Configuration Manual

MSc Research Project
Cloud Computing

Teena Kohli
Student ID: 20226845

School of Computing
National College of Ireland

Supervisor: Vikas Sahni

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Teena Kohli
Student ID:	20226845
Programme:	Cloud Computing
Year:	2021
Module:	MSc Research Project
Supervisor:	Vikas Sahni
Submission Due Date:	15/08/2022
Project Title:	Configuration Manual
Word Count:	933
Page Count:	9

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:	Teena Kohli
Date:	14th August 2022

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

Teena Kohli
20226845

1 Introduction

The configuration guide is a component of the Master's research project in cloud Computing where manual's primary goal is to demonstrate the replica of implementation steps and procedure followed doing research on Topic - Efficient Threat Detection Framework for Docker Containers using AppArmor Profile Zhu and Gehrmann (2021) and Clair Vulnerability Scanning ToolJaved and Toor (2021) on WordPress ApplicationMesa et al. (2018) for Experiment Evaluation.

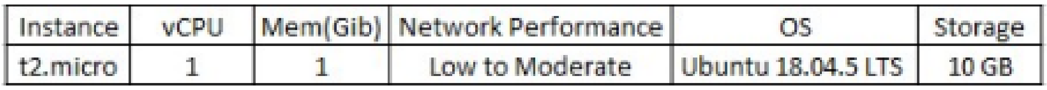
2 Pre-Requisites for Architecture Implementation

2.1 Amazon Host Platform

As the primary objective of this research, a cloud platform provided by NCI - cloud.ncirl.ie has been used to implement and evaluate a defence framework, proposed using AppArmor Profile and Clair Image scanning tool on example application of WordPress.An Ubuntu based AWS Cloud Platform t2.micro free tier eligible EC2 instance in Ireland Region has been configured with below configuration.

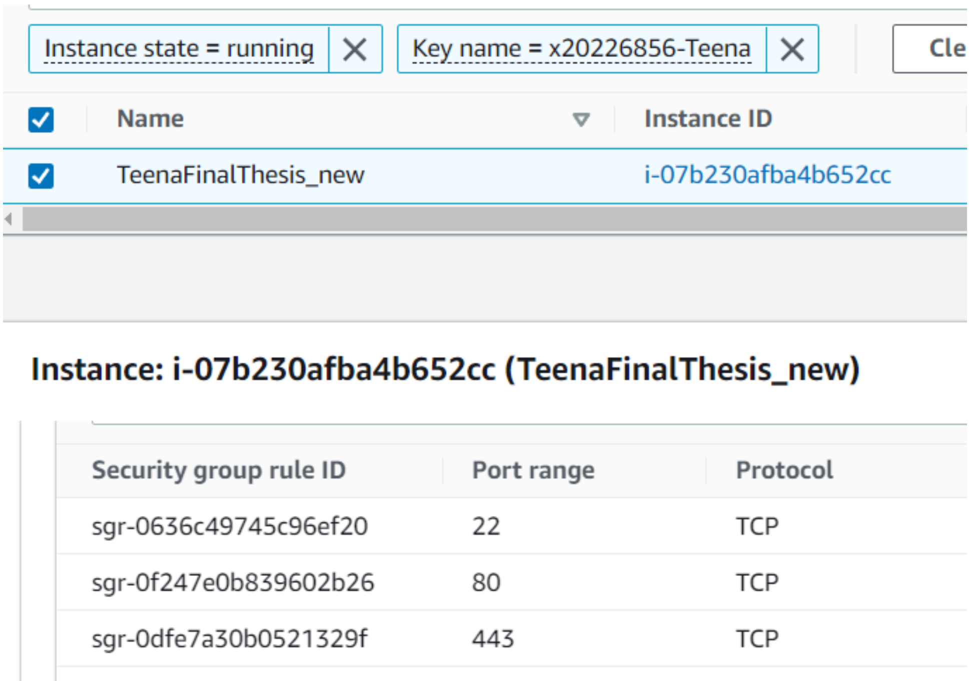
- Step 1: Select Amazon AMI Ubuntu Server with Default available AMI - Canonical, Ubuntu, 22.04 LTS, amd64 jammy image build on 2022-06-09.
- Step 2: Select t2.micro Instance type and select KeyPair(CustomCreated)
- Step 3: Increase default storage to 10GB
- Step 4: Attach Custom Created Security group for access allow.

Fig 1 presents the instance configuration summary and Fig 2 represents the Port range enabled for application access through inbound security group attached to it.



Instance	vCPU	Mem(Gib)	Network Performance	OS	Storage
t2.micro	1	1	Low to Moderate	Ubuntu 18.04.5 LTS	10 GB

Figure 1: Ec2 Instance Configuration



Instance state = running ✕ Key name = x20226856-Teena ✕ Cle

☑	Name	Instance ID
☑	TeenaFinalThesis_new	i-07b230afba4b652cc

Instance: i-07b230afba4b652cc (TeenaFinalThesis_new)

Security group rule ID	Port range	Protocol
sgr-0636c49745c96ef20	22	TCP
sgr-0f247e0b839602b26	80	TCP
sgr-0dfe7a30b0521329f	443	TCP

Figure 2: Security Group Configuration

2.2 Docker and Docker-Compose Installation

- Following the official documentation from docker[1], Docker community version on 20.10.17 and docker compose version 1.29.2 has been installed for defining and running multi-container Docker applications. Fig 3 represents the Docker installation.

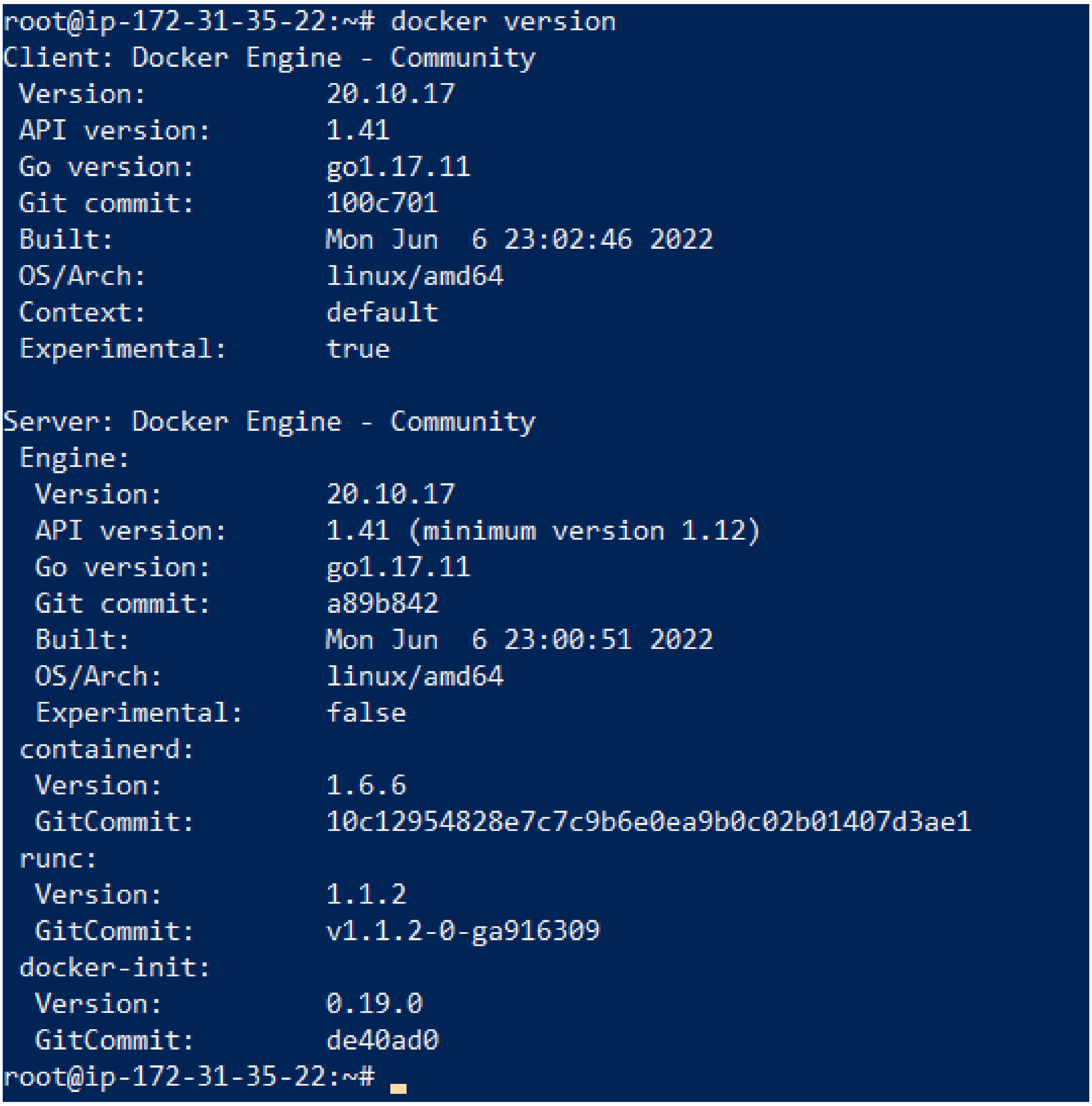
- Step 1: update the apt packages and system install packages.
  $ sudo apt-get update
- Step 2: Install the most recent versions of Docker Engine, containerd, and Docker Compose.
  $ sudo apt-get install docker-ce docker-ce-cli containerd.io docker-compose-plugin
- Step 3: Choose the specific version of docker engine from the repo list and install
  $ sudo apt-get install docker-ce=5:20.10.17 3-0 ubuntu-jammy docker-ce-cli=5:20.10.17 3-0 ubuntu-jammy containerd.io docker-compose-plugin
- Step 4: Verfiy docker
  $ docker –version

2.3 AppArmor Configuration

With Latest linux/Ubuntu Distribution AppArmor comes as default[2] however the custom profile as per the security requirements are then needs to be created and saved under /etc/apparmor.d/ for further implementation. Fig 4 and Fig 5 represents AppArmor status and current status as enforced(Active) mode in Host.

[1] https://docs.docker.com/engine/install/ubuntu/

[2] https://docs.docker.com/engine/security/apparmor/

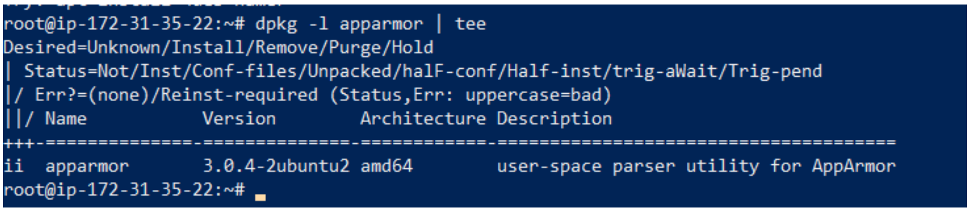


```
root@ip-172-31-35-22:~# docker version
Client: Docker Engine - Community
 Version:           20.10.17
 API version:       1.41
 Go version:        go1.17.11
 Git commit:        100c701
 Built:             Mon Jun  6 23:02:46 2022
 OS/Arch:           linux/amd64
 Context:           default
 Experimental:      true

Server: Docker Engine - Community
 Engine:
  Version:          20.10.17
  API version:      1.41 (minimum version 1.12)
  Go version:       go1.17.11
  Git commit:       a89b842
  Built:            Mon Jun  6 23:00:51 2022
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
  Version:          1.6.6
  GitCommit:        10c12954828e7c7c9b6e0ea9b0c02b01407d3ae1
 runc:
  Version:          1.1.2
  GitCommit:        v1.1.2-0-ga916309
 docker-init:
  Version:          0.19.0
  GitCommit:        de40ad0
root@ip-172-31-35-22:~#
```

Figure 3: Docker community Edition

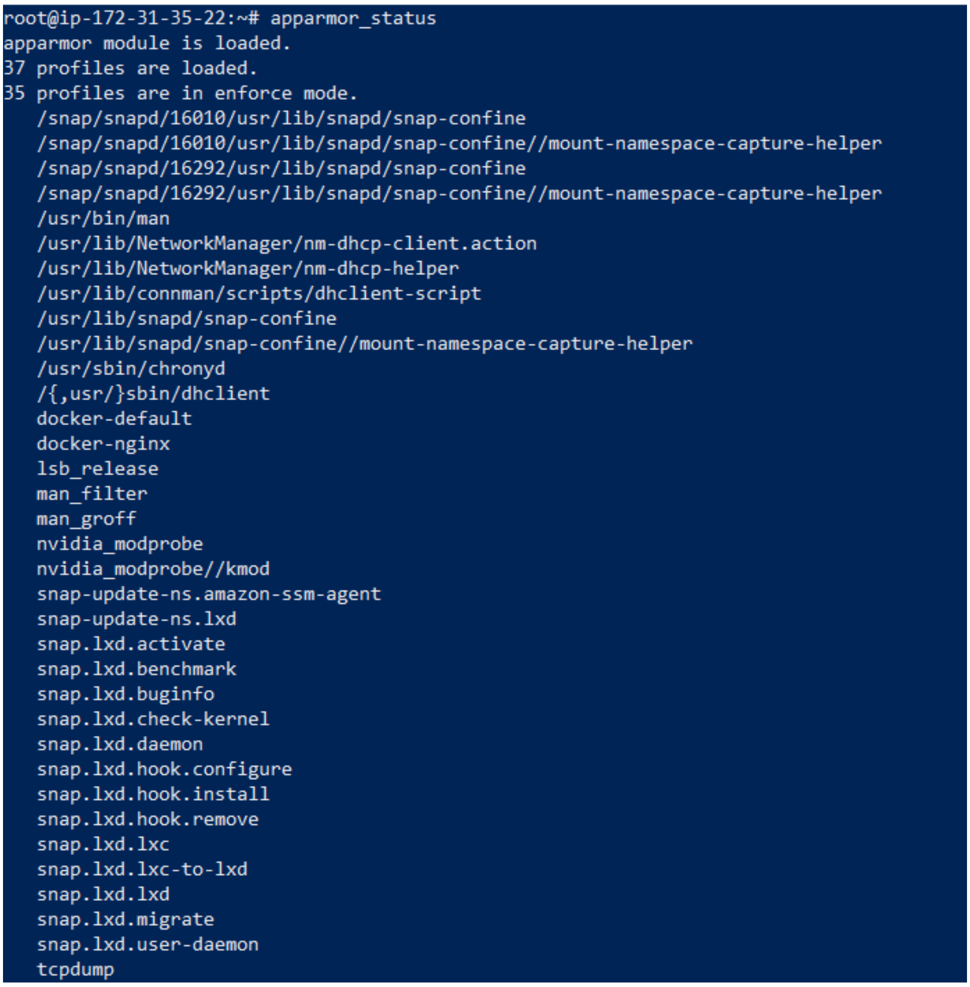
- Step 1 Check AppArmor version
  $ dpkg -l apparmor — tee

- Step 2 Check AppArmor status to ensure it is enable and in enforced mode
  $ apparmor_status



```
root@ip-172-31-35-22:~# dpkg -l apparmor | tee
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Inst/Conf-files/Unpacked/halF-conf/Half-inst/trig-aWait/Trig-pend
|/ Err?=(none)/Reinst-required (Status,Err: uppercase=bad)
||/ Name           Version        Architecture Description
+++-==============-==============-============-=====================================
ii  apparmor       3.0.4-2ubuntu2 amd64        user-space parser utility for AppArmor
root@ip-172-31-35-22:~#
```

Figure 4: AppArmor package installation verification



```
root@ip-172-31-35-22:~# apparmor_status
apparmor module is loaded.
37 profiles are loaded.
35 profiles are in enforce mode.
   /snap/snapd/16010/usr/lib/snapd/snap-confine
   /snap/snapd/16010/usr/lib/snapd/snap-confine//mount-namespace-capture-helper
   /snap/snapd/16292/usr/lib/snapd/snap-confine
   /snap/snapd/16292/usr/lib/snapd/snap-confine//mount-namespace-capture-helper
   /usr/bin/man
   /usr/lib/NetworkManager/nm-dhcp-client.action
   /usr/lib/NetworkManager/nm-dhcp-helper
   /usr/lib/connman/scripts/dhclient-script
   /usr/lib/snapd/snap-confine
   /usr/lib/snapd/snap-confine//mount-namespace-capture-helper
   /usr/sbin/chronyd
   /{,usr/}sbin/dhclient
   docker-default
   docker-nginx
   lsb_release
   man_filter
   man_groff
   nvidia_modprobe
   nvidia_modprobe//kmod
   snap-update-ns.amazon-ssm-agent
   snap-update-ns.lxd
   snap.lxd.activate
   snap.lxd.benchmark
   snap.lxd.buginfo
   snap.lxd.check-kernel
   snap.lxd.daemon
   snap.lxd.hook.configure
   snap.lxd.hook.install
   snap.lxd.hook.remove
   snap.lxd.lxc
   snap.lxd.lxc-to-lxd
   snap.lxd.lxd
   snap.lxd.migrate
   snap.lxd.user-daemon
   tcpdump
```

Figure 5: AppArmor status in enforced mode

2.4 Clair Scanning Tool Installation

Following the official documentation from Clair [3], postgres/clair/clairctl stack has been configured using below commands.Fig 6 represents the Clair postdress db and clair-local-scan tool running containers.

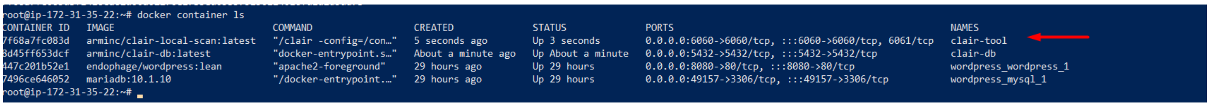
- Step 1 Download Clair from Github repository
  $ curl -L https://github.com/arminc/clair-scanner

- Step 2 Make build and corss compile.
  $ make build
  $ make cross

- Step 3 Run docker container for Clair utility tool
  $ docker run -p 5432:5432 -d –name clair-db clair-db:latest

- Step 4 Run docker container for Clair Postgress local DB acting as CVE repository.
  $ docker run -p 6060:6060 –link clair-db:postgres -d –name clair-local-scan:latest

- Step 5 Verify running container
  $ docker container ls



```
root@ip-172-31-35-22:~# docker container ls
CONTAINER ID   IMAGE                             COMMAND                  CREATED              STATUS              PORTS                                               NAMES
7f68a7fc083d   arminc/clair-local-scan:latest   "/clair -config=/con…"   5 seconds ago        Up 3 seconds        0.0.0.0:6060->6060/tcp, :::6060->6060/tcp, 6061/tcp   clair-tool
8d45ff653dcf   arminc/clair-db:latest           "docker-entrypoint.s…"   About a minute ago   Up About a minute   0.0.0.0:5432->5432/tcp, :::5432->5432/tcp            clair-db
447c201b52e1   endophage/wordpress:lean         "apache2-foreground"     29 hours ago         Up 29 hours         0.0.0.0:8080->80/tcp, :::8080->80/tcp                wordpress_wordpress_1
7496ce646052   mariadb:10.1.10                  "/docker-entrypoint.…"   29 hours ago         Up 29 hours         0.0.0.0:49157->3306/tcp, :::49157->3306/tcp          wordpress_mysql_1
root@ip-172-31-35-22:~#
```

Figure 6: Clair tool running in container

2.5 Nginx Sample Application

For first case experiment Nginx latest image has been pulled from Docker central repository [4], docker.io using below commands

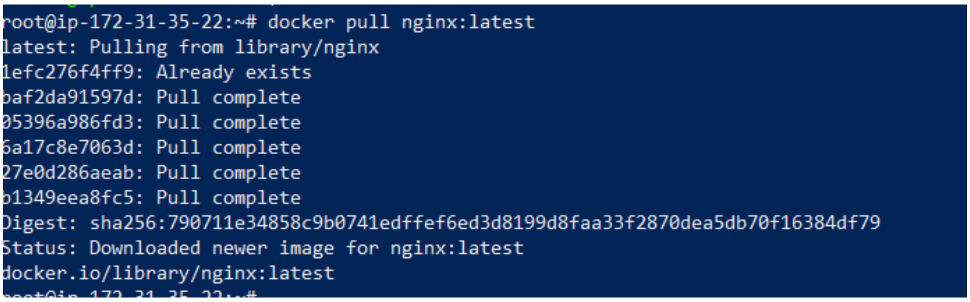
- docker pull nginx:latest

Fig 7 presents the Standalone Nginx Image Docker Pull output

2.6 WordPress Sample Application

For second case experiment, WordPress application stack including Apache, PHP, MYSQL images are containerized in a multi-container stack using docker compose [5]following below procedure.

[3] https://github.com/quay/clair
[4] https://hub.docker.com//*nginx*/
[5] https://docs.docker.com/samples/wordpress/



```
root@ip-172-31-35-22:~# docker pull nginx:latest
latest: Pulling from library/nginx
1efc276f4ff9: Already exists
baf2da91597d: Pull complete
05396a986fd3: Pull complete
6a17c8e7063d: Pull complete
27e0d286aeab: Pull complete
b1349eea8fc5: Pull complete
Digest: sha256:790711e34858c9b0741edffef6ed3d8199d8faa33f2870dea5db70f16384df79
Status: Downloaded newer image for nginx:latest
docker.io/library/nginx:latest
root@ip-172-31-35-22:~#
```

Figure 7: Latest Nginx Docker Pull

- Step 1 Create a docker File for PHP container for static content
  $ nano DockerFile
  $ FROM php:5.6-apache
  $ COPY ./html /var/www/html
  $ COPY ./zues /var/www/html/wp-content/themes/zues
  $ COPY ./php.ini /usr/local/etc/php/
  $ RUN mkdir /var/www/html/wp-content/uploads
  $ RUN chown -R www-data:www-data /var/www/html

- Step 2 Create Docker compose File including mysl and WordPress image
  Fig 8 presents the WordPress docker-compose yaml structure.

- Step 3 Bring the docker-compose in background
  $docker-compose up

3 Framework Deployment

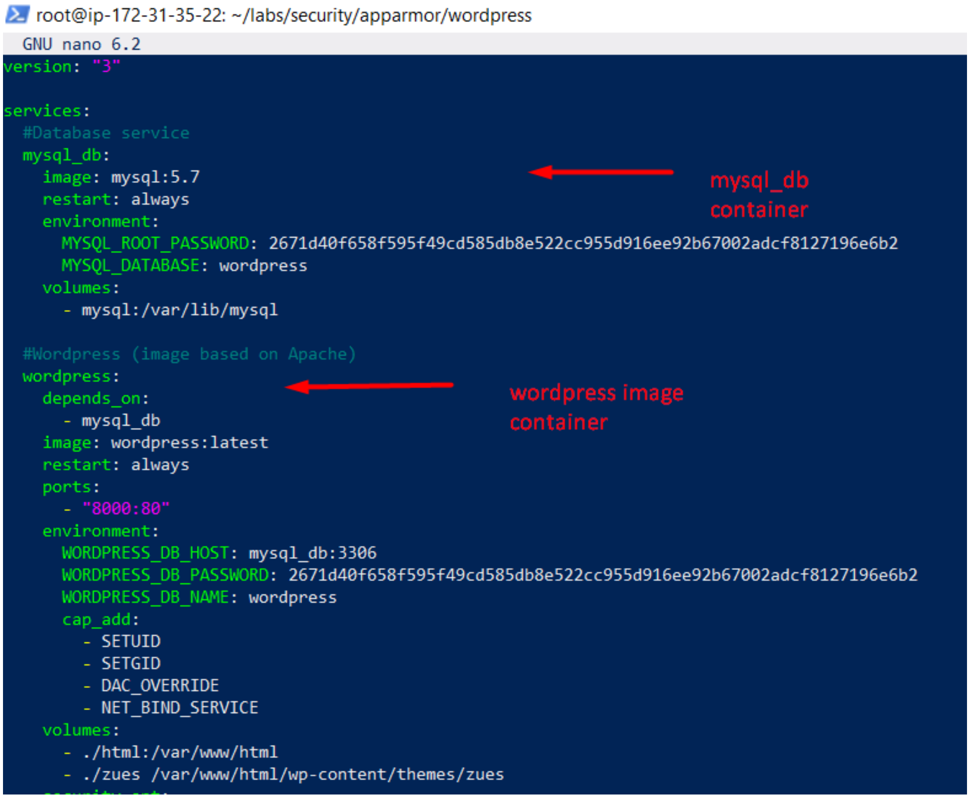
Upon successful installation of all pre-requisites packages and tool the final step of project is Framework integration and deployment.As mentioned two case studies have been created and procedure of Deployment is as follows:

3.1 Case-1 Nginx Image Evaluation

- Step 1: Run Clair Scanner tool on Nginx Latest image and report has been saved as nginxscan-report.json
  $ clair-scanner -r nginxscan-report.json –ip 172.17.0.1 nginx:latest

- Step 2: Based on scan report,an custom AppArmor Profile has been designed to mitigate OS package vulnerability which includes deny rules and apply restriction on kernel capabilities.
  $ nano customaprmor Codefile attached separately.

- Step 3: Load AppArmor into Host
  $ apparmor_parser -r -W /etc/apparmor.d/CustomProfile-Nginx

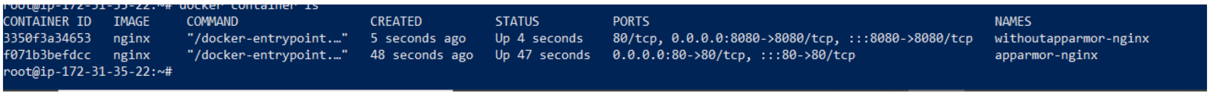


```
version: "3"

services:
  #Database service
  mysql_db:
    image: mysql:5.7
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: 2671d40f658f595f49cd585db8e522cc955d916ee92b67002adcf8127196e6b2
      MYSQL_DATABASE: wordpress
    volumes:
      - mysql:/var/lib/mysql

  #Wordpress (image based on Apache)
  wordpress:
    depends_on:
      - mysql_db
    image: wordpress:latest
    restart: always
    ports:
      - "8000:80"
    environment:
      WORDPRESS_DB_HOST: mysql_db:3306
      WORDPRESS_DB_PASSWORD: 2671d40f658f595f49cd585db8e522cc955d916ee92b67002adcf8127196e6b2
      WORDPRESS_DB_NAME: wordpress
      cap_add:
        - SETUID
        - SETGID
        - DAC_OVERRIDE
        - NET_BIND_SERVICE
    volumes:
      - ./html:/var/www/html
      - ./zues /var/www/html/wp-content/themes/zues
```

Figure 8: WordPress Docker Compose configuration

- Step 4: Start Nginx container with attached security profile with port publish
  $ docker run –security-opt ”apparmor=CustomProfile-Nginx” -p 80:80 -d –name apparmor-nginx nginx

In Fig 9 both Nginx containers are displayed, one with a CustomAppArmor profile and the other without one.



```
root@ip-172-31-35-22:~# docker container ls
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS          PORTS                                                  NAMES
3350f3a34653   nginx     "/docker-entrypoint.…"   5 seconds ago    Up 4 seconds    80/tcp, 0.0.0.0:8080->8080/tcp, :::8080->8080/tcp   withoutapparmor-nginx
f071b3befdcc   nginx     "/docker-entrypoint.…"   48 seconds ago   Up 47 seconds   0.0.0.0:80->80/tcp, :::80->80/tcp                      apparmor-nginx
root@ip-172-31-35-22:~#
```

Figure 9: Nginx containers running

3.2 Case-2 WordPress Application Evaluation

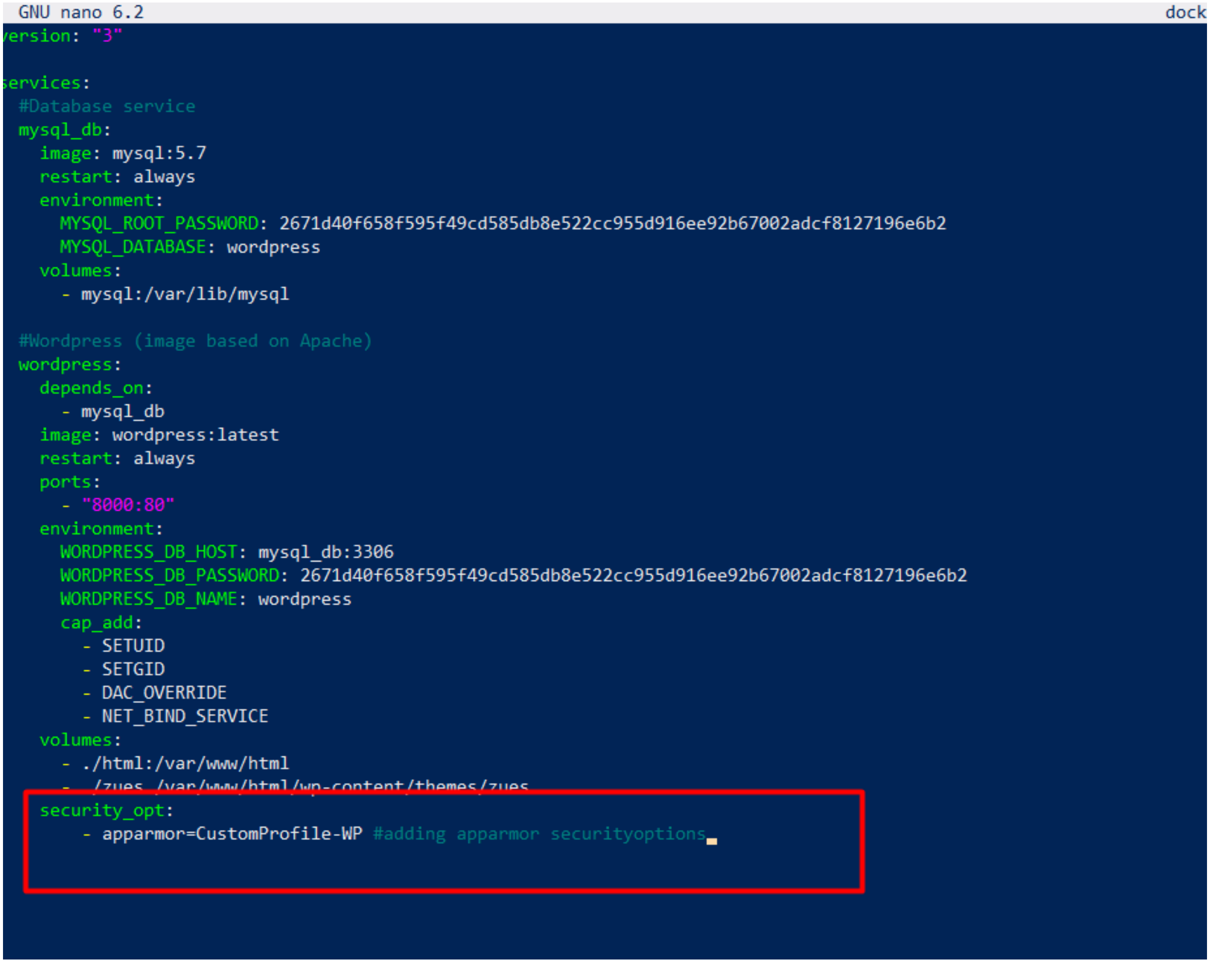
- Step 1: Created CustomAppArmor Profile for denying plugin upload for WordPress Application
  $ nano CustomProfile-WP Codefile attached seprately.

- Step 2:Load AppArmor profile in enforced mode.
  $ apparmor_parser -r -W /etc/apparmor.d/CustomProfile-WP

- Step 3: Add security option into WordPress dockercompose file.
  $ security_opt: - apparmor=CustomProfile-WP

- Step 4: Bring the WordPress Application up for Testing.
  $ docker-compose up

Fig 10 represents the security option of AppArmor profile added in docker compose file of WordPress installation in order to get the container started with secure profile.



```
 GNU nano 6.2                                                                          dock
version: "3"

services:
  #Database service
  mysql_db:
    image: mysql:5.7
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: 2671d40f658f595f49cd585db8e522cc955d916ee92b67002adcf8127196e6b2
      MYSQL_DATABASE: wordpress
    volumes:
      - mysql:/var/lib/mysql

  #Wordpress (image based on Apache)
  wordpress:
    depends_on:
      - mysql_db
    image: wordpress:latest
    restart: always
    ports:
      - "8000:80"
    environment:
      WORDPRESS_DB_HOST: mysql_db:3306
      WORDPRESS_DB_PASSWORD: 2671d40f658f595f49cd585db8e522cc955d916ee92b67002adcf8127196e6b2
      WORDPRESS_DB_NAME: wordpress
      cap_add:
        - SETUID
        - SETGID
        - DAC_OVERRIDE
        - NET_BIND_SERVICE
    volumes:
      - ./html:/var/www/html
      - ./zues:/var/www/html/wp-content/themes/zues
    security_opt:
        - apparmor=CustomProfile-WP #adding apparmor securityoptions
```

Figure 10: Security option added in WordPress Docker Compose Application

References

Javed, O. and Toor, S. (2021). An evaluation of container security vulnerability detection tools, pp. 95–101.

Mesa, O., Vieira, R., Viana, M., Durelli, V. H. S., Cirilo, E., Kalinowski, M. and Lucena, C. (2018). Understanding vulnerabilities in plugin-based web systems: An exploratory study of wordpress, *Proceedings of the 22nd International Systems and Software Product Line Conference - Volume 1*, SPLC '18, Association for Computing Machinery, New York, NY, USA, p. 149–159.
URL: *https://doi.org/10.1145/3233027.3233042*

Zhu, H. and Gehrmann, C. (2021). Lic-sec: An enhanced apparmor docker security profile generator, *Journal of Information Security and Applications* **61**: 102924.
URL: *https://www.sciencedirect.com/science/article/pii/S2214212621001435*