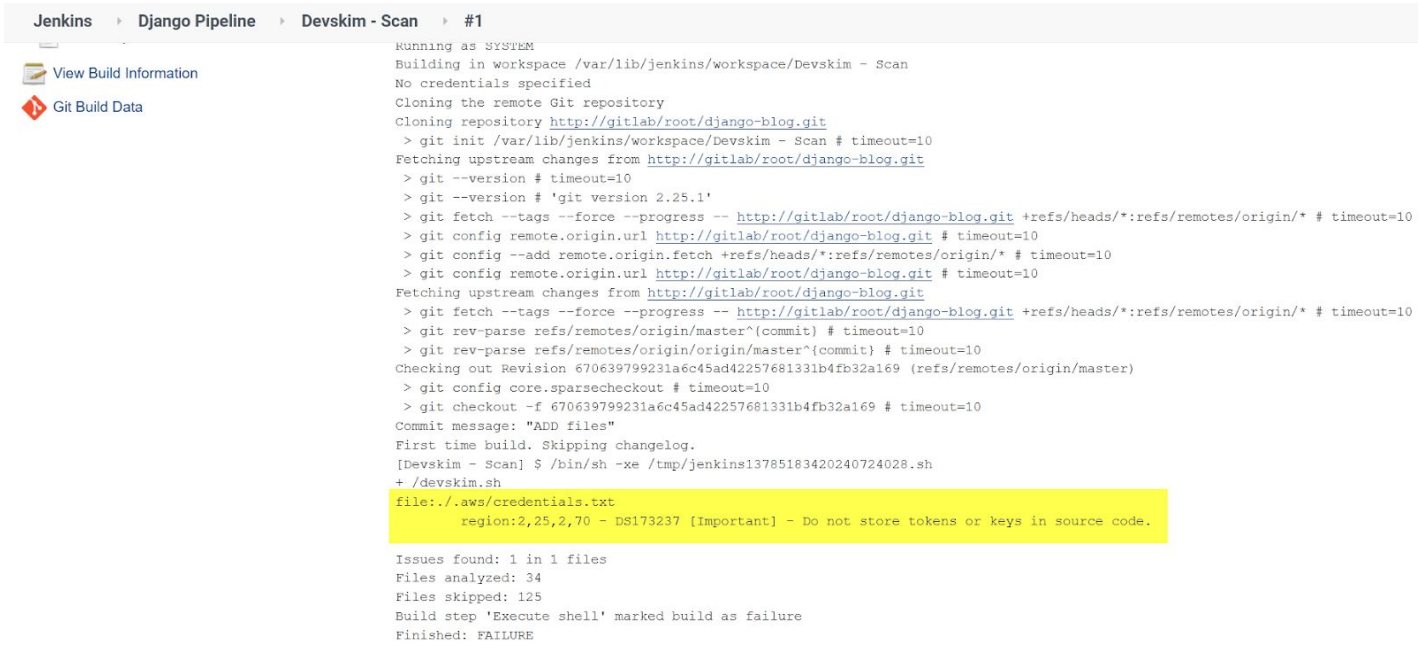
The build failed.

Devskim Issue

Step 1: Click on the 'Devskim - Scan' to check the job build page.

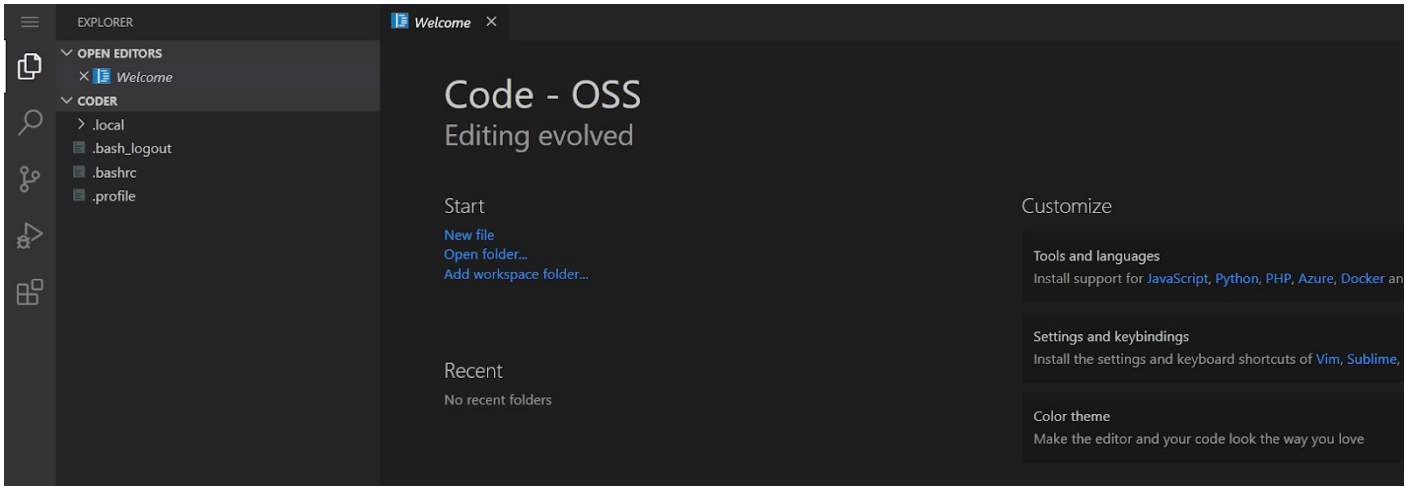


Step 2: Click on the “Console Output” to check the issues found by devskim tool.

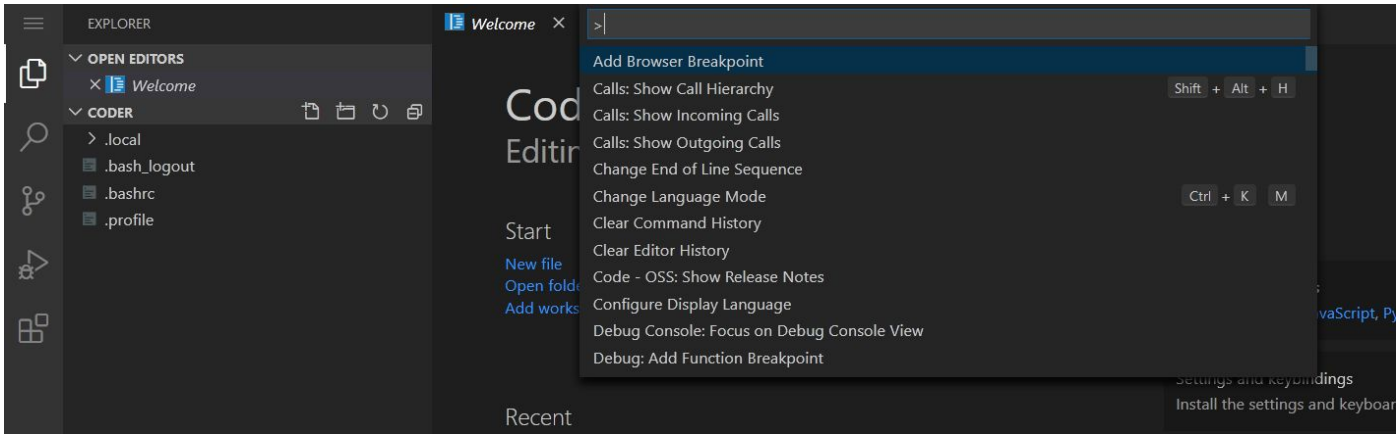


The devskim identified AWS credentials hardcoded in the .aws directory.

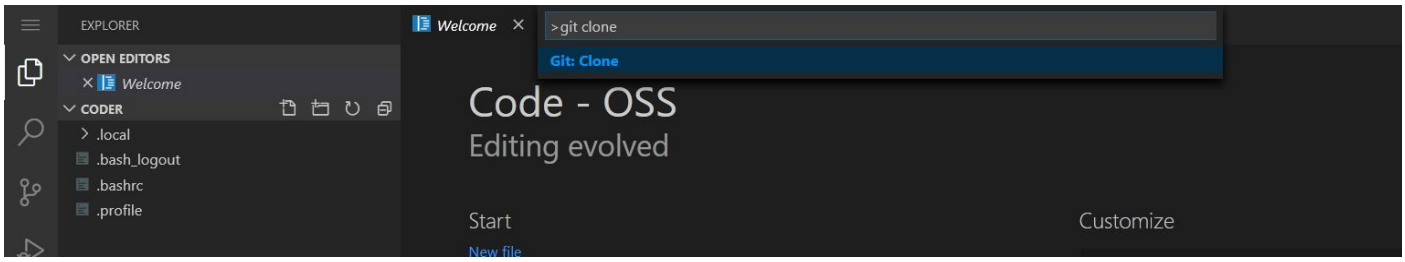
Step 3: Open the vscode server page.



Press CTRL + SHIFT + P to open the command palette or click on the settings bar available in the bottom left and choose the “Command Palette” option.



Step 4: Enter the command “git clone” in the command palette in order to clone the repository and make changes.



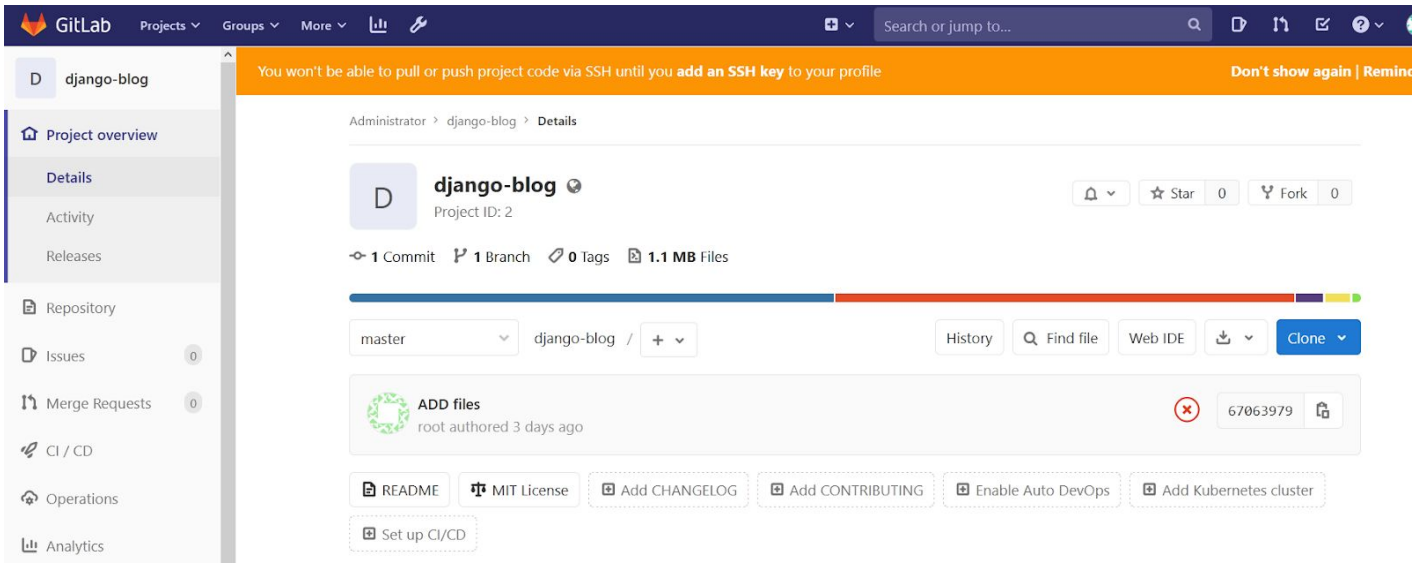
Press enter to choose the Git Clone option.

Step 5: Open the gitlab page and log in using the credentials provided in the challenge description.

Credentials:

- **Username:** root
- **Password:** welcome123

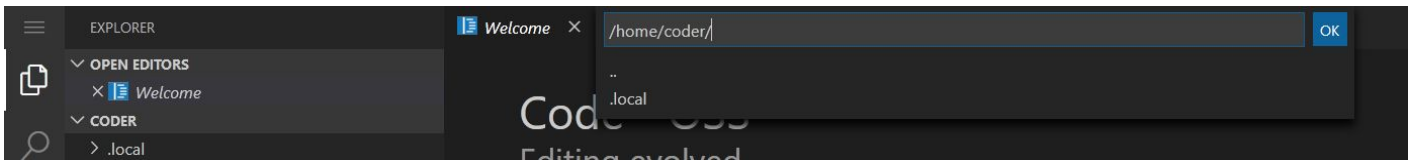
Click on the django-blog link to open the repository page.



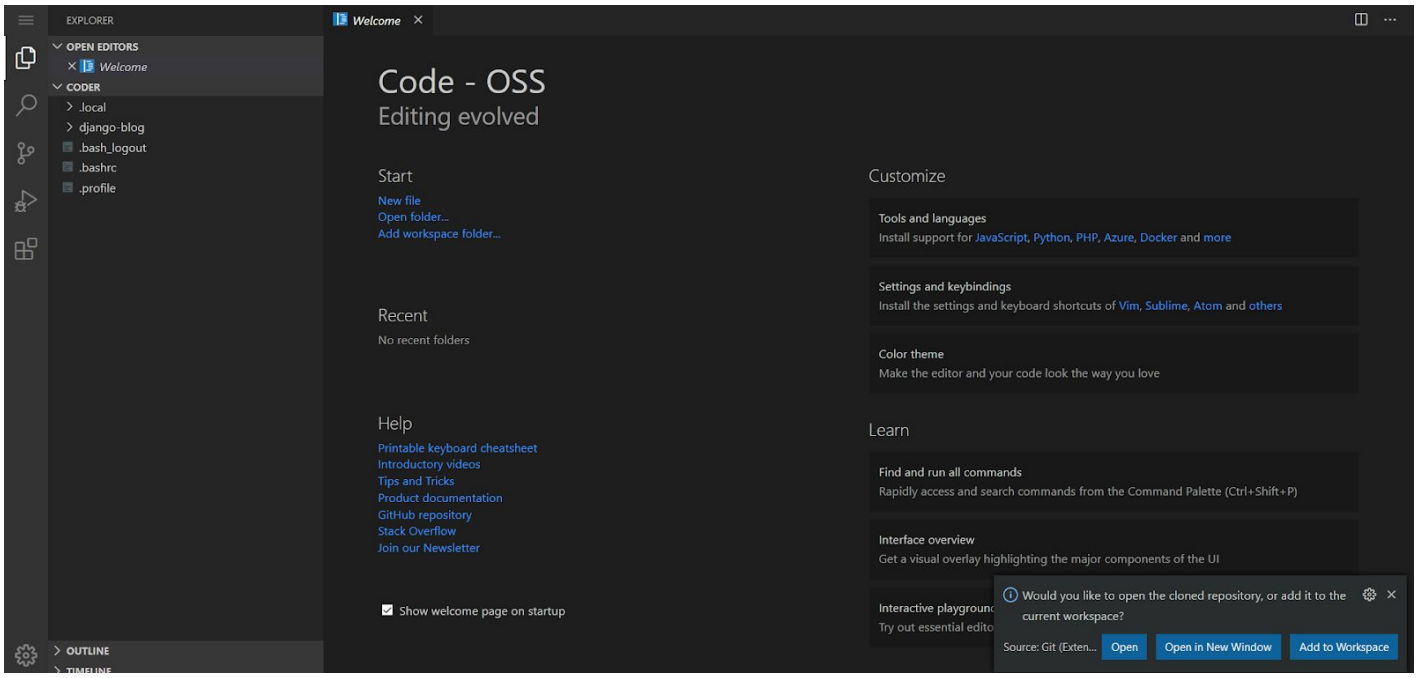
Copy the git link to the GitLab repository.

Step 6: Enter the link into the clone function at vscode server.

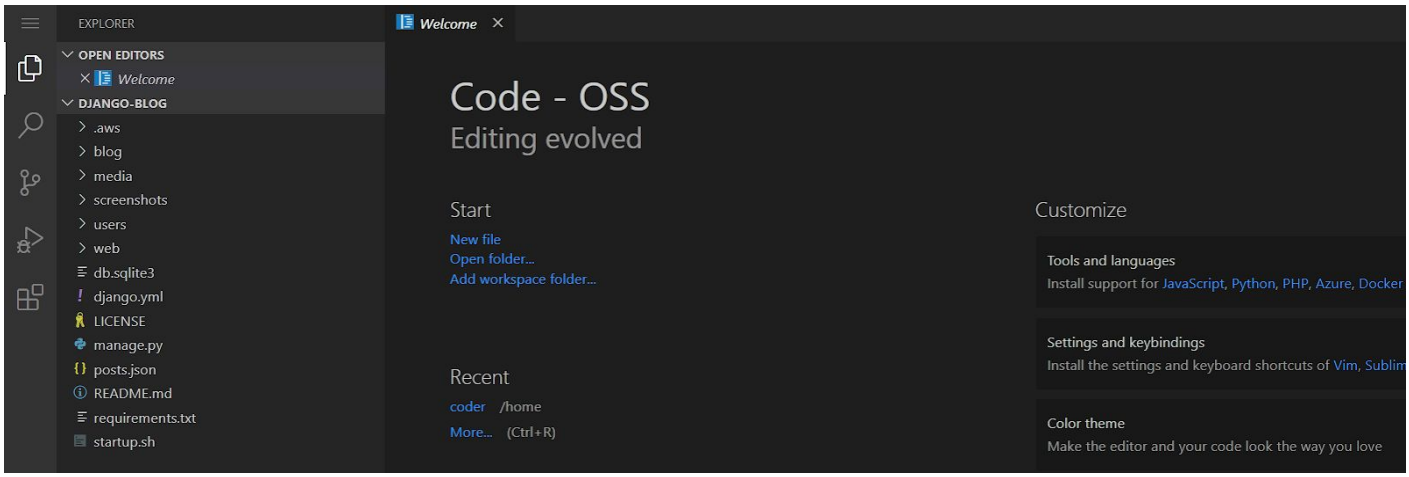
Press Enter.



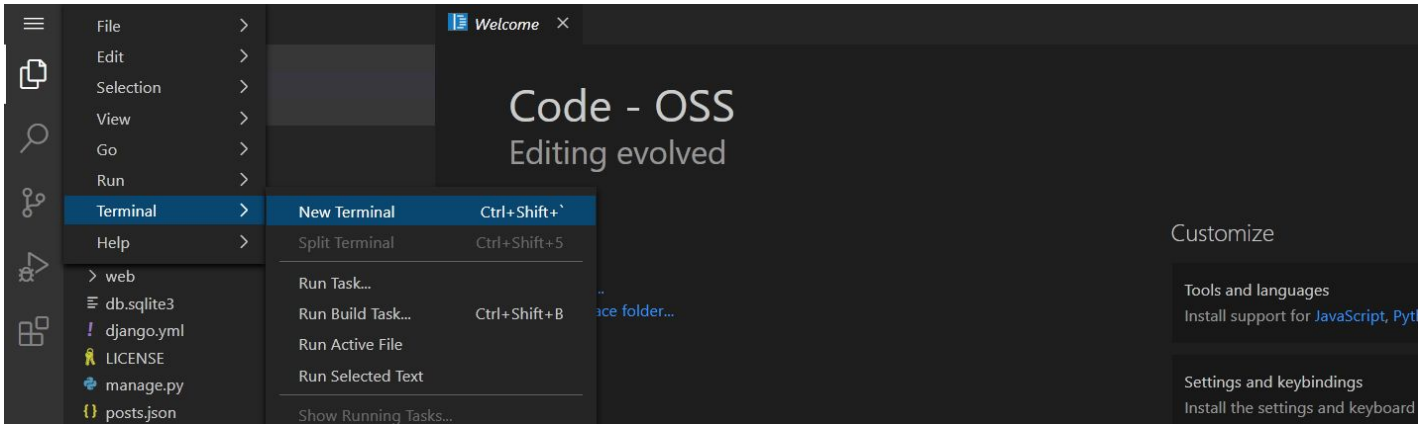
Choose the path to clone to the repository.



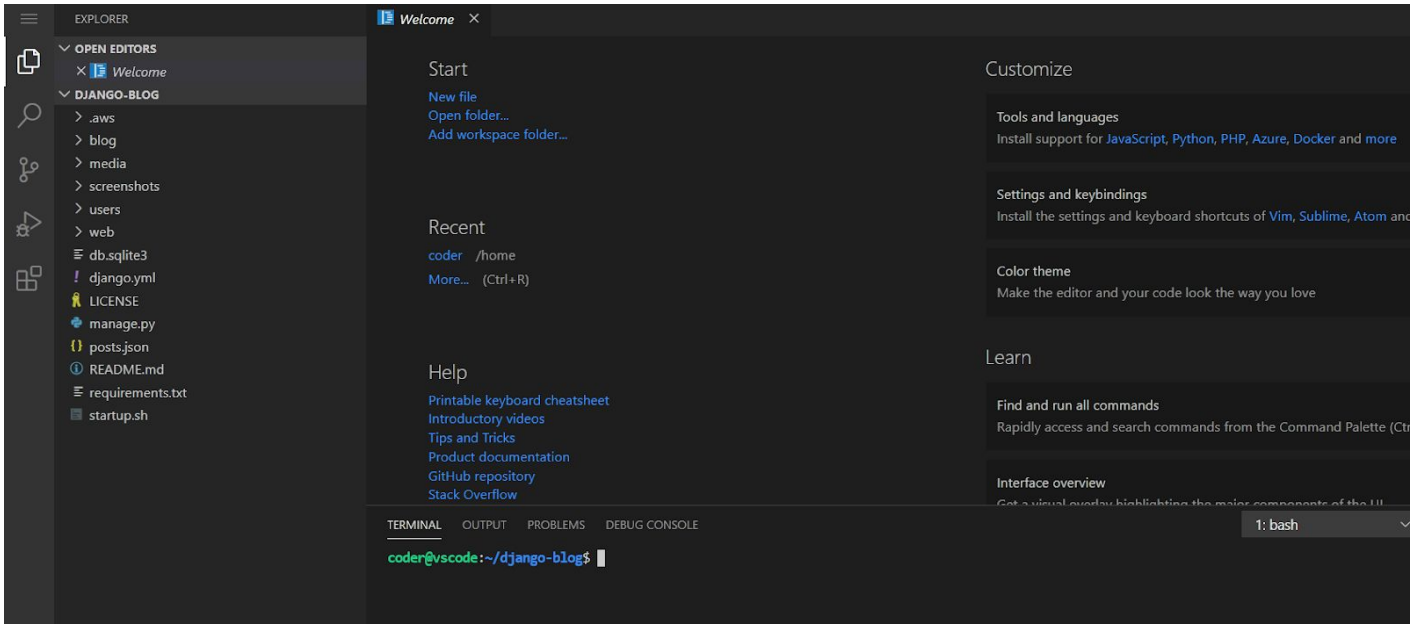
Click on the “Open” button to open the source code directory of the django-blog.



Step 7: Open the Terminal in the directory.



Select the “New Terminal” option.



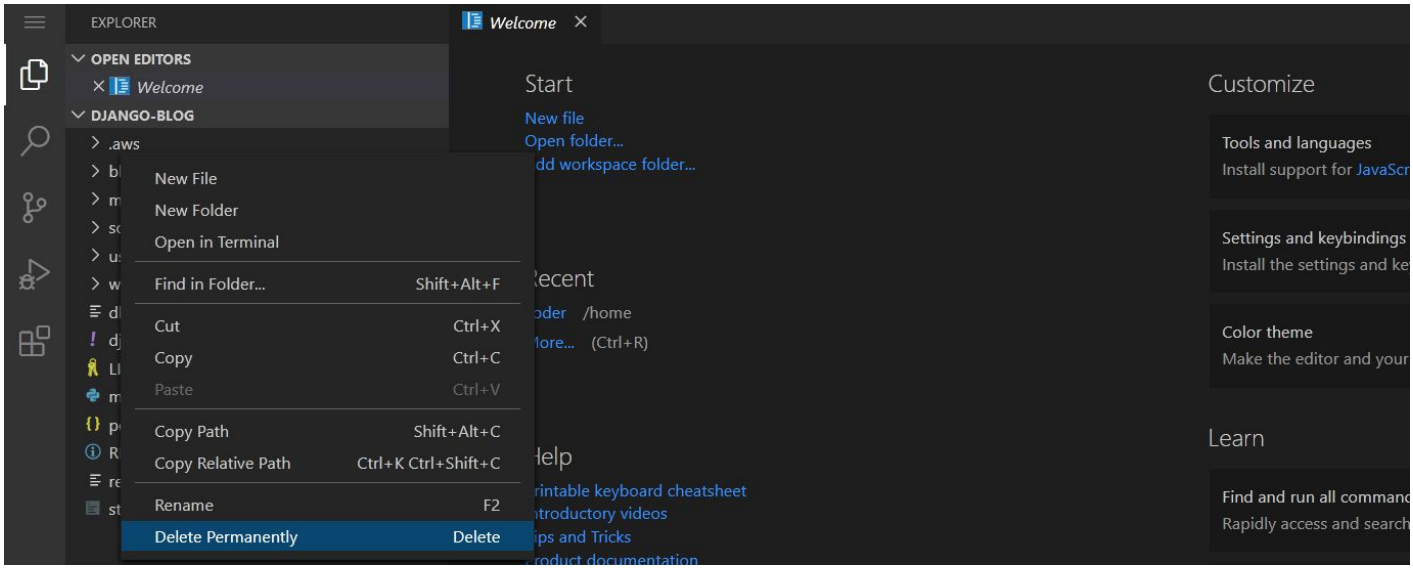
Step 8: Enter the git credentials into the terminal to save the identity of the user.

Commands:

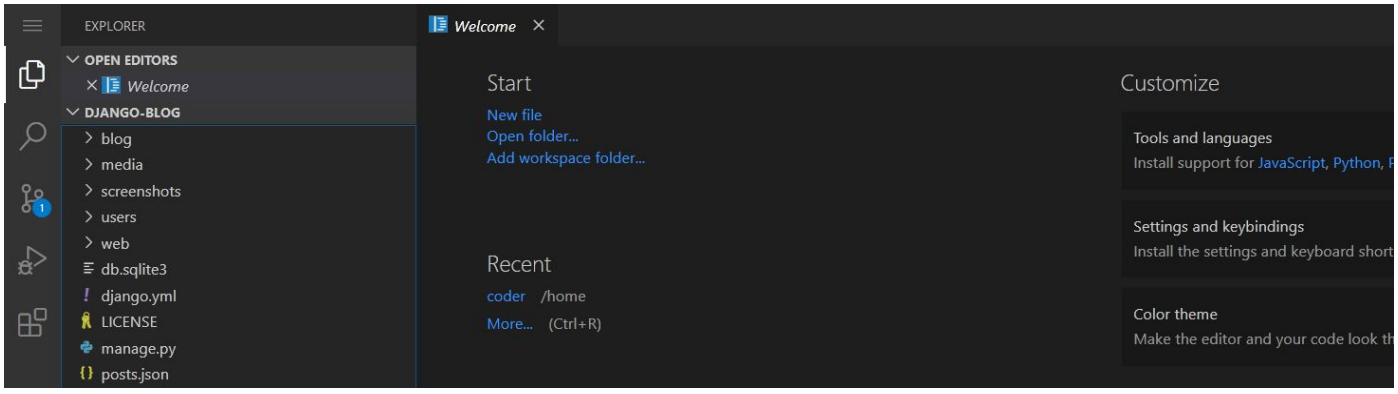
```
git config --global user.email "root@attacker.xyz"
git config --global user.name "root"
```

```
coder@vscode:~/django-blog$
coder@vscode:~/django-blog$ git config --global user.email "root@attacker.xyz"
coder@vscode:~/django-blog$ git config --global user.name "root"
```

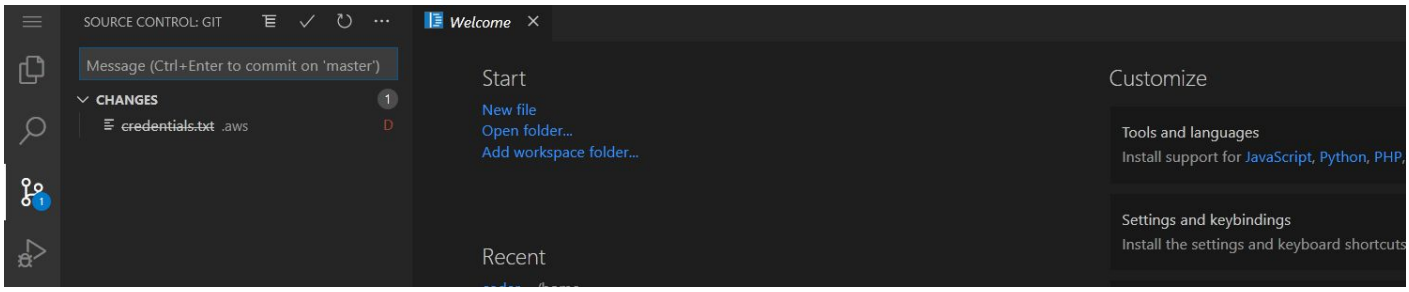
Step 9: Right-click on the AWS (.aws) directory.



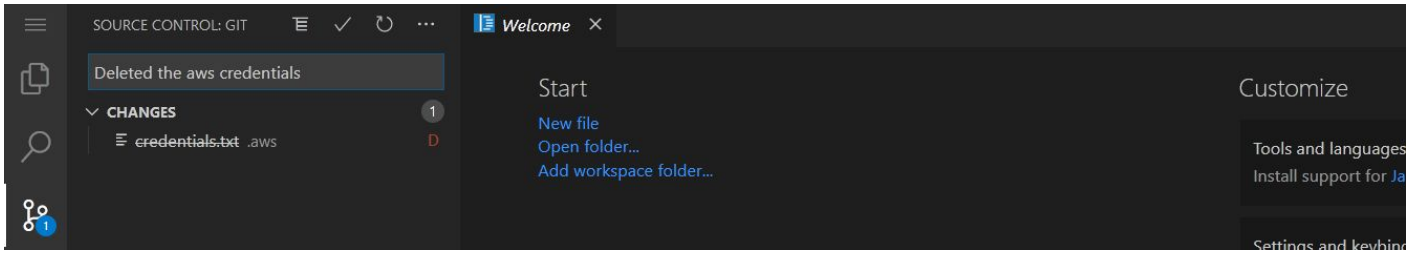
Select “Delete Permanently” to delete the AWS directory from the project.



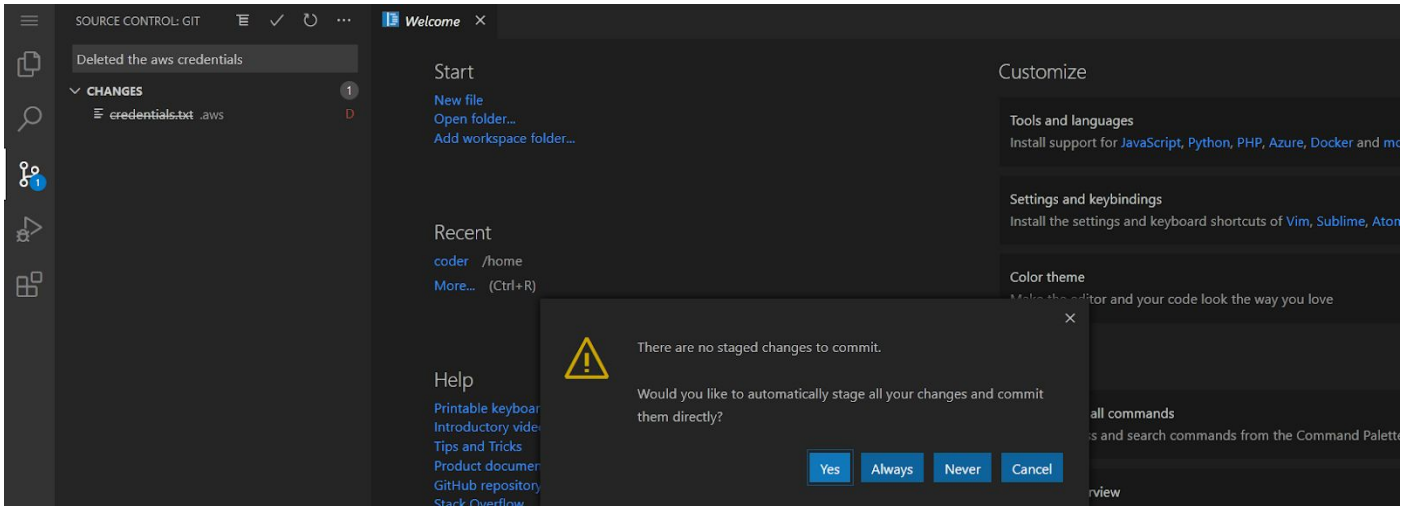
Step 10: Navigate to the Source Control section.



Enter a message in the message field to set a comment on the commit.

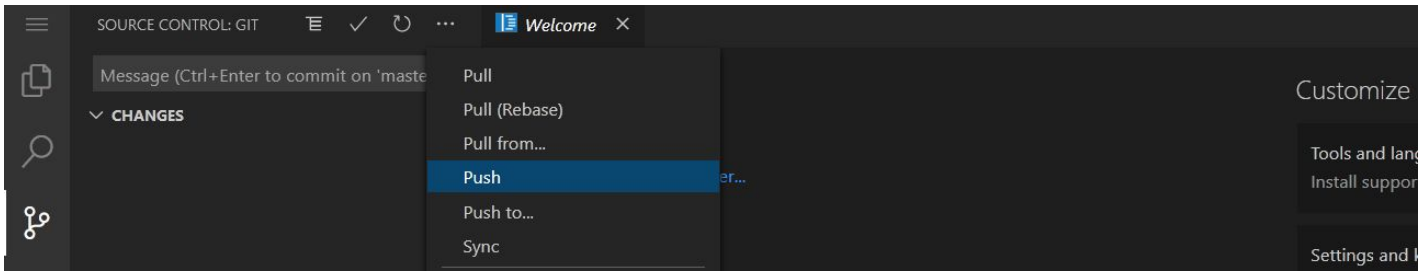


Click on the Commit button (Tick icon).



Choose the option “Always” to stage the commits always after the changes are made to the files.

Step 11: Click on the “More Actions” button (3 dots)

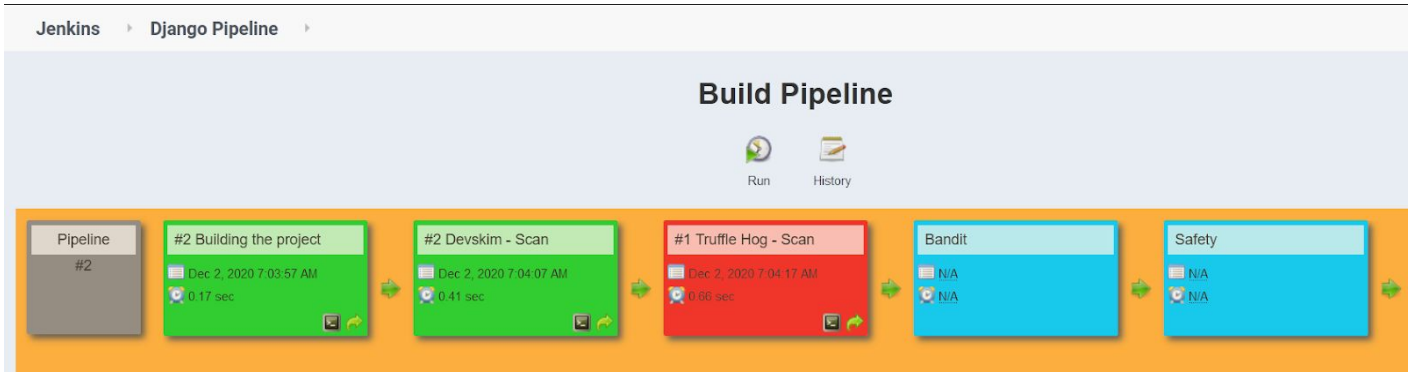


And select the Push button to push the changes to the remote repository.

Enter the git username and password when asked in the fields. The credentials are the same from gitlab.

Note: As soon as the code is pushed to the remote repository, the pipeline will start building automatically.

Step 12: Navigate back to the Pipeline view to check if Devskim stage passes or not.



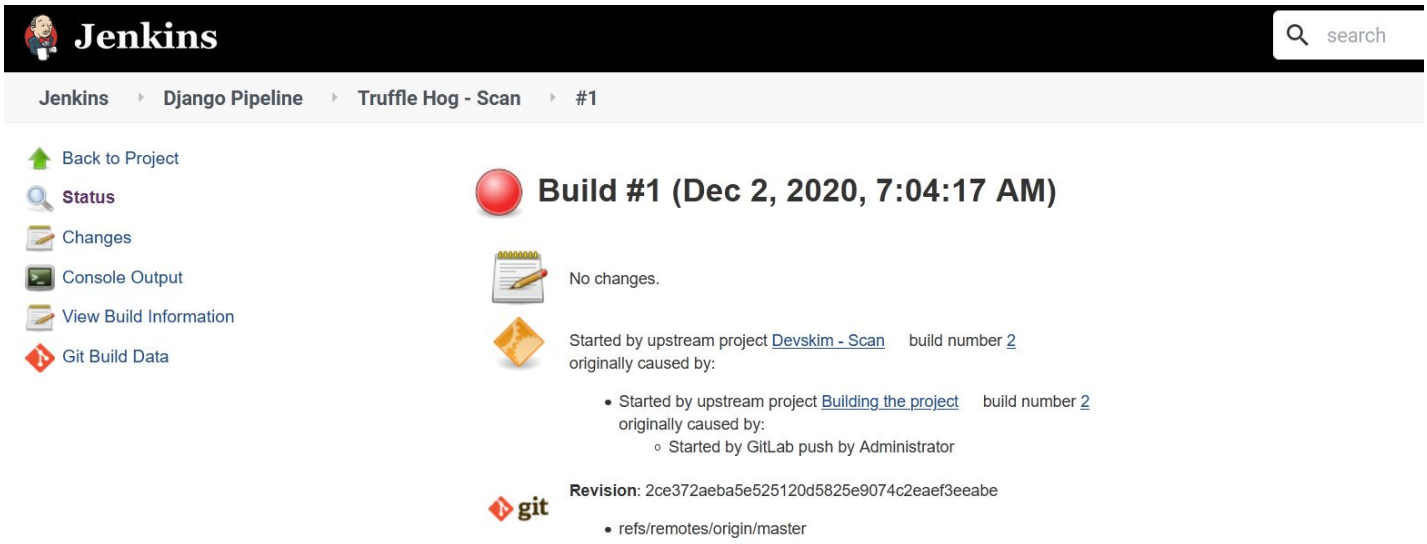
The Devskim stage has passed successfully. The logs can be checked from clicking on the "Devskim - Scan" and opening the console output.

```
 > git rev-list --no-walk 670639799231a6c45ad42257681331b4fb32a169 # timeout=10
[Devskim - Scan] $ /bin/sh -xe /tmp/jenkins15761689454421044862.sh
+ /devskim.sh
Issues found: 0 in 0 files
Files analyzed: 33
Files skipped: 127
FLAG 1: 8aa6cc853503696b5740a462b57aebb6
Triggering a new build of Truffle Hog - Scan
Finished: SUCCESS
```

FLAG 1: 8aa6cc853503696b5740a462b57aebb6

TruffleHog Issue

Step 1: Click on the 'TruffleHog- Scan' to check the job build page.



Step 2: Click on the "Console Output" to check the issues found by TruffleHog tool.

```
+ /trufflehog.sh
Cloning into 'django-blog'...
{
  "branch": "origin/master",
  "commit": "deleted aws creds\n",
  "commitHash": "4ce20a8b7da8d3eae7c9b13aafd6f29c040a1f7b",
  "date": "2020-12-15 13:37:52",
  "diff": "@@ -0,0 +1,3 @@\n+aws_access_key_id = 123456789012 \n+aws_secret_access_key = 'febcc175b9c0f1b6a831c399eadd2912697726617de'
\n+aws_session_token = 'AQoEXAMPLEH4aoAH0gNCAPyJxz4BlCFFxWNE1OPTgk5TthT+FvwqnKwRcOIfrRh3c/LTo6UDdyJwOOvEVPvLXCrrrUtdnniCEXAMPLE
/IvU1dYUg2RVAJBanLiHb4IgRmpRV3zrkuWJOgQs8IZZaIv2BXIa2R4Olgk'\n",
  "path": ".aws/credentials.txt",
  "printDiff": "@@ -0,0 +1,3 @@\n+aws_access_key_id = 123456789012 \n+aws_secret_access_key =
'\u001b[93mfebcc175b9c0f1b6a831c399eadd2912697726617de\u001b[0m' \n+aws_session_token =
'\u001b[93mAQoEXAMPLEH4aoAH0gNCAPyJxz4BlCFFxWNE1OPTgk5TthT+FvwqnKwRcOIfrRh3c/LTo6UDdyJwOOvEVPvLXCrrrUtdnniCEXAMPLE
/IvU1dYUg2RVAJBanLiHb4IgRmpRV3zrkuWJOgQs8IZZaIv2BXIa2R4Olgk\u001b[0m'\n",
  "reason": "High Entropy",
  "stringsFound": [
    "febcc175b9c0f1b6a831c399eadd2912697726617de",
    "AQoEXAMPLEH4aoAH0gNCAPyJxz4BlCFFxWNE1OPTgk5TthT+FvwqnKwRcOIfrRh3c/LTo6UDdyJwOOvEVPvLXCrrrUtdnniCEXAMPLE
/IvU1dYUg2RVAJBanLiHb4IgRmpRV3zrkuWJOgQs8IZZaIv2BXIa2R4Olgk"
  ]
}
{
  "branch": "origin/master",
  "commit": "ADD files\n",
  "commitHash": "2c9e88534ef80eae1369c79a505daa03c0e82592",
  "date": "2020-12-15 08:13:39",
  "diff": "@@ -0,0 +1,3 @@\n+aws_access_key_id = 123456789012 \n+aws_secret_access_key = 'febcc175b9c0f1b6a831c399eadd2912697726617de'
\n+aws_session_token = 'AQoEXAMPLEH4aoAH0gNCAPyJxz4BlCFFxWNE1OPTgk5TthT+FvwqnKwRcOIfrRh3c/LTo6UDdyJwOOvEVPvLXCrrrUtdnniCEXAMPLE
/IvU1dYUg2RVAJBanLiHb4IgRmpRV3zrkuWJOgQs8IZZaIv2BXIa2R4Olgk'\n",
  "path": ".aws/credentials.txt",
  "printDiff": "@@ -0,0 +1,3 @@\n+aws_access_key_id = 123456789012 \n+aws_secret_access_key =
'\u001b[93mfebcc175b9c0f1b6a831c399eadd2912697726617de\u001b[0m' \n+aws_session_token =
'\u001b[93mAQoEXAMPLEH4aoAH0gNCAPyJxz4BlCFFxWNE1OPTgk5TthT+FvwqnKwRcOIfrRh3c/LTo6UDdyJwOOvEVPvLXCrrrUtdnniCEXAMPLE
/IvU1dYUg2RVAJBanLiHb4IgRmpRV3zrkuWJOgQs8IZZaIv2BXIa2R4Olgk\u001b[0m'\n",
  "reason": "High Entropy",
  "stringsFound": [
    "febcc175b9c0f1b6a831c399eadd2912697726617de",
    "AQoEXAMPLEH4aoAH0gNCAPyJxz4BlCFFxWNE1OPTgk5TthT+FvwqnKwRcOIfrRh3c/LTo6UDdyJwOOvEVPvLXCrrrUtdnniCEXAMPLE
/IvU1dYUg2RVAJBanLiHb4IgRmpRV3zrkuWJOgQs8IZZaIv2BXIa2R4Olgk"
  ]
}
Build step 'Execute shell' marked build as failure
Finished: FAILURE
```

The trufflehog identified AWS credentials in the commit history of the repository.

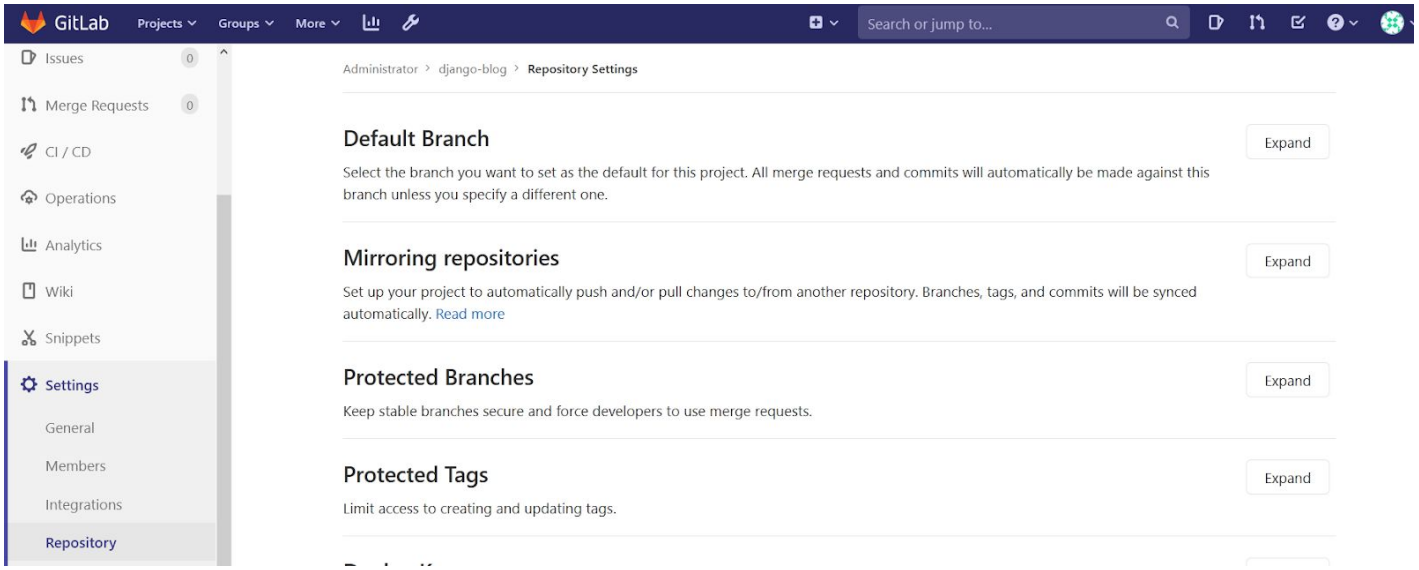
Step 3: Remove the AWS credentials file from the commit history. (Switch to VSCode terminal and execute the command)

Command: git filter-branch --force --index-filter \
"git rm --cached --ignore-unmatch .aws/credentials.txt" \
--prune-empty --tag-name-filter cat -- --all

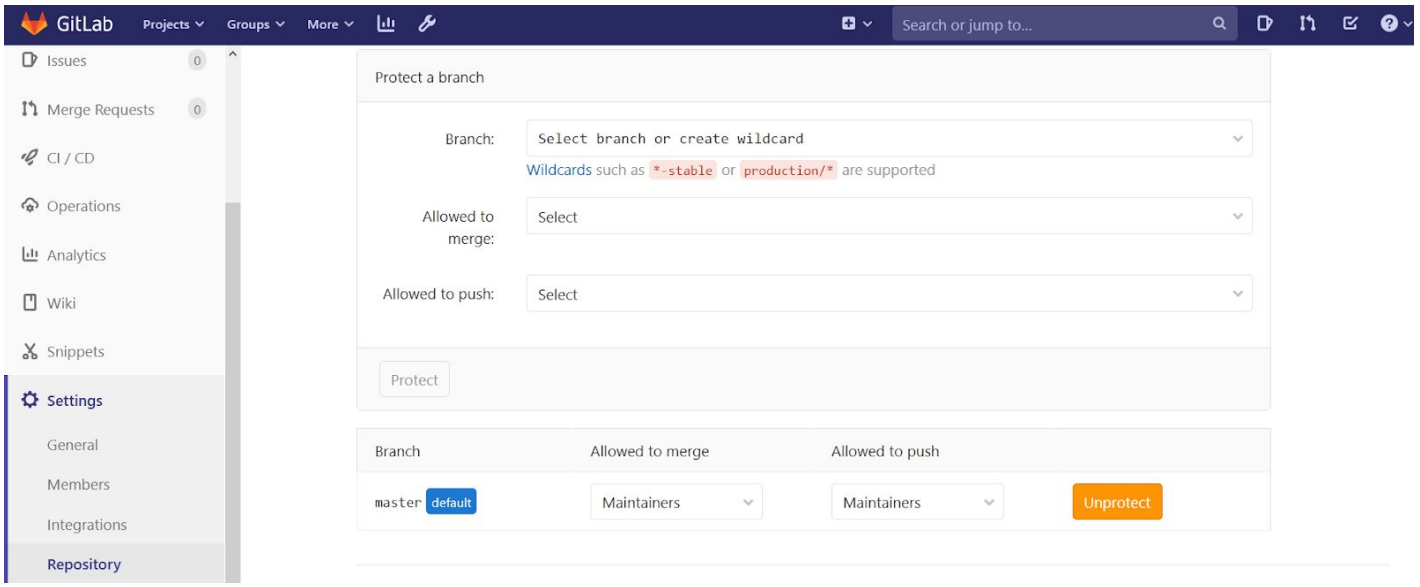
The command will remove all the commits having the file credentials.txt and filter-branch will create commits in order to fix up the commits history.

```
coder@vscode:~/django-blog$ git filter-branch --force --index-filter \
>   "git rm --cached --ignore-unmatch .aws/credentials.txt" \
>   --prune-empty --tag-name-filter cat -- --all
Rewrite 670639799231a6c45ad42257681331b4fb32a169 (1/2) (0 seconds passed, remaining 0 predicted)    rm '.aws/credentials.txt'
Rewrite 2ce372aeba5e525120d5825e9074c2eaef3eeabe (2/2) (0 seconds passed, remaining 0 predicted)
Ref 'refs/heads/master' was rewritten
Ref 'refs/remotes/origin/master' was rewritten
WARNING: Ref 'refs/remotes/origin/master' is unchanged
coder@vscode:~/django-blog$
```

Step 4: Navigate to the gitlab website and open the repository settings (Repository Settings)

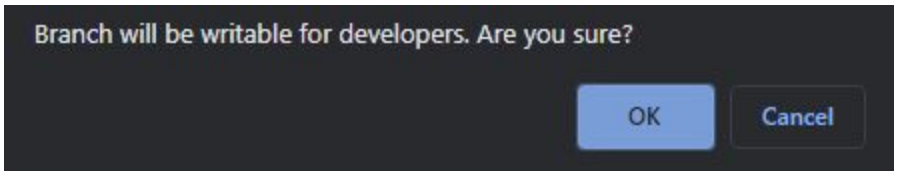


Expand the Protected Branches section.

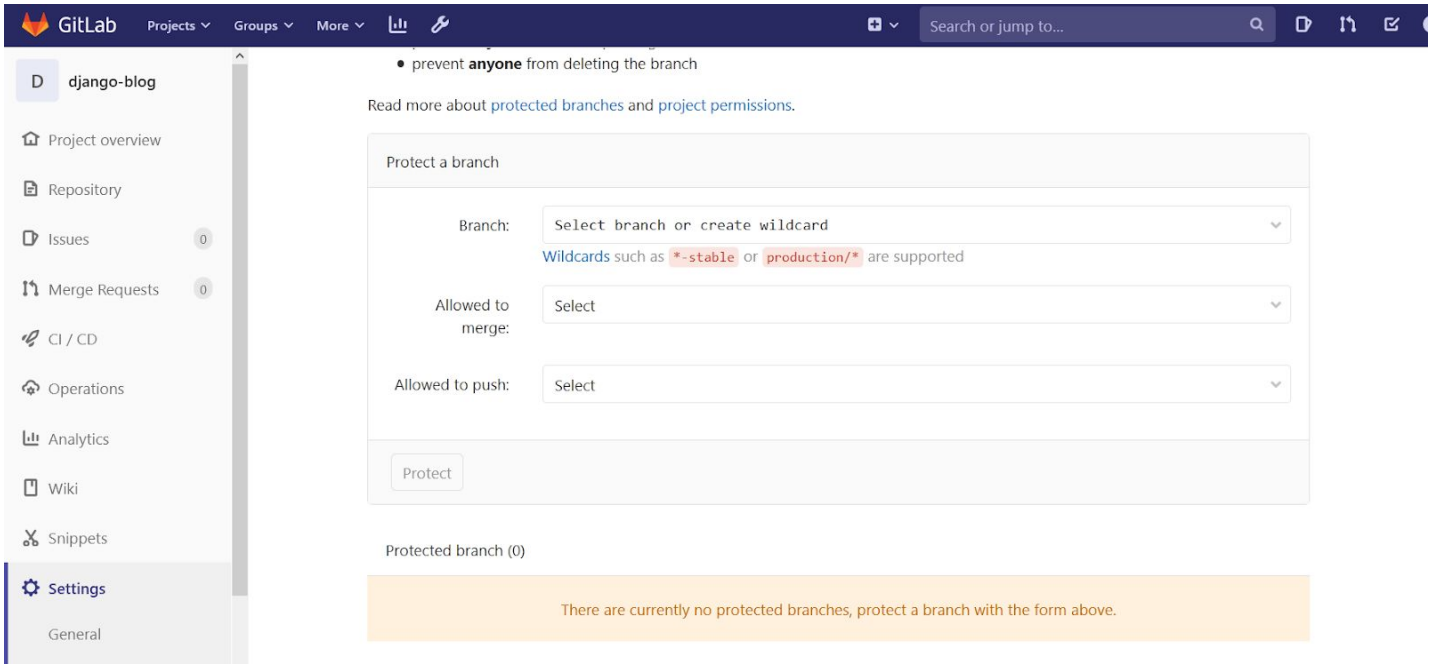


In order to push the changes which include altering the base commits (previous), we need to pass a force push to the remote repository and to do that the main branch needs to be set to “Unprotect” mode.

Step 5: Click on the Unprotect button.



Press ‘OK’ when prompted.



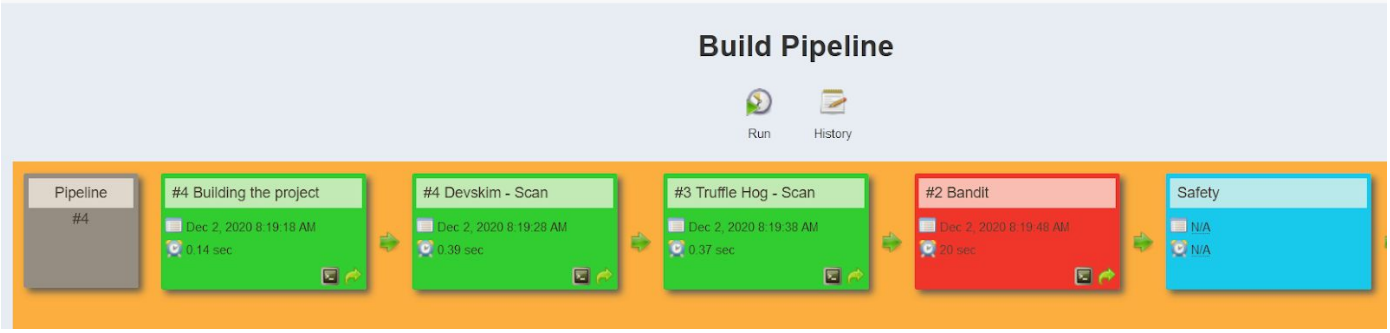
Step 6: Pass a force push to the remote repository. (From VSCode Terminal)

Command: git push --force origin

Use GitLab Credentials when prompted.

```
coder@vscode:~/django-blog$ git push --force origin
Username for 'http://gitlab': root
Password for 'http://root@gitlab':
Enumerating objects: 72, done.
Counting objects: 100% (72/72), done.
Delta compression using up to 48 threads
Compressing objects: 100% (66/66), done.
Writing objects: 100% (72/72), 456.20 KiB | 22.81 MiB/s, done.
Total 72 (delta 8), reused 0 (delta 0)
To http://gitlab/root/django-blog.git
 + 2ce372a...0667a31 master -> master (forced update)
coder@vscode:~/django-blog$
```

Step 7: Check the pipeline for the changes.



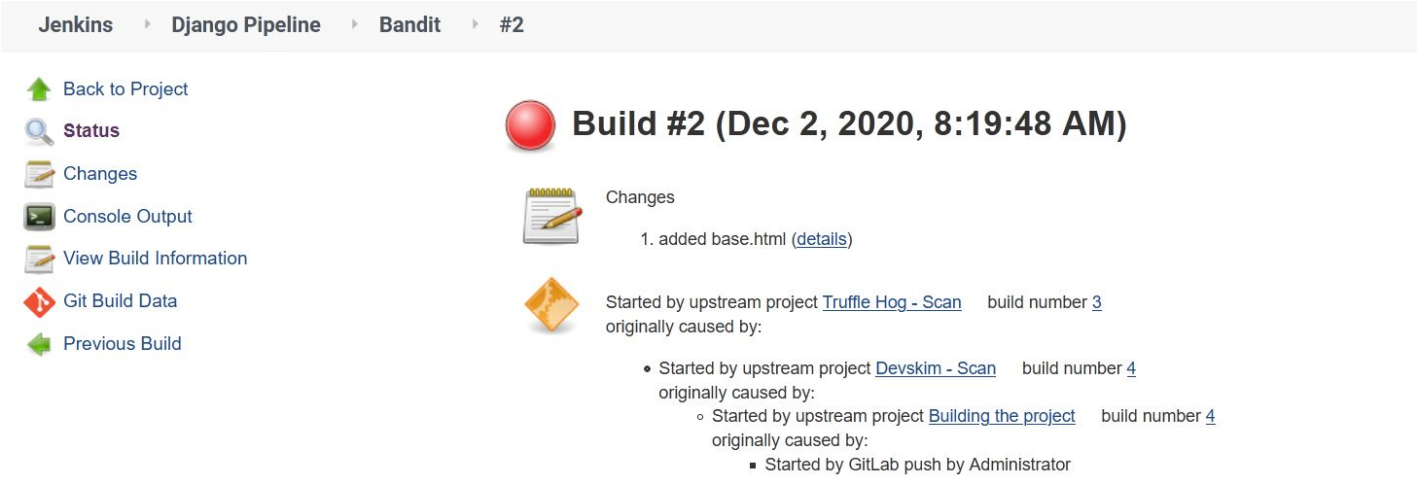
The TruffleHog stage has passed successfully. The logs can be checked from clicking on the "Truffle Hog - Scan" and opening the console output.

```
 > git rev-list --no-walk 0667a319a6d077d3b8e014c9cb4fe7d6b065032a # timeout=10
[Truffle Hog - Scan] $ /bin/sh -xe /tmp/jenkins1943273363200362818.sh
+ /trufflehog.sh
Cloning into 'django-blog'...
FLAG 2: cb4f2a3413adc795a5088dd7e7e01a2f
Triggering a new build of Bandit
Finished: SUCCESS
```

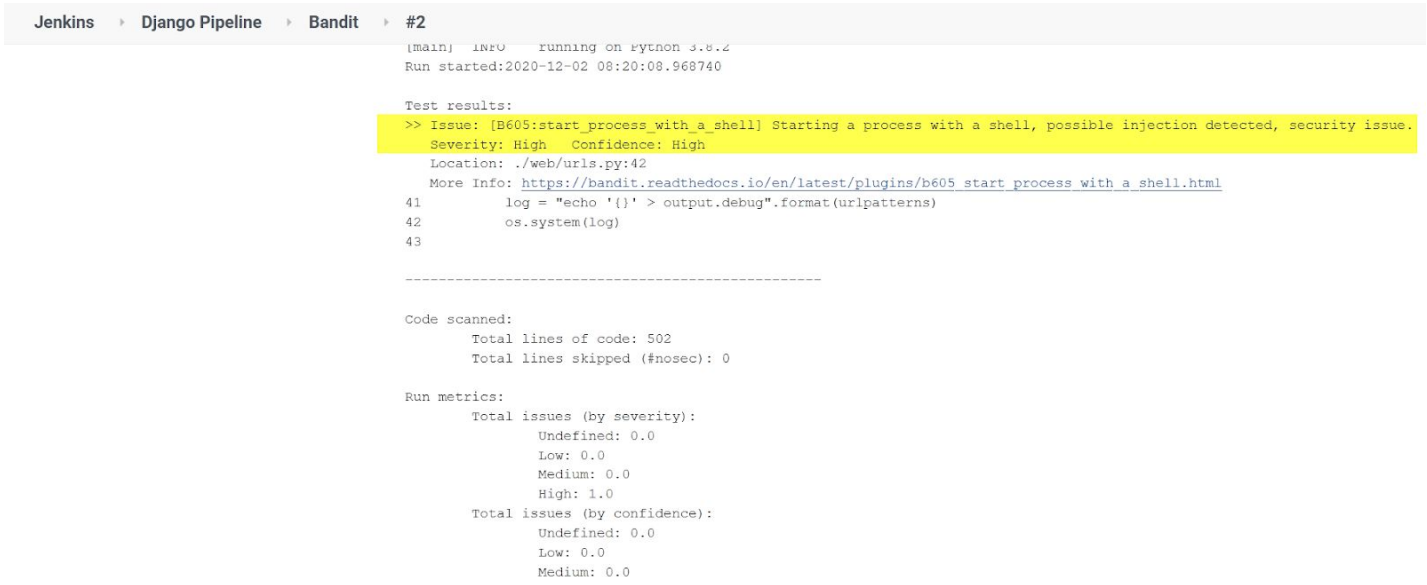
FIAG 2: cb4f2a3413adc795a5088dd7e7e01a2f

Bandit Issue

Step 1: Click on the 'Bandit' to check the job build page.



Step 2: Click on the “Console Output” to check the issues found by Bandit tool.

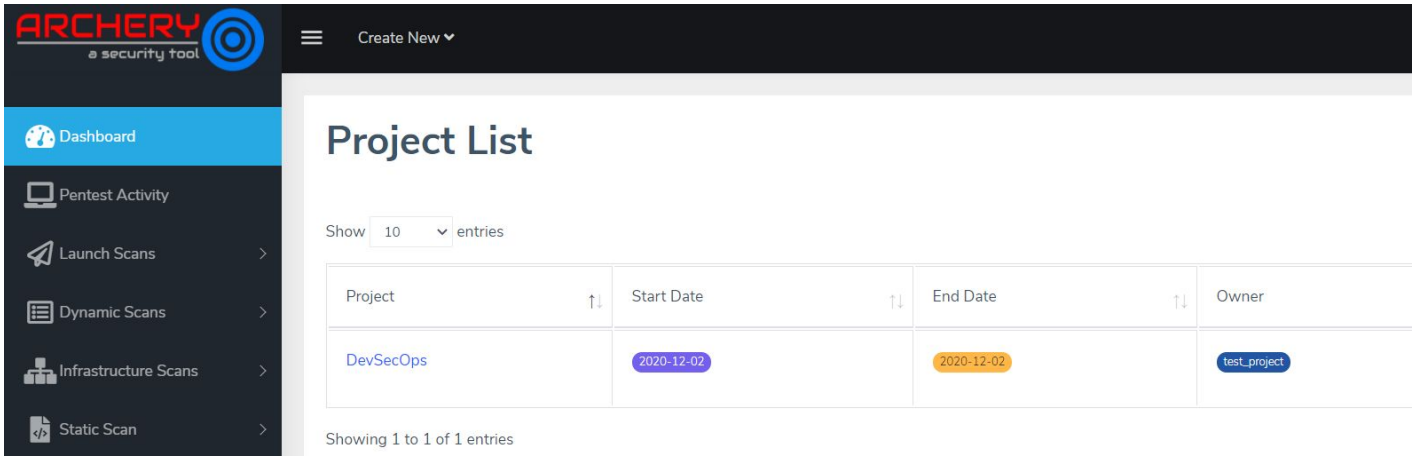


The results of the scan can be checked on archery sec portal.

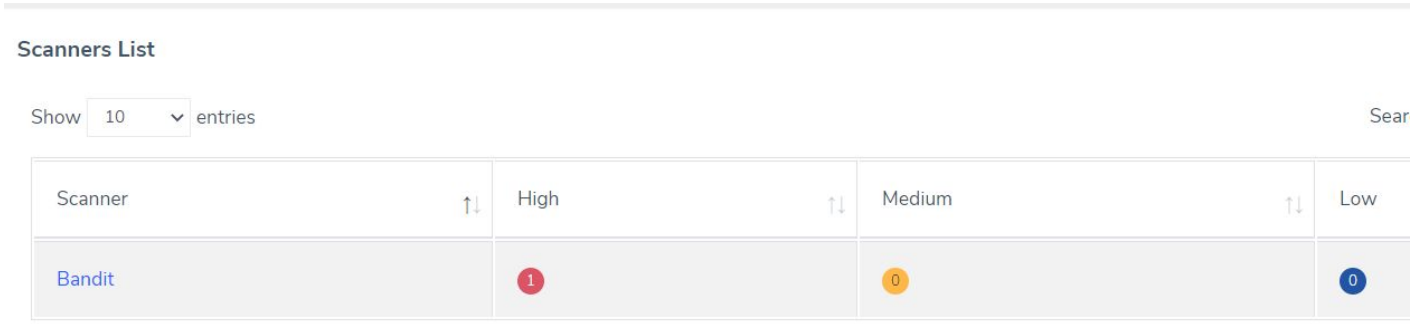
Step 3: Open the ArcherySec page and log in using the credentials provided.

Credentials:

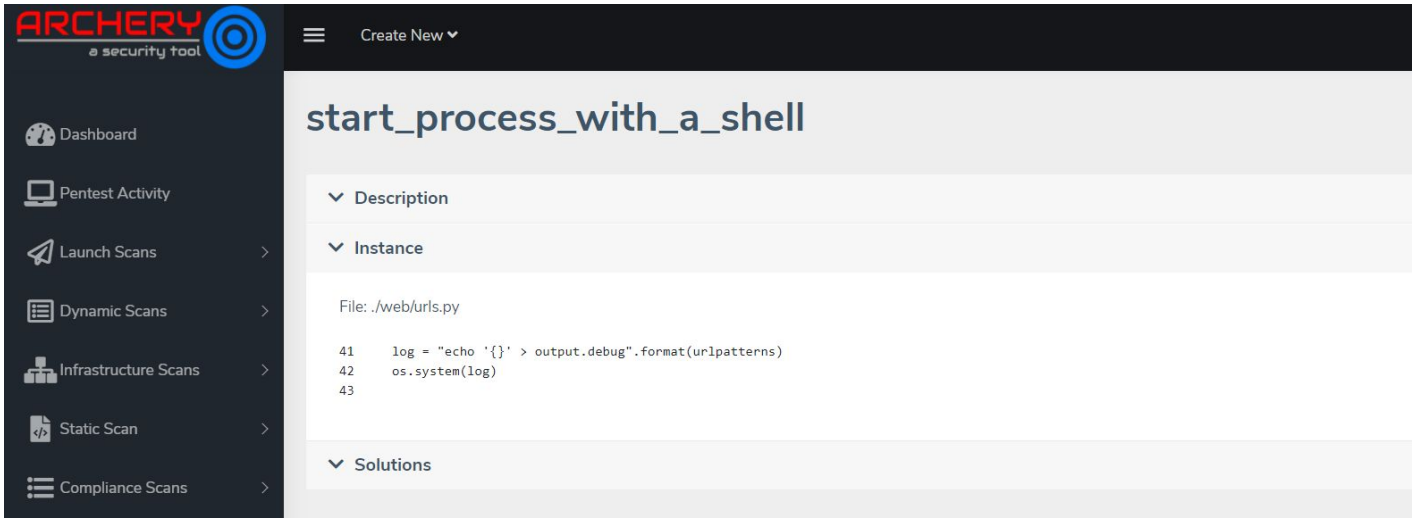
- **Username:** admin
- **Password:** admin



Click on the project name “DevSecOps” and see results from the bandit tool.

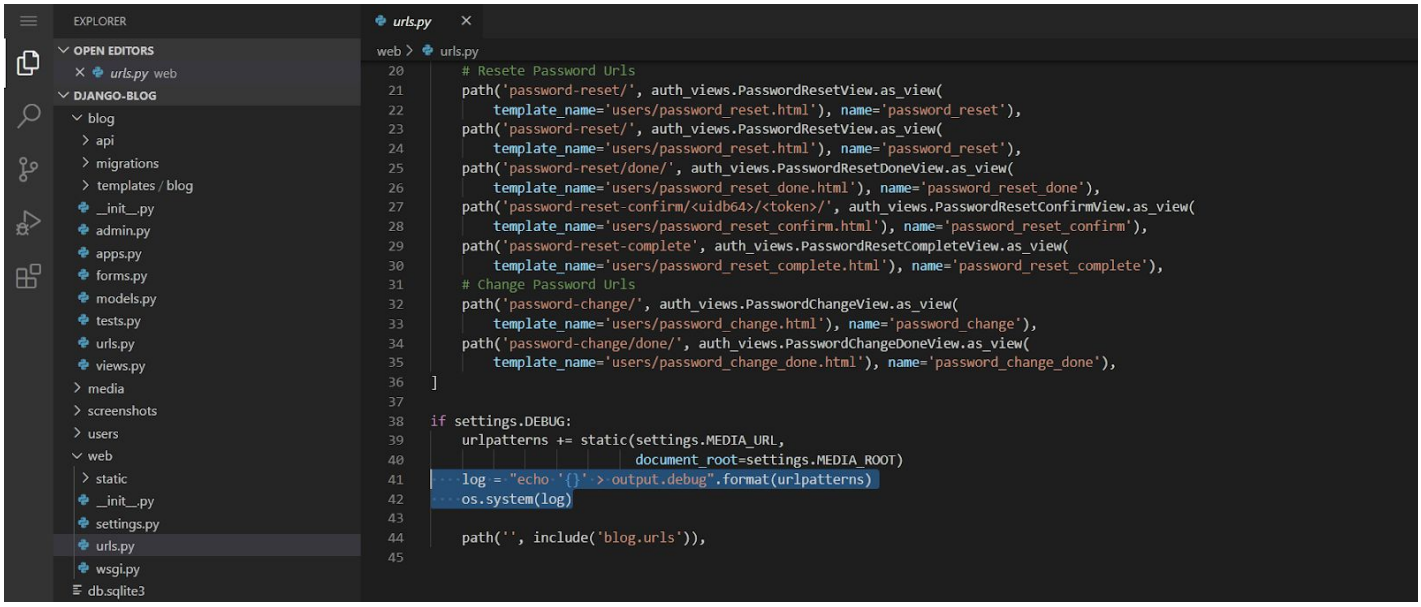


Click on the Bandit and check the issue found. Click on ‘start_process_with_a_shell’ twice on consecutive pages to reach the description of the vulnerability. Check details under ‘Instance’ section.

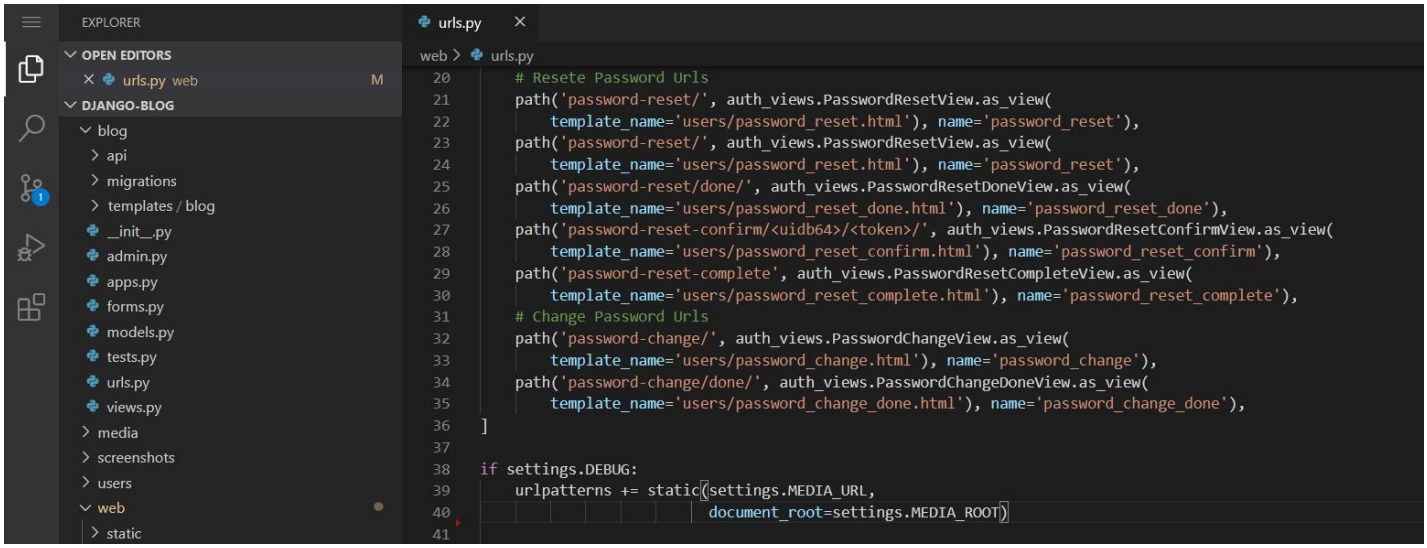


The file web/urls.py has a logging function which is using os class to execute commands on the system which could be abused and lead to remote code execution on the server.

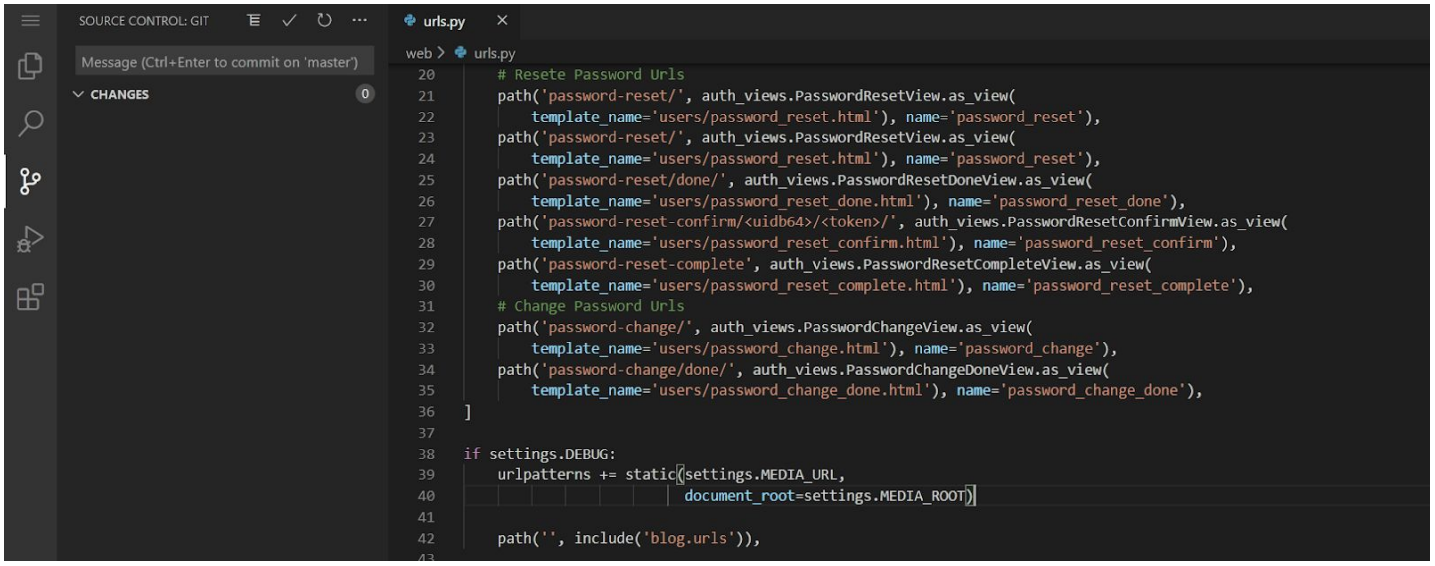
Step 4: Open the file in vs code server.



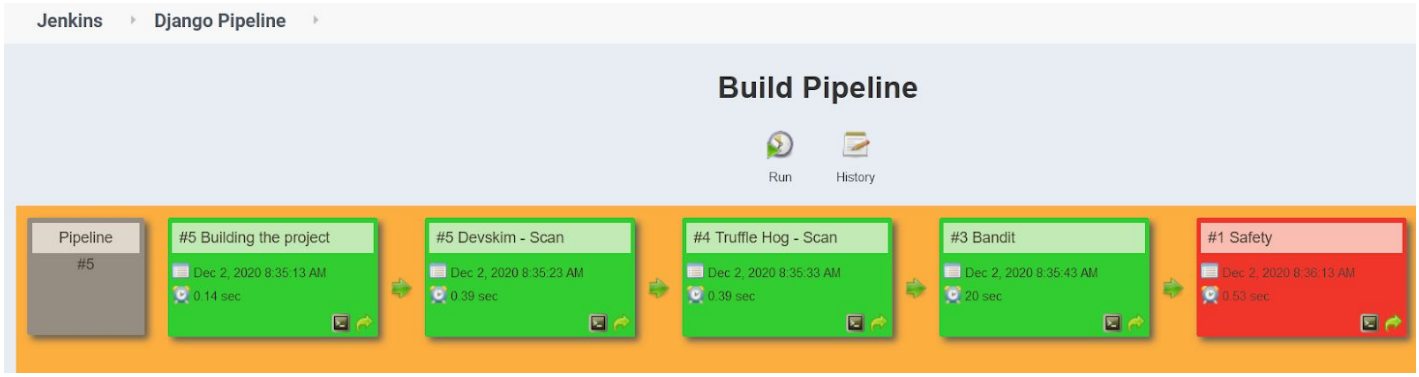
Remove the logging implementation from the file.



Commit and push the changes to the remote repository.
(Same as Step 10 onwards mentioned in ‘Devskim Issue’ section)



Step 5: Check the pipeline to see any changes.



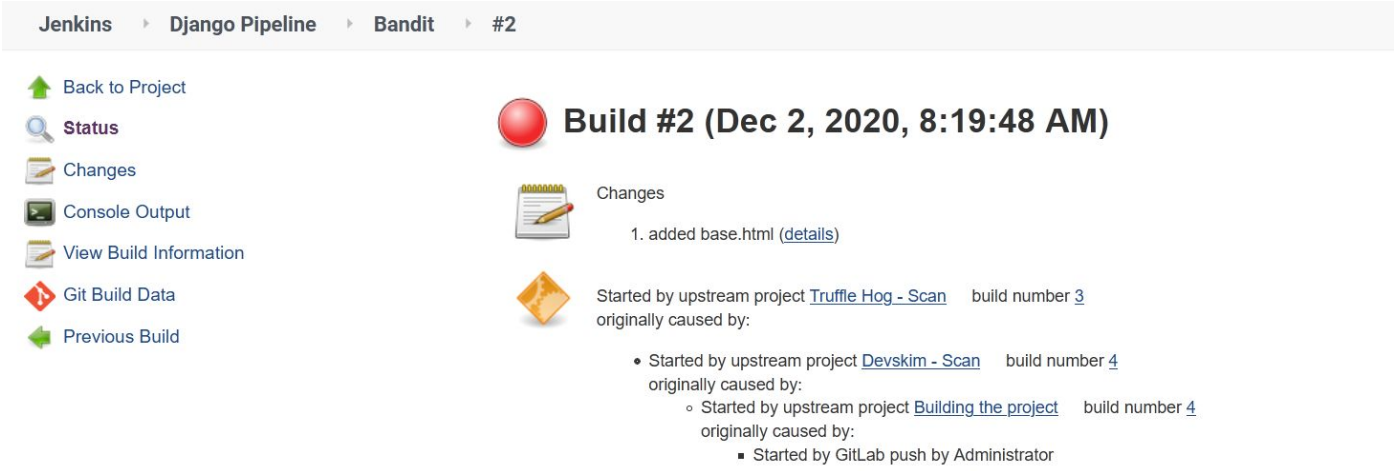
The Bandit stage has passed successfully. The logs can be checked from clicking on the "Bandit" and opening the console output.

```
+ /bandit.sh


                _
    /\          | |
   /  \   _ __ ___| |__   ___ _ __ _   _
  / /\ \ | '__/ __| '_ \ / _ \ '__| | | |
 / ____ \| | | (__| | | |  __/ |  | |_| |
/_/    \_\_|  \___|_| |_|\___|_|   \__, |
                                     __/ |
                                    |___/
 Copyright (C) 2018 ArcherySec
 Open Source Vulnerability Assessment and Management.


{"message":"Scan Data Uploaded","project_id":"356d8291-2173-49a0-9598-2cccd5df4f9a","scan_id":"5a7b9e8e-fc90-4acf-835d-7a47780747b3","scanner":"banditscan"}
FLAG 3: f0ed5839d6addaf6eebc5fc5750ede1c
Triggering a new build of Safety
Finished: SUCCESS
```

FLAG 3: f0ed5839d6addaf6eebc5fc5750ede1c

Safety Issue

Step 1: Click on the 'Safety' to check the job build page.

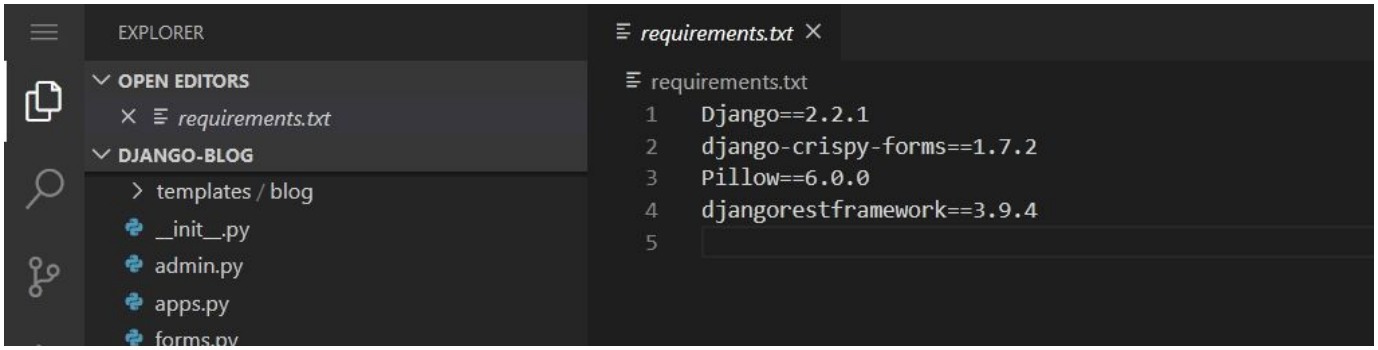


Step 2: Click on the "Console Output" to check the issues found by Safety tool.

```
| REPORT                                                                                |
| checked 4 packages, using local DB                                                    |
+===========================+===========+=========================+==========+
| package                   | installed | affected                | ID       |
+===========================+===========+=========================+==========+
| django                    | 2.2.1     | <2.2.3,>2.2             | 37261    |
| django                    | 2.2.1     | ==2.2.1                 | 37184    |
| django                    | 2.2.1     | >=2.0a1,<2.2.9          | 37771    |
| django                    | 2.2.1     | >=2.2,<2.2.10           | 37970    |
| django                    | 2.2.1     | >=2.2,<2.2.2            | 37191    |

| django                    | 2.2.1     | >=2.2,<2.2.8            | 37766    |
| django                    | 2.2.1     | >=2.2.0,<2.2.11         | 38010    |
| django                    | 2.2.1     | >=2.2.0,<2.2.4          | 37357    |
| django                    | 2.2.1     | >=2.2.0,<2.2.4          | 37331    |
| django                    | 2.2.1     | >=2.2.0,<2.2.4          | 37330    |
| django                    | 2.2.1     | >=2.2.0,<2.2.4          | 37329    |
| django                    | 2.2.1     | >=2.2a1,<2.2.13         | 38372    |
| django                    | 2.2.1     | >=2.2a1,<2.2.13         | 38373    |
| pillow                    | 6.0.0     | <6.2.2                  | 37782    |
| pillow                    | 6.0.0     | <6.2.2                  | 37781    |
| pillow                    | 6.0.0     | <6.2.2                  | 37780    |
| pillow                    | 6.0.0     | <6.2.2                  | 37779    |
| pillow                    | 6.0.0     | <6.2.3                  | 38449    |
| pillow                    | 6.0.0     | <6.2.3                  | 38450    |
| pillow                    | 6.0.0     | <7.0.0                  | 38451    |
| pillow                    | 6.0.0     | <=7.0.0                 | 38452    |
```

There are multiple issues found with the version installed of django and pillow pip module.

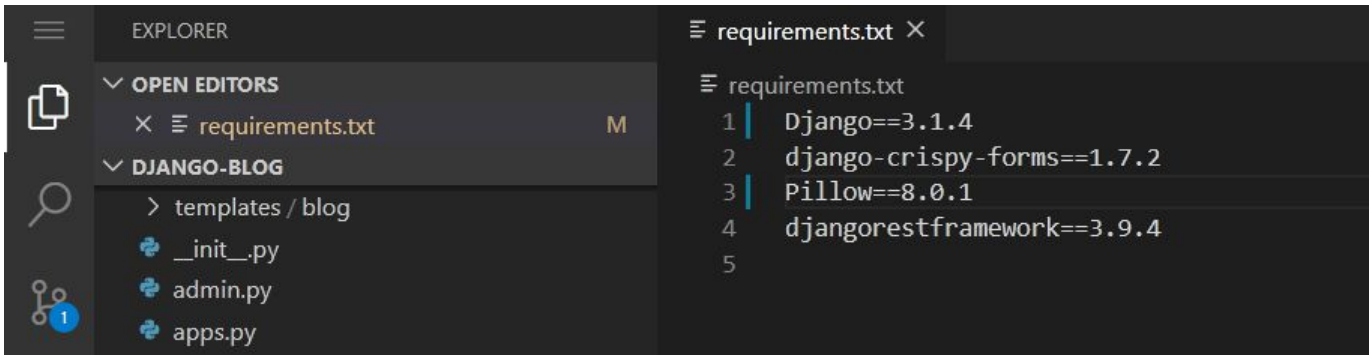
Step 3: Open the requirements.txt file for the django-blog project. (On VS Code instance)



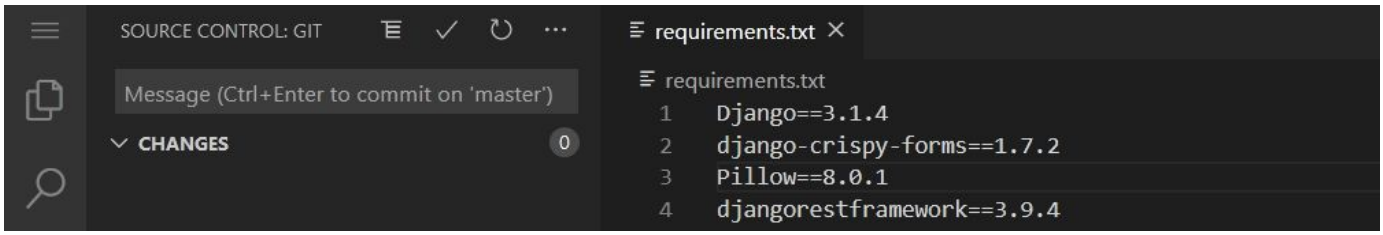
In order to fix the issue, Modify the version numbers from the requirements.txt to the latest in order to resolve the issue.

FROM:
Django==2.2.1
Pillow==6.0.0

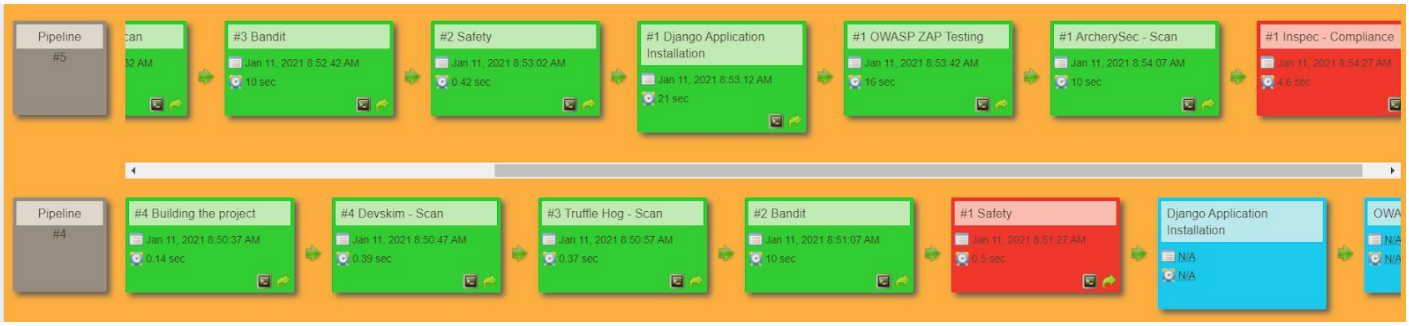
TO:
Django==3.1.4
Pillow==8.0.1



Commit the changes and push the files to the remote repository.
(Same as Step 10 onwards mentioned in 'Devskim Issue' section)



Step 5: Check the pipeline to see any changes.



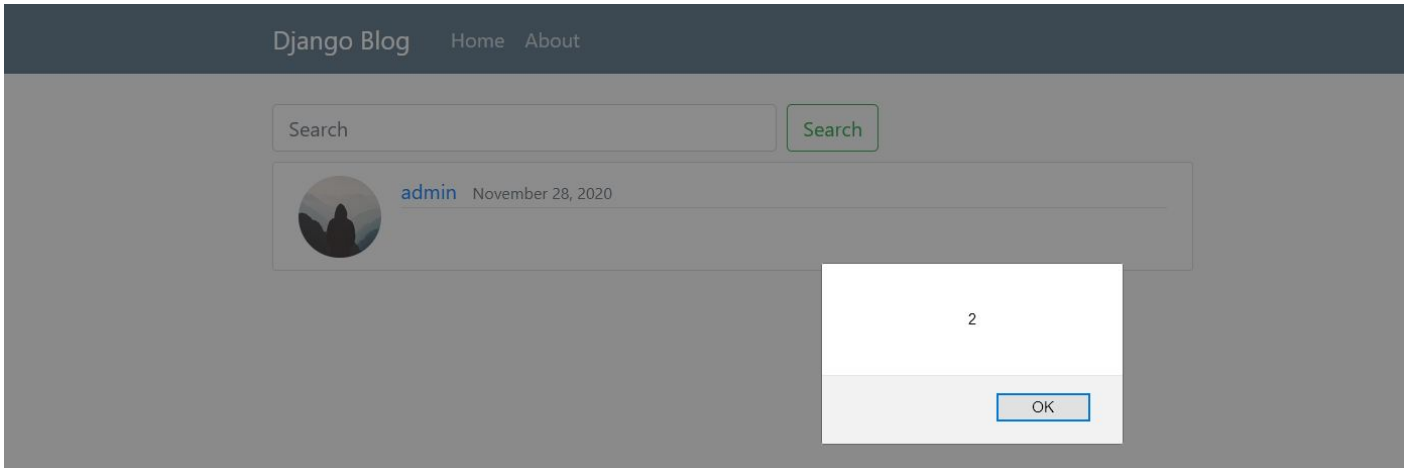
The Safety stage has passed successfully. The logs can be checked from clicking on the "Safety" and opening the console output.

```
+ /safety.sh
+==============================================================================+
|                                                                              |
|                               /$$$$$$            /$$                         |
|                              /$$__  $$          | $$                         |
|           /$$$$$$$  /$$$$$$ | $$  \__//$$$$$$  /$$$$$$   /$$   /$$         |
|          /$$_____/ |____  $$| $$$$   /$$__  $$|_  $$_/  | $$  | $$         |
|         |  $$$$$$   /$$$$$$$| $$_/  | $$$$$$$$  | $$    | $$  | $$         |
|          \____  $$ /$$__  $$| $$    | $$_____/  | $$ /$$| $$  | $$         |
|          /$$$$$$$/|  $$$$$$$| $$    |  $$$$$$$  |  $$$$/|  $$$$$$$         |
|         |_______/  \_______/|__/     \_______/   \___/   \____  $$         |
|                                                             /$$  | $$         |
|                                                            |  $$$$$$/         |
|  by pyup.io                                                 \______/          |
|                                                                              |
+==============================================================================+
| REPORT                                                                       |
| checked 4 packages, using local DB                                           |
+==============================================================================+
| No known security vulnerabilities found.                                     |
+==============================================================================+
FLAG 4: 46d5b8115e47963392f66c5b6941c2e0
Triggering a new build of Django Application Installation
Finished: SUCCESS
```

FLAG 4: 46d5b8115e47963392f66c5b6941c2e0

Cross site scripting and Inspec Issue

Step 1: Open the test-server and check where the vulnerable parameter is located.



The XSS exists in the application.

Step 2: Right-click and check the source code to find ‘2’ in the alert box.

```
68          <article class="media content-section">
69            <img class="rounded-circle article-img" src="/media/default.jpg">
70            <div class="media-body">
71              <div class="article-metadata">
72                <a class="mr-2" href="/user/admin/">admin</a>
73                <small class="text-muted">November 28, 2020</small>
74              </div>
75            <h2><a class="article-title" href="/post/36/"><script>alert(2)</script></a></h2>
76              <p class="article-content"><p>XSS</p></p>
77            </div>
78          </article>
```

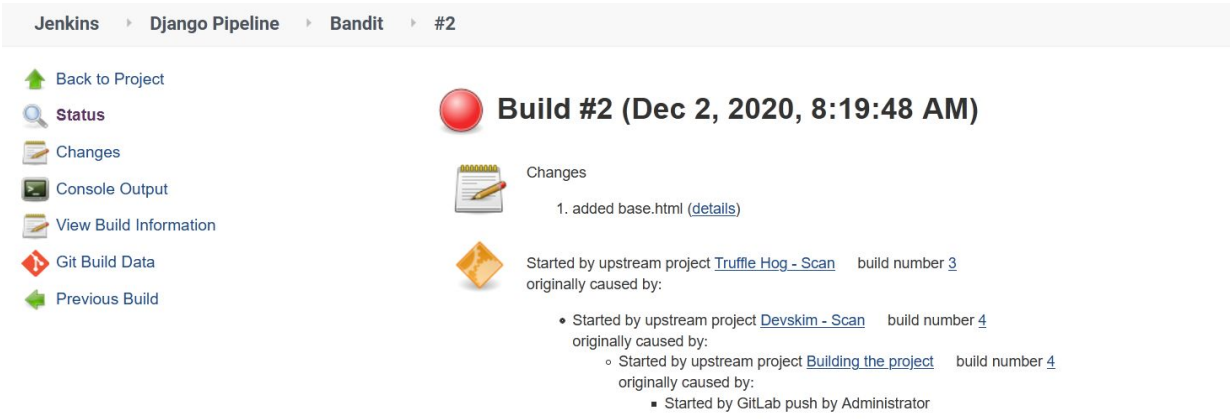
The class ‘article-title’ can be seen which is actually parsing the data without any sanitization.

Step 3: Open the index page of the website and find the reference to the ‘title’ parameter. (On VS Code instance)

The title parameter is found in the index.html of blog/templates/blog/ directory. The function ‘safe’ of Django will disable the HTML escaping on the field.

Step 4: Remove the ‘|safe’ from the post.title section of the code to prevent XSS attack.

Step 5: Click on the 'Inspec' to check the job build page.

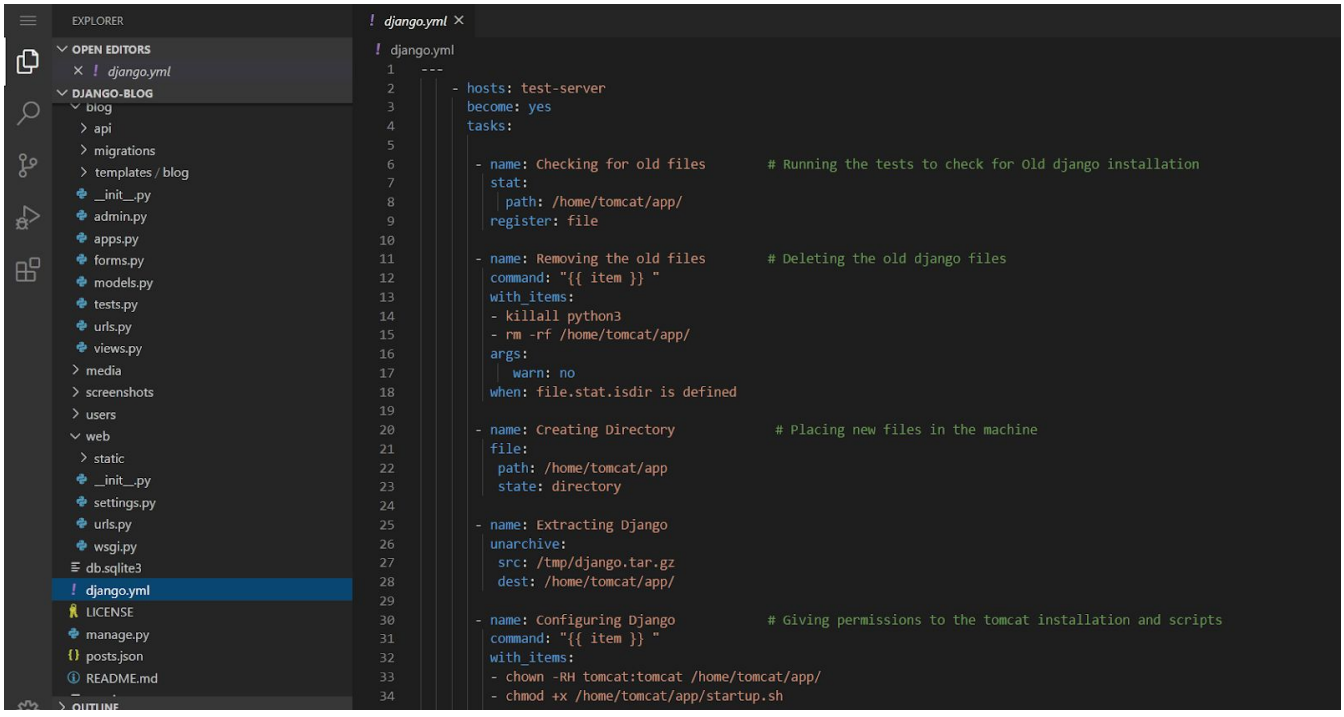


Step 6: Click on the "Console Output" to check the issues found in Inspec.

```
Profile: tests from /django.rb (tests from .django.rb)
Version: (not specified)
Target:  ssh://tomcat@test-server:22

[38;5;9m  ××  tomcat.directories: Check for existence and correct permissions of application directory (1 failed)[0m
[38;5;41m     ×××  Directory /home/tomcat/app/ is expected to be directory[0m
[38;5;41m     ×××  Directory /home/tomcat/app/ owner is expected to eq "tomcat"[0m
[38;5;9m     ××  Directory /home/tomcat/app/ mode is expected to cmp == "0750"

     expected: 0750
          got: 0755

     (compared using `cmp` matcher)
[0m


Profile Summary: 0 successful controls, [38;5;9m1 control failure[0m, 0 controls skipped
Test Summary: [38;5;41m2 successful[0m, [38;5;9m1 failure[0m, 0 skipped
Build step 'Execute shell' marked build as failure
Finished: FAILURE
```

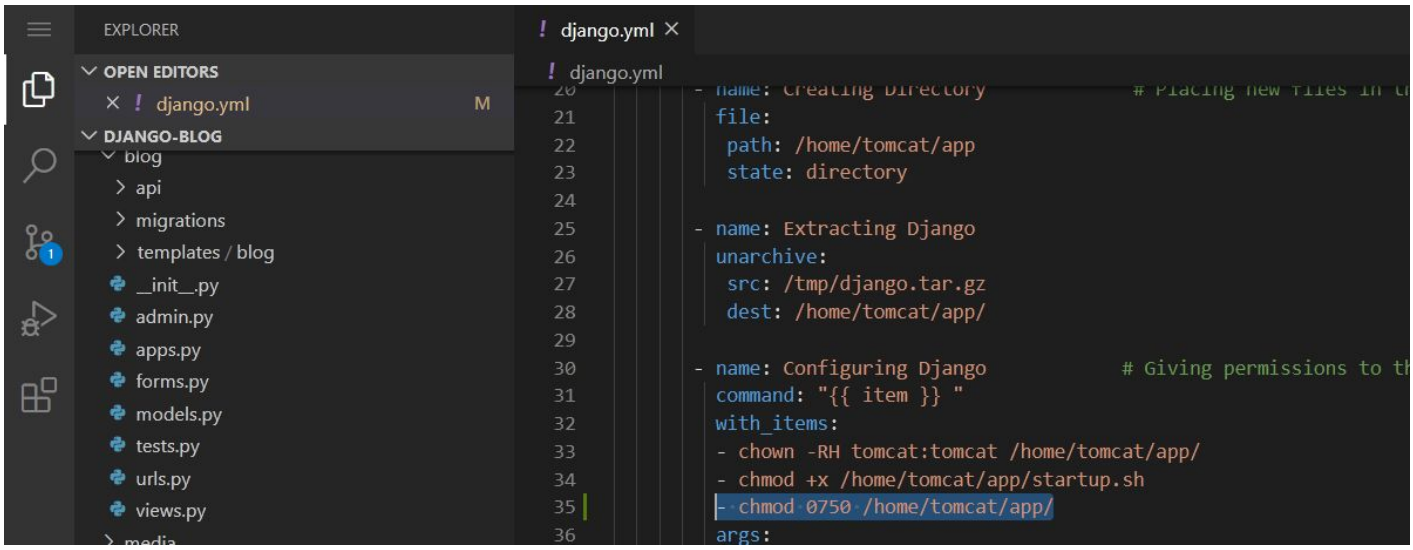
The directory /home/tomcat/app in the remote server needs to be set with permissions 0750 but currently, the permission of the directory is 0755.

Step 7: Open the Django.yml file which is used by ansible to deploy the application on the remote server.

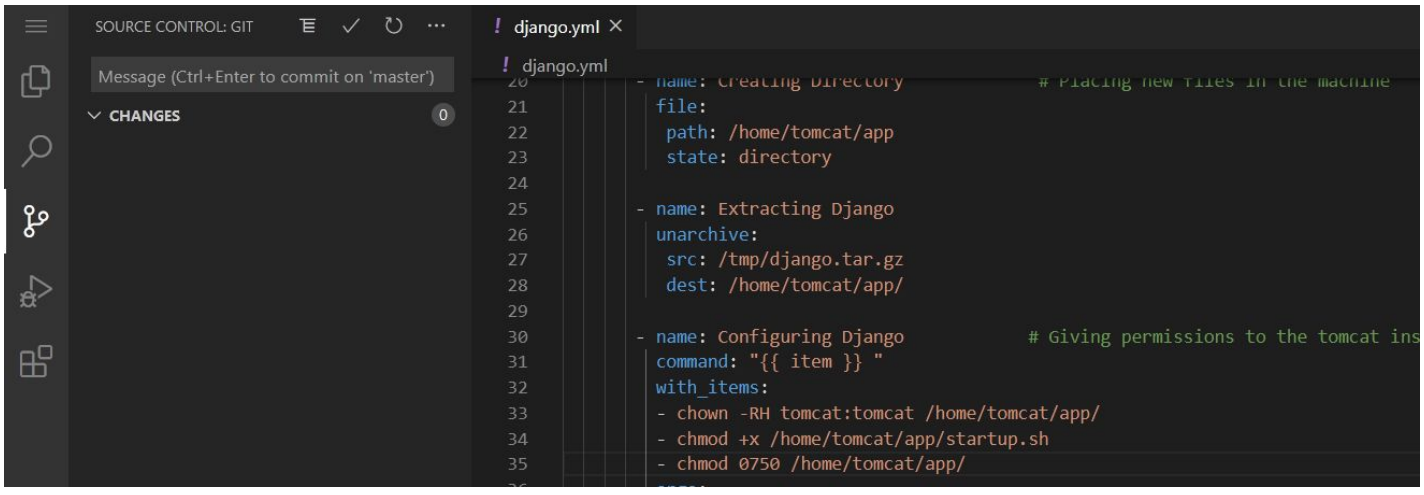


Step 8: Place command to set the appropriate permissions on the ‘app’ directory.

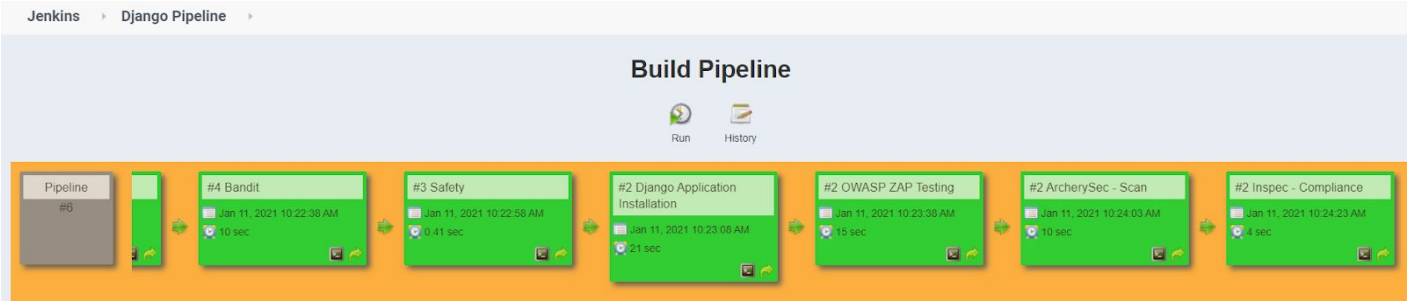
Add the following after line 34.
- chmod 0750 /home/tomcat/app/



Commit the changes and push the files to the remote repository.
(Same as Step 10 onwards mentioned in ‘Devskim Issue’ section)



Step 9: Check the pipeline for the changes.



The Inspec stage has passed successfully. The logs can be checked from clicking on the "Inspec" and opening the console output.

```
+ /inspec.sh

Profile: tests from /django.rb (tests from .django.rb)
Version: (not specified)
Target:  ssh://tomcat@test-server:22

[38;5;41m  [0m  tomcat.directories: Check for existence and correct permissions of application directory[0m
[38;5;41m     [0m  Directory /home/tomcat/app/ is expected to be directory[0m
[38;5;41m     [0m  Directory /home/tomcat/app/ owner is expected to eq "tomcat"[0m
[38;5;41m     [0m  Directory /home/tomcat/app/ mode is expected to cmp == "0750"[0m


Profile Summary: [38;5;41m1 successful control[0m, 0 control failures, 0 controls skipped
Test Summary: [38;5;41m3 successful[0m, 0 failures, 0 skipped
FLAG 5: 3d5ef9972e4954dd883106240858bf6b
Finished: SUCCESS
```

FLAG 5: 3d5ef9972e4954dd883106240858bf6b

Learning

Working on a simple DevSecOps pipeline consisting of different components to fix the issues present in the pipeline