PRISMA CLOUD

techNOTE

# Code-To-Cloud Prisma Cloud Demo Attack

| | |
|---|---|
| Date: | 01/02/2025 |
| Revision: | 2.1 |
| Author: | David Avila, Prisma Cloud SA LATAM |

## Overview

This guide describes how to deploy a code-to-cloud Prisma Cloud Demo Attack.

## Why to Use

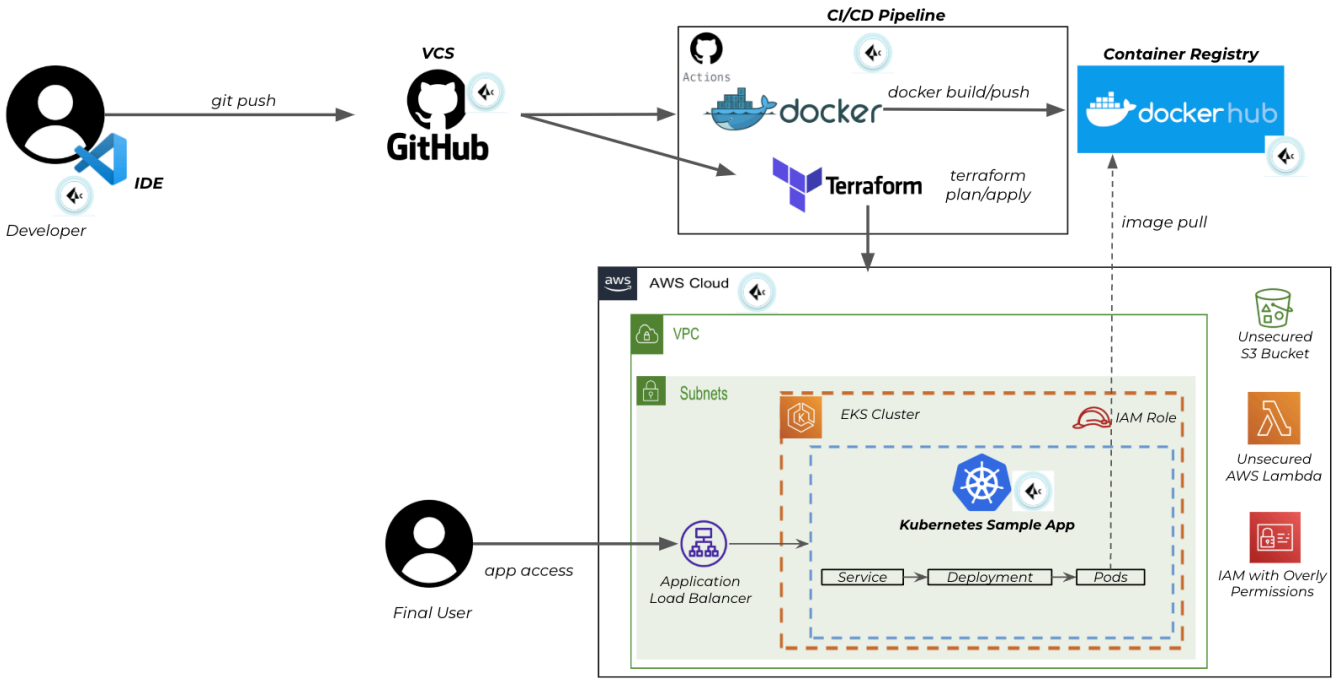
You can easily show all the potential that Prisma Cloud has. Easy to deploy, easy to show.

## When to Use

When you want to show a Prisma Cloud full demo.

# Getting Started

This Prisma Cloud demo simply deploys an AWS EKS Cluster with a Spring (vulnerable) app using Github Actions as CI/CD Tool to deploy the infrastructure with Terraform on AWS and then it builds the Docker images to deploy the app on the Kubernetes Cluster. Additionally, it deploys some additional components on AWS, like AWS S3 bucket and AWS Lambda to show the misconfigurations.




https://docs.google.com/presentation/d/1k7_4cWSGbpxtnE_47D4brDrxSS5dAxi5r0s5hWMeLzU/edit

You'll use Docker to build/push your own image, to push it to your Container Registry. Otherwise, you can use the pre-built image.

As the image above describes, you can start scanning your code from your IDE (Visual Code for example), deploy the code using pre-built Github Actions to deploy the infra and the app, and then, we can protect the runtime using Prisma Cloud.

# Before You Begin

You will use these services or tools to be able to complete the demo environment.

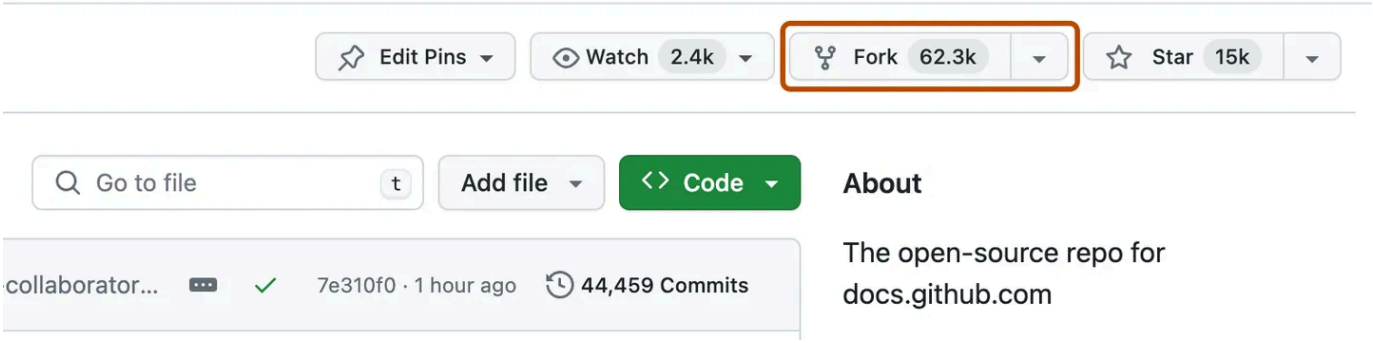
- **Prisma Cloud Enterprise tenant access and Access Key/Secret**
  - https://docs.prismacloud.io/en/enterprise-edition/content-collections/administration/create-access-keys
- **An AWS Account with Access Key/Secret**
  - Keep in mind that you'll deploy a EKS Cluster; it costs about $ 0.15 per hour + EKS nodes, in this template we're using t3.small size; that's about 2 cents per hour per node (*this is an estimate*)
- **AWS CLI configured with your credentials**
  - Installation: https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html
  - Configuration: https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-configure.html
- **Terraform CLI**
  - Installation: https://developer.hashicorp.com/terraform/tutorials/aws-get-started/install-cli
- **Kubernetes CLI (*kubectl*)**
  - Installation: https://kubernetes.io/docs/tasks/tools/
- **Python**
  - Installation: https://www.python.org/downloads/
- **A GitHub account and Github Desktop**
  - https://github.com
  - https://desktop.github.com/download/
- **IDE (Visual Code or IntelliJ)**
  - https://code.visualstudio.com/download
- **DockerHub account:**
  - https://hub.docker.com/

# Deployment Flow

## Procedure 1: Fork the Demo Attack Repository

First thing you need to do is to fork the demo attack files you'll need to deploy the demo attack environment.

**Step 1** Go to https://github.com/davidaavilar/aws-demo-attack-lab-v2 and then click on "**Fork**" and then "**Download ZIP**"



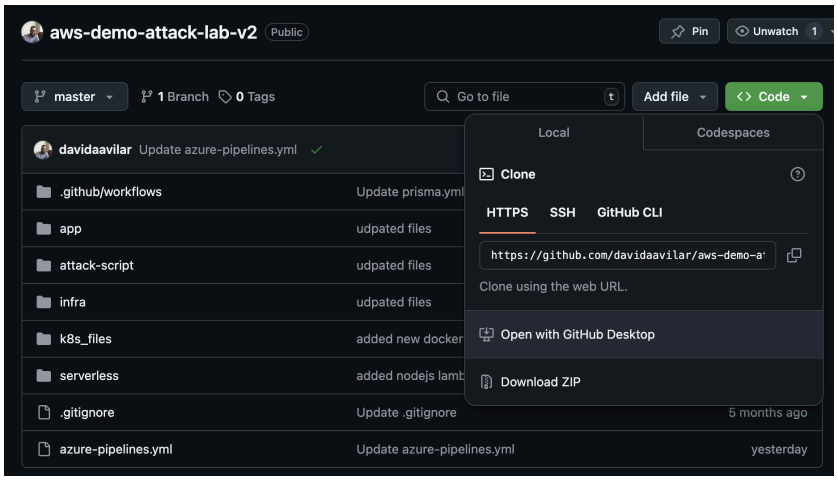

**Step 2** Name your new repo

This will create the repository into your Github Account

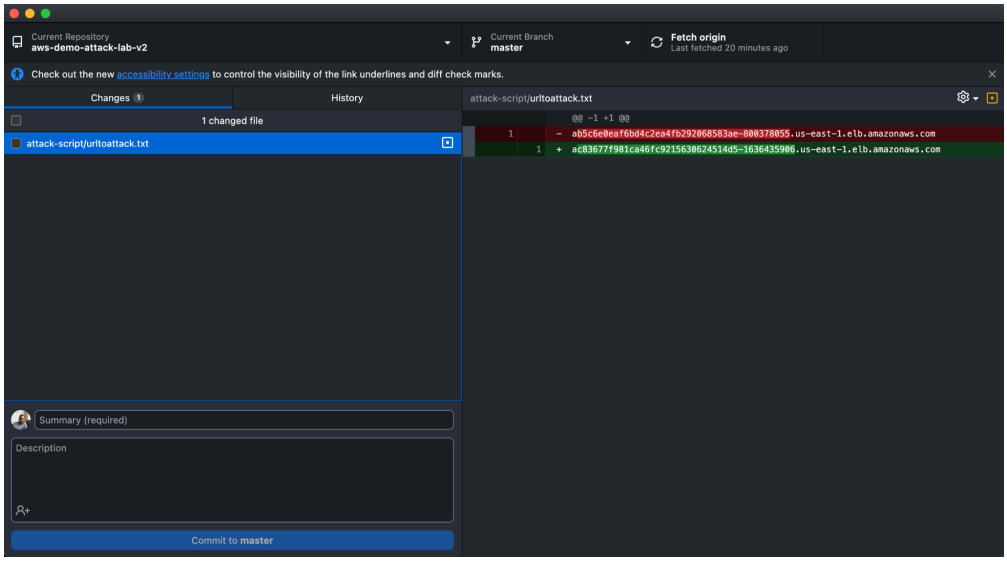
## Procedure 2: Clone your repo locally

**Step 1** Open your Github Desktop and login with your Github Account

**Step 2** Open your forked repo in Github.com and then click on "**Code**" and then, "**Open with Github Desktop**". This will clone your repo locally.




**Step 3** Go under *Documents/GitHub/<YOUR-REPO-NAME>* to confirm you have the files locally. You will see something like this under GitHub Desktop



## Procedure 3: Test your tools

NOTE: Skip this procedure if you already tested the tools and you're sure it's working fine.

**Step 1** In your terminal type "***terraform version***". You'll be able to see the terraform version. If not, check the installation procedure above ('Before You Begin' section)

```
davidaavilar 🚀 ~ % terraform version
Terraform v1.4.6
on darwin_arm64

Your version of Terraform is out of date! The latest version
is 1.5.5. You can update by downloading from https://www.terraform.io/downloads.html
```

**Step 2** In your terminal type "***aws sts get-caller-identity***". You'll be able to see your AWS user logged. If not, check the installation procedure above ('Before You Begin' section)

```
davidaavilar 🚀 ~ % aws sts get-caller-identity
{
    "UserId": "AIDARSWGFMIYNUBVKKZNH",
    "Account": "108863513136",
    "Arn": "arn:aws:iam::108863513136:user/davidaavilar"
}
```

**Step 3** In your terminal type "***kubectl version --short***". You'll be able to see the version of kubectl. If not, check the installation procedure above ('Before You Begin' section)

```
davidaavilar 🚀 ~ % kubectl version --short
Flag --short has been deprecated, and will be removed in the future. The --short output will become the default.
Client Version: v1.27.1-eks-2f008fe
Kustomize Version: v5.0.1
```

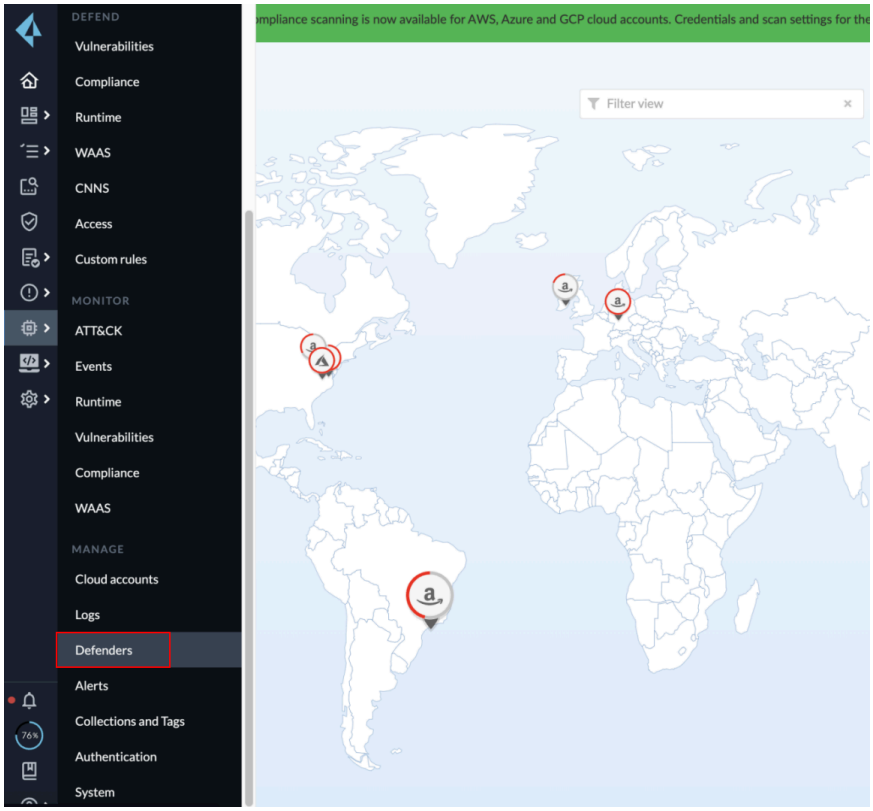
**NOTE**: If you are in any other stack, just replace **appX** for **apiX.**

## Procedure 3: Get your Prisma Cloud Defender's manifest for Kubernetes

To be able to install Prisma Cloud Defenders in the EKS Cluster you need to download the kubernetes manifest (YAML file) from the Prisma Cloud Console.

**Step 1** Go to your Prisma Cloud Enterprise console and then go to the **Compute** tab.

**Step 2** On the Compute tab settings , go down and click on **Manage > Defenders**

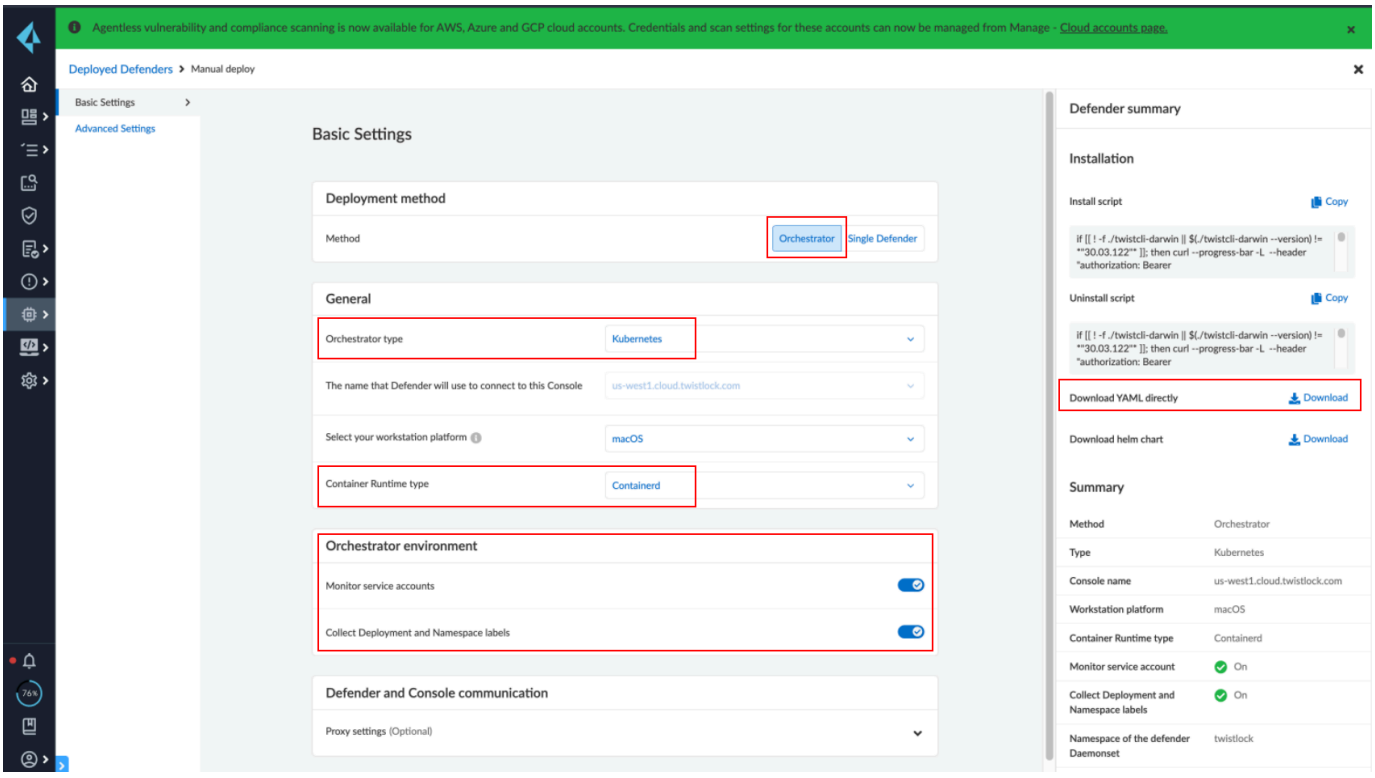


**Step 3** Click on "**Manual deploy**" and configure the settings:

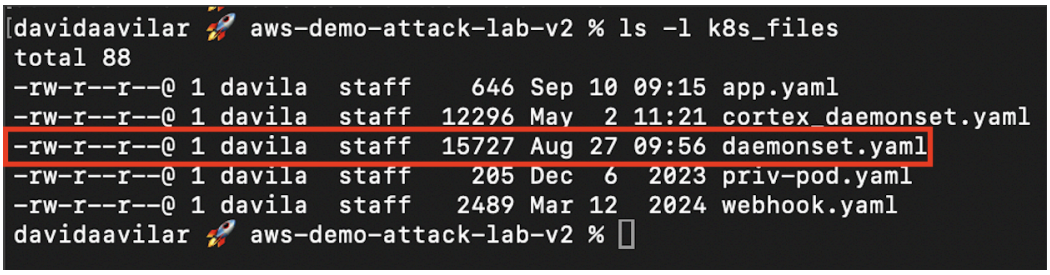
- **Method**: Orchestrator
- **Orchestrator type**: Kubernetes
- **Container Runtime type**: Containerd
- On Orchestrator environment select "*Monitor Service Accounts*" and "*Collect Deployment and Namespaces Labels*"

**Step 4** When ready, on the right panel download the YAML file (**Download** button) and save it (replace it) at the **k8s_files** folder of the deployment under your project's folder (the one you created in Procedure 1).

**NOTE**: DO NOT CHANGE THE FILENAME. Should be *daemonset.yaml*



**Step 4** Go to your Terminal and move to the repo's folder. Then execute "**ls -l k8s_files**" to see the content. You'll be able to the **daemonset.yaml**



## Procedure 4: Prepare your Deployment

Customize your deployment.

**Step 1** Open your folder using Visual Code (for better handling). If you don't have the IDE, you can use any Text Editor.

**Step 2** Go to "infra/*variables.tf*" file and customize your deployment:

- deployment_name = SELECT YOUR DEPLOYMENT NAME

## Procedure 5: Create S3 Bucket on AWS

This procedure will create a S3 bucket on AWS for Terraform state.

**Step 1** Go to your AWS Console and then S3 service

**Step 2** Make sure you are on North Virginia region (us-east-1)

**Step 3** Click on "Create Bucket"

**Step 4** Name it with a unique name

**Step 5** Next next next (make sure it's private and encrypted, it's by default)

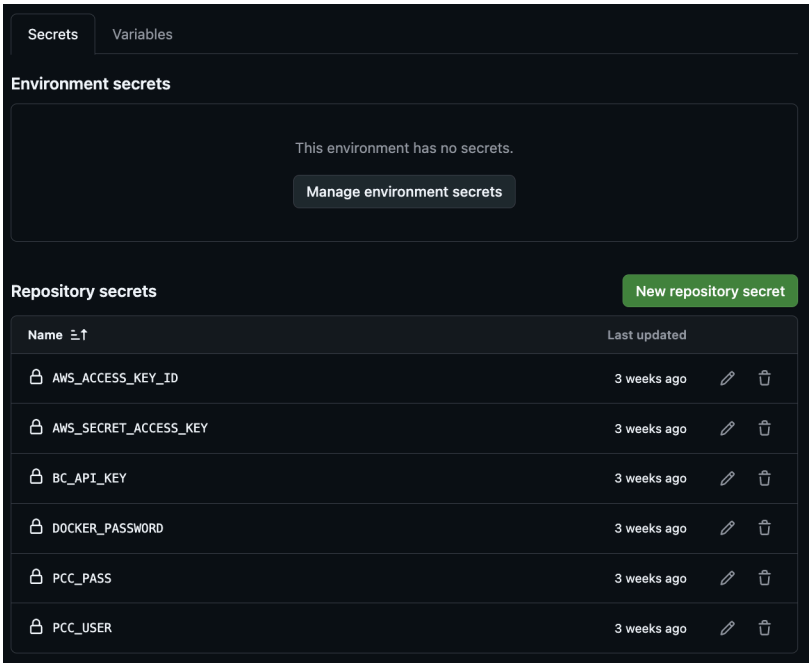
**Step 6** Save the name for later.

https://docs.aws.amazon.com/AmazonS3/latest/userguide/create-bucket-overview.html

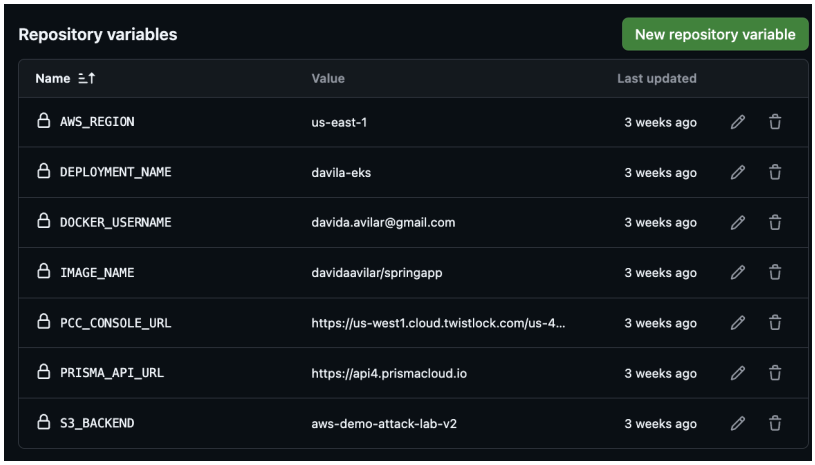
## Procedure 6: Create Variable Library

**Step 1** On your GitHub Repository, go to "Settings" and then click on "Security" > “Secrets and Variables” > Actions:

**Step 2** Secrets:




**Step 3** Variables:



**Step 4** Create your secrets and variables using the following table:

**SECRETS:**

| Name | Value | Fixed | Comments |
|---|---|---|---|
| BC_API_KEY | < ACCESS KEY >::< SECRET > | No | your Prisma Cloud creds |
| AWS_ACCESS_KEY_ID | < YOUR AWS ACCESS KEY > | No | |
| AWS_SECRET_ACCESS_KEY | < YOUR AWS SECRET KEY > | No | |
| DOCKER_PASSWORD | < YOUR DOCKER HUB PASSWORD > | No | |
| PCC_PASS | < SECRET > | No | your Prisma Cloud creds |
| PCC_USER | < ACCESS KEY > | No | your Prisma Cloud creds |

**VARIABLES:**

| Name | Value | Fixed | Comments |
|---|---|---|---|
| AWS_REGION | us-east-1 | Yes | |
| DEPLOYMENT_NAME | < YOUR DEPLOYMENT NAME > | No | |
| DOCKER_USERNAME | < YOUR DOCKER HUB USERNAME > | No | |
| IMAGE_NAME | <dockerhub_registry>/springapp | No | |
| PCC_CONSOLE_URL | < YOUR PRISMA COMPUTE URL > | No | |
| PRISMA_API_URL | https://api.prismacloud.io | No | your PC stack api URL |
| S3_BACKEND | < YOUR S3 BUCKET NAME > | No | |

## Procedure 7: Deploy the Lab

Now you're ready to deploy your Lab. Let's create and run a Pipeline

**Step 2** Go to your folder locally and open the "*prisma.yml*" file under your folder ***Github > <YOUR REPO> > .github > workflows*** and in the first line (Name) put your "DEPLOYMENT NAME"

**Step 3** Save it and then Commit and Push on Github Desktop.

**Step 4** You will see your Github Actions will start running automatically. This will take some time (about 15 minutes)

## Procedure 8: Testing the Lab

**Step 1** If your terraform apply gave you a successful message you can execute this command to see if you are already connected to the EKS Cluster from your terminal:

**kubectl get node**

```
davidaavilar 🚀 davila-demo-attack-public-main % kubectl get node
NAME                           STATUS   ROLES    AGE   VERSION
ip-10-0-2-75.ec2.internal      Ready    <none>   10m   v1.26.6-eks-a5565ad
ip-10-0-3-195.ec2.internal     Ready    <none>   10m   v1.26.6-eks-a5565ad
```

**kubectl get pod -A**

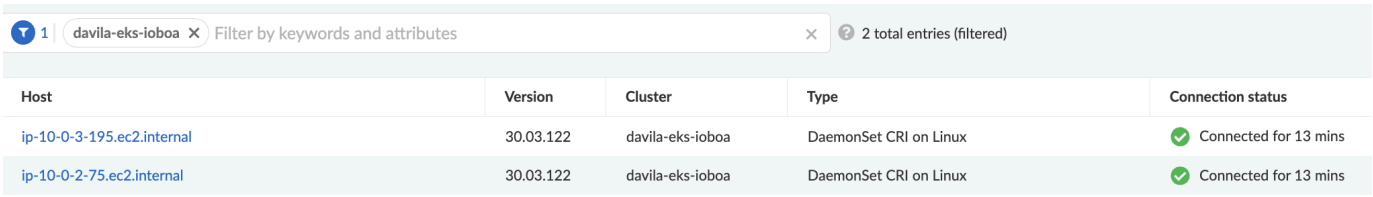
```
davidaavilar 🚀 davila-demo-attack-public-main % kubectl get pod -A
NAMESPACE     NAME                              READY   STATUS    RESTARTS   AGE
kube-system   aws-node-8jqrf                    1/1     Running   0          11m
kube-system   aws-node-n6ct8                    1/1     Running   0          10m
kube-system   coredns-55fb5d545d-jj287          1/1     Running   0          16m
kube-system   coredns-55fb5d545d-xwx2c          1/1     Running   0          16m
kube-system   kube-proxy-lqqxb                  1/1     Running   0          11m
kube-system   kube-proxy-pbzlw                  1/1     Running   0          10m
spring-app    spring-app-6f6f4b87bc-9mqph       1/1     Running   0          6m31s
twistlock     twistlock-defender-ds-cn622       1/1     Running   0          6m29s
twistlock     twistlock-defender-ds-lcll2       1/1     Running   0          6m29s
```

**kubectl get svc -A**

```
davidaavilar 🚀 davila-demo-attack-public-main % kubectl get svc -A
NAMESPACE     NAME         TYPE           CLUSTER-IP       EXTERNAL-IP                                                                    PORT(S)          AGE
default       kubernetes   ClusterIP      172.20.0.1       <none>                                                                         443/TCP          17m
kube-system   kube-dns     ClusterIP      172.20.0.10      <none>                                                                         53/UDP,53/TCP    16m
spring-app    spring-app   LoadBalancer   172.20.162.140   ae035635777b949f7a5ceeeb1c73baff-1514304416.us-east-1.elb.amazonaws.com   80:31512/TCP     7m11s
twistlock     defender     ClusterIP      172.20.24.77     <none>                                                                         443/TCP          7m9s
```

You'll be able to see the nodes, the pods (spring-app and twistlock pods) and the service exposed (spring-app). Additionally, you'll see a new file under your deployment directory **/attack-script** called **urltoattack.txt.** This is the external domain (IP) of your application (output from '*kubectl get svc -A*' command)

Also, you can go to the Prisma Cloud Console and verify that your Defenders are "Connected" under **Compute** tab **> Manage > Defenders > Deployed Defenders** and filter by your Cluster Name (you can find your EKS cluster name from your Terraform Outputs)




Finally, you can go and check your app using your Browser if you used the default image:

**http://a00fec344c0b3443aa1fa191e50baf11-1903546003.us-east-1.elb.amazonaws.com/helloworld/greeting**

Following the example above:

http://ae035635777b949f7a5ceeeb1c73baff-1514304416.us-east-1.elb.amazonaws.com/helloworld/greeting


**Now you can exploit it!**

## Procedure 11: How to attack your app

**Step 1** On your terminal, under your deployment folder, move to **/attack-script** using *'cd'* command.

```
[davidaavilar 🚀 davila-demo-attack-public-main % cd attack-script
[davidaavilar 🚀 attack-script % ls -l
total 40
-rw-r--r--@ 1 davila  staff    72 Aug 14 14:37 urltoattack.txt
-rw-rw-r--@ 1 davila  staff  5046 Aug 14 11:04 waas-attack-owasp.py
-rw-rw-r--@ 1 davila  staff  4648 Aug 14 11:04 waas-attack-vp.py
```

**Step 2** You should be see three files

- *urltoattack.txt;* it's the one that contains your ALB external domain (populated automatically by Terraform)
- *waas-attack-owasp.py;* Python script to attack using OWASP Top 10 attacks.
- *waas-attack-vp.py;* Python script to exploit Spring vulnerability (to use to show Virtual Patching capability in Prisma Cloud WAAS.

**Step 3** To attack the app you simply execute the Python script you want to use

***python3 waas-attack-owasp.py***

**python3 waas-attack-vp.py**

**NOTE:** You must disable GlobalProtect or any VPN client

If you haven't configured Prisma Cloud, you should receive successful attacks:

```
[davidaavilar 🚀 attack-script % python3 waas-attack-owasp.py
[+]-----------------------------------------------------------------------------------------------------------------------
[+] The URL to attack is: http://ae035635777b949f7a5ceeeb1c73baff-1514304416.us-east-1.elb.amazonaws.com/helloworld/greeting
[+]-----------------------------------------------------------------------------------------------------------------------
[+] Trying a SQL Injection
[+] SQL Injection successful! - <Response [200]>
[+]-----------------------------------------------------------------------------------------------------------------------
[+] Trying a Command Injection
[+] Command Injection successful! - <Response [200]>
[+]-----------------------------------------------------------------------------------------------------------------------
[+] Trying a File Inclusion
[+] Local File Inclusion successful! - <Response [200]>
[+]-----------------------------------------------------------------------------------------------------------------------
[+] Trying a Cross-Site Scripting (XSS) - <script>alert(1)</script>
[+] Cross-Site Scripting (XSS) successful! - <Response [200]>
[+]-----------------------------------------------------------------------------------------------------------------------
[+] Trying a Malformed HTTP Request
[+] Malformed HTTP Request successful! - <Response [200]>
[+]-----------------------------------------------------------------------------------------------------------------------
```

```
[davidaavilar 🚀 attack-script % python3 waas-attack-vp.py
[+]-----------------------------------------------------------------------------------------------------------------------
[+] The URL to attack is: http://ae035635777b949f7a5ceeeb1c73baff-1514304416.us-east-1.elb.amazonaws.com/helloworld/greeting
[+]-----------------------------------------------------------------------------------------------------------------------
[*] Resetting Log Variables.
[*] Response code: 200
[*] Modifying Log Configurations
[*] Response code: 200
[*] Send the packet that writes the web shell
[*] Response code: 200
[*] Resetting Log Variables.
[*] Response code: 200
[+] Exploit completed
[+] Check your target for a shell
[+] File: shell.jsp
[+] Shell should be at: http://ae035635777b949f7a5ceeeb1c73baff-1514304416.us-east-1.elb.amazonaws.com/shell.jsp?cmd=id
```

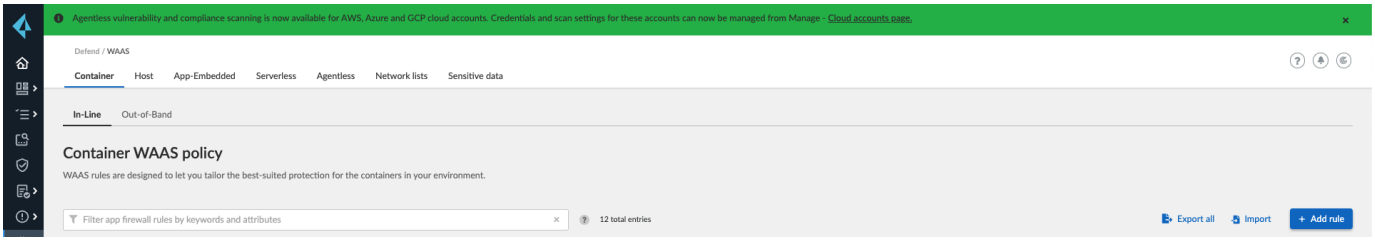
This last one (waas-attack-vp.py) exploit your Spring app, and you can verify the exploitation at the URL at the end, and you should receive a response from the container of the command (cmd) "**id**". This is a successful attack!


**NOTE:** You can delete the app pod to recreate the app. It will create a new one automatically.

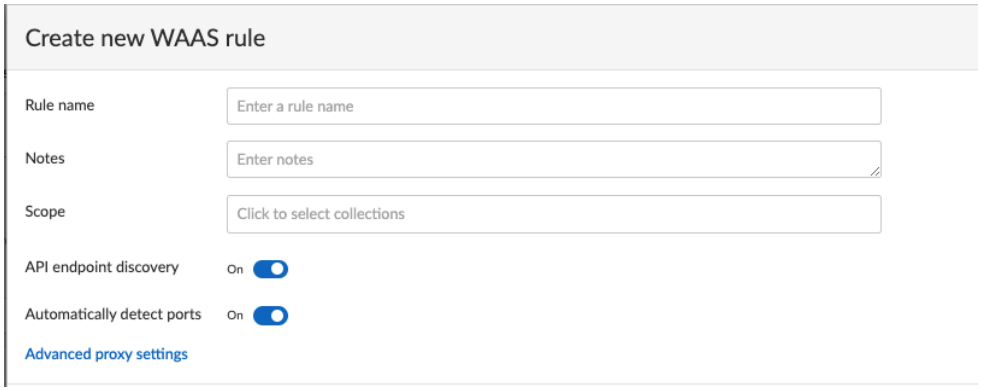
## Procedure 12: How to protect your app with Prisma Cloud

Because the previous attack is OWASP Top-10 based, we need to configure WAAS in Prisma Cloud in order to protect the application from this type of attack.

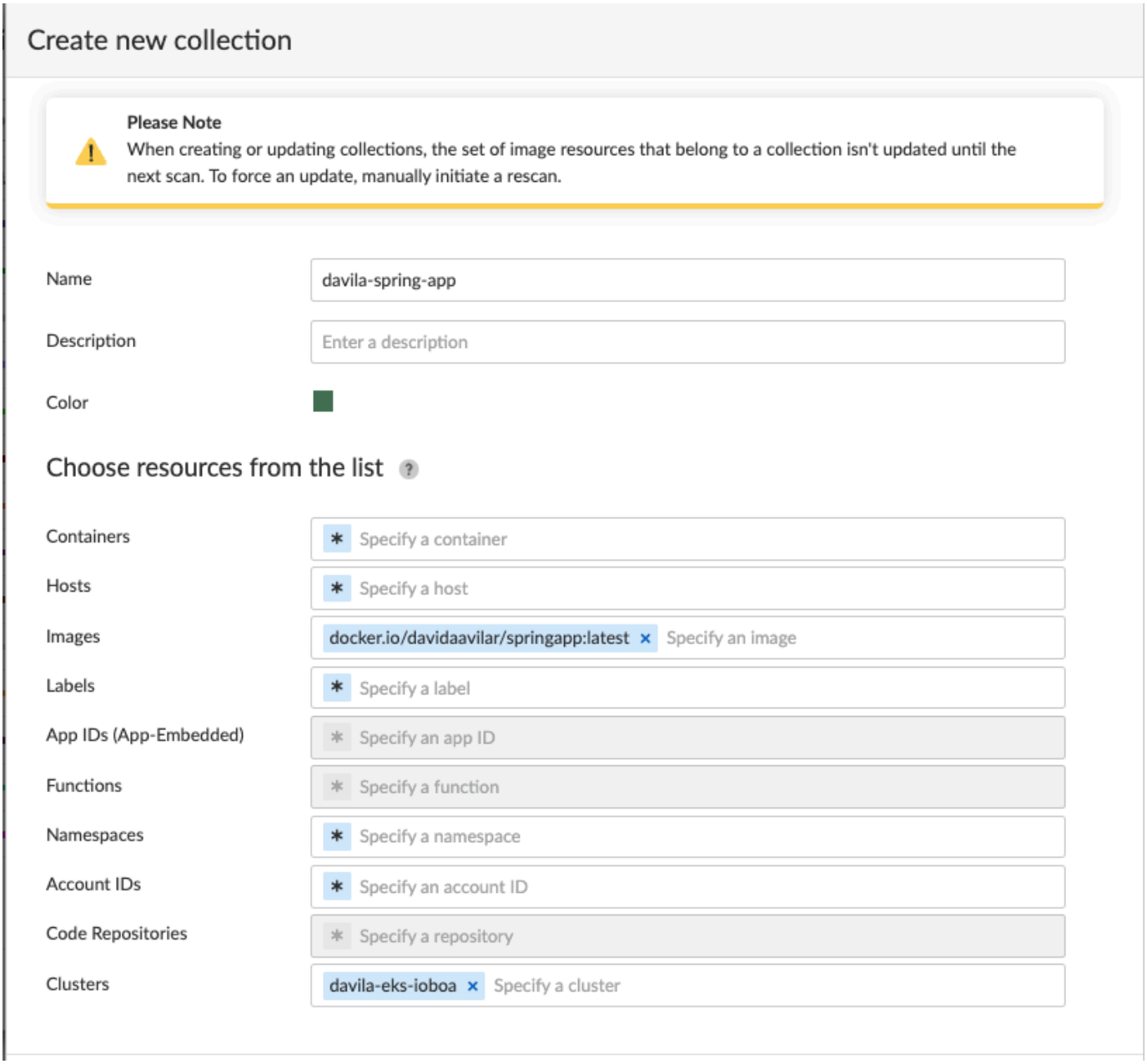
**Step 1** Go to Prisma Cloud Console and go under **Compute** tab **> Defend > WAAS > Container > In-Line > + Add Rule**



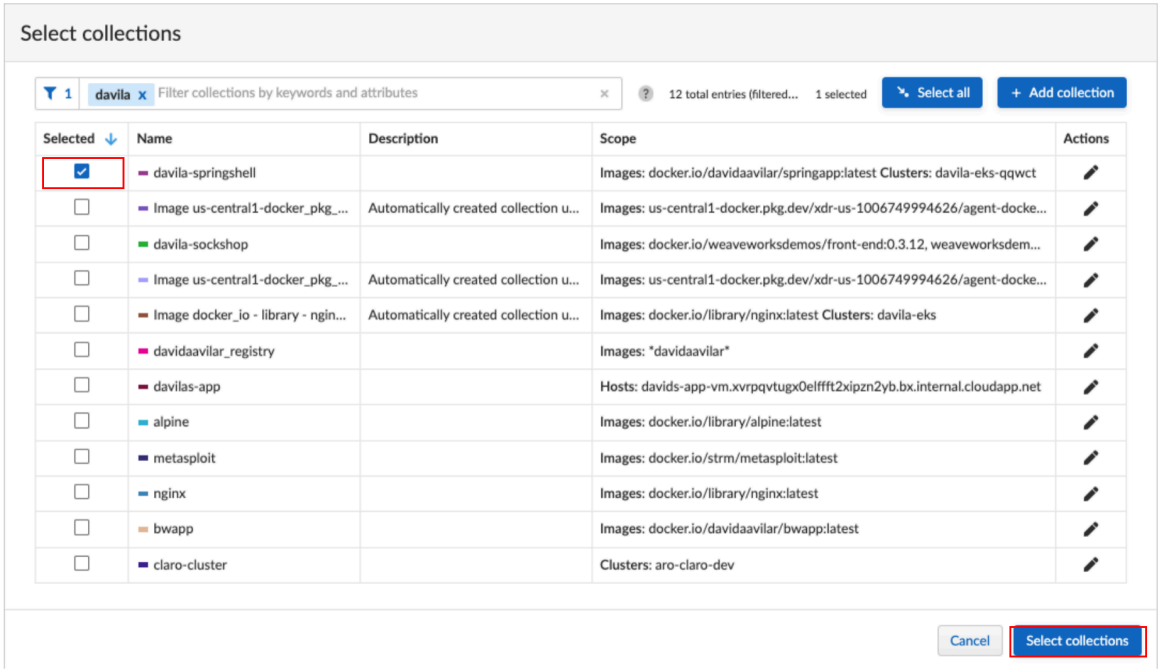

**Step 2** Name your rule and then you need to specify a "**Scope**"; click under "Click to select collections" and add a New Collection.



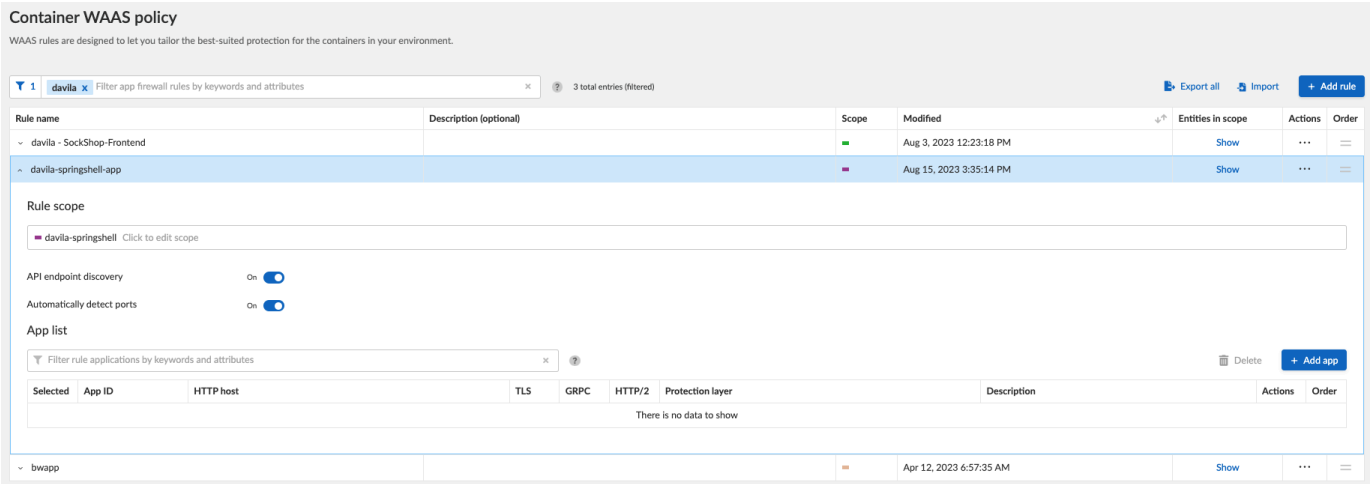
**Step 3** Name your collection with the name of your application (Spring-app for example) and select the **Image** of your application and your **Cluster** (see below) and save it.



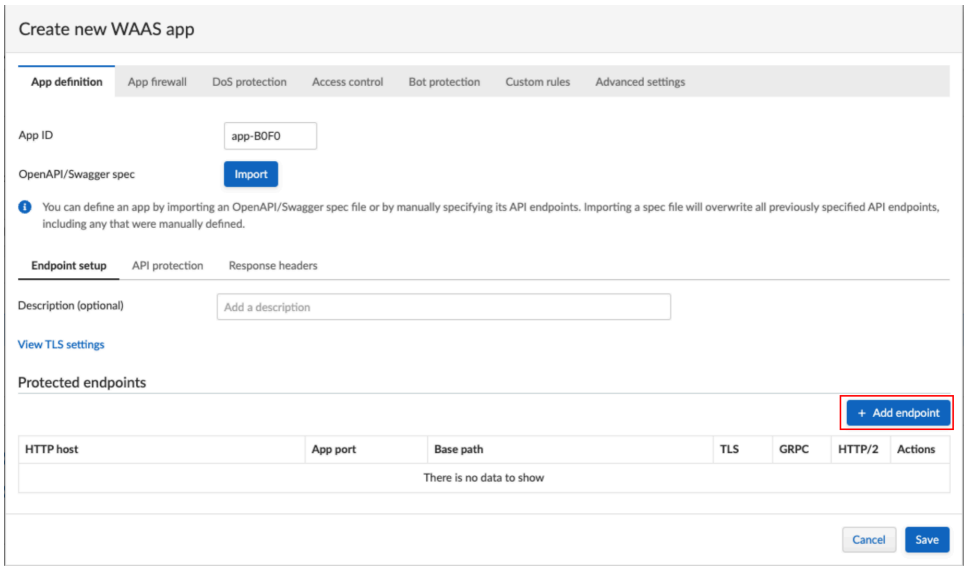

**Step 4** Select your collection you just created and then, click "**Select collections**" at the end, and save it.



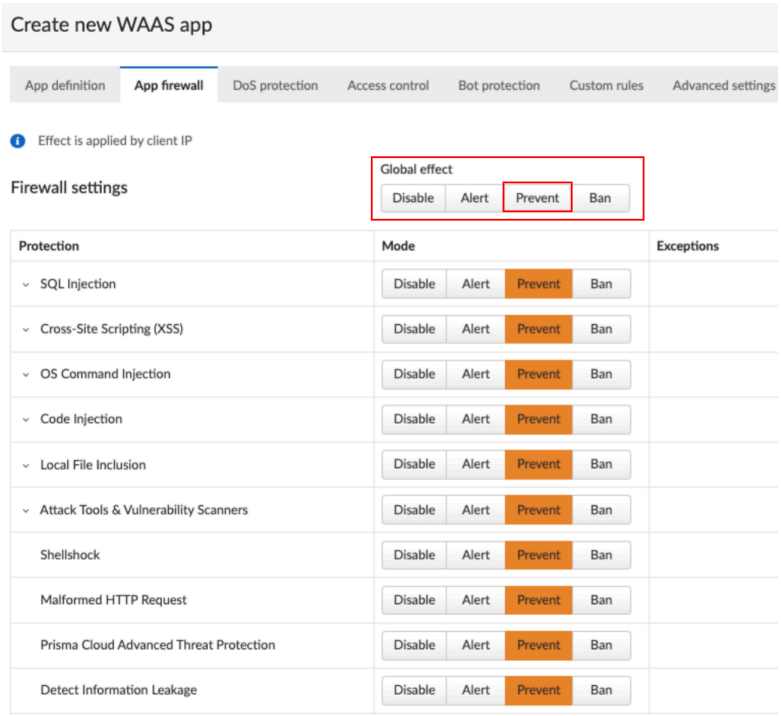

**Step 5** Under your WAAS rules, maximize your rule and then click "**+ Add app**" (just be sure you are into the rule you just created).




**Step 6** Under "**App definition**" tab, click on "**+ Add endpoint**" and leave everything as it is and click "**Create**"

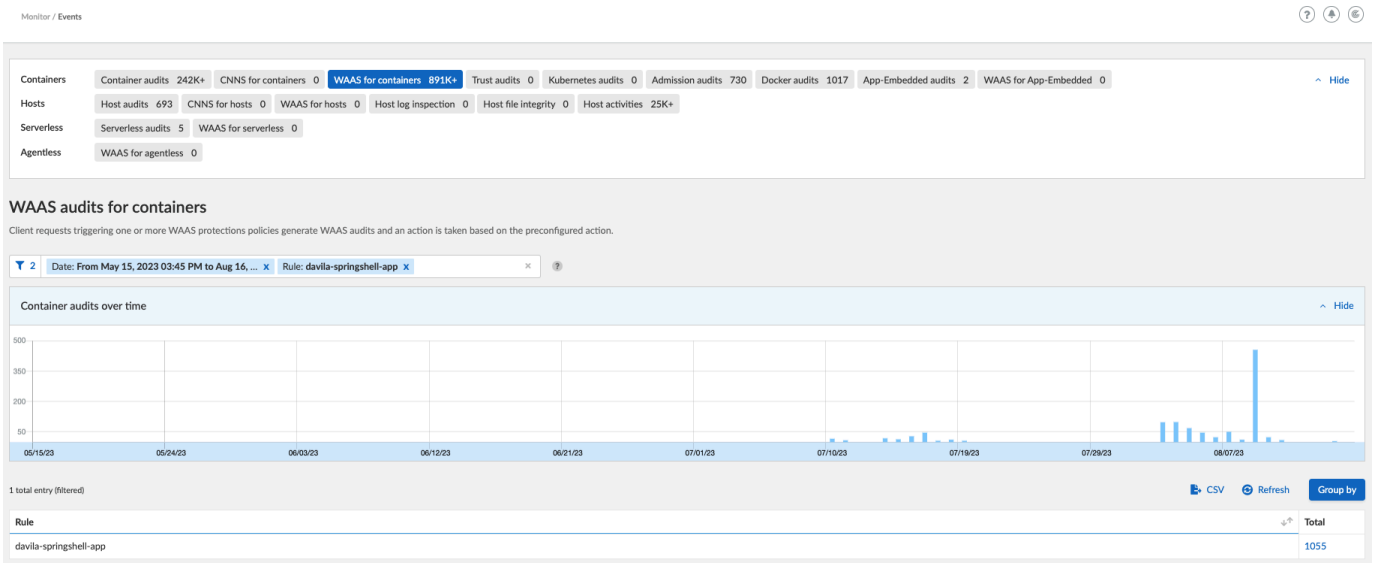


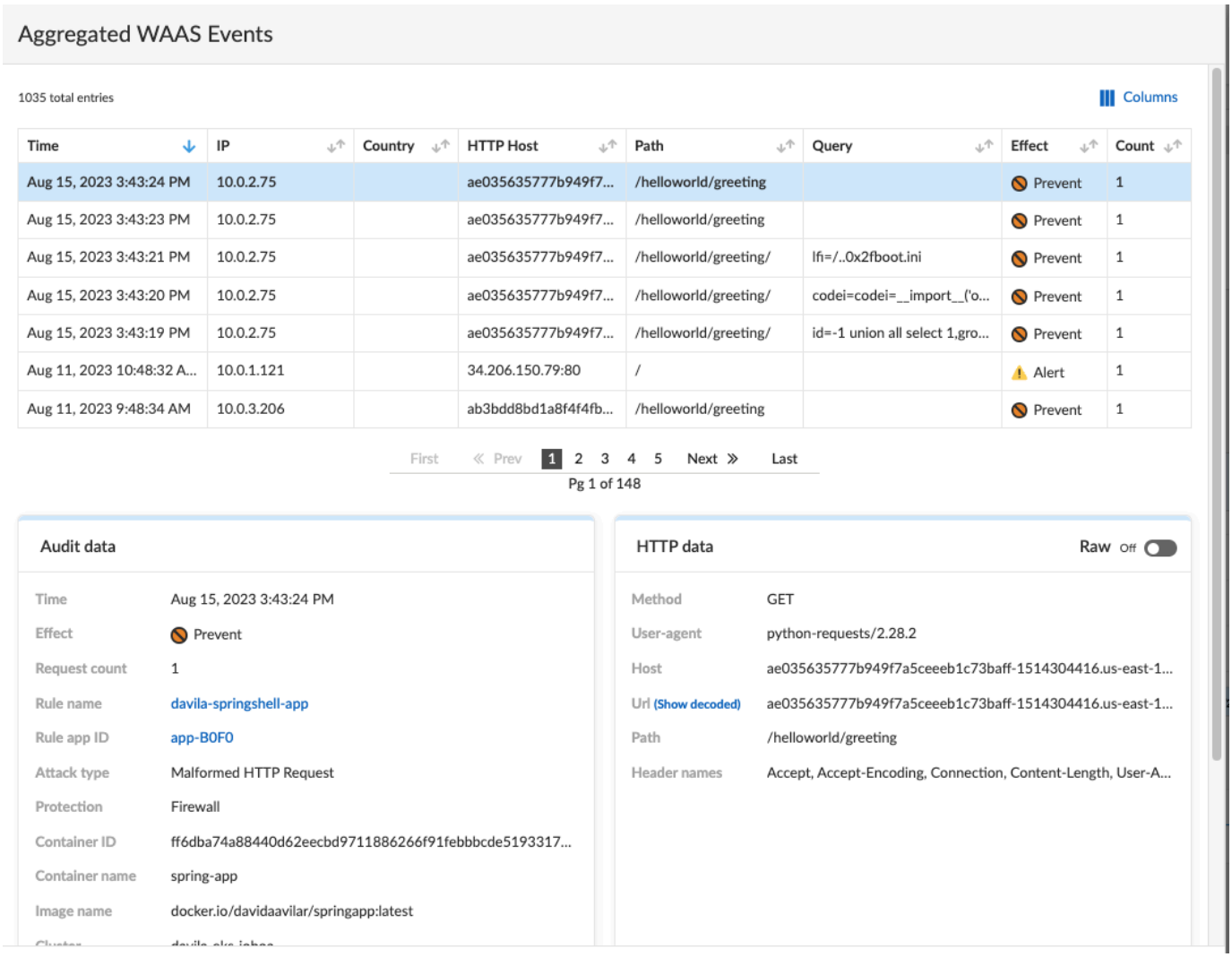

**Step 7** Under the "**App firewall**" tab > **Global Effect** click on "**Prevent**" and save it.




**Step 8** Try to attack again using the OWASP script and you should see all the attacks have been blocked by Prisma Cloud with 403 response error.

If you go to Prisma Cloud Console and go under **Compute** tab **> Monitor > Events > Containers > WAAS for Containers** and filter by your rule, you should see your attempts:





## Aggregated WAAS Events

1035 total entries

Columns

| Time | IP | Country | HTTP Host | Path | Query | Effect | Count |
|---|---|---|---|---|---|---|---|
| Aug 15, 2023 3:43:24 PM | 10.0.2.75 | | ae035635777b949f7... | /helloworld/greeting | | Prevent | 1 |
| Aug 15, 2023 3:43:23 PM | 10.0.2.75 | | ae035635777b949f7... | /helloworld/greeting | | Prevent | 1 |
| Aug 15, 2023 3:43:21 PM | 10.0.2.75 | | ae035635777b949f7... | /helloworld/greeting/ | lfi=/..0x2fboot.ini | Prevent | 1 |
| Aug 15, 2023 3:43:20 PM | 10.0.2.75 | | ae035635777b949f7... | /helloworld/greeting/ | codei=codei=__import__('o... | Prevent | 1 |
| Aug 15, 2023 3:43:19 PM | 10.0.2.75 | | ae035635777b949f7... | /helloworld/greeting/ | id=-1 union all select 1,gro... | Prevent | 1 |
| Aug 11, 2023 10:48:32 A... | 10.0.1.121 | | 34.206.150.79:80 | / | | Alert | 1 |
| Aug 11, 2023 9:48:34 AM | 10.0.3.206 | | ab3bdd8bd1a8f4f4fb... | /helloworld/greeting | | Prevent | 1 |

First « Prev 1 2 3 4 5 Next » Last

Pg 1 of 148

### Audit data

| | |
|---|---|
| Time | Aug 15, 2023 3:43:24 PM |
| Effect | Prevent |
| Request count | 1 |
| Rule name | davila-springshell-app |
| Rule app ID | app-B0F0 |
| Attack type | Malformed HTTP Request |
| Protection | Firewall |
| Container ID | ff6dba74a88440d62eecbd9711886266f91febbbcde5193317... |
| Container name | spring-app |
| Image name | docker.io/davidaavilar/springapp:latest |

### HTTP data

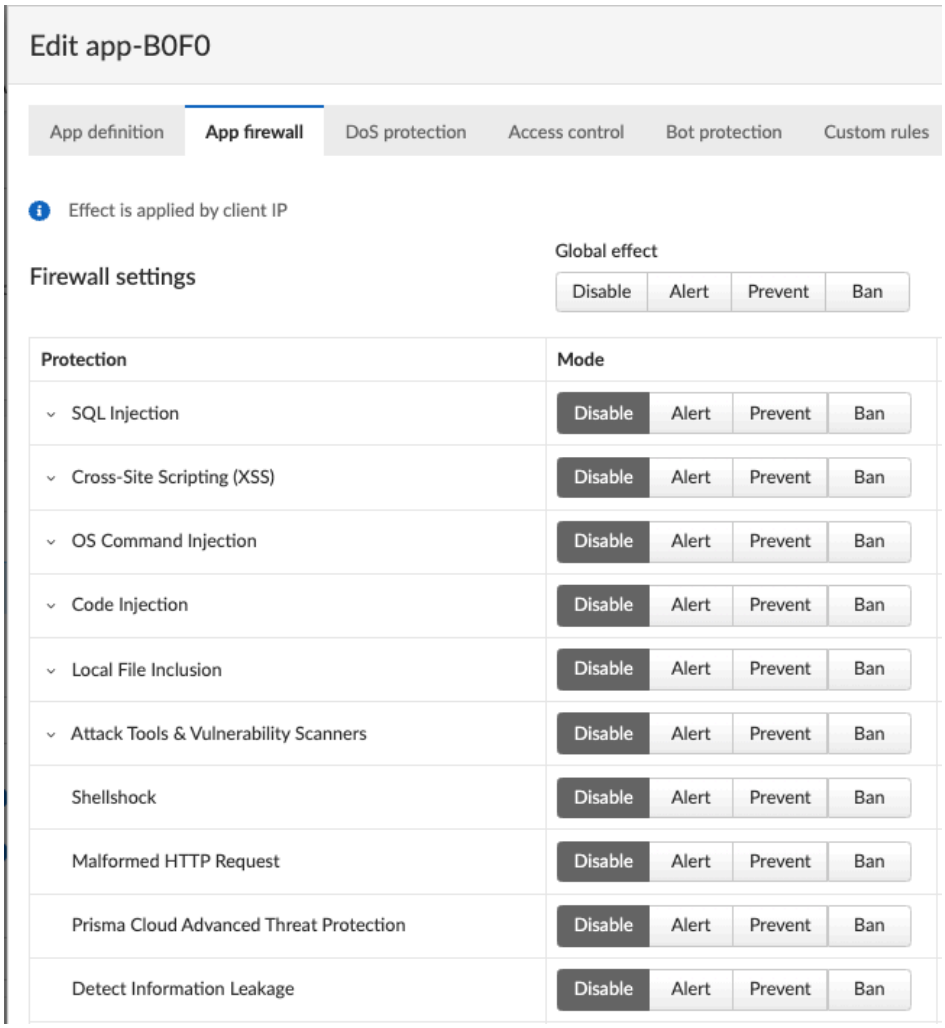
Raw Off

| | |
|---|---|
| Method | GET |
| User-agent | python-requests/2.28.2 |
| Host | ae035635777b949f7a5ceeeb1c73baff-1514304416.us-east-1... |
| Url (Show decoded) | ae035635777b949f7a5ceeeb1c73baff-1514304416.us-east-1... |
| Path | /helloworld/greeting |
| Header names | Accept, Accept-Encoding, Connection, Content-Length, User-A... |

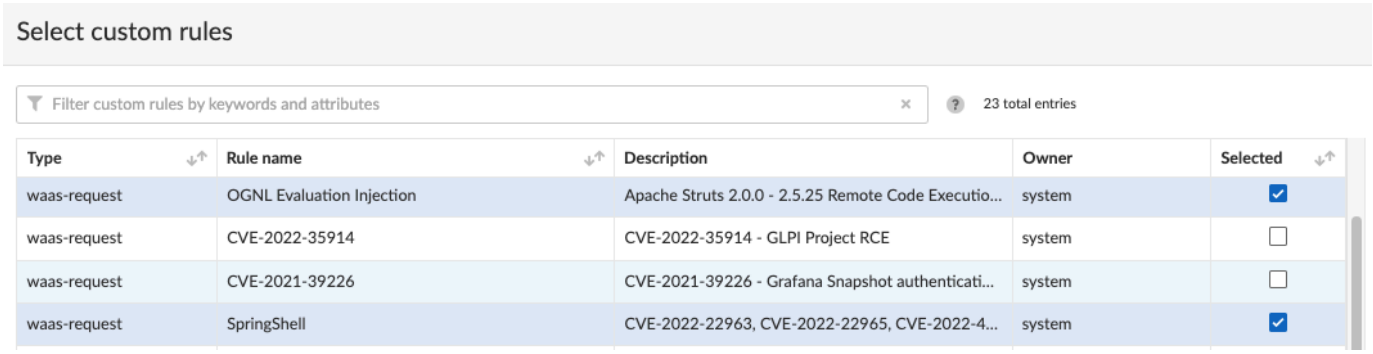
## **Procedure 13:** How to configure Virtual Patching (WAAS) on Prisma Cloud

As you saw, there are two scripts to attack the sample web app you deployed; one for OWASP Top-10 attack (procedure 6) and another one for Virtual Patching.

**Step 1** Under the same WAAS rule you previously created, under the "**App firewall**" tab > **Global Effect** click on "**Disable/Alert**".



**Step 2** Under the "**Custom rules**" tab > click on **"Select rules"** and select **"OGNL Evaluation Injection"**, set it to **"Prevent"** and save it.




**Step 3** Recreate the app pod to be sure it's not vulnerable yet. You can delete the current pod and Kubernetes will create a new one automatically, using these commands below in the image.

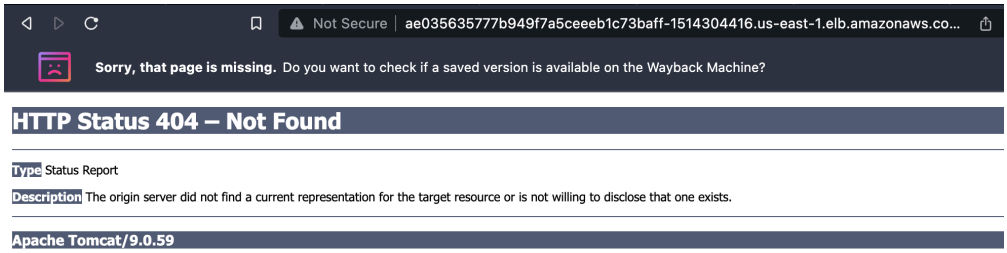
```
davidaavilar 🚀 attack-script % kubectl get pod -n spring-app
NAME                              READY   STATUS    RESTARTS   AGE
spring-app-6f6f4b87bc-rnk5h   1/1     Running   0          5m2s
davidaavilar 🚀 attack-script % kubectl delete pod -n spring-app spring-app-6f6f4b87bc-rnk5h
pod "spring-app-6f6f4b87bc-rnk5h" deleted
davidaavilar 🚀 attack-script % kubectl get pod -n spring-app
NAME                              READY   STATUS    RESTARTS   AGE
spring-app-6f6f4b87bc-fff88   1/1     Running   0          5s
davidaavilar 🚀 attack-script %
```

**Step 4** Try to attack again using the VP script and you should 403 response error.

```
davidaavilar 🚀 attack-script % python3 waas-attack-vp.py
[+]---------------------------------------------------------------------------------------------------------------------------------
[+] The URL to attack is: http://ae035635777b949f7a5ceeeb1c73baff-1514304416.us-east-1.elb.amazonaws.com/helloworld/greeting
[+]---------------------------------------------------------------------------------------------
[*] Resetting Log Variables.
[*] Response code: 200
[*] Modifying Log Configurations
[*] Response code: 403
[*] Send the packet that writes the web shell
[*] Response code: 200
[*] Resetting Log Variables.
[*] Response code: 200
[+] Exploit completed
[+] Check your target for a shell
[+] File: shell.jsp
[+] Shell should be at: http://ae035635777b949f7a5ceeeb1c73baff-1514304416.us-east-1.elb.amazonaws.com/shell.jsp?cmd=id
davidaavilar 🚀 attack-script %
```

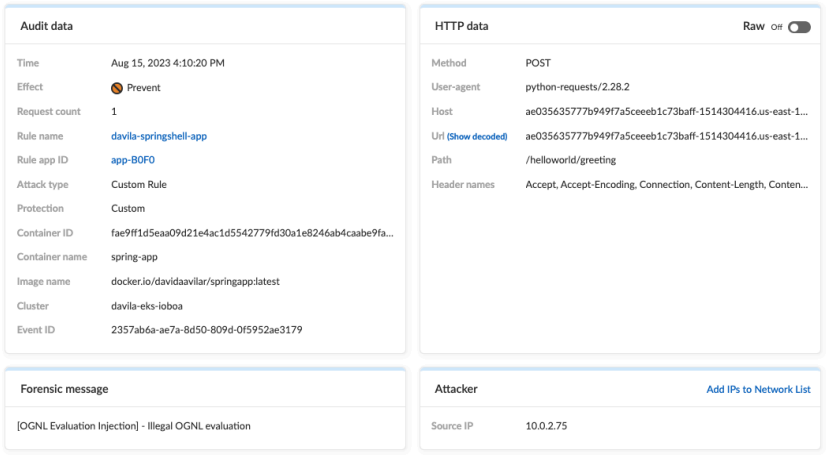
If you go to the URL the exploit should exist, you shouldn't find anything. For example:

*"Shell should be at: http://ae035635777b949f7a5ceeeb1c73baff-1514304416.us-east-1.elb.amazonaws.com/shell.jsp?cmd=id"*




If you get this error "**HTTP Status 404 - Not Found**" is because the attack was **UNSUCCESSFUL** and Prisma Cloud is blocking the attack. The file (script) the exploit is attempting to write wasn't successful.

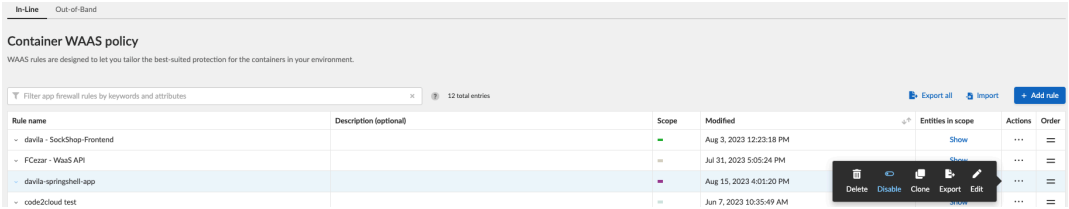
If you go to Prisma Cloud Console and go under **Compute** tab **> Monitor > Events > Containers > WAAS for Containers** and filter by your rule, you should see your attempts:



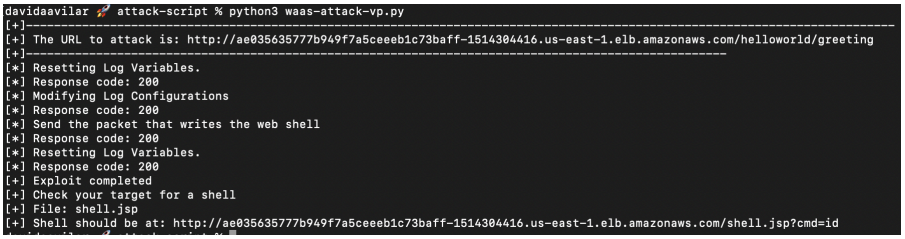
## Procedure 14: How to configure Runtime Protection on Prisma Cloud

Finally, let's suppose you don't have WAAS enabled, neither OWASP Top-10 protection nor virtual patching, you can still stop the process execution from Prisma Cloud using Runtime Protection capabilities.

**Step 1** Disable your WAAS rule under **Compute** tab **> Defend > WAAS > Container > In-Line** and click on the three dots at the right and then **"Disable"**




**Step 2** Attack your application with the VP script and you should see a successful attack:

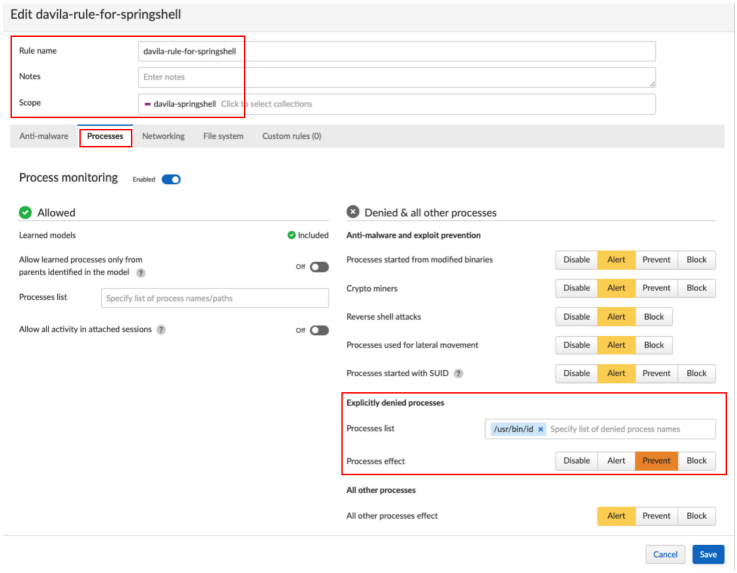



If you go to the URL the exploit should exist, you should find the command is successful. For example:

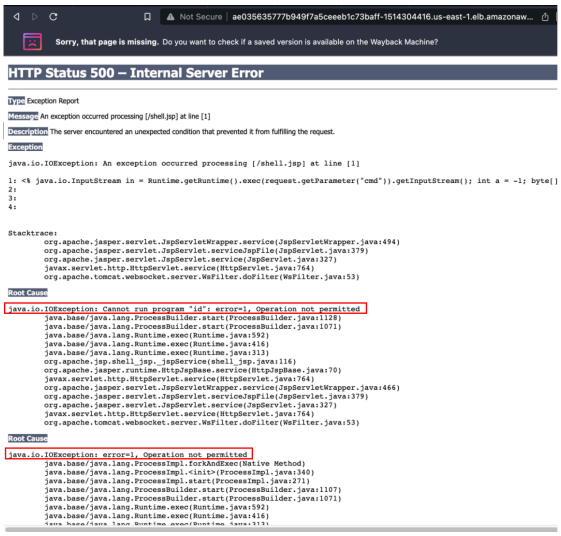

This is a successful attack. The attacker was able to execute the "id" command.

**Step 3** Now, let's create a Runtime Protection rule. Under **Compute** tab **> Defend > Runtime > Container** policy click on **"+ Add rule",** name it and select the Scope you previously created for WAAS rule.

**Step 4** Under the "**Processes**" tab > "**Denied & all other processes"** > "**Explicitly denied processes"** > "**Processes list"** and write "*/usr/bin/id*". And set "**Process effect**" to "**Prevent**" and save it.




**Step 5** If you go to the URL the exploit should exist, you should get an error because Prisma Cloud is blocking the execution of this command



# Revision History

| Date | Revision | Author | Change Summary |
|---|---|---|---|
| 01/02/2025 | 2.1 | David Avila | Add Github Actions procedure |
| | | | |
| | | | |