

Jit



Minimum Viable Security

for Microservices

David Melamed, Co-Founder & CTO at Jit

7 July, 2022





HOME ALONE



Jit



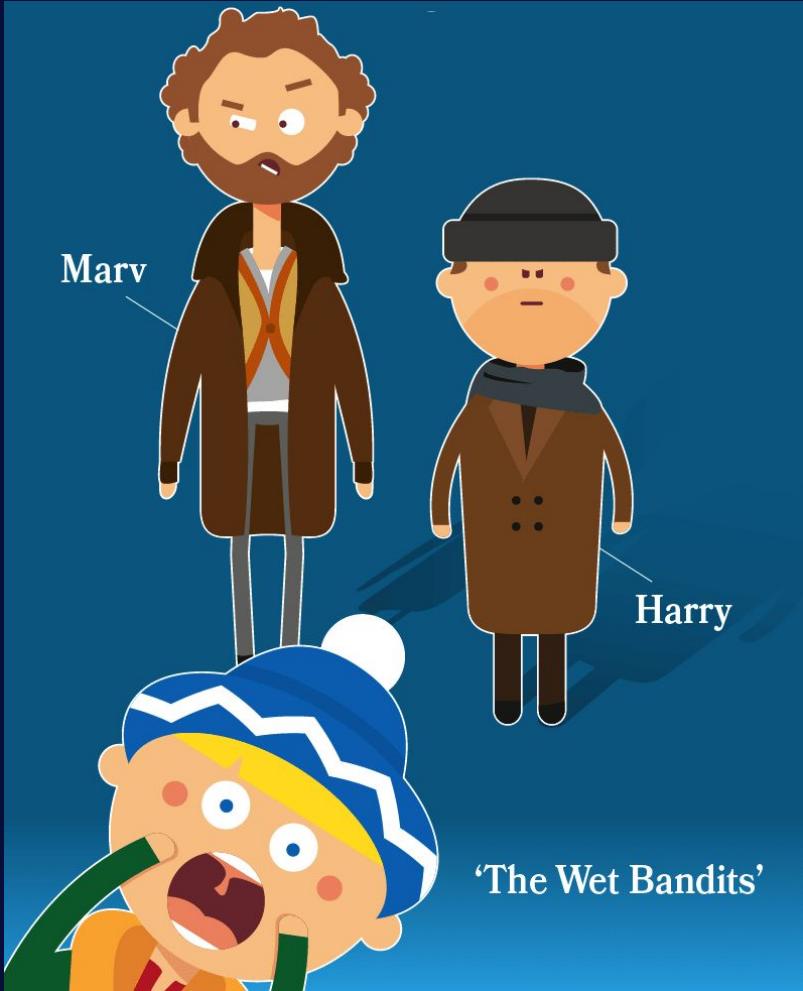
I'm David Melamed

Co-Founder and CTO at **Jit**
Continuous Security Platform for Developers

Tech & Automation Addict

 @dvdmelamed

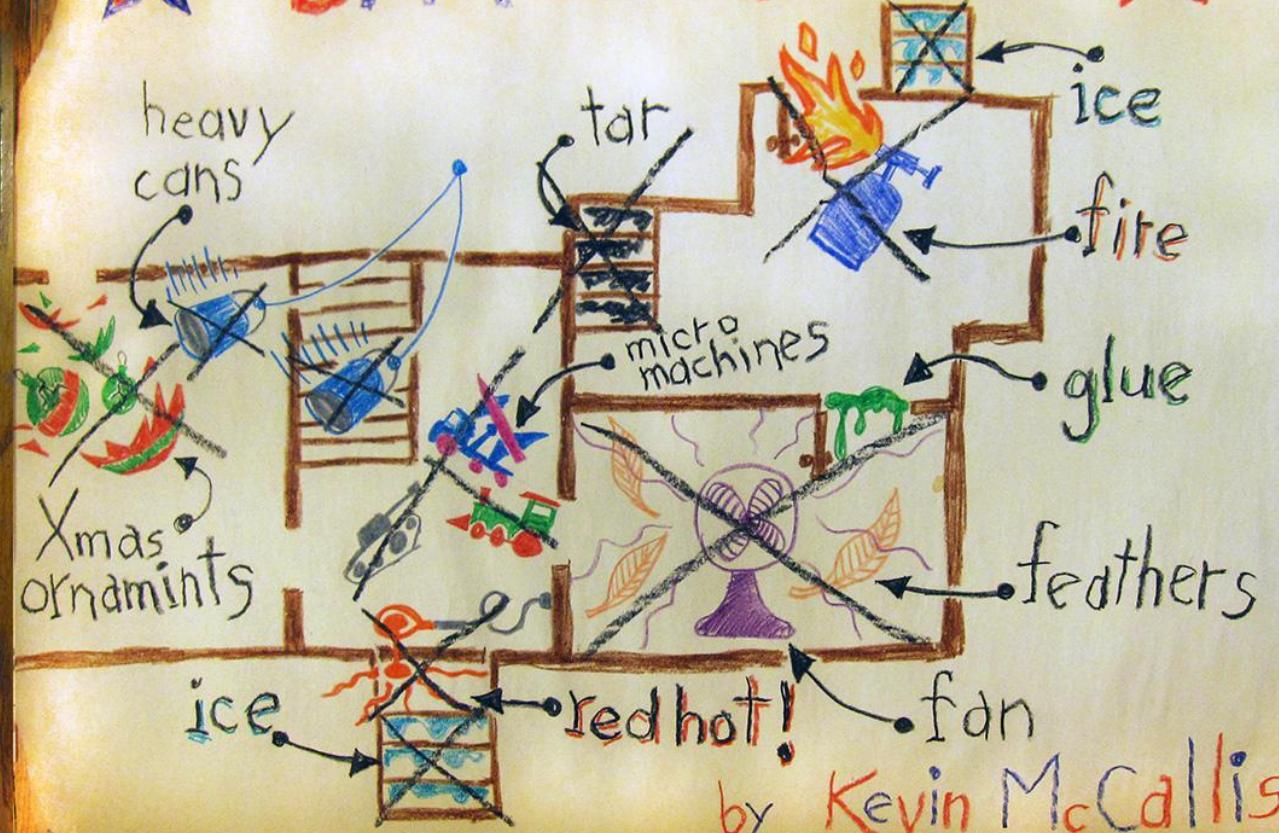
Jit



Source: <https://visual.ly/community/Infographics/entertainment/protecting-your-house-home-alone-style>

Jit

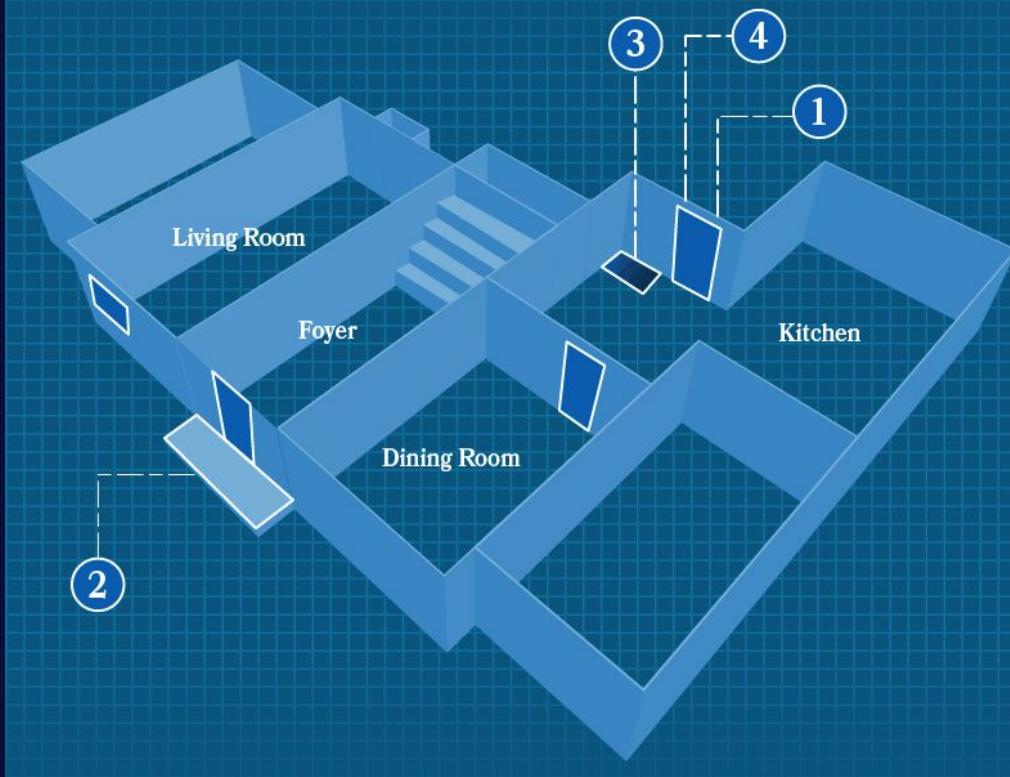
BATTLE PLAN



Jit



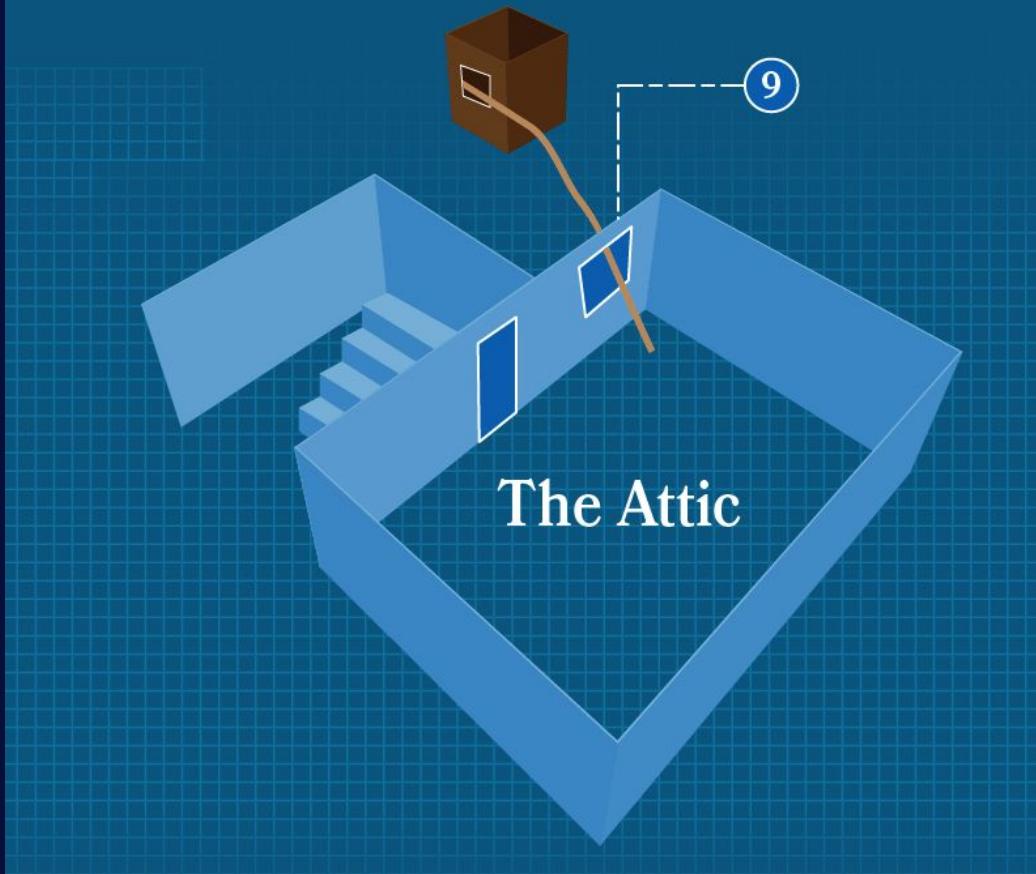
The Ground Floor



Jit



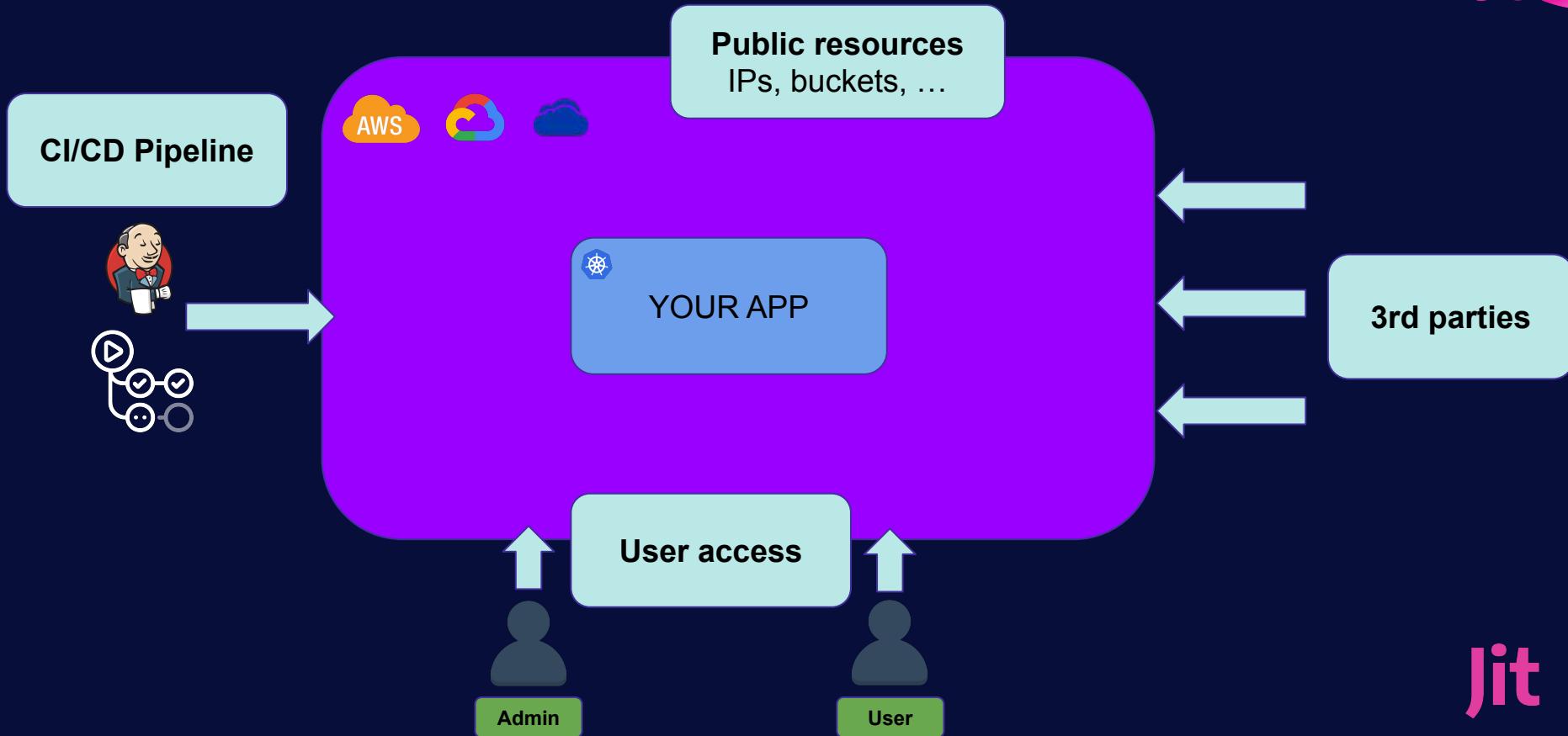
Treehouse



Jit

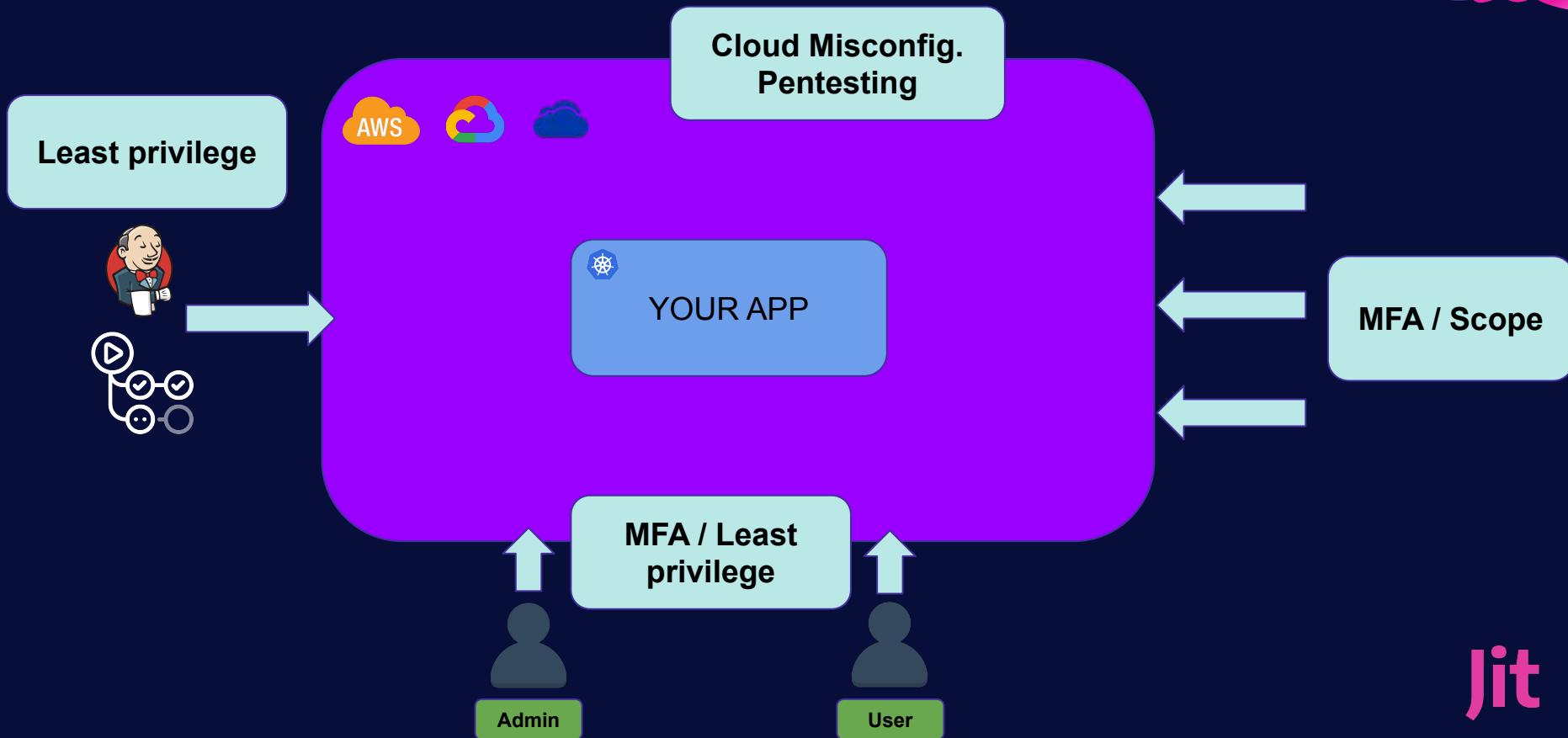


Protecting your perimeter



Jit

Controls to protect your perimeter



Jit



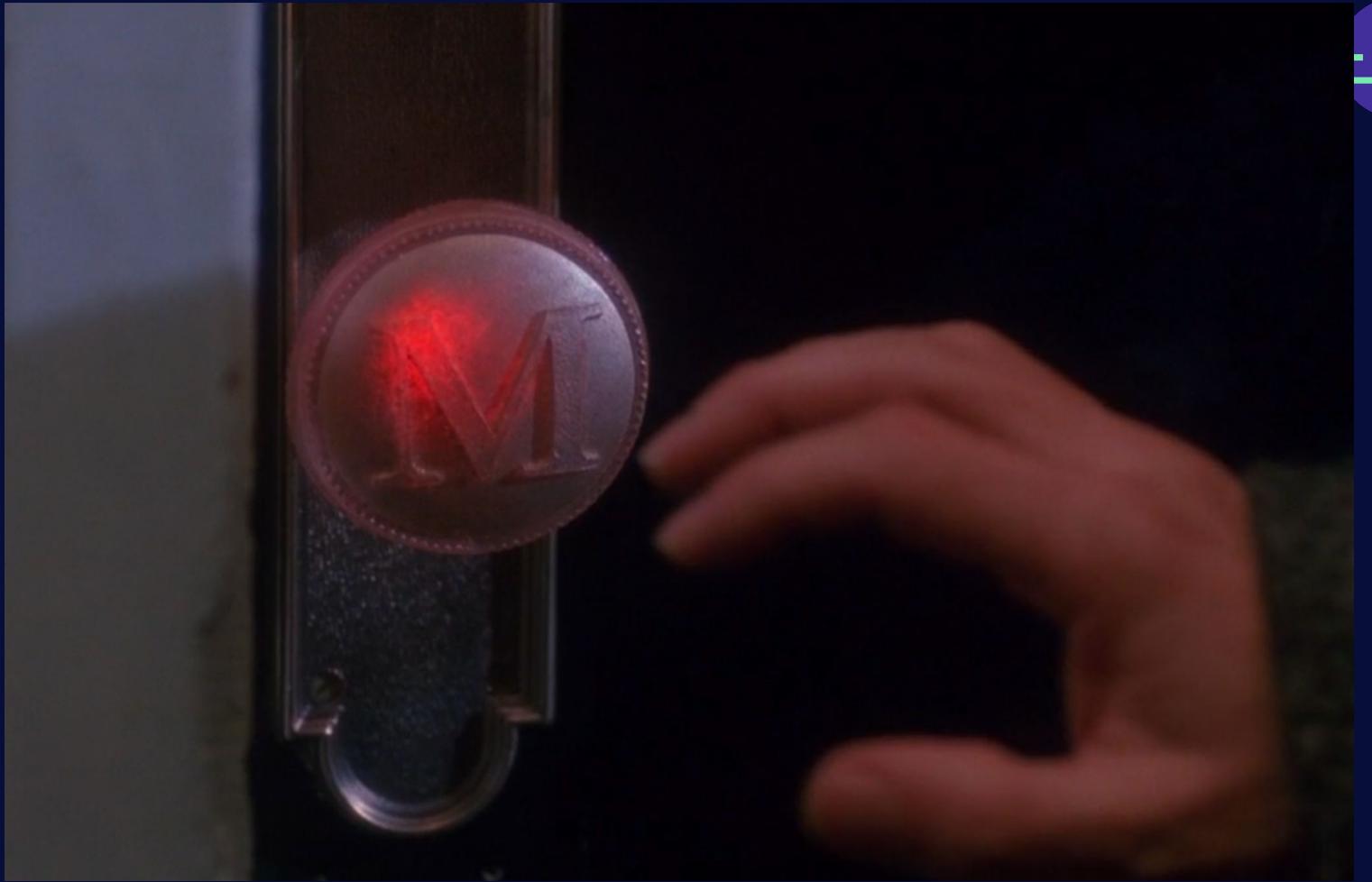
it



Jit



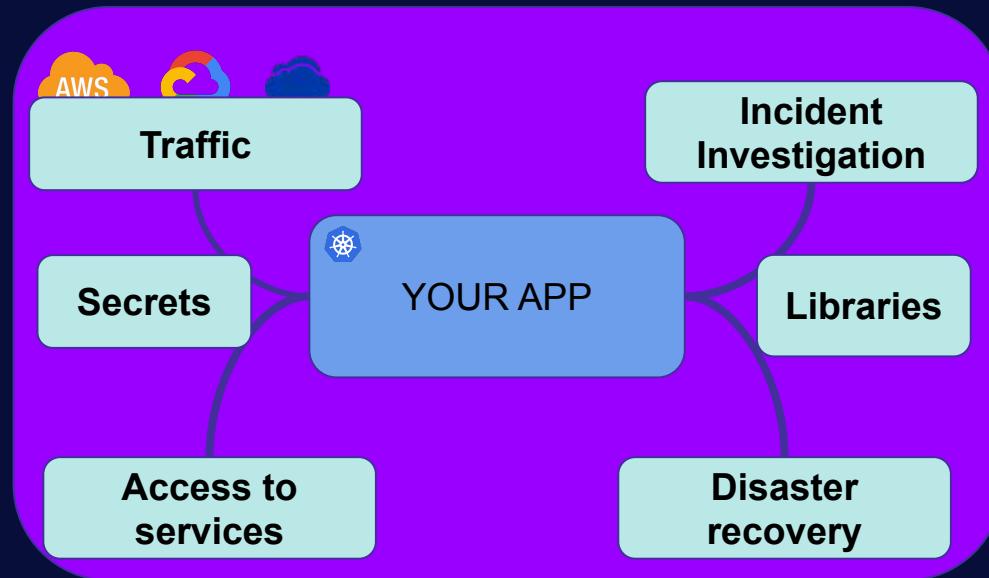
jit



Jit

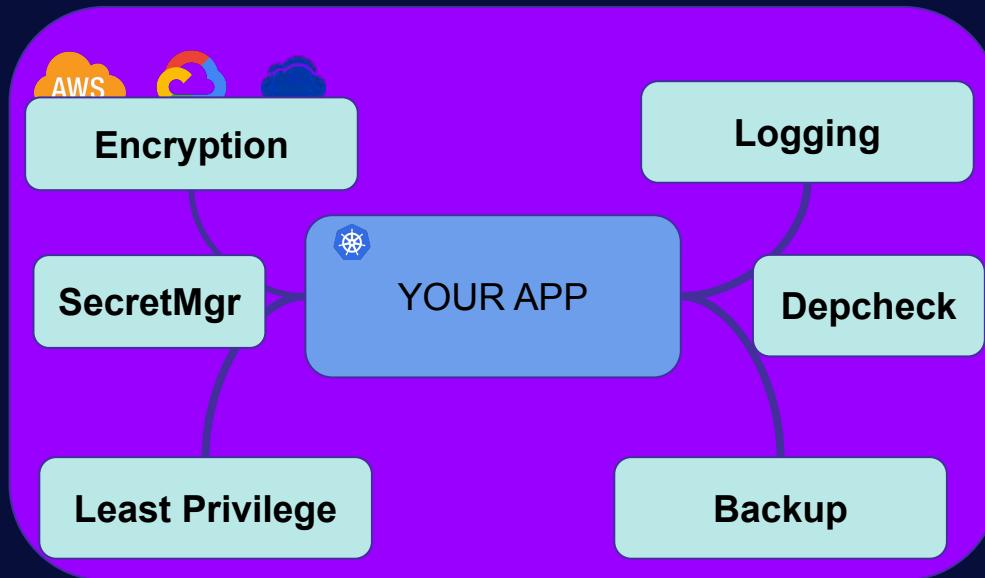


Protecting your app and its data





Controls to protect your app





Jit



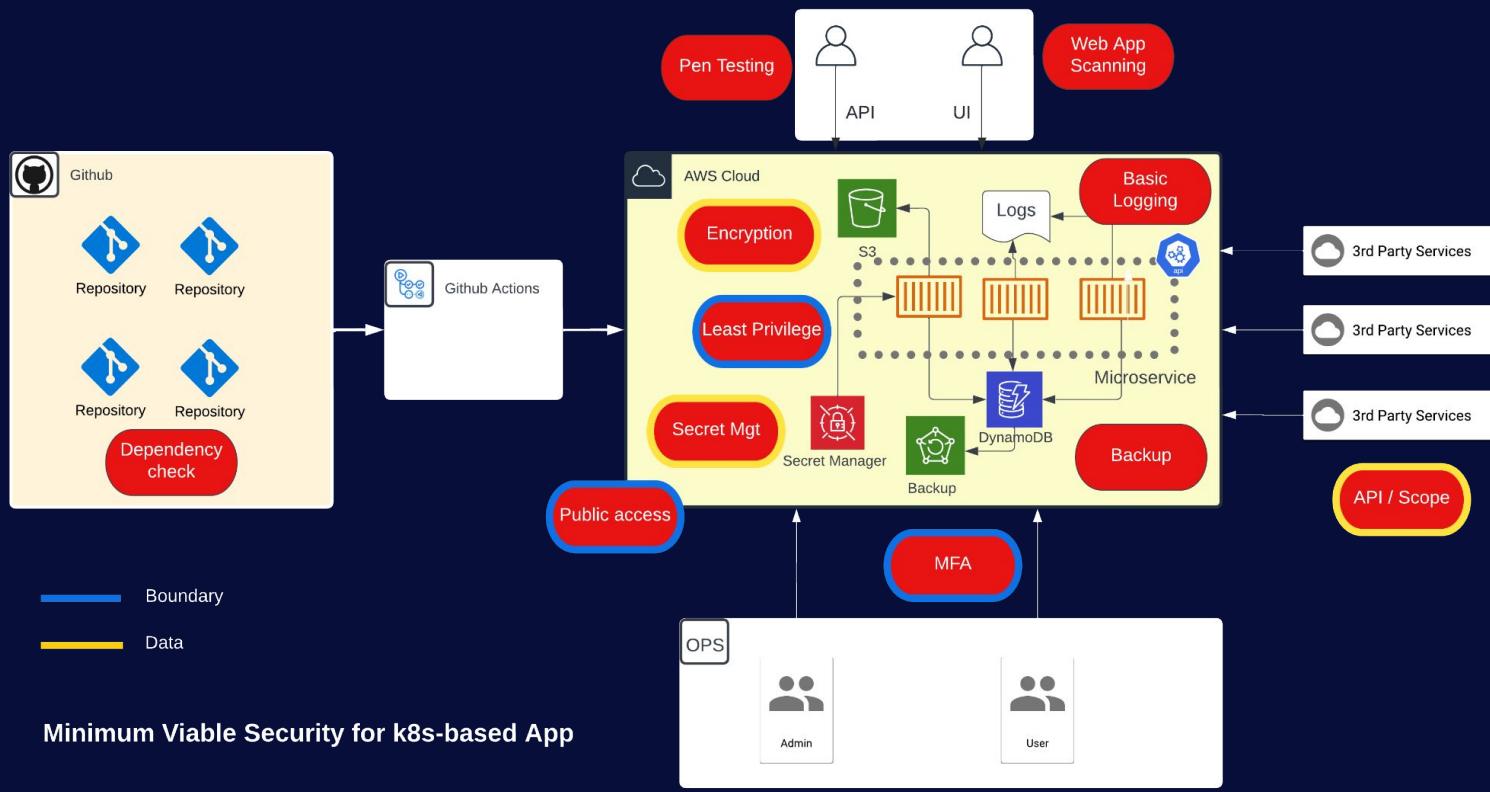
Jit



Jit



Minimum Security Controls



Securing a sample microservice



Example for a Python-based app

- Simple FastAPI-based app to display movies information
- Data persistence: SQLite
- SCM: Github / CI: Github Actions
- Goal: integrate the 7 tools that are part of the MVS in the CI pipeline
- Demo repository
<https://github.com/dvdmelamed/owasp-nz-2022>

Code vulnerabilities



Your Code

- Ensure you don't have vulnerabilities in your code
- Use a Static Application Security Testing (SAST) scanner to detect vulnerabilities based on existing patterns
- Demo: **Bandit**
 - ◆ Security open-source linter for Python source code
 - ◆ Includes 35 rules for detecting vulnerabilities

Secrets

Your Code



- Make sure there are no hard-coded secrets
- Use a scanner that both searches for regexes of well known secret patterns like PAT, Slack token, AWS keys...
- Demo: **Gitleaks**
 - ◆ Supports multiple types of secrets: API keys, AWS credentials, SSH keys...
 - ◆ Supports detecting secrets in git history

Vulnerable Dependencies

Your Code



- Track 3rd parties libraries with disclosed vulnerabilities (CPE / CVE)
- Use a scanner that will track down those vulnerable libraries
- Demo: **dependency-check**
 - ◆ OWASP OSS project
 - ◆ Detects publicly disclosed vulnerabilities contained within a project's dependencies

Infrastructure misconfiguration



Your Infrastructure

- When the infrastructure is expressed as code, it is possible to detect misconfigurations early by scanning the code
- Use a scanner that will look for IaC misconfigurations
- Demo: **KICS**
 - ◆ OSS by Checkmarx supporting many infrastructure types: CloudFormation, Terraform, Ansible, Kubernetes, Helm, Docker, Ansible, ARM...
 - ◆ Include 2000+ checks

Pentesting

Your Runtime



- Simulate attacks on your frontend to ensure it is safe
- Use a pentest / Web Application Scanner
- to test the security of your SaaS
- Demo: **ZED Attack Proxy (ZAP)**
 - ◆ Free web app scanner by OWASP
 - ◆ Includes 17 built-in rules
 - ◆ Supports also API Scanning using OpenAPI or Swagger for endpoint discovery

Vulnerable container images



Your Pipeline

- When building your container images, make sure there is no vulnerability in the base image
- Use a scanner that will scan your container images and enforce your image trust ([Notary](#))
- Demo: [Trivy](#)
 - ◆ OSS by Aqua supporting OS packages and language-based packages
 - ◆ Supports also IaC misconfigurations

Jit

Multi-Factor Authentication (MFA)



Your 3rd Parties

- Ensure you enforce MFA for all 3rd party access
- Make sure MFA is used (custom tool)
- Demo: [MFA on Github](#)

Securing a sample microservice: the tools



Example for a Python-based app

SAST



Bandit

SAST (Secrets)



Gitleaks

IAC



KICS

SCA



OWASP
Dependency-check

Containers



Trivy

DAST



OWASP
ZAP

MFA



Custom

Jit

A Minimum Viable Security plan (1)



01

Your code

Code vulnerability
Secrets
Logging
Vulnerable libraries

02

Your infra

Cloud Misconfiguration
Least Priv. Remote access

03

Your runtime

Pentesting
API Security

04

Your pipeline

Vulnerable containers
Least priv. access

Jit

A Minimum Viable Security plan (2)



05

Your data

Data encryption
Secrets storage

06

Your 3rd parties

Multi-Factor Auth
Secured access

07

Your people

Password manager

08

Your operations

Audit
Backup

Jit

Improving dev-first experience: Jit



The screenshot shows the Jit web interface. At the top, there are navigation links: 'Jit' (highlighted), 'My Plan', 'Findings', and 'Plan Matrix'. Below this, a summary bar shows 'Items Status' with 15 green, 11 red, and 0 blue items, and 'Monitored Repos' with 86 / 87. There are buttons for 'Manage Repos' and 'Commit Plan'. The main area is titled 'Plan Items' with a dropdown menu set to 'My Plan'. It lists three categories: 'Code', 'Infrastructure', and 'Data'. Under 'Code', there are three items: 'Scan code for vulnerabilities' (status 3/3), 'Scan code dependencies for vulnerabilities' (status 3/3), and 'Scan code for hard-coded secrets' (status 3/3). Under 'Infrastructure', there are three items: 'Scan IaC for static misconfigurations' (status 3/5), 'Ensure IAM Roles are Least Privileged' (status 3/5), and 'Conduct periodic vulnerability scans' (status 3/5). Under 'Data', there is one item: 'Allow account direct deletion' (status 5/5). On the right side, a modal window titled 'Scan code for vulnerabilities' provides a detailed description of the tool's purpose and what it will do. It also shows a preview of a pull request comment from 'jit-ci' bot, which detected two important findings related to static misconfigurations in Terraform code.

Customized MVS plan

This screenshot shows a GitHub pull request interface. A comment from the 'jit-ci' bot highlights two important findings. The first finding is about static misconfigurations in Terraform code ('infra.tf') where an S3 bucket lacks MFA Delete. The second finding is about an S3 bucket having public access. Both findings include links to the original code snippets and detailed descriptions of the security controls required to fix them.

infra.tf

```
2 +   bucket = var.bucket_name
3 +   acl    = "public"
4 +
5 + versioning {
```

Security control: Scan IaC For Static Misconfigurations

Type: S3 Bucket Without Enabled Mfa Delete

Description: S3 bucket without MFA Delete Enabled. MFA delete cannot be enabled through Terraform, it can be done by adding a MFA device (https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_mfa_enable.html) and enabling versioning and MFA delete by using AWS CLI: `aws s3api put-bucket-versioning --versioning-configuration>Status=Enabled,MFADelete=Enabled --bucket=<BUCKET_NAME> --mfa=<MFA_SERIAL_NUMBER>`. Please, also notice that MFA delete can not be used with lifecycle configurations

Severity: HIGH

[Learn more about this issue](#)

**Dev-native experience
using PR comments**

Jit

Your next step on the security journey



Jit

Jit



Thank you



Intrigued? Try our free **beta** at jit.io

Inspired? Join us! **We are hiring!**

Questions? Contact me at david@jit.io

