# Prisma Public Cloud Free Public Scanning API Tutorial

Gunjan Patel

Cloud Architect, Palo Alto Networks



# Table of Contents

# Activity 1 – Container Image Scanning for Vulnerabilities

***In this activity, you will:***

- ***Scan your container images for security vulnerabilities using Prisma Public Cloud (formerly Redlock) free public APIs***
- ***Scan publicly available container images for security vulnerabilities***
- ***Patch the images***
- ***Push the patched image to your container registry***

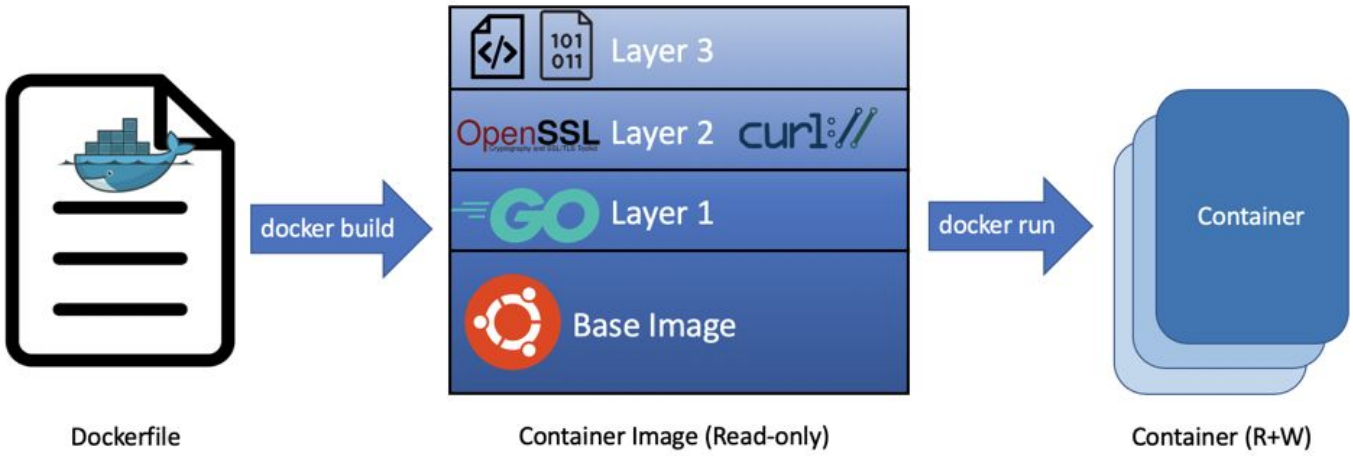
---

## What are container images?

A container image is a lightweight, standalone, executable packaging of software which includes everything needed to run an application: code, runtime, libraries and local configuration.
Container images are made up of different layers. Every container image has a base layer (parent layer) which is usually an Operating System. The subsequent layers are built on top of it which might include language runtimes, libraries, and code (file, executables), etc.
Each of the layers are immutable and built on top of the previous layer. These layers are independent of each other. For example OpenSSL can be installed on many different base images (OS). Most of the layers are reusable such as base layer, libraries, language runtimes, which are pulled from internal or external shared repositories such as DockerHub, GitHub, npm, etc.

## How are containers built?

**Dockerfile:**
Dockerfile is the manifest with build instructions on how to build a specific container image.
**Image:**
Container images are read-only templates from which containers are launched. Each image contains a series of layers as explained above.
**Container:**
A container is a running and mutable (Read + Write) form of the image.




## What does “scanning” my image mean?

**Prisma Public Cloud (formerly Redlock)** Image Scanning service provides free public API to scan your container images. When you “scan” an image, you are getting a list of all the vulnerabilities from all the packages and base OS installed in the image across all the layers. Each layer could contain multiple packages. The scan result will give you all the known vulnerabilities grouped by severity or package.

# Task 1 – Build and Scan the Application Container Image

In this activity we will start by building our app container image, then we will scan it for security vulnerabilities.

We will build the frontend service for our Guestbook app. The Development team wrote the code, and now we are packaging the code in a container.

This is the Dockerfile, which describes what we are including in this image (base OS/image, code and code dependencies/libraries):

```
1  # Copyright 2016 The Kubernetes Authors.
2  #
3  # Licensed under the Apache License, Version 2.0 (the "License");
4  # you may not use this file except in compliance with the License.
5  # You may obtain a copy of the License at
6  #
7  #     http://www.apache.org/licenses/LICENSE-2.0
8  #
9  # Unless required by applicable law or agreed to in writing, software
10 # distributed under the License is distributed on an "AS IS" BASIS,
11 # WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
12 # See the License for the specific language governing permissions and
13 # limitations under the License.
14
15 FROM php:5-apache
16
17 RUN apt-get update
18 RUN pear channel-discover pear.nrk.io
19 RUN pear install nrk/Predis
20
21 # If the container's stdio is connected to systemd-journald,
22 # /proc/self/fd/{1,2} are Unix sockets and apache will not be able to open()
23 # them. Use "cat" to write directly to the already opened fds without opening
24 # them again.
25 RUN sed -i 's#ErrorLog /proc/self/fd/2#ErrorLog "|$/bin/cat 1>\&2"#' /etc/apache2/apache2.conf
26 RUN sed -i 's#CustomLog /proc/self/fd/1 combined#CustomLog "|/bin/cat" combined#' /etc/apache2/apache2.conf
27
28 # Add the application code to the image
29 ADD guestbook.php /var/www/html/guestbook.php
30 ADD controllers.js /var/www/html/controllers.js
31 ADD index.html /var/www/html/index.html
```

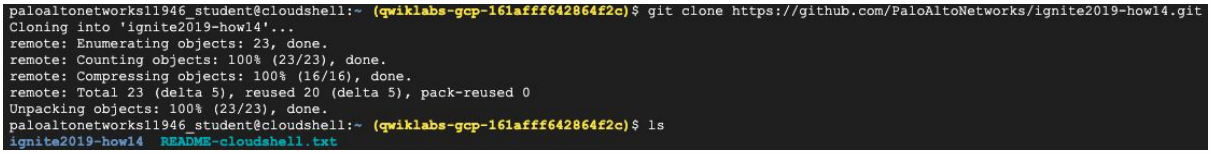
In this Dockerfile…

- We are using PHP:5-apache ***base image*** for our frontend app container (line 15) https://hub.docker.com/_/php
- Installing and updating ***dependencies*** (line 17-19)
- Adding the frontend ***app code***, PHP, JavaScript, and HTML to the image (line 29-31)

Now, let's build and scan this image.

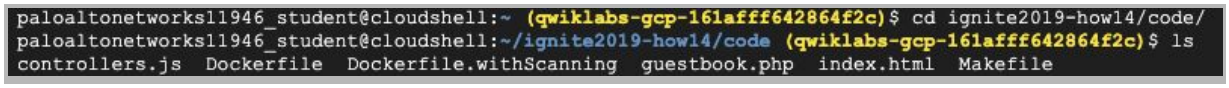
➔ Download the lab repo in Cloud Shell by running below cmd:

```
git clone https://github.com/PaloAltoNetworks/ignite2019-how14.git
```




➔ Access the Dockerfile

```
cd ignite2019-how14/code
```




➔ Edit the Dockerfile to add Prisma Public Cloud image scanning API call

➔ Open the Dockerfile in your favorite editor

```
nano Dockerfile
```

➔ Go to the end of the file and append the following two lines at the end

```
ARG rl_args
```

```
RUN SCAN_CMD=$(eval "curl https://vscanapidoc.redlock.io/scan.sh
2>/dev/null") && echo "$SCAN_CMD" | sh
```

➔ Save and exit

Press `Ctrl + o` to save and then `Ctrl + x` to exit

After making the changes Dockerfile should look like this:

```
paloaltonetworks11946_student@cloudshell:~/ignite2019-how14/code (qwiklabs-gcp-161afff642864f2c)$ tail Dockerfile
# them. Use "cat" to write directly to the already opened fds without opening
# them again.
RUN sed -i 's#ErrorLog /proc/self/fd/2#ErrorLog "|$/bin/cat 1>\&2"#' /etc/apache2/apache2.conf
RUN sed -i 's#CustomLog /proc/self/fd/1 combined#CustomLog "|/bin/cat" combined#' /etc/apache2/apache2.conf

ADD guestbook.php /var/www/html/guestbook.php
ADD controllers.js /var/www/html/controllers.js
ADD index.html /var/www/html/index.html
ARG r1_args
RUN SCAN_CMD=$(eval "curl https://vscanapidoc.redlock.io/scan.sh 2>/dev/null") && echo "$SCAN_CMD" | sh
```

What we're doing here by adding the r1_args and SCAN_CMD is, we are listing all the packages installed in this image and getting the list of all the vulnerabilities associated with those packages from the **Prisma Public Cloud free public image scanning API**. The **Prisma Public Cloud** Infrastructure as Code Scanner will provide a pass/fail for the build based on the list of vulnerabilities we get back.

**ARG rl_args** is for passing the build arguments to configure when to pass/fail the build and how to group/see the scan result. See https://vscanapidoc.redlock.io/ for more information.

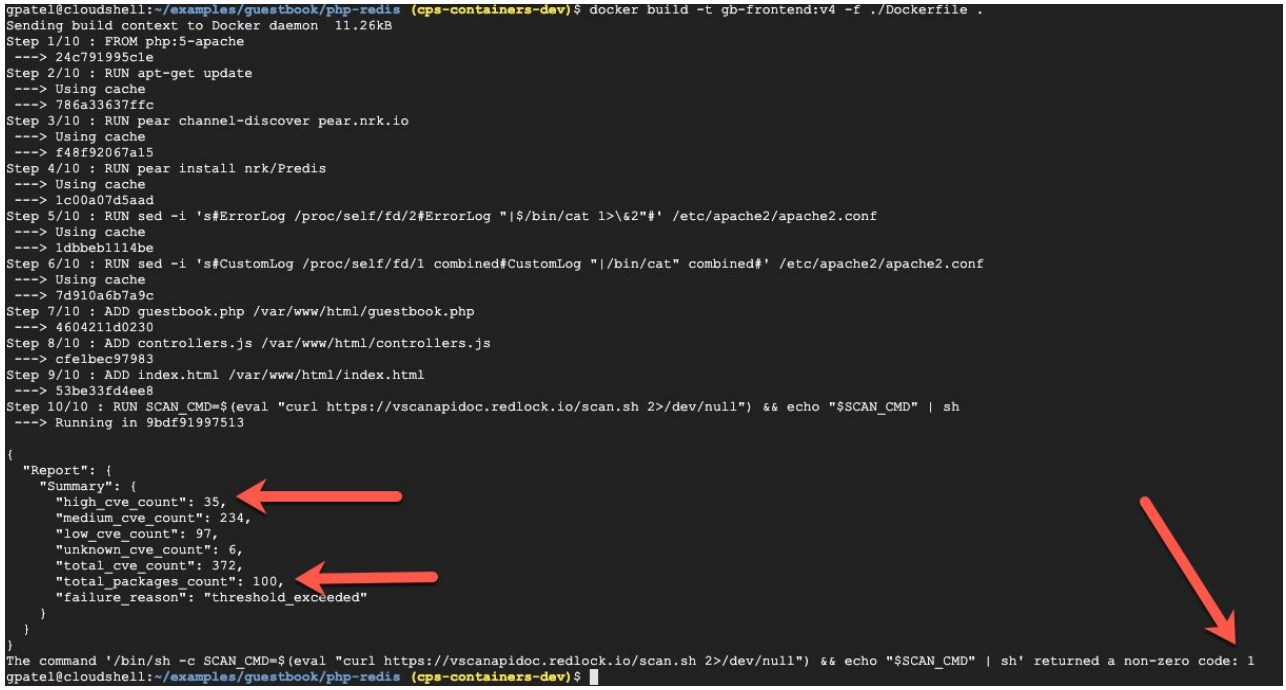
**Note:** *For your convenience, we have placed the final Dockerfile as* `Dockerfile.withScanning` in the `ignite2019-how14/code` folder.

➔ *You can copy that one using the following command:*

```
cp Dockerfile.withScanning Dockerfile
```

➔ Build the Docker image using the following command. This will make the actual API call during the build and display the scan result.

```
docker build -t gb-frontend:v4 . -f ./Dockerfile
```




➔ Next, analyze the completed results and take note of the following:

a. Notice the docker build failing with a non-zero exit code

b. It fails because the vulnerability scan result received from the Prisma Public Cloud image scan API endpoint indicate more than one packages have known vulnerabilities

c. Notice that the final image would have had 38 high severity CVEs, 248 medium and 102 low severity CVEs, totaling 394 CVEs.

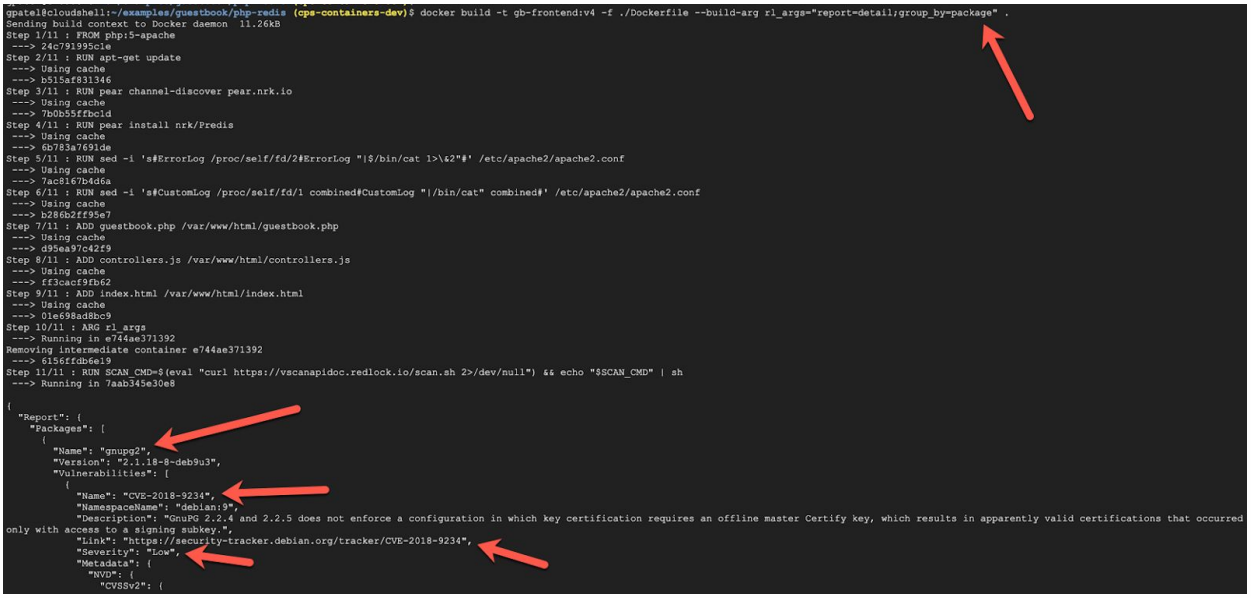
   **Note:** Your results may be different as new CVEs are being identified.

d. The number of packages analyzed are 100

e. Failure reason is the number of CVEs exceeded the threshold (by default 1)

➔ Next, get the list of CVEs grouped by the packages by passing the ***--build-arg rl_args="report=detail;group_by=package"*** argument to the docker build command

```
docker build -t gb-frontend:v4 . -f ./Dockerfile --build-arg rl_args="report=detail;group_by=package"
```




**Note**: The output might look different

**End of Activity 1**

# Activity 2 – Kubernetes App Manifest Scanning for Security Misconfigurations

***In this activity, you will:***

- ***Scan your kubernetes application deployment manifest using Prisma Public Cloud Infrastructure-as-Code (IaC) public API for security best practices***
- ***Analyze the result***
- ***Fix all the applicable misconfigurations***

---

In this activity, we will start using Kubernetes specific terms such as Pods, Services, etc. Here is a good primer: https://kubernetes.io/docs/concepts/workloads/pods/pod-overview/

## What is Kubernetes?

Kubernetes is an open-source container-orchestration system for automating application deployment, scaling, and management.

## What is a Kubernetes Manifest?

Kubernetes manifest file describes how your containerized application is deployed in kubernetes. There can be one or more objects in a manifest file such as Deployment (replicated group of Pods), Services (proxy), Volumes, and ConfigMaps (configuration for the application pods/containers). Manifest files can be in JSON or YAML format. YAML format is more common in kubernetes world, so we will use that in this lab, but Prisma Public Cloud Infrastructure-as-Code (IaC) API supports both JSON and YAML format.

### What does “scanning” the manifest file mean?

When you scan your kubernetes manifest files using the free Prisma Public Cloud Infrastructure-as-Code (IaC) API, you get back the analysis result that points of any configuration which is vulnerable to exploitation. The scan result will have severity associated with each of the rule violations.

You can include this scan into your CI/CD (Continuous Integration/ Continuous Delivery) pipeline, so all your kubernetes manifests go through an automated sanity check before they are applied to production. CI Build should fail if any of your manifests have a high severity security misconfiguration.

This API also allows you to scan Terraform and CFT files for security best practices violations. Detailed documentation can be found here: https://iacscanapidoc.redlock.io/

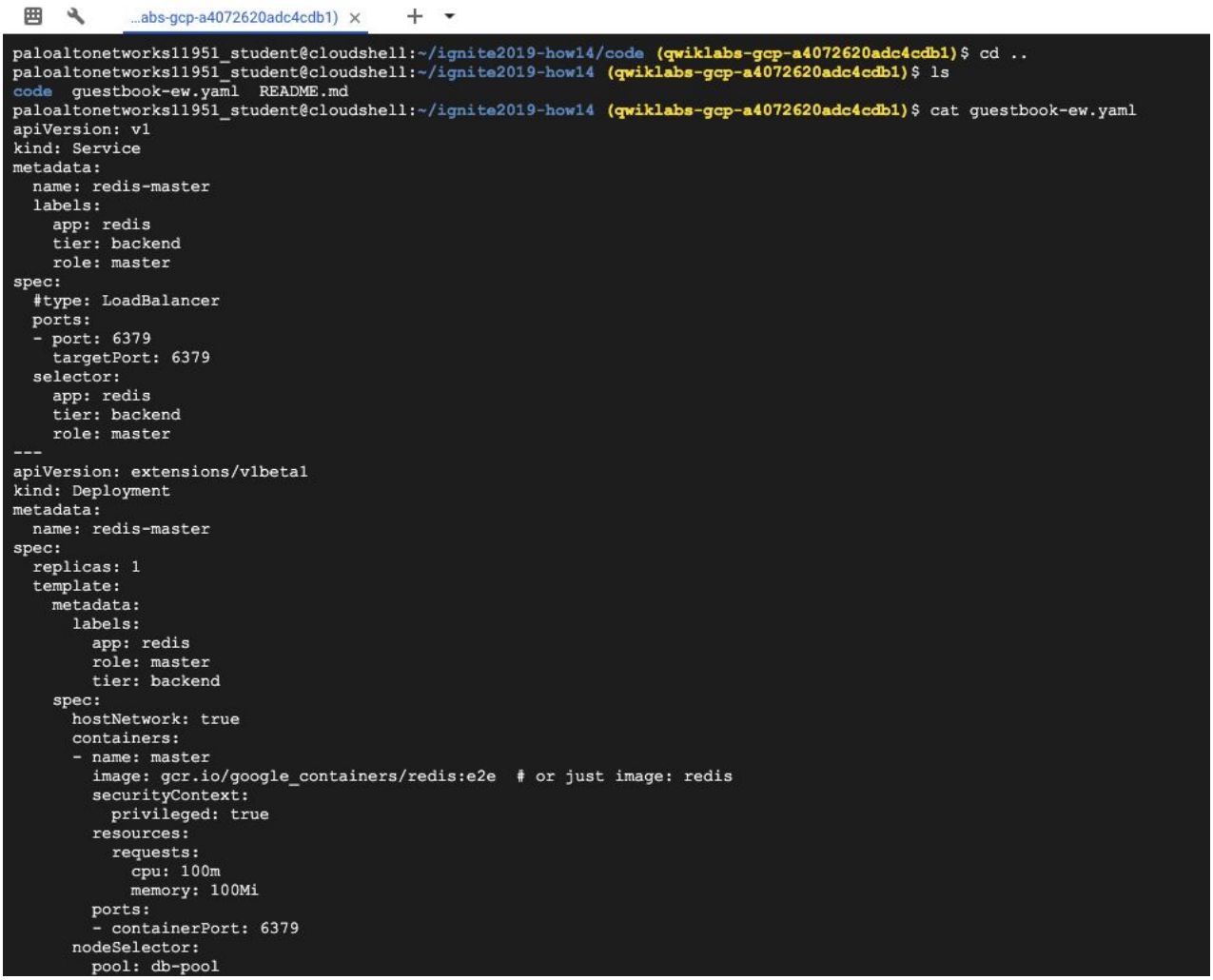
## Task 1 – Scan the Application Manifest for Security Best Practices

➔ Back in the Cloud Shell, explore the manifest files. First, go back to the repo base folder *ignite2019-how14/* by executing the following command:

```
cd ..
```

➔ Next view the guestbook application manifest by executing the following command:

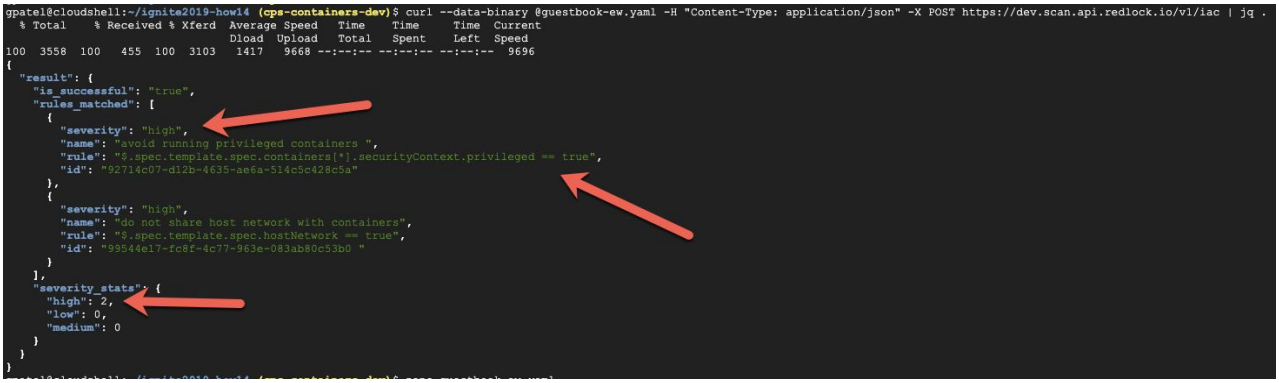
```
cat guestbook-ew.yaml
```




➔ Scan the guestbook app manifest with Prisma Public Cloud IaC (Infrastructure-as-Code) Scan API

```
curl --data-binary @guestbook-ew.yaml -H "Content-Type: application/json"
-X POST https://scanapi.redlock.io/v1/iac | jq .
```

➔ Analyze the results after the previous curl call:




As you can see from the scan result, we have 2 potential security misconfigurations in our manifest:

a. A container is running in privileged mode which can be dangerous
b. Pods in a deployment are sharing network namespace with the host

Both of these are classified as high severity security best practice violations, as you can see from the severity field for both of the violations.

## Task 2 – Update the Manifest to Fix the Policy Violations

If these configuration lines are not absolutely necessary then we should remove them. You should work with your developer and security team to discuss other options to avoid these offending configurations which can be potentially exploited. In our case, we will assume we have consulted with our dev and security team and decided to remove both offending violations.

➔ Open the manifest in your favorite text editor :)

```
nano guestbook-ew.yaml
```

➔ Remove the following lines (line 32)

```
hostNetwork: true
```

and (line 36-37)

```
securityContext:
    privileged: true
```

Use your choice of editor (vi/nano) to modify the `guestbook-ew.yaml` file. To delete a

line in the nano editor, you can move your cursor to the line you want to delete and then press `Ctrl + k` to delete that line

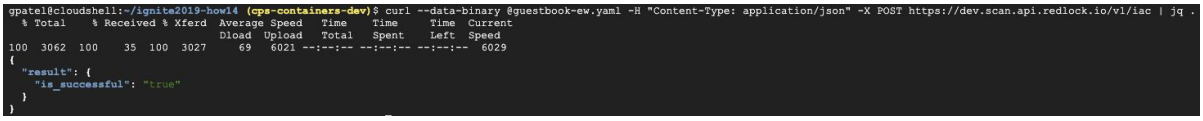
➔ Save and exit

Press `Ctrl + o` to save and then `Ctrl + x` to exit

➔ Rescan the guestbook app manifest again to make sure the policy violations are cleared by executing the following command again:

```
curl --data-binary @guestbook-ew.yaml -H "Content-Type: application/json"
-X POST https://scanapi.redlock.io/v1/iac | jq .
```

➔ Validate that the policy violations are gone!




You can also scan your Terraform and CFT template files with the same Prisma Public Cloud IaC public API endpoint, and they're all provided for free.

For more details on Kubernetes manifest scanning and IaC API documentation, check out the documentation page: https://scanapidoc.redlock.io/

**End of Activity 2**

# Conclusion

Congratulations! You have now successfully tested the Prisma Public Cloud Free Scanning APIs for container image scanning and Kubernetes app manifest scanning. For the full documentation, check out the documentation at https://scanapidoc.redlock.io/