National College of Ireland

Performance Evaluation of Various Container Runtimes and Process ID Based Escape Detection

MSc Research Project
Msc In Cloud Computing

Jogindersingh Ramani
Student ID: 22129588

School of Computing
National College of Ireland

Supervisor: Dr. Rashid Mijumbi

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Jogindersingh Ramani
Student ID:	22129588
Programme:	Cloud Computing
Year:	2023/2024
Module:	MSc Research Project
Supervisor:	Rashid Mijumbi
Submission Due Date:	14/12/2023
Project Title:	Performance Evaluation of Various Container Runtimes and Process ID Based Escape Detection
Word Count:	2110
Page Count:	11

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:	Jogindersingh Ramani
Date:	13th December 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

Jogindersingh Ramani
x2212958

1 Introduction

This configuration manual provides detailed steps to set up the testing environment and tools required to perform performance benchmarking of container runtimes and experiments related to the research paper - "Performance Evaluation of Various Container Runtimes and Process ID Based Escape Detection". Critical runtimes like Docker, Gvisor, Kata, and Youki should be installed with benchmarking tools like Hyperfine and Sysbench. Jenkins is also used for the detection of container escape through pipelines. This manual covers various sections like launching EC2 instances, installing prerequisites, executing benchmark tests, and creating Jenkins jobs for automation. Following this manual will help to quickly set up the required tools and infrastructure to run experiments and evaluate the performance of various container runtimes.

2 Launching EC2 instance in AWS

In this project, all the experimentation and setup are done on the AWS EC2 instance. So here are the below steps to launch an AWS instance. So login to the AWS console, search EC2 and click it, and go to instances. Then click "Launch Instance", and it will open a new page asking for all the information like name, OS AMI(Amazon Machine Image), Instance type, Key pair (used for login into the server), Network settings, storage, and advanced settings. So type the name and select Ubuntu as OS in the ami section, and in configure storage, make it 20 and choose instance type according to the experiment needs. Finally, click on the launch instance, which will redirect to the ec2 instance page, where you can see your instance. One can refer to the AWS Documentation for detailed steps to launch an EC2 instance.

Documentation Link: `https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EC2_GetStarted.html`

3 Prerequisite

Below are the steps to install the necessary prerequisites that are required to run experiments.

3.1 Docker Installation

To install docker Docker (2020) login to the AWS instance that was launched and run the below commands.

- Setting up the apt repository:

```
# Add Docker's official GPG key:
sudo apt-get update
sudo apt-get install ca-certificates curl gnupg
sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | \
    sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
sudo chmod a+r /etc/apt/keyrings/docker.gpg

# Add the repository to Apt sources:
echo "deb [arch=$(dpkg --print-architecture) \
    signed-by=/etc/apt/keyrings/docker.gpg] \
    https://download.docker.com/linux/ubuntu \
    $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update
```

- Install docker related packages.

```
sudo apt-get install docker-ce docker-ce-cli \
    containerd.io docker-buildx-plugin docker-compose-plugin
```

- Verify installation by running the below command.

```
sudo docker run hello-world
```

3.2 Gvisor Installation

To install Gvisor login to the ec2 instance that was launched and run the following commands.

- Install the necessary package that will be required to install Gvisor gVisor Project (2023).

```
sudo apt-get update && sudo apt-get install -y \
    apt-transport-https && ca-certificates && curl && gnupg
```

- Prepare the apt repository to install and download Gvisor.

```
curl -fsSL https://gvisor.dev/archive.key | \
    sudo gpg --dearmor -o /usr/share/keyrings/gvisor-archive-keyring.gpg
echo "deb [arch=$(dpkg --print-architecture) \
    signed-by=/usr/share/keyrings/gvisor-archive-keyring.gpg] \
    https://storage.googleapis.com/gvisor/releases release main" | \
    sudo tee /etc/apt/sources.list.d/gvisor.list > /dev/null
```

- Install Gvisor by running the below command and verify the installation

```
sudo apt-get update && sudo apt-get install -y runsc
sudo systemctl reload docker
docker run --rm --runtime=runsc hello-world
```

3.3 Kata Container installation

To install Kata Container Kata Containers Project (2023), log in to the ec2 instance that was launched and run the following commands. Also, install docker as mentioned in the above steps. Use the following commands to install and verify kata containers.

```
$ ARCH=$(arch)
$ BRANCH="${BRANCH:-master}"
$ sudo sh -c "echo 'deb http://download.opensuse.org/repositories/home:/\
katacontainers:/releases:/${ARCH}:/${BRANCH}/xUbuntu_$(lsb_release -rs)/ /' \
> /etc/apt/sources.list.d/kata-containers.list"\
$ curl -sL http://download.opensuse.org/repositories/home:/katacontainers:\
/releases:/${ARCH}:/${BRANCH}/xUbuntu_$(lsb_release -rs)/Release.key \
| sudo apt-key add -
$ sudo -E apt-get update
$ sudo -E apt-get -y install kata-runtime kata-proxy kata-shim
$ kata-runtime check
```

3.4 Youki Installation

To install Youki Youki Project (2023), log in to the ec2 instance that was launched and run the following commands. Also, install docker as mentioned in the above steps. Rust installation is also needed as Youki is built on Rust language.

- Installing rust on the Ubuntu ec2 server using the following commands. After running the below command enter 1 and press enter.

  ```
  curl --proto '=https' --tlsv1.3 https://sh.rustup.rs -sSf | sh
  source $HOME/.cargo/env
  rustc --version
  ```

- Installation of Compiler and necessary packages.

  ```
  sudo apt update && sudo apt upgrade && sudo apt install build-essential\
  git clone 'https://mpr.makedeb.org/just' && cd just && makedeb -si \
  export PATH="$PATH:$HOME/bin" && just --help
  ```

- Additional packages required.

  ```
  $ sudo apt-get install && pkg-config && libsystemd-dev && build-essential \
  libelf-dev && libseccomp-dev && libclang-dev && glibc-static && libssl-dev
  ```

- Install Youki using the below commands.

  ```
   git clone https://github.com/containers/youki.git
  cd youki && just youki-dev # or youki-release
  ./youki -h
  ```

3.5 Quark Container Installation

To install Quark Container QuarkContainer Project (2023), log in to the ec2 instance that was launched and run the following commands. Also, install docker and rust as mentioned in the above steps.

- Install prerequisite which is required to install quark containers.

```
rustup toolchain install nightly-2022-08-11-x86_64-unknown-linux-gnu
rustup default nightly-2022-08-11-x86_64-unknown-linux-gnu
sudo apt-get install libcap-dev
sudo apt-get install build-essential cmake gcc libudev-dev \
    libnl-3-dev libnl-route-3-dev ninja-build pkg-config valgrind \
    python3-dev cython3 python3-docutils pandoc libclang-dev
rustup component add rust-src
cargo install cargo-xbuild
```

- Build and install quark containers.

```
git clone https://github.com/QuarkContainer/Quark.git
cd Quark && make && make install
sudo mkdir /var/log/quark && sudo systemctl restart docker
```

- Check and verify the installation.

```
sudo systemctl restart docker
sudo systemctl restart docker.service
docker run --rm --runtime=quark hello-world
```

3.6 Benchmarking Tools Installation

In this project, two benchmarking tools are used which are hyperfine and sysbench. Its installation steps are given below.

- Hyperfine sharkdp (2023) Installation steps:

```
wget https://github.com/sharkdp/hyperfine/releases/download/v1.16.1/\
hyperfine_1.16.1_amd64.deb
sudo dpkg -i hyperfine_1.16.1_amd64.deb
hyperfine --version
```

- Sysbench Kopytov (2004) installation steps:

```
wget -qO - https://packagecloud.io/install/repositories/akopytov/sysbench/\
script.deb.sh | sudo bash
sudo apt install -y sysbench
sysbench --version
```

3.7 Jenkins installation

To install Jenkins, log in to the ec2 instance that was launched and run the following commands.

```
sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \
  https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
  https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
  /etc/apt/sources.list.d/jenkins.list > /dev/null
sudo apt-get update
sudo apt-get install jenkins
sudo apt install fontconfig openjdk-17-jre
java -version
sudo systemctl enable jenkins
sudo systemctl start jenkins
sudo systemctl status jenkins
```

Once the installation process is completed the Jenkins console can be accessed through a browser with the URL `http://you_servers_publicip:8080/`. After loading this URL one will get the below screen to unlock Jenkins. Get your initial password from the location mentioned on the page and set up your initial username and password. Once you log in you can access various functionality of Jenkins.

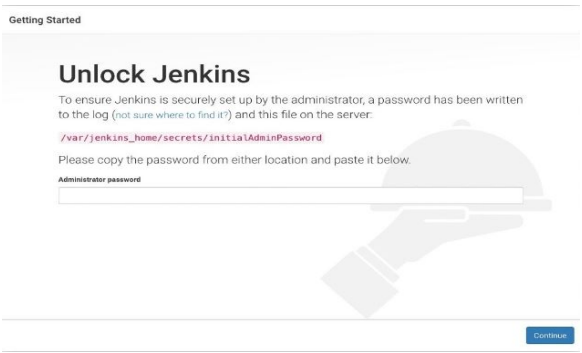

Figure 1: Unlock Jenkins getting the initial password.

3.8 Installation of GRYPE

Grype is the vulnerability scanner that is used to scan the docker images Kalaiselvi et al. (2023). The output shows several vulnerabilities that are high, critical, and low.

```
    curl -sSfL https://raw.githubusercontent.com/anchore/grype/main/\
install.sh | sh -s -- -b /usr/local/bin
```

To test the installation run the below command which will do a vulnerability test on the docker image. This can be integrated with Jenkins by adding a stage for it.

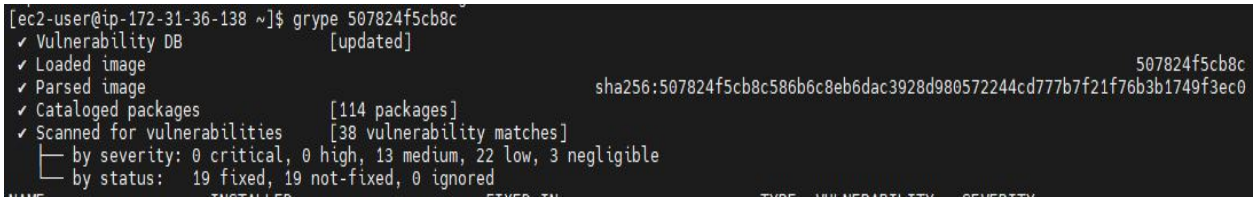

Figure 2: Grype installation working.

3.9 Build a Docker image that will be used everywhere in the test

Below is the Dockerfile that will be used to create a Docker image. This image will be used for benchmarking and also experiments that are used to detect container escape.

```
FROM busybox:latest
RUN apt-get update && apt-get install -y procps
```

Build this docker image using using the below command.

```
docker build -t imageName:imageTag -f Dockerfile .
```

4 Running Benchmarks

Below are the commands that are used to run various benchmarking by logging into instances where runtimes are installed.

4.0.1 Benchmarking using sysbench

The sysbench is used for introducing various workloads on a runtime that will generate metrics.

- CPU Benchmarking: Using this command the performance of the CPU under the computation load. The workload is given by performing the task of calculating prime numbers up to 20000. It will return various parameters which are used for evaluation in the research.
- Memory Benchmarking: Sysbench does memory benchmarking by using 1KB size and allocating a total of 10GB of memory to evaluate system memory performance.

  ```
  docker run -it ubuntu:sysbench sysbench memory --memory-block-size=1K\
  --memory-total-size=10G run
  ```

- I/O Benchmarking: The below command is used to perform random read write operation of around 10GB in three stages which are prepare, run, and cleanup.

  ```
  docker run ubuntu:sysbench sysbench fileio --file-total-size=10G \
  --file-test-mode=rndrw prepare
  docker run ubuntu:sysbench sysbench fileio --file-total-size=10G \
  ```

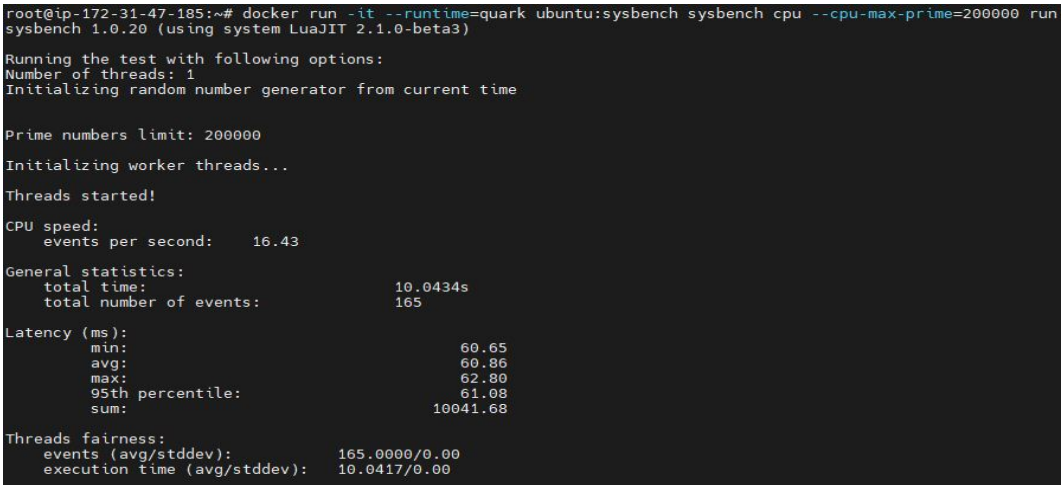

Figure 3: CPU benchmarking using sysbench.

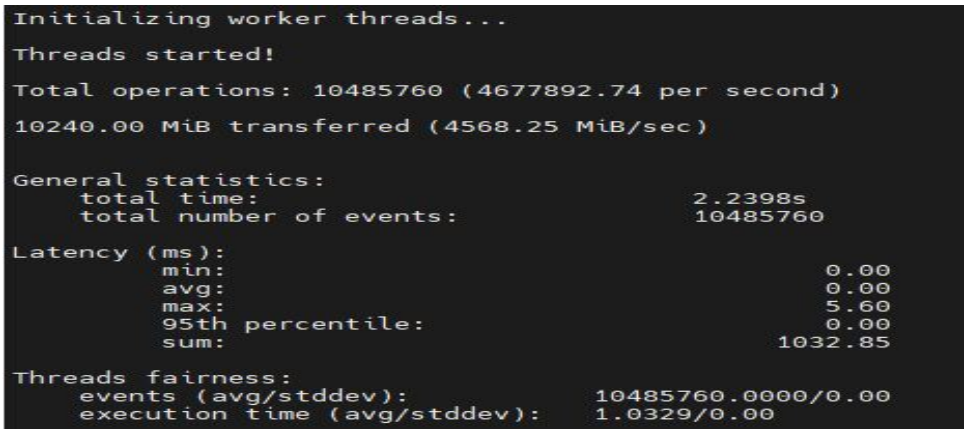

Figure 4: Memory benchmarking using sysbench.

```
--file-test-mode=rndrw run
docker run ubuntu:sysbench sysbench fileio --file-total-size=10G \
--file-test-mode=rndrw cleanup
```

- Container Startup: The below command is used to measure the container startup time using the docker command.

```
date +%s%N; docker run --rm -it ubuntu /bin/date +%s%N
```

4.1 Benchmarking using Hyperfine

It is a built-in Linux command line utility that is used for benchmarking by running scripts multiple times. For Container lifecycle evaluation multiple three scripts were made one was used to build docker images, the other was used to run the container from the docker image and the last script was to kill the running container. This was done using the below command.

```
hyperfine --prepare 'sudo sync; echo 3 | sudo tee /proc/sys/vm/drop_caches'\
```

```
--warmup 10 --min-runs 100 'sudo ./build.sh &&\
sudo ./run.sh && sudo ./delete.sh' --show-output

 Time (mean ± $\sigma$): 1.836 s ±  0.079 s [User: 0.219 s, System: 0.166 s]
 Range (min ... max): 1.670 s ... 2.047 s 100 runs
```

5 Jenkins Job creation

In this section, the algorithms that are proposed are integrated with Jenkins. After logging into Jenkins one will land on the login page where the Jenkins job can be created. After clicking on a new item, it will ask the name of the pipeline. After that click on the pipeline, it will open a job configuration page and in the pipeline section add the below pipeline code which is written in Grovvy.

Code:-

```
pipeline {
    agent any
    environment {
        DOCKER_IMAGE_ID = ""  // Declare an environment variable
           to store the image ID
    }
    stages {
        stage('Checkout') {
            steps {
                // Checkout code from Git repository
                git branch: 'main', url:
                   'https://github.com/jogindersingh1913/RIC-SEM3.git'
            }
        }

        stage('Build') {
            steps {
                script {
                    // Build the Docker image and capture the
                       image ID\
                    //Test case 1
                    DOCKER_IMAGE_ID = sh(script: 'docker␣build␣
                       -t␣busybox:latest␣-f␣ubuntu.Dockerfile␣.␣
                       -q', returnStdout: true).trim()
                    //test case 2
                    // DOCKER_IMAGE_ID = sh(script: 'docker␣
                       build␣-t␣busybox:latest␣-f␣
                       ubuntu_escaped.Dockerfile␣.␣-q',
                       returnStdout: true).trim()
                    // test case 4
                    // DOCKER_IMAGE_ID = sh(script: 'docker␣
                       build␣-t␣busybox:latest␣-f␣
```

```
                nginx.Dockerfile␣.␣-q', returnStdout:
                true).trim()
            echo "Docker␣Image␣ID:␣${DOCKER_IMAGE_ID}"
        }
    }
}

stage('Scan␣with␣Grype') {
    steps {
        script {
            // Update Grype database
            sh 'grype␣db␣update'
            // Scan the Docker image using the captured
               image ID
            sh "grype␣${DOCKER_IMAGE_ID}"
            //test case 4
            // sh "grype␣${DOCKER_IMAGE_ID}␣␣-f␣Critical␣
               >=1"
        }
    }
}

stage('Run') {
    steps {
        script {
            // Run a test container
            // test case 1
            sh 'docker␣run␣-itd␣busybox:latest'

            //test case 2 escape container
            // sh 'docker␣run␣-d␣-v␣/:/host/␣
               --cap-add=ALL␣--security-opt␣
               apparmor=unconfined␣--security-opt␣
               seccomp=unconfined␣--security-opt␣
               label:disable␣--pid=host␣--userns=host␣
               --uts=host␣--cgroupns=host␣busybox:latest␣
               chroot␣/host/␣bash'
            sh 'docker␣ps'
        }
    }
}

stage('Check␣Container␣Escape') {
    steps {
        script {
            // Check container escape
            // sh 'sh␣cescapeByPid.sh'
            sh 'sh␣containerEscape.sh'
        }
    }
}
```

```
        stage('other checks') {
            steps {
                script {
                    // Additional checks
                    // sh 'sh otherparaescapes.sh'
                    echo 'container intact'
                }
            }
        }
    }

    post {
        always {
            // Stop and remove all containers
            sh 'docker stop $(docker ps -q)'
            sh 'docker rm $(docker ps -qa)'
        }
    }
}
```

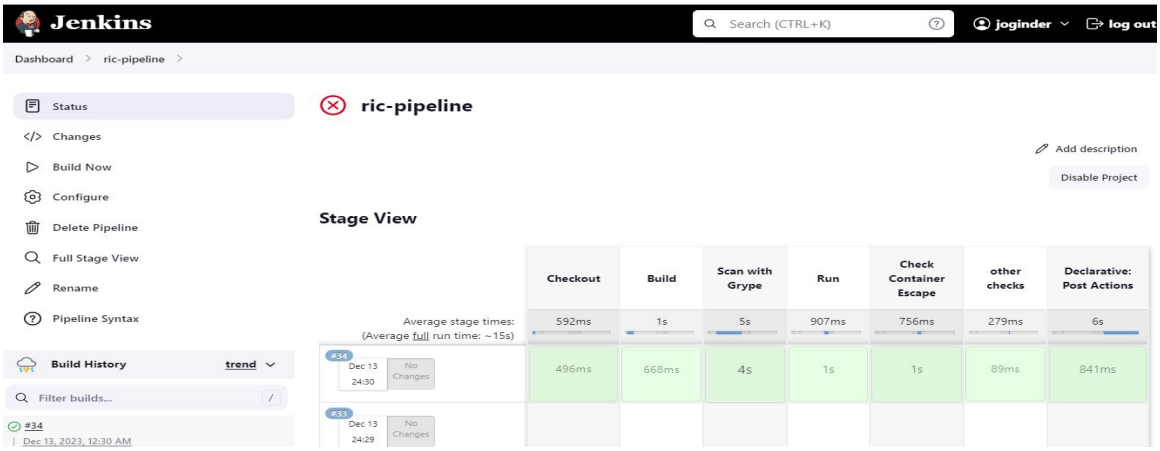

Figure 5: Jenkins pipeline

After adding this code click on save and to run the pipeline just click on build and it will run the algorithm.

References

Docker, I. (2020). Docker, *línea].[Junio de 2017]. Disponible en: https://www. docker. com/what-docker* .

gVisor Project (2023). gVisor User Guide - Installation. Accessed 13 December 2023. **URL:** *https://gvisor.dev/docs/user*$_g$*uide/install/*

Kalaiselvi, R., Ravisankar, S., Varun, M. and Ravindran, D. (2023). Enhancing the container image scanning tool-grype, *2023 2nd International Conference on Advancements*

in Electrical, Electronics, Communication, Computing and Automation (ICAECA), IEEE, pp. 1–6.

Kata Containers Project (2023). Kata Containers Documentation. Accessed 13 December 2023.
URL: *https://katacontainers.io/docs/*

Kopytov, A. (2004). Sysbench: a system performance benchmark, *http://sysbench. sourceforge. net/* .

QuarkContainer Project (2023). QuarkContainer GitHub Repository. Accessed 13 December 2023.
URL: *https://github.com/QuarkContainer/Quark*

sharkdp (2023). hyperfine: A command-line benchmarking tool. Accessed 13 December 2023.
URL: *https://github.com/sharkdp/hyperfine*

Youki Project (2023). Youki Documentation - Basic Setup. Accessed 13 December 2023.
URL: *https://containers.github.io/youki/user/basic*$_s$*etup.html*