

ATTACK DEFENSE LABS COURSES  
PENTESTER ACADEMY TOOL BOX PENTESTING  
JOINT WORLD-CLASS TRAINERS TRAINING HACKER  
TOOL BOX PATV HACKER  
HACKER PENTESTING  
PATV RED TEAM LABS ATTACK DEFENSE LABS  
TRAINING COURSES ACCESS POINT PENTESTER  
TEAM LABS PENTESTER TOOL BOX PENTESTING  
ACCESS POINT WORLD-CLASS TRAINERS TRAINING  
WORLD-CLASS TRAINERS  
ATTACK DEFENSE LABS TRAINING COURSES PATV ACCESS  
PENTESTER ACADEMY TOOL BOX PENTESTING  
ATTACK DEFENSE LABS TRAINING COURSES PENTESTER ACADEMY  
COURSES PENTESTER ACADEMY TOOL BOX PENTESTING  
TOOL BOX HACKER PENTESTING  
PATV RED TEAM LABS ATTACK DEFENSE LABS  
COURSES PENTESTER ACADEMY  
PENTESTER ACADEMY ATTACK DEFENSE LABS  
ATTACK DEFENSE LABS TRAINING COURSES  
WORLD-CLASS TRAINERS  
RED TEAM TRAINING COURSES  
PENTESTER ACADEMY TOOL BOX PENTESTING

# ATTACK DEFENSE

by PentesterAcademy

<b>Name</b>	Fix the App: Django WebApp
<b>URL</b>	<a href="https://attackdefense.com/challengedetails?cid=2174">https://attackdefense.com/challengedetails?cid=2174</a>
<b>Type</b>	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:

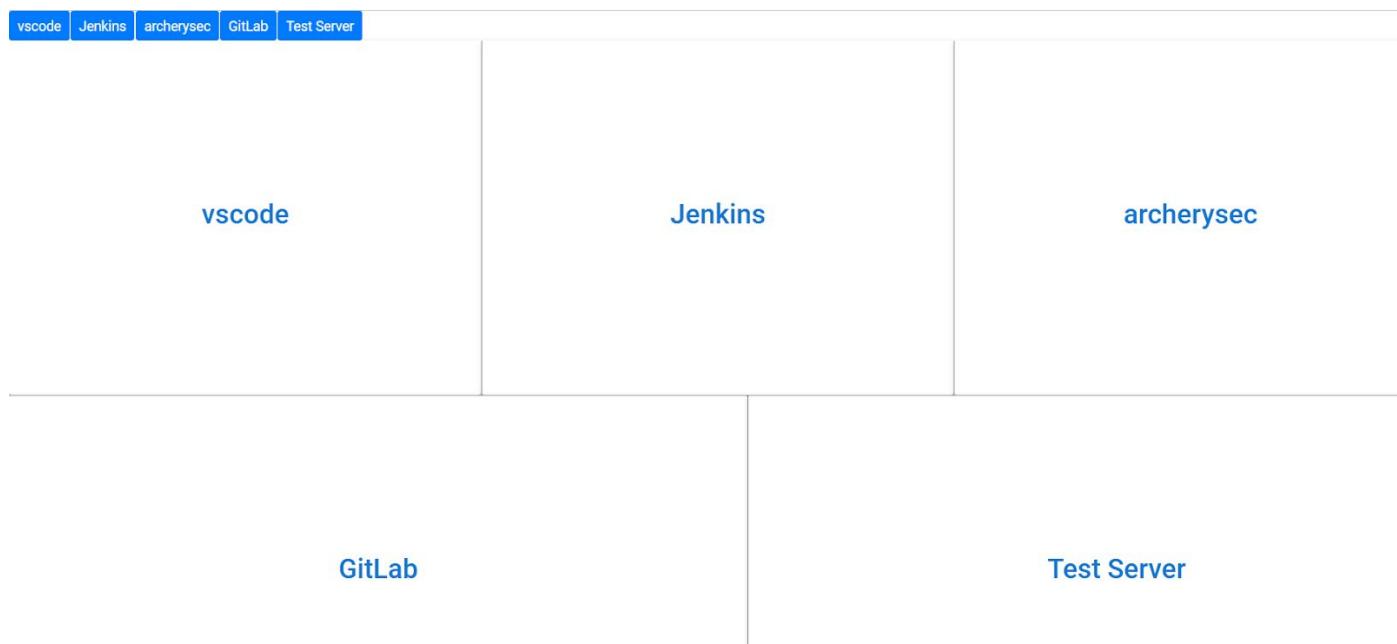
Username	Password
admin	admin

**References:**

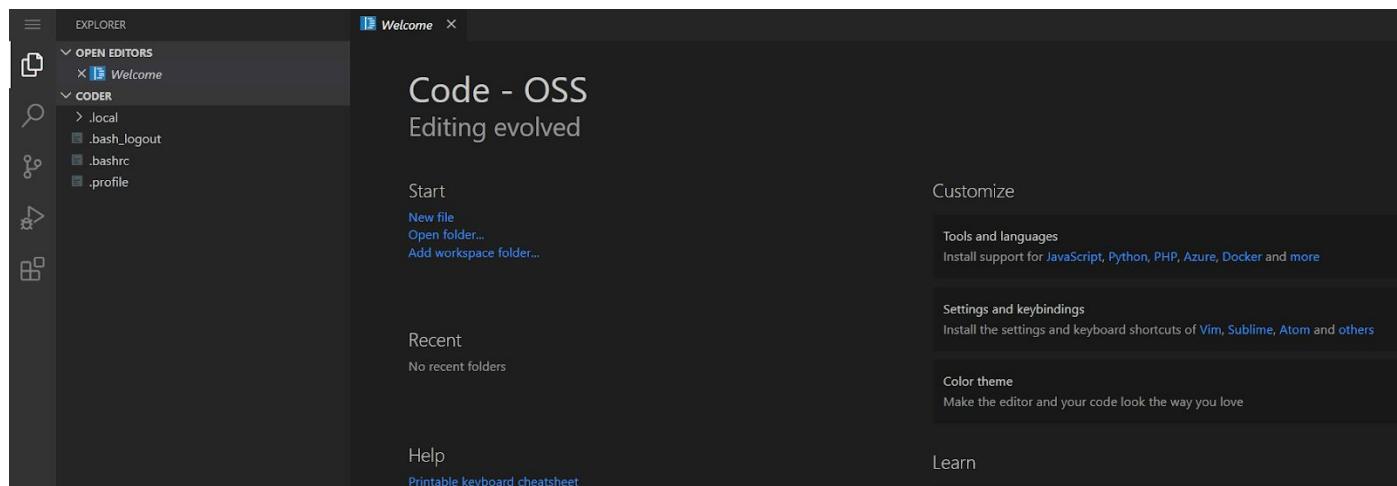
- Django-blog (<https://github.com/AmirAhrari/django-blog>)

## 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.

The screenshot shows the Jenkins dashboard. On the left, there's a sidebar with links like People, Build History, Project Relationship, Check File Fingerprint, Lockable Resources, and Credentials. Below that are sections for Build Queue (empty) and Build Executor Status (2 Idle). The main area has tabs for All and Django Pipeline. A table lists ten build jobs: ArcherySec - Scan, Bandit, Building the project, Devskim - Scan, Django Application Installation, Inspec - Compliance, OWASP ZAP Testing, Safety, Selenium Testing, and Truffle Hog - Scan. Each job entry includes a status icon (green for success, yellow for warning), the job name, the last success date (N/A), the last failure date (N/A), and the last duration (N/A). At the bottom, there are icons for S, M, and L, a legend, and links for Atom feed for all, failures, and latest builds.

S	W	Name ↓	Last Success	Last Failure	Last Duration
Green	Yellow	ArcherySec - Scan	N/A	N/A	N/A
Green	Yellow	Bandit	N/A	N/A	N/A
Green	Yellow	Building the project	N/A	N/A	N/A
Green	Yellow	Devskim - Scan	N/A	N/A	N/A
Green	Yellow	Django Application Installation	N/A	N/A	N/A
Green	Yellow	Inspec - Compliance	N/A	N/A	N/A
Green	Yellow	OWASP ZAP Testing	N/A	N/A	N/A
Green	Yellow	Safety	N/A	N/A	N/A
Green	Yellow	Selenium Testing	N/A	N/A	N/A
Green	Yellow	Truffle Hog - Scan	N/A	N/A	N/A

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

The screenshot shows the ArcherySec login page. It features a logo at the top center with the text "ARCHERY" and "a security tool". Below the logo are two input fields: "Username" with a green person icon and "Password" with an orange lock icon. At the bottom are two buttons: "Sign Up" on the left and "Sign in" on the right.

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



## GitLab Community Edition

### Open source software to collaborate on code

Manage Git repositories with fine-grained access controls that keep your code secure. Perform code reviews and enhance collaboration with merge requests. Each project can also have an issue tracker and a wiki.

<a href="#">Sign in</a>	<a href="#">Register</a>
Username or email	
<input type="text"/>	
Password	
<input type="password"/>	
<input type="checkbox"/> Remember me	<a href="#">Forgot your password?</a>
<a href="#">Sign in</a>	

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

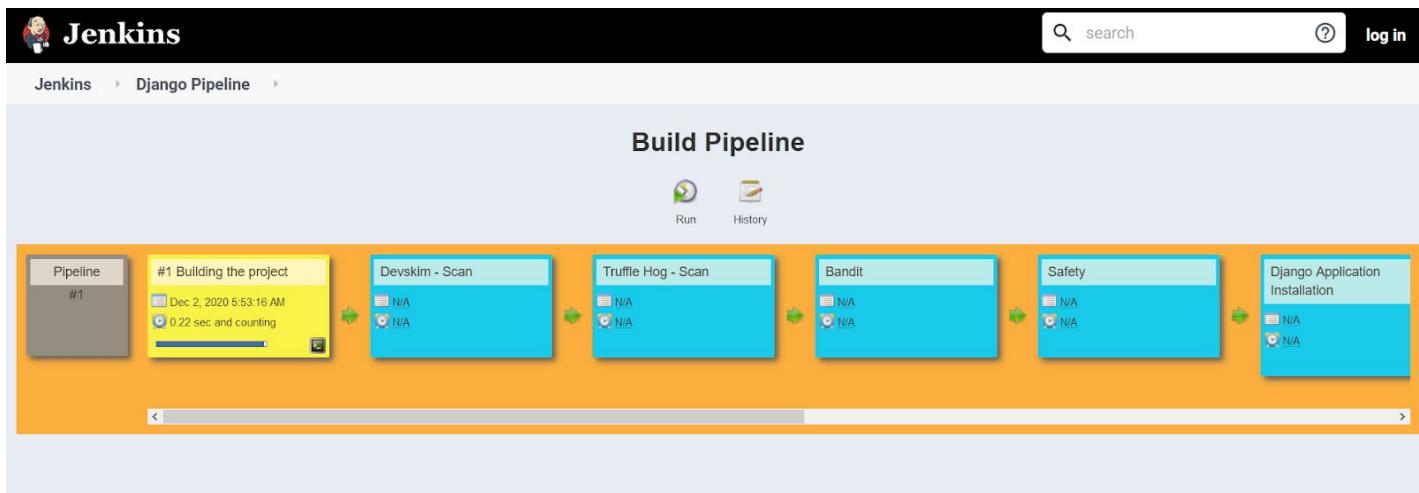
The screenshot shows the Jenkins web interface. At the top, there is a navigation bar with a Jenkins logo, a search bar, and a 'log in' button. Below the navigation bar, the page title is 'Jenkins > Django Pipeline'. On the left side, there is a sidebar with links to 'People', 'Build History', 'Project Relationship', 'Check File Fingerprint', 'Lockable Resources', and 'Credentials'. Under 'Build Queue', it says 'No builds in the queue.' Under 'Build Executor Status', it shows '1 Idle' and '2 Idle'. The main content area displays a table of build jobs. The table has columns for 'S' (Status), 'W' (Last Build Result), 'Name', 'Last Success', 'Last Failure', and 'Last Duration'. There are 10 rows in the table, each representing a different build job: 'ArcherySec - Scan', 'Bandit', 'Building the project', 'Devskim - Scan', 'Django Application Installation', 'Inspec - Compliance', 'OWASP ZAP Testing', 'Safety', 'Selenium Testing', and 'Truffle Hog - Scan'. Each row includes a small icon next to the name and a green circular progress bar indicating the status. At the bottom of the table, there are icons for 'Icon', 'S', 'M', and 'L', and links for 'Legend', 'Atom feed for all', 'Atom feed for failures', and 'Atom feed for just latest builds'.

All	Django Pipeline				
S	W	Name ↓	Last Success	Last Failure	Last Duration
Idle	Green	ArcherySec - Scan	N/A	N/A	N/A
Idle	Green	Bandit	N/A	N/A	N/A
Idle	Green	Building the project	N/A	N/A	N/A
Idle	Green	Devskim - Scan	N/A	N/A	N/A
Idle	Green	Django Application Installation	N/A	N/A	N/A
Idle	Green	Inspec - Compliance	N/A	N/A	N/A
Idle	Green	OWASP ZAP Testing	N/A	N/A	N/A
Idle	Green	Safety	N/A	N/A	N/A
Idle	Green	Selenium Testing	N/A	N/A	N/A
Idle	Green	Truffle Hog - Scan	N/A	N/A	N/A

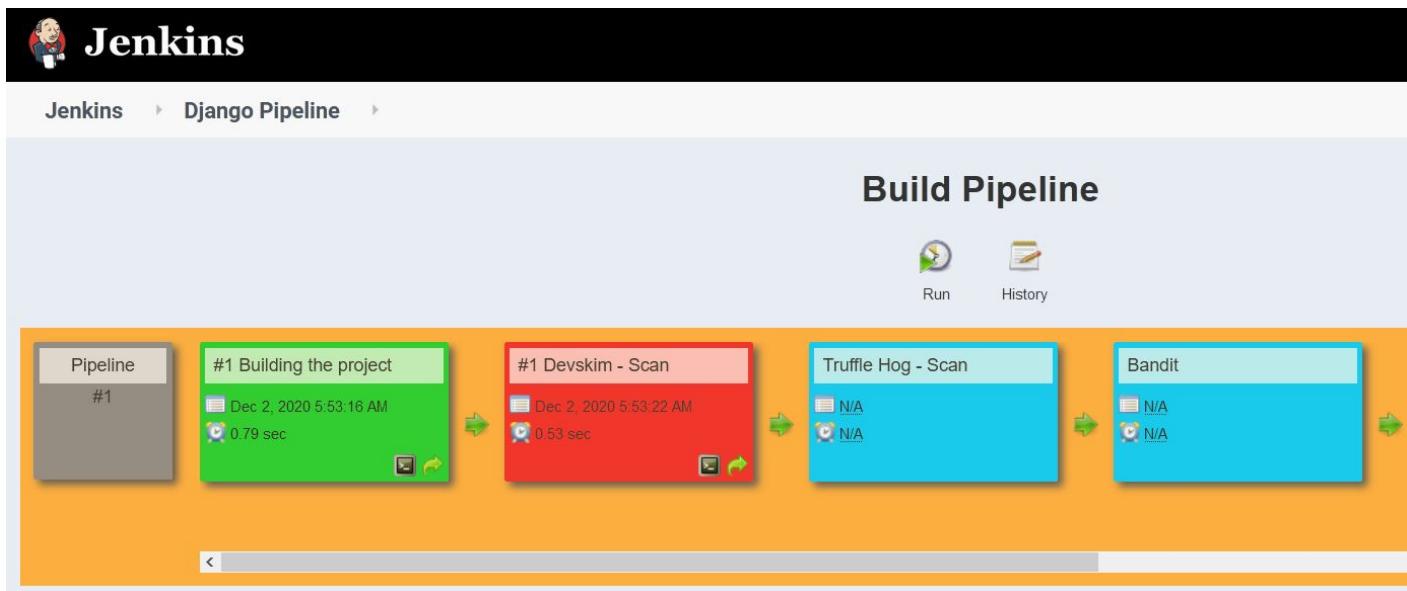
There are 10 jobs present in the Jenkins interface. Navigate to the Django Pipeline view section.

The screenshot shows the 'Build Pipeline' view within Jenkins. At the top, there is a navigation bar with a Jenkins logo, a search bar, and a 'log in' button. Below the navigation bar, the page title is 'Jenkins > Django Pipeline > Build Pipeline'. The main content area is titled 'Build Pipeline'. It contains two buttons: 'Run' and 'History'. Below the buttons, a message says 'This view has no jobs associated with it.'

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.

The Jenkins pipeline interface shows the first step of a pipeline named "Django Pipeline". The step is titled "Devskim - Scan" and has a build number "#1". The status is shown as a red circle with a white exclamation mark, indicating a failure. The pipeline navigation bar at the top includes "Jenkins", "Django Pipeline", "Devskim - Scan", and "#1". On the left, there is a sidebar with links: "Back to Project", "Status", "Changes", "Console Output", "View Build Information", and "Git Build Data".

## Build #1 (Dec 2, 2020, 5:53:22 AM)



No changes.



Started by upstream project [Building the project](#) build number 1 originally caused by:

- Started by anonymous user



**Revision:** 670639799231a6c45ad42257681331b4fb32a169

- refs/remotes/origin/master

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

The Jenkins pipeline interface shows the console output for the "Devskim - Scan" step. The output shows the execution of a shell script that performs a git fetch and then runs the devskim tool. A yellow box highlights the command "file:./.aws/credentials.txt" and its output "region:2,25,2,70 - DS173237 [Important] - Do not store tokens or keys in source code.". The output also indicates 1 issue found in 1 file, 34 files analyzed, 125 files skipped, and marks the build as failure.

```

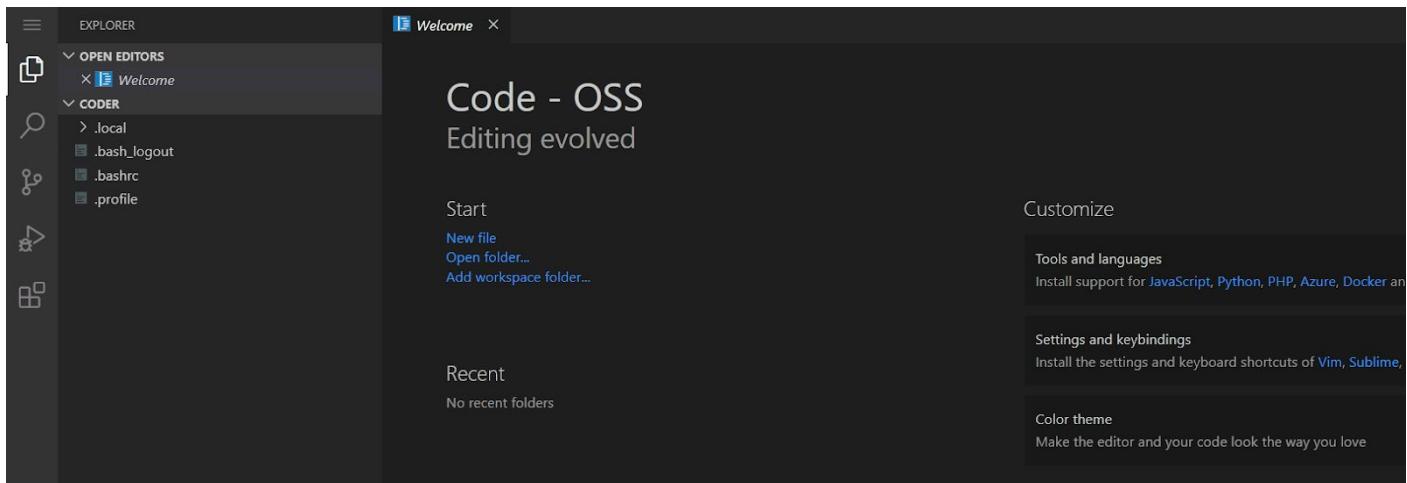
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/Devskim - Scan
No credentials specified
Cloning the remote Git repository
Cloning repository http://gitlab/root/django-blog.git
> git init /var/lib/jenkins/workspace/Devskim - Scan # timeout=10
Fetching upstream changes from http://gitlab/root/django-blog.git
> git --version # timeout=10
> git --version # 'git version 2.25.1'
> git fetch --tags --force --progress -- http://gitlab/root/django-blog.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git config remote.origin.url http://gitlab/root/django-blog.git # timeout=10
> git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
> git config remote.origin.url http://gitlab/root/django-blog.git # timeout=10
Fetching upstream changes from http://gitlab/root/django-blog.git
> git fetch --tags --force --progress -- http://gitlab/root/django-blog.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git rev-parse refs/remotes/origin/master^{commit} # timeout=10
> git rev-parse refs/remotes/origin/master^{commit} # timeout=10
Checking out Revision 670639799231a6c45ad42257681331b4fb32a169 (refs/remotes/origin/master)
> git config core.sparsecheckout # timeout=10
> git checkout -f 670639799231a6c45ad42257681331b4fb32a169 # timeout=10
Commit message: "ADD files"
First time build. Skipping changelog.
[Devskim - Scan] $ /bin/sh -xe /tmp/jenkins13785183420240724028.sh
+ ./devskim.sh
file:./.aws/credentials.txt
region:2,25,2,70 - DS173237 [Important] - Do not store tokens or keys in source code.

Issues found: 1 in 1 files
Files analyzed: 34
Files skipped: 125
Build step 'Execute shell' marked build as failure
Finished: FAILURE

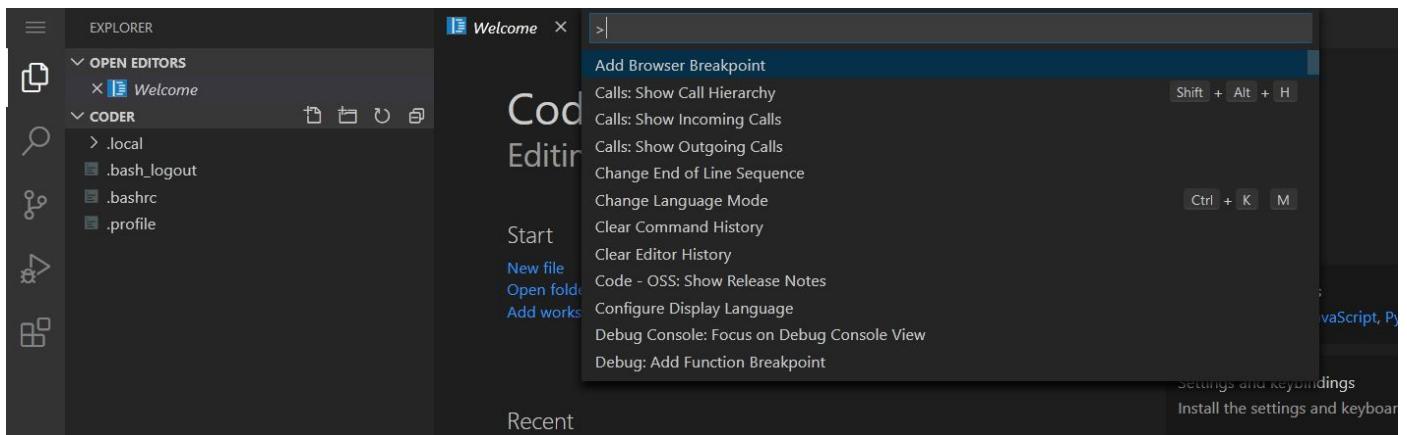
```

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

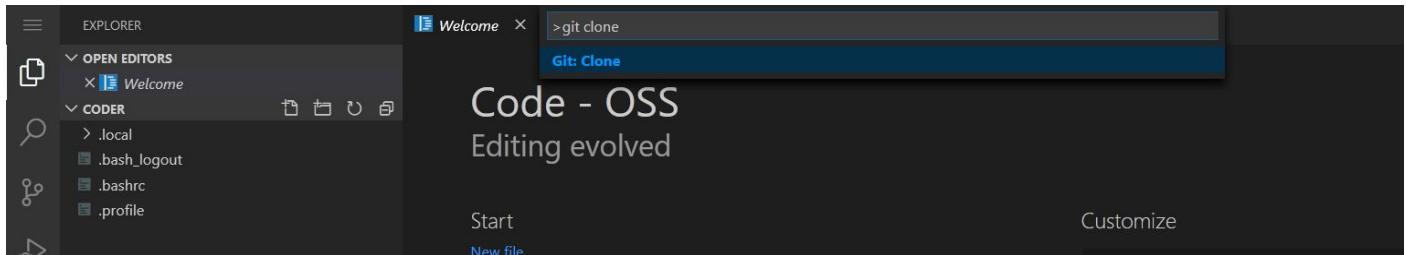
**Step 3:** Open the vscode server page.



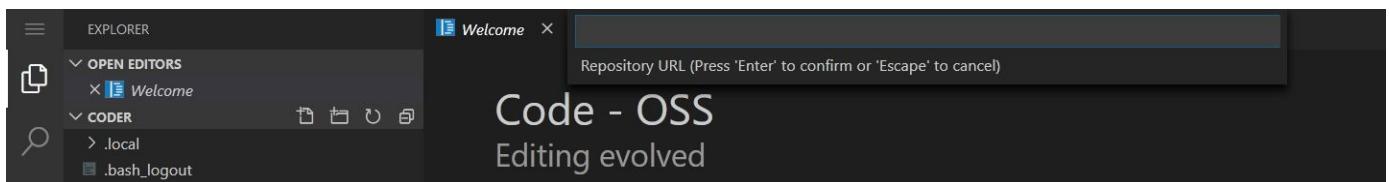
Press CTRL + SHIFT + P to open the command palette or click on the settings bar available in the bottom left and choose the “Command Palette” option.



**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

A screenshot of the GitLab interface. The top navigation bar includes links for 'Projects', 'Groups', 'More', and a search bar. The main area is titled 'Projects' and shows a list of projects. One project, 'django-blog', is listed, showing its status as 'Administrator / django-blog' and 'Maintainer'. The repository has 0 stars, 0 forks, 0 issues, and 0 pull requests.

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

The screenshot shows the GitLab interface for a project named 'django-blog'. The top navigation bar includes links for 'Projects', 'Groups', 'More', and search. A prominent yellow banner at the top states: 'You won't be able to pull or push project code via SSH until you add an SSH key to your profile'. Below this, the project details page is displayed. It features a sidebar with links for 'Project overview', 'Details', 'Activity', 'Releases', 'Repository', 'Issues', 'Merge Requests', 'CI / CD', 'Operations', and 'Analytics'. The main content area shows the project name 'django-blog' with a 'D' icon, a 'Project ID: 2', and statistics: 1 Commit, 1 Branch, 0 Tags, and 1.1 MB Files. A dropdown menu shows 'master' selected. To the right are buttons for 'History', 'Find file', 'Web IDE', 'Download', and 'Clone'. Below these are sections for 'ADD files' (root authored 3 days ago), 'README', 'MIT License', 'Add CHANGELOG', 'Add CONTRIBUTING', 'Enable Auto DevOps', and 'Add Kubernetes cluster'. A 'Set up CI/CD' button is also present.

Copy the git link to the GitLab repository.

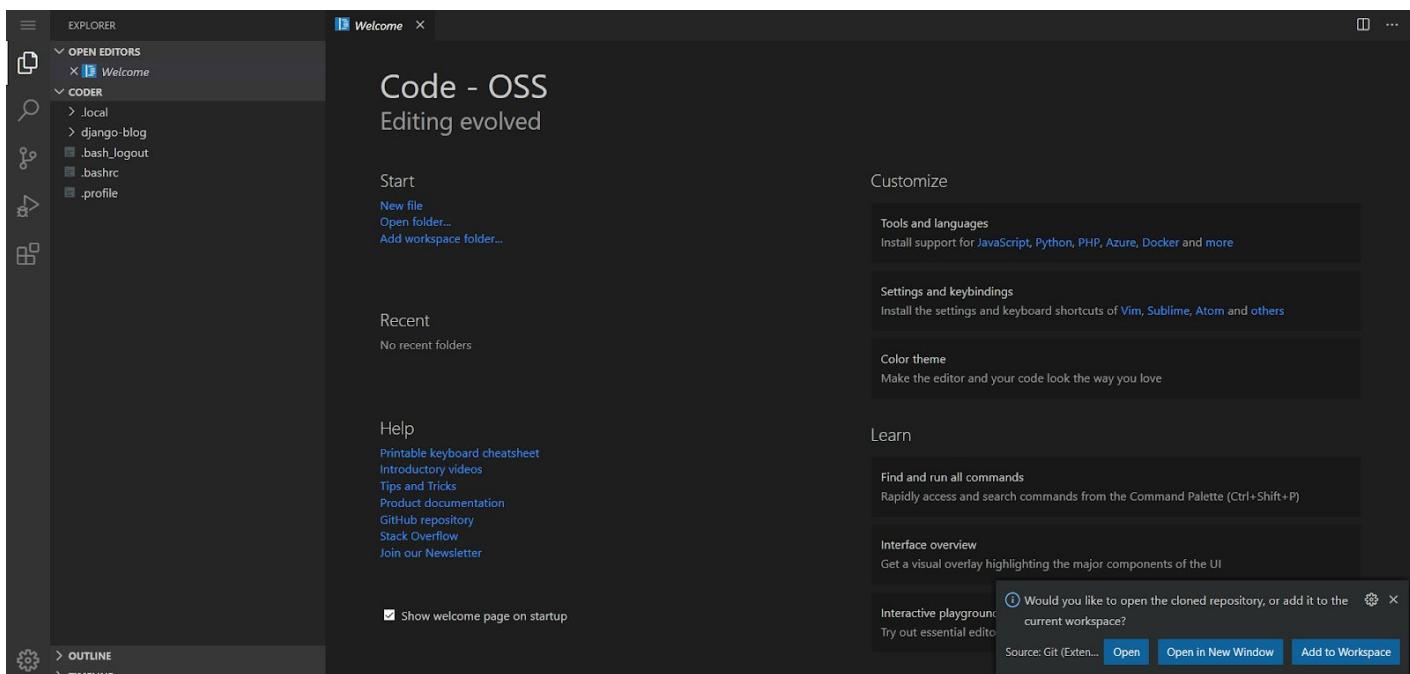
**Step 6:** Enter the link into the clone function at vscode server.

The screenshot shows the VS Code interface with the 'Welcome' tab active. In the center, it says 'Code - OSS' and 'Editing evolved'. On the left, the 'EXPLORER' sidebar shows 'OPEN EDITORS' containing 'Welcome' and 'CODER' containing '.local', '.bash\_logout', and '.bashrc'. At the top, a search bar contains the URL 'http://gitlab/root/django-blog.git'. Below the search bar is a placeholder text: 'Repository URL (Press 'Enter' to confirm or 'Escape' to cancel)'. A blue 'OK' button is located in the bottom right corner of the input field.

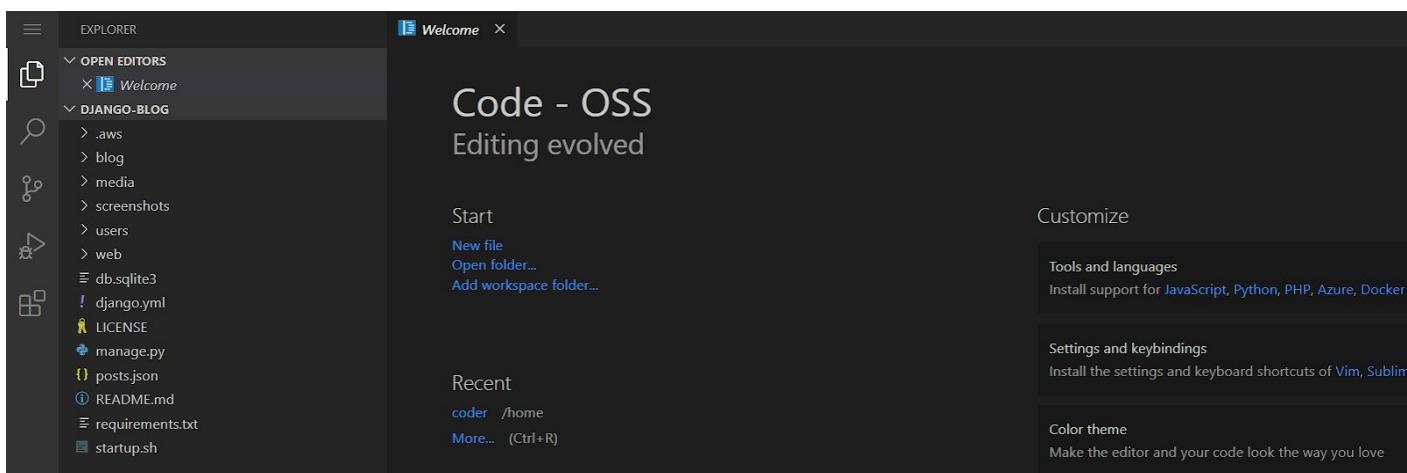
Press Enter.

This screenshot is similar to the previous one, but the URL in the search bar has been confirmed. The 'OK' button is now highlighted in blue. The repository path '/home/coder/' is visible in the input field.

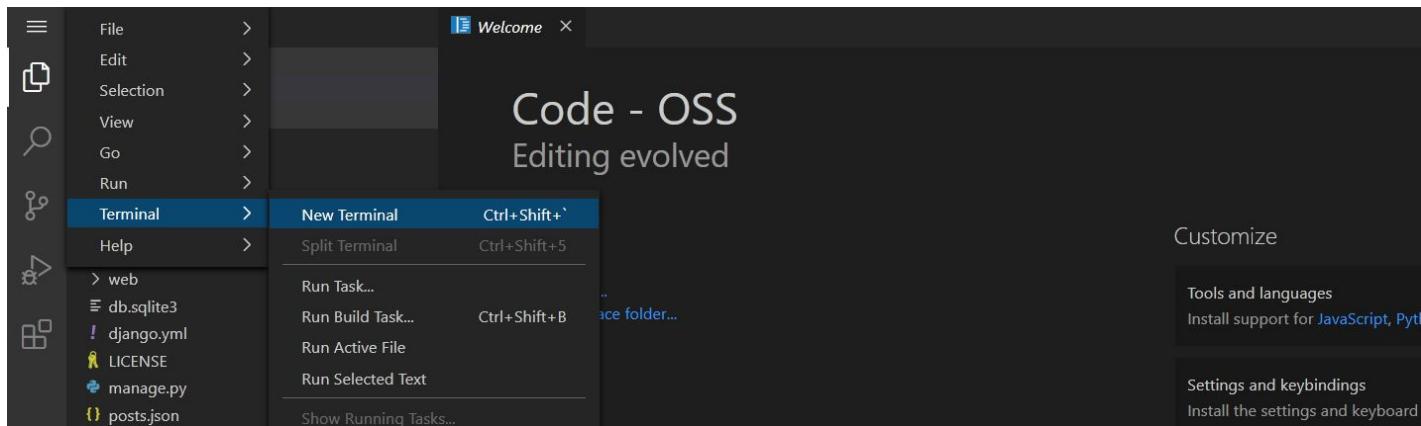
Choose the path to clone to the repository.



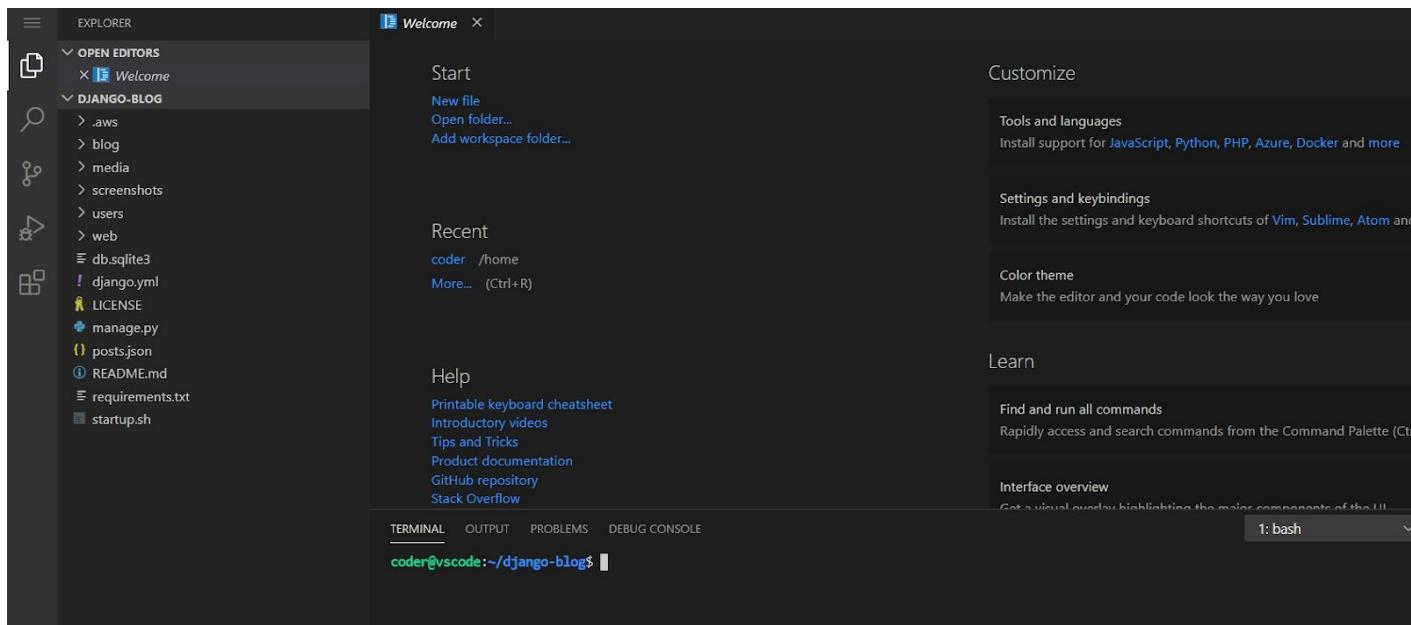
Click on the “Open” button to open the source code directory of the django-blog.



**Step 7:** Open the Terminal in the directory.



Select the “New Terminal” option.



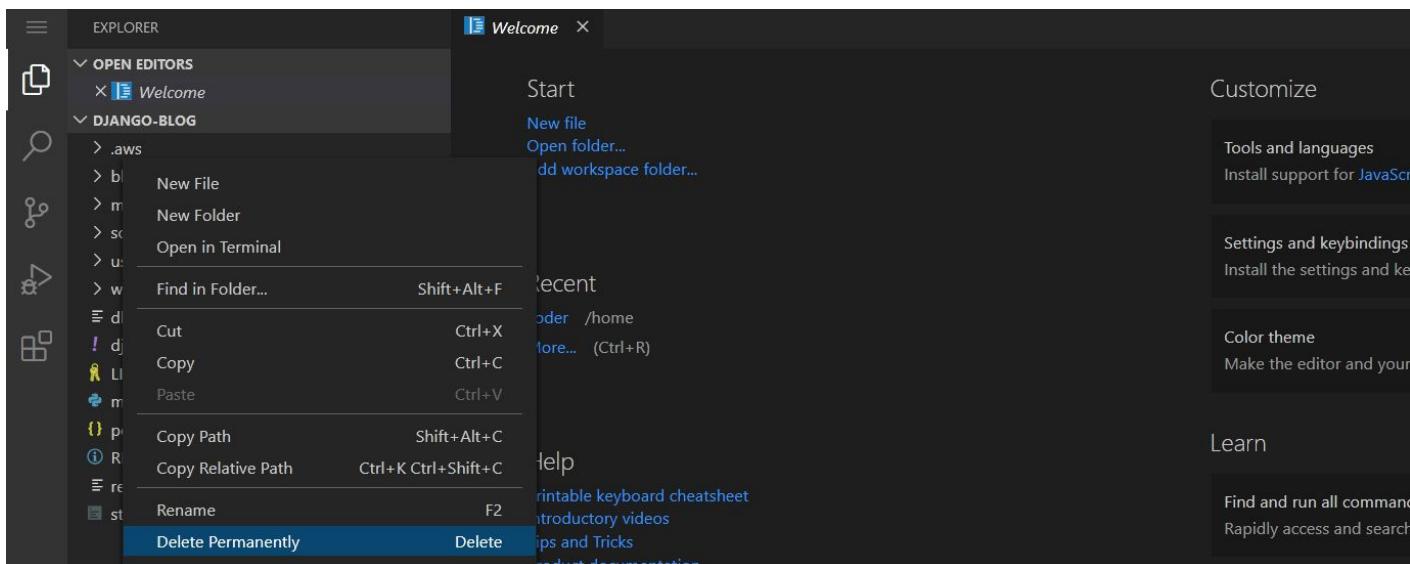
**Step 8:** Enter the git credentials into the terminal to save the identity of the user.

### Commands:

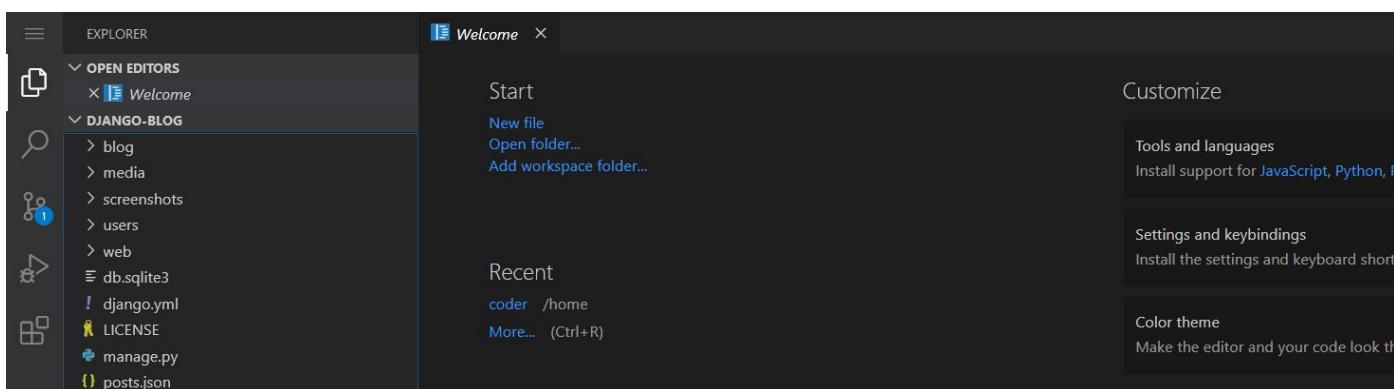
```
git config --global user.email "root@attacker.xyz"  
git config --global user.name "root"
```

```
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"
```

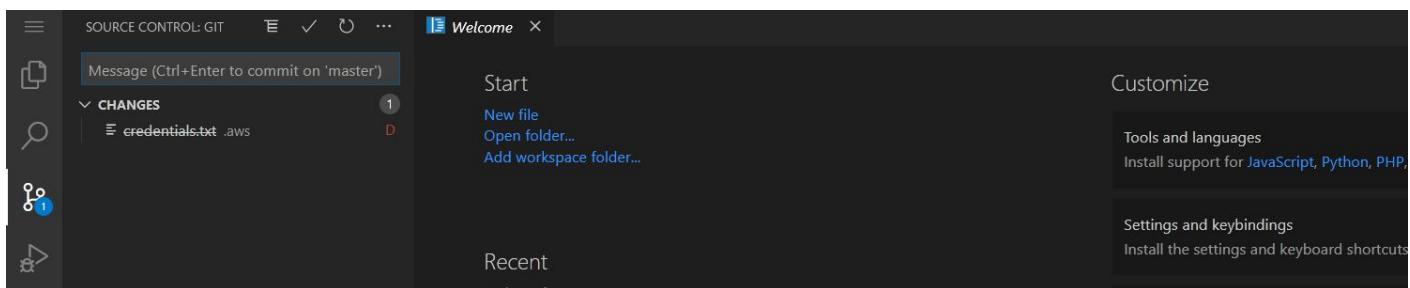
**Step 9:** Right-click on the AWS (.aws) directory.



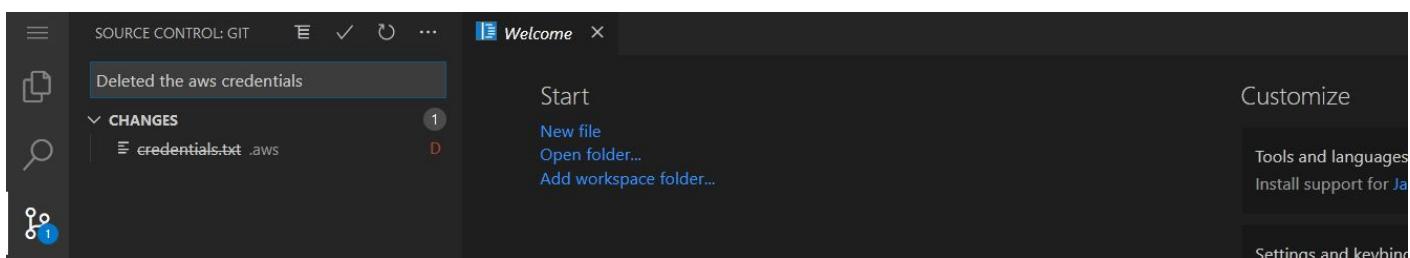
Select “Delete Permanently” to delete the AWS directory from the project.



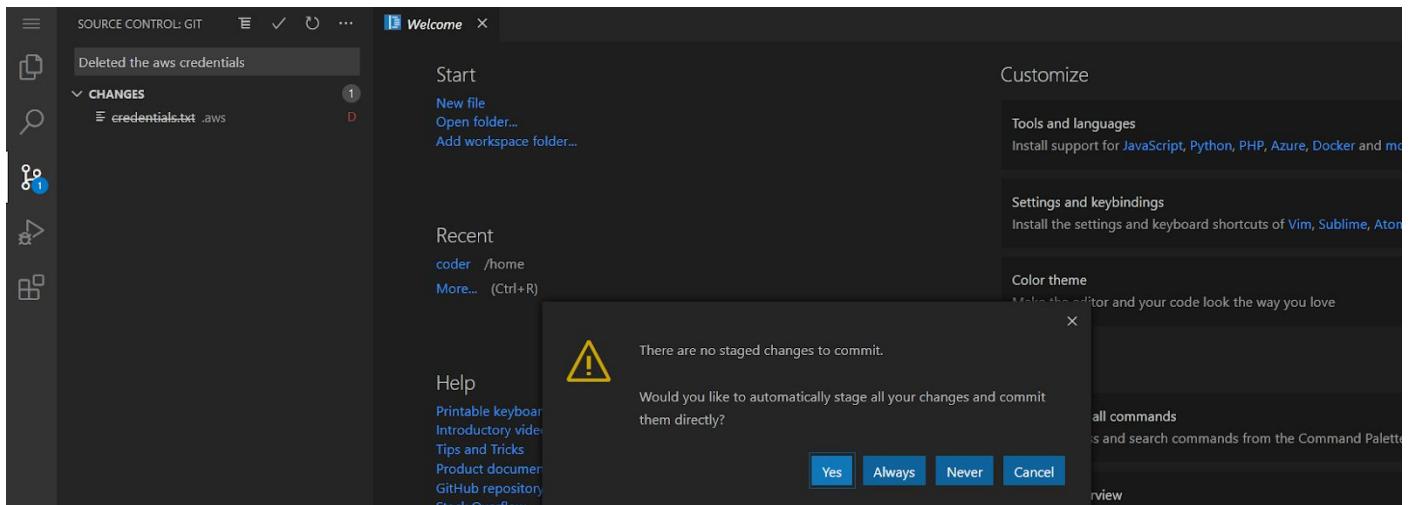
**Step 10:** Navigate to the Source Control section.



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

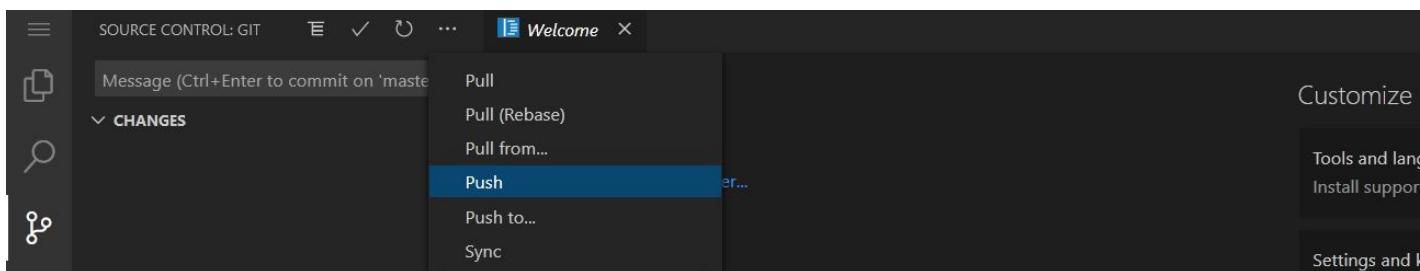


Click on the Commit button (Tick icon).

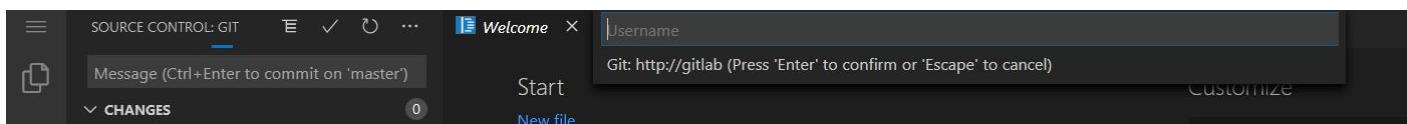


Choose the option “Always” to stage the commits always after the changes are made to the files.

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



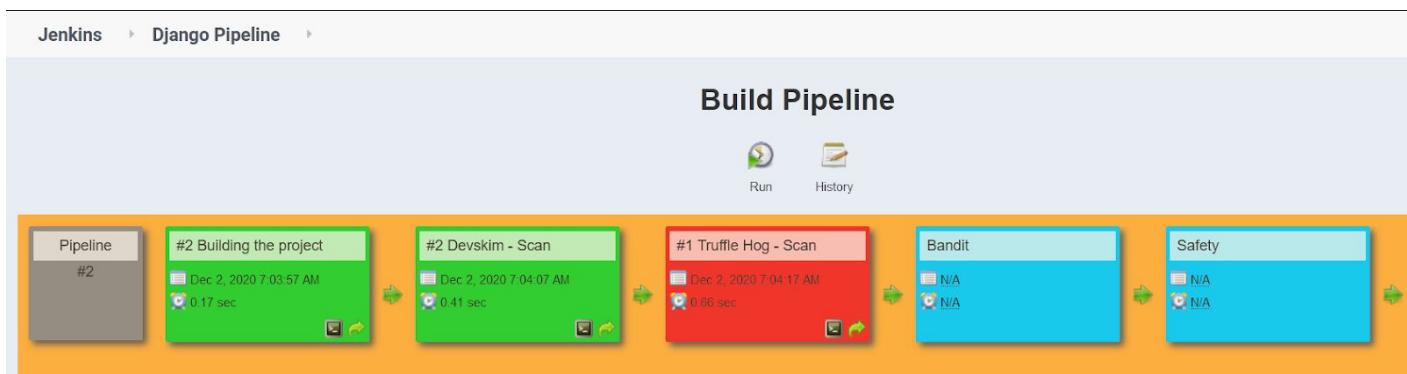
And select the Push button to push the changes to the remote repository.



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.

The screenshot shows a Jenkins build page for the 'Truffle Hog - Scan' job, specifically build #1. The page has a dark header with the Jenkins logo and a search bar. Below the header, the breadcrumb navigation shows 'Jenkins > Django Pipeline > Truffle Hog - Scan > #1'. On the left, there's a sidebar with links: 'Back to Project', 'Status', 'Changes', 'Console Output', 'View Build Information', and 'Git Build Data'. The main content area starts with a large red circle icon and the text 'Build #1 (Dec 2, 2020, 7:04:17 AM)'. It says 'No changes.' and 'Started by upstream project Devskim - Scan build number 2 originally caused by:'. It lists two items under 'originally caused by': 'Started by upstream project Building the project build number 2 originally caused by: Started by GitLab push by Administrator'. At the bottom, it shows 'Revision: 2ce372aeba5e525120d5825e9074c2eaef3eeabe refs/remotes/origin/master'.

**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": "4ce20a8b7da8d3ae7c9b13aaafdf6f29c040a1f7b",
  "date": "2020-12-15 13:37:52",
  "diff": "@@ -0,0 +1,3 @@\naws_access_key_id = 123456789012 \naws_secret_access_key = 'febcc175b9c0f1b6a831c399eadd2912697726617de'
\naws_session_token = 'AQoEXAMPLEH4aoAH0gNCAPyJxz4BlCFFxWNE1OPTgk5TthT+FvwqnKwRcOIfrrh3c/LTo6UDdyJwOovEVpvLXCr
/IvU1dYUg2RAVABanLiHb4IgRmpRV3zrkuWJ0gQs8IZzaIv2BXia2R40lgk'\n",
  "path": ".aws/credentials.txt",
  "printDiff": "@@ -0,0 +1,3 @@\naws_access_key_id = 123456789012 \naws_secret_access_key =
\u001b[93mfebcc175b9c0f1b6a831c399eadd2912697726617de\u001b[0m \naws_session_token =
'\u001b[93mAQoEXAMPLEH4aoAH0gNCAPyJxz4BlCFFxWNE1OPTgk5TthT+FvwqnKwRcOIfrrh3c/LTo6UDdyJwOovEVpvLXCr
/IvU1dYUg2RAVABanLiHb4IgRmpRV3zrkuWJ0gQs8IZzaIv2BXia2R40lgk'\n',
  "reason": "High Entropy",
  "stringsFound": [
    "febcc175b9c0f1b6a831c399eadd2912697726617de",
    "AQoEXAMPLEH4aoAH0gNCAPyJxz4BlCFFxWNE1OPTgk5TthT+FvwqnKwRcOIfrrh3c/LTo6UDdyJwOovEVpvLXCr
/IvU1dYUg2RAVABanLiHb4IgRmpRV3zrkuWJ0gQs8IZzaIv2BXia2R40lgk'\n",
  ]
}
{
  "branch": "origin/master",
  "commit": "ADD files\n",
  "commitHash": "2c9e88534ef80eae1369c79a505daa03c0e82592",
  "date": "2020-12-15 08:13:39",
  "diff": "@@ -0,0 +1,3 @@\naws_access_key_id = 123456789012 \naws_secret_access_key = 'febcc175b9c0f1b6a831c399eadd2912697726617de'
\naws_session_token = 'AQoEXAMPLEH4aoAH0gNCAPyJxz4BlCFFxWNE1OPTgk5TthT+FvwqnKwRcOIfrrh3c/LTo6UDdyJwOovEVpvLXCr
/IvU1dYUg2RAVABanLiHb4IgRmpRV3zrkuWJ0gQs8IZzaIv2BXia2R40lgk'\n",
  "path": ".aws/credentials.txt",
  "printDiff": "@@ -0,0 +1,3 @@\naws_access_key_id = 123456789012 \naws_secret_access_key =
'\u001b[93mfebcc175b9c0f1b6a831c399eadd2912697726617de\u001b[0m \naws_session_token =
'\u001b[93mAQoEXAMPLEH4aoAH0gNCAPyJxz4BlCFFxWNE1OPTgk5TthT+FvwqnKwRcOIfrrh3c/LTo6UDdyJwOovEVpvLXCr
/IvU1dYUg2RAVABanLiHb4IgRmpRV3zrkuWJ0gQs8IZzaIv2BXia2R40lgk'\n",
  "reason": "High Entropy",
  "stringsFound": [
    "febcc175b9c0f1b6a831c399eadd2912697726617de",
  ]
}
{
  "branch": "origin/master",
  "commit": "ADD files\n",
  "commitHash": "2c9e88534ef80eae1369c79a505daa03c0e82592",
  "date": "2020-12-15 08:13:39",
  "diff": "@@ -0,0 +1,3 @@\naws_access_key_id = 123456789012 \naws_secret_access_key = 'febcc175b9c0f1b6a831c399eadd2912697726617de'
\naws_session_token = 'AQoEXAMPLEH4aoAH0gNCAPyJxz4BlCFFxWNE1OPTgk5TthT+FvwqnKwRcOIfrrh3c/LTo6UDdyJwOovEVpvLXCr
/IvU1dYUg2RAVABanLiHb4IgRmpRV3zrkuWJ0gQs8IZzaIv2BXia2R40lgk'\n",
  "path": ".aws/credentials.txt",
  "printDiff": "@@ -0,0 +1,3 @@\naws_access_key_id = 123456789012 \naws_secret_access_key =
'\u001b[93mfebcc175b9c0f1b6a831c399eadd2912697726617de\u001b[0m \naws_session_token =
'\u001b[93mAQoEXAMPLEH4aoAH0gNCAPyJxz4BlCFFxWNE1OPTgk5TthT+FvwqnKwRcOIfrrh3c/LTo6UDdyJwOovEVpvLXCr
/IvU1dYUg2RAVABanLiHb4IgRmpRV3zrkuWJ0gQs8IZzaIv2BXia2R40lgk'\n",
  "reason": "High Entropy",
  "stringsFound": [
    "febcc175b9c0f1b6a831c399eadd2912697726617de",
    "AQoEXAMPLEH4aoAH0gNCAPyJxz4BlCFFxWNE1OPTgk5TthT+FvwqnKwRcOIfrrh3c/LTo6UDdyJwOovEVpvLXCr
/IvU1dYUg2RAVABanLiHb4IgRmpRV3zrkuWJ0gQs8IZzaIv2BXia2R40lgk"
  ]
}
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)

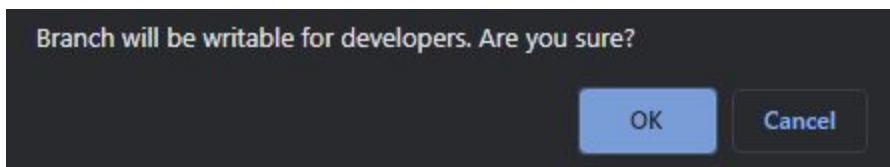
The screenshot shows the 'Repository Settings' page for the 'django-blog' project on GitLab. The left sidebar has a 'Settings' section selected, which includes 'General', 'Members', 'Integrations', and 'Repository'. The main content area shows sections for 'Default Branch', 'Mirroring repositories', 'Protected Branches', and 'Protected Tags'. The 'Protected Branches' section is expanded, showing the instruction 'Keep stable branches secure and force developers to use merge requests.' Below it, the 'Protected Tags' section is also expanded, showing the instruction 'Limit access to creating and updating tags.' Navigation icons at the bottom include back, forward, and search functions.

Expand the Protected Branches section.

The screenshot shows the GitLab web interface. On the left, there's a sidebar with various project management links: Issues (0), Merge Requests (0), CI / CD, Operations, Analytics, Wiki, Snippets, and Settings. Under Settings, the 'Repository' tab is selected. The main content area is titled 'Protect a branch'. It has three dropdown menus: 'Branch:' (set to 'Select branch or create wildcard'), 'Allowed to merge:' (set to 'Select'), and 'Allowed to push:' (set to 'Select'). Below these is a 'Protect' button. At the bottom, there's a summary table with columns 'Branch', 'Allowed to merge', and 'Allowed to push'. The 'Branch' row shows 'master' with a 'default' button. The 'Allowed to merge' and 'Allowed to push' rows both show 'Maintainers' with dropdown menus. An orange 'Unprotect' button is located to the right of the push settings.

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.

The screenshot shows the GitLab web interface for a project named 'django-blog'. On the left, there's a sidebar with various project management links like 'Project overview', 'Repository', 'Issues', 'Merge Requests', 'CI / CD', 'Operations', 'Analytics', 'Wiki', 'Snippets', and 'Settings'. The 'Settings' link is currently selected. The main content area is titled 'Protect a branch' and contains three dropdown menus: 'Branch:' (set to 'Select branch or create wildcard'), 'Allowed to merge:' (set to 'Select'), and 'Allowed to push:' (set to 'Select'). Below these is a large 'Protect' button. A note at the bottom states 'Protected branch (0)' and 'There are currently no protected branches, protect a branch with the form above.'

**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
```

**FLAG 2:** cb4f2a3413adc795a5088dd7e7e01a2f

## Bandit Issue

**Step 1:** Click on the ‘Bandit’ to check the job build page.

Jenkins > Django Pipeline > Bandit > #2

[Back to Project](#)

[Status](#)

[Changes](#)

[Console Output](#)

[View Build Information](#)

[Git Build Data](#)

[Previous Build](#)

**Build #2 (Dec 2, 2020, 8:19:48 AM)**

[Changes](#)

1. added base.html ([details](#))

[Started by upstream project Truffle Hog - Scan build number 3 originally caused by:](#)

- Started by upstream project Devskim - Scan build number 4 originally caused by:
  - Started by upstream project Building the project build number 4 originally caused by:
    - Started by GitLab push by Administrator

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

```
Jenkins > Django Pipeline > Bandit > #2
[main] INFO    running on python 3.6.2
Run started:2020-12-02 08:20:08.968740

Test results:
>> Issue: [B605:start_process_with_a_shell] Starting a process with a shell, possible injection detected, security issue.
Severity: High   Confidence: High
Location: ./web/urls.py:42
More Info: https://bandit.readthedocs.io/en/latest/plugins/b605_start_process_with_a_shell.html
41      log = "echo '{}' > output.debug".format(urlpatterns)
42      os.system(log)
43

-----
Code scanned:
Total lines of code: 502
Total lines skipped (#nosec): 0

Run metrics:
Total issues (by severity):
Undefined: 0.0
Low: 0.0
Medium: 0.0
High: 1.0
Total issues (by confidence):
Undefined: 0.0
Low: 0.0
Medium: 0.0
```

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

The screenshot shows the ArcherySec web interface. At the top, there's a navigation bar with the logo 'ARCHERY a security tool' and a 'Create New' dropdown. Below it is a sidebar with links: 'Dashboard', 'Pentest Activity', 'Launch Scans', 'Dynamic Scans', 'Infrastructure Scans', and 'Static Scan'. The main content area is titled 'Project List' and displays a table with one row. The table has columns for 'Project', 'Start Date', 'End Date', and 'Owner'. The data in the table is as follows:

Project	Start Date	End Date	Owner
DevSecOps	2020-12-02	2020-12-02	test_project

At the bottom of the table, it says 'Showing 1 to 1 of 1 entries'. The entire interface has a dark theme with red and blue highlights.

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

#### Scanners List

Show 10 entries

Search

Scanner	High	Medium	Low
Bandit	1	0	0

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 screenshot shows the ARCHERY security tool interface. On the left, there's a sidebar with navigation links: Dashboard, Pentest Activity, Launch Scans, Dynamic Scans, Infrastructure Scans, Static Scan, and Compliance Scans. The main area has a header "Create New" and a title "start\_process\_with\_a\_shell". Below the title are three expandable sections: "Description", "Instance", and "Solutions". The "Instance" section shows a file named "urls.py" with the following code snippet:

```
File: ./web/urls.py
41     log = "echo '{}' > output.debug".format(urlpatterns)
42     os.system(log)
43
```

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.

```

EXPLORER
OPEN EDITORS
urls.py web
DJANGO-BLOG
blog
api
migrations
templates / blog
_init_.py
admin.py
apps.py
forms.py
models.py
tests.py
urls.py
views.py
media
screenshots
users
web
static
_init_.py
settings.py
urls.py
wsgi.py
db.sqlite3

urls.py
20     # Reset Password URLs
21     path('password-reset/', auth_views.PasswordResetView.as_view(
22         template_name='users/password_reset.html'), name='password_reset'),
23     path('password-reset/', auth_views.PasswordResetView.as_view(
24         template_name='users/password_reset.html'), name='password_reset'),
25     path('password-reset/done/', auth_views.PasswordResetDoneView.as_view(
26         template_name='users/password_reset_done.html'), name='password_reset_done'),
27     path('password-reset-confirm///', auth_views.PasswordResetConfirmView.as_view(
28         template_name='users/password_reset_confirm.html'), name='password_reset_confirm'),
29     path('password-reset-complete/', auth_views.PasswordResetCompleteView.as_view(
30         template_name='users/password_reset_complete.html'), name='password_reset_complete'),
31     # Change Password URLs
32     path('password-change/', auth_views.PasswordChangeView.as_view(
33         template_name='users/password_change.html'), name='password_change'),
34     path('password-change/done/', auth_views.PasswordChangeDoneView.as_view(
35         template_name='users/password_change_done.html'), name='password_change_done'),
36 ]
37
38 if settings.DEBUG:
39     urlpatterns += static(settings.MEDIA_URL,
40                           document_root=settings.MEDIA_ROOT)
41     log = "echo '{}' > output.debug".format(urlpatterns)
42     os.system(log)
43
44 path('', include('blog.urls')),
45

```

Remove the logging implementation from the file.

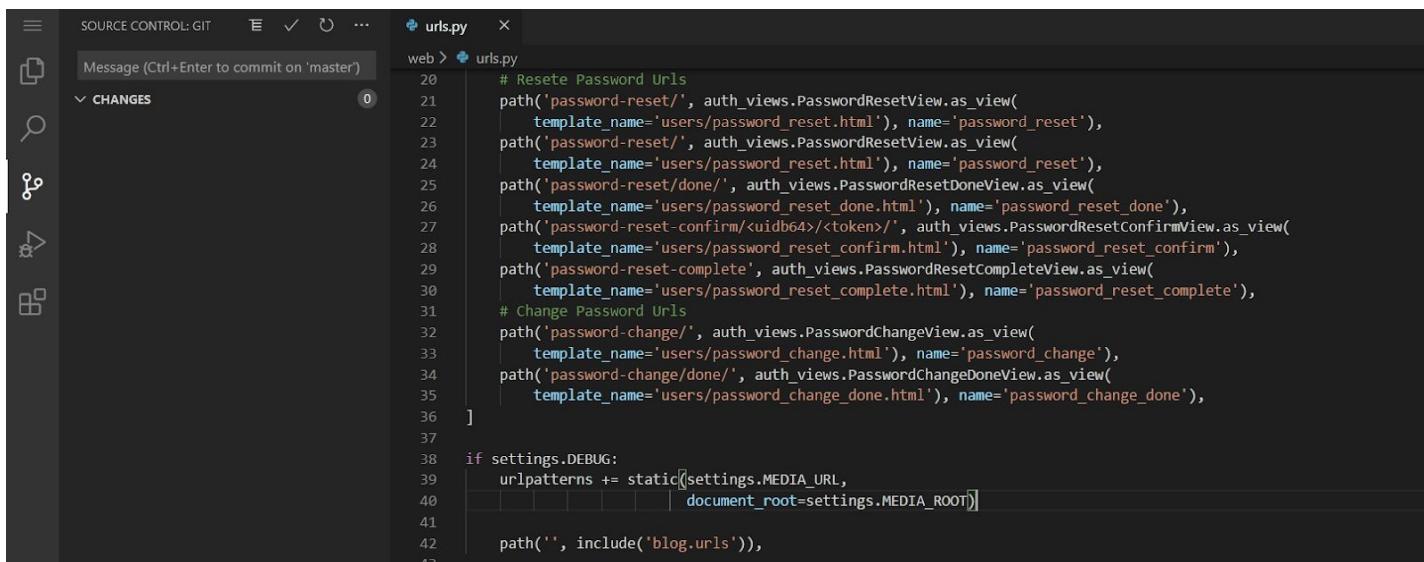
```

EXPLORER
OPEN EDITORS
urls.py web
DJANGO-BLOG
blog
api
migrations
templates / blog
_init_.py
admin.py
apps.py
forms.py
models.py
tests.py
urls.py
views.py
media
screenshots
users
web
static

urls.py
20     # Reset Password URLs
21     path('password-reset/', auth_views.PasswordResetView.as_view(
22         template_name='users/password_reset.html'), name='password_reset'),
23     path('password-reset/', auth_views.PasswordResetView.as_view(
24         template_name='users/password_reset.html'), name='password_reset'),
25     path('password-reset/done/', auth_views.PasswordResetDoneView.as_view(
26         template_name='users/password_reset_done.html'), name='password_reset_done'),
27     path('password-reset-confirm///', auth_views.PasswordResetConfirmView.as_view(
28         template_name='users/password_reset_confirm.html'), name='password_reset_confirm'),
29     path('password-reset-complete/', auth_views.PasswordResetCompleteView.as_view(
30         template_name='users/password_reset_complete.html'), name='password_reset_complete'),
31     # Change Password URLs
32     path('password-change/', auth_views.PasswordChangeView.as_view(
33         template_name='users/password_change.html'), name='password_change'),
34     path('password-change/done/', auth_views.PasswordChangeDoneView.as_view(
35         template_name='users/password_change_done.html'), name='password_change_done'),
36 ]
37
38 if settings.DEBUG:
39     urlpatterns += static(settings.MEDIA_URL,
40                           document_root=settings.MEDIA_ROOT)
41

```

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



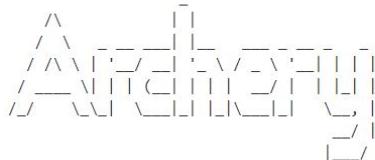
```
 20     # Reset Password Urls
 21     path('password-reset/', auth_views.PasswordResetView.as_view(
 22         template_name='users/password_reset.html'), name='password_reset'),
 23     path('password-reset/', auth_views.PasswordResetView.as_view(
 24         template_name='users/password_reset.html'), name='password_reset'),
 25     path('password-reset/done/', auth_views.PasswordResetDoneView.as_view(
 26         template_name='users/password_reset_done.html'), name='password_reset_done'),
 27     path('password-reset-confirm/{uidb64}/{token}/', auth_views.PasswordResetConfirmView.as_view(
 28         template_name='users/password_reset_confirm.html'), name='password_reset_confirm'),
 29     path('password-reset-complete', auth_views.PasswordResetCompleteView.as_view(
 30         template_name='users/password_reset_complete.html'), name='password_reset_complete'),
 31     # Change Password Urls
 32     path('password-change/', auth_views.PasswordChangeView.as_view(
 33         template_name='users/password_change.html'), name='password_change'),
 34     path('password-change/done/', auth_views.PasswordChangeDoneView.as_view(
 35         template_name='users/password_change_done.html'), name='password_change_done'),
 36 ]
 37
 38 if settings.DEBUG:
 39     urlpatterns += static(settings.MEDIA_URL,
 40                           document_root=settings.MEDIA_ROOT)
 41
 42     path('', include('blog.urls')),
```

## 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.

Jenkins > Django Pipeline > Bandit > #2

**Build #2 (Dec 2, 2020, 8:19:48 AM)**

**Changes**

1. added base.html ([details](#))

**Started by upstream project [Truffle Hog - Scan](#) build number 3**  
originally caused by:

- Started by upstream project [Devskim - Scan](#) build number 4  
originally caused by:
  - Started by upstream project [Building the project](#) build number 4  
originally caused by:
    - Started by GitLab push by Administrator

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

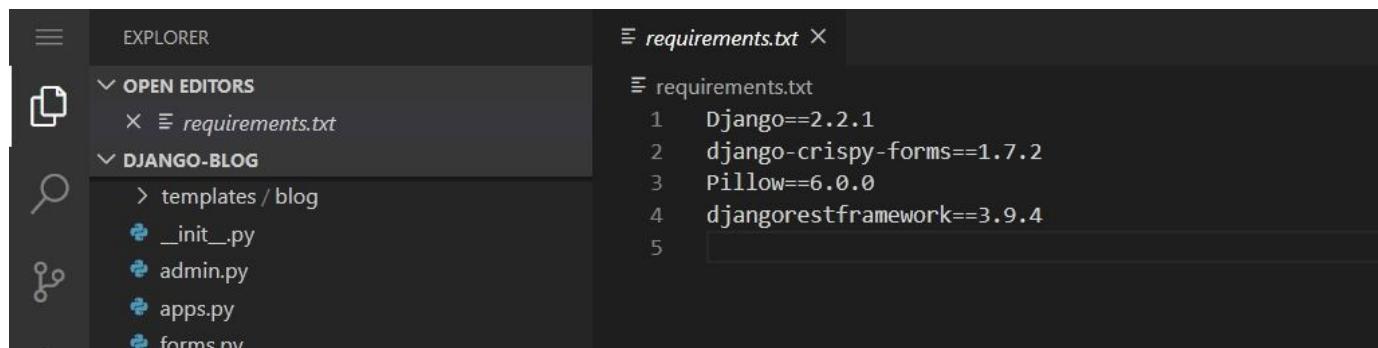
```

| 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)



```

EXPLORER
OPEN EDITORS
requirements.txt
DJANGO-BLOG
templates/blog
__init__.py
admin.py
apps.py
forms.py

requirements.txt
1 Django==2.2.1
2 django-crispy-forms==1.7.2
3 Pillow==6.0.0
4 djangorestframework==3.9.4
5

```

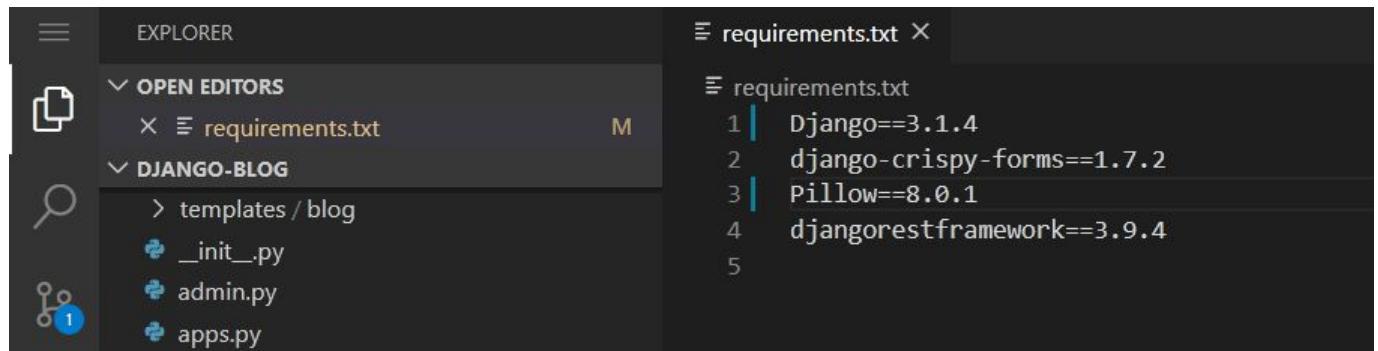
In order to fix the issue, Modify the version numbers from the requirements.txt to the latest in order to resolve the issue.

FROM:  
Django==2.2.1  
Pillow==6.0.0

TO:

Django==3.1.4

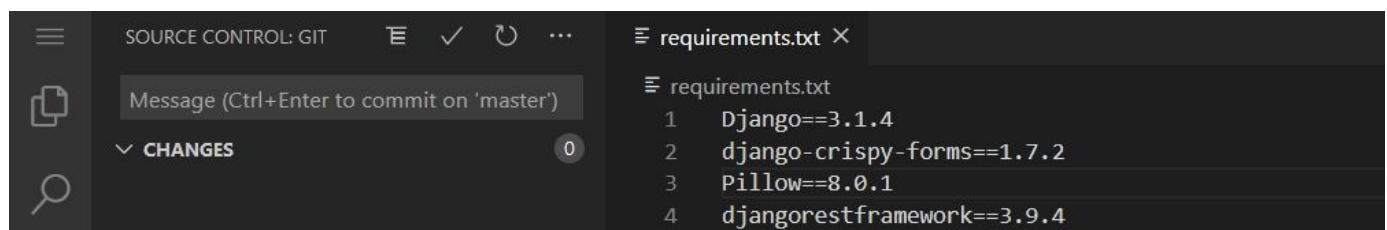
Pillow==8.0.1



The screenshot shows the VS Code interface. On the left is the Explorer sidebar with a tree view. Under 'OPEN EDITORS', there is a file named 'requirements.txt'. Under the 'Django-Blog' application, there are files for templates, \_\_init\_\_.py, admin.py, and apps.py. On the right is the main code editor window titled 'requirements.txt'. It contains the following text:

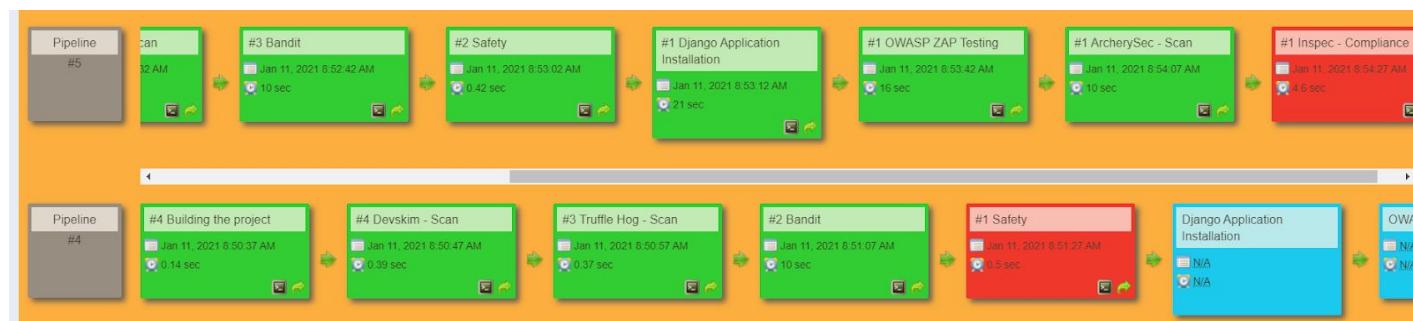
```
Django==3.1.4
django-crispy-forms==1.7.2
Pillow==8.0.1
djangorestframework==3.9.4
```

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



The screenshot shows the VS Code interface with the Source Control: Git extension open. A message box in the center says 'Message (Ctrl+Enter to commit on 'master')'. Below it, under 'CHANGES', there is a list of staged files: requirements.txt, Django==3.1.4, django-crispy-forms==1.7.2, Pillow==8.0.1, and djangorestframework==3.9.4.

### Step 5: Check the pipeline to see any changes.

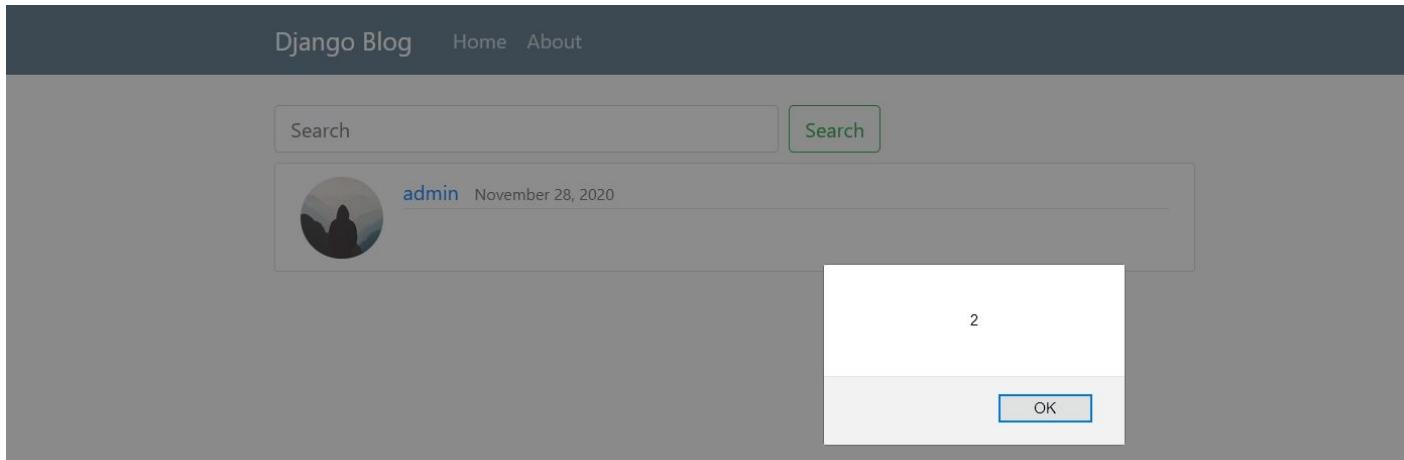


The Safety stage has passed successfully. The logs can be checked from clicking on the "Safety" and opening the console output.

**FLAG 4:** 46d5b8115e47963392f66c5b6941c2e0

## Cross site scripting and Inspec Issue

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



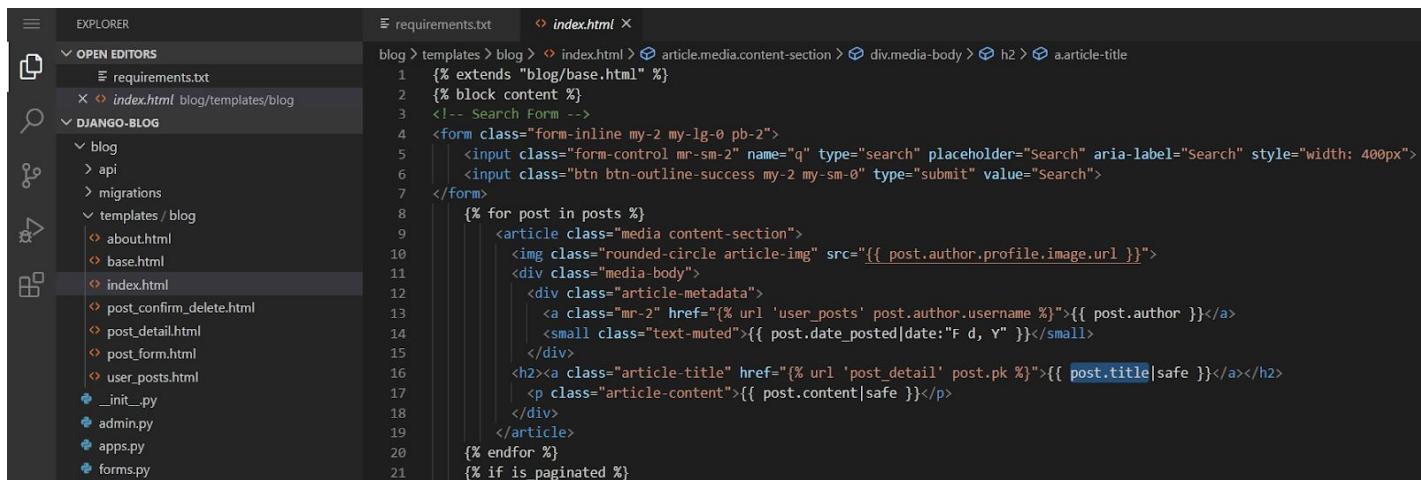
The XSS exists in the application.

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

```
68      <article class="media content-section">
69          
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 screenshot shows the VS Code interface with the 'EXPLORER' view on the left and the 'index.html' file open in the center. The file content is as follows:

```
blog > templates > blog > index.html
1  {% extends "blog/base.html" %}
2  {% block content %}
3  <!-- Search Form -->
4  <form class="form-inline my-2 my-lg-0 pb-2">
5      <input class="form-control mr-sm-2" name="q" type="search" placeholder="Search" aria-label="Search" style="width: 400px">
6      <input class="btn btn-outline-success my-2 my-sm-0" type="submit" value="Search">
7  </form>
8  {% for post in posts %}
9      <article class="media content-section">
10         
11         <div class="media-body">
12             <div class="article-metadata">
13                 <a class="mr-2" href="{% url 'user_posts' post.author.username %}">{{ post.author }}</a>
14                 <small class="text-muted">{{ post.date_posted|date:"F d, Y" }}</small>
15             </div>
16             <h2><a class="article-title" href="{% url 'post_detail' post.pk %}">{{ post.title|safe }}</a></h2>
17             <p class="article-content">{{ post.content|safe }}</p>
18         </div>
19     </article>
20  {% endfor %}
21  {% if is_paginated %}
```

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.



The screenshot shows a code editor interface with two tabs: "requirements.txt" and "index.html". The "index.html" tab is active, displaying Django template code. The code includes HTML structure like forms and articles, and uses Jinja2 syntax with variables like {{ post }} and {{ post.title }}.

```

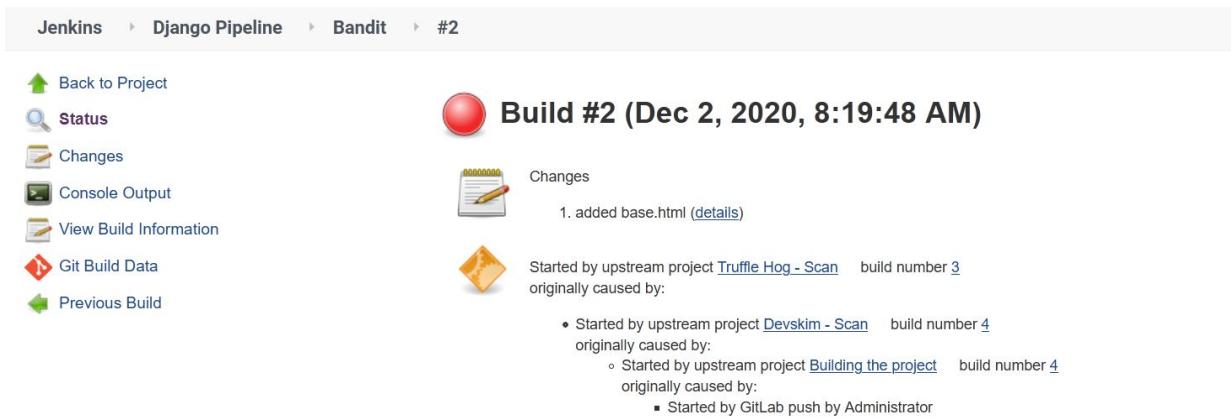
EXPLORER
OPEN EDITORS
requirements.txt
index.html blog/templates/blog
DJANGO-BLOG
blog
api
migrations
templates / blog
about.html
base.html
index.html
post_confirm_delete.html
post_detail.html
post_form.html
user_posts.html
_init_.py

requirements.txt
index.html
index.html

blog > templates > blog > index.html > article.media.content-section > div.media-body > h2 > a.article-title
1  {% extends "blog/base.html" %}
2  {% block content %}
3  <!-- Search Form -->
4  <form class="form-inline my-2 my-lg-0 pb-2">
5    <input class="form-control mr-sm-2" name="q" type="search" placeholder="Search" aria-label="Search" style="width: 150px;">
6    <input class="btn btn-outline-success my-2 my-sm-0" type="submit" value="Search">
7  </form>
8  {% for post in posts %}
9    <article class="media content-section">
10      
12        <div class="article-metadata">
13          <a class="mr-2" href="{% url 'user_posts' post.author.username %}">{{ post.author }}</a>
14          <small class="text-muted">{{ post.date_posted|date:"F d, Y" }}</small>
15        </div>
16        <h2><a class="article-title" href="{% url 'post_detail' post.pk %}">{{ post.title }}</a></h2>
17        <p class="article-content">{{ post.content|safe }}</p>

```

**Step 5:** Click on the ‘Inspec’ to check the job build page.



The Jenkins build page for "Django Pipeline" shows a successful build (#2) from December 2, 2020, at 8:19:48 AM. The page includes links for "Back to Project", "Status", "Changes", "Console Output", "View Build Information", "Git Build Data", and "Previous Build". The "Changes" section lists "1. added base.html". The "Console Output" section shows the build log, which includes the message "Started by upstream project Truffle Hog - Scan build number 3 originally caused by:". The log then traces back through multiple upstream projects and a GitLab push.

**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

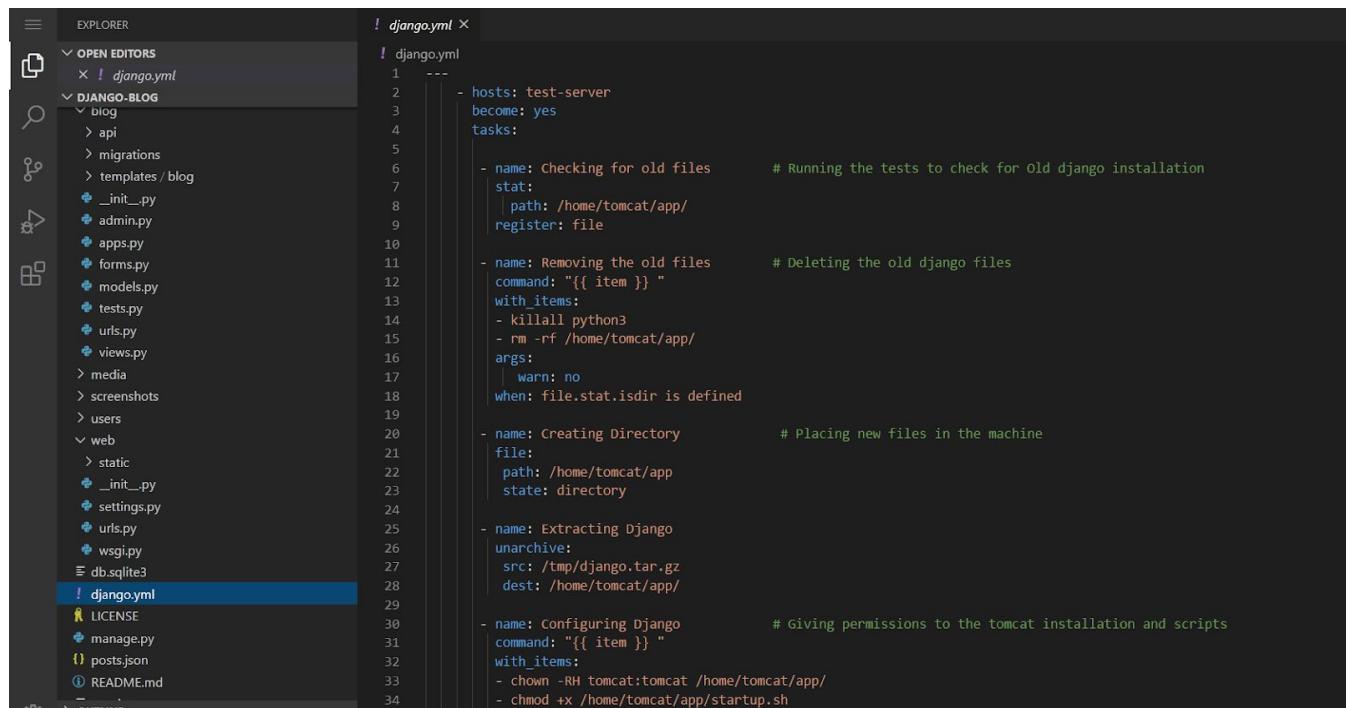
(compared using `cmp` matcher)
[0m

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.



```

! django.yml
! django.yml
1  ---
2
3  - hosts: test-server
4    become: yes
5    tasks:
6
7      - name: Checking for old files          # Running the tests to check for old django installation
8        stat:
9          path: /home/tomcat/app
10         register: file
11
12      - name: Removing the old files        # Deleting the old django files
13        command: "{{ item }}"
14        with_items:
15          - killall python3
16          - rm -rf /home/tomcat/app/
17          args:
18            warn: no
19          when: file.stat.isdir is defined
20
21      - name: Creating Directory           # Placing new files in the machine
22        file:
23          path: /home/tomcat/app
24          state: directory
25
26      - name: Extracting Django
27        unarchive:
28          src: /tmp/django.tar.gz
29          dest: /home/tomcat/app/
30
31      - name: Configuring Django          # Giving permissions to the tomcat installation and scripts
32        command: "{{ item }}"
33        with_items:
34          - chown -R tomcat:tomcat /home/tomcat/app/
35          - chmod +x /home/tomcat/app/startup.sh

```

**Step 8:** Place command to set the appropriate permissions on the 'app' directory.

Add the following after line 34.

```
- chmod 0750 /home/tomcat/app/
```

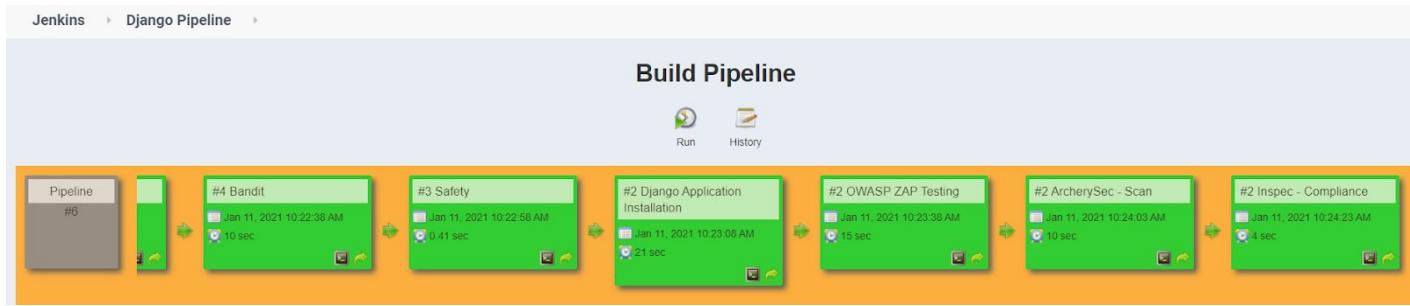
```
! django.yml ×
! django.yml
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
# Placing new files in the machine
- name: Creating directory
  file:
    path: /home/tomcat/app
  state: directory
- name: Extracting Django
  unarchive:
    src: /tmp/django.tar.gz
    dest: /home/tomcat/app/
- name: Configuring Django
  command: "{{ item }}"
  with_items:
    - chown -RH tomcat:tomcat /home/tomcat/app/
    - chmod +x /home/tomcat/app/startup.sh
    - chmod 0750 /home/tomcat/app/
# Giving permissions to the tomcat instance
```

Commit the changes and push the files to the remote repository.

(Same as Step 10 onwards mentioned in 'Devskim Issue' section)

```
! django.yml ×
! django.yml
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
# Placing new files in the machine
- name: Creating directory
  file:
    path: /home/tomcat/app
  state: directory
- name: Extracting Django
  unarchive:
    src: /tmp/django.tar.gz
    dest: /home/tomcat/app/
- name: Configuring Django
  command: "{{ item }}"
  with_items:
    - chown -RH tomcat:tomcat /home/tomcat/app/
    - chmod +x /home/tomcat/app/startup.sh
    - chmod 0750 /home/tomcat/app/
# Giving permissions to the tomcat instance
```

**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    tomcat.directories: Check for existence and correct permissions of application directory[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;41m        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