Audrey Long

JHU Web Security

Container Assignment

02/23/2021

JHU WebSec Application Container Security Module Assignment

Introduction

This week we will look at a container security scanning tool called Anchore. We will also use Docker to run the tool inside of a container! I hope it will help get you some hands-on experience Docker and an appreciate for, although containers are simple to use, their security cannot be neglected.

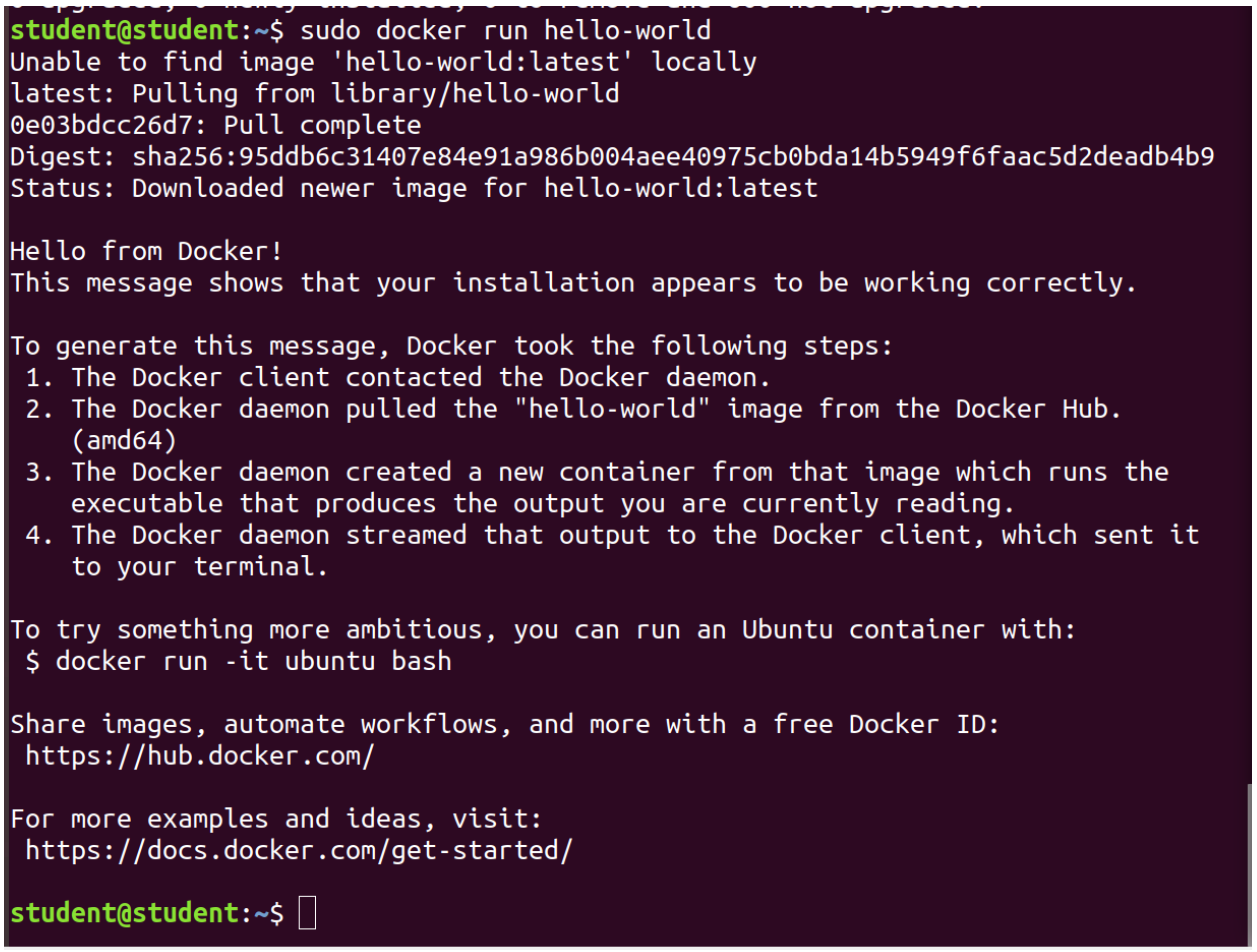
Use the VM setup:

You may use the previous VM for this assignment. Optionally you can set up a new VM, begin by downloading the course VM (it is a fresh Ubuntu install, feel free to rename the VM after you download and import it) and importing it into Oracle's VirtualBox. Once download, open VirtualBox and Click 'Machine' -> 'Add' to import the VM you downloaded.

A. Installing Docker

Please follow the instructions for installing Docker CE (Community Edition) on Ubuntu 18 found here: https://docs.docker.com/engine/install/ubuntu/ You can copy-and-paste the instructions from the ¨Install using the repository¨ section directly into your Terminal window. You can skip the command for installing a specific version of Docker and just install the latest stable release.

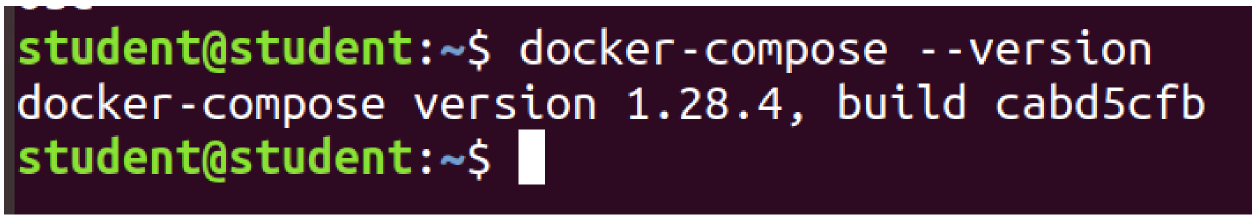
Deliverable: Please provide a screenshot of the output from the command: sudo docker run hello-world.



```
student@student:~$ sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
0e03bdcc26d7: Pull complete
Digest: sha256:95ddb6c31407e84e91a986b004aee40975cb0bda14b5949f6faac5d2deadb4b9
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/

student@student:~$
```

Figure 1: Hello World Docker

You will also need to install Docker Compose which is an orchestration tool for configuring multiple containers a once. For example, the Anchore Engine will require the Anchore container as well as a PostgreSQL database. Docker Compose will help make the installation a cinch. Please follow Docker's instruction for installing Compose here: https://docs.docker.com/compose/install/ Click the 'Linux' tab for installation instructions for Ubuntu.



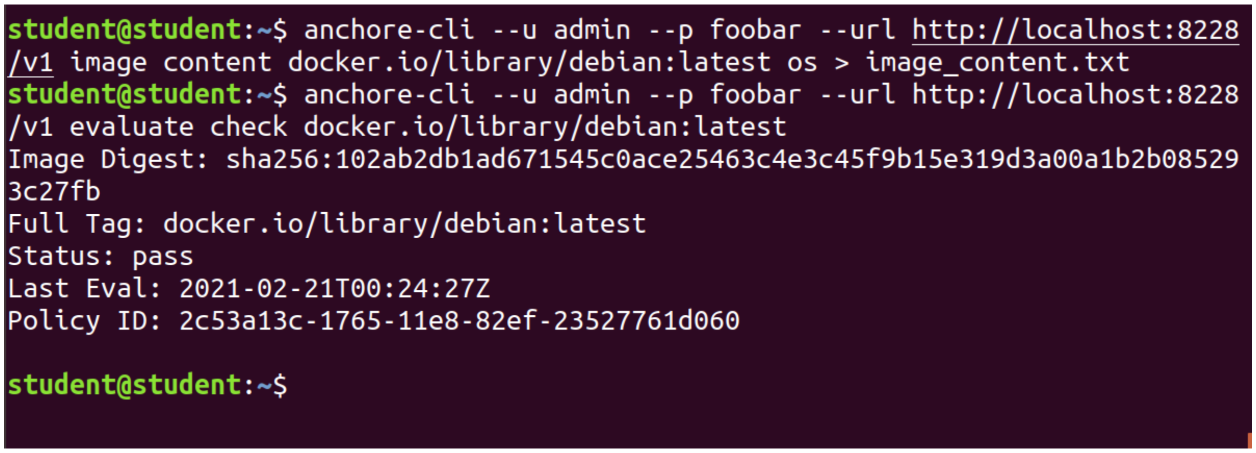
```
student@student:~$ docker-compose --version
docker-compose version 1.28.4, build cabd5cfb
student@student:~$
```

Figure 2: Docker Compose Version

B. Installing Anchore Engine & the Anchore CLI

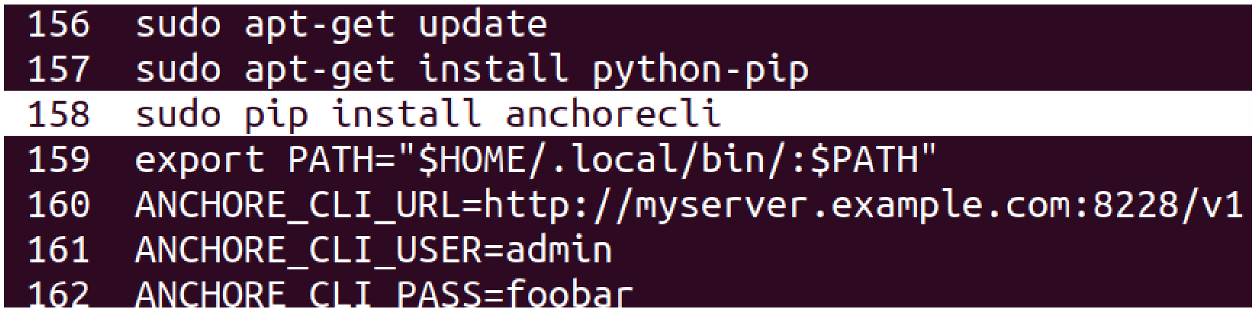
Next you will install Anchore Engine, the actual container security scanner, and it is command line API. Start by installing Anchore Engine using Docker Compose: https://github.com/anchore/anchore-engine by completing the Installation, Configuration, and Running Docker Engine using Docker Compose sections.

Deliverable: Please provide a screenshot showing the successful completion of the "docker-compose up" command.



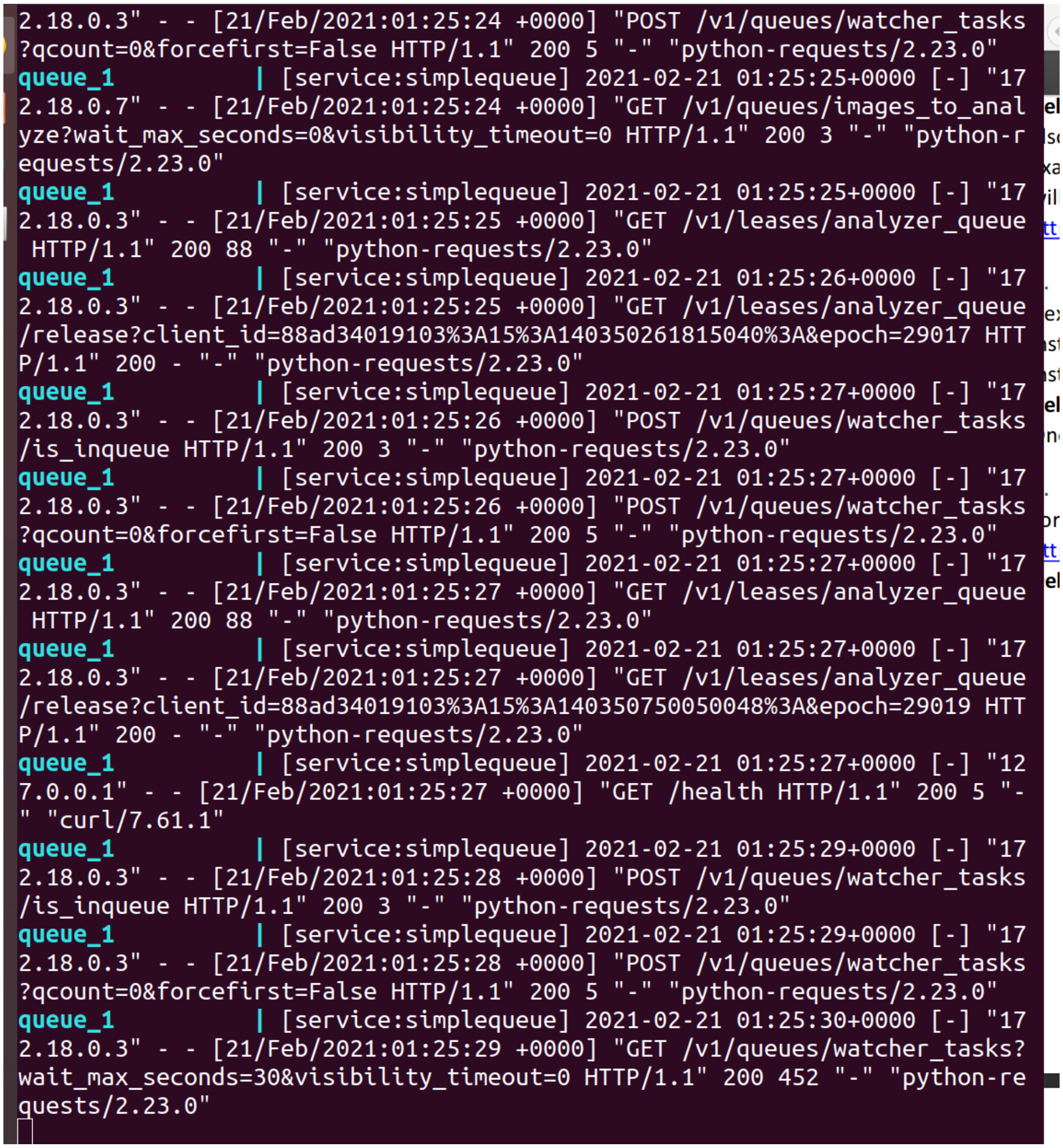
```
student@student:~$ anchore-cli --u admin --p foobar --url http://localhost:8228
/v1 image content docker.io/library/debian:latest os > image_content.txt
student@student:~$ anchore-cli --u admin --p foobar --url http://localhost:8228
/v1 evaluate check docker.io/library/debian:latest
Image Digest: sha256:102ab2db1ad671545c0ace25463c4e3c45f9b15e319d3a00a1b2b08529
3c27fb
Full Tag: docker.io/library/debian:latest
Status: pass
Last Eval: 2021-02-21T00:24:27Z
Policy ID: 2c53a13c-1765-11e8-82ef-23527761d060

student@student:~$
```

Figure 3: Anchore evaluate check



```
 156  sudo apt-get update
 157  sudo apt-get install python-pip
 158  sudo pip install anchorecli
 159  export PATH="$HOME/.local/bin/:$PATH"
 160  ANCHORE_CLI_URL=http://myserver.example.com:8228/v1
 161  ANCHORE_CLI_USER=admin
 162  ANCHORE_CLI_PASS=foobar
```

Figure 4: proof of anchorcli installation



```
2.18.0.3" - - [21/Feb/2021:01:25:24 +0000] "POST /v1/queues/watcher_tasks
?qcount=0&forcefirst=False HTTP/1.1" 200 5 "-" "python-requests/2.23.0"
queue_1           | [service:simplequeue] 2021-02-21 01:25:25+0000 [-] "17
2.18.0.7" - - [21/Feb/2021:01:25:24 +0000] "GET /v1/queues/images_to_anal
yze?wait_max_seconds=0&visibility_timeout=0 HTTP/1.1" 200 3 "-" "python-r
equests/2.23.0"
queue_1           | [service:simplequeue] 2021-02-21 01:25:25+0000 [-] "17
2.18.0.3" - - [21/Feb/2021:01:25:25 +0000] "GET /v1/leases/analyzer_queue
 HTTP/1.1" 200 88 "-" "python-requests/2.23.0"
queue_1           | [service:simplequeue] 2021-02-21 01:25:26+0000 [-] "17
2.18.0.3" - - [21/Feb/2021:01:25:25 +0000] "GET /v1/leases/analyzer_queue
/release?client_id=88ad34019103%3A15%3A140350261815040%3A&epoch=29017 HTT
P/1.1" 200 - "-" "python-requests/2.23.0"
queue_1           | [service:simplequeue] 2021-02-21 01:25:27+0000 [-] "17
2.18.0.3" - - [21/Feb/2021:01:25:26 +0000] "POST /v1/queues/watcher_tasks
/is_inqueue HTTP/1.1" 200 3 "-" "python-requests/2.23.0"
queue_1           | [service:simplequeue] 2021-02-21 01:25:27+0000 [-] "17
2.18.0.3" - - [21/Feb/2021:01:25:26 +0000] "POST /v1/queues/watcher_tasks
?qcount=0&forcefirst=False HTTP/1.1" 200 5 "-" "python-requests/2.23.0"
queue_1           | [service:simplequeue] 2021-02-21 01:25:27+0000 [-] "17
2.18.0.3" - - [21/Feb/2021:01:25:27 +0000] "GET /v1/leases/analyzer_queue
 HTTP/1.1" 200 88 "-" "python-requests/2.23.0"
queue_1           | [service:simplequeue] 2021-02-21 01:25:27+0000 [-] "17
2.18.0.3" - - [21/Feb/2021:01:25:27 +0000] "GET /v1/leases/analyzer_queue
/release?client_id=88ad34019103%3A15%3A140350750050048%3A&epoch=29019 HTT
P/1.1" 200 - "-" "python-requests/2.23.0"
queue_1           | [service:simplequeue] 2021-02-21 01:25:27+0000 [-] "12
7.0.0.1" - - [21/Feb/2021:01:25:27 +0000] "GET /health HTTP/1.1" 200 5 "-
" "curl/7.61.1"
queue_1           | [service:simplequeue] 2021-02-21 01:25:29+0000 [-] "17
2.18.0.3" - - [21/Feb/2021:01:25:28 +0000] "POST /v1/queues/watcher_tasks
/is_inqueue HTTP/1.1" 200 3 "-" "python-requests/2.23.0"
queue_1           | [service:simplequeue] 2021-02-21 01:25:29+0000 [-] "17
2.18.0.3" - - [21/Feb/2021:01:25:28 +0000] "POST /v1/queues/watcher_tasks
?qcount=0&forcefirst=False HTTP/1.1" 200 5 "-" "python-requests/2.23.0"
queue_1           | [service:simplequeue] 2021-02-21 01:25:30+0000 [-] "17
2.18.0.3" - - [21/Feb/2021:01:25:29 +0000] "GET /v1/queues/watcher_tasks?
wait_max_seconds=30&visibility_timeout=0 HTTP/1.1" 200 452 "-" "python-re
quests/2.23.0"
```

Figure 5: Output from the docker-compose up command

Once complete, install the Anchore CLI tools: ¨ https://github.com/anchore/anchore-cli .

C. Scanning an Image with Anchore Engine

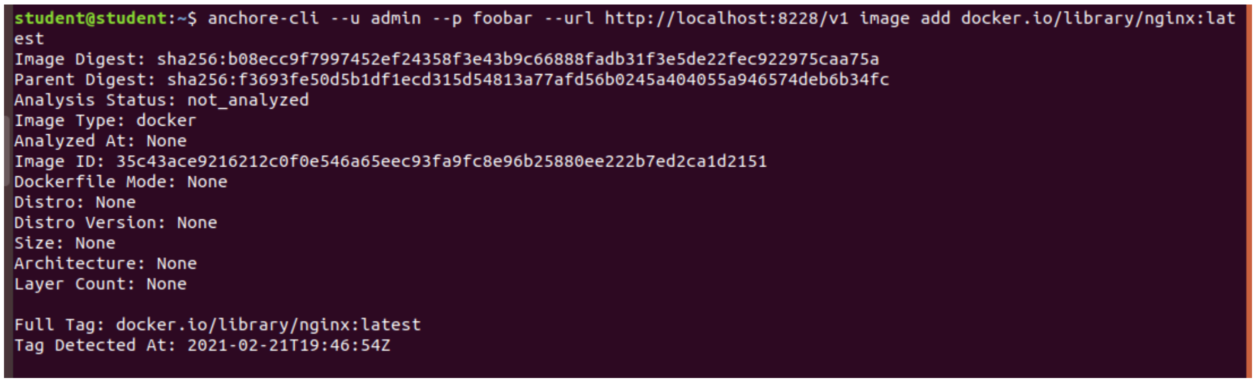
For the last part of the assignment, go back to the Anchore Engine installation instructions:

https://github.com/anchore/anchore-engine/wiki and complete the section entitled Getting Started using the CLI.

Deliverable: After setting the 3 environment variables,

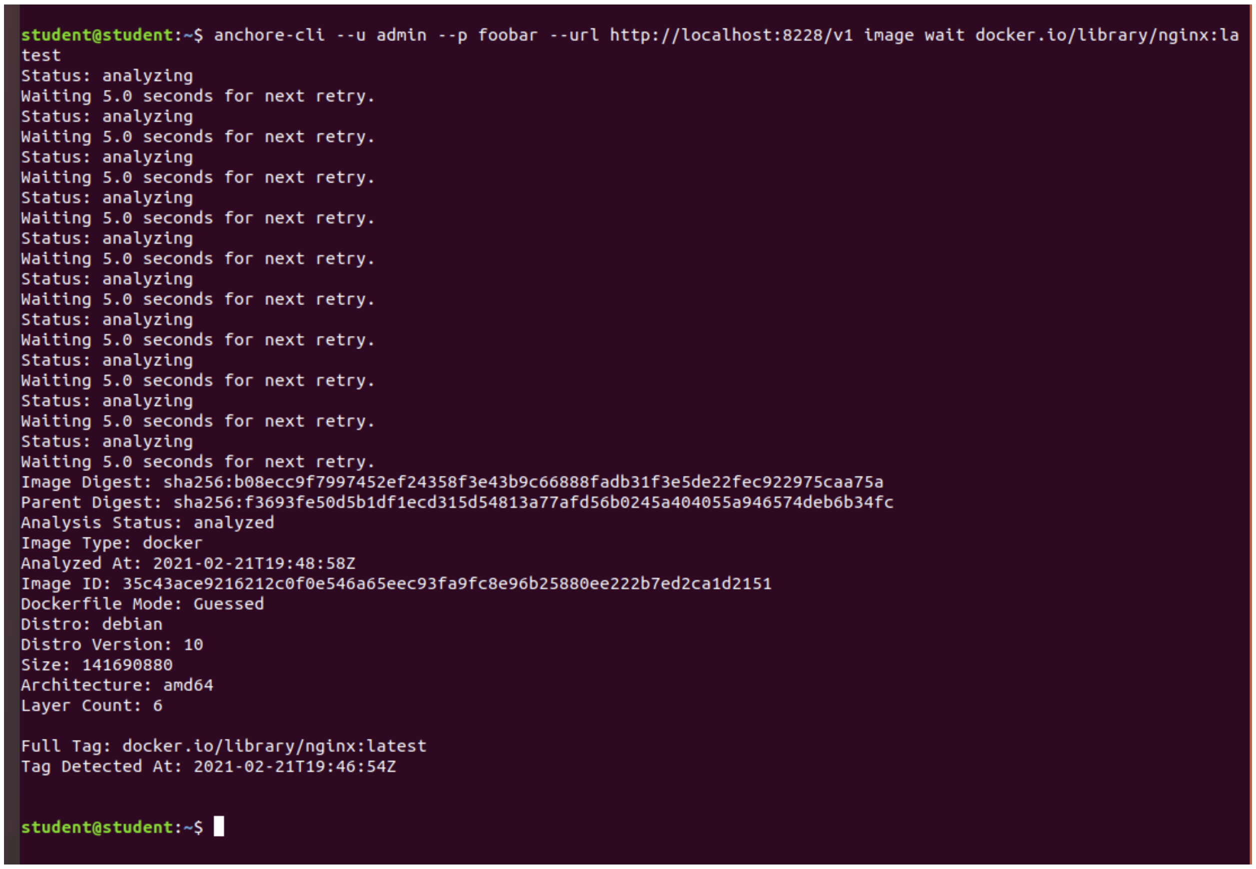
1. provide a screenshot of the successful completion:

a. Use the 'add' command to import an image of your choice (please use an image other than the Debian one in the example).



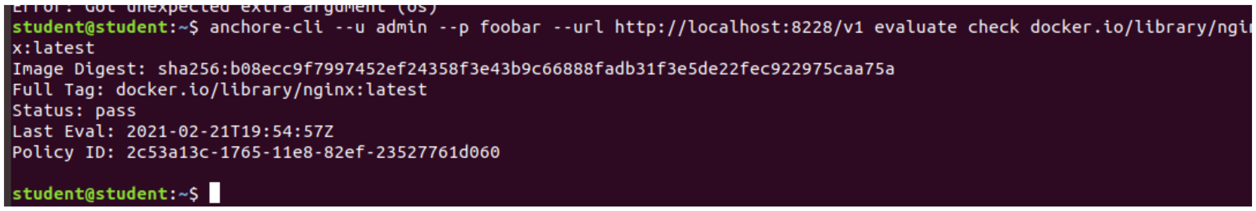
```
student@student:~$ anchore-cli --u admin --p foobar --url http://localhost:8228/v1 image add docker.io/library/nginx:lat
est
Image Digest: sha256:b08ecc9f7997452ef24358f3e43b9c66888fadb31f3e5de22fec922975caa75a
Parent Digest: sha256:f3693fe50d5b1df1ecd315d54813a77afd56b0245a404055a946574deb6b34fc
Analysis Status: not_analyzed
Image Type: docker
Analyzed At: None
Image ID: 35c43ace9216212c0f0e546a65eec93fa9fc8e96b25880ee222b7ed2ca1d2151
Dockerfile Mode: None
Distro: None
Distro Version: None
Size: None
Architecture: None
Layer Count: None

Full Tag: docker.io/library/nginx:latest
Tag Detected At: 2021-02-21T19:46:54Z
```

Figure 6: adding the nginx image



```
student@student:~$ anchore-cli --u admin --p foobar --url http://localhost:8228/v1 image wait docker.io/library/nginx:la
test
Status: analyzing
Waiting 5.0 seconds for next retry.
Status: analyzing
Waiting 5.0 seconds for next retry.
Status: analyzing
Waiting 5.0 seconds for next retry.
Status: analyzing
Waiting 5.0 seconds for next retry.
Status: analyzing
Waiting 5.0 seconds for next retry.
Status: analyzing
Waiting 5.0 seconds for next retry.
Status: analyzing
Waiting 5.0 seconds for next retry.
Status: analyzing
Waiting 5.0 seconds for next retry.
Status: analyzing
Waiting 5.0 seconds for next retry.
Status: analyzing
Waiting 5.0 seconds for next retry.
Image Digest: sha256:b08ecc9f7997452ef24358f3e43b9c66888fadb31f3e5de22fec922975caa75a
Parent Digest: sha256:f3693fe50d5b1df1ecd315d54813a77afd56b0245a404055a946574deb6b34fc
Analysis Status: analyzed
Image Type: docker
Analyzed At: 2021-02-21T19:48:58Z
Image ID: 35c43ace9216212c0f0e546a65eec93fa9fc8e96b25880ee222b7ed2ca1d2151
Dockerfile Mode: Guessed
Distro: debian
Distro Version: 10
Size: 141690880
Architecture: amd64
Layer Count: 6

Full Tag: docker.io/library/nginx:latest
Tag Detected At: 2021-02-21T19:46:54Z


student@student:~$ 
```

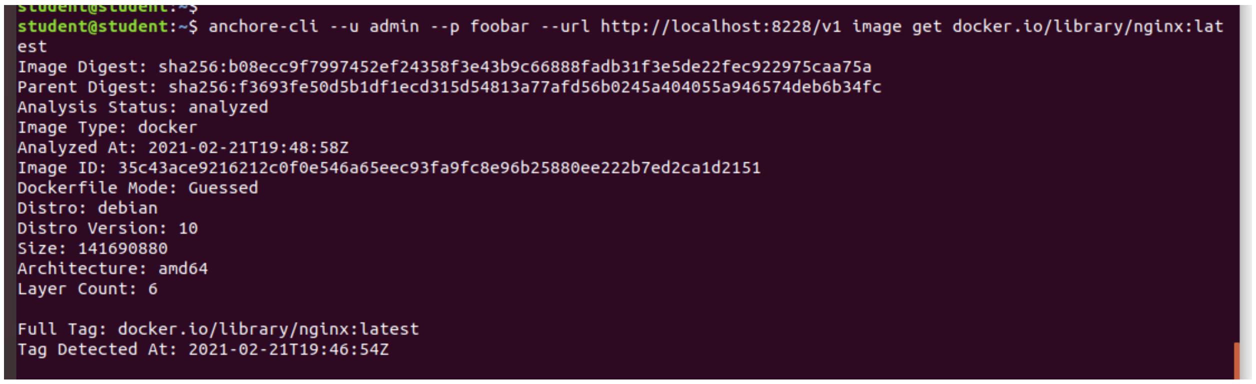
Figure 6: image wait on the nginx image



```
Error: Got unexpected extra argument (os)
student@student:~$ anchore-cli --u admin --p foobar --url http://localhost:8228/v1 evaluate check docker.io/library/ngi
x:latest
Image Digest: sha256:b08ecc9f7997452ef24358f3e43b9c66888fadb31f3e5de22fec922975caa75a
Full Tag: docker.io/library/nginx:latest
Status: pass
Last Eval: 2021-02-21T19:54:57Z
Policy ID: 2c53a13c-1765-11e8-82ef-23527761d060

student@student:~$ 
```

Figure 7: evaluate check on the nginx image

b. Use the ’get’ command to see when the image scan completes.



```
student@student:~$ anchore-cli --u admin --p foobar --url http://localhost:8228/v1 image get docker.io/library/nginx:lat
est
Image Digest: sha256:b08ecc9f7997452ef24358f3e43b9c66888fadb31f3e5de22fec922975caa75a
Parent Digest: sha256:f3693fe50d5b1df1ecd315d54813a77afd56b0245a404055a946574deb6b34fc
Analysis Status: analyzed
Image Type: docker
Analyzed At: 2021-02-21T19:48:58Z
Image ID: 35c43ace9216212c0f0e546a65eec93fa9fc8e96b25880ee222b7ed2ca1d2151
Dockerfile Mode: Guessed
Distro: debian
Distro Version: 10
Size: 141690880
Architecture: amd64
Layer Count: 6

Full Tag: docker.io/library/nginx:latest
Tag Detected At: 2021-02-21T19:46:54Z
```

Figure 8: image get on the nginx image

2. Use the 'vuln' command to show all OS vulnerabilities, select two of the vulnerabilities (CVE), look them up, and briefly describe the vulnerabilities.

student@student:~$ anchore-cli --u admin --p foobar --url http://localhost:8228/v1 image vuln docker.io/library/nginx:latest os
Vulnerability ID Package Severity Fix CVE Refs Vulnerability URL Type Feed Group Package Path
CVE-2011-3389 libgnutls30-3.6.7-4+deb10u6 Medium None https://security-tracker.debian.org/tracker/CVE-2011-3389 dpkg debian:10 pkgdb
CVE-2019-1551 libssl1.1-1.1.1d-0+deb10u4 Medium 1.1.1d-0+deb10u5 https://security-tracker.debian.org/tracker/CVE-2019-1551 dpkg debian:10 pkgdb

I decided to check out the first two vulnerabilities that showed up in the scan. The first vulnerability CVE-2011-3389 is due to the SSL protocol which has a configuration susceptible to man in the middle attacks due to chained initialization vector. Essentially this allows MitM to obtain plaintext HTTP headers in an HTTPS session. The second vulnerability CVE 2019-1551 aligns to an overflow bug found in x86_64 Montgomery squaring procedure used in exponentiation with 512 bit moduli also used in the OpenSSL which can be used to compromise key materials.

D. Application of Knowledge

As the new CTO of the company, you want to improve security practices in the development pipeline, what would you recommend for action after you hear the following:

If I was a new CEO in a company and I hear about all of the complaints above the first thing I would do is assess the current situation with the development and engineer teams to see what the current process is to evaluate all the coding and security best practices, whats being tested, how its being tested, how its deployed and what procedures are in place. After collecting enough data I would then address each complaint bullet by bullet because in my opinion these can all be mitigated.

• “They don’t tell us about issues until it’s too late!”

This can be mitigated by adding tools into the CI/CD pipelines in either development or prod to detect vulnerabilities before reaching production. This can be done with image scanners, static code analysis tools, and memory/performance tools.

• “We have to do a ton of manual spreadsheets for them.”

Once we figure out why there are manual spreadsheets we can work together with a development team to figure out a way to automate this process.

• “The policy documents for what we can and can’t use are super confusing and not updated.”

Engage the cyber security team members to figure out how to set up policies in the pipeline through the use of policy orchestration tools to ensure this process is automated and alerting systems are in place.

The security testers raise issues about the development process, like:

• “Developers aren’t paying attention to what dependencies they use.”

• ‘”We’re not told about major changes to the software until it’s almost delivered.”

• “We’re not involved early enough in the process.”

These concerns can be mitigated by using the pipeline tools mentioned above. Also just some agile/management tools can connect these teams to work more harmoniously and more cohesively to ensure the product is at its more secure and working. It is also a good idea to hold some architecture planning meetings with all of the stakeholders to ensure everyone is on the same page.

Deliverables:

Upload a Word or PDF document with the following:

A. A screenshot from Step A of the output from command: sudo docker run hello-world.

B. A screenshot from Step B of the docker-compose up command.

C. A screenshots from Step C of successful completion of the 2 steps, and briefly describe two vulnerabilities.

D. A paragraph applying your knowledge to the scenario given.