KodeKloud

DevSecOps

linkedin.com/in/barahalikar-siddharth/

mcd-level-2.slack.com

Kubernetes, DevOps & Security

Course Details

Course Structure

- Lecturers
  - Slides
- Demos
  - Step-by-step instructions

Pre-requisites

- Basic Knowledge
  - Linux
  - Shell Scripting
  - DevOps
  - Jenkins
  - Docker
  - Kubernetes

Code and Samples

- Code Snippets
  - GitHub Repo
  - Downloaded Material

Course Objective

#1 Introduction

- Introducing DevSecOps
- Security Aspects
- Tools Explored

#2 Simple DevOps Pipeline

- Setting up the VM
- Installing Software
- Understanding the Usecase
- Basic DevOps Pipeline (4 stages)

#3 DevSecOps Pipeline

- Security to Pipeline (13 stages)
- Deploy to K8S Dev Namespace
- Basic Slack Notifications

#4 Kubernetes Security

- Istio Service Mesh
- Deploy to K8S Prod Namespace
- Monitoring - Falco
- Advanced Slack Notifications

Section #1

#1 Introduction

- Introducing DevSecOps
  - DevOps vs DevSecOps
  - Typical DevOps Process
- Security Aspects
  - Integrate Security into every step of the SDLC
  - Security at Different Stages
- Various Tools
  - Overview of tools used in this course

#2 Simple DevOps Pipeline

- Setting up the VM
- Installing Software
- Understanding the Usecase
- Basic DevOps Pipeline (4 stages)

#3 DevSecOps Pipeline

- Adding Security to Pipeline
- Deploy to K8S Dev environment
- Basic Slack Notifications

#4 Kubernetes Security

- Istio
- Deploy to Prod Environment
- Monitoring - Falco
- Advanced Slack Notifications

KodeKloud

Section #2

#2 Simple DevOps Pipeline

- Setting up the VM
  - Azure Account Creation
  - Azure VM Deployment
- Installing Software
  - Install Docker
  - Setting K8S Cluster
  - Install Java and Maven
  - Install Jenkins/Plugins
- Understanding the Usecase
  - SpringBoot Application
  - NodeJS Application
- Basic DevOps Pipeline (4 stages)
  - Build
  - Test
  - Docker Build
  - Kubernetes Deploy

#1 Introduction

- Introducing DevSecOps
- Security Aspects
- Various Tools

#3 DevSecOps Pipeline

- Adding Security to Pipeline
- Deploy to K8S Dev environment
- Basic Slack Notifications

#4 Kubernetes Security

- Istio Service Mesh
- Deploy to K8S Prod Environment
- Monitoring - Falco
- Advanced Slack Notifications

KodeKloud

Section #3

#3 DevSecOps Pipeline

- Adding security to pipeline
  - Git Hooks - Talisman
  - Testing
    - Mutation - PIT
    - Integration
  - Scans
    - Dependency Checks
    - Trivy - Images
    - OPA - Dockerfile
    - Kubesec - K8S
  - SAST - SonarQube
  - DAST - OWASP ZAP
- Deploy to K8S Dev Namespace
  - Deployment
  - Rollout Status
- Basic Slack Notifications
  - Legacy App Notifications

#1 Introduction

- Introducing DevSecOps
- Security Aspects
- Various Tools

#2 Simple DevOps Pipeline

- Setting up the VM
- Installing Software
- Understanding the Usecase
- Basic DevOps Pipeline (4 stages)

#4 Kubernetes Security

- Istio Service Mesh
- Deploy to Prod Environment
- Monitoring - Falco
- Advanced Slack Notifications

KodeKloud

Section #4

#4 Kubernetes Security

- Istio Service Mesh
  - Installation
  - Kiali
  - mTLS
  - Metrics
- Deploy to Prod Environment
  - CIS Benchmarking
  - Testing
  - Limits
- Monitoring
  - Kube-scan
  - Falco
  - Falco UI
  - Falco Slack
- HELM
- Advanced Slack Notifications
  - Rich Message Layout
  - Emojis

#1 Introduction

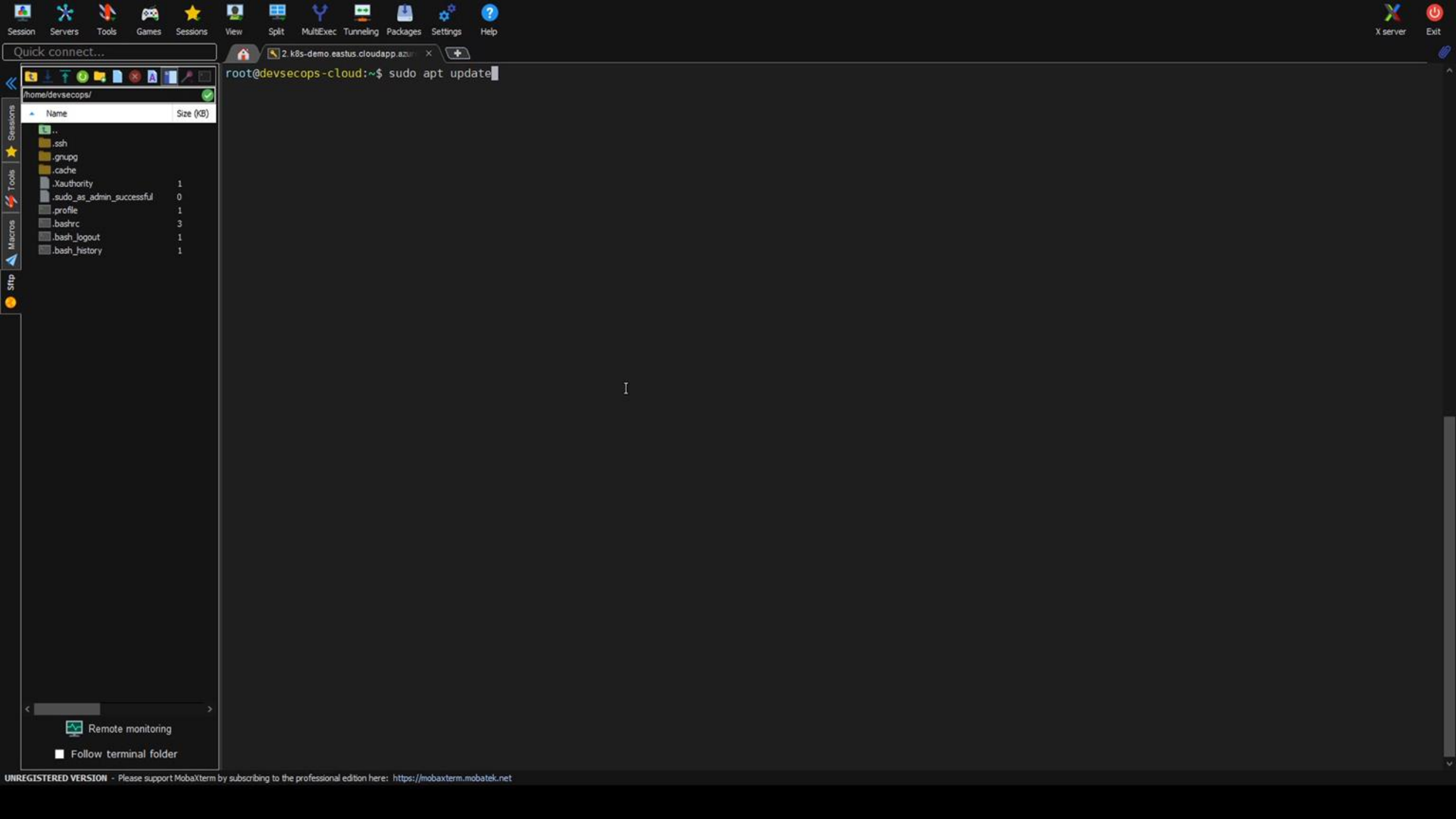
- Introducing DevSecOps
- Security Aspects
- Various Tools

#2 Simple DevOps Pipeline

- Setting up the VM
- Installing Software
- Understanding the Usecase
- Basic DevOps Pipeline (4 stages)

#3 DevSecOps Pipeline

- Adding security to pipeline
- Deploy to K8S Dev Namespace
- Basic Slack Notifications


Session
Servers
Tools
Games
Sessions
View
Split
MultiExec
Tunneling
Packages
Settings
Help
X server
Exit
Quick connect...
2. k8s-demo.eastus.cloudapp.azu
root@devsecops-cloud:~$ sudo apt update
/home/devsecops/
Name
Size (KB)
..
.ssh
.gnupg
.cache
.Xauthority 1
.sudo_as_admin_successful 0
.profile 1
.bashrc 3
.bash_logout 1
.bash_history 1
Sessions
Tools
Macros
Sftp
Remote monitoring
Follow terminal folder
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net

Section #1

#1 Introduction

- Introducing DevSecOps
  - DevOps vs DevSecOps
  - Typical DevOps Process
- Security Aspects
  - Integrate Security into every step of the SDLC
  - Security at Different Stages
- Various Tools
  - Overview of tools used in this course

#2 Simple DevOps Pipeline

- Setting up the VM
- Installing Software
- Understanding the Usecase
- Basic DevOps Pipeline (4 stages)

#3 DevSecOps Pipeline

- Adding Security to Pipeline
- Deploy to K8S Dev environment
- Basic Slack Notifications

#4 Kubernetes Security

- Istio
- Deploy to Prod Environment
- Monitoring - Falco
- Advanced Slack Notifications

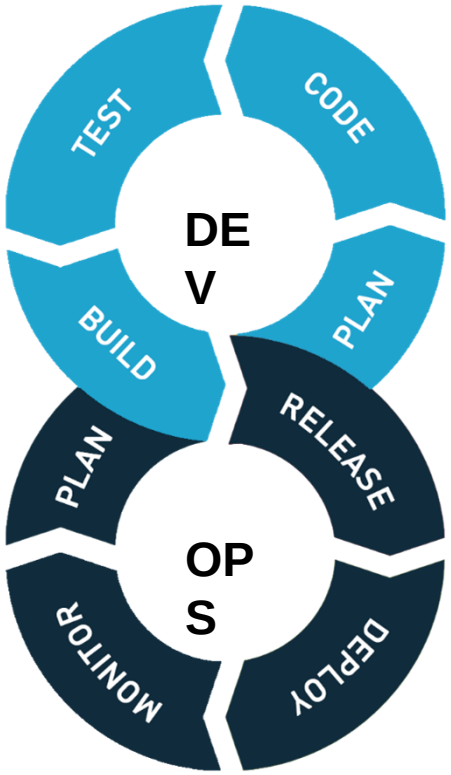
KodeKloud

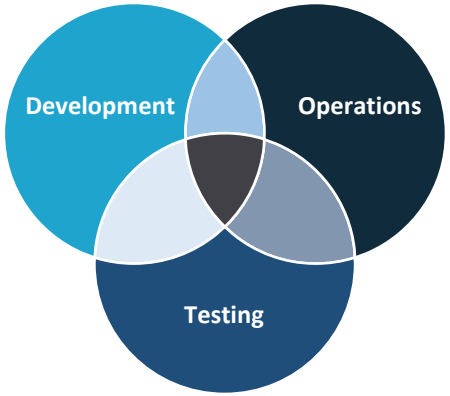
KodeKloud

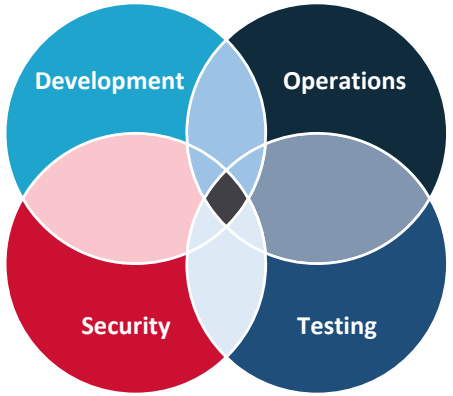
Introducing DevSecOps

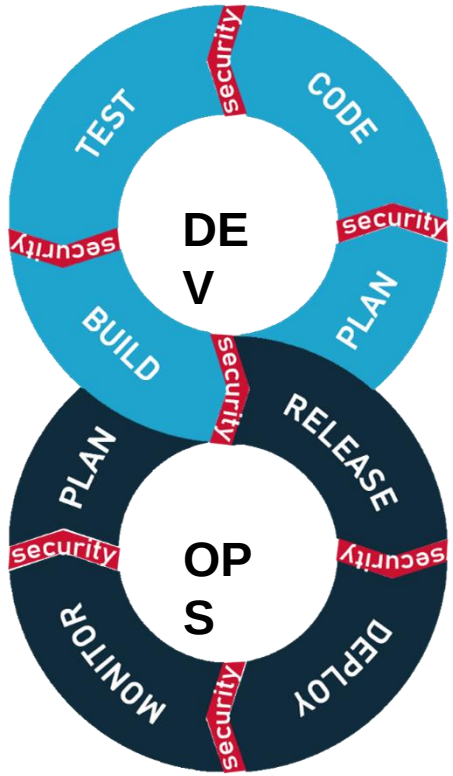
- DevOps vs DevSecOps
- Generic DevOps Process

DevOps vs DevSecOps



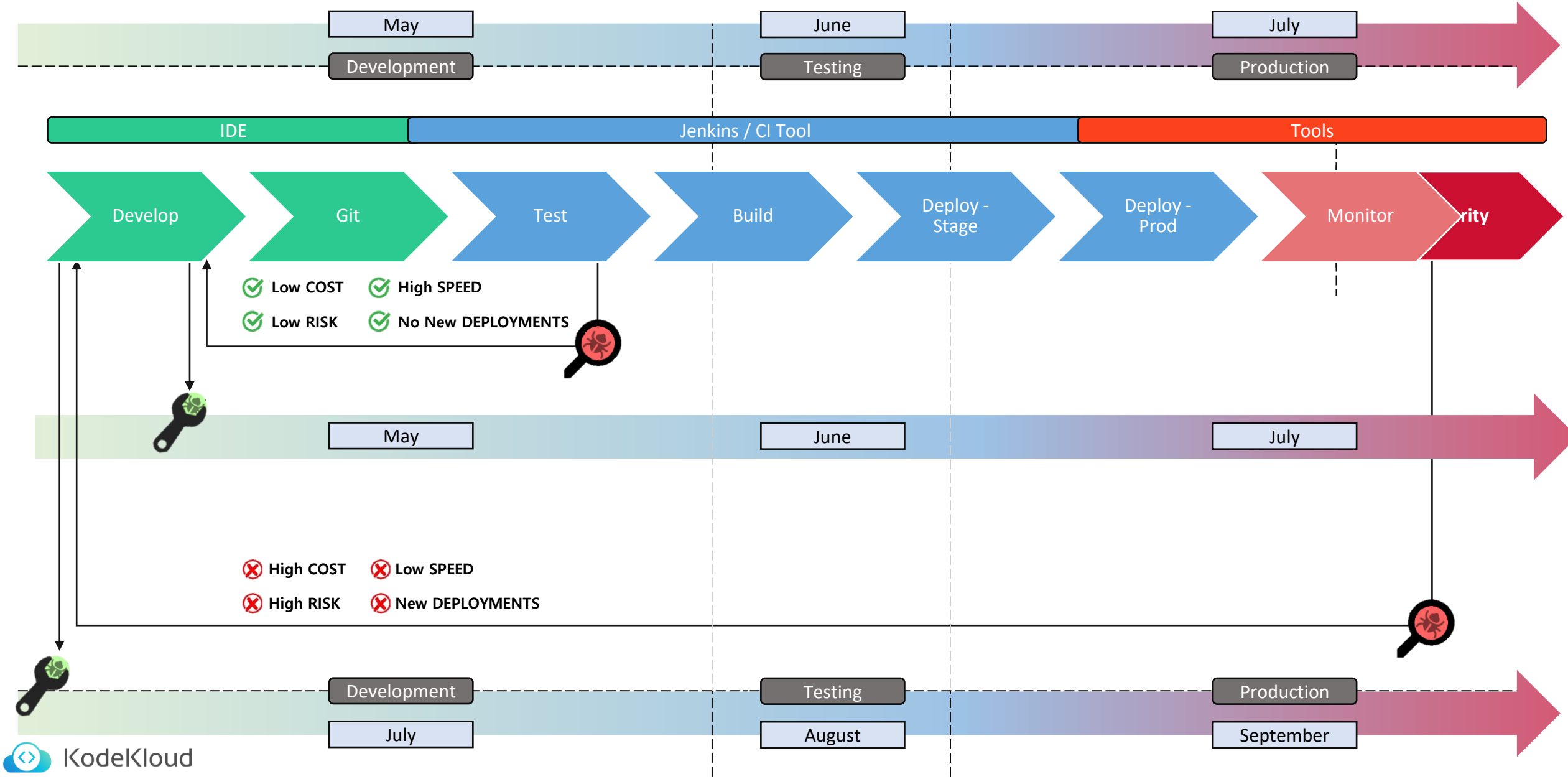






KodeKloud

Typical DevOps Process



KodeKloud

Security Aspects

- Security at each step

KodeKloud

Integrate Security into every step of the SDLC

Configuration Governance

Secure Configurations

Static Code Analysis

Secure Code

Dependency Scanning

Secure Dependencies

Dynamic Application Security Testing

Secure Interfaces

Container Scanning

Secure Artifacts

Continuous Compliance

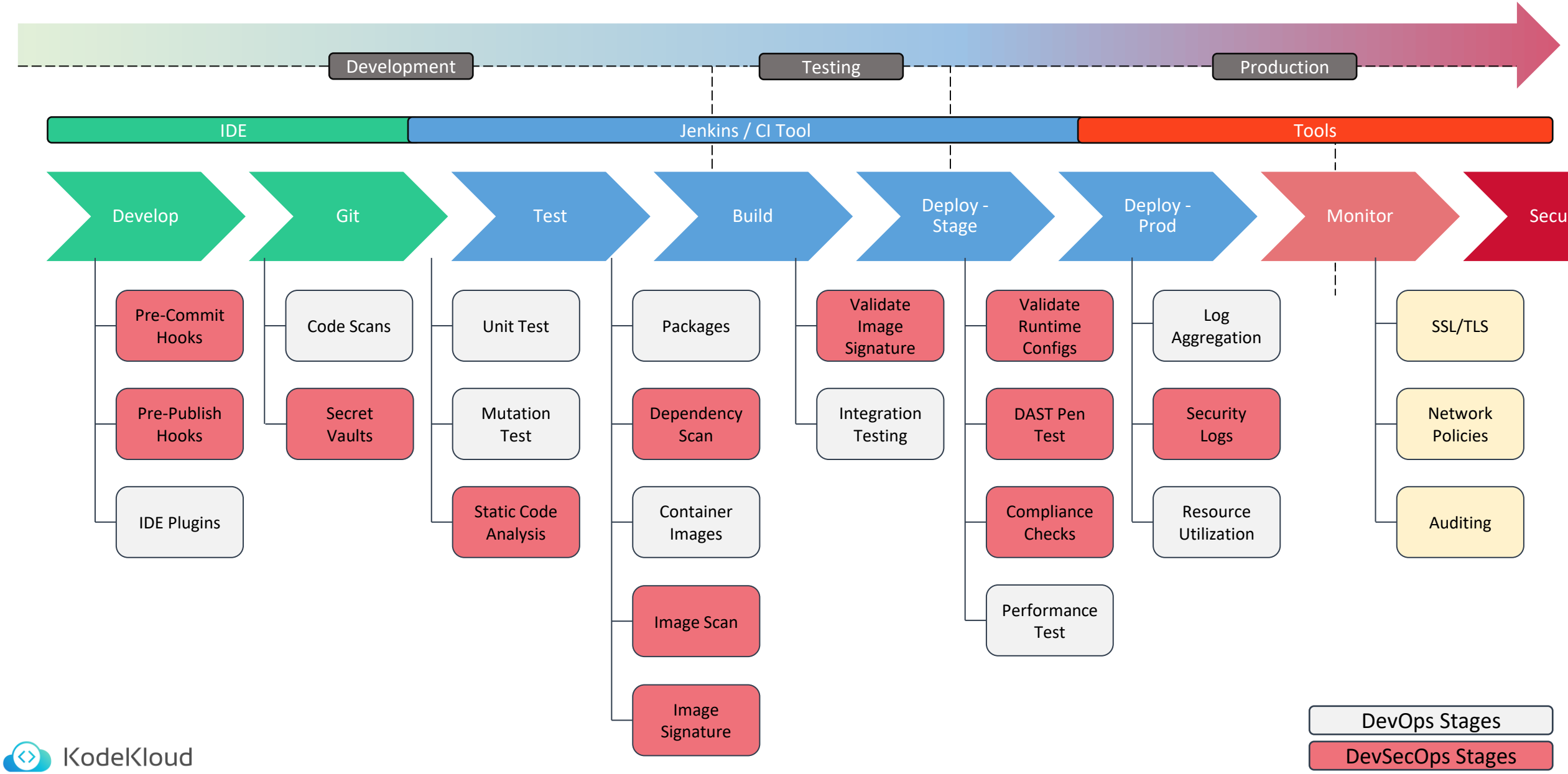
Secure Infrastructure

Runtime Security

Monitor and Prevent

KodeKloud

Security at Different Stages



KodeKloud

KodeKloud

Tools

- Various tool explored

Tools

KodeKloud

Section #2

#2 Simple DevOps Pipeline

- Setting up the VM
  - Azure Account Creation
  - Azure VM Deployment
- Installing Software
  - Install Docker
  - Setting K8S Cluster
  - Install Java and Maven
  - Install Jenkins/Plugins
- Understanding the Usecase
  - SpringBoot Application
  - NodeJS Application
- Basic DevOps Pipeline (4 stages)
  - Build
  - Test
  - Docker Build
  - Kubernetes Deploy

#1 Introduction

- Introducing DevSecOps
- Security Aspects
- Various Tools

#3 DevSecOps Pipeline

- Adding Security to Pipeline
- Deploy to K8S Dev environment
- Basic Slack Notifications

#4 Kubernetes Security

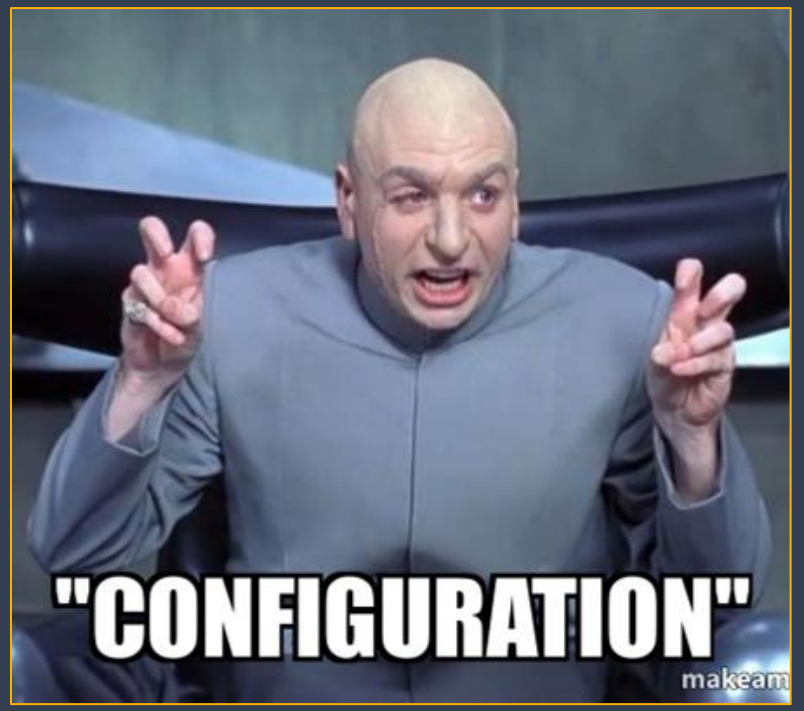
- Istio Service Mesh
- Deploy to K8S Prod Environment
- Monitoring - Falco
- Advanced Slack Notifications

KodeKloud

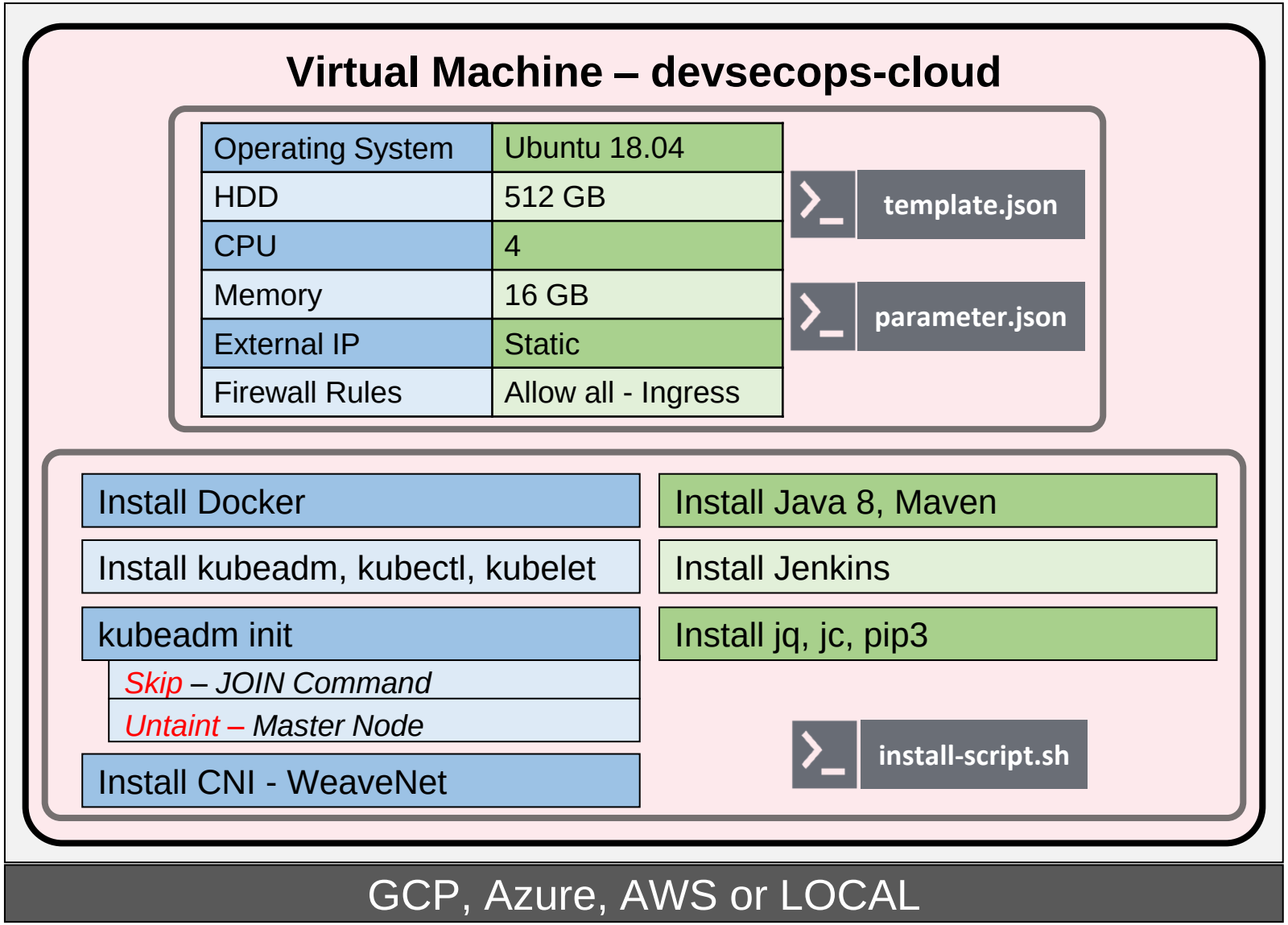
KodeKloud

Setting up the VM

- Virtual Machine Configuration



Virtual Machine



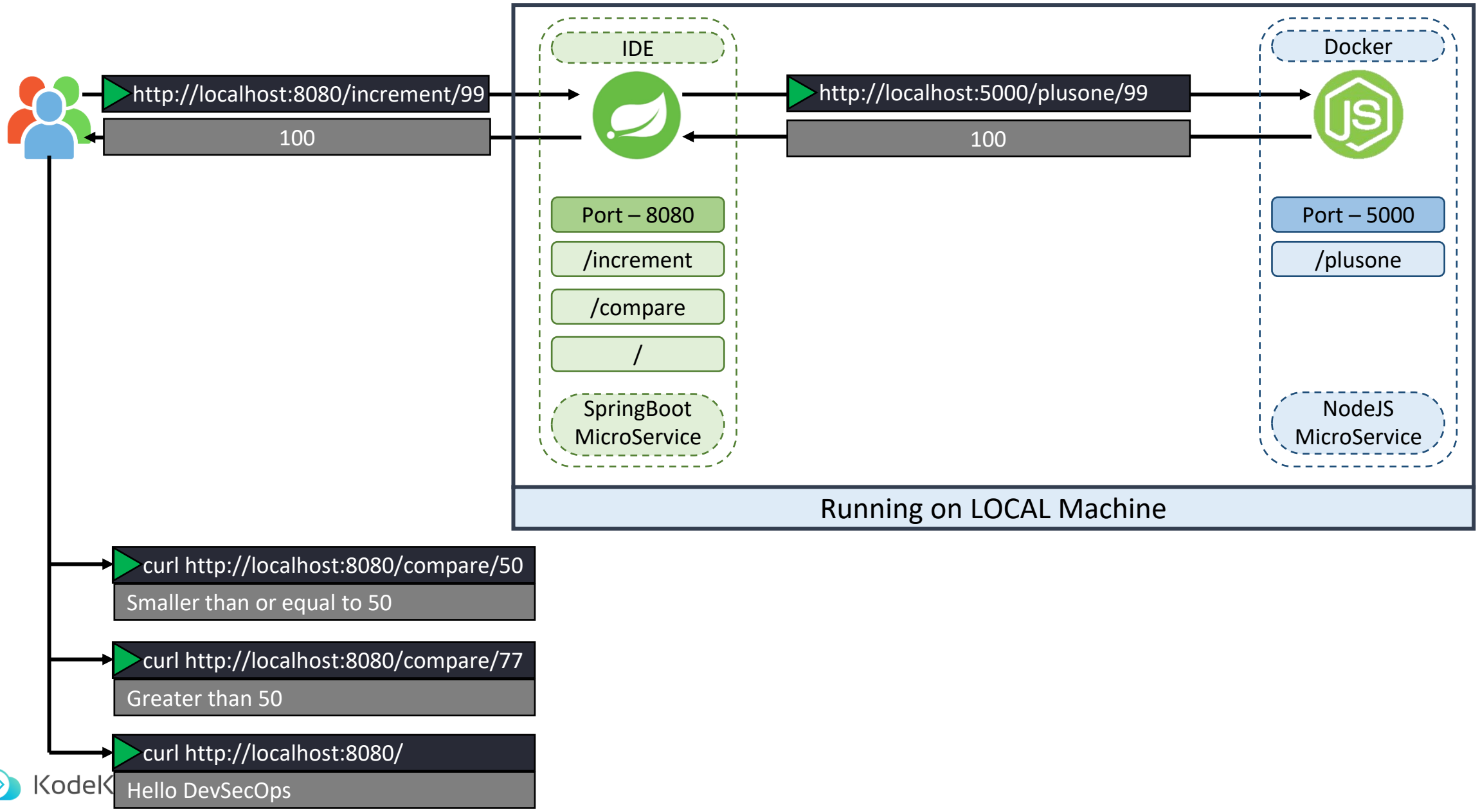
KodeKloud

KodeKloud

Understanding the Usecase

- SpringBoot Application
- NodeJS Application
- API Endpoints

Usecase


IDE
Docker
http://localhost:8080/increment/99
100
http://localhost:5000/plusone/99
100
Port – 8080
/increment
/compare
/
SpringBoot MicroService
Port – 5000
/plusone
NodeJS MicroService
Running on LOCAL Machine
curl http://localhost:8080/compare/50
Smaller than or equal to 50
curl http://localhost:8080/compare/77
Greater than 50
curl http://localhost:8080/
Hello DevSecOps
KodeK

KodeKloud

Maven

- What
- Why
- How

Maven Basics

Apache Maven *is a software project management and comprehension tool.*
*Based on the concept of a project object model (**POM**).*

- Maven can automate a Java based project by,
  - download **dependencies**,
  - **test** the code,
  - **compile** written code into binary code ,
  - **packaging** compiled code into JAR,WAR (artefacts)
  - **deploy** them to the application server or repository.

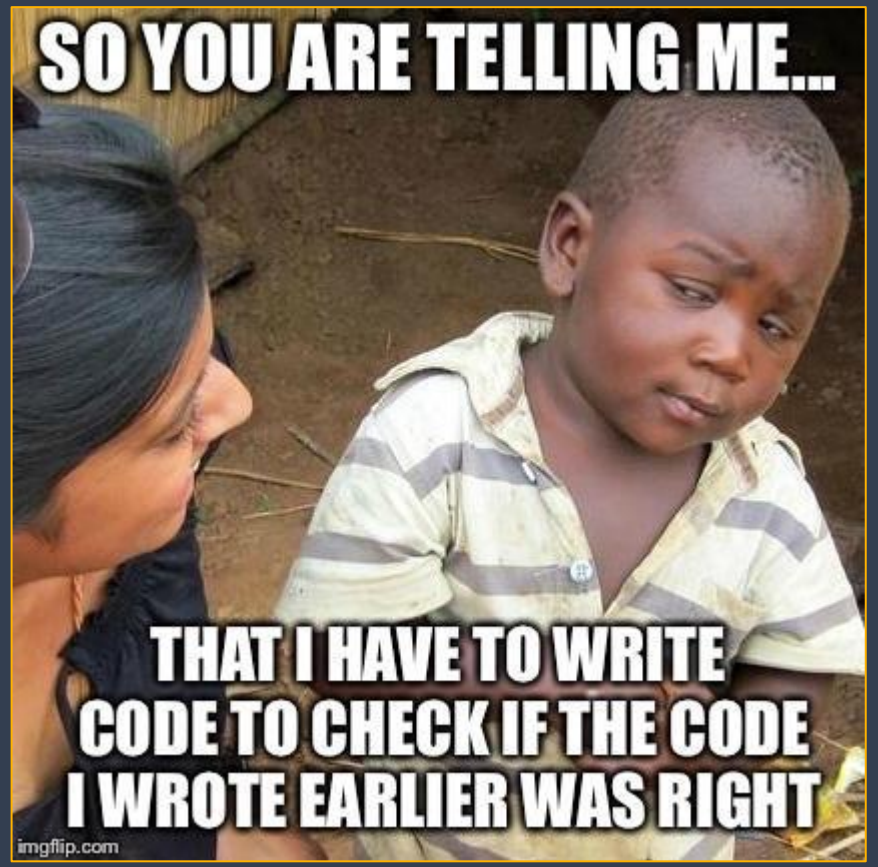
Item	Default
source code	/src/main/java
resources	/src/main/resources
tests	/src/test
complied byte code	/target
distributable JAR	/target/classes

Phase	Handles	Description
validate	validates information	Validates if the project is correct and if all necessary information is available.
compile	compilation	Source code compilation is done in this phase.
test	testing	Tests the compiled source code suitable for testing framework.
package	packaging	This phase creates the JAR/WAR package as mentioned in the packaging in POM.xml.
install	installation	This phase installs the package in local/remote maven repository.
deploy	deploying	Copies the final package to the remote repository.

KodeKloud

Unit Testing

- What, How
- JaCoCo Plugin



Unit Tests

- Unit Testing is a one of the testing done by the developers to make sure individual unit or component functionalities are working fine.
  - **Finds** Bugs Early
  - **Reduce** Cost of Change
  - **Improves** the quality of the code

```
mvn test
```

```
[INFO]
[ERROR] Tests run: 3, Failures: 1, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD FAILURE
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  32.898 s
[INFO] Finished at: 2021-05-29T22:04:53+05:30
[INFO] ------------------------------------------------------------------------
```

- **JaCoCo** is an actively developed line coverage tool, that is **used** to measure how many lines of our code are tested.

Jacoco - Overall Coverage Summary

INSTRUCTION	41%	
BRANCH	100%	
COMPLEXITY	75%	
LINE	61%	
METHOD	71%	
CLASS	100%	

KodeKloud

KodeKloud

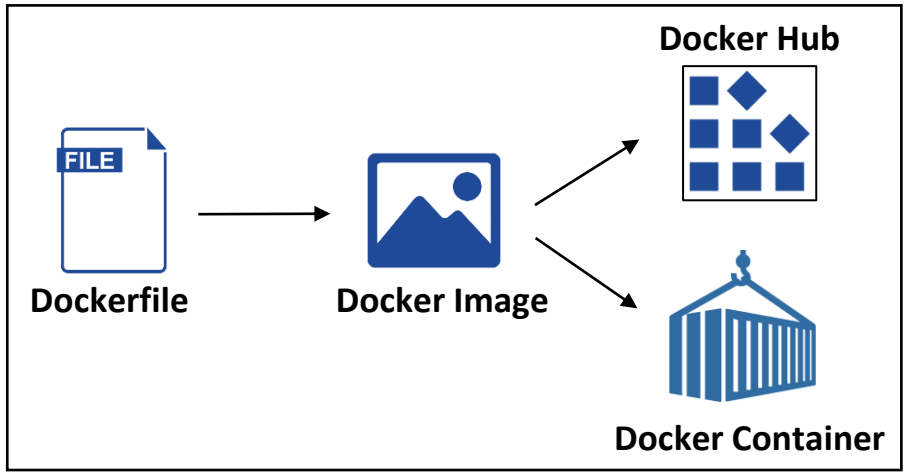
Docker

- Basics
- Commands

KodeKloud

Docker

Docker containers encapsulate everything an application needs to run (and only those things), they allow applications to be shuttled easily between environments.



Dockerfile

```
FROM openjdk:8-jdk-alpine
EXPOSE 9999
ARG JAR_FILE=target/*.jar
ADD ${JAR_FILE} app.jar
ENTRYPOINT ["java","-jar","/app.jar"]
```

```
docker build -t docker-hub-username/my-best-image-ever:v7 .
```

```
docker push docker-hub-username/my-best-image-ever:v7
```

```
docker run -p 9999:9999 docker-hub-username/my-best-image-ever:v7
```

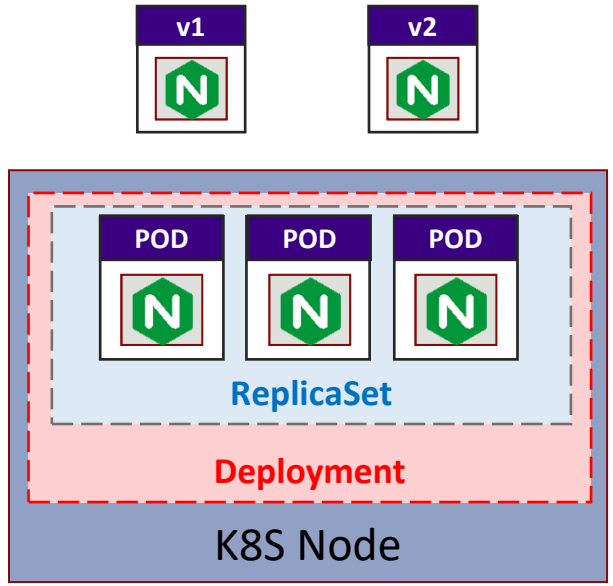
KodeKloud

Kubernetes

- What?
- Pod
- Replica Set
- Deployment

Kubernetes Deployment

A **Deployment** provides updates for both **Pods** and **ReplicaSets**.



pod.yam

```yaml
apiVersion: v1
kind: Pod
metadata:
  labels:
    run: nginx-pod
    app: frontend
  name: nginx-pod
spec:
  containers:
  - image: nginx
    name: nginx-pod
```

replicaset.yam

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: frontend-rs
  labels:
    app: frontend
spec:
  replicas: 2
  selector:
    matchLabels:
      app: frontend
  template:
    metadata:
      labels:
       app: frontend
       run: nginx-pod
    spec:
      containers:
      - name: nginx-pod
        image: nginx
```

deployment.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend-deploy
  labels:
    app: frontend
spec:
  replicas: 2
  selector:
    matchLabels:
      app: frontend
  template:
    metadata:
      labels:
       app: frontend
       run: nginx-pod
    spec:
      containers:
      - name: nginx-pod
        image: nginx
```

kubectl apply -f deployment.yaml

kubectl get pod,deploy,replicaset

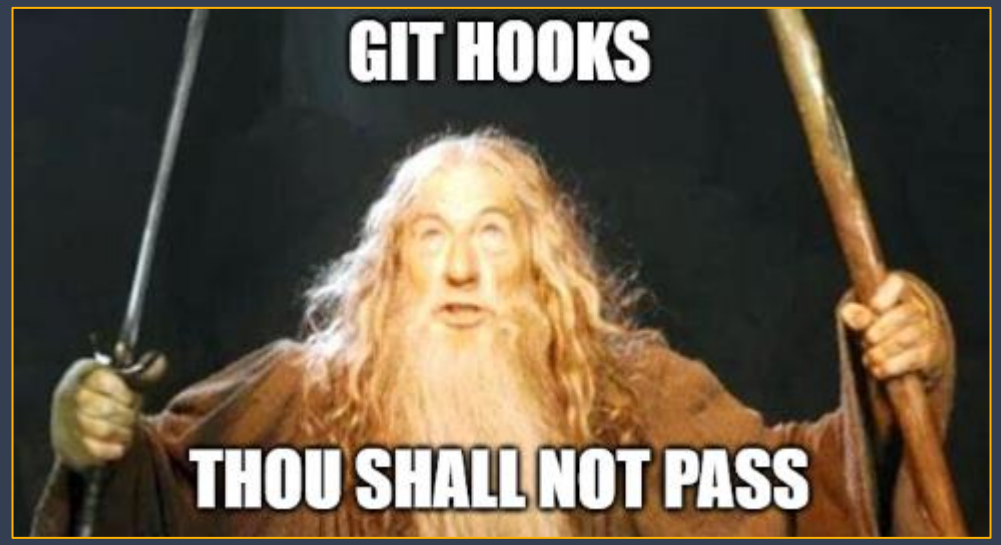
```
NAME                                    READY   STATUS    RESTARTS   AGE
pod/frontend-deploy-6bcf78fbf7-mdd5j    1/1     Running   0          43s
pod/frontend-deploy-6bcf78fbf7-xsdmc    1/1     Running   0          43s

NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/frontend-deploy     2/2     2            2           43s

NAME                                          DESIRED   CURRENT   READY
replicaset.apps/frontend-deploy-6bcf78fbf7    2         2         2
```

KodeKloud

KodeKloud

Git Hooks

- Hooks
  - Pre-commit
  - Pre-push
- Talisman



KodeKloud

Pre-commit / Pre-push Hooks

- Sensitive information such as the **access keys, access tokens, SSH keys etc**. are often erroneously **leaked** due to accidental git commits.
- Pre-commit hooks can be installed on developer's workstations to avoid them.
- Work on pure Regex-based approach for filtering sensitive data
- If developers want they can bypass this step.

Dev put AWS keys on Github. Then BAD THINGS happened

Fertile fields for Bitcoin yields - with a nasty financial sting

Darren Pauli Tue 6 Jan 2015 // 13:02 UTC

Bots are crawling all over GitHub seeking secret keys, a developer served with a $2,375 Bitcoin mining bill found.

DevFactor founder Andrew Hoffman said he used Figaro to secure Rails apps which published his Amazon S3 keys to his GitHub account.

He noticed the blunder and pulled the keys within five minutes, but that was enough for a bot to pounce and spin up instances for Bitcoin mining.

https://www.theregister.com/2015/01/06/dev_blunder_shows_github_crawling_with_keyslurping_bots/

KodeKloud

Talisman

- Talisman installs a hook to your repository to ensure that potential secrets or sensitive information do not leave the developer's workstation.
- It validates the outgoing change for things that look suspicious like potential SSH keys, authorization tokens, private keys, etc.

Global Installation - Talisman will thus be present, not only in your existing git repositories, but also in any new repository that you 'init' or 'clone'.

Global - pre-commit hook:

```
curl --silent https://raw.githubusercontent.com/thoughtworks/talisman/master/global_install_scripts/install.bash > /tmp/install_talisman.bash
&& /bin/bash /tmp/install_talisman.bash
```

Global - pre-push hook:

```
curl --silent  https://raw.githubusercontent.com/thoughtworks/talisman/master/global_install_scripts/install.bash > /tmp/install_talisman.bash
&& /bin/bash /tmp/install_talisman.bash pre-push
```

Single Project Installation - Talisman will be present only in a single git repository.

Single project - pre-push hook:

```
curl https://thoughtworks.github.io/talisman/install.sh > ~/install-talisman.sh
```

```
chmod +x ~/install-talisman.sh
```

```
cd my-git-project
```

```
~/install-talisman.sh
```

```
Downloading talisman_linux_amd64 from https://github.com/thoughtworks/talisman/releases/download/v1.11.0/talisman_linux_amd64
Downloading checksums from https://github.com/thoughtworks/talisman/releases/download/v1.11.0/checksums
talisman_linux_amd64: OK

Talisman successfully installed to '.git/hooks/pre-push'.
```

Talisman Reports and Ignore Contents

git push main

```
Talisman Report:
+----------+--------------------------------+----------+
|   FILE   |             ERRORS             | SEVERITY |
+----------+--------------------------------+----------+
| test.txt | Potential secret pattern :     | low      |
|          | password=password              |          |
+----------+--------------------------------+----------+
```

Talisman works based on pattern matching

- Encoded Values
- File Content
- File Size
- Entropy
- Credit Card Num
- File Name

git push main

```
Talisman Scan: 3 / 3 <----------------------------------------------------

Talisman Report:
+------+--------------------------------+----------+
| FILE |             ERRORS             | SEVERITY |
+------+--------------------------------+----------+
| aws  | Potential secret pattern :     | low      |
|      | pikey=5589 4513 5412 4562      |          |
+------+--------------------------------+----------+

filename: aws
checksum: 14e3763161c34c1855181f0624588b3f5f1cebf4f241dd23f4f01a5f9793ba45

? Do you want to add aws with above checksum in talismanrc ? No
```

Bypass/Skip Talisman Hook

git push origin --no-verify

Remove Talisman

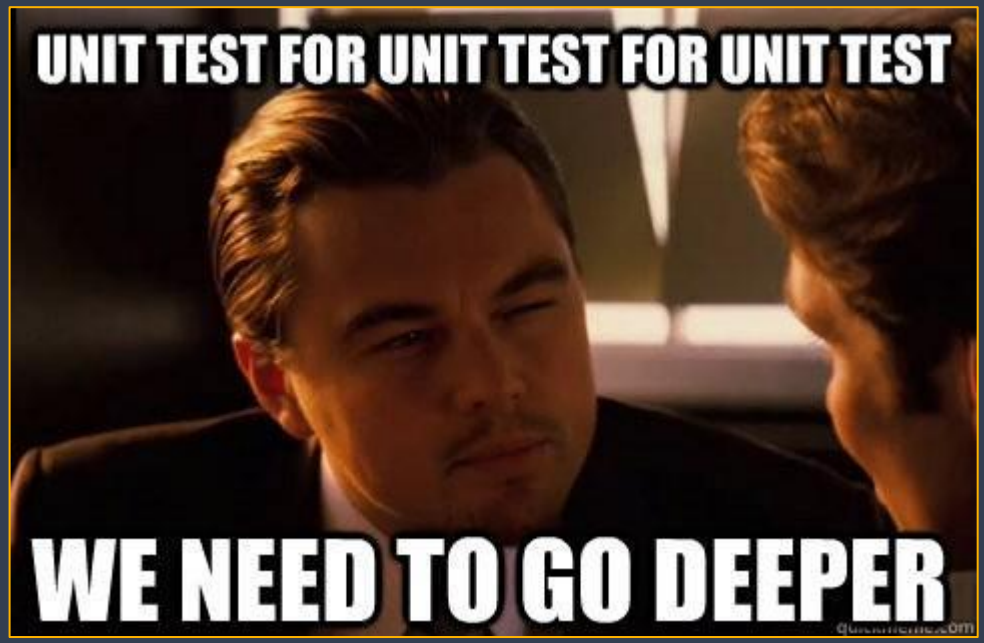
remove the hook manually by deleting the Talisman pre-commit or pre-push hook from .git/hooks folder in repository.

```
? Do you want to add aws with above checksum in talismanrc ? Ye
                                                              s
root@controlplane:~/test# cat .talismanrc
fileignoreconfig:
- filename: aws
  checksum: 14e3763161c34c1855181f0624588b3f5f1cebf4f241dd23f4f01a5f9793ba45
```

KodeKloud

Mutation Tests

- What?
- Why?
- How?
- Demo



KodeKloud

Mutation Tests – PIT Test (PIT)

Mutation testing in some aspect is testing your actual Unit Test cases.

What?

- PIT runs your unit tests against automatically modified versions of your application code.
- When the application code changes, it should produce different results and cause the unit tests to **fail**.
- If a unit test does not **fail** in this situation, it may indicate an issue with the test.

```
31 1        return "Kubernetes DevSecOps";
32  }
33
34  @GetMapping("/compare/{value}")
35  public String compareToFifty(@PathVariable int value) {
36          String message = "Could not determine comparison";
37 2        if (value > 50) {
38                  message = "Greater than 50";
39          } else {
40                  message = "Smaller than or equal to 50";
41          }
42
43 1        return message;
```

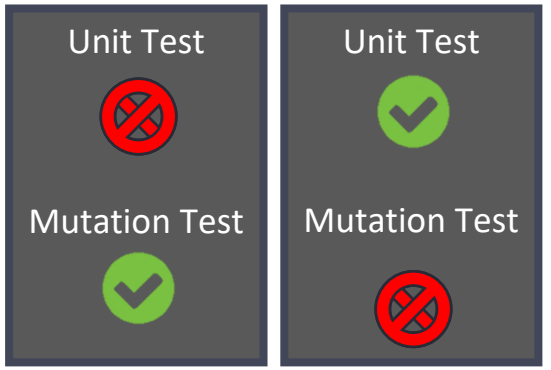
light green shows line coverage,
dark green shows mutation coverage.
light pink show lack of line coverage,
dark pink shows lack of mutation coverage.

Why?

- Traditional test coverage (i.e line) measures only which code is **executed** by your tests.
- It does **not** check that your tests are actually able to **detect faults** in the executed code.
- It is therefore only able to identify code that is definitely **not tested**.

How?

- Mutation Testing changes the behavior of our classes, these changes are called as **Mutation**.
  - if your tests **fails** after the mutation, then we can say the ***mutation was caught and Killed by our Junit***.
  - if your tests **pass** after the mutation, then we can say the ***mutation was not caught thus the mutation Survived.***
  - the quality of your tests can be gauged from the percentage of mutations ***killed***.



KodeKloud

Why Mutation Test Survived?

Unit Test

```
@GetMapping("/compare/{value}")
public String compareToFifty(@PathVariable int value) {
    String message = "Could not determine comparison";
    if (value > 50) {
        message = "Greater than 50";
    } else {
        message = "Smaller than or equal to 50";
    }
    return message;
}
```

```
@Test
public void smallerThanOrEqualToFiftyMessage() throws Exception {
    this.mockMvc.perform(get("/compare/49")).andDo(print()).andExpect(status().isOk())
            .andExpect(content().string("Smaller than or equal to 50"));
}
```

actual condition is **value > 50**
value = 49
message = Smaller than or equal to 50

Unit Test ✓

Mutation Test

```
@GetMapping("/compare/{value}")
public String compareToFifty(@PathVariable int value) {
    String message = "Could not determine comparison";
    if (value >= 50) {
        message = "Greater than 50";
    } else {
        message = "Smaller than or equal to 50";
    }
    return message;
}
```

```
@Test
public void smallerThanOrEqualToFiftyMessage() throws Exception {
    this.mockMvc.perform(get("/compare/49")).andDo(print()).andExpect(status().isOk())
            .andExpect(content().string("Smaller than or equal to 50"));
}
```

mutation changed condition to **value >= 50**
value = 49
message = Smaller than or equal to 50

Unit Test ✓
Mutation Test ⊘

Mutation Test

```
@GetMapping("/compare/{value}")
public String compareToFifty(@PathVariable int value) {
    String message = "Could not determine comparison";
    if (value >= 50) {
        message = "Greater than 50";
    } else {
        message = "Smaller than or equal to 50";
    }
    return message;
}
```

```
@Test
public void smallerThanOrEqualToFiftyMessage() throws Exception {
    this.mockMvc.perform(get("/compare/50")).andDo(print()).andExpect(status().isOk())
            .andExpect(content().string("Smaller than or equal to 50"));
}
```

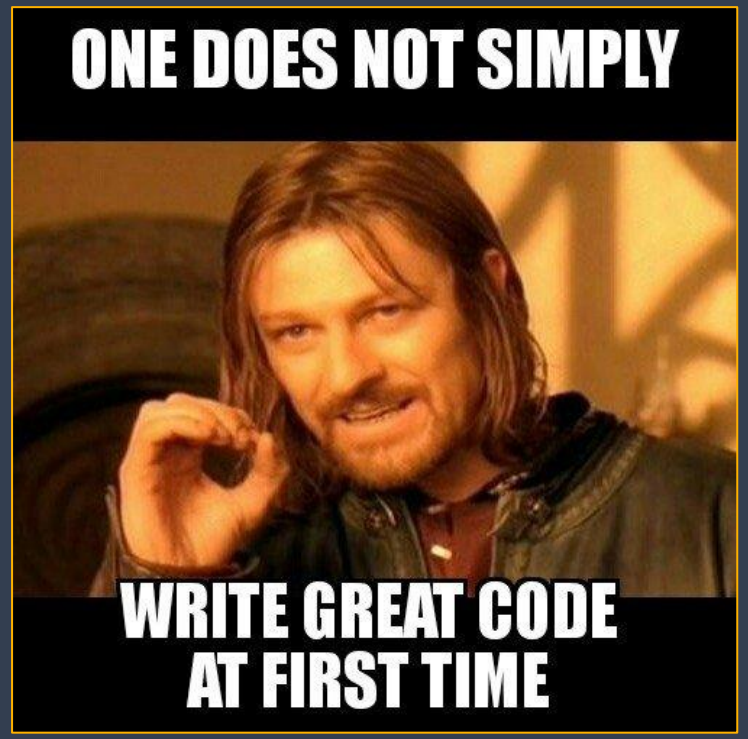
mutation changed condition to **value >= 50**
value = **50**
message = Greater than 50

Unit Test ⊘
Mutation Test ✓

KodeKloud

SAST - SonarQube

- SonarQube
  - Why?
  - Problem?
  - Solution

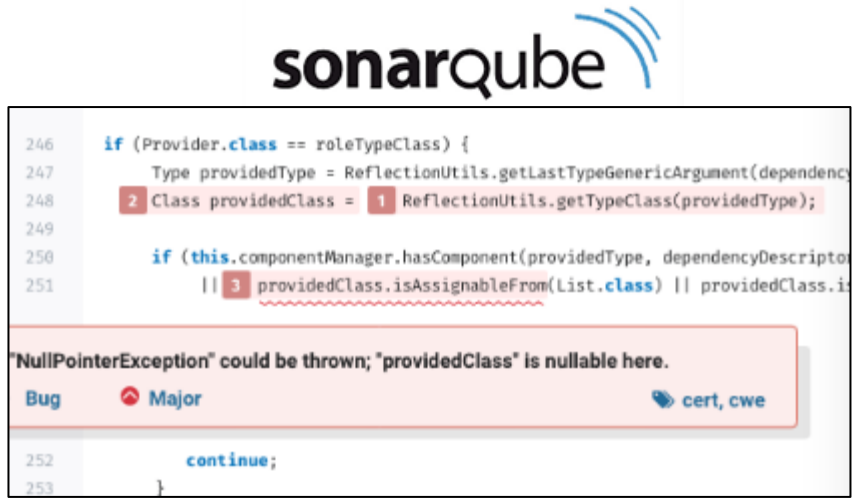


SonarQube

SonarQube is an open-source platform developed by SonarSource for continuous inspection of code quality to perform automatic reviews with static analysis of code.

Why?

- It helps in detecting areas in the code that needs refactoring or simplification.
- It can help to find the bug early in the development cycle, which means less cost to fix them.
- We can define project specific rules which will then be implemented without manual intervention.



Problem?

- Simply having visibility into code is not enough and in order to address the issues flagged by code analysis, we need to make use of different data insights that we get from SonarQube.

Solution?

- **Quality Gates**
  - is the best way to ensure that standards are met and regulated in projects.
  - can be defined as a set of threshold measures set on your project.

Conditions on New Code

Metric	Operator	Value
Code Smells	is greater than	10

Conditions on Overall Code

Metric	Operator	Value
Condition Coverage	is less than	60.0%

```
docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest
```

KodeKloud

Vulnerabilities

- What?
  - Vulnerability,
  - Exploit,
  - Threat,
  - Risk,
  - CVE,
  - CVSS,
  - NVD
- Dependency Checks
- Trivy
- **OPA Conftest**
- Kubesec

KodeKloud

What is Vulnerability, Exploit, Threat, Risk, CVE, CVSS, NVD?

VULNERABILITY

Application Bugs that hackers could **exploit**

+

EXPLOIT

The next step through which a **vulnerability** can be leveraged for malicious activity by hackers

+

THREAT

Hackers could use multiple **exploits** to damage or make use of the application

=

RISK

The potential for damage when a **threat exploits** a **vulnerability**

Application Code

- Broken **Authentication**
- **SQL** Injections
- **Security** Misconfigurations
- Outdated **Dependency**

Vulnerability Scanning

- **NVD** - National Vulnerability Database
- **CVE** - Common Vulnerability or Exposure
- **CVSS** - Common Vulnerability Scoring System
- **CWE** - Common Weakness Enumeration

Vulnerability Assessment

- 67 Issues Found
- 02 **CVE** - Severity - Critical
- 19 **CVE** - Severity - Medium
- 46 **CVE** - Severity - Low

Vulnerability Remediation

- **Detecting** Vulnerabilities
- **Prioritize** CVEs
- **Patch** Vulnerabilities
- **Monitor** for newly discovered

KodeKloud

Dependency Check

What?

- Dependency-check is an open source project created by OWASP
- It is a software composition analysis tool that identifies project dependencies on open-source code and checks if there are known vulnerabilities associated with that code.

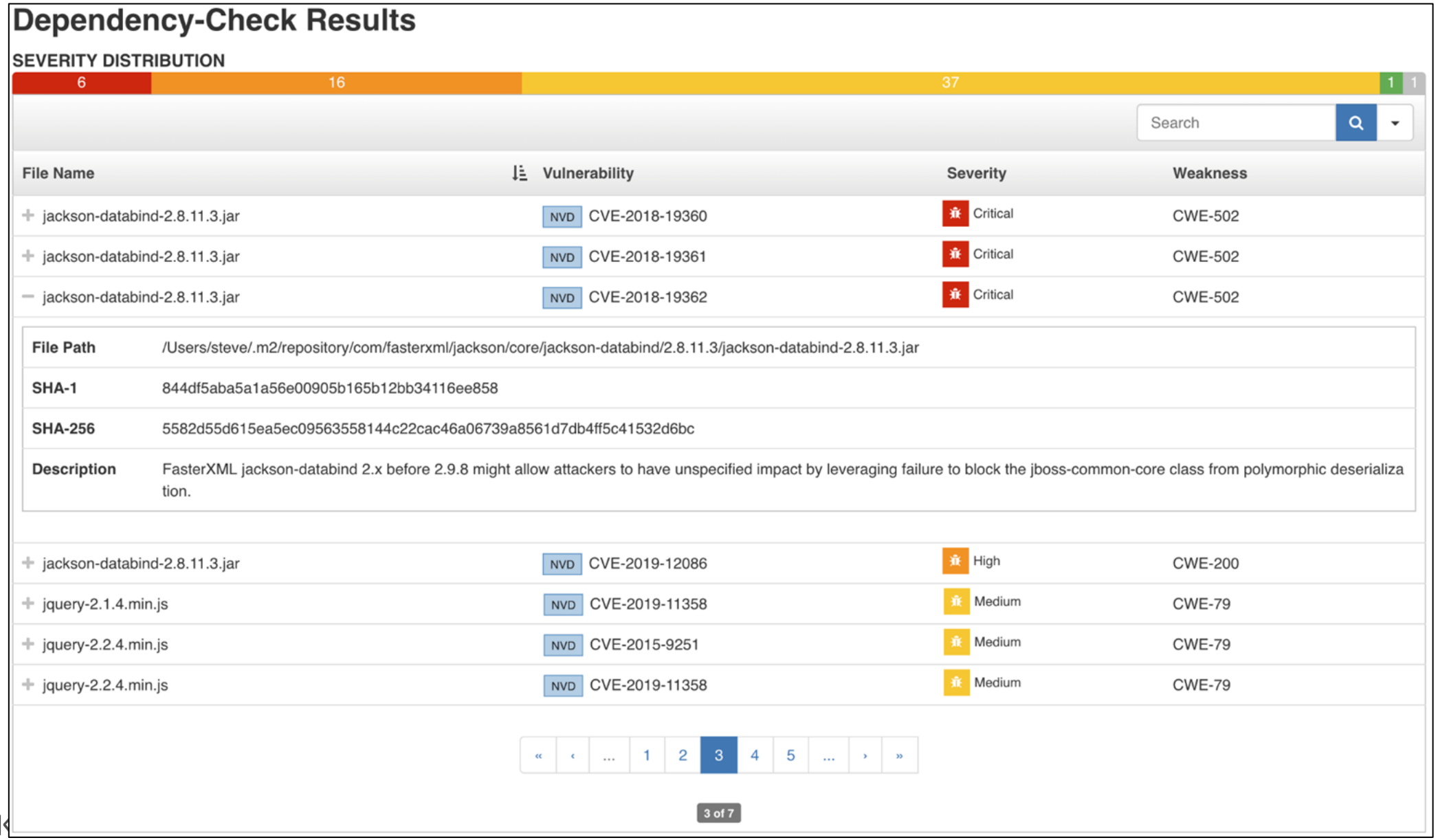
Problem?

- Our products include open-source dependencies, many of which have known vulnerabilities.
- In order to ensure that you are releasing secure products, you must have a solution in place that evaluates these dependencies and provides actionable information on fixes.

Solution?

- Dependency-Check gathers the information from the **pom.xml and packages.**
- The evidence gathered is used to identify the **Common Platform Enumeration** (CPE) of the dependency.
  - CPE is a standardized name given to software versions for universal identification.
- Once the CPE is identified, it is stored in a index and compared to a list of **Common Vulnerability and Exposure** (CVE) entries.
  - Dependency-Check uses the list of CVE entries present in the **National Vulnerability Database**(NVD).

Sample Dependency Check HTML Report



Dependency-Check Results

SEVERITY DISTRIBUTION

6 | 16 | 37 | 1 | 1

Search

File Name	Vulnerability	Severity	Weakness
+ jackson-databind-2.8.11.3.jar	NVD CVE-2018-19360	Critical	CWE-502
+ jackson-databind-2.8.11.3.jar	NVD CVE-2018-19361	Critical	CWE-502
− jackson-databind-2.8.11.3.jar	NVD CVE-2018-19362	Critical	CWE-502

File Path	/Users/steve/.m2/repository/com/fasterxml/jackson/core/jackson-databind/2.8.11.3/jackson-databind-2.8.11.3.jar
SHA-1	844df5aba5a1a56e00905b165b12bb34116ee858
SHA-256	5582d55d615ea5ec09563558144c22cac46a06739a8561d7db4ff5c41532d6bc
Description	FasterXML jackson-databind 2.x before 2.9.8 might allow attackers to have unspecified impact by leveraging failure to block the jboss-common-core class from polymorphic deserialization.

+ jackson-databind-2.8.11.3.jar	NVD CVE-2019-12086	High	CWE-200
+ jquery-2.1.4.min.js	NVD CVE-2019-11358	Medium	CWE-79
+ jquery-2.2.4.min.js	NVD CVE-2015-9251	Medium	CWE-79
+ jquery-2.2.4.min.js	NVD CVE-2019-11358	Medium	CWE-79

« ‹ ... 1 2 3 4 5 ... › »

3 of 7

Trivy

Trivy is a simple and comprehensive vulnerability scanner for containers and other artifacts.

- Detect comprehensive vulnerabilities
  - **OS packages**
    - Alpine,
    - Red Hat Universal Base Image,
    - Red Hat Enterprise Linux,
    - CentOS,
    - Oracle Linux,
    - Debian,
    - Ubuntu,
    - Amazon Linux,
    - openSUSE Leap,
    - SUSE Enterprise Linux,
    - Photon OS and
    - Distroless
  - **Application dependencies**
    - Bundler,
    - Composer,
    - Pipenv,
    - Poetry,
    - npm,
    - yarn,
    - Cargo,
    - NuGet,
    - **Maven**, and
    - Go

```
trivy image nginx:alpine
```

```
trivy image --severity HIGH,CRITICAL ruby:2.4.0
```

```
trivy image --vuln-type os nodejs-image:1.2
```

```
trivy image --skip-update python:3.4-alpine3.9
```

```
docker run --rm -v $WORKSPACE:/root/.cache/ aquasec/trivy:0.17.2 --exit-code 0 --severity HIGH nginx
```

```
docker run --rm -v $WORKSPACE:/root/.cache/ aquasec/trivy:0.17.2 --exit-code 1 --severity CRITICAL nginx
```

```
--format value      format (table, json, template) (default: "table")
--exit-code value   Exit code when vulnerabilities were found (default: 0)
--list-all-pkgs     enabling the option will output all packages regardless of vulnerability
```

KodeKloud

Sample Trivy Scan Table Report

```
Total: 4 (UNKNOWN: 0, LOW: 0, MEDIUM: 3, HIGH: 1, CRITICAL: 0)

+---------+------------------+----------+-------------------+--------------+--------------------------------+
| LIBRARY | VULNERABILITY ID | SEVERITY | INSTALLED VERSION | FIXED VERSION |             TITLE              |
+---------+------------------+----------+-------------------+--------------+--------------------------------+
| jquery  | CVE-2019-5428    | MEDIUM   | 3.3.9             | >=3.4.0      | Modification of                |
|         |                  |          |                   |              | Assumed-Immutable Data (MAID)  |
+         +------------------+          +                   +              +--------------------------------+
|         | CVE-2019-11358   |          |                   |              | js-jquery: prototype pollution |
|         |                  |          |                   |              | in object's prototype leading  |
|         |                  |          |                   |              | to denial of service or...     |
+---------+------------------+----------+-------------------+--------------+--------------------------------+
| lodash  | CVE-2018-16487   | HIGH     | 4.17.4            | >=4.17.11    | lodash: Prototype pollution in |
|         |                  |          |                   |              | utilities function             |
+         +------------------+----------+                   +--------------+                                +
|         | CVE-2018-3721    | MEDIUM   |                   | >=4.17.5     |                                |
|         |                  |          |                   |              |                                |
+---------+------------------+----------+-------------------+--------------+--------------------------------+

python-app/Pipfile.lock
=======================
Total: 1 (UNKNOWN: 0, LOW: 0, MEDIUM: 1, HIGH: 0, CRITICAL: 0)

+---------+------------------+----------+-------------------+--------------+---------------------------------+
| LIBRARY | VULNERABILITY ID | SEVERITY | INSTALLED VERSION | FIXED VERSION |              TITLE              |
+---------+------------------+----------+-------------------+--------------+---------------------------------+
| django  | CVE-2019-6975    | MEDIUM   | 2.0.9             | 2.0.11       | python-django:                  |
|         |                  |          |                   |              | memory exhaustion in            |
|         |                  |          |                   |              | django.utils.numberformat.format() |
+---------+------------------+----------+-------------------+--------------+---------------------------------+

ruby-app/Gemfile.lock
=====================
Total: 1 (UNKNOWN: 0, LOW: 0, MEDIUM: 1, HIGH: 0, CRITICAL: 0)

+---------------------+------------------+----------+-------------------+--------------+--------------------------------+
|       LIBRARY       | VULNERABILITY ID | SEVERITY | INSTALLED VERSION | FIXED VERSION |             TITLE              |
+---------------------+------------------+----------+-------------------+--------------+--------------------------------+
| rails-html-sanitizer | CVE-2018-3741    | MEDIUM   | 1.0.3             | >= 1.0.4     | rubygem-rails-html-sanitizer:  |
|                     |                  |          |                   |              | non-whitelisted attributes     |
|                     |                  |          |                   |              | are present in sanitized       |
|                     |                  |          |                   |              | output when input with         |
|                     |                  |          |                   |              | specially-crafted...           |
+---------------------+------------------+----------+-------------------+--------------+--------------------------------+
```

KodeKloud

OPA Conftest

The Open Policy Agent (OPA, pronounced "oh-pa") is an open source, general-purpose policy engine that unifies policy enforcement across the stack.

Open Policy Agent

- Conftest is a utility to help you write tests **(statically analyze)** against structured configuration data.
- Using Conftest you can write tests for,
  - Kubernetes Configurations
  - Terraform code,
  - Dockerfile or any other config files.
- Conftest uses the **Rego** language from Open Policy Agent for writing the assertions.

Dockerfile

```
1  FROM openjdk:latest
2  EXPOSE 8080
3  ARG JAR_FILE=target/*.jar
4  ADD ${JAR_FILE} app.jar
5  ENTRYPOINT ["java","-jar","/app.jar"]
```

docker.reg

```
1  package main
2
3  deny[msg] {
4      input[i].Cmd == "from"
5      val := split(input[i].Value[0], ":")
6      contains(lower(val[1]), "latest")
7      msg = sprintf("Line %d: do not use 'latest' tag for base images", [i])
8  }
```

docker run openpolicyagent/conftest test --policy docker.rego Dockerfile

```
FAIL - Dockerfile - main - Line 0: do not use 'latest' tag for base images

1 tests, 0 passed, 0 warnings, 1 failures, 0 exceptions
```

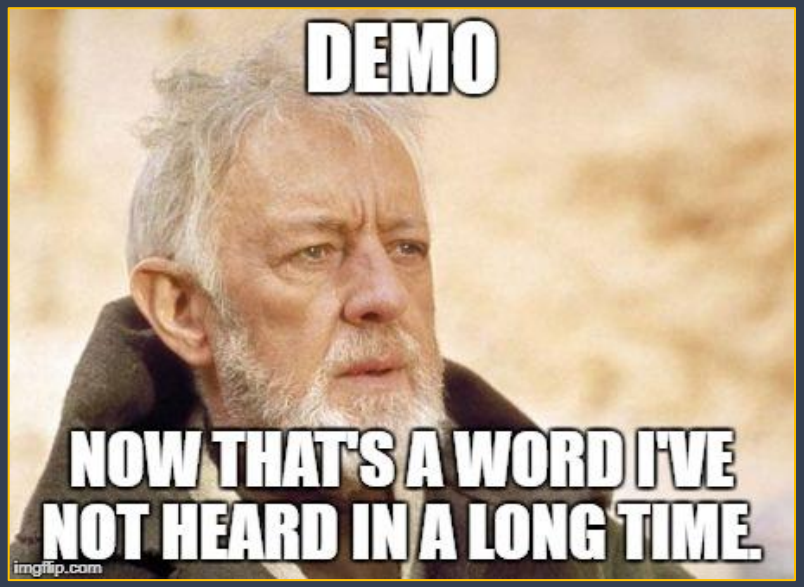
KodeKloud

KodeKloud

HANDS ON

- Jenkins
  - Dependency Checks
  - Trivy
  - OPA Conftest

DEMO



KodeKloud

Kubernetes Security Concepts

A security context defines privilege and access control settings for a Pod or Container

runAsUser	When you set **runAsUser: 100** you require that the container will run with a user with UID 100
runAsNonRoot	When you set **runAsNonRoot: true** you require that the container will run with a user with any UID other than 0
readOnlyRootFilesystem	Mounts the container's root filesystem **as read-only**. Can make use of emptyDir Volume to allow certain Dir.

```
securityContext:
  runAsUser: 100
  readOnlyRootFilesystem: true
  runAsNonRoot: true
```

AppArmor	**SELinux**	**Sandboxing**
Privilege Escalations	**Auditing**	**Capabilities**
Network Policies	**RBAC**	**TLS**

KodeKloud

OPA Conftest - Kubernetes

The Open Policy Agent (OPA, pronounced "oh-pa") is an open source, general-purpose policy engine that unifies policy enforcement across the stack.

Open Policy Agent

- Conftest is a utility to help you write tests **(statically analyze)** against structured configuration data.
- Using Conftest you can write tests for,
  - Kubernetes Configurations
  - Terraform code,
  - Dockerfile or any other config files.
- Conftest uses the **Rego** language from Open Policy Agent for writing the assertions.

pod.yam

```yaml
apiVersion: v1
kind: Pod
metadata:
  labels:
    run: nginx-pod
    app: frontend
  name: nginx-pod
spec:
  containers:
  - image: nginx
    name: nginx-pod
```

pod.rego

```
package main

deny[msg] {
  input.kind == "Pod"
  not input.metadata.labels.devsecops

  msg := "Must add a devsecops label"
}
```

```
docker run openpolicyagent/conftest test --policy pod.rego
```

```
FAIL - pod.yaml - Must add a devsecops label

1 tests, 0 passed, 0 warnings, 1 failures, 0 exceptions
```

KodeKloud

Trivy

Trivy is a simple and comprehensive vulnerability scanner for containers and other artifacts.

- Detect comprehensive vulnerabilities
  - **OS packages**
    - Alpine,
    - Red Hat Universal Base Image,
    - Red Hat Enterprise Linux,
    - CentOS,
    - Oracle Linux,
    - Debian,
    - Ubuntu,
    - Amazon Linux,
    - openSUSE Leap,
    - SUSE Enterprise Linux,
    - Photon OS and
    - Distroless
  - **Application dependencies**
    - Bundler,
    - Composer,
    - Pipenv,
    - Poetry,
    - npm,
    - yarn,
    - Cargo,
    - NuGet,
    - **Maven**, and
    - Go

```
trivy image nginx:alpine
```

```
trivy image --severity HIGH,CRITICAL ruby:2.4.0
```

```
trivy image --vuln-type os nodejs-image:1.2
```

```
trivy image --skip-update python:3.4-alpine3.9
```

```
docker run --rm -v $WORKSPACE:/root/.cache/ aquasec/trivy:0.17.2 --exit-code 0 --severity HIGH nginx
```

```
docker run --rm -v $WORKSPACE:/root/.cache/ aquasec/trivy:0.17.2 --exit-code 1 --severity CRITICAL nginx
```

```
--format value      format (table, json, template) (default: "table")
--exit-code value   Exit code when vulnerabilities were found (default: 0)
--list-all-pkgs     enabling the option will output all packages regardless of vulnerability
```

KodeKloud

Kubesec

Kubesec is an open-source Kubernetes security scanner and analysis tool. It scans your Kubernetes cluster for common exploitable risks such as privileged capabilities and provides a severity score for each found vulnerability.

- Security risk analysis for Kubernetes resources.
- Take in a single YAML file as input.
  - One YAML can connect multiple Kubernetes resources.
- Kubesec is available as,
  - Docker container image at *docker.io/kubesec/kubesec:v2*
  - Linux/MacOS/Win binary (get the latest release)
  - Kubernetes Admission Controller
  - Kubectl plugin

pod.yam

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: kubesec-demo
spec:
  containers:
  - name: kubesec-demo
    image: gcr.io/google-samples/node-hello:1.0
    securityContext:
      readOnlyRootFilesystem: true
```

```
docker run -i kubesec/kubesec:512c5e0 scan /dev/stdin < pod.yaml
curl -sSX POST --data-binary @"pod.yaml" https://v2.kubesec.io/scan
```

```
{
  "object": "Pod/kubesec-demo.default",
  "valid": true,
  "fileName": "API",
  "message": "Passed with a score of 1 points",
  "score": 1,
  "scoring": {
    "passed": [
      {
        "id": "ReadOnlyRootFilesystem",
        "selector": "containers[] .securityContext .readOnlyRootFilesy
        "reason": "An immutable root filesystem can prevent malicious
        "points": 1
      }
    ],
    "advise": [

      {
        "id": "ServiceAccountName",
        "selector": ".spec .serviceAccountName",
        "reason": "Service accounts restrict Kubernetes API access and
        "points": 3
      },
```

KodeKloud

KodeKloud

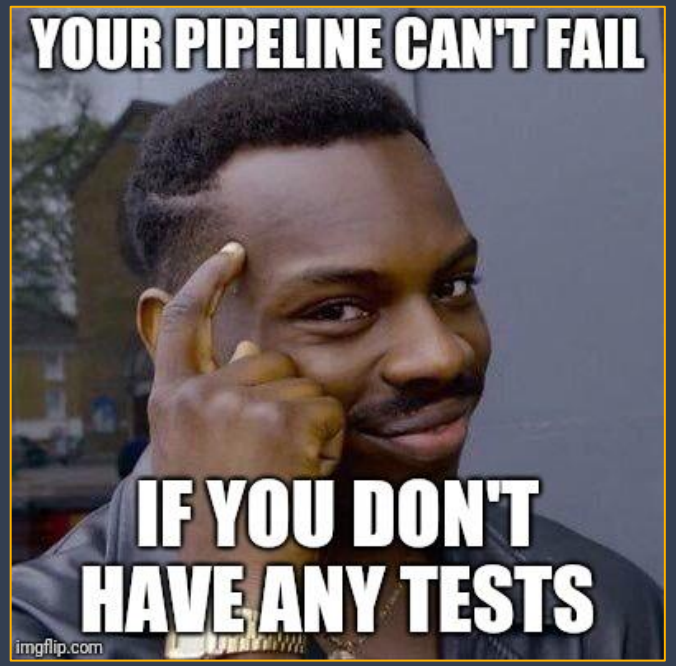
Vulnerabilities

- Kubernetes Vulnerabilities
  - OPA Conftest (k8s deployment yaml)
  - Kubesec
  - **Trivy (scan app image)**

HANDS ON

- What?
- Jenkins
  - Integration Test

DEMO

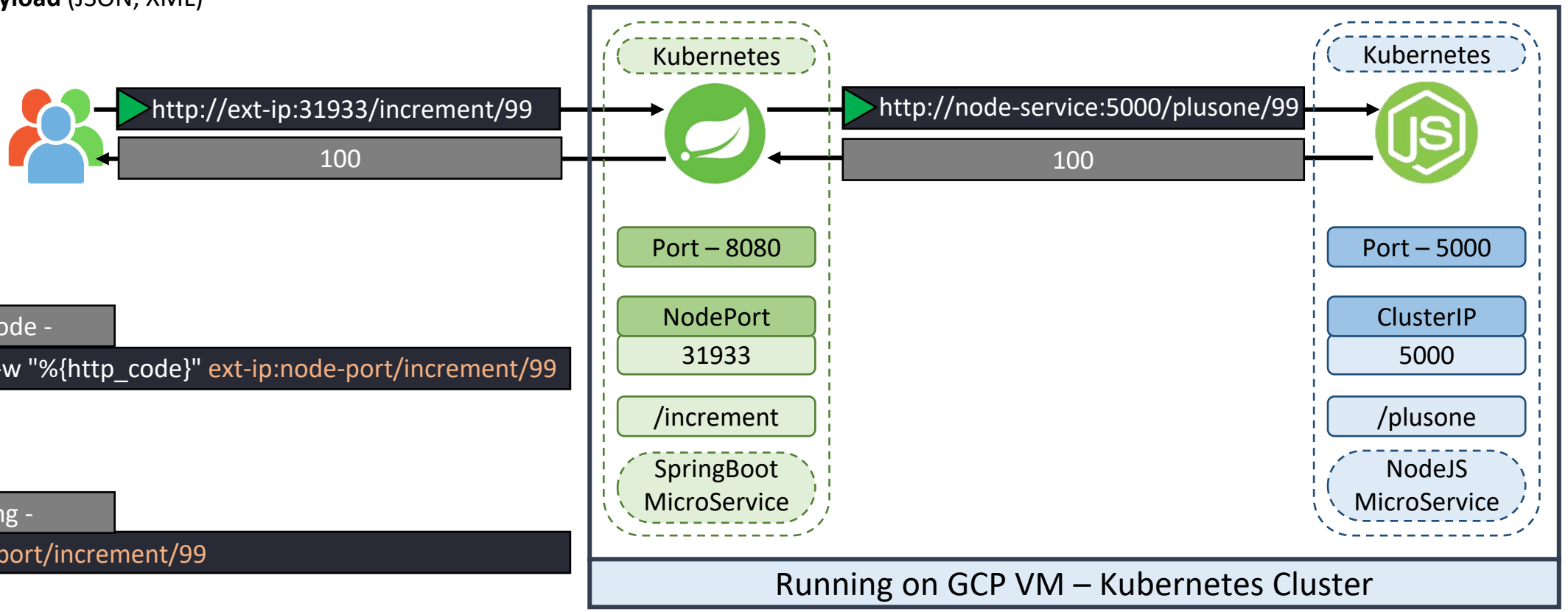


KodeKloud

K8S

Integration Tests

- Integration tests determine if independently developed units of software work correctly when they are connected to each other.
- The main goal in our application is to test the basic correctness of the API by consuming the REST API after it has already been deployed.
- When testing a REST resource, there are usually a few orthogonal responsibilities the tests should focus on:
  - the HTTP **response code**
  - other HTTP **headers** in the response
  - the **payload** (JSON, XML)



Testing HTTP Status Code -

```
curl -s -o /dev/null -w "%{http_code}" ext-ip:node-port/increment/99
```

200

Testing Response String -

```
curl -s ext-ip:node-port/increment/99
```

100

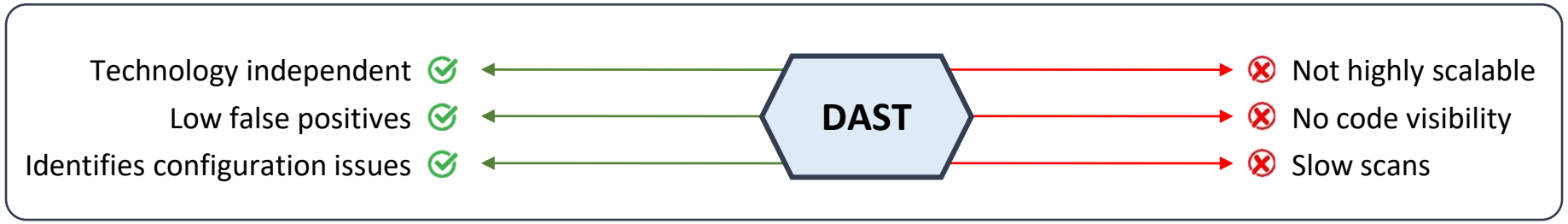
KodeKloud

DAST

DAST looks for security vulnerabilities by simulating external attacks on an application while the application is running.

- DAST test is performed in a dynamic environment.
- **SAST** scans an application's code line by line when the application is at rest.
- DAST has **no access to an application's source code**, it detects security vulnerabilities by attacking the application externally.

- DAST can be used in production, testing usually is carried out in a QA environment.
- DAST is extremely good at finding externally visible issues and vulnerabilities. This includes a number of security risks from OWASP's top ten, such as
  - cross-site scripting,
  - injection errors like SQL injection or command injection,
  - path traversal, and
  - insecure server configuration



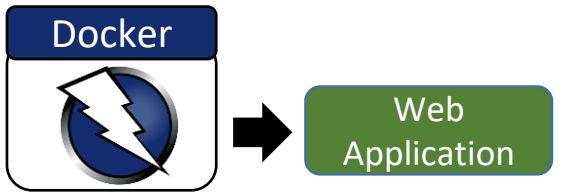
KodeKloud

DAST

- Dynamic Application Security Testing
- **OWASP ZAP**

OWASP ZAP

- OWASP ZAP (Zed Attack Proxy) is an open-source web application security scanner. ZAP is designed specifically for testing web applications.
- ZAP is what is known as a “man-in-the-middle proxy.



Docker	owasp/zap2docker-stable	**stable** release
	owasp/zap2docker-weekly	**weekly** release
	owasp/zap2docker-live	**live** release *(built whenever the zaproxy project is changed)*
Scans Sc	**Baseline** Scan	a time limited spider which reports issues found **passively**
	Full Scan	a full spider, optional ajax spider and active scan which reports issues found **actively** and passively
Scan	**Passive** Scan	**scans** all HTTP messages (requests and responses) sent to the web application being tested
	Active Scan	attempts to find potential vulnerabilities by using known attacks against the selected targets

ZAP API Scan

- It is tuned for performing scans against APIs defined by **OpenAPI**, SOAP, or GraphQL via either a local file or a URL.
- It imports the definition that you specify and then runs **an Active Scan** against the URLs found.
- The **Active Scan** is tuned to **APIs**, so it doesn't bother looking for things like XSSs (Cross Site Scripting).
- It also includes 2 scripts that:
  - Raise alerts for any **HTTP Server Error** response codes
  - Raise alerts for any URLs that return **content types** that are not usually associated with APIs

```
docker run -v $(pwd):/zap/wrk/:rw -t owasp/zap2docker-weekly zap-api-scan.py -t http://$hostURL:$PORT/v3/api-docs -f openapi -c rules -r report.html
```

```
PASS: Hash Disclosure [10097]
PASS: Cross-Domain Misconfiguration [10098]
PASS: Weak Authentication Method [10105]
PASS: Reverse Tabnabbing [10108]
PASS: Modern Web Application [10109]
PASS: Private IP Disclosure [2]
PASS: Session ID in URL Rewrite [3]
PASS: Script Passive Scan Rules [50001]
PASS: Insecure JSF ViewState [90001]
PASS: Charset Mismatch [90011]
PASS: Loosely Scoped Cookie [90033]
WARN-NEW: Application Error Disclosure [90022] x 2
  http://192.168.18.23:5000/update (500 INTERNAL SERVER ERROR)
  http://192.168.18.23:5000/?__debugger__=yes&cmd=paste (500 INTERNAL SERVER ERROR)

FAIL-NEW: 0 FAIL-INPROG: 0  WARN-NEW: 6 WARN-INPROG: 0  INFO: 0 IGNORE: 0 PASS: 45
```

Arguments used in docker cmd

-v volume	docker volume mount reports & configs
-t target	target URL including protocol
-f format	openapi, soap, or graphql
-c config_file	config file to use override warnings
-r report_html	full ZAP HTML report
-w report_md	full ZAP Wiki (Markdown) report
-x report_xml	full ZAP XML report
-J report_json	full ZAP JSON document

KodeKloud

Slack

- Basic Slack Notifications

KodeKloud

HANDS ON

- Slack
- Jenkins Slack Notifications

DEMO

KodeKloud

Slack

***Slack** is a messaging app for business that connects people to the information they need.*

- At certain stages of Jenkins pipeline, you may want to send out a **Slack** notification to you and your team to notify the build status.

Tools

SpringBoot	Dependency Check	SonarQube	OSS Review Toolkit
Python	Bandit	Safety	Sonatype
NodeJS	NodeJS Scan	npm-check	Clearly Defined
PHP	Sonatype	SonarQube	OSS Review Toolkit
.NET	DotNET Retire SafeNuGet	Puma Scan	DotNet Security Guard
Ruby/Rails	Brakeman	SonarQube	OSS Review Toolkit

Section #3

#3 DevSecOps Pipeline

- Adding security to pipeline
  - Git Hooks - Talisman
  - Testing
    - Mutation - PIT
    - Integration
  - Scans
    - Dependency Checks
    - Trivy - Images
    - OPA - Dockerfile
    - Kubesec - K8S
  - SAST - SonarQube
  - DAST - OWASP ZAP
- Deploy to K8S Dev Namespace
  - Deployment
  - Rollout Status
- Basic Slack Notifications
  - Legacy App Notifications

#1 Introduction

- Introducing DevSecOps
- Security Aspects
- Various Tools

#2 Simple DevOps Pipeline

- Setting up the VM
- Installing Software
- Understanding the Usecase
- Basic DevOps Pipeline (4 stages)

#4 Kubernetes Security

- Istio Service Mesh
- Deploy to Prod Environment
- Monitoring - Falco
- Advanced Slack Notifications

KodeKloud

Section #4

#4 Kubernetes Security

- Deploy to Prod Environment
  - CIS Benchmarking
  - Testing
  - Limits
- Istio Service Mesh
  - Installation
  - Kiali
  - mTLS
  - Metrics
- Monitoring
  - Kube-scan
  - Falco
  - Falco UI
  - Falco Slack
- HELM
- Advanced Slack Notifications
  - Rich Message Layout
  - Emojis

#1 Introduction

- Introducing DevSecOps
- Security Aspects
- Various Tools

#2 Simple DevOps Pipeline

- Setting up the VM
- Installing Software
- Understanding the Usecase
- Basic DevOps Pipeline (4 stages)

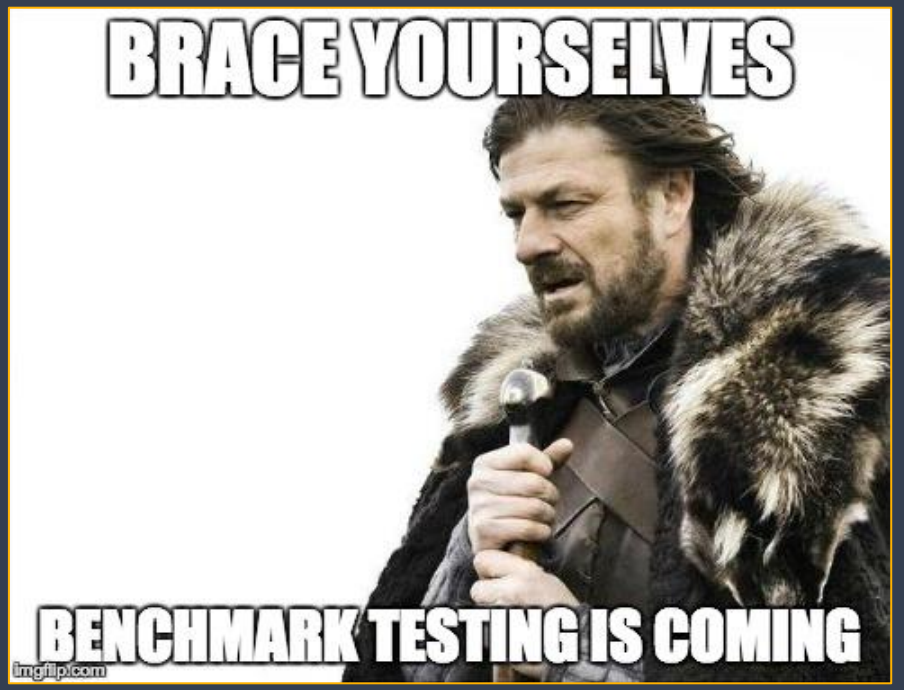
#3 DevSecOps Pipeline

- Adding security to pipeline
- Deploy to K8S Dev Namespace
- Basic Slack Notifications

KodeKloud

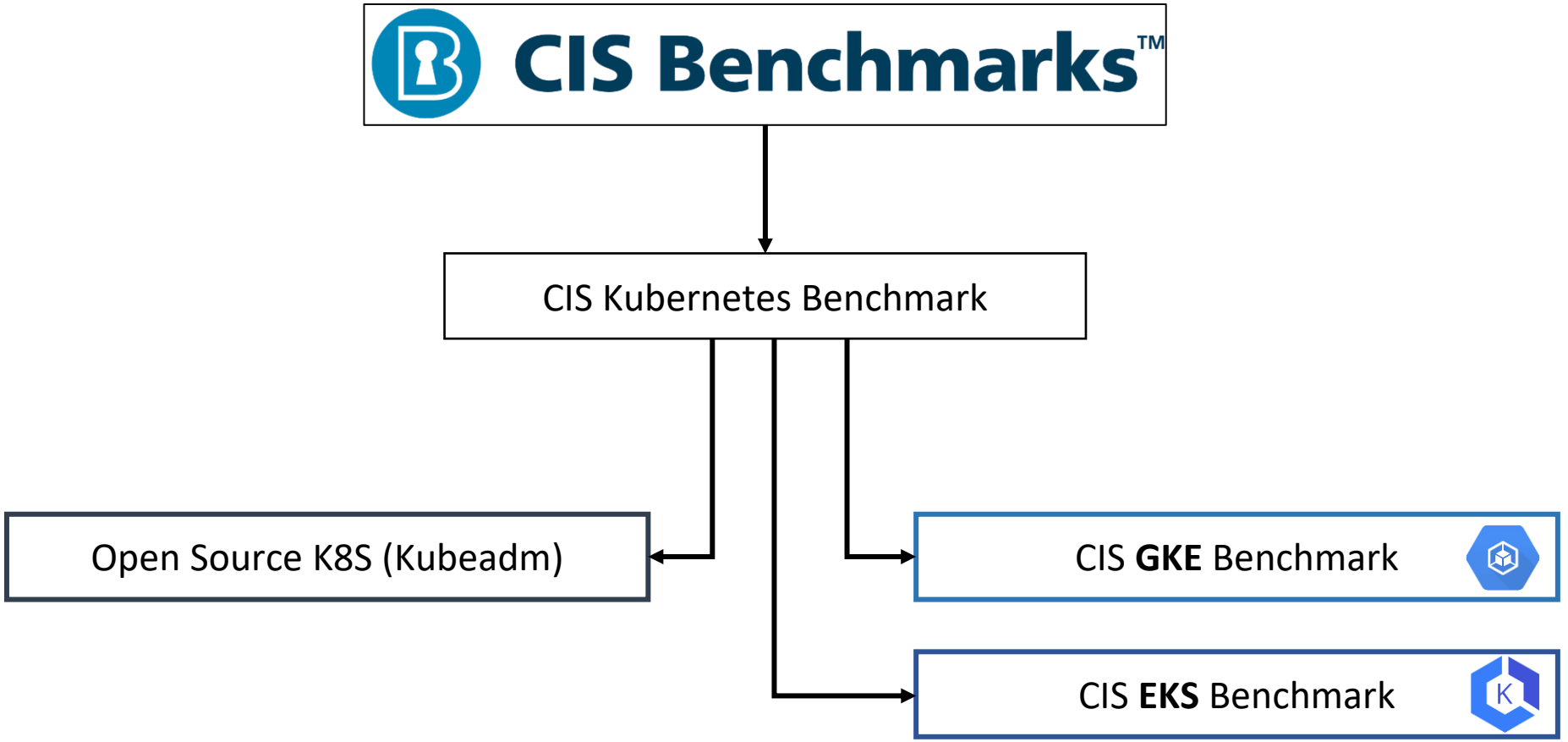
Benchmark Test

- CIS Benchmark Testing
- Kube-bench



CIS Benchmarks

- Configuration guidelines for various technology groups to safeguard systems against today's evolving cyber threats
- The **Center for Internet Security** (CIS) releases benchmarks for best practice security recommendations.



kube-bench

kube-bench *is a Go application that checks whether Kubernetes is deployed securely by running the checks documented in the CIS Kubernetes Benchmark.*

```
docker run --pid=host -v /etc:/etc:ro -v /var:/var:ro -t aquasec/kube-bench:latest master --version 1.19
```

```
[INFO] 1 Master Node Security Configuration
[INFO] 1.1 API Server
[FAIL] 1.1.1 Ensure that the --allow-privileged argument is set to false (Scored)
[FAIL] 1.1.2 Ensure that the --anonymous-auth argument is set to false (Scored)
[PASS] 1.1.3 Ensure that the --basic-auth-file argument is not set (Scored)
[PASS] 1.1.4 Ensure that the --insecure-allow-any-token argument is not set (Scored)
[FAIL] 1.1.5 Ensure that the --kubelet-https argument is set to true (Scored)
[PASS] 1.1.6 Ensure that the --insecure-bind-address argument is not set (Scored)
[PASS] 1.1.7 Ensure that the --insecure-port argument is set to 0 (Scored)
[PASS] 1.1.8 Ensure that the --secure-port argument is not set to 0 (Scored)
[FAIL] 1.1.9 Ensure that the --profiling argument is set to false (Scored)
[FAIL] 1.1.10 Ensure that the --repair-malformed-updates argument is set to false (Scored)
[PASS] 1.1.11 Ensure that the admission control policy is not set to AlwaysAdmit (Scored)
[FAIL] 1.1.12 Ensure that the admission control policy is set to AlwaysPullImages (Scored)
[FAIL] 1.1.13 Ensure that the admission control policy is set to DenyEscalatingExec (Scored)
[FAIL] 1.1.14 Ensure that the admission control policy is set to SecurityContextDeny (Scored)
[PASS] 1.1.15 Ensure that the admission control policy is set to NamespaceLifecycle (Scored)
[FAIL] 1.1.16 Ensure that the --audit-log-path argument is set as appropriate (Scored)
[FAIL] 1.1.17 Ensure that the --audit-log-maxage argument is set to 30 or as appropriate (Scored)
[FAIL] 1.1.18 Ensure that the --audit-log-maxbackup argument is set to 10 or as appropriate (Scored)
[FAIL] 1.1.19 Ensure that the --audit-log-maxsize argument is set to 100 or as appropriate (Scored)
[PASS] 1.1.20 Ensure that the --authorization-mode argument is not set to AlwaysAllow (Scored)
[PASS] 1.1.21 Ensure that the --token-auth-file parameter is not set (Scored)
[FAIL] 1.1.22 Ensure that the --kubelet-certificate-authority argument is set as appropriate (Scored)
```

```
kube-bench master  --version 1.15 --check 1.2.7,1.2.8,1.2.9 --json
```

KodeKloud

Service Mesh

- Istio
  - What?
  - Why?
  - How?
- Installation
- mTLS
- Kiali (metrics)

KodeKloud

Istio

Istio is an open framework for connecting, securing, managing and monitoring microservices

Istio include,

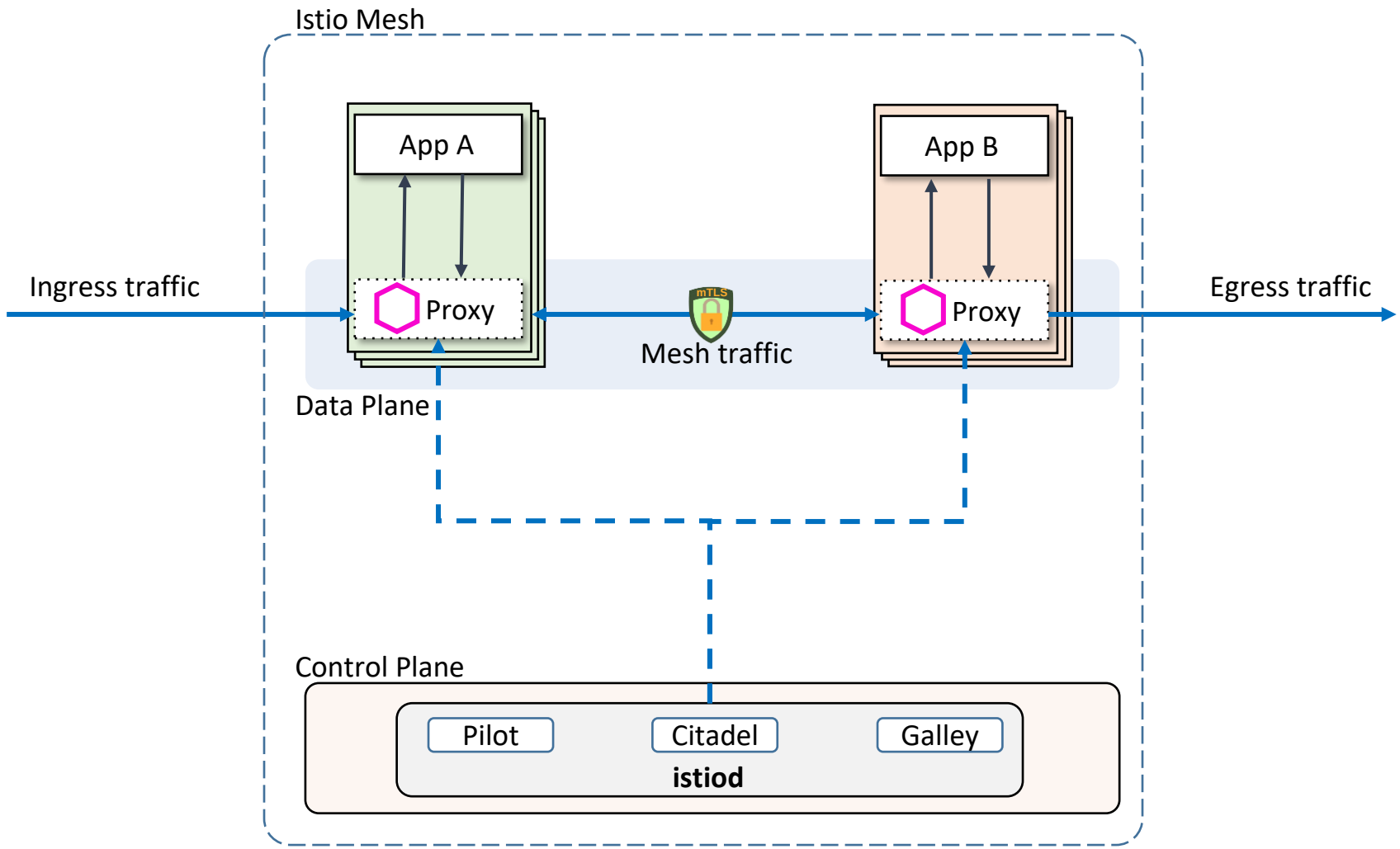
1. **discovery,**
2. **load balancing,**
3. **failure recovery,**
4. **metrics, and monitoring**

Istio can also handle more complex operational requirements,

1. **A/B testing,**
2. **canary releases,**
3. **rate limiting,**
4. **dark launches,**
5. **access control, and**
6. **end-to-end authentication**

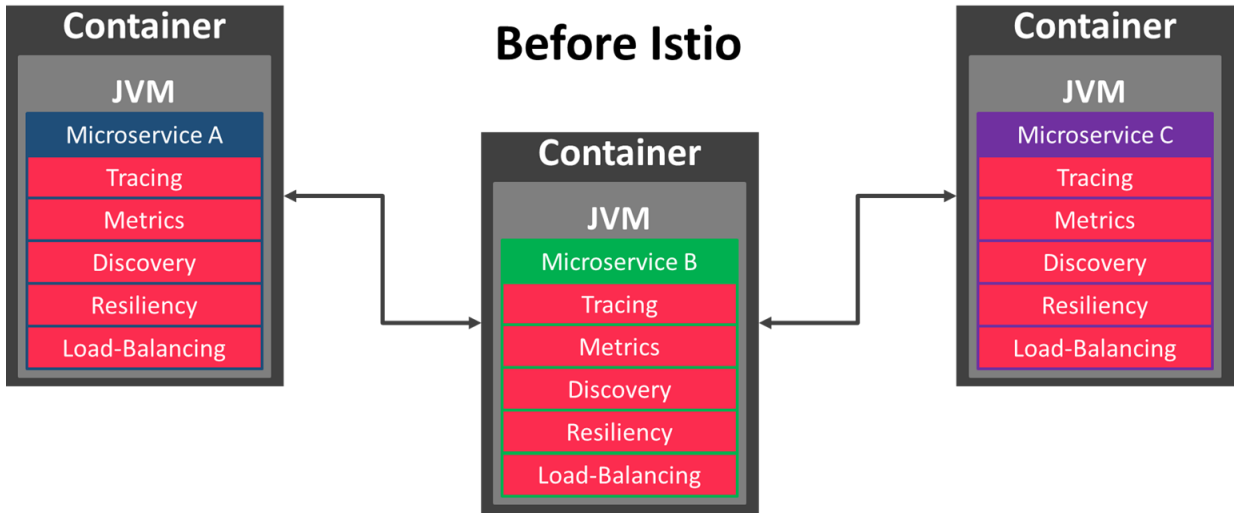
What makes Istio so unique is that all these functionalities come with NO CHANGE OF CODE required.

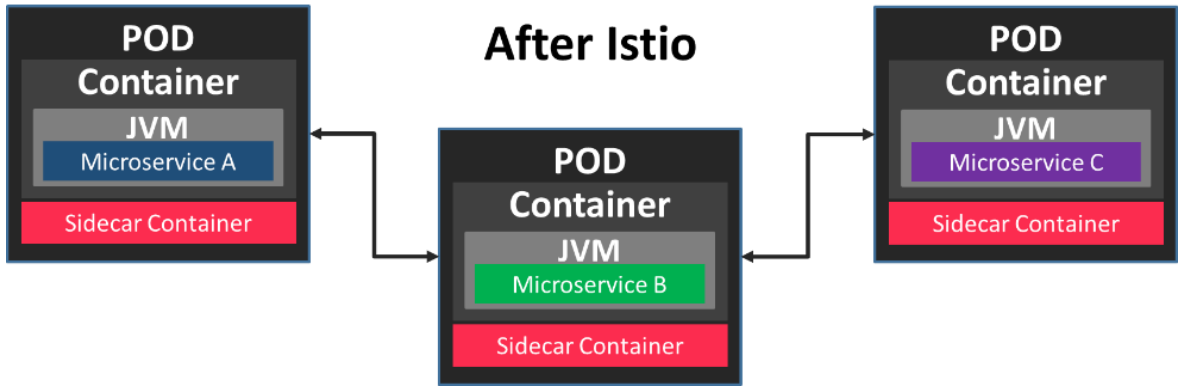
Istio Architecture



KodeKloud

Microservices externalizing Capabilities





Netflix OSS

Config Server

Sidecar Container

KodeKloud

HANDS ON

- Istio
  - Installation

DEMO

KodeKloud

Installation

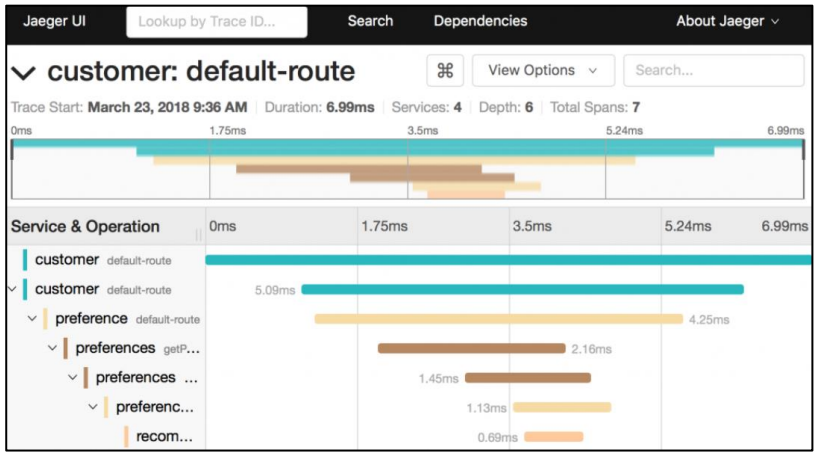
ISTIO Installation on Kubernetes

Before you can install **Istio**, you need a **cluster/server** running a compatible version of **Kubernetes**.

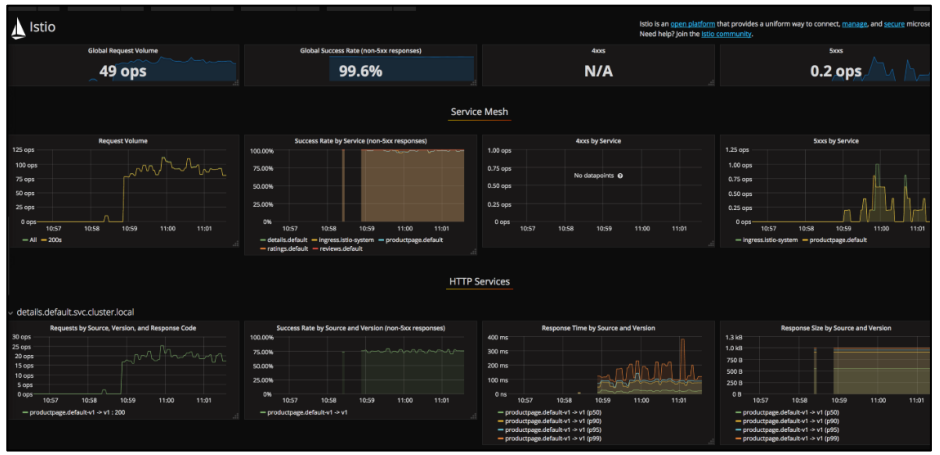
```
curl -Ls https://istio.io/downloadIstio | ISTIO_VERSION=1.9.0 sh -
cd istio-1.9.0
export PATH=$PWD/bin:$PATH
istioctl install --set profile=demo -y && kubectl apply -f samples/addons
```

```
namespace "istio-system" created
configmap "istio-sidecar-injector" created
customresourcedefinition "virtualservices.networking.istio.io" configured
customresourcedefinition "destinationrules.networking.istio.io" configured
deployment "grafana" created
deployment “kiali" created
service “kiali" created
service “grafana" created
```

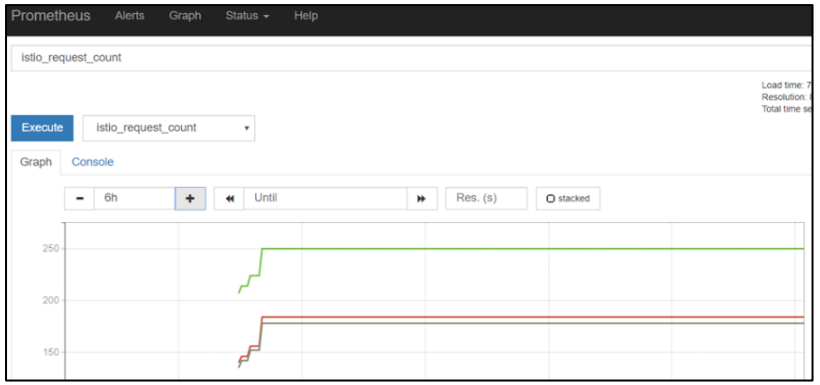
Jaeger, Grafana, Prometheus, Kiali Metrics



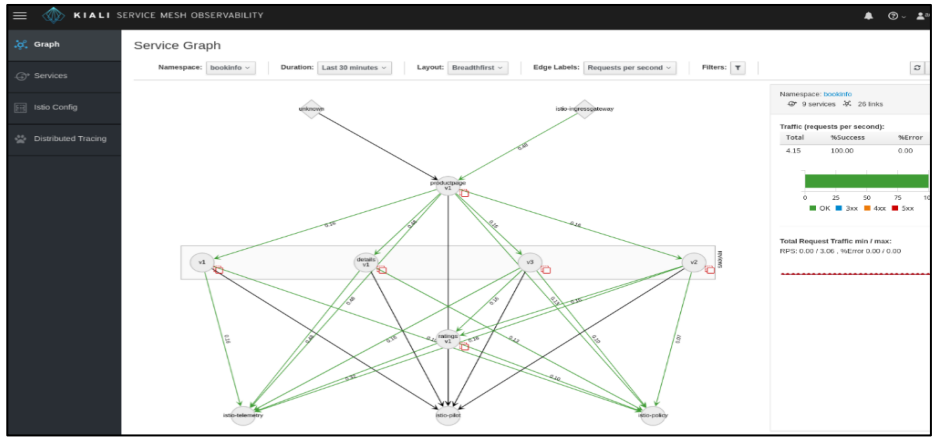
Jaeger, is used for monitoring and troubleshooting microservices-based distributed systems.



Grafana is most commonly used for visualizing time series data for infrastructure and application analytics.



Prometheus is a monitoring and alerting tool. It monitors multiple microservices running in your system.



Kiali works with Istio to visualise the service mesh topology, features like request rates.

KodeKloud

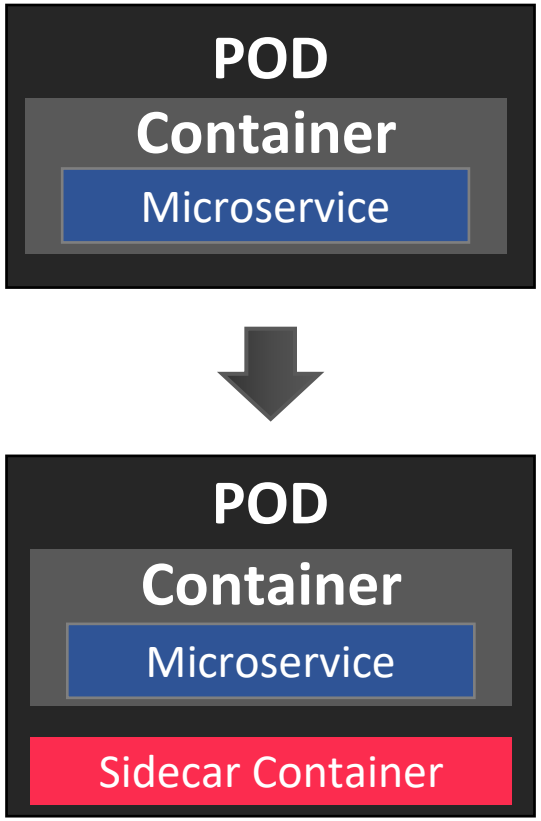
HANDS ON

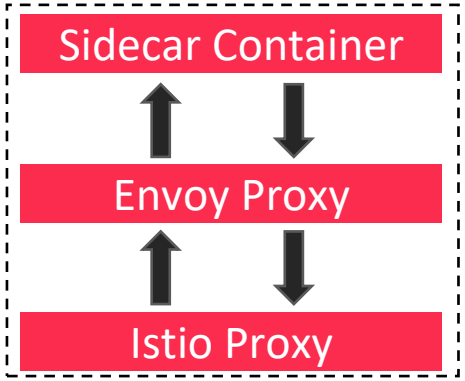
- Istio
  - Sidecar Injection

DEMO

KodeKloud

Sidecar/Envoy/Istio Proxy Injection





Sidecar Container can be INJECTED in 2 ways

Manual Injection

kubectl apply -f <(**istioctl kube-inject** -f movies.yaml)

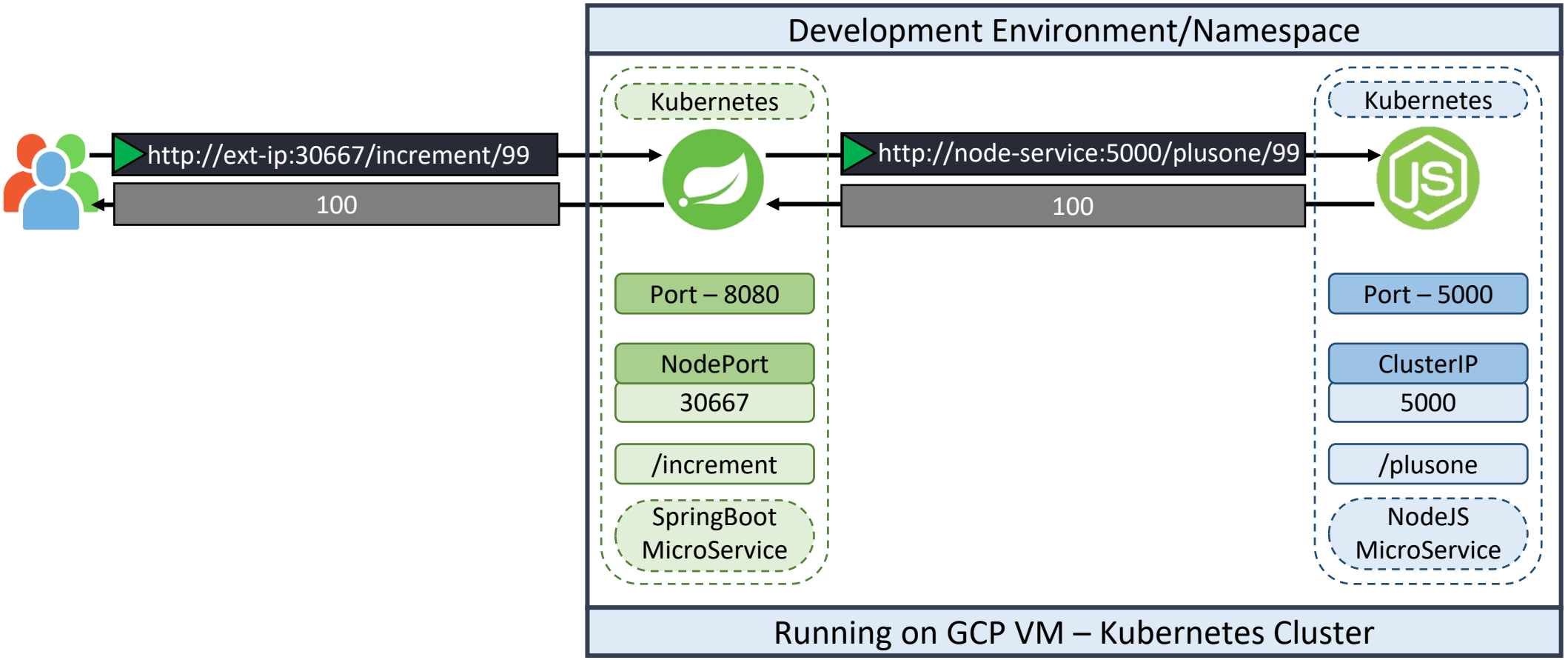
Automatic Injection

kubectl label namespace <name> **istio-injection=enabled**

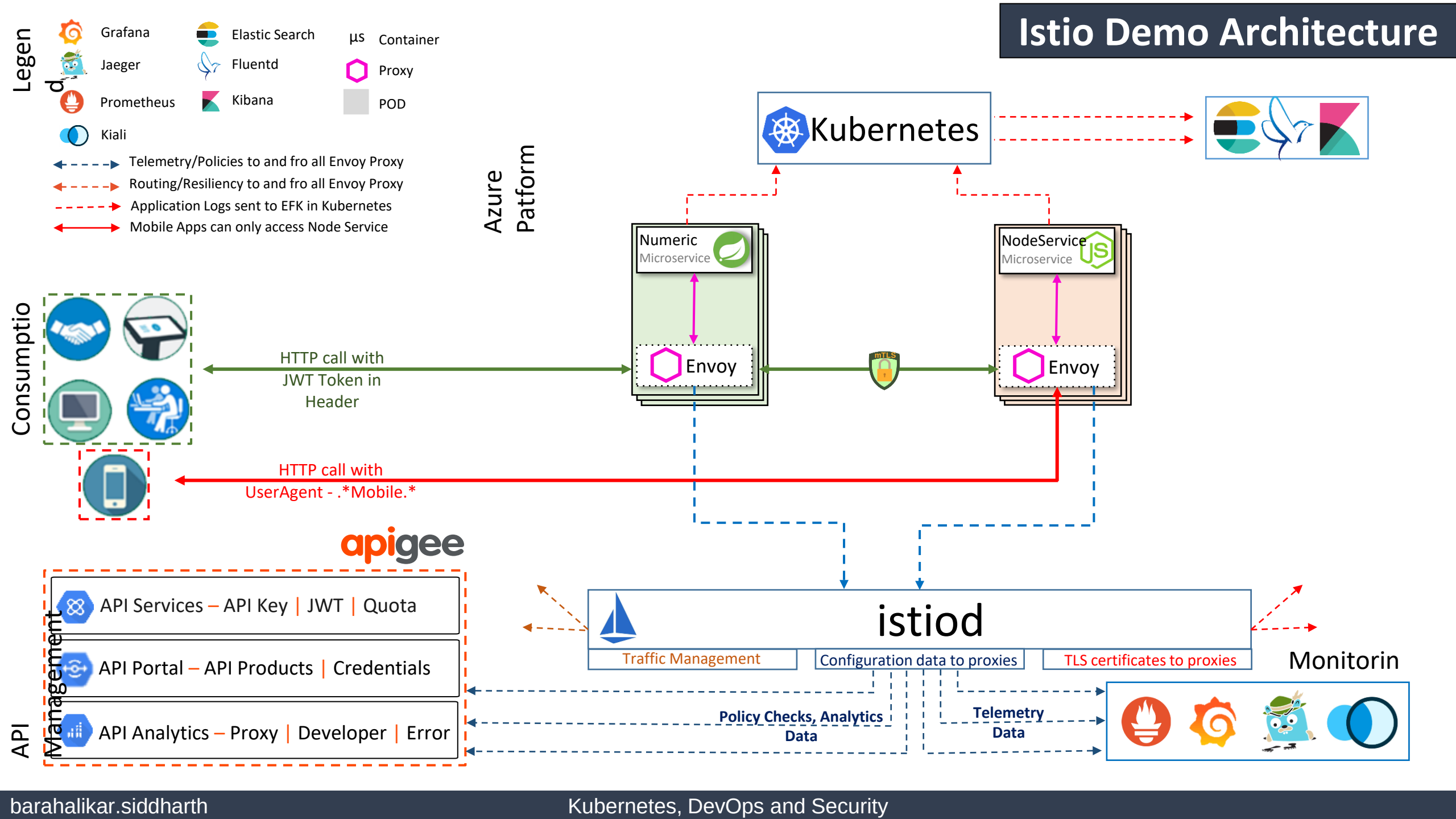
kubectl apply -f movies.yaml

No need to run ***istioctl*** or do anything with your **YAML**

Application Architecture



Istio Demo Architecture



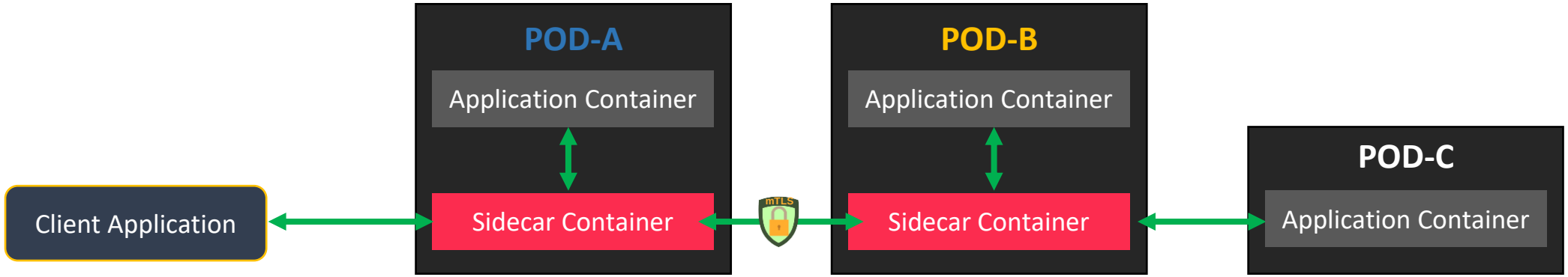
KodeKloud

Mutual Authentication

- Istio
  - mTLS

KodeKloud

Istio mTLS - Auto mutual TLS



PeerAuthentication defines how traffic will be tunneled (or not) to the sidecar.

mode = PERMISSIVE

```yaml
apiVersion: security.istio.io/v1beta1
kind: PeerAuthentication
metadata:
  name: default
  namespace: istio-system
spec:
  mtls:
    mode: PERMISSIVE
```

mode = STRICT

```yaml
apiVersion: security.istio.io/v1beta1
kind: PeerAuthentication
metadata:
  name: default
  namespace: foo
spec:
  mtls:
    mode: STRICT
```

mode = DISABLE

```yaml
apiVersion: security.istio.io/v1beta1
kind: PeerAuthentication
metadata:
  name: default
  namespace: foo
spec:
  mtls:
    mode: DISABLE
```

Mode	Traffic within Istio Mesh	Traffic from Outside Istio Mesh
PERMISSIVE	🔒	🔓
STRICT	🔒	⛔
DISABLE	🔓	🔓

🔒 Encrypted

🔓 Un-encrypted

KodeKloud

KodeKloud

Monitoring

- **Kube-scan**
- Falco
- Helm
- Slack

KodeKloud

Kubernetes Monitoring

- Kubernetes has the potential to simplify the process of deploying your application in containers – and across clouds, but in doing so, it leaves you blind as to,
  - what is actually happening,
  - what resources are being utilized

Monitoring Kubernetes clusters and nodes

- Cluster resource usage
- Node availability and health

Monitoring Kubernetes deployments, pods

- Missing and failed pods
- Running vs. desired instances
- Pod resource usage requests and limits

Monitoring Kubernetes applications

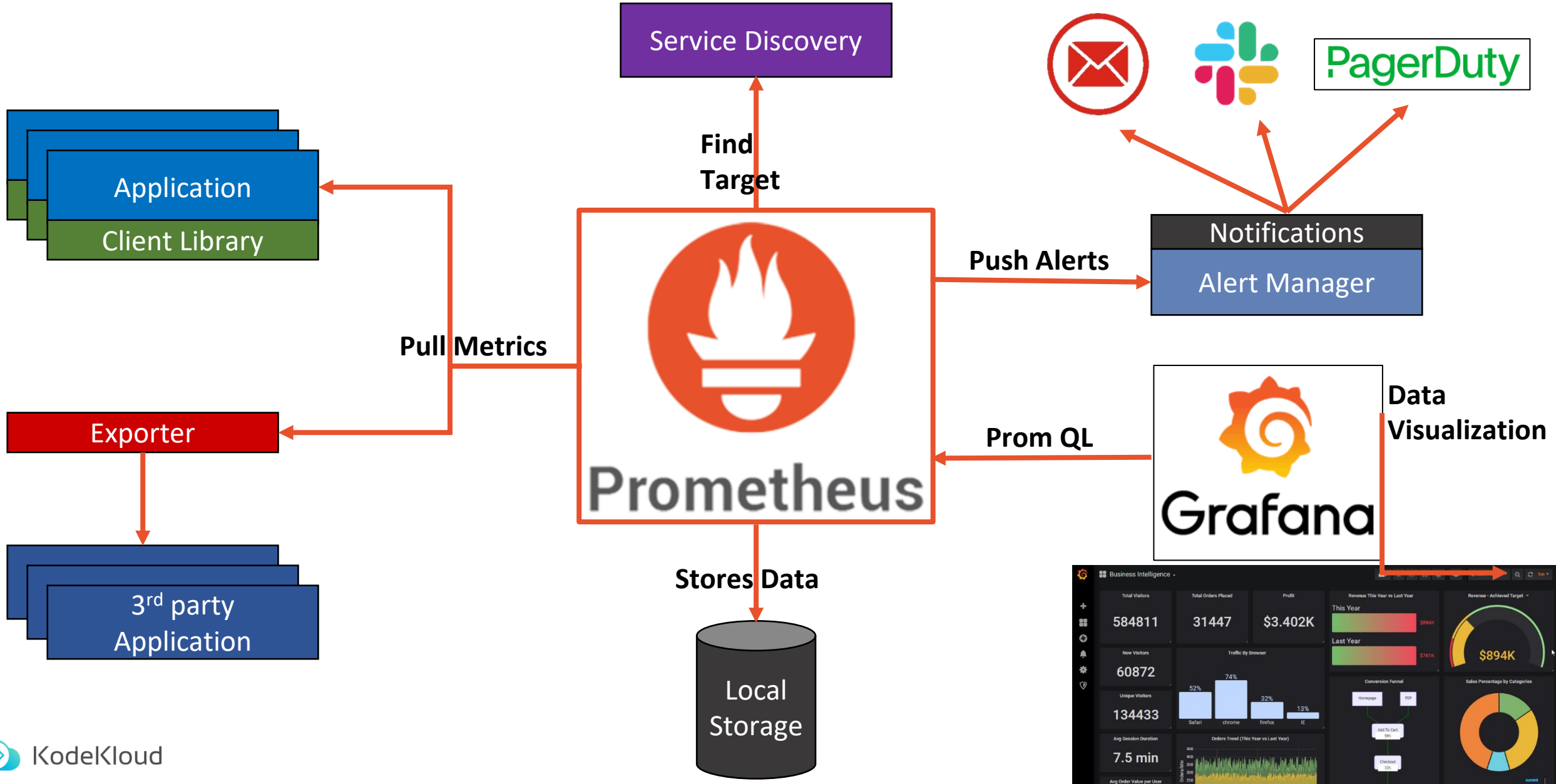
- Application availability:
- Application health and performance

Kubernetes Monitoring Tools

- Kubernetes Dashboard
- Kubernetes Metrics Server
- CAdvisor and Heapster

KodeKloud

Prometheus & Grafana



KodeKloud

Monitoring

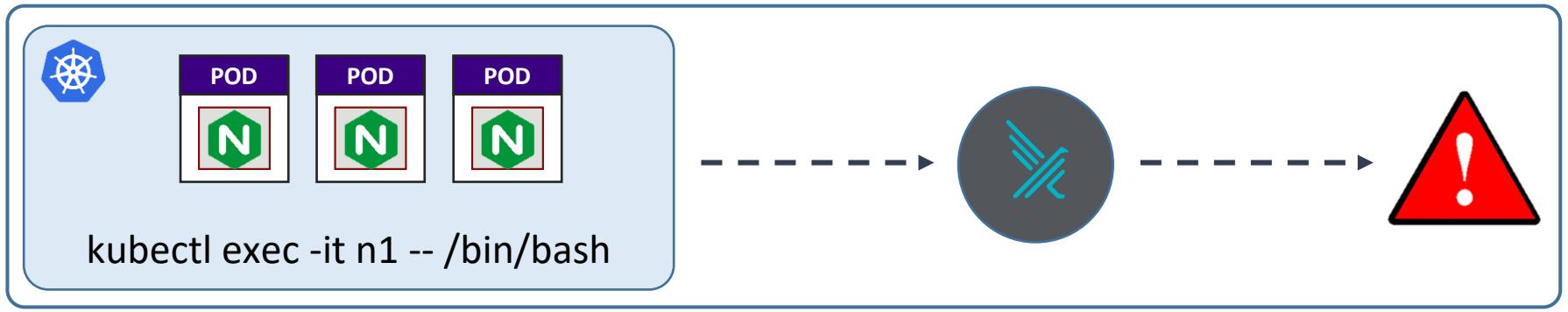
- Kube-scan
- **Falco**
- Helm
- Slack

KodeKloud

Falco

Falco, the open-source cloud-native runtime security project.

- Falco detects unexpected application behavior and alerts on threats at runtime.
- It has complete container visibility through a single sensor that allows us to gain insight into application and container behavior.
- Falco is based on **rules** that will trigger alerts when conditions are met, for example Falco is by default able to detect when:
  - A **shell** is run inside a container
  - A server **process spawns** a child process of an unexpected type
  - A sensitive file, like /**etc/shadow**, is unexpectedly read




```
Events detected: 6
Rule counts by severity:
   INFO: 2
   NOTICE: 3
   WARNING: 1
Triggered rules by rule name:
   Delete or rename shell history: 1
   Launch Privileged Container: 2
   Terminal shell in container: 3
Syscall event drop monitoring:
   - event drop detected: 0 occurrences
   - num times actions taken: 0
```

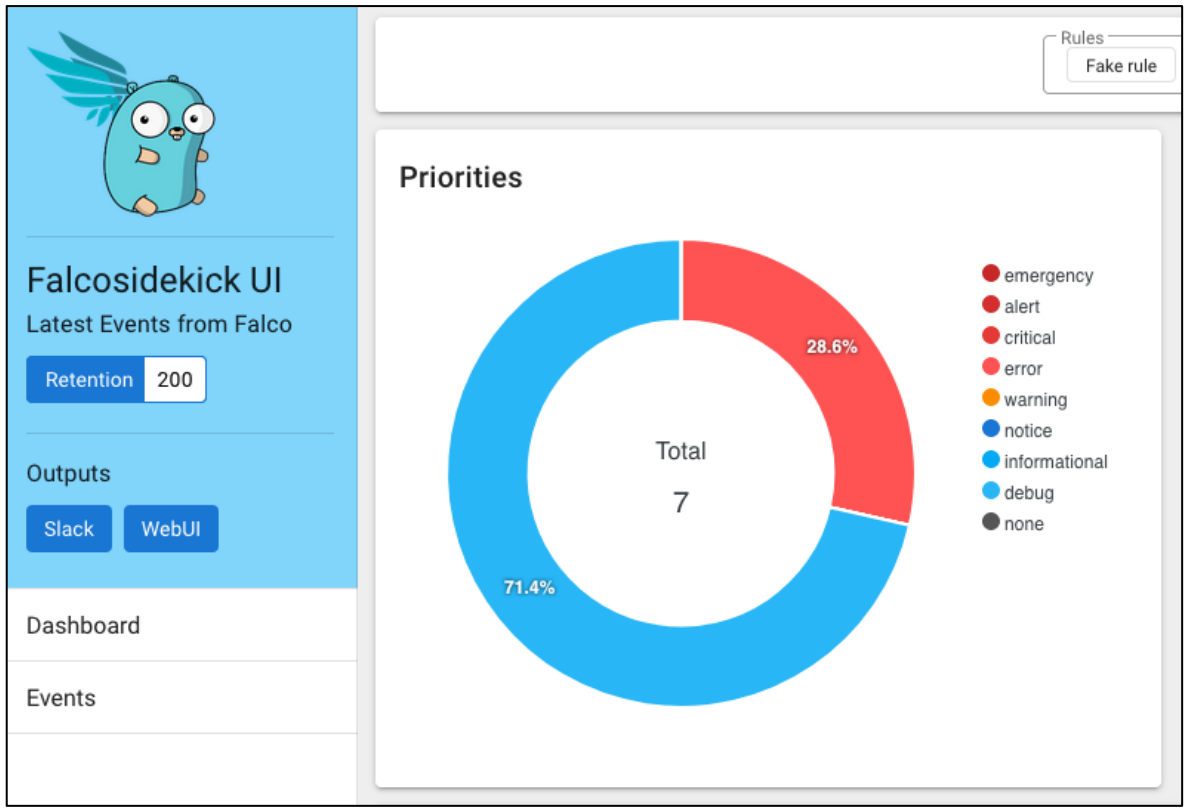
```
15:00:46.499981927: Notice A shell was spawned in a container with an attached terminal (user=root user loginuid=-1 k8s_n1_n1_default_3aad6671-0f60-4b43-bebf-f3
9048a10aaf_7 (id=cbfb5e6d4e96) shell=bash parent=runc cmdline=bash terminal=34816 container_id=cbfb5e6d4e96 image=nginx)
```

KodeKloud

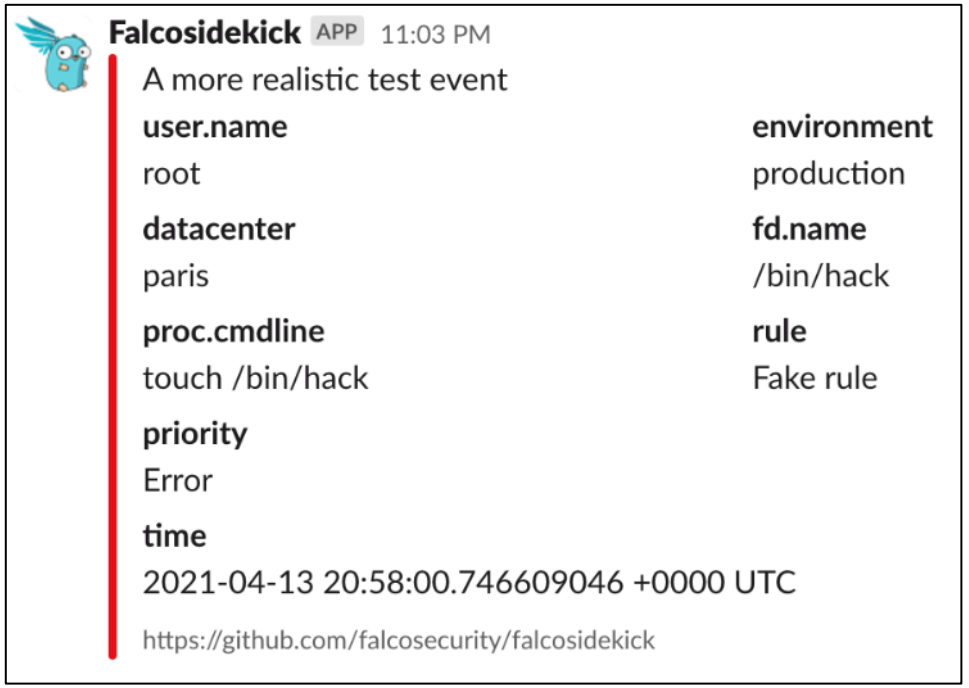
Extend Falco outputs with falcosidekick

- By default, Falco has 5 outputs for its events: **stdout**, **file**, **gRPC**, **shell** and **http**.
- These can be integrated with other components using **falcosidekick,** a daemon that extends that number of possible outputs.

Falcosidekick UI



Falco Slack Notification



KodeKloud

HANDS ON

- Falco
  - Terminal
  - UI
  - HELM
  - **Slack**

DEMO

KodeKloud

HELM

What?

- **Helm charts** are packages (like **rpms**) It contains pre-configured kubernetes resources such as
  - ConfigMaps,
  - Deployments,
  - Services.
- **Helm** is the package manager for Kubernetes (like **yum**) that allows easily package, configure, and deploy applications onto Kubernetes clusters.

```
$ rpm -i git.rpm
$ rpm -i git-sub-module.rpm
$ rpm -i git-sub-module.rpm
```

```
$ yum install git
```

Architecture

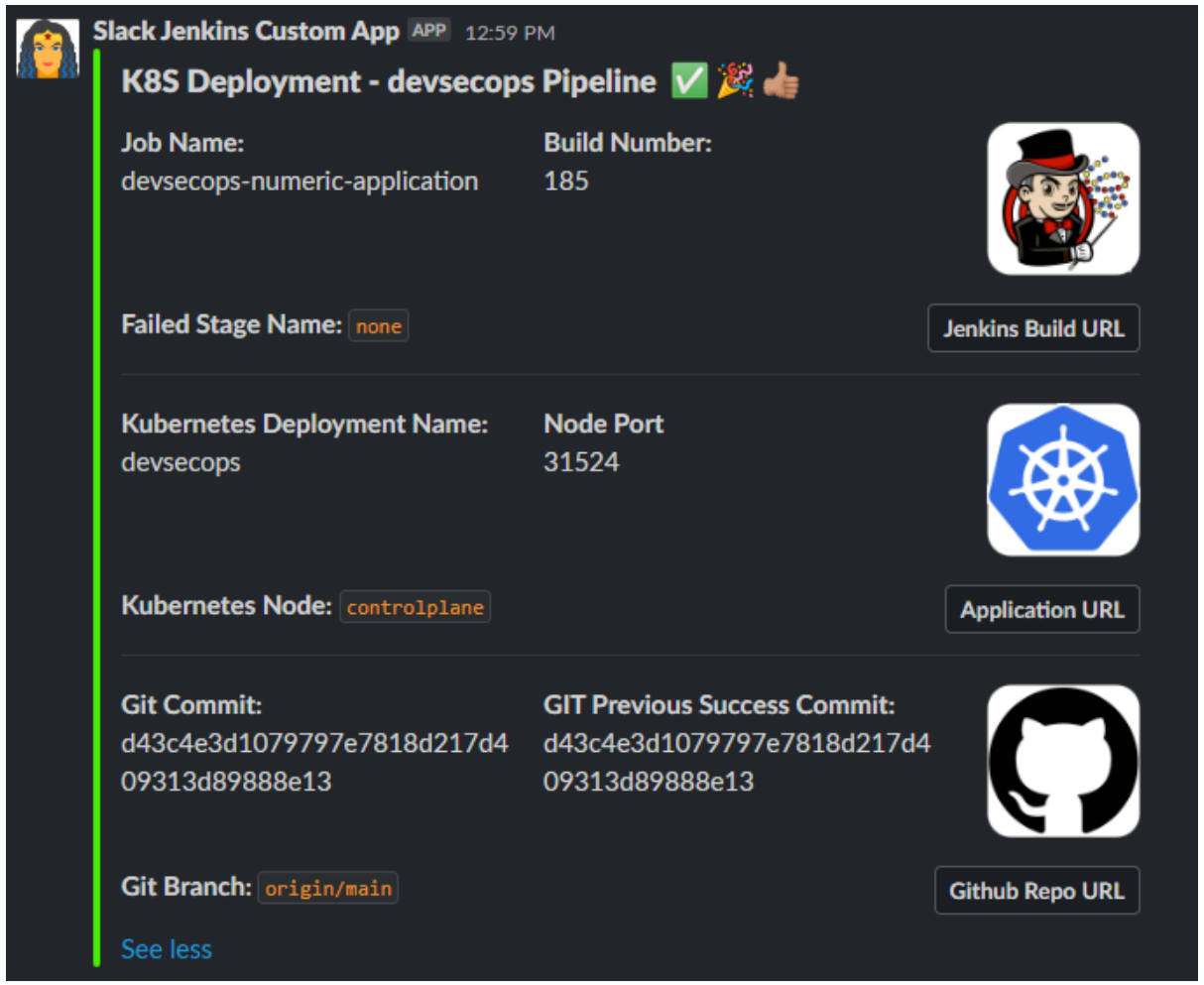
- Helm installs charts into Kubernetes, creating a new release for each installation. And to find new charts, you can search Helm chart repositories.
  - **Repository** is the place where charts can be collected and shared.
  - **Chart** is a Helm package. It contains all of the resource definitions necessary to run an application on Kubernetes Cluster
  - **Release** is an instance of a chart running in a Kubernetes cluster.

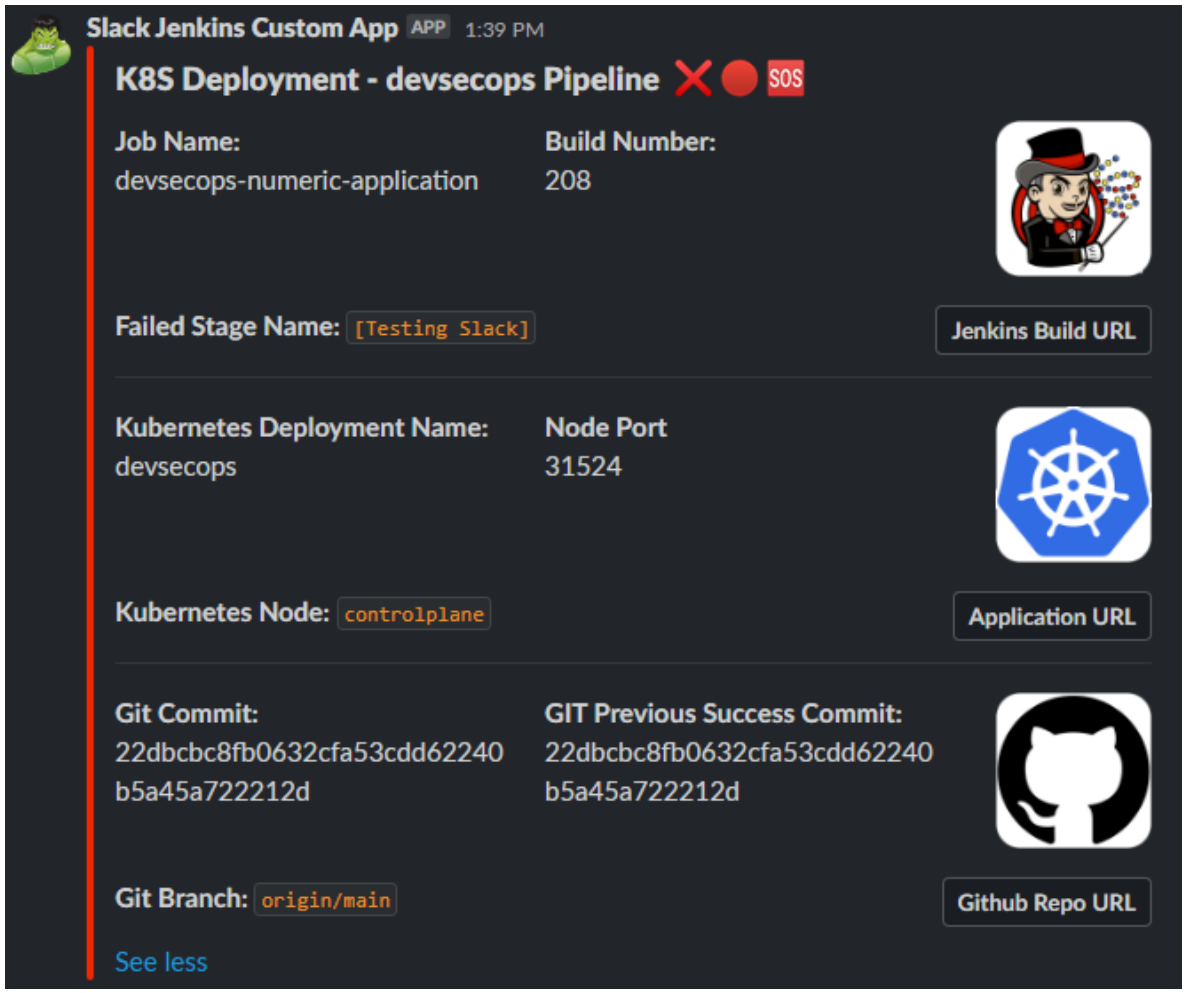
Installation

- export VERIFY_CHECKSUM=false
- curl https://raw.githubusercontent.com/helm/helm/master/scripts/get-helm-3 | bash
- helm version

KodeKloud

Slack – Rich Message Layout

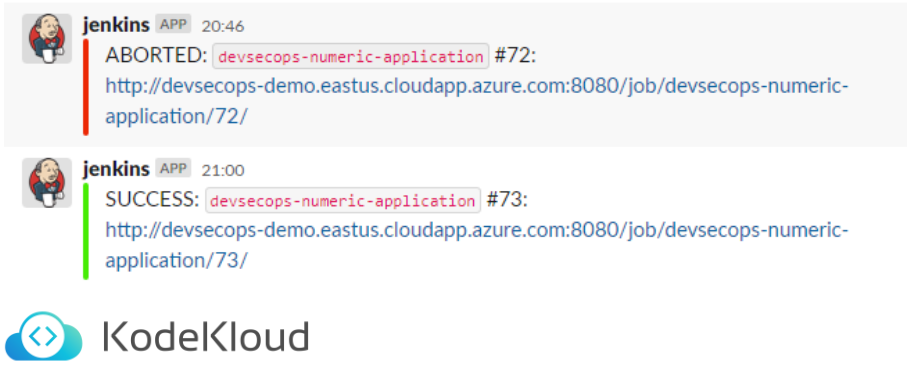


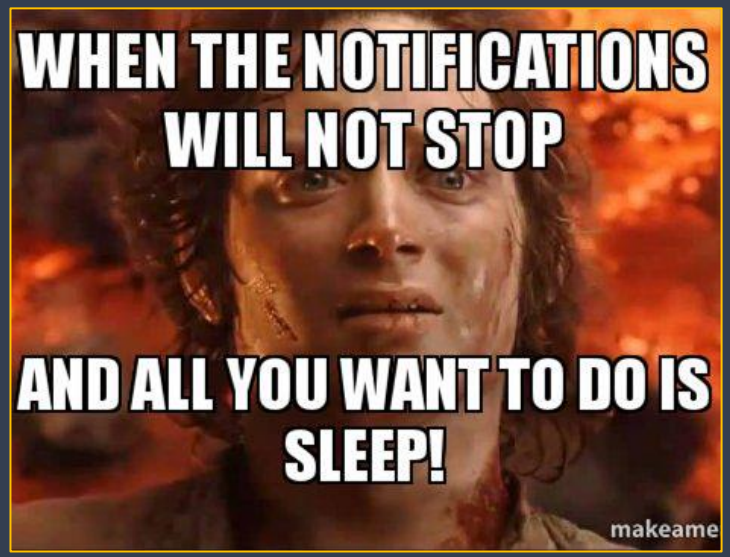


HANDS ON

DEMO

- Slack
  - Creating rich message layouts
    - Custom Slack Channel
    - Add Bot Permissions
    - Update Jenkins Slack Config
    - Use attachments
    - Get FailedStage Name





KodeKloud

Failed Stage Name – Programmatic Approach

```
/////// ****************************** Code for fectching Failed Stage Name ***************
import io.jenkins.blueocean.rest.impl.pipeline.PipelineNodeGraphVisitor
import io.jenkins.blueocean.rest.impl.pipeline.FlowNodeWrapper
import org.jenkinsci.plugins.workflow.support.steps.build.RunWrapper
import org.jenkinsci.plugins.workflow.actions.ErrorAction

// Get information about all stages, including the failure cases
// Returns a list of maps: [[id, failedStageName, result, errors]]
@NonCPS
List<Map> getStageResults( RunWrapper build ) {

    // Get all pipeline nodes that represent stages
    def visitor = new PipelineNodeGraphVisitor( build.rawBuild )
    def stages = visitor.pipelineNodes.findAll{ it.type == FlowNodeWrapper.NodeType.STAGE }

    return stages.collect{ stage ->

        // Get all the errors from the stage
        def errorActions = stage.getPipelineActions( ErrorAction )
        def errors = errorActions?.collect{ it.error }.unique()

        return [
            id: stage.id,
            failedStageName: stage.displayName,
            result: "${stage.status.result}",
            errors: errors
        ]
    }
}

// Get information of all failed stages
@NonCPS
List<Map> getFailedStages( RunWrapper build ) {
    return getStageResults( build ).findAll{ it.result == 'FAILURE' }
}
```

```
[
  id: 15,
  failedStageName: TestingSlack,
  result: FAILURE,
  errors: [
    hudson.AbortException: scriptreturnedexitcode1
  ]
]
```

```
stage('Ok') {
  steps {
    echo 'do thing'
  }
  post {
    failure {
      echo 'FAILED (in stage OK - should
    }
  }
}
stage('NotOK') {
    steps {
      sh 'make fail'
    }
    post {
      failure {
        echo 'FAILED (in stage NotOK)'
      }
    }
```

```
failure {
    script {
      //Fetch information about  failed stage
      def failedStages = getFailedStages( currentBuild )
      env.failedStage = failedStages.failedStageName
      env.emoji = ":x: :red_circle: :sos:"
      sendNotifications currentBuild.result
    }
}
```

Scripts not permitted to use method org.jenkinsci.plugins.workflow.support.steps.build.RunWrapper getRawBuild.
Administrators can decide whether to approve or reject this signature.

KodeKloud

KodeKloud

HashiCorp Vault

Kubernetes - Self-healing | Automatic Rollouts | Auto Scaling | Load Balancing | Storage Orchestration | Service Discovery

Kubernetes - Secrets

KodeKloud

Vault

Vault is a tool for securely accessing secrets.

- Installation
- Initialization
- Secrets
- Authorization
- Authentication

Vault – Installation

Vault is a tool for securely accessing secrets.

- Installation
- Initialization
- Secrets
- Authorization
- Authentication

Linux Package | Precompiled Binary | From Source | HELM

```
git clone --branch v0.16.0 https://github.com/hashicorp/vault-helm.git
helm install vault --set='ui.enabled=true'  ./vault-helm
```

PACKAGE MANAGER

Ubuntu/Debian | CentOS/RHEL | Fedora | Amazon Linux | Homebrew

```
$ curl -fsSL https://apt.releases.hashicorp.com/gpg | sudo apt-key add -
$ sudo apt-add-repository "deb [arch=amd64] https://apt.releases.hashicorp.com $(lsb_release -cs) main"
$ sudo apt-get update && sudo apt-get install vault
```

KodeKloud

Vault – Initialization

Vault is a tool for securely accessing secrets.

Encryption Key

Unseal Keys

Initial Root Token

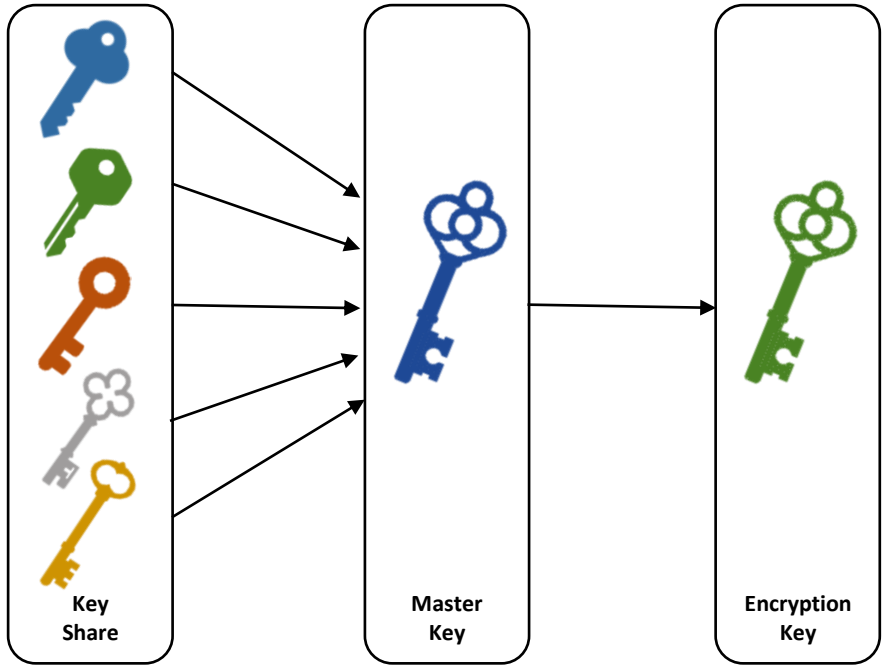
- Installation
- Initialization
- Secrets
- Authorization
- Authentication

vault operator init

```
Unseal Key 1: 4jYbl2CBIv6SpkKj6Hos9iD32k5RfGkLzlosrrq/JgOm
Unseal Key 2: B05G1DRtfYckFV5BbdBvXq0wkK5HFqB9g2jcDmNfTQiS
Unseal Key 3: Arig0N9rN9ezkTRo7qTB7gsIZDaonOcc53EHo83F5chA
Unseal Key 4: 0cZE0C/gEk3YHaKjIWxhyyfs8REhqkRW/CSXTnmTilv+
Unseal Key 5: fYhZOseRgzxmJCmIqUdxEm9C3jB5Q27AowER9w4FC2Ck

Initial Root Token: s.KkNJYWF5g0pomcCLEmDdOVCW
```

vault operator init -key-shares=3 -key-threshold=2

```
Unseal Key 1: 4jYbl2CBIv6SpkKj6Hos9iD32k5RfGkLzlosrrq/JgOm
Unseal Key 2: B05G1DRtfYckFV5BbdBvXq0wkK5HFqB9g2jcDmNfTQiS
Unseal Key 3: Arig0N9rN9ezkTRo7qTB7gsIZDaonOcc53EHo83F5chA



Initial Root Token: s.KkNJYWF5g0pomcCLEmDdOVCW
```



vault operator unseal <token 1>

vault operator unseal <token 2>

vault operator unseal <token 3>

vault login <Initial Root Token>

KodeKloud

Vault – Secrets

Vault is a tool for securely accessing secrets.

Key Values | Dynamic Credentials | PKI Certificates | SSH CA Certificates

- Installation
- Initialization
- Secrets
- Authorization
- Authentication

```
vault secrets enable -path=crds kv-v2
Success! Enabled the kv secrets engine at: crds/
```

```
vault kv get crds/mysql
No value found at crds/mysql
```

```
vault kv put crds/mysql username=root
Key             Value
---             -----
created_time    2021-08-31T11:17:38.755927206Z
deletion_time   n/a
destroyed       false
version         1
```

```
vault kv put crds/mysql username=root password=12345
Key             Value
---             -----
created_time    2021-08-31T11:19:45.645227215Z
deletion_time   n/a
destroyed       false
version         2
```

```
vault kv metadata get crds/mysql
```

```
vault kv get crds/mysql
====== Metadata ======
Key             Value
---             -----
created_time    2021-08-31T11:19:45.645227215Z
deletion_time   n/a
destroyed       false
version         2

====== Data ======
Key             Value
---             -----
password        12345
username        root
```

```
vault kv delete crds/mysql
Success! Data deleted (if it existed) at: secret/mysql
```

```
vault kv get crds/mysql
====== Metadata ======
Key             Value
---             -----
created_time    2021-08-31T11:19:45.645227215Z
deletion_time   2021-08-31T11:32:43.345208438Z
destroyed       false
version         2
```

KodeKloud

Vault – Authorization

Vault is a tool for securely accessing secrets.

- Installation
- Initialization
- Secrets
- **Authorization**
- Authentication

```
vault secrets enable -path=crds kv-v2
Success! Enabled the kv secrets engine at: crds/
```

```
# policies grant permission on paths
cat <<EOF > /home/vault/app-policy.hcl
path "crds/data/mongodb" {
  capabilities = ["create", "update"]
}

path "crds/data/mysql" {
  capabilities = ["read"]
}
EOF
```

```
vault policy write app /home/vault/app-policy.hcl
Success! Uploaded policy: app
```

```
vault policy list
app
default
root
```

```
vault policy read app
path "crds/data/mongodb" {
  capabilities = ["create", "update"]
}

path "crds/data/mysql" {
  capabilities = ["read"]
}
```

```
export VAULT_TOKEN="$(vault token create -field token -policy=app)"
echo $VAULT_TOKEN
s.mvJt75n7cccJL1Pg7ht0y2Mv
```

```
vault kv put crds/mysql username=root
Error writing data to crds/data/mysql: Error making API request.


URL: PUT http://127.0.0.1:8200/v1/crds/data/mysql
Code: 403. Errors:


* 1 error occurred:
        * permission denied
```

KodeKloud

Vault – Authentication

Vault is a tool for securely accessing secrets.

AWS | GCP | Azure

Kubernetes

Okta

LDAP/AD

Username & Password

PKI Certificates

GitHub

JWT/OIDC

- Installation
- Initialization
- Secrets
- Authorization
- Authentication

```
vault token create

Key                  Value
---                  -----
token                s.t3Z3QzflCI4AFRideymyCDYn
token_accessor       3gZeW9G8OcK80WVyewuizgth
token_duration       ∞
token_renewable      false
token_policies       ["root"]
identity_policies    []
policies             ["root"]
```

```
vault login s.t3Z3QzflCI4AFRideymyCDYn

Success! You are now authenticated.
```

```
vault token revoke s.t3Z3QzflCI4AFRideymyCDYn

Success! Revoked token (if it existed)
```

```
vault login s.t3Z3QzflCI4AFRideymyCDYn

Error authenticating: error looking up token: Error
making API request.

URL: GET
Code: 403. Errors: * permission denied
```

```
vault auth enable kubernetes

Success! Enabled kubernetes auth method at: kubernetes/
```

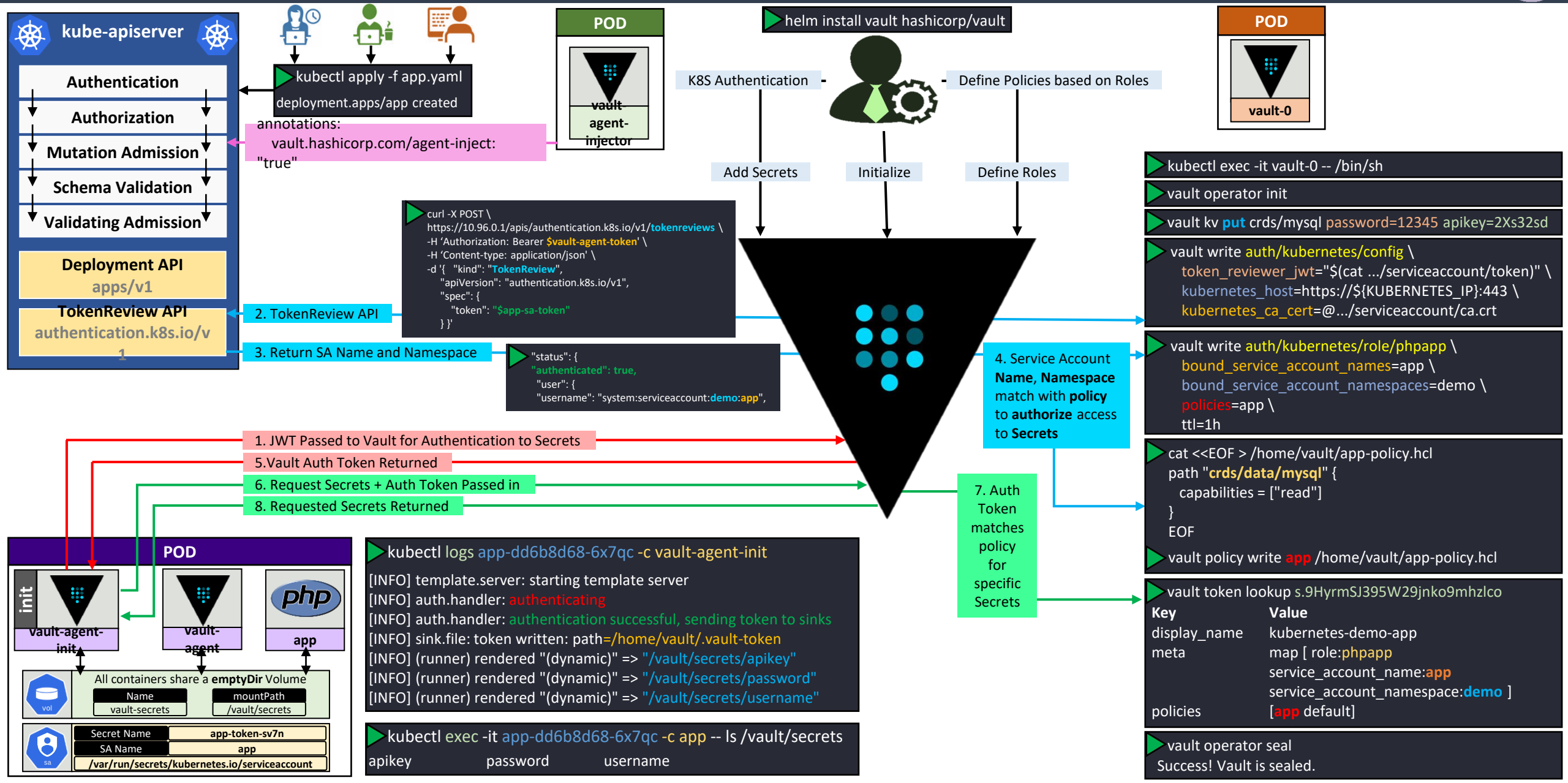
```
vault write auth/kubernetes/config \
    token_reviewer_jwt="$(cat /var/run/secrets/kubernetes.io/serviceaccount/token)" \
    kubernetes_host=https://${KUBERNETES_PORT_443_TCP_ADDR}:443 \
    kubernetes_ca_cert=@/var/run/secrets/kubernetes.io/serviceaccount/ca.crt

Success! Data written to: auth/kubernetes/config
```

```
vault write auth/kubernetes/role/phpapp \
    bound_service_account_names=app \
    bound_service_account_namespaces=demo \
    policies=app \
    ttl=1h

Success! Data written to: auth/kubernetes/role/phpapp
```

```
kubectl describe clusterrolebinding vault-server-binding
Role:
  Kind:  ClusterRole
  Name:  system:auth-delegator
Subjects:
  Kind                 Name     Namespace
  ----                 ----     ---------
  ServiceAccount       vault    demo
```

KodeKloud

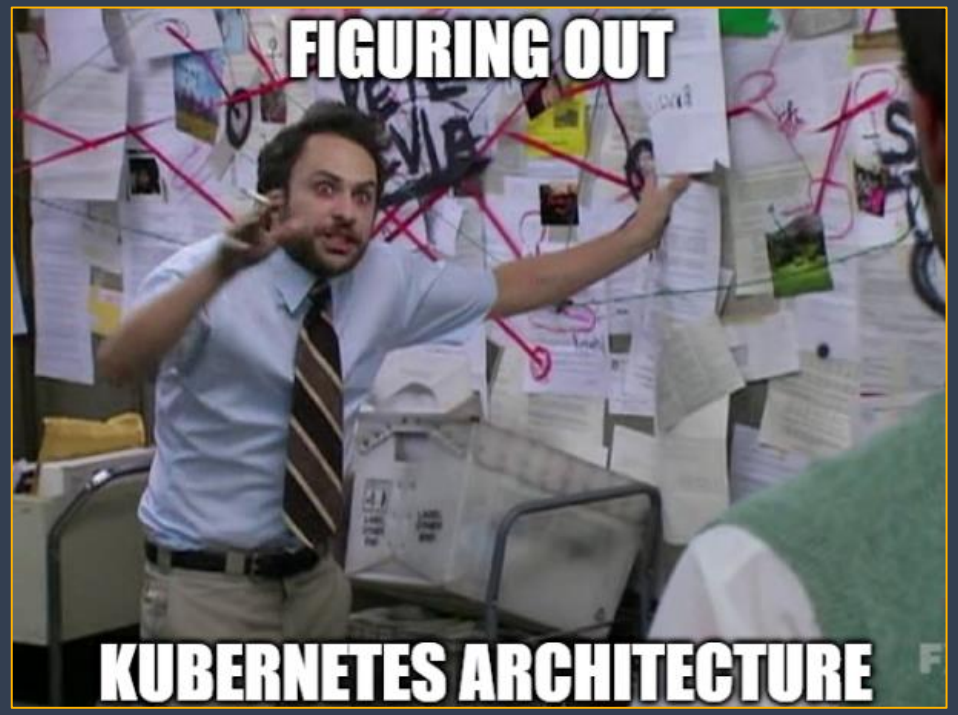
Kubernetes Secret Authentication & Access with Vault



KodeKloud

Kubernetes

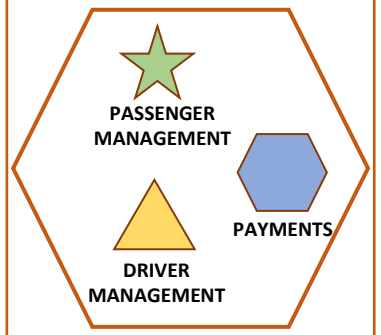
- What/Why/When?
- Architecture



KodeKloud

What? Why? When?

Monolith



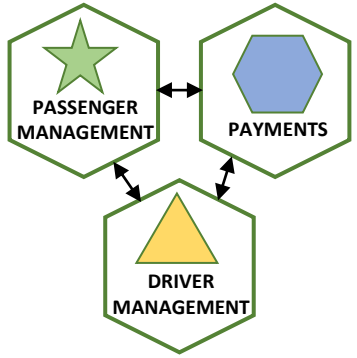
BENEFITS

Simple to Develop.
Simple to Deploy.
Simple Testing.

CHALLENGES

Hard to scale.
Too Large/Complex.
Redeploy the entire application on each update.

Microservices



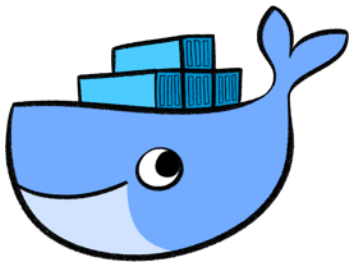
BENEFITS

Better Scaling.
New technologies.
Frequent Releases.

CHALLENGES

Slower Deployment.
Service Discovery.
Complex Configs.
Load Balancing.
Central Logging.

Docker



BENEFITS

Faster Deployment.
Faster start time.
Lightweight Image.

CHALLENGES

No Storage Option.
No Autoscaling.
No health-checks.

Kubernetes

BENEFITS

Storage options.
Auto-scaling.
Self-healing.

CHALLENGES

Hard to install.
No Building Image.
No built in Jenkins.
Separate install for Dashboard.

Openshift

BENEFITS

OOTB Webconsole.
OOTB Build Images.
Built in Jenkins.
Build Images and store in Registry.

CHALLENGES

Requires additional API Management.
Limited installation options.

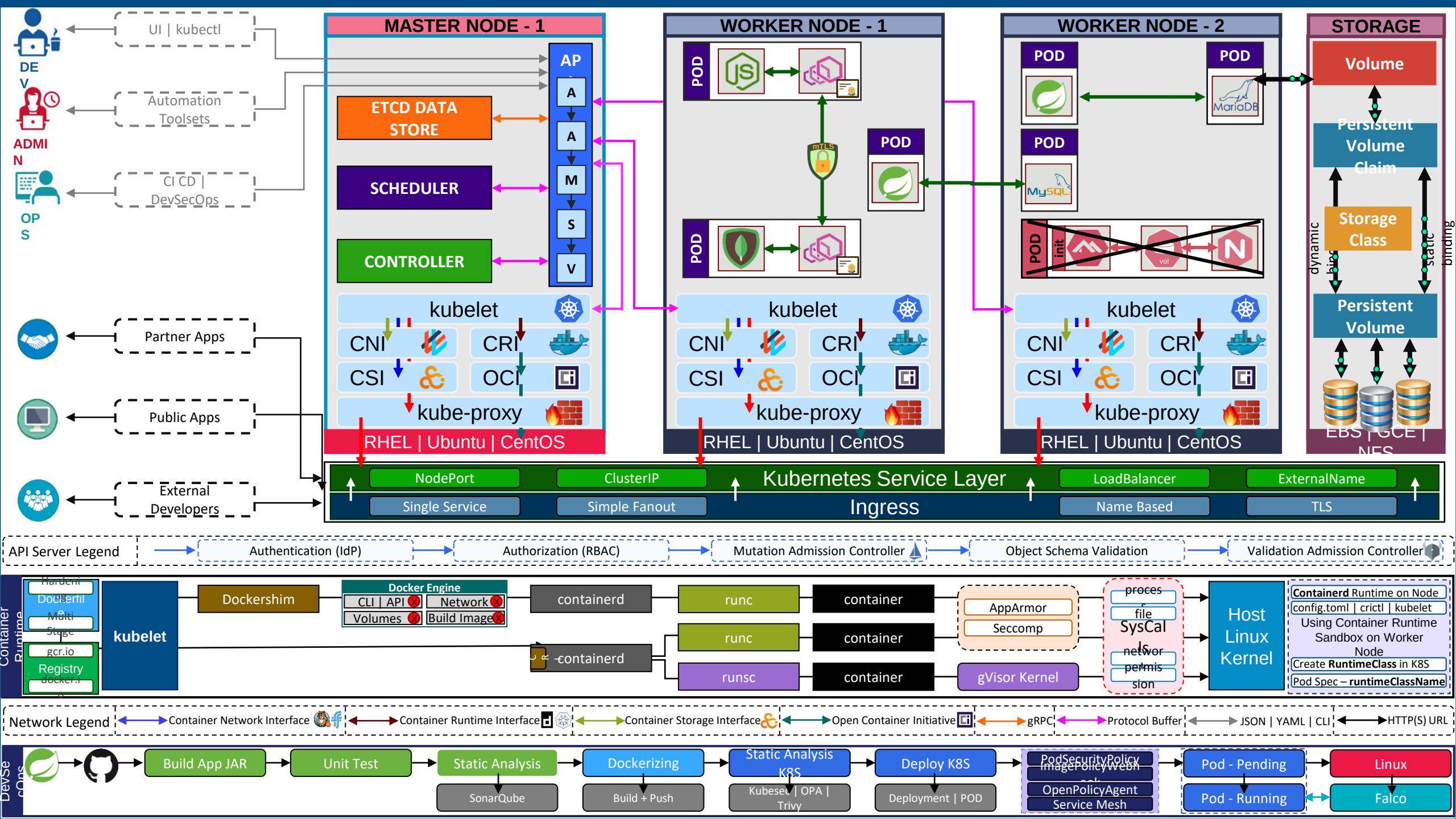
Istio

BENEFITS

Tracing.
Circuit Breaker.
Canary Release.
Dark Launches.
Telemetry.

CHALLENGES

Added Complexity.
Adds Overhead.
Required Expertise.


DEV
ADMIN
OPS
UI | kubectl
Automation Toolsets
CI CD | DevSecOps
Partner Apps
Public Apps
External Developers
MASTER NODE - 1
API
A
A
M
S
V
ETCD DATA STORE
SCHEDULER
CONTROLLER
kubelet
CNI
CRI
CSI
OCI
kube-proxy
RHEL | Ubuntu | CentOS
WORKER NODE - 1
POD
mTLS
WORKER NODE - 2
MariaDB
MySQL
init
vol
STORAGE
Volume
Persistent Volume Claim
Storage Class
dynamic
static
binding
Persistent Volume
EBS | GCE | NFS
NodePort
ClusterIP
Kubernetes Service Layer
LoadBalancer
ExternalName
Single Service
Simple Fanout
Ingress
Name Based
TLS
API Server Legend
Authentication (IdP)
Authorization (RBAC)
Mutation Admission Controller
Object Schema Validation
Validation Admission Controller
Container Runtime
Dockerfil
Multi Stage
gcr.io
Registry
kubelet
Dockershim
Docker Engine
CLI | API
Network
Volumes
Build Image
containerd
runc
container
AppArmor
Seccomp
runsc
gVisor Kernel
SysCalls
Host Linux Kernel
Containerd Runtime on Node
config.toml | crictl | kubelet
Using Container Runtime Sandbox on Worker Node
Create RuntimeClass in K8S
Pod Spec – runtimeClassName
Network Legend
Container Network Interface
Container Runtime Interface
Container Storage Interface
Open Container Initiative
gRPC
Protocol Buffer
JSON | YAML | CLI
HTTP(S) URL
DevSecOps
Build App JAR
Unit Test
Static Analysis
SonarQube
Dockerizing
Build + Push
Static Analysis K8S
Deploy K8S
Deployment | POD
PodSecurityPolicy
OpenPolicyAgent
Service Mesh
Pod - Pending
Pod - Running
Linux
Falco