

CLOUD AND NETWORK SECURITY

CS-CNS04-23040

ASSIGNMENT 1 WEEK 10

MICROSOFT DEVOPS

NICOLE JEPKEMOI KIPKORIR

Introduction

Integrating static analysis tools into the development lifecycle
(Defender for DevOps)

In this lab, you will configure Defender for DevOps. Microsoft Security DevOps is a command line application that integrates static analysis tools into the development lifecycle. Microsoft Security DevOps installs, configures, and runs the latest versions of static analysis tools (including, but not limited to, SDL/security and compliance tools).

The Microsoft Security DevOps uses the following Open Source tools: Bandit, BinSkim, ESLint, Credscan, Template Analyzer, Terrascan and Trivy.

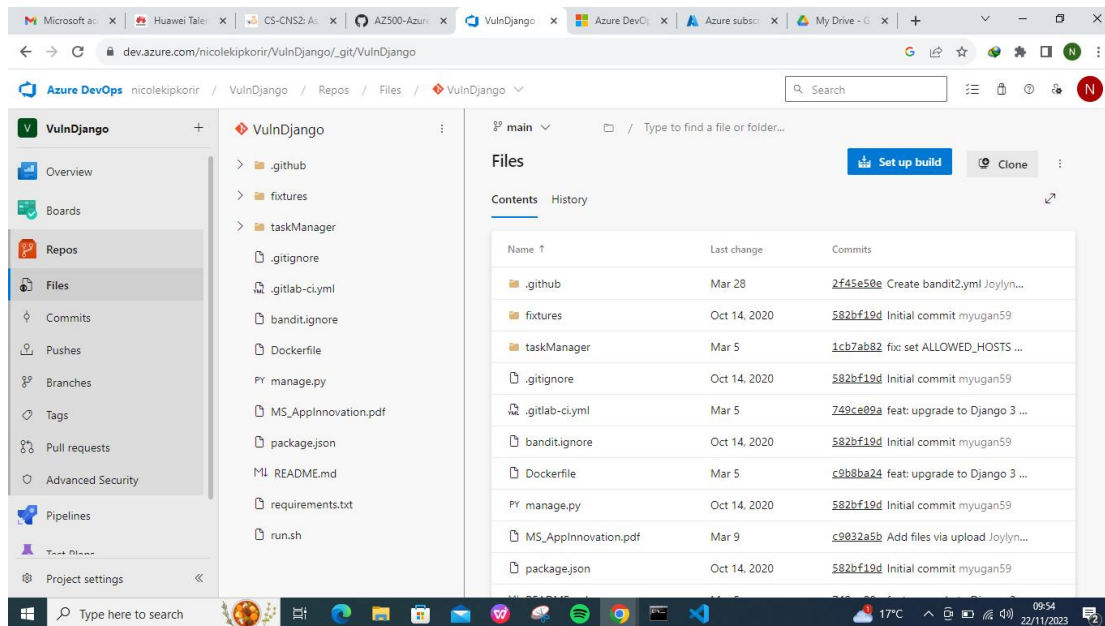
Objectives

After completing this lab, you will be able to:

- SAST scan using Bandit locally.
- Configure the Microsoft Security DevOps Azure DevOps extension.
- Configure the Microsoft Security DevOps GitHub actions
- DevOps Security Workbook

Exercise 1: Import the vulnerable code

1. On GitHub, fork the vulnerable code from S2FrdQ /VulnDjango (github.com)
2. On Azure DevOps, create a new Private project and name it VulnDjango. Navigate to Repos, and import the vulnerable code from [https://github.com/S2FrdQ /VulnDjango.git](https://github.com/S2FrdQ/VulnDjango.git)



Exercise 2: SAST scan using Bandit locally.

Bandit - The Bandit is a tool designed to find common security issues in Python code. To do this Bandit, processes each file, builds an AST, and runs appropriate plugins against the AST nodes. Once Bandit has finished scanning all the files it generates a report. Bandit was originally developed within the OpenStack Security Project and later rehomed to PyCQA.

- i. Download the source code locally – git clone
<https://github.com/S2FrdQ/VulnDjango> webappdjango then cd webappdjango
- ii. Install Bandit pip3 install bandit
- iii. If a warning is issued to add Directory to path, add using the below command.

```
export PATH="/home/kali/.local/bin:$PATH"
```

To explore bandit --help

```
MINGW64/c/Users/USER/webappdjango
Downloading bandit-1.7.5-py3-none-any.whl (123 kB)
123.3/123.3 kB 314.8 kB/s eta 0:00:00
Collecting GitPython==1.0.1 (from bandit)
Downloading GitPython-3.1.40-py3-none-any.whl.metadata (12 kB)
Requirement already satisfied: PyYAML>=5.3.1 in c:\program files\python312\lib\site-packages (from bandit) (6.0.1)
Collecting stevedore==1.20.0 (from bandit)
Downloading stevedore-1.20.0-py3-none-any.whl.metadata (2.2 kB)
Collecting rich (from bandit)
Downloading rich-13.7.0-py3-none-any.whl.metadata (18 kB)
Requirement already satisfied: colorama>=0.3.9 in c:\program files\python312\lib\site-packages (from bandit) (0.4.6)
Collecting gitdb<=4.0.11-py3-none-any.whl (190 kB)
Downloading gitdb-4.0.11-py3-none-any.whl.metadata (1.2 kB)
Collecting pbr==2.1.0, >=2.0.0 (from stevedore==1.20.0->bandit)
Downloading pbr-6.0.0-py2.py3-none-any.whl.metadata (1.3 kB)
Collecting markdown-it-py<=3.0.0-py3-none-any.whl (6.9 kB)
Requirement already satisfied: pygments<3.0.0, >=2.13.0 in c:\program files\python312\lib\site-packages (from rich->bandit) (2.17.1)
Collecting smmap<=5.0.1-py3-none-any.whl (4.3 kB)
Downloading smmap-5.0.1-py3-none-any.whl.metadata (4.3 kB)
Collecting mdurl<=0.1 (from markdown-it-py<=3.0.0->rich->bandit)
Downloading mdurl-0.1.2-py3-none-any.whl (10.0 kB)
Downloading GitPython-3.1.40-py3-none-any.whl (190 kB)
----- 190.6/190.6 kB 959.2 kB/s eta 0:00:00
Downloading stevedore-1.20.0-py3-none-any.whl (49 kB)
----- 49.6/49.6 kB 168.4 kB/s eta 0:00:00
Downloading rich-13.7.0-py3-none-any.whl (240 kB)
----- 240.6/240.6 kB 1.2 MB/s eta 0:00:00
Downloading gitdb-4.0.11-py3-none-any.whl (82 kB)
----- 82.7/82.7 kB 305.1 kB/s eta 0:00:00
Downloading markdown-it-py-3.0.0-py3-none-any.whl (87 kB)
----- 87.5/87.5 kB 618.3 kB/s eta 0:00:00
Downloading pbr-6.0.0-py2.py3-none-any.whl (107 kB)
----- 107.5/107.5 kB 57.1 kB/s eta 0:00:00
Downloading smmap-5.0.1-py3-none-any.whl (24 kB)
Installing collected packages: smmap, pbr, mdurl, stevedore, markdown-it-py, gitdb, rich, GitPython, bandit
WARNING: The script pbr.exe is installed in 'C:\Users\USER\AppData\Roaming\Python\Python312\Scripts' which is not on PATH.
Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
WARNING: The script markdown-it.exe is installed in 'C:\Users\USER\AppData\Roaming\Python\Python312\Scripts' which is not on PATH.
Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
WARNING: The scripts bandit-baseline.exe, bandit-config-generator.exe and bandit.exe are installed in 'C:\Users\USER\AppData\Roaming\Python\Python312\Scripts' which is not on PATH.
Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
Successfully installed GitPython-3.1.40 bandit-1.7.5 gitdb-4.0.11 markdown-it-py-3.0.0 mdurl-0.1.2 pbr-6.0.0 rich-13.7.0 smmap-5.0.1 stevedore-1.20.0

USERDESKTOP-H40TQ16 MINGW64 ~\webappdjango (main)
$ export PATH="/c/Users/USER/.local/bin:$PATH"

USERDESKTOP-H40TQ16 MINGW64 ~\webappdjango (main)
$
```

iv.Run the scanner - We are using the tee command here to show the output and store it in a file simultaneously. bandit -r . Basic scan

bandit -r . -f json | tee bandit-output.json

```
MINGW64/c/Users/USER/webappdjango
[notice] To update, run: python.exe -m pip install --upgrade pip

USERDESKTOP-H40TQ16 MINGW64 ~\webappdjango (main)
$ bandit -r . -f json | tee bandit-output.json
[main] INFO profile include tests: None
[main] INFO cli include tests: None
[main] INFO cli exclude tests: None
{
  "errors": [],
  "generated_at": "2023-11-22T10:28:39Z",
  "metrics": {
    ".\\manage.py": {
      "CONFIDENCE.HIGH": 0,
      "CONFIDENCE.LOW": 0,
      "CONFIDENCE.MEDIUM": 0,
      "CONFIDENCE.UNDEFINED": 0,
      "SEVERITY.HIGH": 0,
      "SEVERITY.LOW": 0,
      "SEVERITY.MEDIUM": 0,
      "SEVERITY.UNDEFINED": 0,
      "loc": 6,
      "nosec": 0,
      "skipped_tests": 0
    },
    ".\\taskManager\\__init__.py": {
      "CONFIDENCE.HIGH": 0,
      "CONFIDENCE.LOW": 0,
      "CONFIDENCE.MEDIUM": 0,
      "CONFIDENCE.UNDEFINED": 0,
      "SEVERITY.HIGH": 0,
      "SEVERITY.LOW": 0,
      "SEVERITY.MEDIUM": 0,
      "SEVERITY.UNDEFINED": 0,
      "loc": 0,
      "nosec": 0,
      "skipped_tests": 0
    },
    ".\\taskManager\\forms.py": {
      "CONFIDENCE.HIGH": 0,
      "CONFIDENCE.LOW": 0,
      "CONFIDENCE.MEDIUM": 0,
      "CONFIDENCE.UNDEFINED": 0,
      "SEVERITY.HIGH": 0,
      "SEVERITY.LOW": 0,
      "SEVERITY.MEDIUM": 0,
      "SEVERITY.UNDEFINED": 0,
      "loc": 56,
      "nosec": 0,
      "skipped_tests": 0
    }
  }
}
```

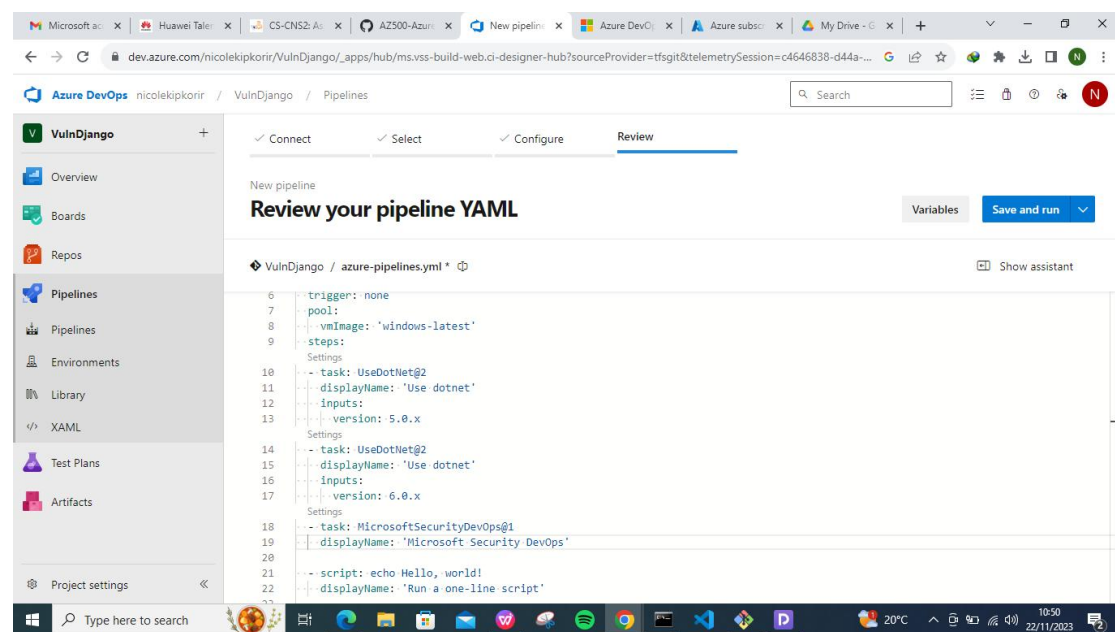
Exercise 3: Configure the Microsoft Security DevOps Azure DevOps extension.

Note: Admin privileges to the Azure DevOps organization are required to install the extension.

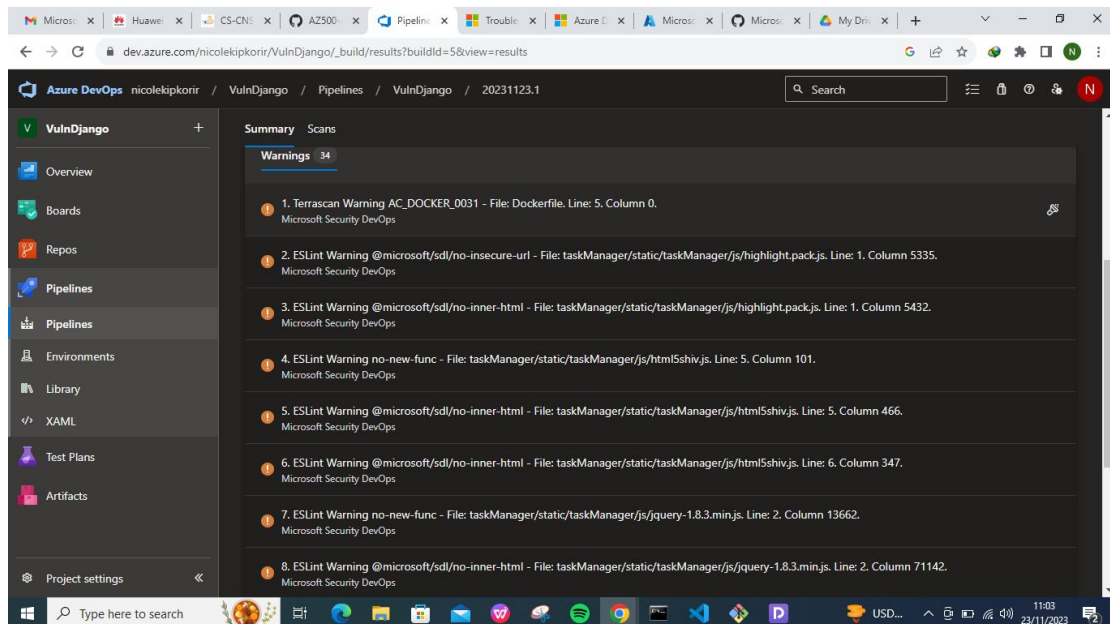
1. Activate Microsoft Security DevOps extension – on your Azure DevOps portal with

the VulnDjango project open, click on the marketplace icon > Browse Marketplace.

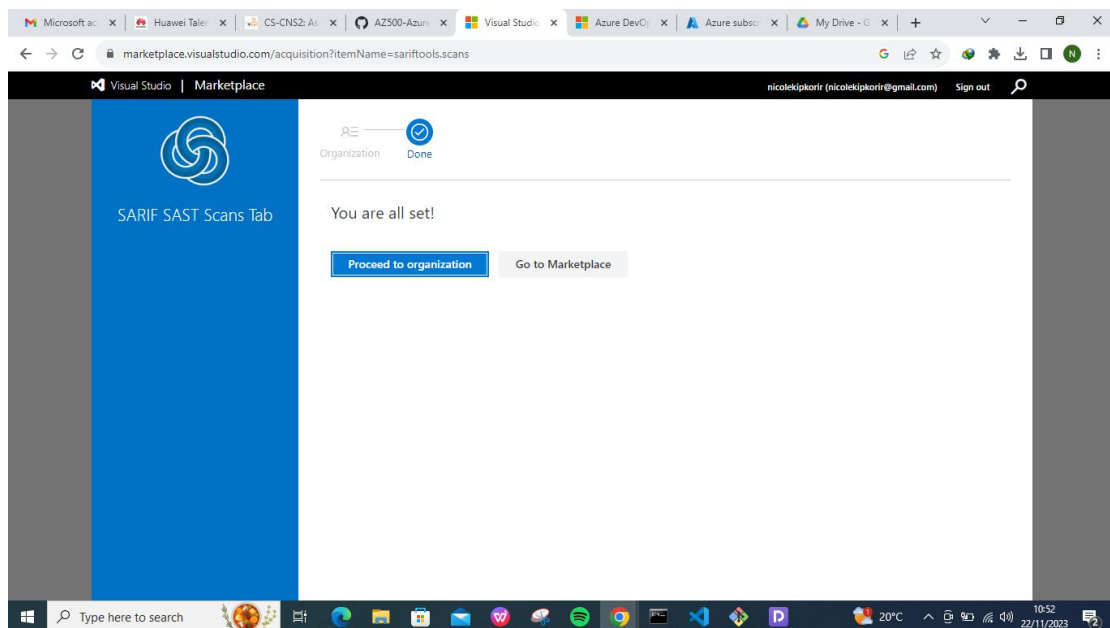
2. On the Marketplace, search for Microsoft Security DevOps and open it.
3. On the Microsoft Security DevOps page, click on Get it for free.
4. On the next page, select the desired Azure DevOps organization and Install. Proceed to organization once installed.
5. Navigate to your VulnDjango project, then Pipelines and Click New pipeline.
6. On the Where is your code? window, select Azure Repos Git (YAML) and select the VulnDjango repository.
7. On Add the following scripts as in into the yaml file.



8. Click Save and run and let the pipeline run. You can check progress by going to Pipeline- Pipelines and select the running pipeline.
9. When done, you can view security vulnerabilities found by Microsoft Security DevOps , by clicking Scans.



Note: Install the SARIF SAST Scans Tab extension on the Azure DevOps organization in order to ensure that the generated analysis results will be displayed automatically under the Scans tab.



```
Administrator: Windows PowerShell

Mode                LastWriteTime         Length Name
----                -
d-----          25/11/2023    17:17         agent

PS C:\USERS\USER> cd agent
PS C:\USERS\USER\agent> .\config.cmd

Azure Pipelines
agent v3.230.0 (commit 38c1c98)

>> Connect:
Enter server URL > https://dev.azure.com/nicolekipkorir/
Enter authentication type (press enter for PAT) >
Enter personal access token > *****
Connecting to server ...

>> Register Agent:
Enter agent pool (press enter for default) >
Enter agent name (press enter for DESKTOP-H40TQ38) >
Scanning for tool capabilities.
Connecting to the server.
Pool Default already contains an agent with name DESKTOP-H40TQ38.
Enter replace? (Y/N) (press enter for N) > y
Successfully replaced the agent
Testing agent connection.
Enter work folder (press enter for work) > VulnDjango
2023-11-25 17:20:31Z: Settings Saved.
Enter run agent as service? (Y/N) (press enter for N) > y
Enter enable SERVICE_SID_TYPE_UNRESTRICTED for agent service (Y/N) (press enter for N) > y
Enter User account to use for the service (press enter for NT AUTHORITY\NETWORK SERVICE) >
Granting file permissions to 'NT AUTHORITY\NETWORK SERVICE'.
Service vstsagent.nicolekipkorir.Default.DESKTOP-H40TQ38 successfully installed
Service vstsagent.nicolekipkorir.Default.DESKTOP-H40TQ38 successfully set recovery option
Service vstsagent.nicolekipkorir.Default.DESKTOP-H40TQ38 successfully set to delayed auto start
Service vstsagent.nicolekipkorir.Default.DESKTOP-H40TQ38 successfully set SID type
Service vstsagent.nicolekipkorir.Default.DESKTOP-H40TQ38 successfully configured
Enter whether to prevent service starting immediately after configuration is finished? (Y/N) (press enter for N) > y
PS C:\USERS\USER\agent>
```

Exercise 4: Configure the Microsoft Security DevOps GitHub actions

1. Navigate to your VulnDjango GitHub repo.
2. Select Actions
3. Select New workflow.
4. On the Get started with GitHub Actions page, select set up a workflow yourself
5. In the text box, enter a name for your workflow file. For example, msdevopssec.yml.
6. Copy and paste the following sample action workflow into the Edit new file tab.

name: MSDO windows-latest on:

push:

branches: [main]

pull_request:

branches: [main]

workflow_dispatch:

jobs:

sample:

MSDO runs on windows-latest and ubuntu-latest.

macos-latest supporting coming soon runs-on: windows-latest

steps:

- uses: actions/checkout@v3

- uses: actions/setup-dotnet@v3 with:

dotnet-version: |

5.0.x

6.0.x

Run analyzers

- name: Run Microsoft Security DevOps Analysis

uses: microsoft/security-devops-action@preview id: msdo

Upload alerts to the Security tab

- name: Upload alerts to Security tab

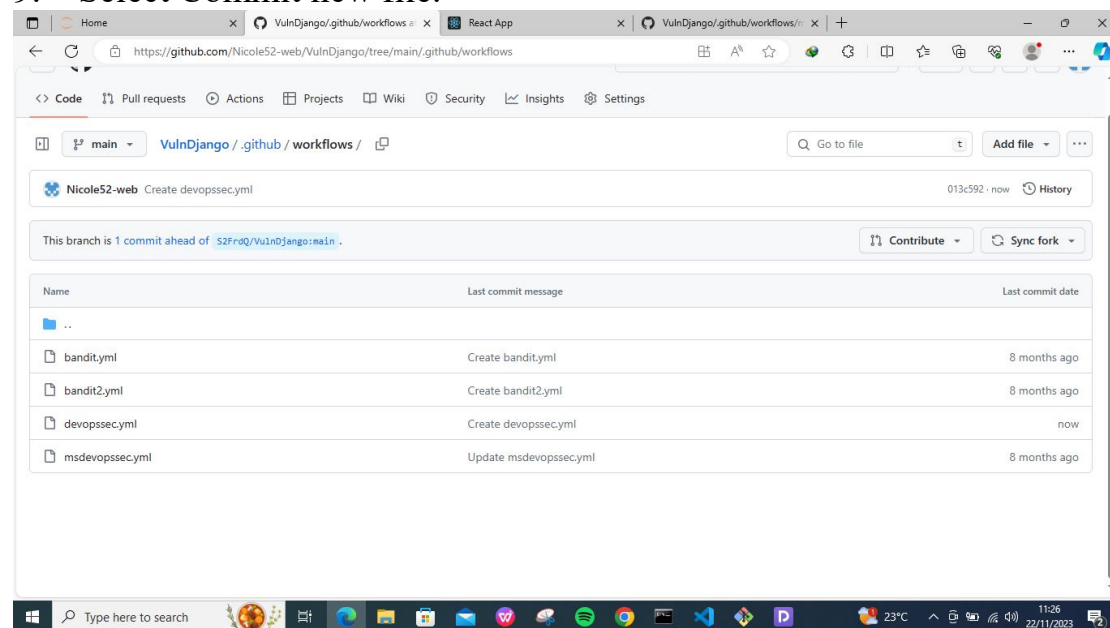
uses: github/codeql-action/upload-sarif@v2 with:

sarif_file: \${ steps.msdo.outputs.sarifFile }

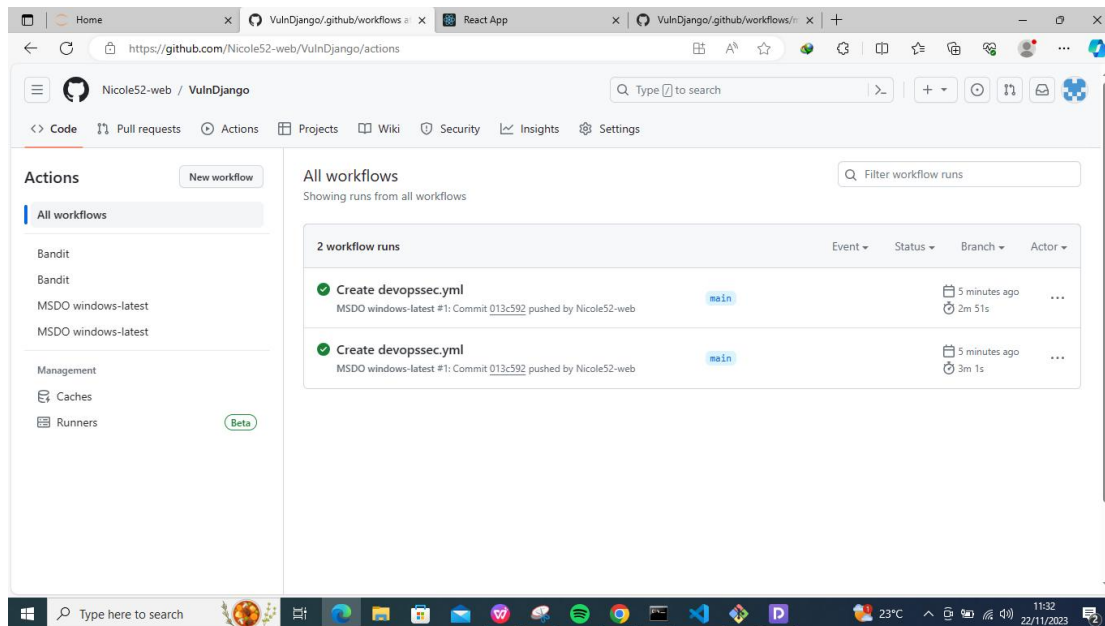
7. For details on various input options, see action.yml

8. Select Start commit

9. Select Commit new file.



The process can take up to one minute to complete.



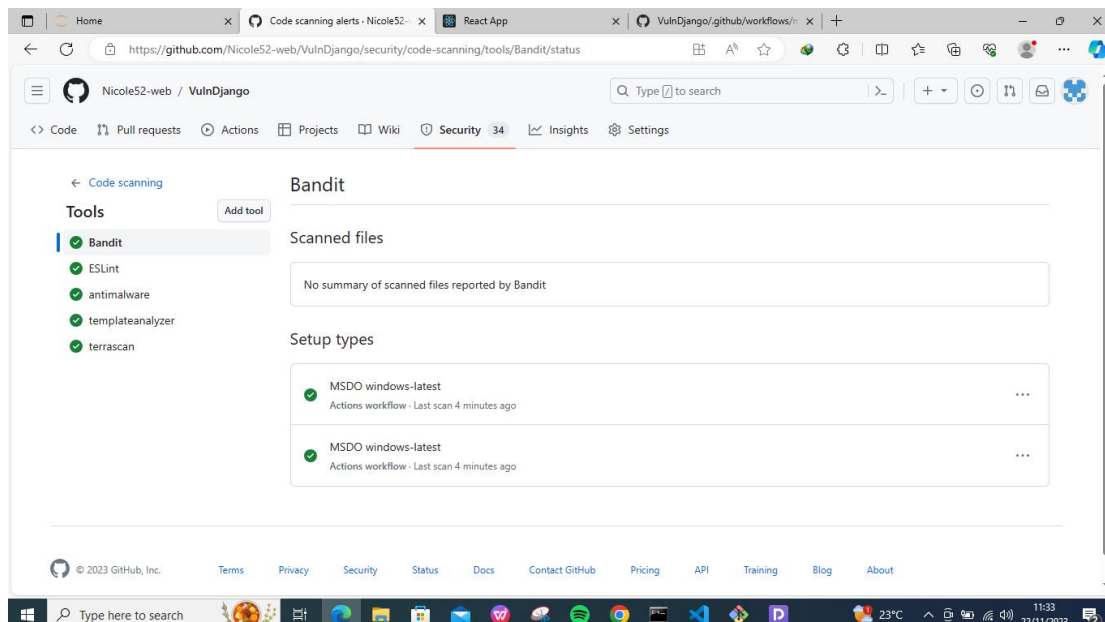
10. Select Actions and verify the new action is running.

View Scan Results

To view your scan results:

Navigate to Security > Code scanning alerts > Tool. From the dropdown menu, select Filter by tool.

Code scanning findings will be filtered by specific MSDO tools in GitHub. These code scanning results are also pulled into Defender for Cloud recommendations.



Conclusion

In conclusion, the Microsoft DevOps involves hands-on experience integrated workflows, continuous integration and deployment pipelines, collaboration tools and security practices. Covered on configuring pipelines, implementing security checks, monitoring tools, and responding to incidents. Gained insights into building scalable, flexible, and efficient DevOps processes using Microsoft tools such as Azure DevOps, Azure Resource Manager, and GitHub actions.