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

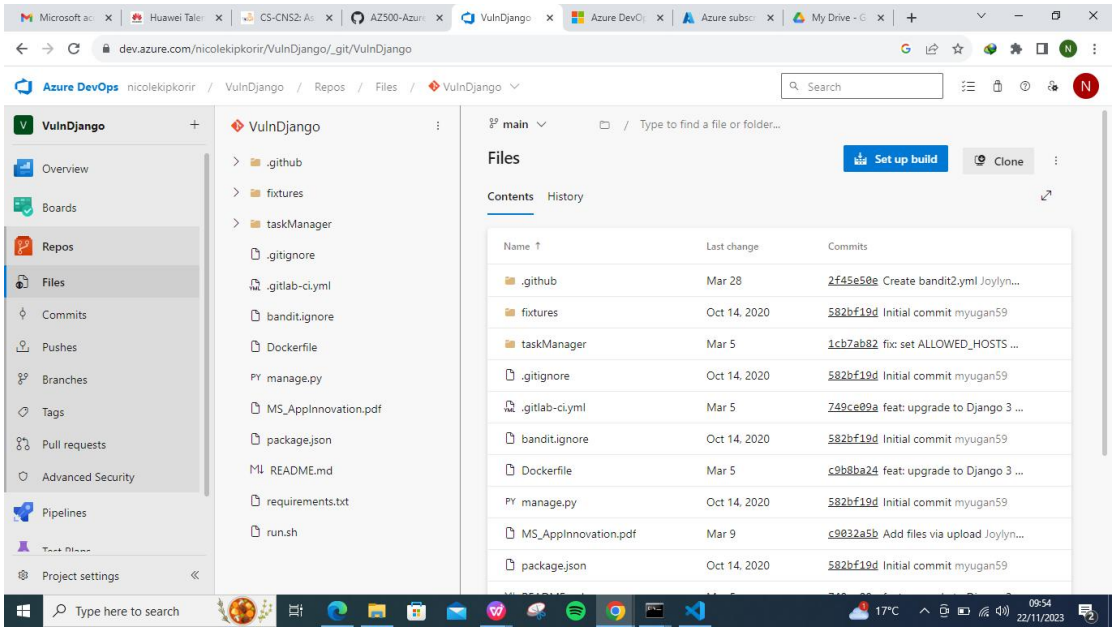
After completing this lab, you will be able to:

- 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



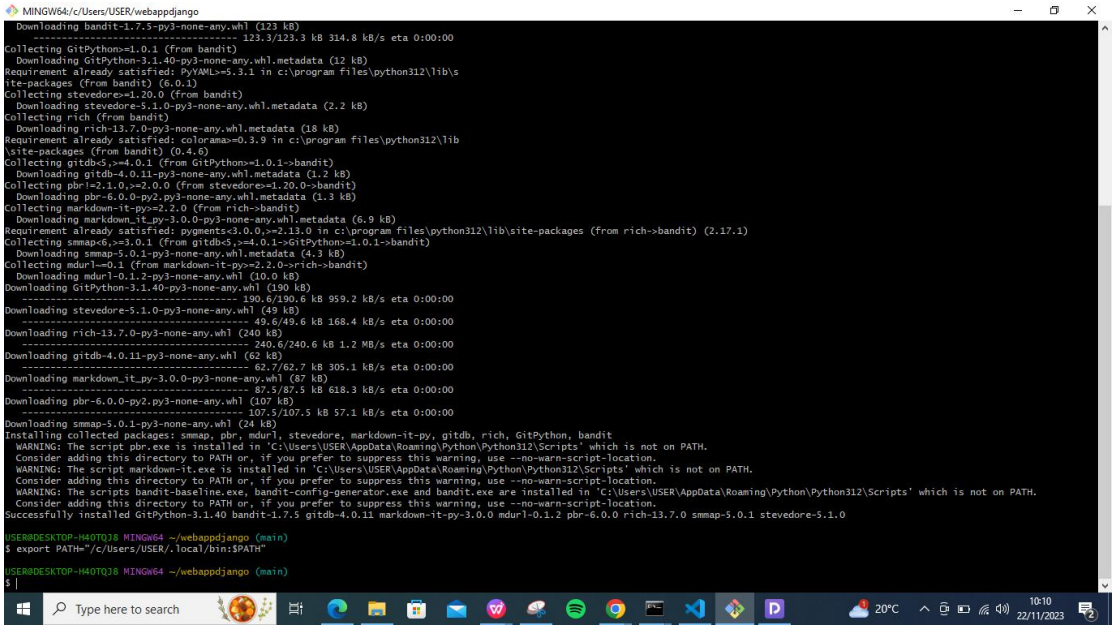
Exercise 2: SAST scan using Bandit locally.

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 webappdjangothen cd webappdjango
ii.Install Bandit pip3 install bandit
iii.If a warning is issued to add Directory to path, add using the below command.

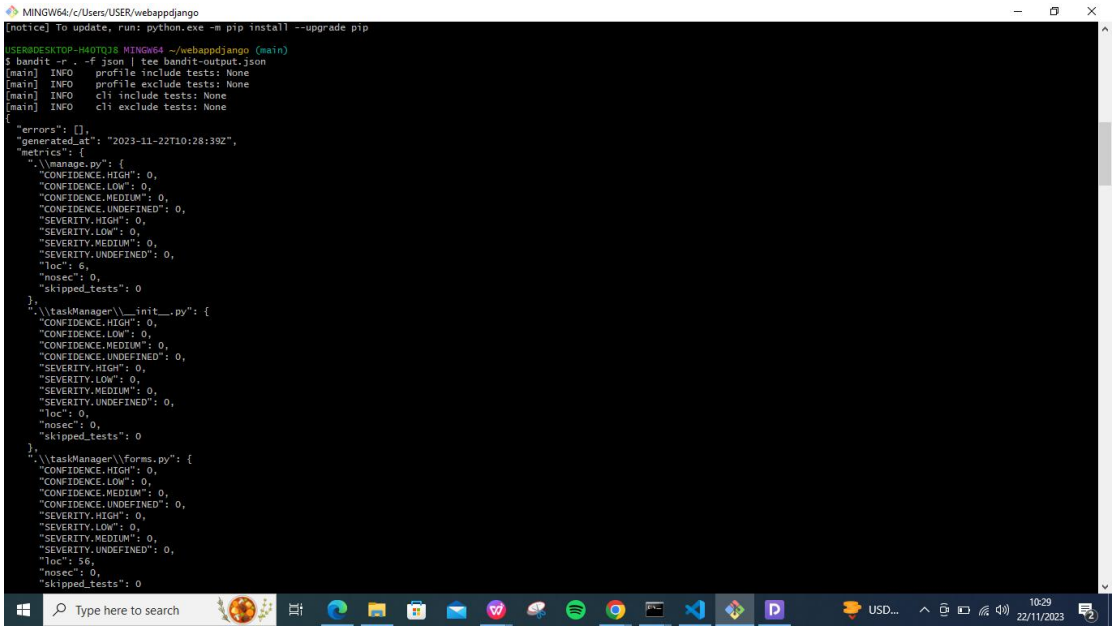
export PATH="/home/kali/.local/bin:$PATH"

To explore bandit --help



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



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

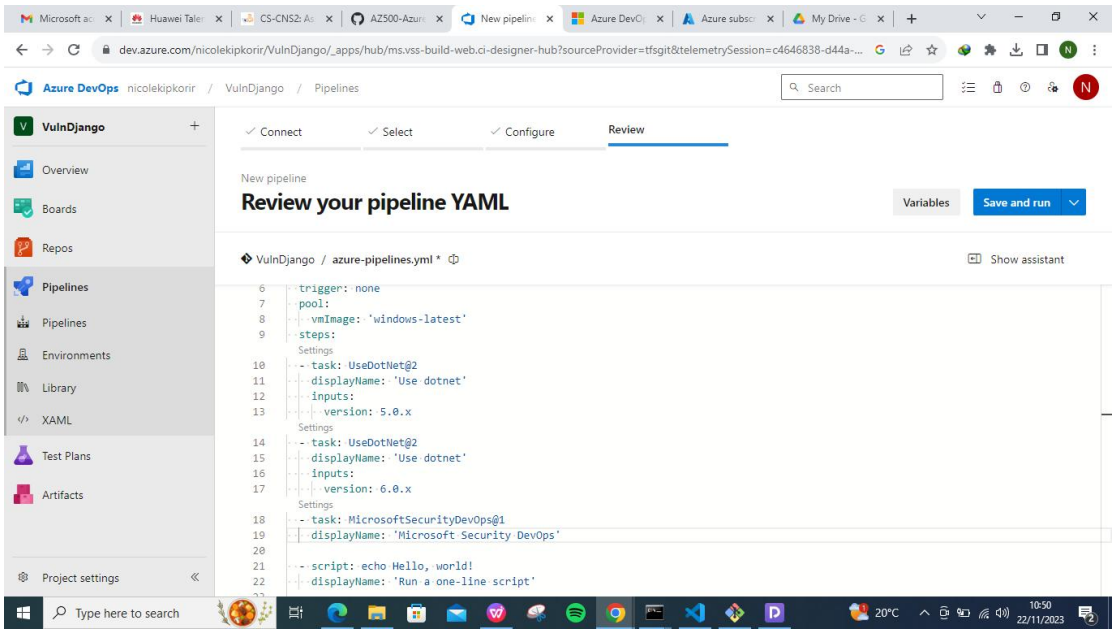
Note: Admin privileges to the Azure DevOps organization are required to install the extension.

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

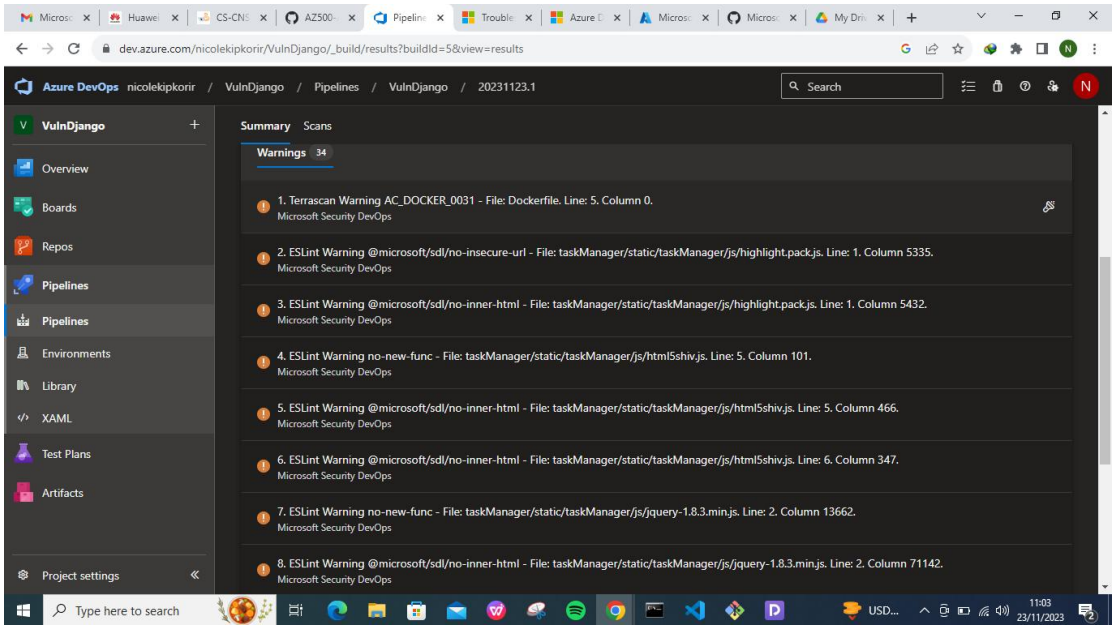
the VulnDjango project open, click on the marketplace icon > Browse Marketplace.

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.

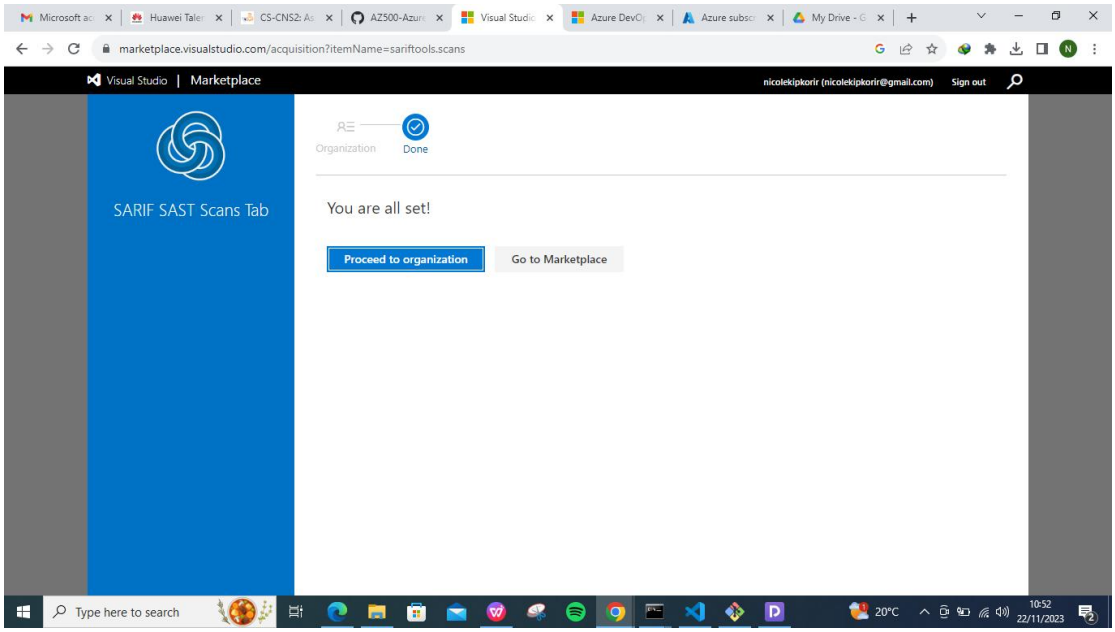
4. On the next page, select the desired Azure DevOps organization and Install. Proceed to organization once installed.
5. Navigate to your VulnDjango project, then Pipelines and Click New pipeline.
6. On the Where is your code? window, select Azure Repos Git (YAML) and select the VulnDjango repository.
7. On Add the following scripts as in into the yaml file.

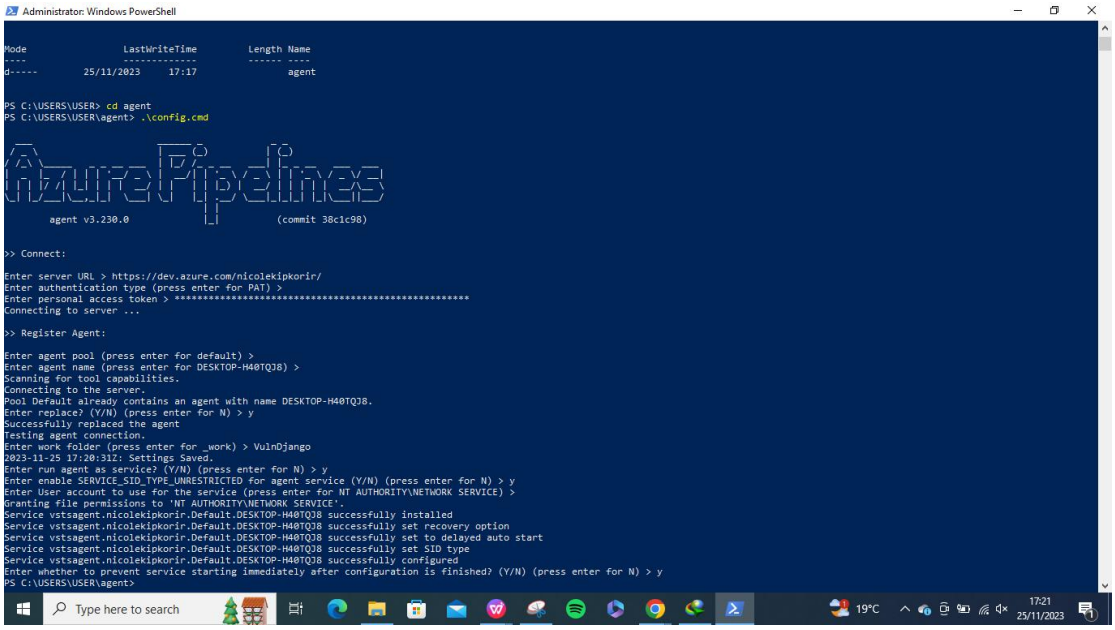


8. Click Save and run and let the pipeline run. You can check progress by going to Pipeline- Pipelines and select the running pipeline.
9. When done, you can view security vulnerabilities found by Microsoft Security DevOps , by clicking Scans.



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.





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:
```

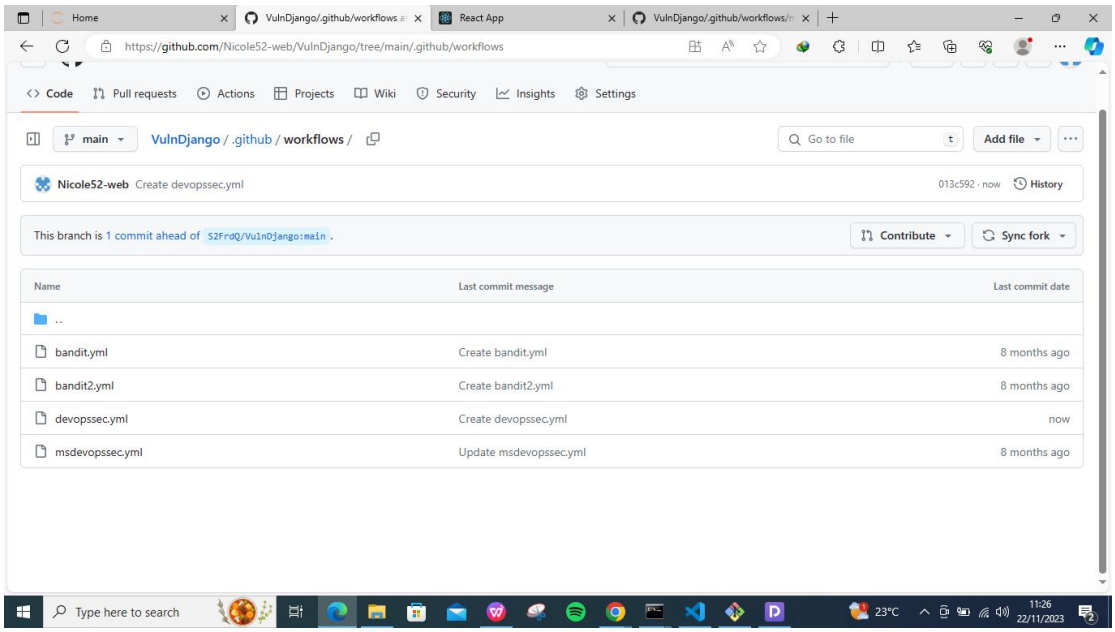
- uses: actions/checkout@v3

- uses: actions/setup-dotnet@v3 with:
dotnet-version: |
5.0.x
6.0.x

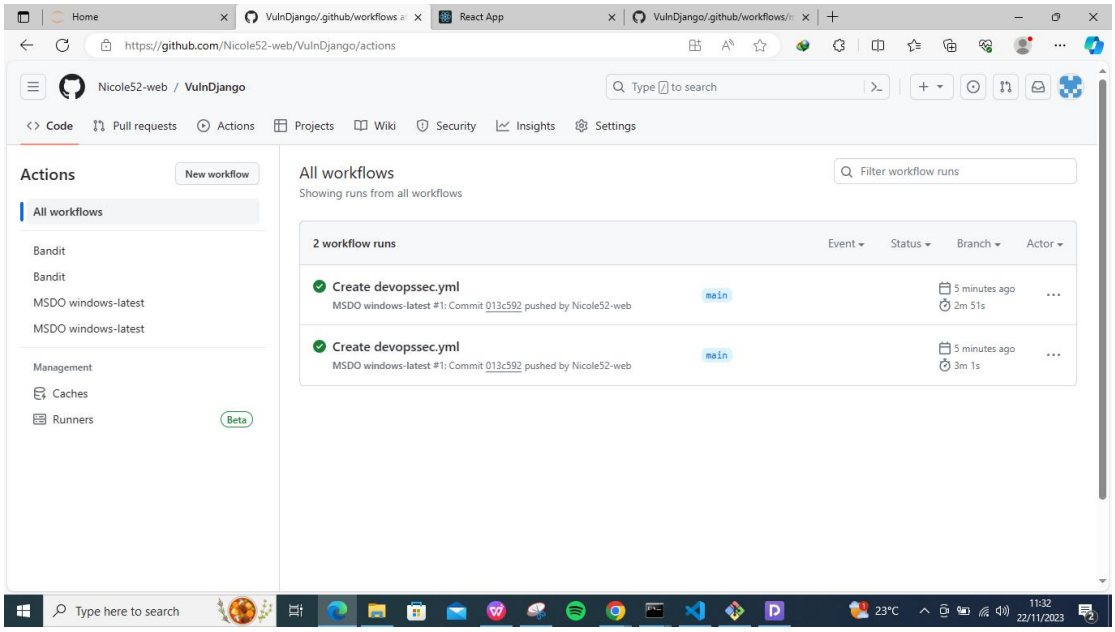
Run analyzers
- name: Run Microsoft Security DevOps Analysis
uses: microsoft/security-devops-action@preview id: msdo

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.



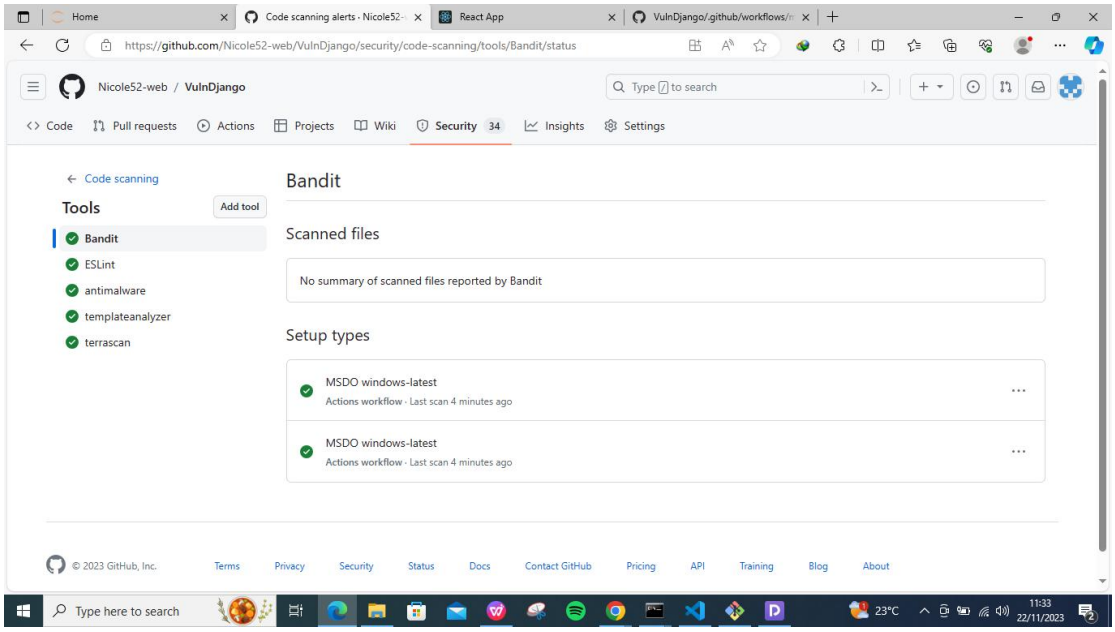
10. Select Actions and verify the new action is running.

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.