National College of Ireland

Configuration Manual

MSc Research Project
Cybersecurity

Shivam Rajesh Tiwari
Student ID: 22102396

School of Computing
National College of Ireland

Supervisor: Jawad Salahuddin

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: Shivam Rajesh Tiwari

Student ID: 22102396

Programme: MSc Cybersecurity **Year:** 2023

Module: Academic Internship

Lecturer: Jawad Salahuddin

Submission Due Date: 14-12-2023

Project Title: Enhancing Container Security Orchestration and Management Tool

Word Count: 442 **Page Count:** 8

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Shivam Rajesh Tiwari

Date: 14-12-2023

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Shivam Rajesh Tiwari
Student ID: 22102396

Introduction:

In this research project we did the parallel execution of the two docker images for the detection of the vulnerabilities which is achieved by implementation of a multithreading method using python where two docker images can be scanned simultaneously and the workload is split between the threads. This configuration manual is a step-by-step guide to install, setup, implement and automate complete workflow.

Configurations:

Tool	Version
Ubuntu	20.04
Git	2.25
Python	3.7
Java (for jenkins installation)	openJDK 17.0.9
Jenkins	2.435
Trivy	0.48.0
Docker	24.0.5

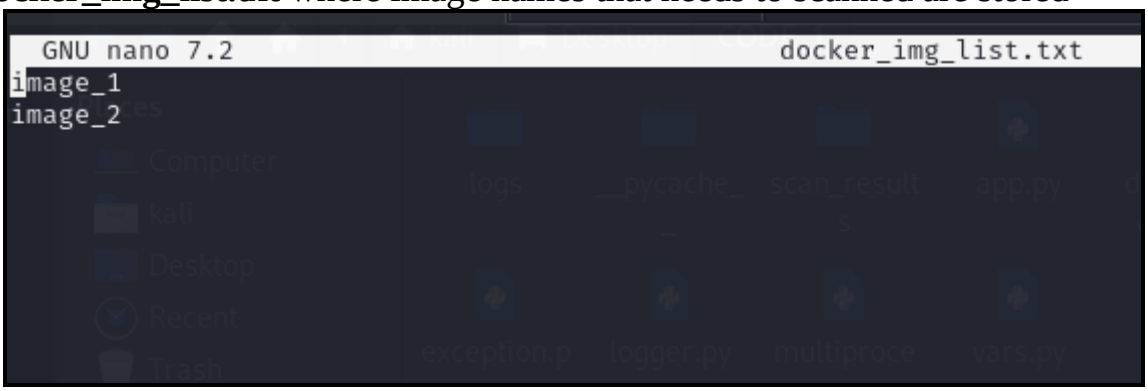
Implementation:

Step: 1 Setting up Logger.py and exception.py modules for tracking and handling the errors during the application run time. (krishnaik06, 2023)

```python
def error_message_detail(error, error_detail:sys):
    _,_,exc_tb = error_detail.exc_info()
    file_name = exc_tb.tb_frame.f_code.co_filename

    error_message= "Error raised in the script name [{0}] line number [{1}] error message
    [{2}]".format(file_name, exc_tb.tb_lineno,str(error))
    return error_message

class CustomException(Exception):
    def __init__(self, error_message, error_detail:sys):
        super().__init__(error_message)
        self.error_message = error_message_detail(error_message, error_detail=error_detail)

    def __str__(self):
        return self.error_message
```

```python
logging.basicConfig(
    filename=LOG_FILE_PATH,
    format="[ %(asctime)s ] %(lineno)d %(name)s - %(levelname)s - %(message)s",
    level=logging.INFO,
)
```

Step 2: Get the images for scanning is obtained by reading the **docker_img_list.txt** where image names that needs to scanned are stored



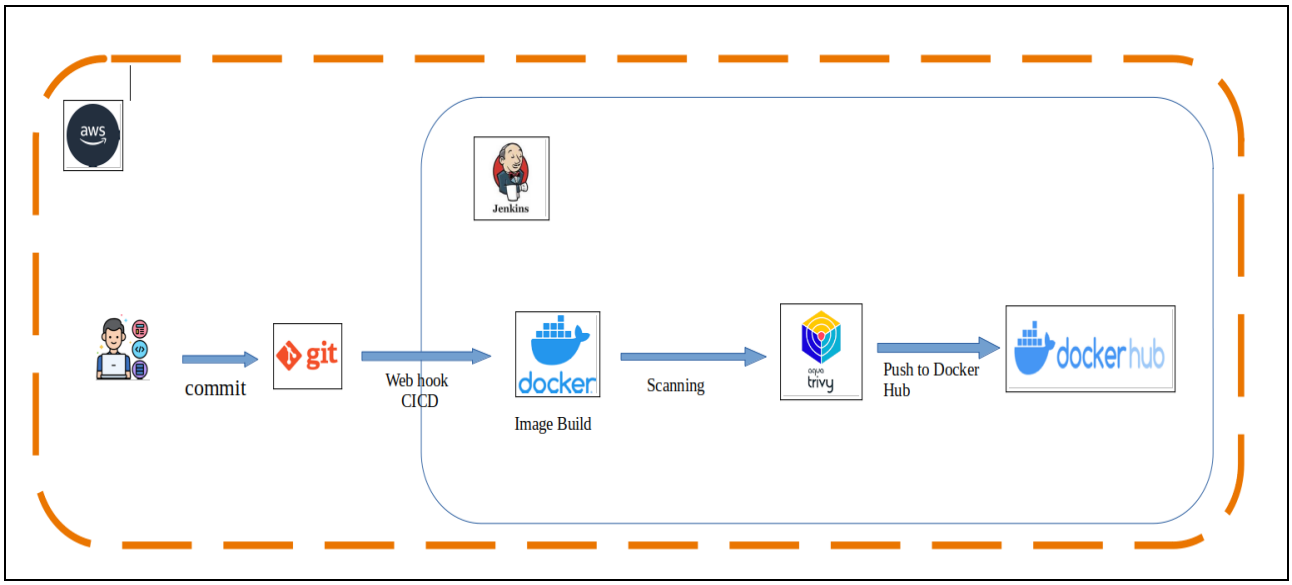
Step 3: In the multiprocess.py module the scanning method is implemented for scanning the images.

```
def io_bound(image_name):
    logging.info("Entering image scanning function")
    os.makedirs('scan_results', exist_ok = True)

    try:
        with open(os.path.join('scan_results', f'{image_name}.txt'), 'w') as f:
            pid = os.getpid()
            threadName = current_thread().name
            processName = current_process().name

            print(f"{pid} * {processName} * {threadName} * {image_name} \
                    ---> Start Scanning...")

            logging.info("Scanning Docker with trivy")

            scan_image = "trivy -q image -f table {}".format(image_name)
            logging.info("processing the scan results")
            scan_result = subprocess.check_output(scan_image.split()).decode('utf-8')
            logging.info("creating the report")
            f.writelines(scan_result)

            print(f"{pid} * {processName} * {threadName} * {image_name}  \
                    ---> Finished Scanning...")


    except Exception as e:
        CustomException(e, sys)
```

Step 4: Bind the scanning function with the multithreading function to achieve parallel scanning in the app.py module.

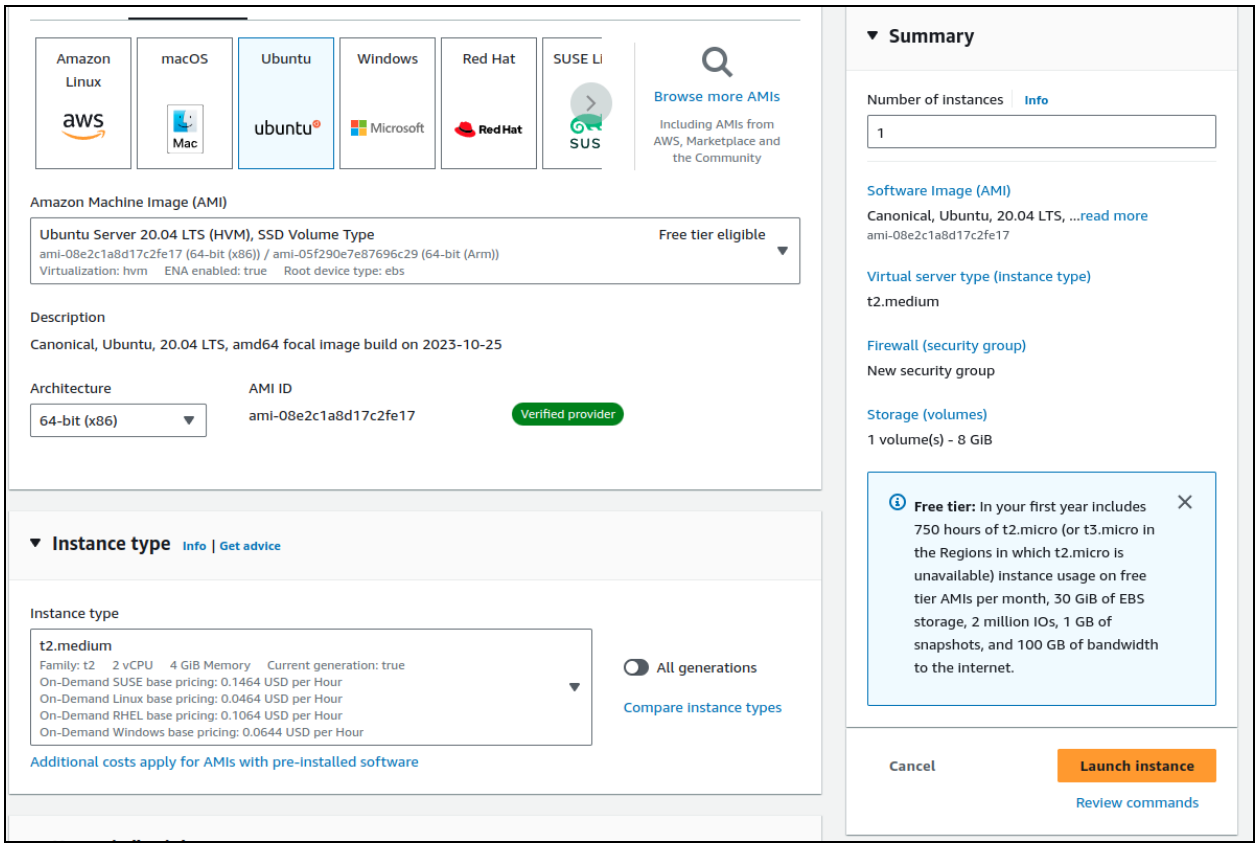
```
class Scanner:
    logging.info("Initializing the scanner")

    def __init__(self) -> None:
        self.docker_images = list(docker_images.values())

    def processing(self):
        logging.info("Entering the multithreading function")

        try:

            threads = []
            for image_name in self.docker_images:
                logging.info("entered for loop")
                t = Thread(target = multiprocess.io_bound, args =(image_name, ))
                logging.info("first thread Initialized")
                t.start()
                threads.append(t)


            for t in threads:
                t.join()

        except Exception as e:
            CustomException(e, sys)
```

Automating the Process:
Jenkins used for automating the pipeline.



Setup the environment:

Create the AWS EC2 instance with t2.medium and ubuntu 20.04 AMI.
In the AWS dashboard click “Launch Instance” and choose Ubuntu server 20.04 LTS as a AMI and Instance type as t2.medium



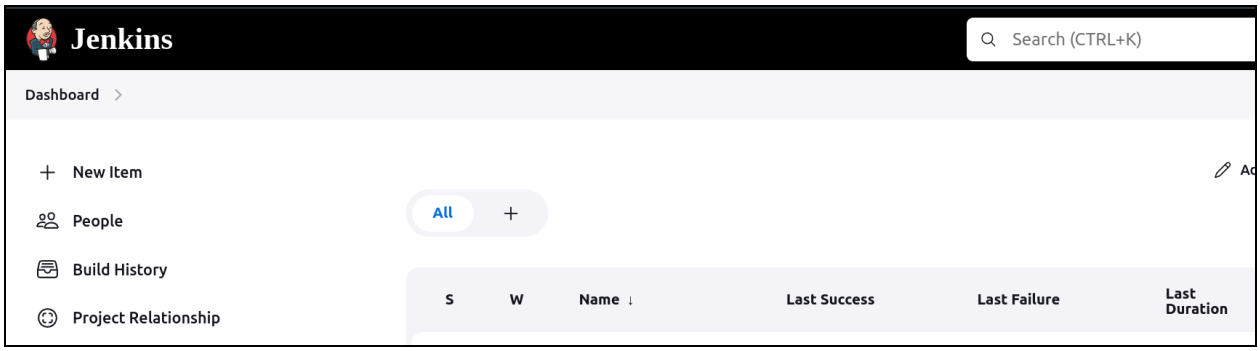
Install required packages:

- Jenkins
- Trivy

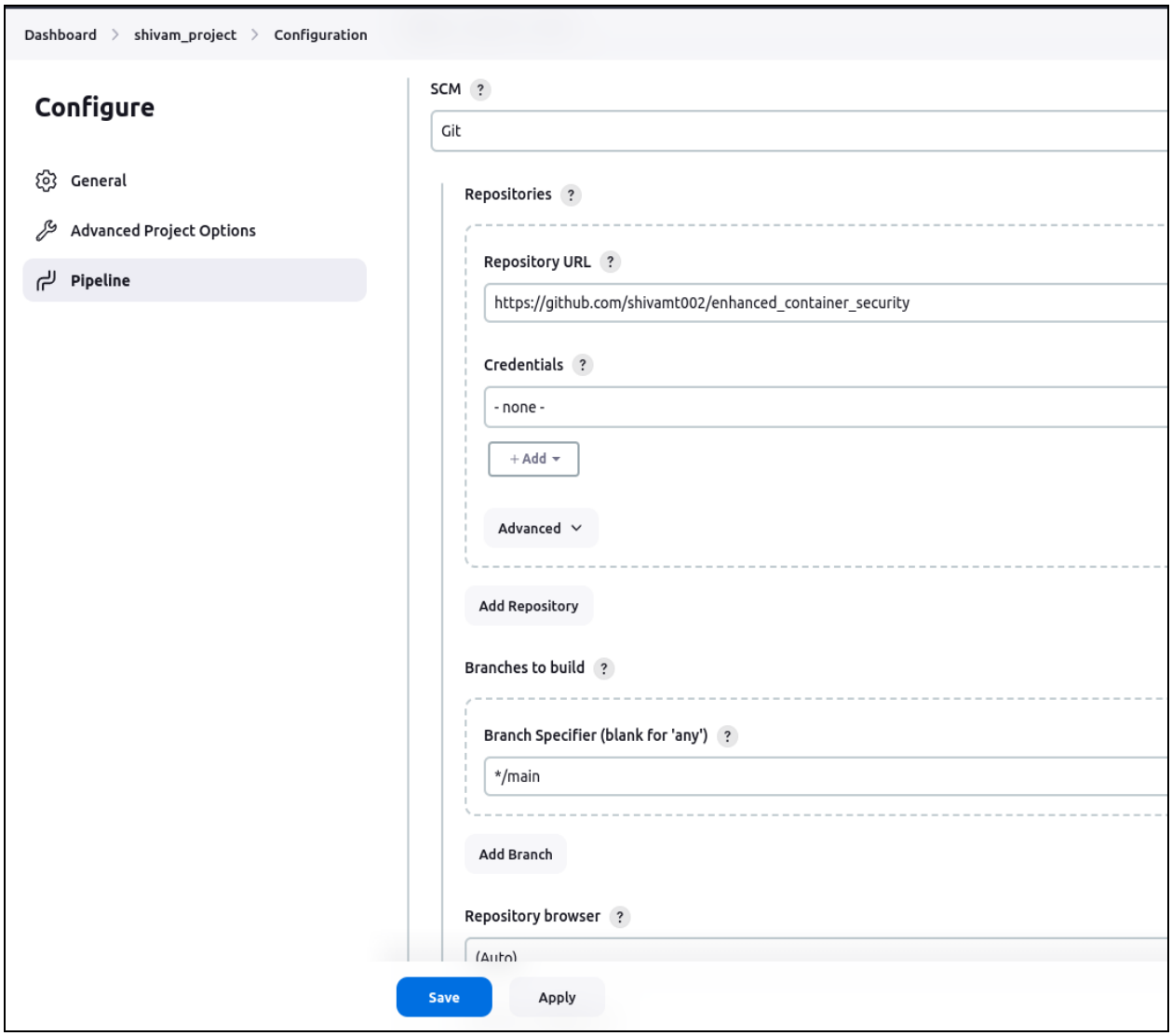
- Docker

The above said packages are installed by shell scripts inside the installation directory

Step:5 Once installed, it's accessible at the port number <public_ip>:8080 from the server.
By adding the new item in the dashboard, we can add the new job based on the type of pipeline.



Step: 6 This job type takes a Jenkins file from the GitHub repository which is written in the groovy language which can be modified easily according to the needs of the user.
The configuration of the pipeline is shown below



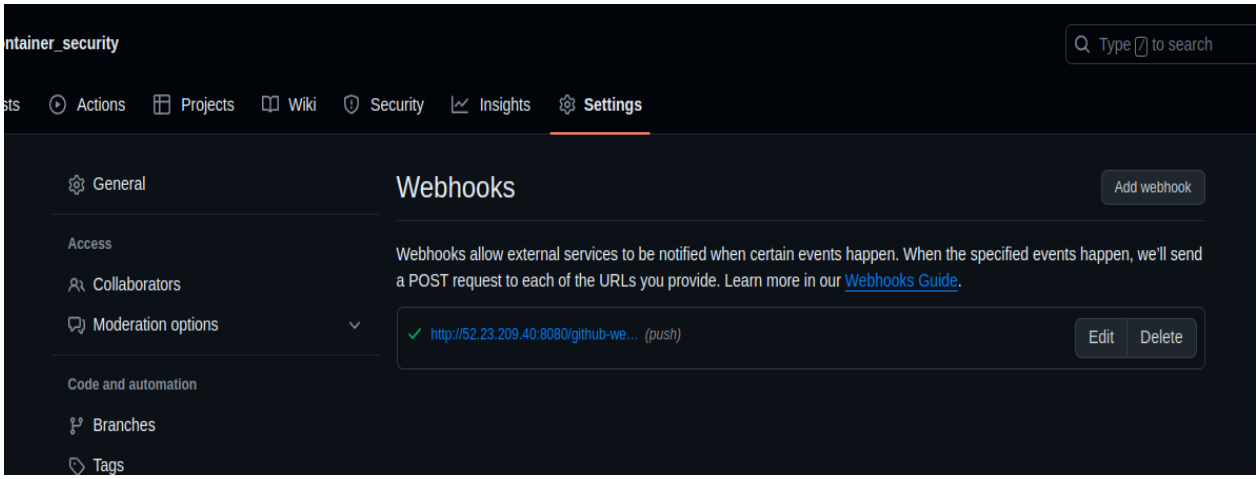
Step : 7 The above pipeline has three stages

- SCM checkout
- Image Build
- Image Scan

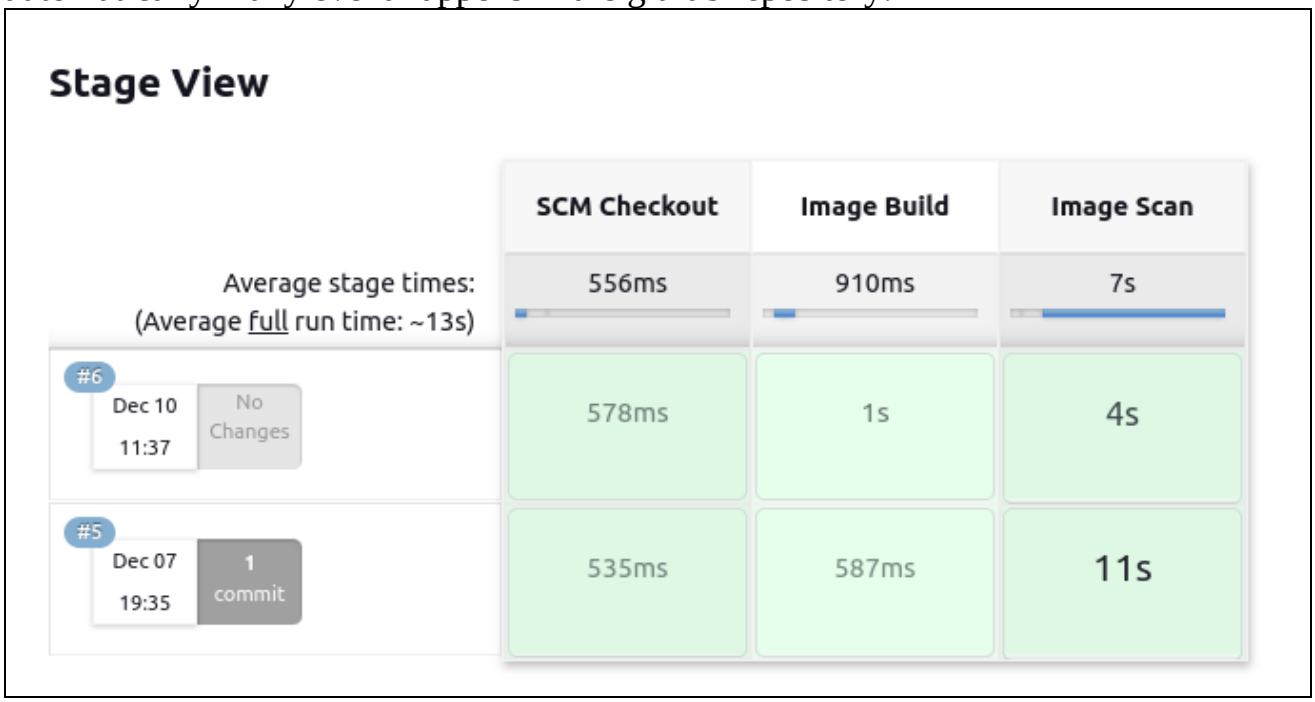
Step: 8 For a docker image, a sample to-do list application is used from official docker hub documentation. It can be found here, https://docs.docker.com/get-started/02_our_app/

```
# syntax=docker/dockerfile:1

FROM node:18-alpine
WORKDIR /app
COPY . .
RUN yarn install --production
CMD ["node", "src/index.js"]
EXPOSE 3000
```

Step: 8 All the artifacts and the code has been stored in GitHub repository and GitHub webhook is then integrated with Jenkins pipeline to trigger the pipeline in case any changes occur.



Step: 9 Now, our pipeline is ready to run the jobs which can be triggered automatically if any event happens in the github repository.



Step: 10 The result files can be found in the jenkins workspace which is " /var/lib/jenkins/workspace/
<job_name> "

References

Docker Documentation. (2022). *Sample application*. [online] Available at: https://docs.docker.com/get-started/02_our_app/.

krishnaik06 (2023). *The-Grand-Complete-Data-Science-Materials/ML Projects/End-to-End-Heart-Disease-Prediction/src/Heart at main · krishnaik06/The-Grand-Complete-Data-Science-Materials*. [online] GitHub. Available at: https://github.com/krishnaik06/The-Grand-Complete-Data-Science-Materials/tree/main/ML%20Projects/End-to-End-Heart-Disease-Prediction/src/Heart [Accessed 13 Dec. 2023].