DevOps and IT Delivery

DevOps Architecture

Introducing DevOps in IT delivery

- In its essence, DevOps is the *development* and *operations* stages working as one team, on the same product and managing it.

The benefits of DevOps are as follows:

- *It brings business and development*
- *Enterprises can respond faster to demands*
- *Products are continuously improved*
- *Enterprises can reduce costs in terms of both development and operations*

IT delivery in enterprises

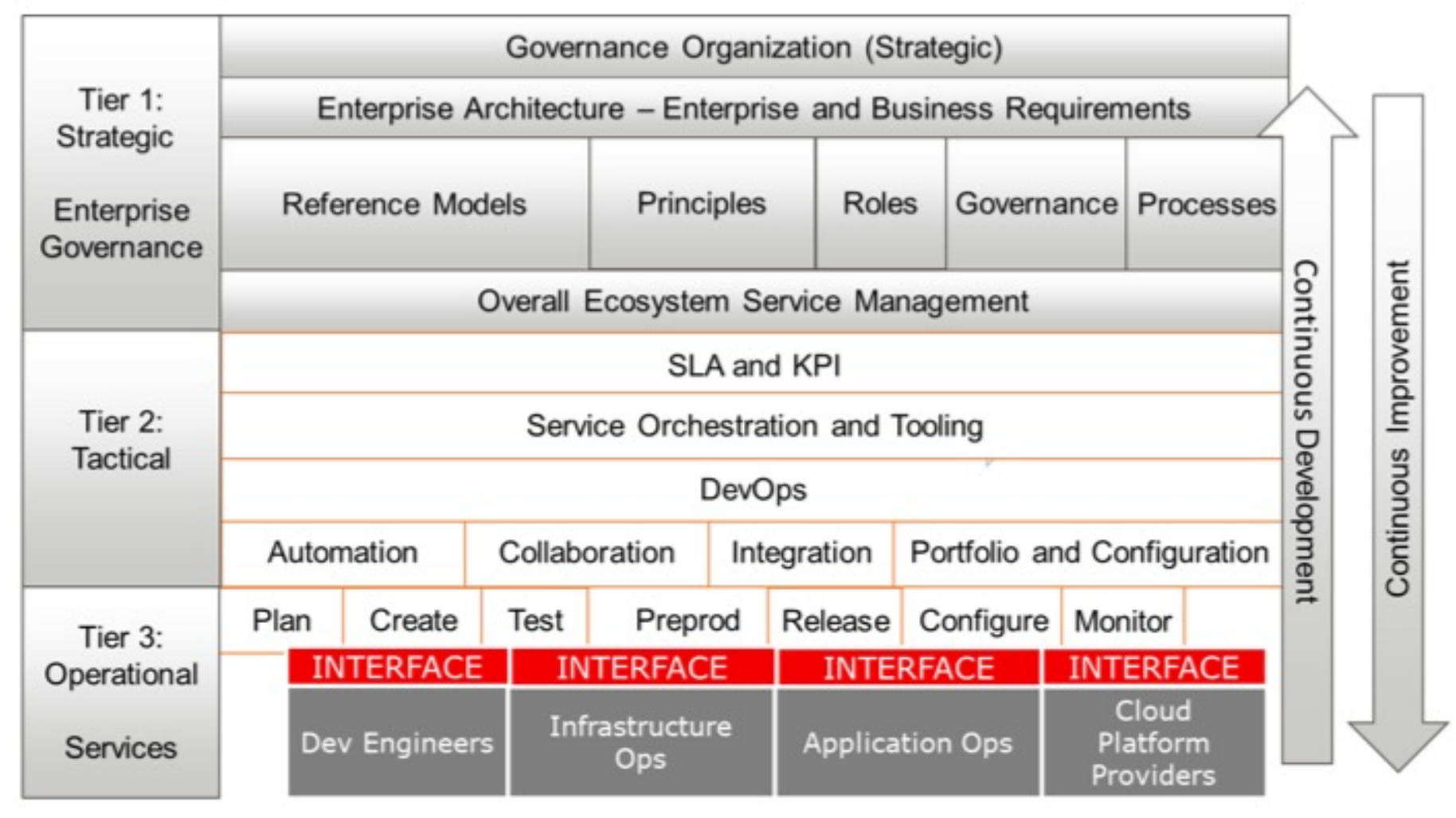
The main processes are in IT delivery.

- Business demand
- Business planning
- Development
- Deployment
- Operations

IT delivery in enterprises

- Continuous integration (CI): CI is built on the principle of a shared repository, where code is frequently updated and shared across teams that work in the cloud environments.
- Continuous delivery (CD): This is the automated transfer of software to test environments.

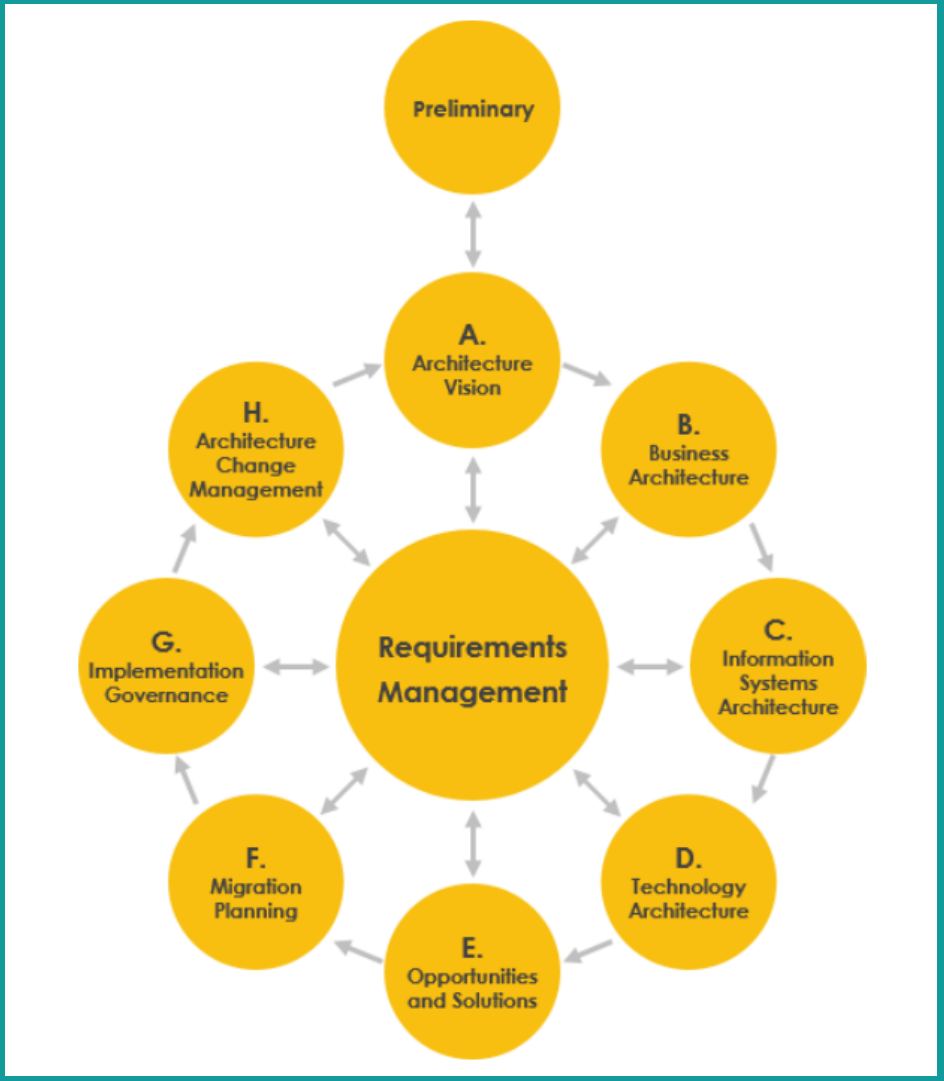
IT delivery in sourcing models



Basic Architecture

DevOps Architecture

Creating a reference architecture



Understanding the DevOps principles

Six principles from the DevOps Agile Skills Association (DASA):

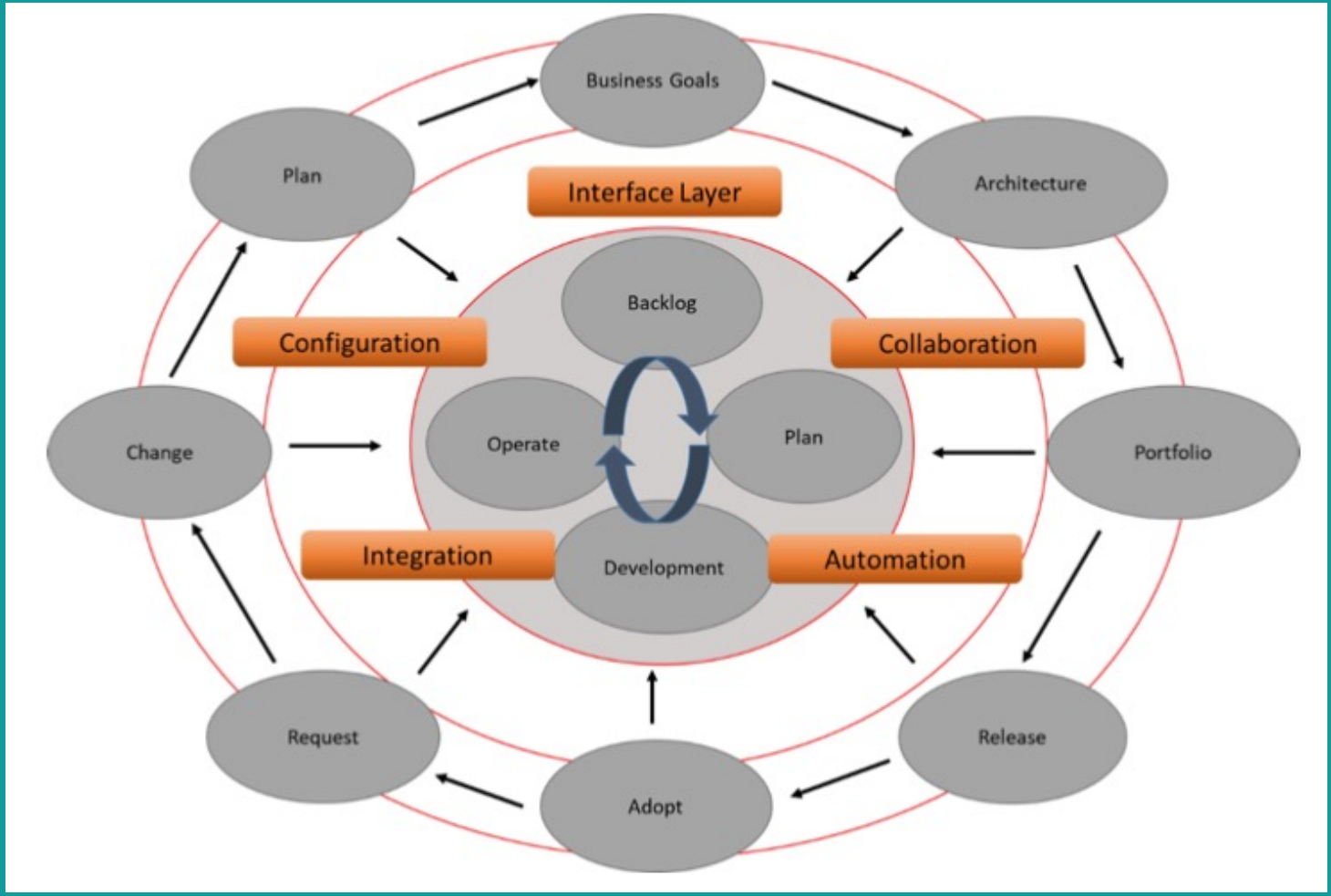
- Customer-centric action
- Create with the end-result in mind
- End-to-end responsibility
- Cross-functional autonomous teams
- Continuous improvement
- Automate as much as possible

Understanding the DevOps principles

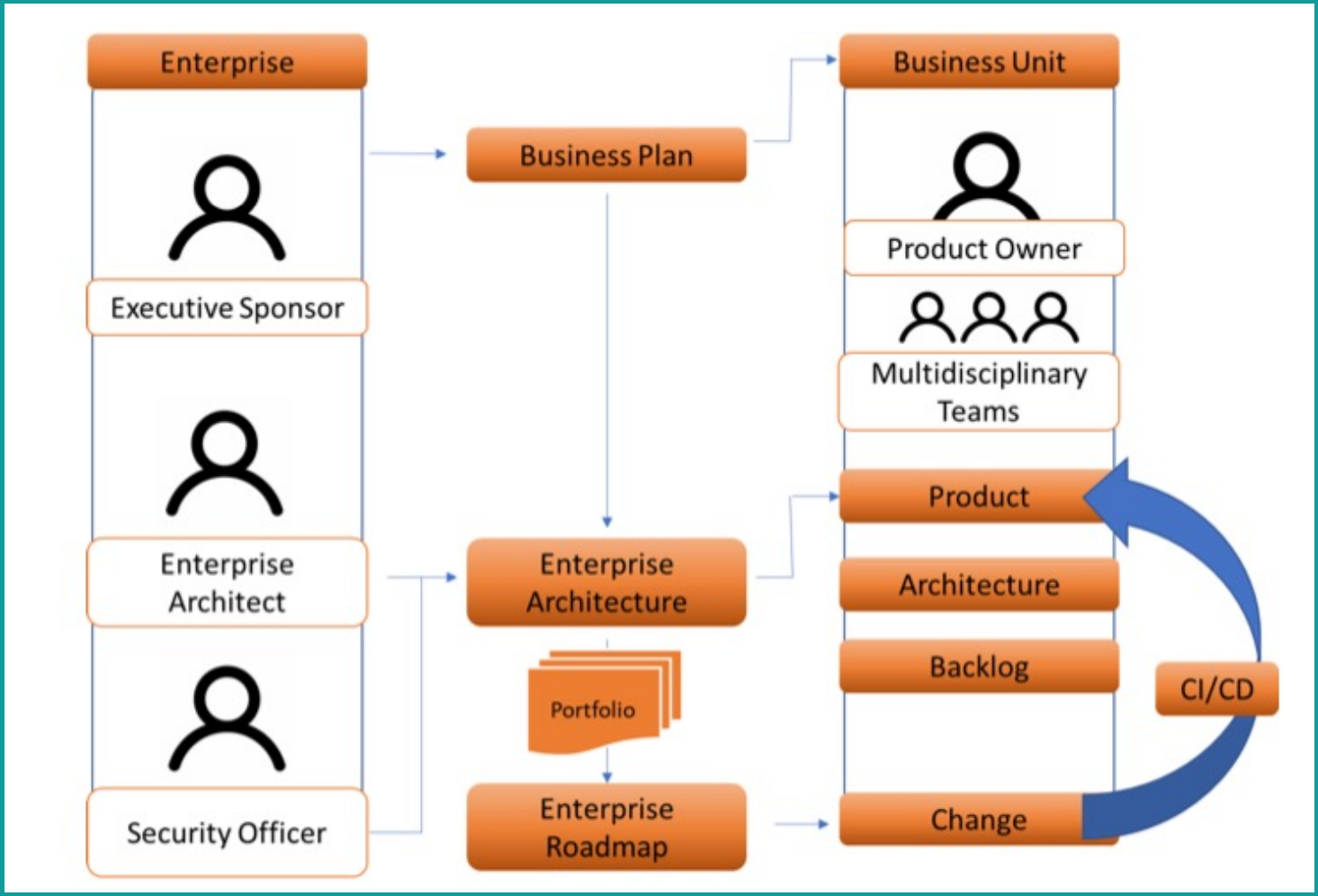
The following are the architecture statements, which are at the core of DevOps:

- Automation
- Collaboration
- Portfolio and configuration management
- Integration

DevOps architecture reference model



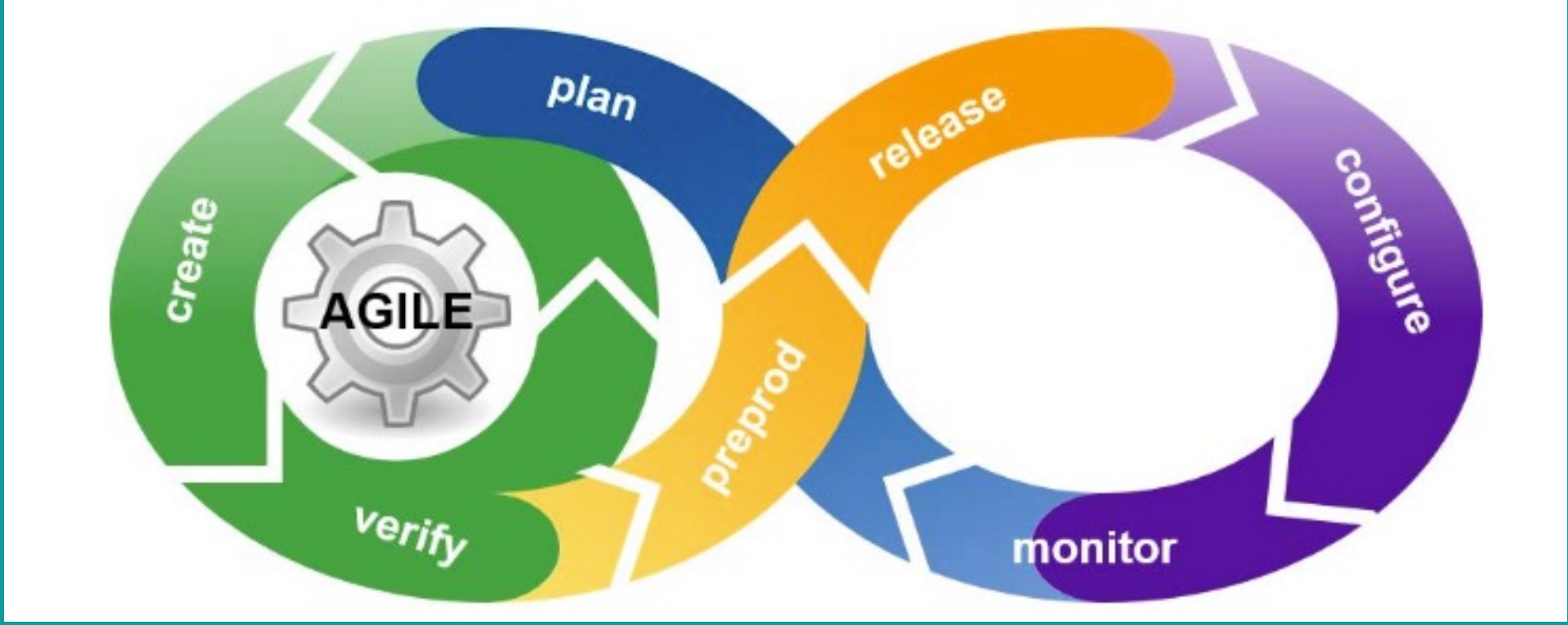
DevOps architecture reference model



DevOps Components

DevOps Architecture

Introducing DevOps components

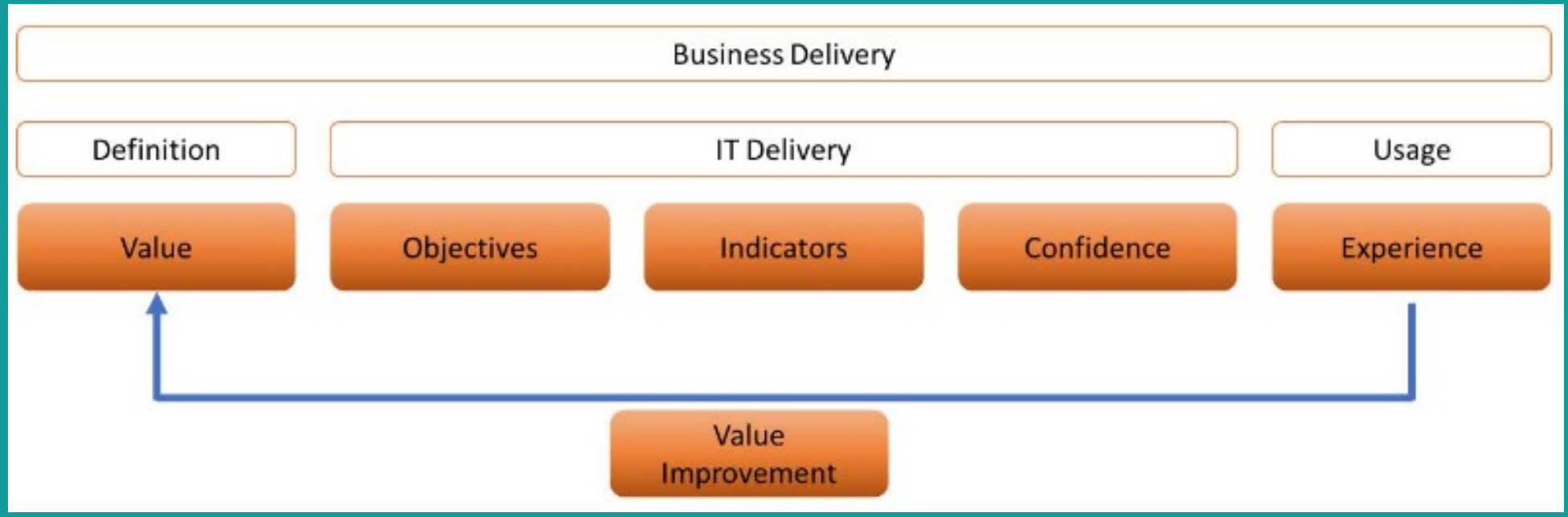


Understanding SLAs and KPIs in DevOps

The six most important metrics that should be included in SLAs for DevOps are as follows:

- Frequency of deployments
- Deployment time
- Deployment failure rate
- Deployment failure detection time
- Change lead time
- Full cycle time

Working with the VOICE model



DevSecOps Ecosystem

DevSecOps Architecture

DevSecOps Ecosystem

DevSecOps consists of three layers:

- Culture
- Security by design
- Automation

To manage these layers, DevSecOps relies on the following components:

- Harnessing repositories
- Application (code) security
- Cloud platform security
- Vulnerability assessments and testing

Creating the reference architecture

We need security from the start of the DevOps process. This calls for role-based access control (RBAC) and Identity and Access Management (IAM) on repositories.

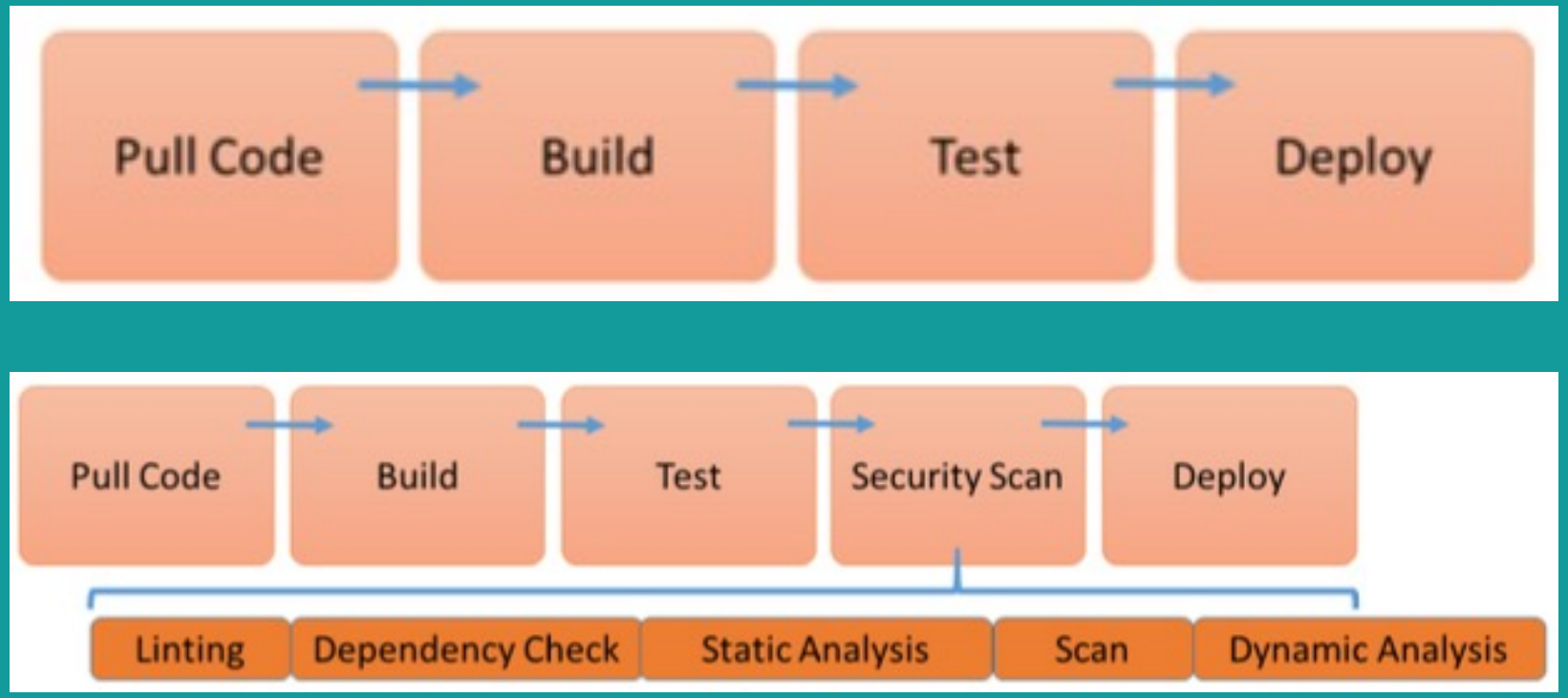
We can create the reference architecture for DevSecOps with the following components:

- Repository access with RBAC
- Static Application Security Testing (SAST)
- Software Composition Analysis (SCA)
- Dynamic Application Security Testing (DAST)

DevSecOps Pipline

DevSecOps Architecture

Creating the reference architecture



Using secured containers in the pipeline

Most developers will use containers to wrap and deploy their code, typically Docker containers. Applications consisting of containers are defined by Dockerfiles.

```
docker run --rm -i hadolint/hadolint
```

Applying secrets management

The best practices for secret management are as follows:

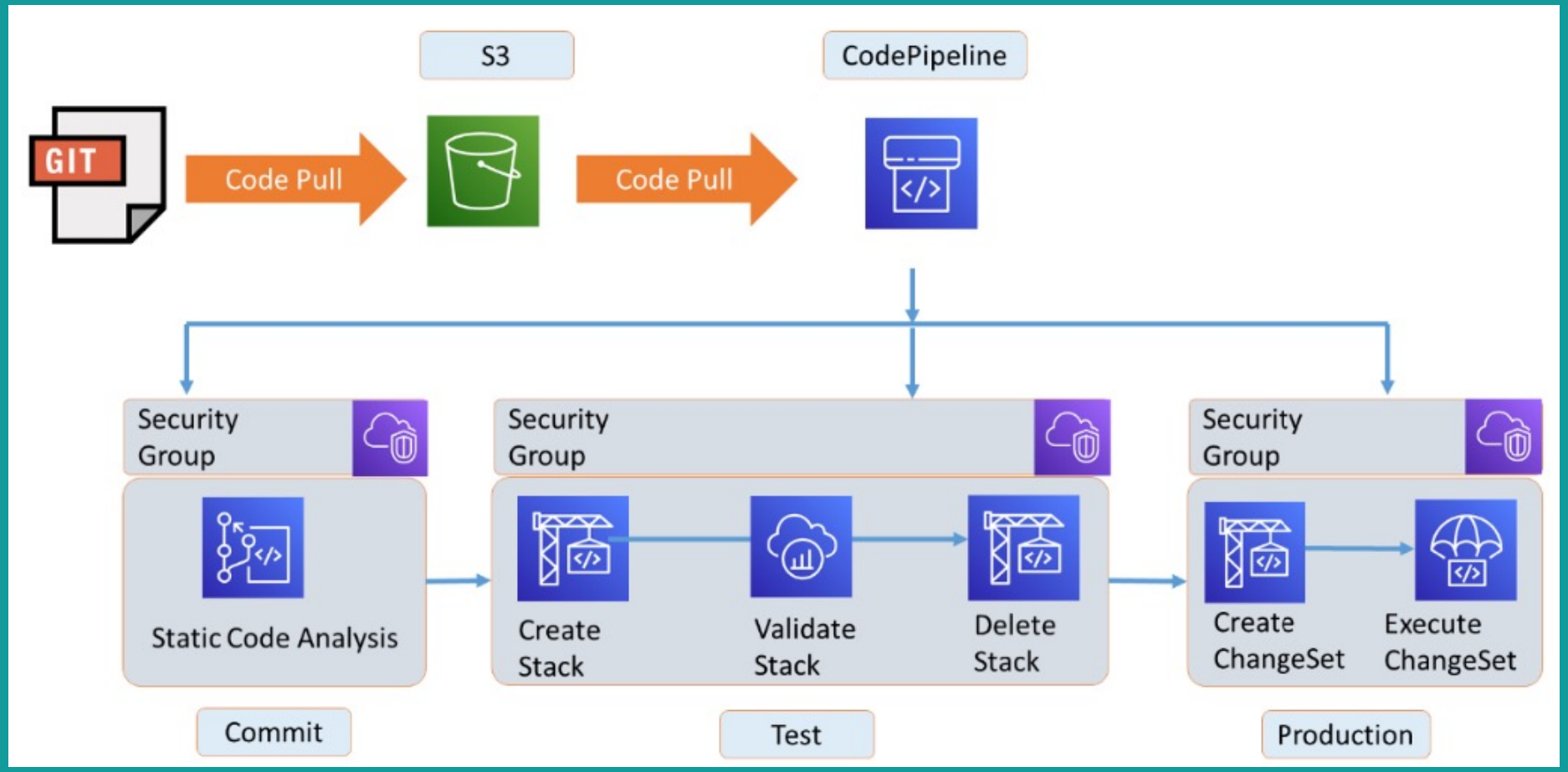
- Encryption at rest and in transit. AES-256 encryption keys are recommended.
- Secrets, such as keys, must never be stored in Git/GitHub repositories.
- It's advised that secrets are injected into the application via a secure string as an environment variable.

Implementing DevSecOps

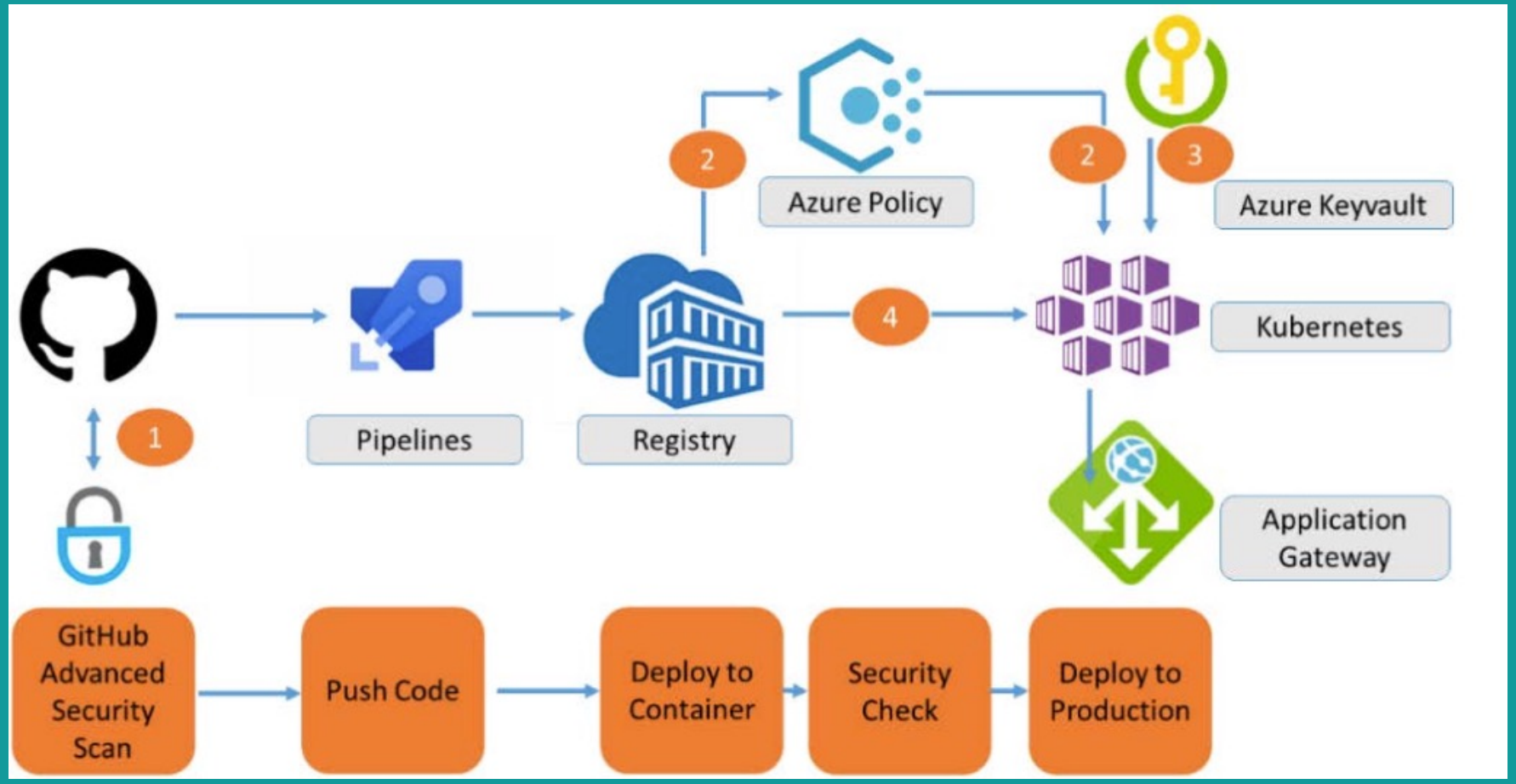
DevSecOps with AWS, Azure and Google Cloud

DevSecOps Architecture

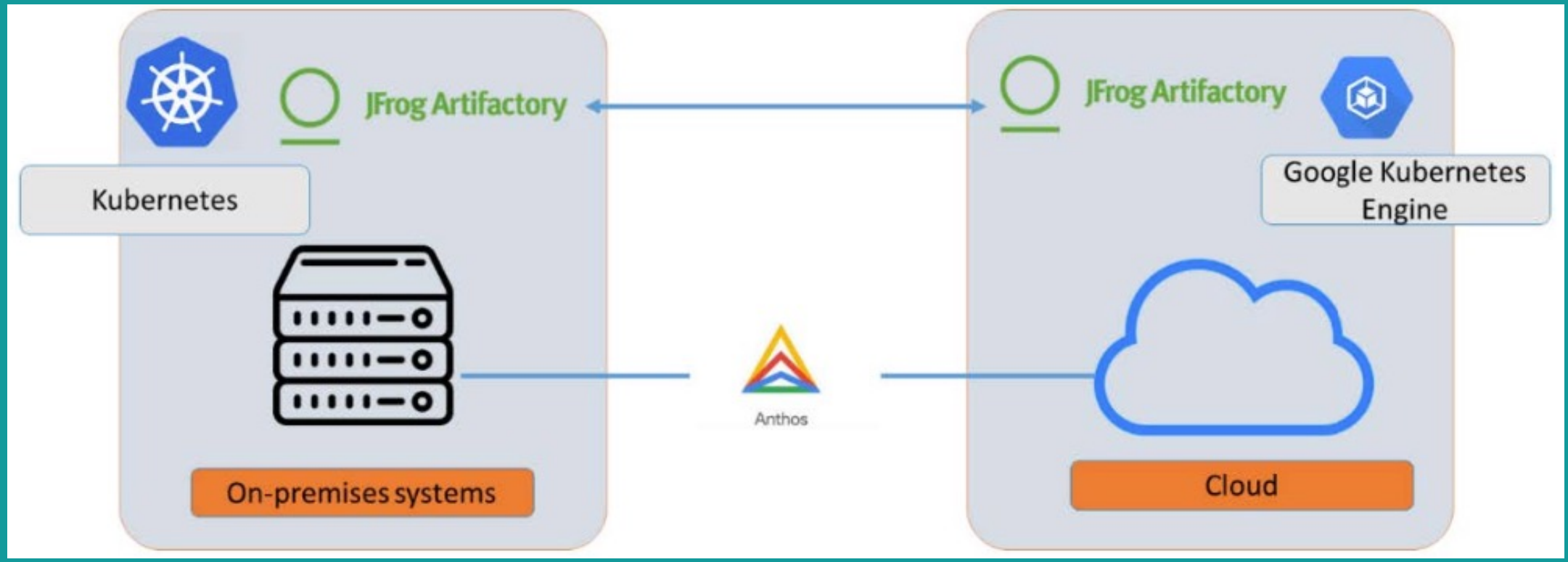
Working with DevSecOps in AWS CodePipeline



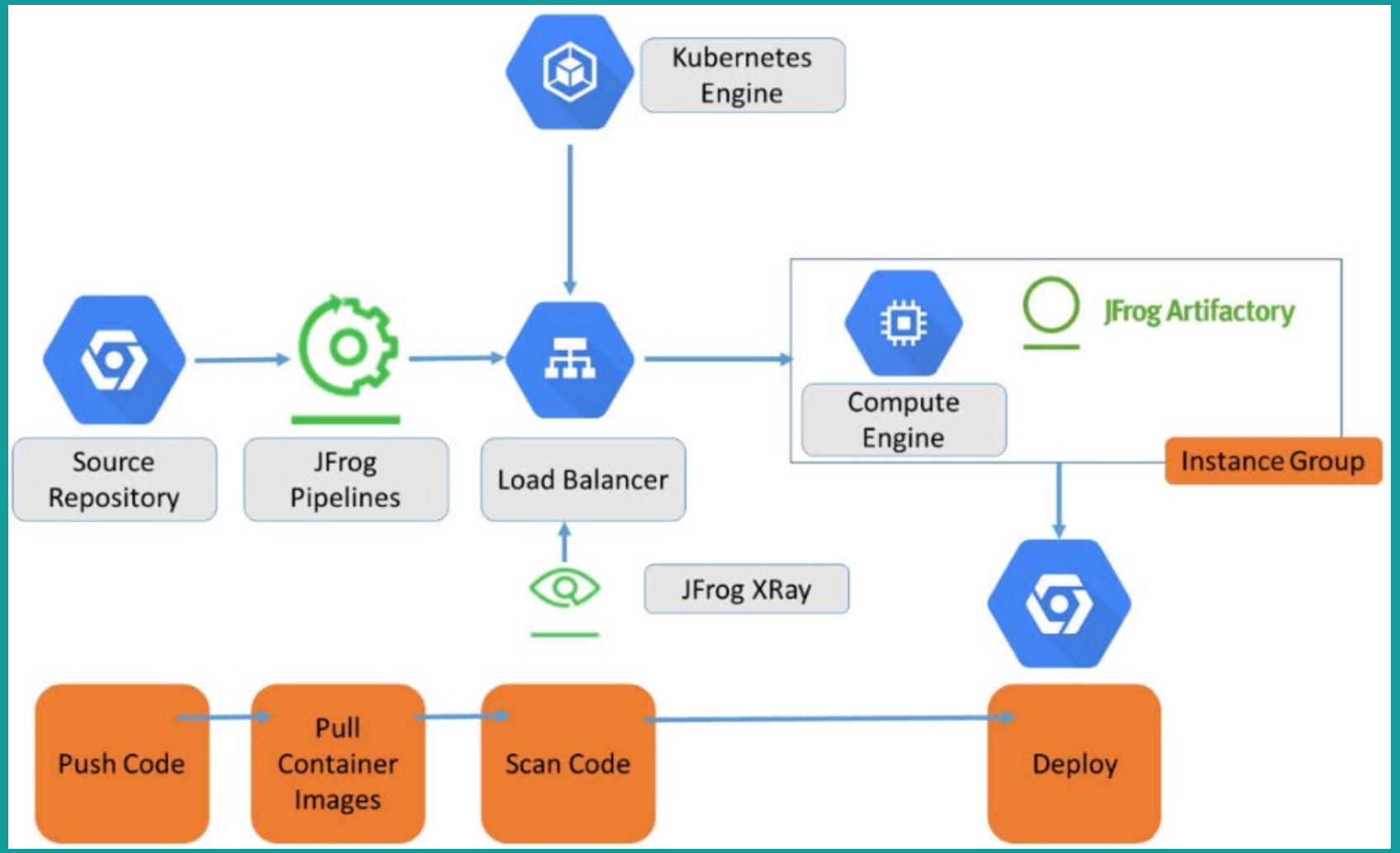
Working with DevSecOps using GitHub and Azure services



Working with DevSecOps in Google Cloud using Anthos and JFrog



Working with DevSecOps in Google Cloud using Anthos and JFrog



Deployment and Industry Security

DevSecOps Architecture

Planning for deployment

There are three major steps that enterprises will need to follow to implement DevSecOps:

- Assess the enterprise security:
- Embed security into DevOps:
- Train, train, train

Enterprises are recommended to include the following tools as a minimum:

- Testing
- Alerting
- Automated Remediation
- Visualization

Understanding industry security frameworks

Generic IT security frameworks include

- ISO IEC 27001/ISO 2700212
- National Institute of Standards and Technology (NIST) Cybersecurity Framework
- Center for Internet Security (CIS)
- Control Objectives for Information and Related Technologies (COBIT).

Implementing DevSecOps

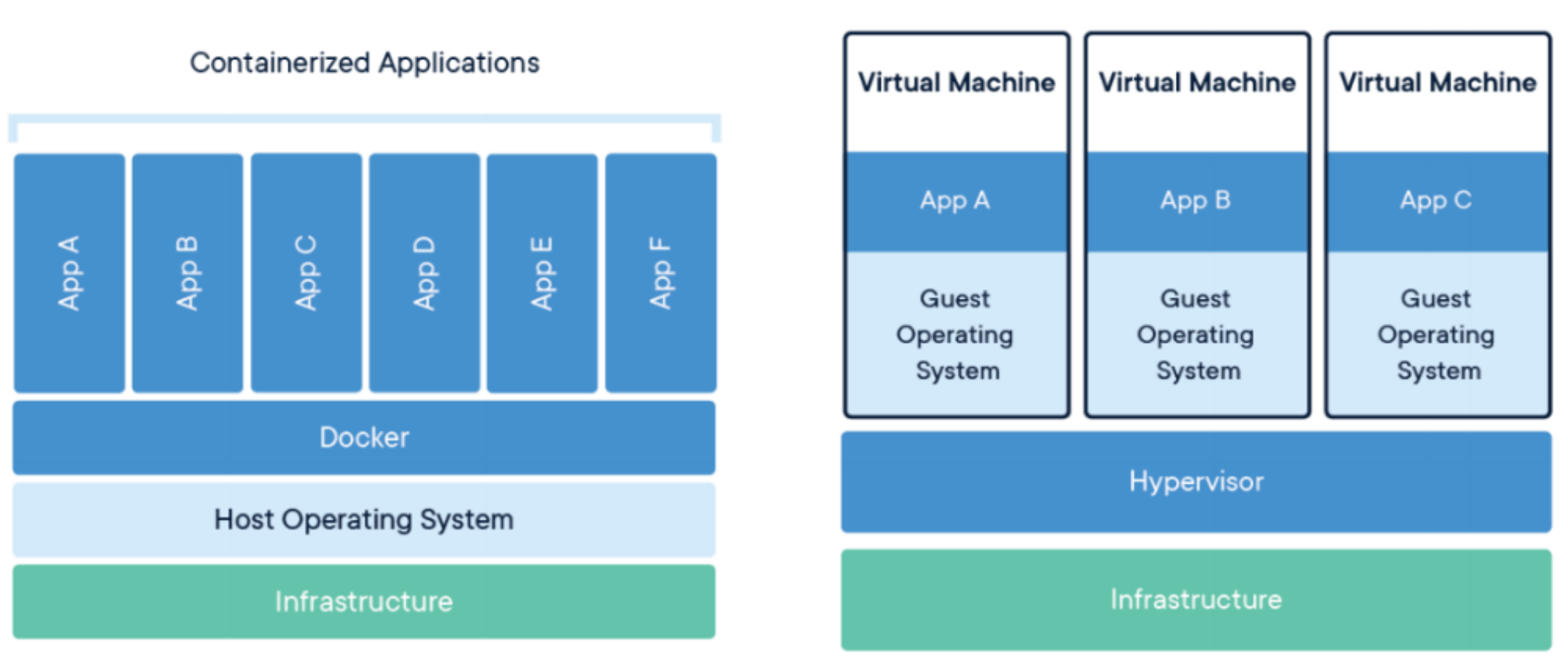
Docker Containers

Container Platforms

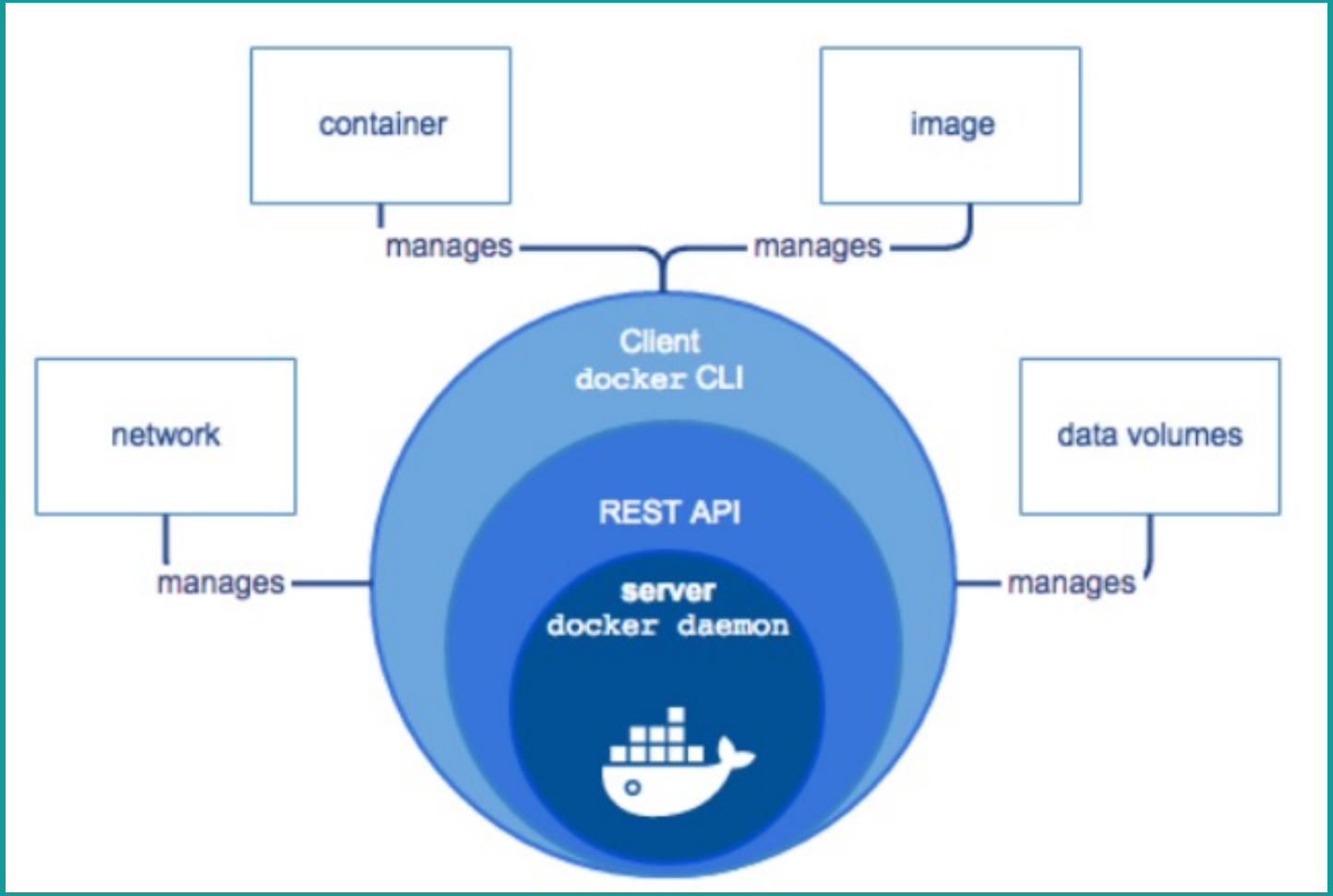
Docker

- Docker is a container platform to quickly develop, deploy, and manage applications, and it packages software into standardized units called **containers** that include everything necessary for the software to run, including libraries, system tools, and code.

Containers vs VMs



Docker architecture



Docker engine

Docker daemon: Docker engine uses a daemon process as a server that works in the background of the host system and allows central control of the Docker engine. It is also responsible for creating and managing all images, containers, or networks.

REST API: Specifies a series of interfaces that allows other applications to interact with the Docker daemon.

CLI: Docker uses the terminal of the operating system as a client program, which interacts with the daemon through the REST API .

Implementing DevSecOps

Working with Docker

Container Platforms

Implementing DevSecOps

Podman and Container Management

Container Platforms

Understanding industry security frameworks

- **Root-less:** It allows us to lift containers without having root privileges.
- **Daemon-less:** Podman does not need to raise a single daemon of many services to work.
- **Pods:** Podman coined the term pod as we know it with Kubernetes so that we could lift pods from one or more containers and isolate them from other pods.
- **Command line:** The commands are equivalent to those of Docker, and there are no differences.

Podman Features

Podman has some characteristics that make it interesting:

- It has a syntax equivalent to Docker, so you don’t need to learn a new set of instructions to manage your images and containers with Podman.
- Containers can be run as root or as a user without administrator rights. Podman manages the entire container ecosystem, including pods,
- containers, images, volumes, and all using the libpod library. Podman only works on Linux platforms, although it supports different
- image formats, including OCI and Docker.
- You don’t need a daemon or background application running permanently.

Container orchestration

A container orchestrator is responsible for the following tasks:

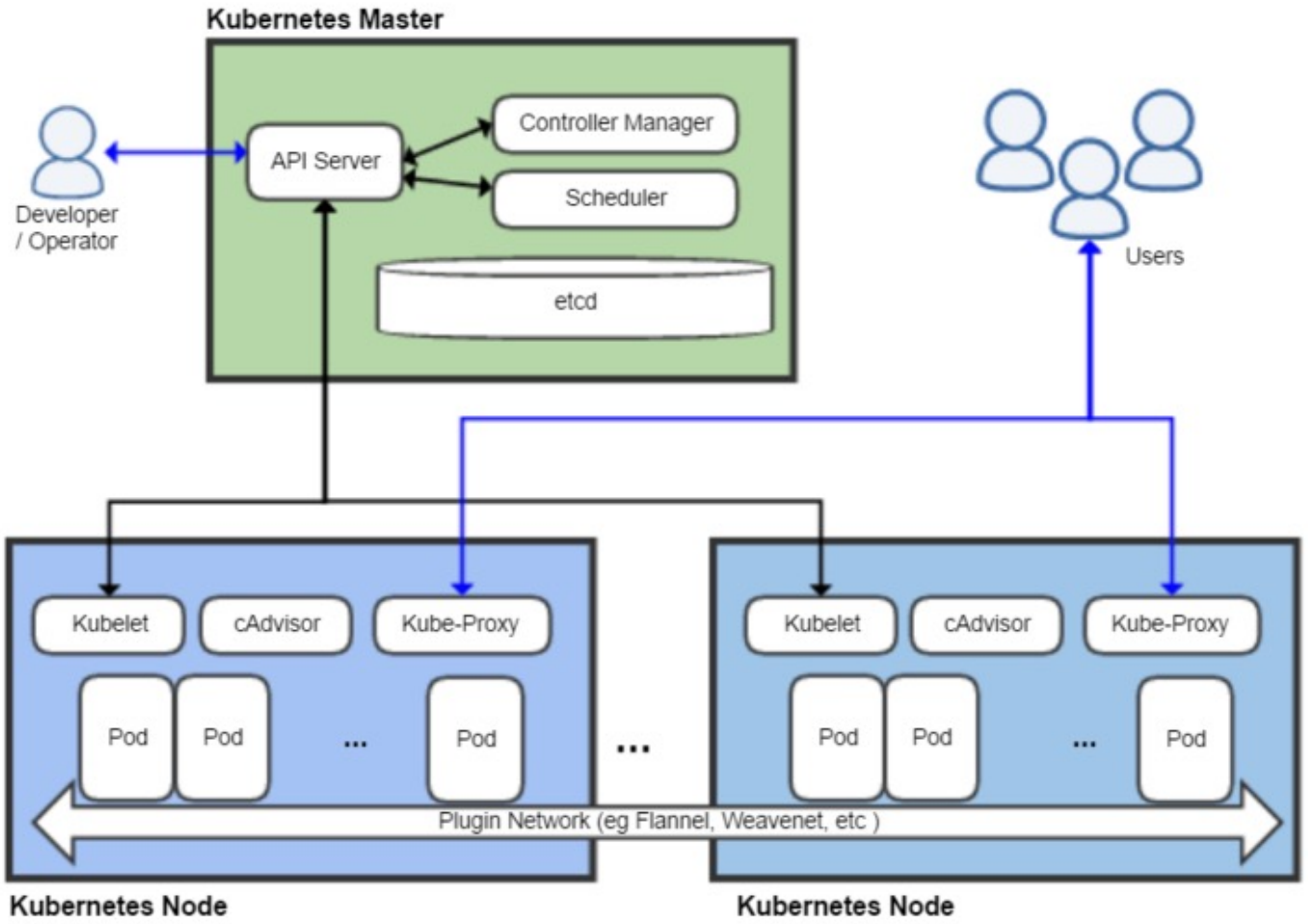
- Deployment and raised automatic container-based services
- Self-scaling and load balancing
- Control of the “health” of each container
- Secrets management in parameters and configurations

Implementing DevSecOps

Kubernetes

Container Platforms

Kubernetes


Kubernetes Master
API Server
Controller Manager
Scheduler
etcd
Developer
/ Operator
Users
Kubelet
cAdvisor
Kube-Proxy
Pod
Pod
...
Pod
...
Kubelet
cAdvisor
Kube-Proxy
Pod
Pod
...
Pod
Plugin Network (eg Flannel, Weavenet, etc)
Kubernetes Node
Kubernetes Node

Key items

Here are some of the capabilities that Kubernetes can provide:

- **Service discovery and load balancing**
- **Storage orchestration**
- **Automatic deployments and rollbacks**
- **Resource management**
- **Secret and configuration management**

Key items

Here are some of the capabilities that Kubernetes can provide:

- **Service discovery and load balancing**
- **Storage orchestration**
- **Automatic deployments and rollbacks**
- **Resource management**
- **Secret and configuration management**

Kubernetes elements

These are some terms that we should understand when we dive deeper into Kubernetes:

- **Cluster**
- **Pod**
- **Replication controller**
- **Services**
- **Labels**

Best Practices

- **Minimalist images:** Docker images get benefits from the point of view of stability, security, and loading time while smaller.
- **Choosing a base image:** The base image can contain many layers and add many capacities.

Managing Docker Images

Managing Containers in Docker

Dockerfile commands

Managing Containers in Docker

Dockerfile

```
docker build --file <Dockerfile_path> --tag <repository>:<tag>
```

Dockerfile

```
FROM ubuntu
RUN apt-get update
RUN apt-get install vim
```

Dockerfile

```
docker build --file <Dockerfile_path> --tag <repository>:<tag>
```

Managing Docker Containers

Managing Containers in Docker

Organizing Docker Images

Managing Containers in Docker

Optimizing Docker Images

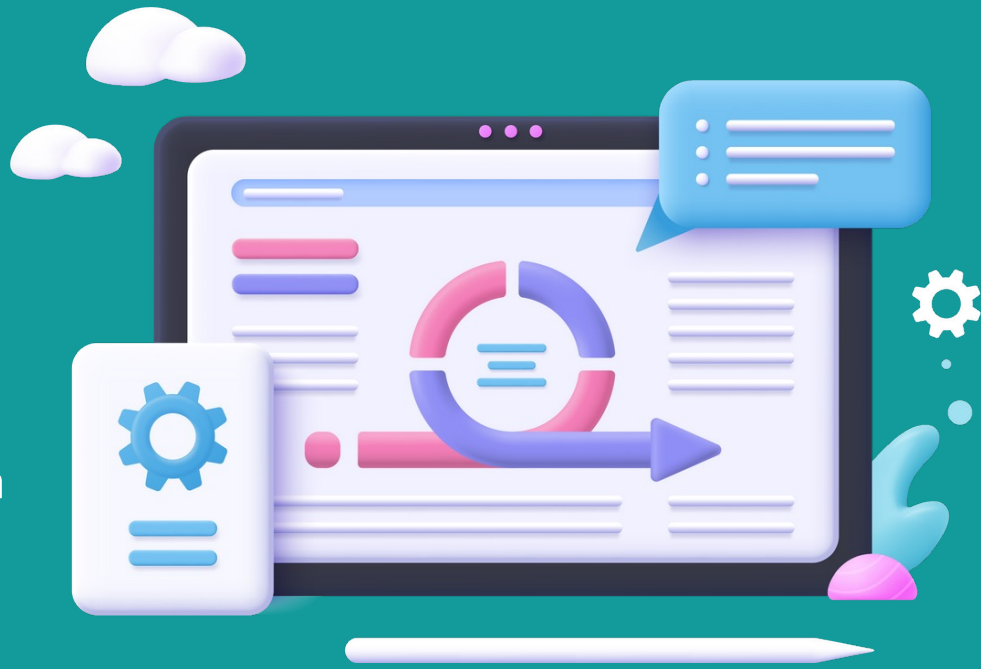
```
# RUN apt-get update -y
# RUN apt-get install -y curl
# RUN apt-get install -y postgresql
# RUN apt-get install -y postgresql-client
# RUN apt-get update -y && \ apt-get install -y curl postgresql postgresql-client
```

Optimizing Docker Images

The construction of a Docker image from a Dockerfile can be an expensive process since it can involve the installation of a large number of libraries.

Note the following aspects about the Docker cache:

- The Docker cache is local, that is, all Dockerfile instructions will be executed if you're building a Dockerfile for the first time on a given machine
- The cache is invalidated if an instruction has changed and you cannot use the cache, and the following Dockerfile instructions will be executed without using the cache.
- The behavior of the ADD and COPY instructions is different in terms of the behavior of the cache.



Docker Security Principles

Docker Security

Security Principals

Linux kernel functions:

- **Namespaces**
- **Control groups**

AppArmor

SELinux

Seccomp

Best Practices

1. Run the daemon Docker process on a dedicated server isolated from other virtual machines
2. Link certain Docker host directories as volumes
3. Use SSL-based authentication
4. Avoid running processes with root privileges inside the containers
   Enable *AppArmor* and *SELinux*, on the Docker host
5. It is important to have the kernel updated with the latest security patches

Best Practices

The following best practices can help create services improving container security:

1. Do not run containers as root and disable SETUID permissions.
2. Use the -cap-drop and - cap-add flags to remove and add capabilities in the container.
3. Don’t use environment variables or run containers in privileged mode if you are going to share secrets.
4. Special care should be taken to keep the Linux Kernel with the latest update.

Docker Capabilities

Docker Security

Docker Capabilities

- CAP_SYSLOG: For modifying the behavior of the Kernel log
- CAP_NET_ADMIN: For modifying the network configuration
- CAP_SYS_MODULE: For managing Kernel modules
- CAP_SYS_RAWIO: For modifying the Kernel memory
- CAP_SYS_NICE: For modifying the priority of the processes
- CAP_SYS_TIME: For modifying the system clock
- CAP_SYS_TTY_CONFIG: For configuring tty devices
- CAP_AUDIT_CONTROL: For configuring the audit subsystem

Docker Capabilities

The Linux libcap packages incorporate commands and binaries for listing and managing capabilities:

- getcap: Allows listing the capabilities of a file
- setcap: Allows assigning and deleting the capabilities of a file
- getpcaps:Allows listing the capabilities of a process
- capsh: Provides a command line interface for testing and exploring

Docker Capabilities

The Linux libcap packages incorporate those managing capabilities:

- chown
- dac_override
- Fowner
- kill
- Setgid
- Setuid
- Setpcap
- net_bind_service
- net_raw
- sys_
- chroot mknod
- setfcap
- audit_write.

Docker Content Trust

Docker Security

Content Trust

Docker Content Trust (**DCT**) is a mechanism that allows developers to sign their content, completing the reliable distribution mechanism.

export DOCKER_CONTENT_TRUST=1

Docker Content Trust can protect against certain attack scenarios, including:

- **Protection of malicious code in images**
- **Protection against repeated attacks**
- **Protection against key commitments**

Docker Registry

Docker Security

Docker Registry

Docker provides a software distribution mechanism, also known as “Registry”, which facilitates the discovery and distribution of Docker images.

Docker Capabilities

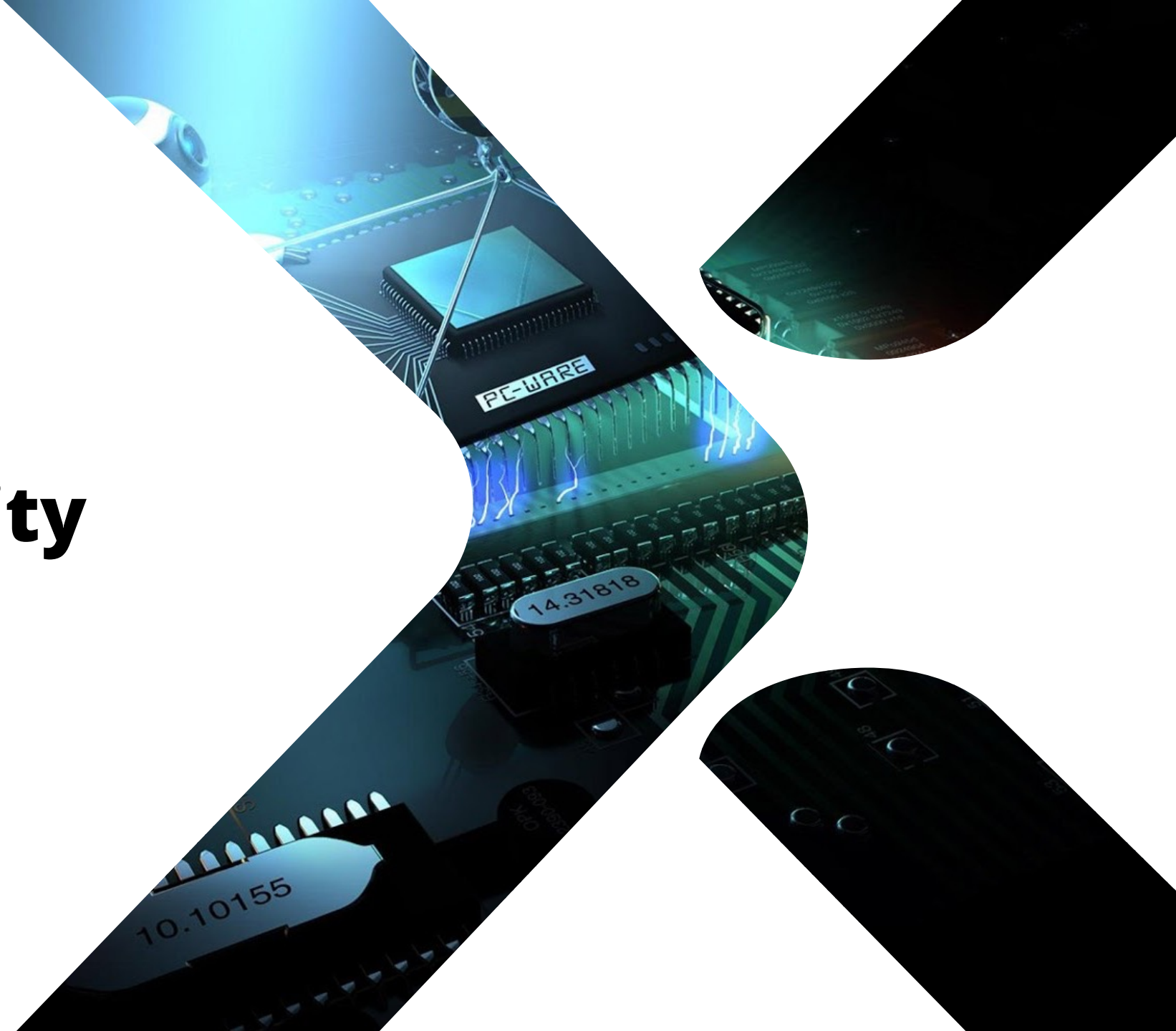
The Linux libcap packages incorporate commands and binaries for listing and managing capabilities:

- getcap: Allows listing the capabilities of a file
- setcap: Allows assigning and deleting the capabilities of a file
- getpcaps:Allows listing the capabilities of a process
- capsh: Provides a command line interface for testing and exploring

Docker Capabilities

The Linux libcap packages incorporate commands and binaries for listing and managing capabilities:

- getcap: Allows listing the capabilities of a file
- setcap: Allows assigning and deleting the capabilities of a file
- getpcaps:Allows listing the capabilities of a process
- capsh: Provides a command line interface for testing and exploring



Daemon Security

Docker Host Security

Daemon Security

AppArmor and Seccomp profiles, which provide kernel-enhancement features to limit system calls.

Docker bench security and Lynis, which follow security best practices in the Docker environment

Daemon Security

http://

dockerd

Daemon Security

File/folder	User:group	Permissions
docker.service	root:root	644(rw-r--r--)
/etc/docker	root:root	755(rwxr-xr-x)
/etc/default/docker	root:root	644(rw-r--r--)
Docker registration certificate	root:root	444(r--r--r--)

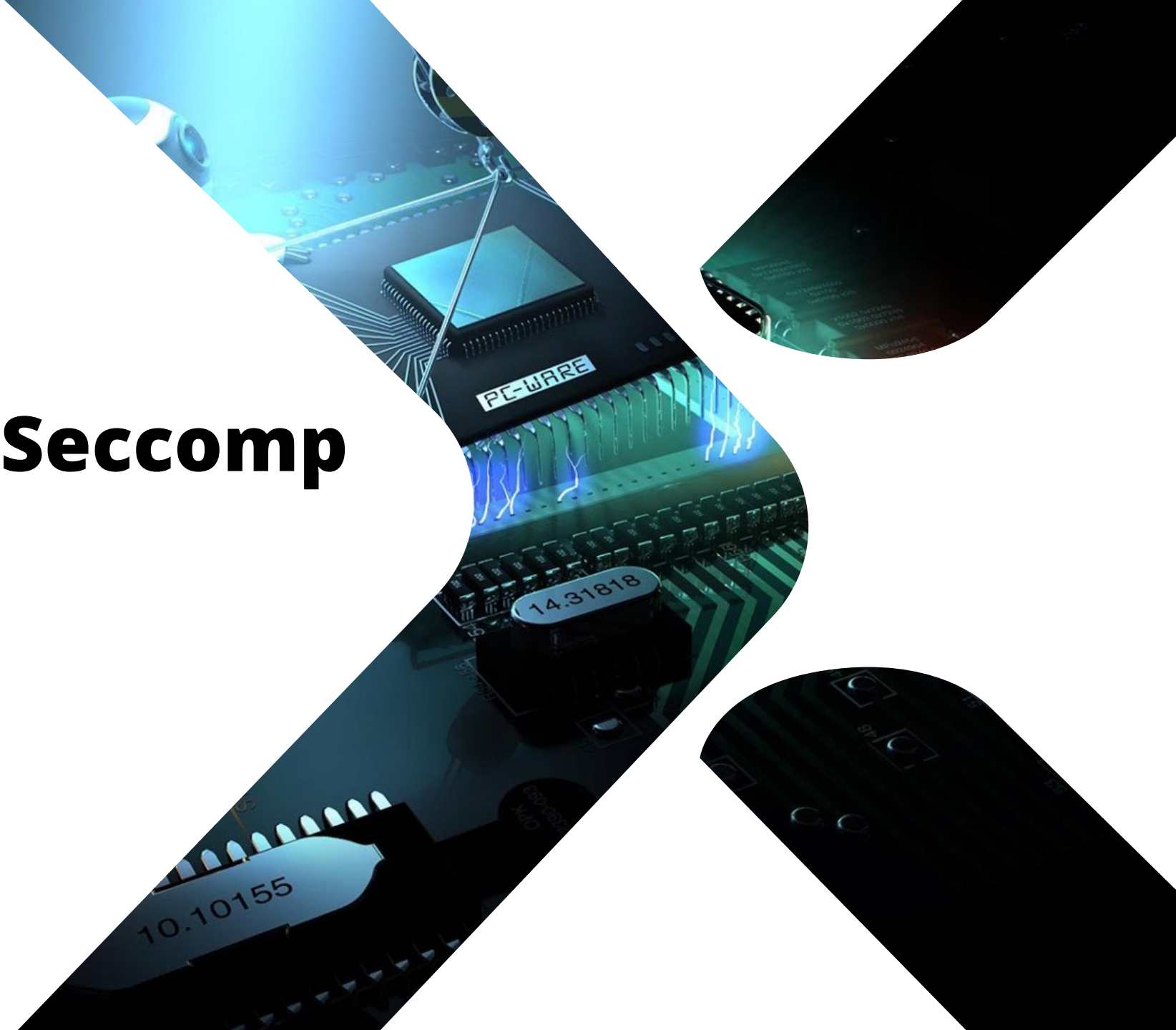
Auditing files

The Linux audit daemon framework has the following features:

- Audit processes and file modification
- Monitor system calls
- Detecting intrusions
- Register commands by users

auditd.conf

audit.rules



Apparmor and Seccomp Profiles

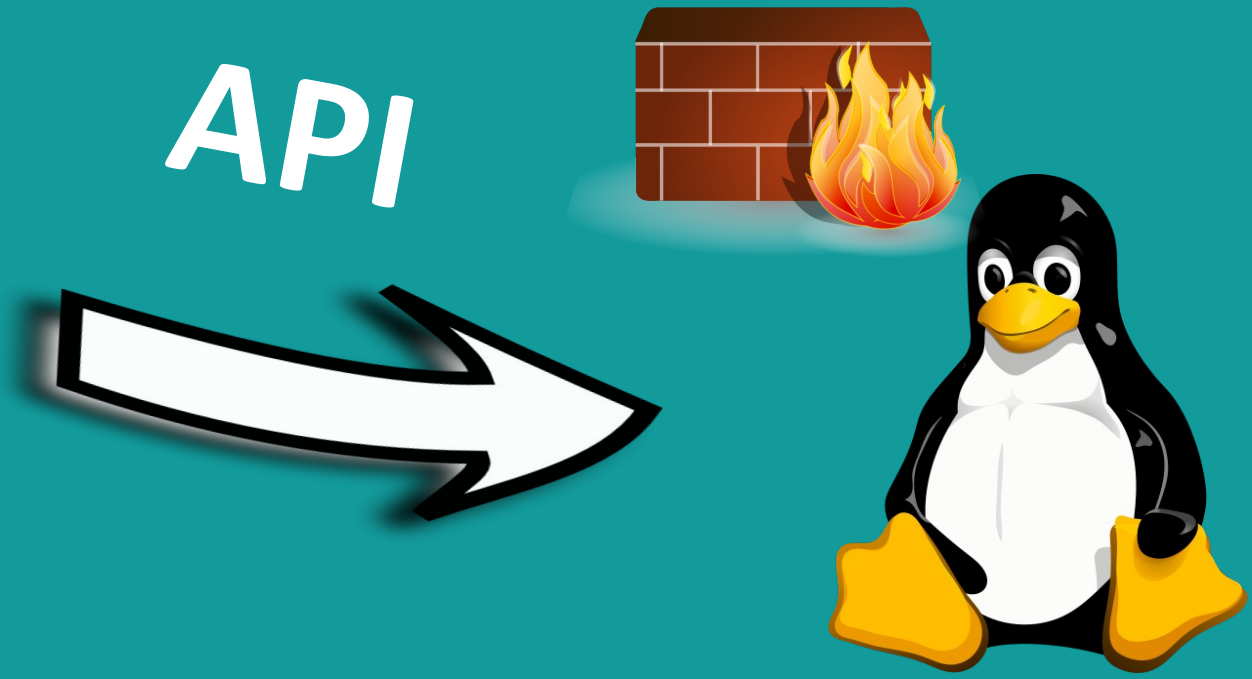
Docker Host Security

Apparmor

Process

Seccomp

API



Docker Bench Security

Docker Host Security

Docker Bench Security

Docker bench security perform checks against a container and generates security report

- **Host configuration**
- **The Daemon Docker configuration**
- **Docker daemon configuration files**
- **Image container and compilation files**
- **Runtime container**
- **Docker security operations**

https://www.cisecurity.org/benchmark/docker/

Docker Bench Security

Host configuration

Daemon Docker configuration

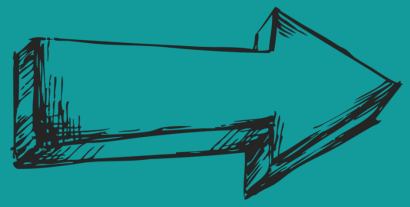
Docker daemon configuration files

Daemon Docker configuration

Docker Hub Security Scanning

Docker Images Security

Docker Images Security



- Clair
- Anchore

Docker Hub Images

Official Images

- Maintained by main suppliers such as Apache, Ngnix, Python etc.

User Images

- Created by users and customized according their needs for a project.

Docker Security Scanning

Common Vulnerabilities and Exposures

- Static analysis depth and integrity
- Vulnerability feeds quality

Docker Security Scanning Process

- Public
- Private

Docker Security Scanning Process

Common Vulnerability Score System (CVSS)

- High: Vulnerability has a score within the range [8-10]
- Medium: Vulnerability has a score within the range [4-7.9]
- Low: Vulnerability has a score within the range [0.0-3.9]

Docker Security Scanning Process

Dockerfile

```
FROM maven:3.6-jdk-8-alpine
WORKDIR /app
COPY pom.xml .
RUN mvn -e -B dependency:resolve
COPY src ./src
RUN mvn -e -B package
CMD ["java", "-jar", "/app/app.jar"]
```

Registry

Open-Source Tools

- Source Code Control
- CI/CD Tools
- JFrog Xray

Clair Security Scanning

Vulnerability Analysis

https://github.com/quay/clair

main · Go to file · Code

dependabot[bot] and crozzy build(deps): bump pe... · 8 days ago · 1,576

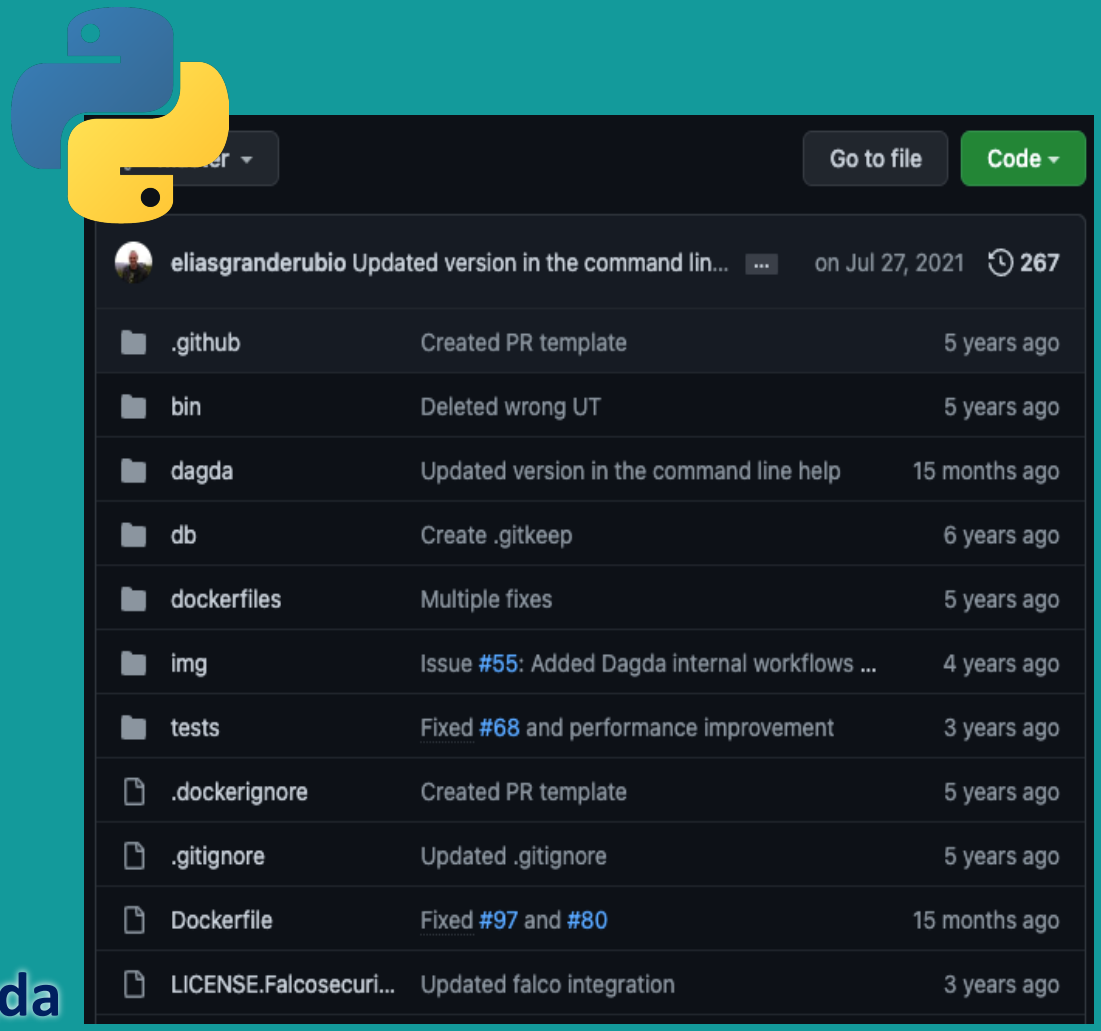
.chglog	chore: update changelog to cope with subm...	11 months ago
.github	build(deps): bump peter-evans/create-pull-...	8 days ago
Documentation	all: remove Quay keyserver support	4 months ago
clair-error	notifier: log failed delivery reason	2 years ago
cmd	config: Don't use flag default combo	2 months ago
config	all: remove Quay keyserver support	4 months ago
contrib/openshift	contrib: remove rpmscanner files on startup	2 months ago
health	introspection: enable readiness endpoint	17 months ago
httptransport	indexer: Return 4XX status code when Inde...	2 months ago
indexer	indexer: add DeleteManifests method	9 months ago
initialize	services: update initialization	3 months ago

Dagda Security Scanning

Vulnerabilities Database

MongoDB

Packages
Dependencies
Modules

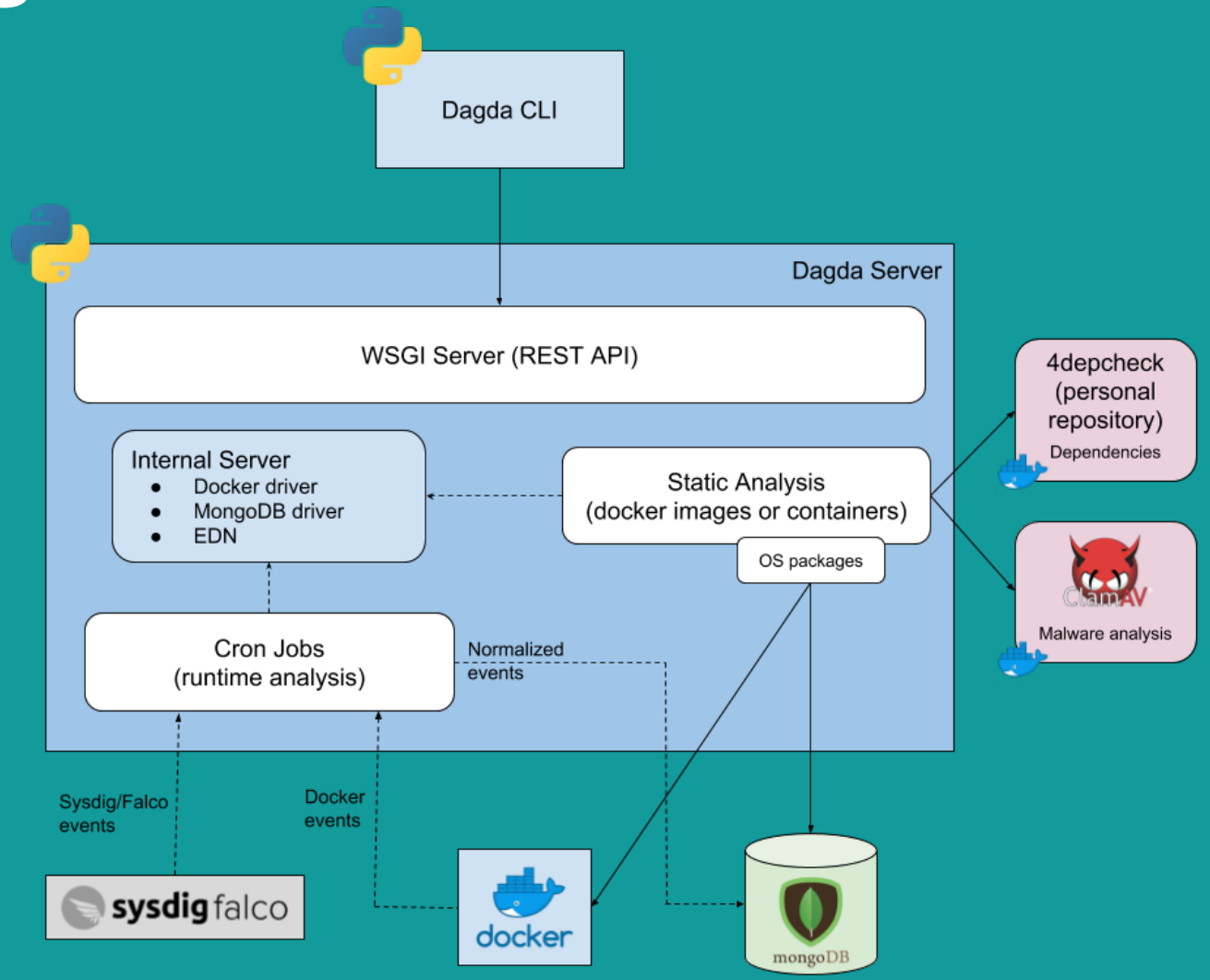


https://github.com/eliasgranderubio/dagda

Dagda Security Scanning

Dagda Docker base Linux images:

- Red Hat/CentOS/Fedora
- Debian/Ubuntu
- OpenSUSE
- Alpine Linux



OWASP Dependency Check

pom.xml (Book)

```
        <groupId>com.example.maven</groupId>
        <version>1.0-SNAPSHOT</version>
    </parent>
    <modelVersion>4.0.0</modelVersion>
    <artifactId>Book</artifactId>
    <dependencies>
        <dependency...>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit-dep</artifactId>
            <version>4.10</version>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot</artifactId>
            <version>2.1.7.RELEASE</version>
        </dependency>
    </dependencies>
</project>
```

Pom.xml **Package.json**

Trivy

Vulnerability Check

Library

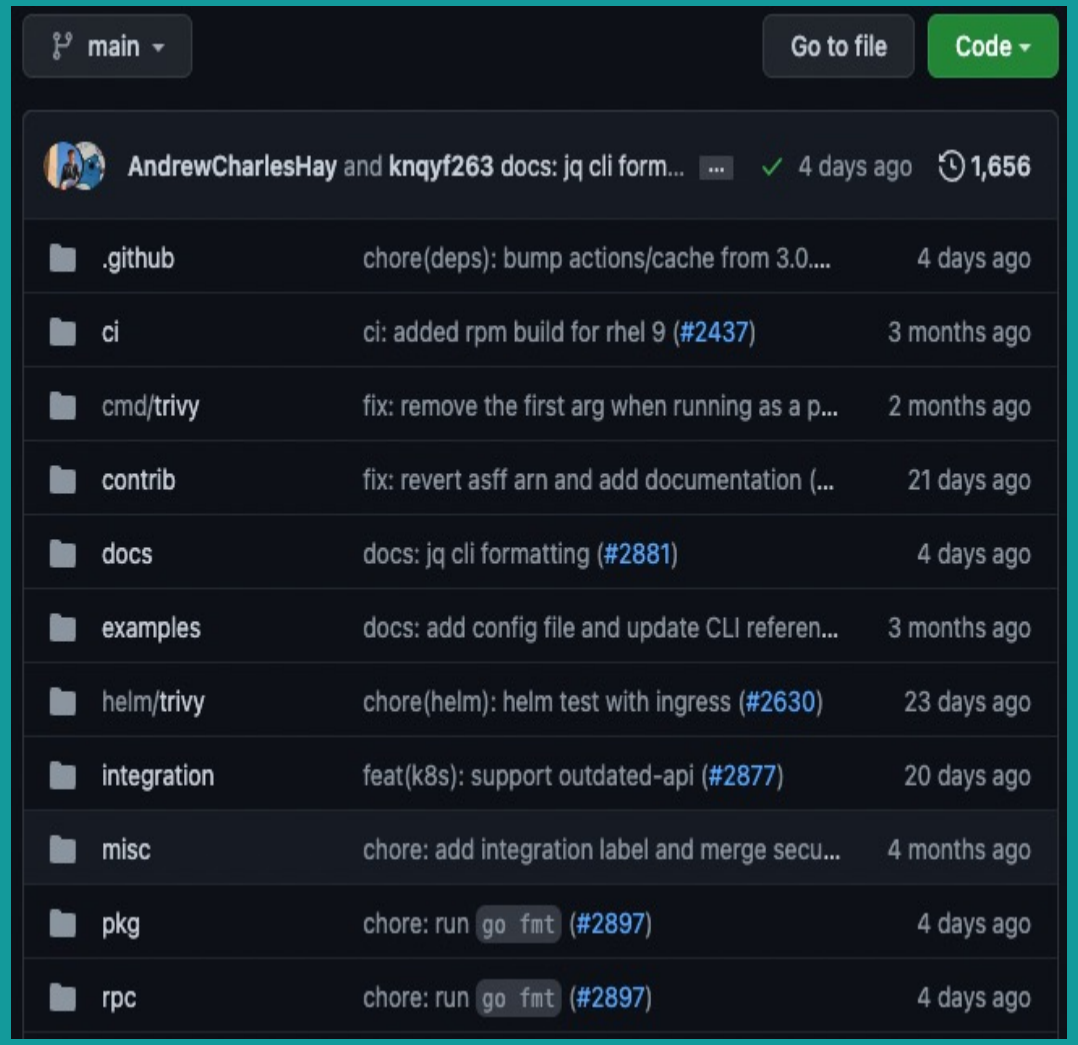
Vulnerability ID

Severity

- Critical (score 9.0-10.0)
- High (score 7.0-8.9)
- Medium (score 4.0-6.9)
- Low (score 0.1-3.9)

Installed version

Fixed version

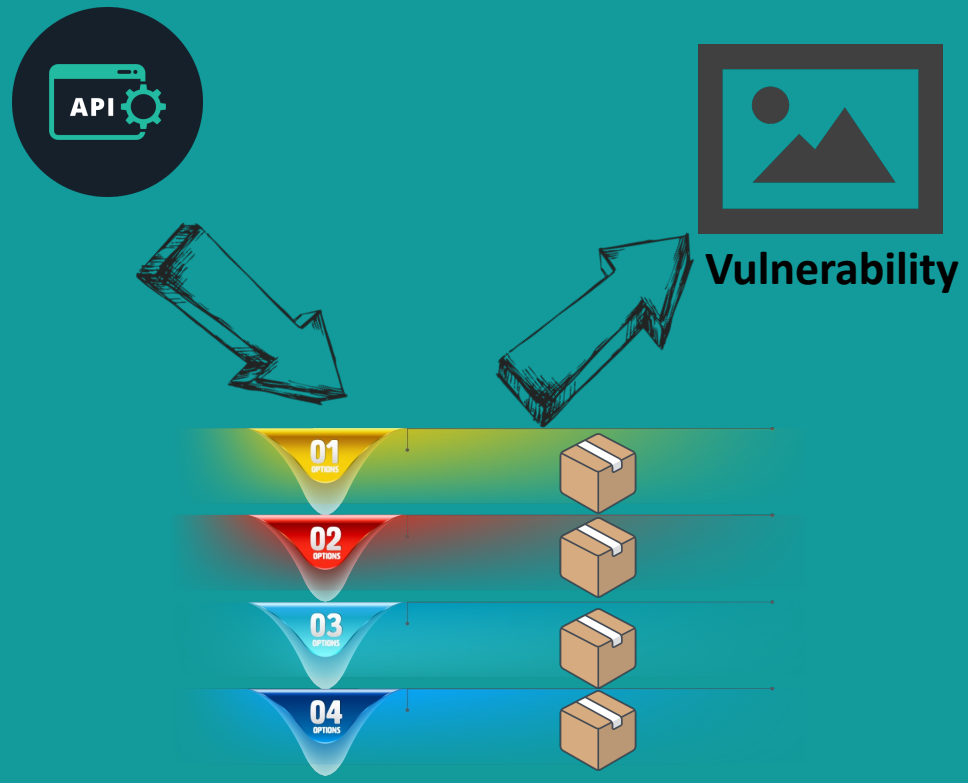


https://github.com/aquasecurity/trivy

Scanning with Clair and Quay

Docker Images Security

Scanning with Clair and Quay



https://github.com/quay/clair#docker-compose

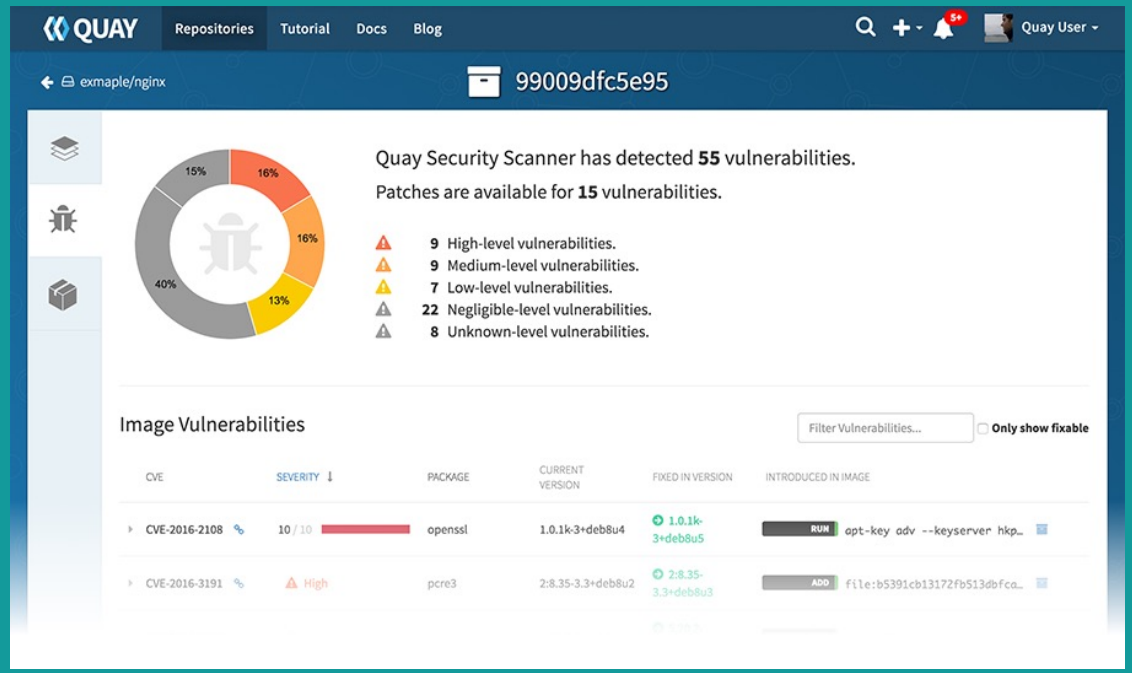
main · Go to file · Code

dependabot[bot] and crozzy build(deps): bump pe... · 8 days ago · 1,576

.chglog	chore: update changelog to cope with subm...	11 months ago
.github	build(deps): bump peter-evans/create-pull-...	8 days ago
Documentation	all: remove Quay keyserver support	4 months ago
clair-error	notifier: log failed delivery reason	2 years ago
cmd	config: Don't use flag default combo	2 months ago
config	all: remove Quay keyserver support	4 months ago
contrib/openshift	contrib: remove rpmscanner files on startup	2 months ago
health	introspection: enable readiness endpoint	17 months ago
httptransport	indexer: Return 4XX status code when Inde...	2 months ago
indexer	indexer: add DeleteManifests method	9 months ago
initialize	services: update initialization	3 months ago
internal	all: move config package to new module	11 months ago

Quay Security Scanning

- Access complexity
- Authentication
- Confidentiality Impact
- Integrity Impact

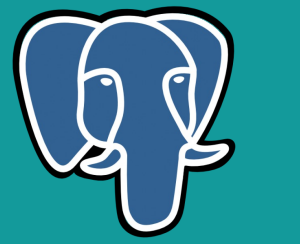


Analyzing Images with Anchore

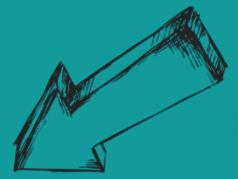
Docker Images Security

Anchore

- Open source
- Inspects and analyzes Docker images



Artifacts
Packages
Image files
Vulnerabilities



Logs

Anchore Components

anchore

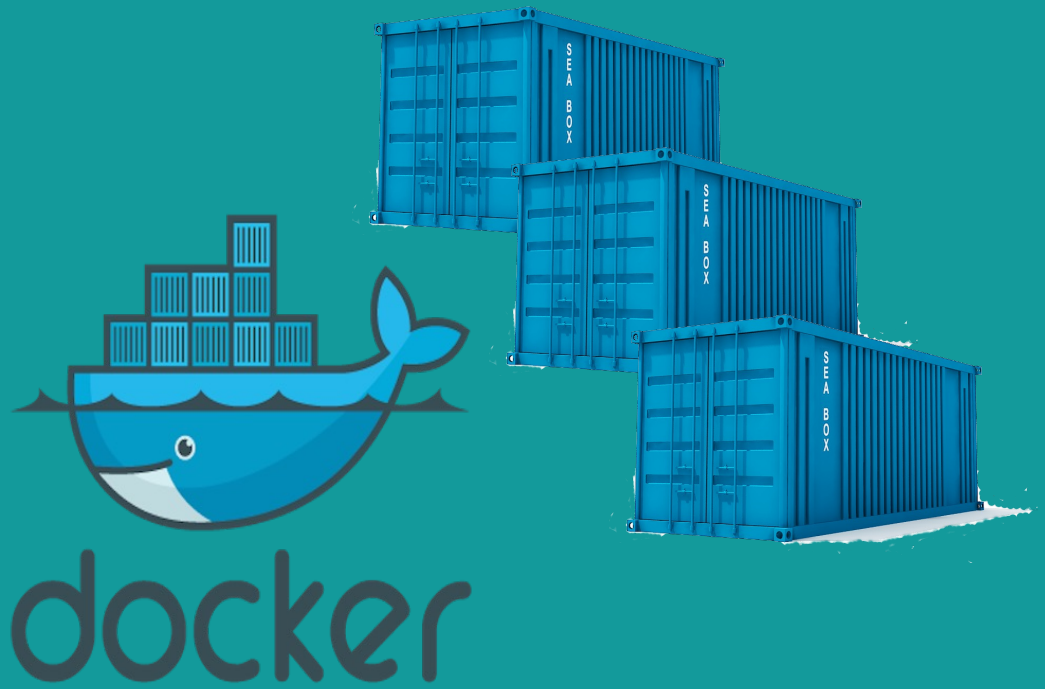
- Anchore Engine CLI
- Anchore Engine API
- Anchore Policy Engine
- Anchore Enginer Analyzer
- Anchore Engine Database

- Extract packages and components from Docker images
- Scan images for known vulnerabilities

Docker threats and attacks

Auditing Vulnerabilities in Docker Containers

Scanning with Clair and Quay



Kernel Attacks

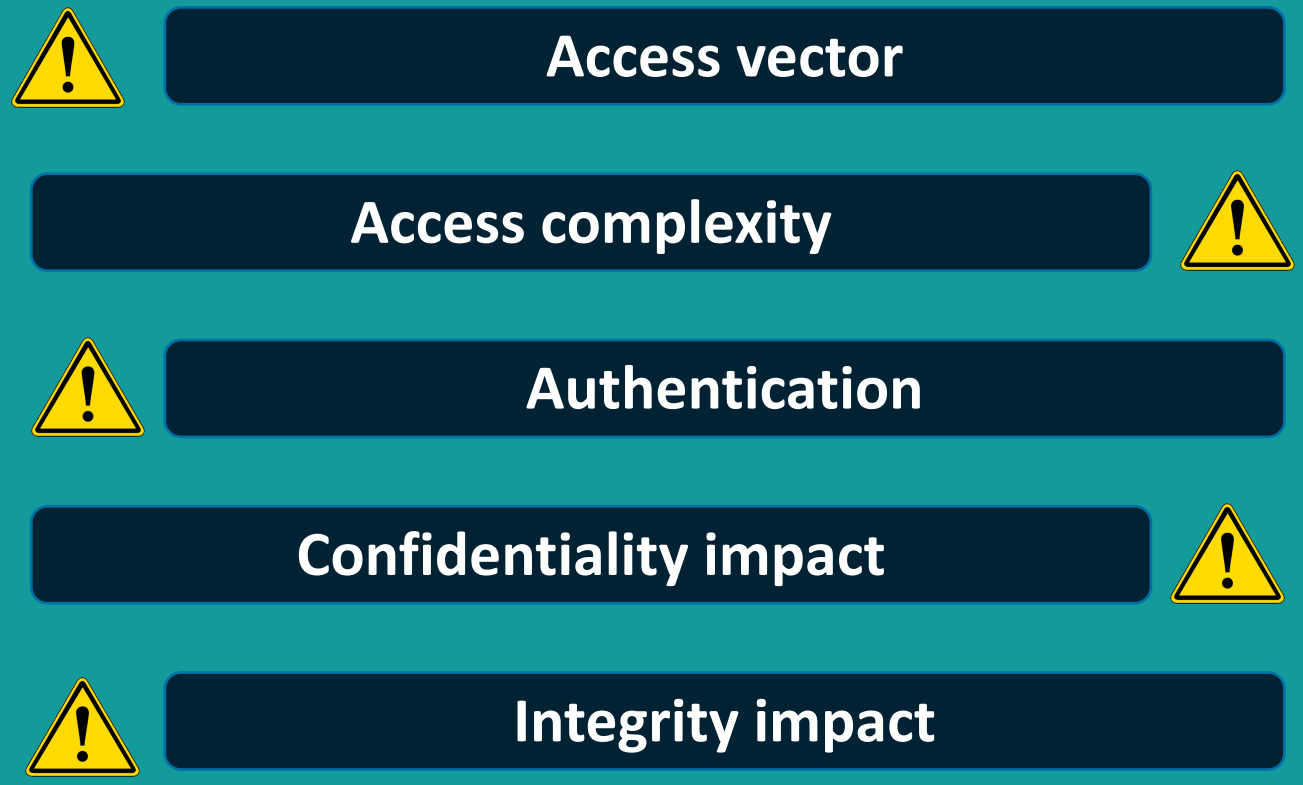
DoS Attacks

Trojan images

Docker Vulnerabilities

- Containers that attempt to download additional malware
- The *Dirty Cow* exploits in the Linux kernel allows root privilege escalation
- Ransomware attacks on insecure server containers by *MongoDB* and *ElasticSearch* containers
- Buffer overflow vulnerability in specific programming language libraries
- SQL injection attacks allow you to take control of a database container in order to steal data

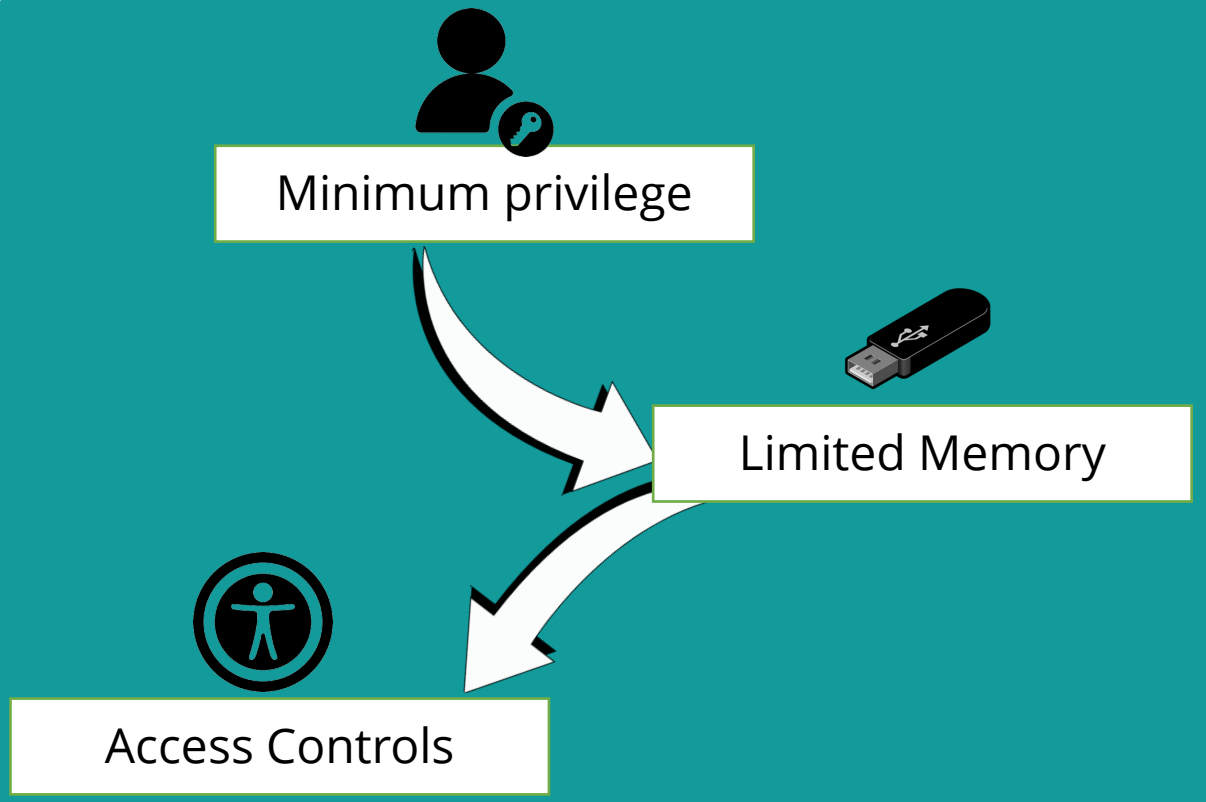
Vulnerability Components



Analyzing Vulnerabilities in Docker Images

Auditing Vulnerabilities in Docker Containers

Analyzing Vulnerabilities



Vulnerability Classification

High-Capacity Vulnerability

Medium-Capacity Vulnerability

Medium-Capacity Vulnerability

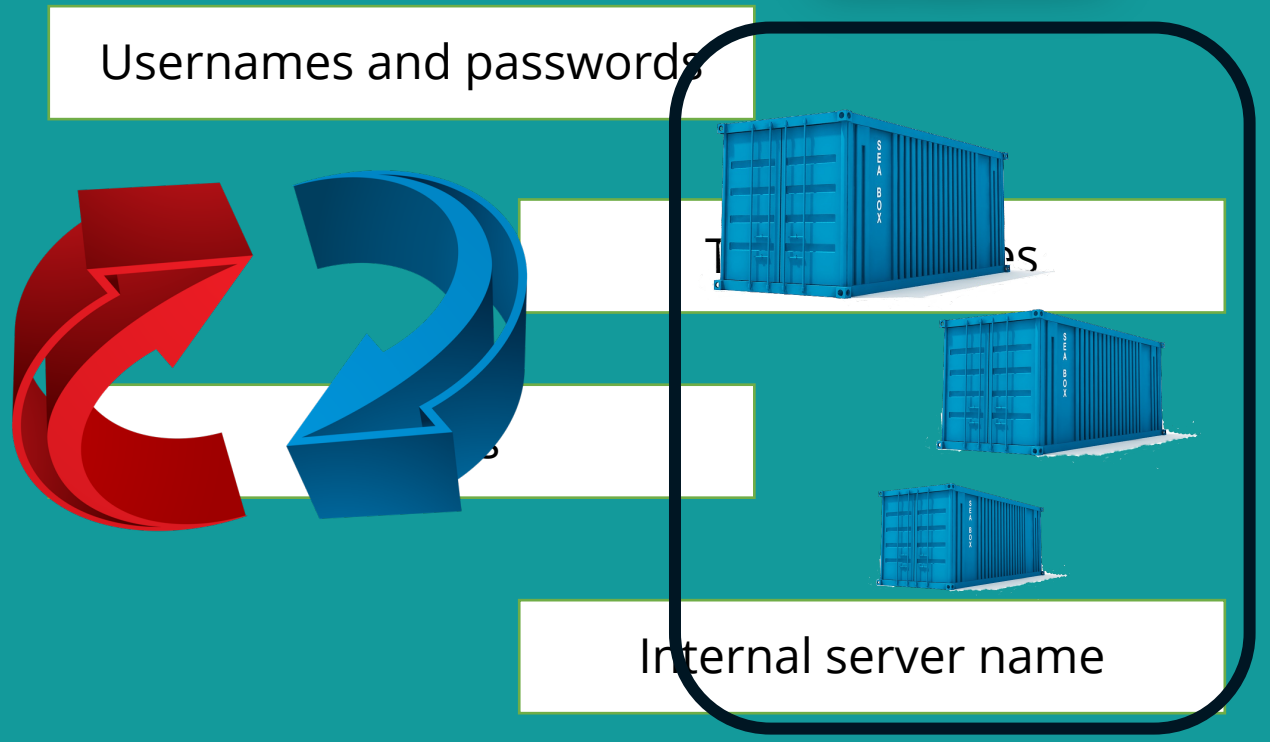
Managing secrets in Docker

Docker Secrets and Networks

Managing Secrets

Credentials

Containers



Managing Secrets in Docker

/run/secrets/<secret_name>

Host 1

Host 2

Host 3

Add

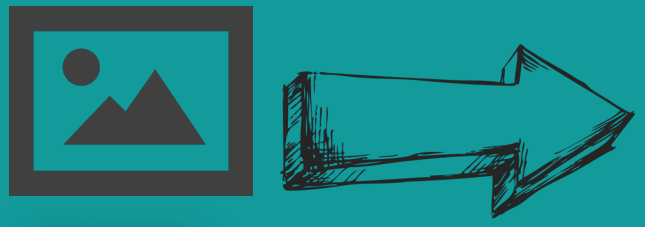
Copy

SWARM Service

Container Networking

Docker Secrets and Networks

Container Networking



Image

Stack

--net Options

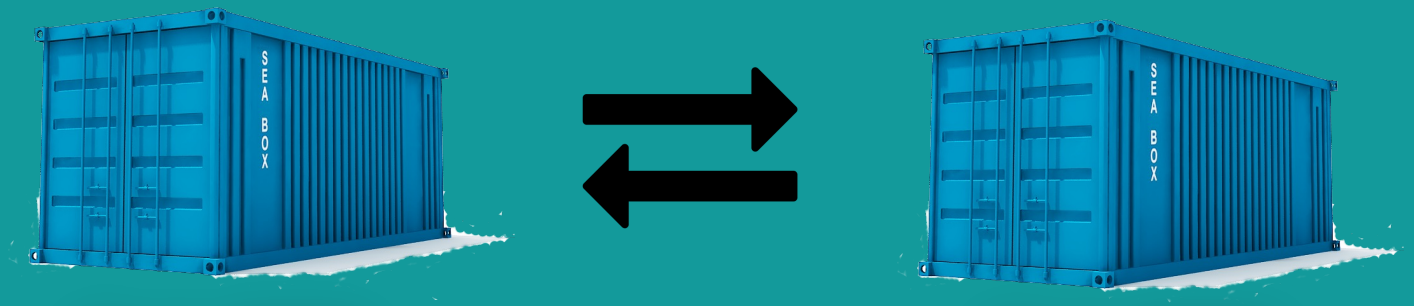
--net = bridge
--net = none
--net = host
--net = mycontainer

Port Mapping

Docker Secrets and Networks

Container Networking

- --dns
- --dns-search
- -h
- --link
- --expose
- --publish



Port: 8080

Port: 8080

Ports Mapping

Port forwarding is the easiest way to expose the services that are running in containers.

```
$ docker run -p ip:host_port: container_port
$ docker run -p ip::container_port
$ docker run -p host_port:container_port
```

Linked Containers

Runtime exposure

Managing Docker Networks

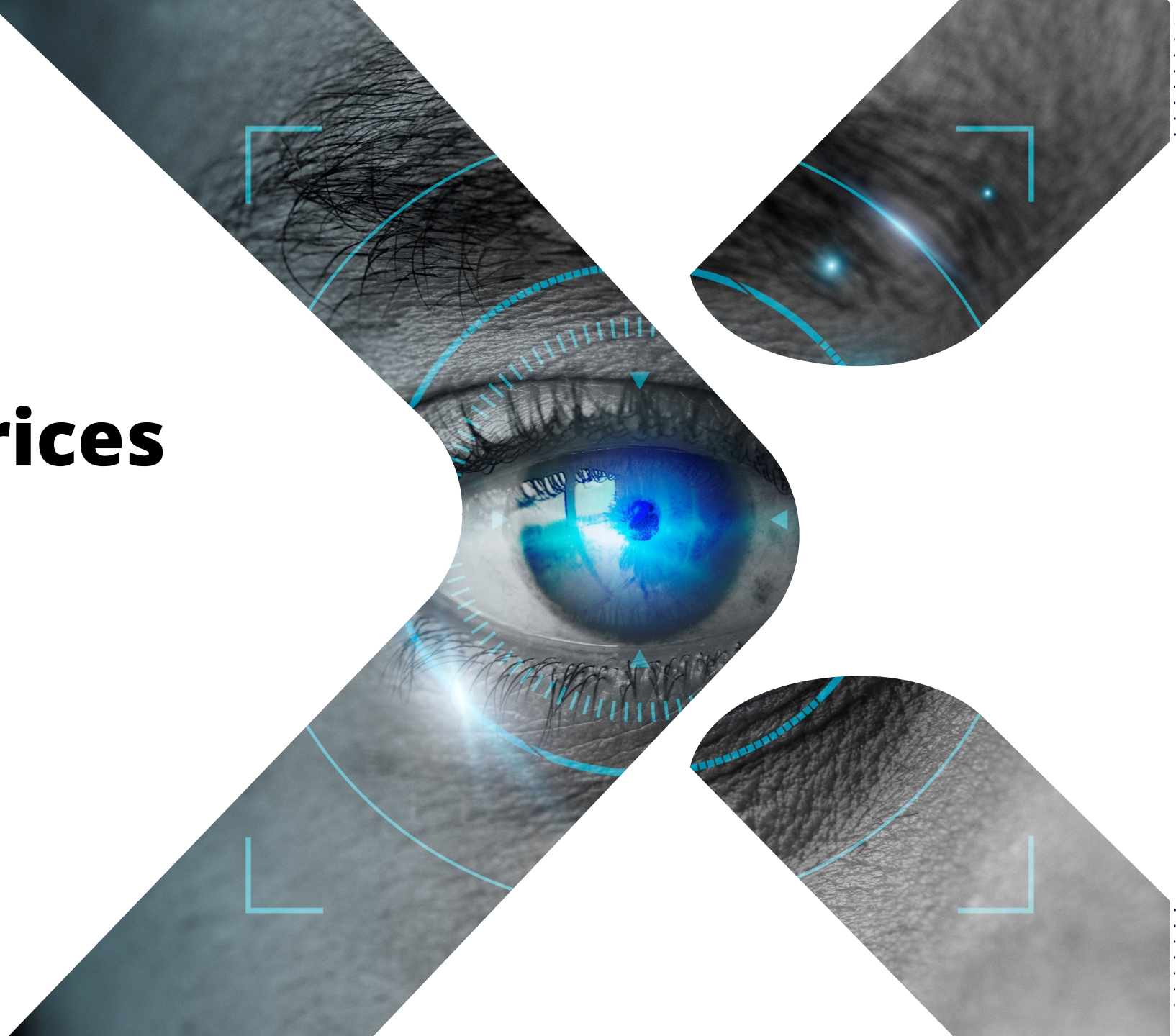
Docker Secrets and Networks

Managing Docker Networks

Project 1

Project 2

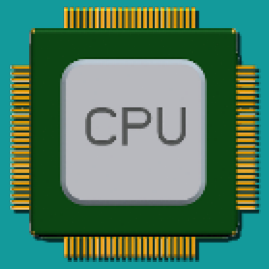
Project 3



Container Matrices and Events

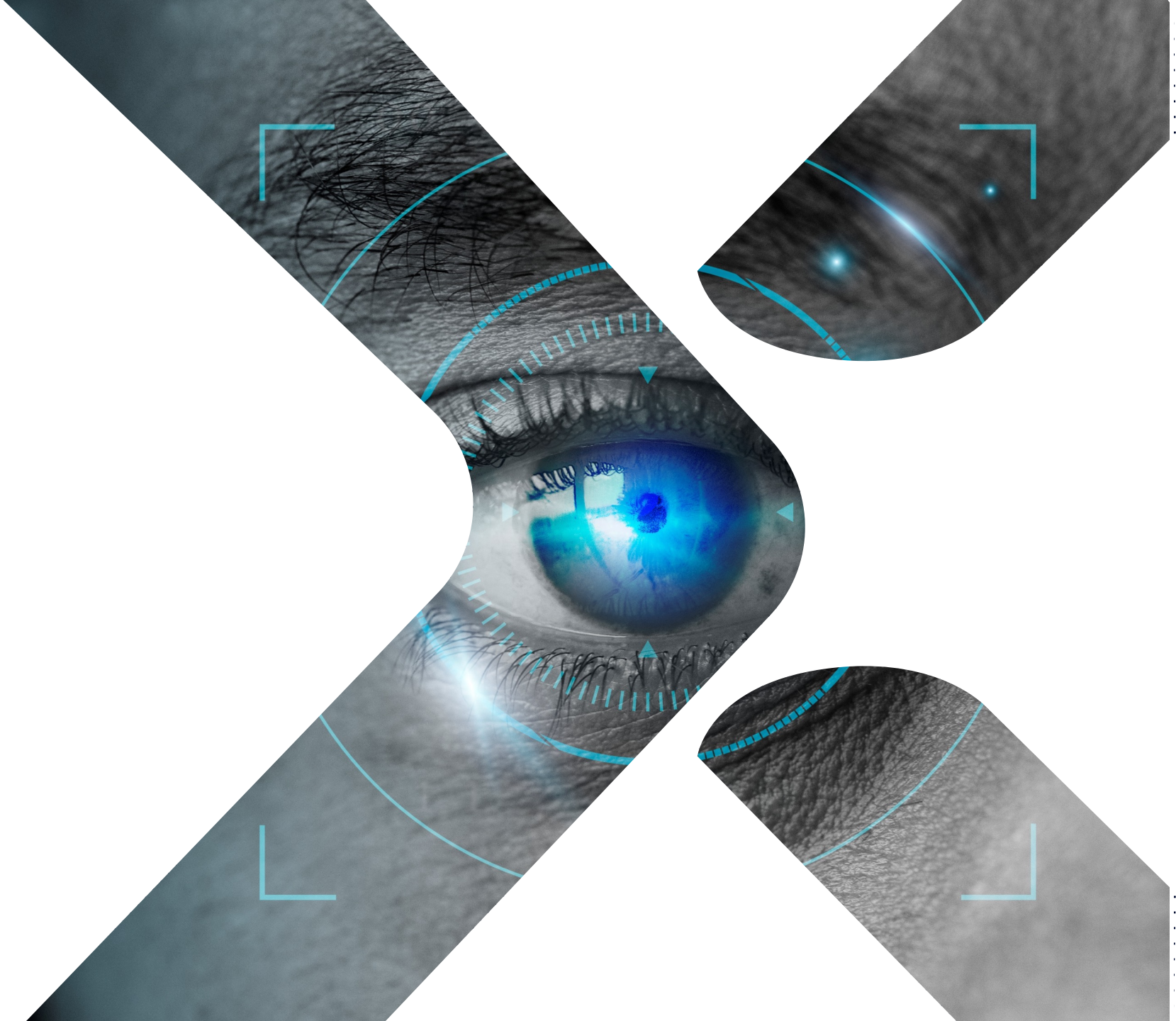
Docker Container Monitoring

Container Matrices and Events




```
2022/10/06 20:48:18 [notice] 1#1: start worke
2022/10/06 20:48:18 [notice] 1#1: start worker process 31
2022/10/06 20:48:18 [notice] 1#1: start worker process 32
2022/10/06 20:48:18 [notice] 1#1: start worker process 33
2022/10/06 20:48:18 [notice] 1#1: start worker process 34
```

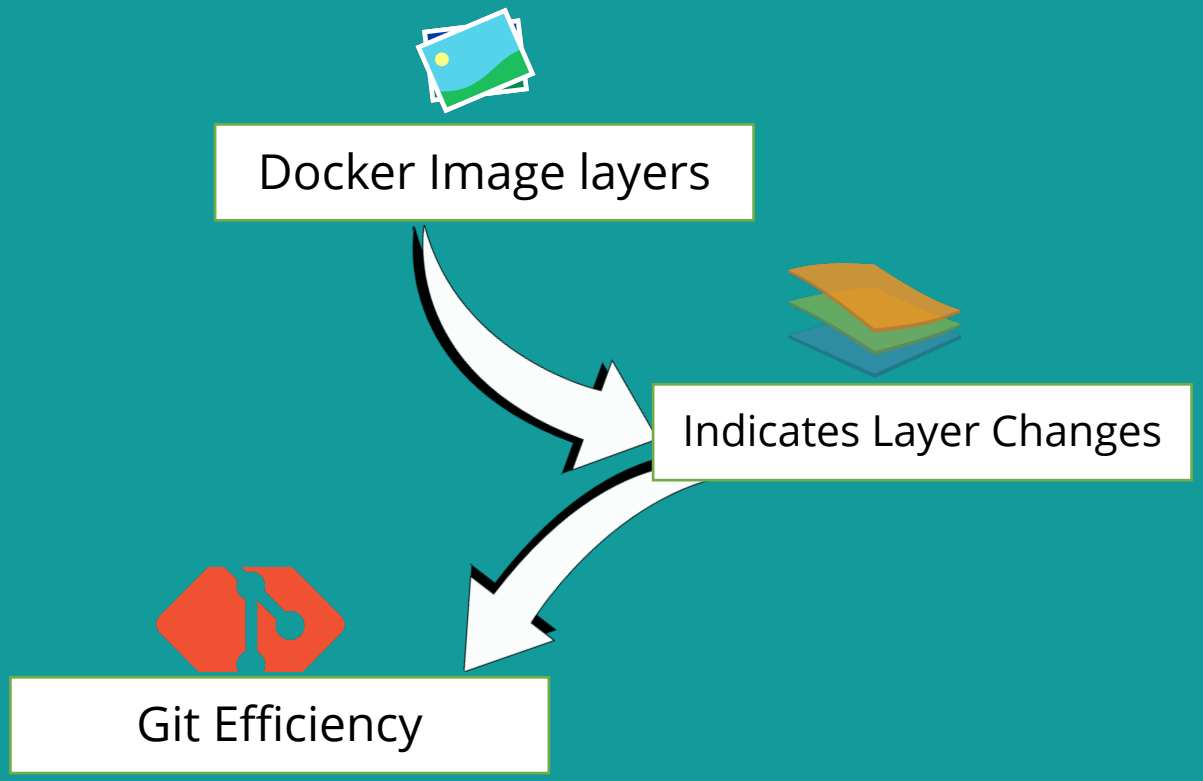
Container Matrices and Events



Performance Monitoring

Docker Container Monitoring

Performance Monitoring with Dive



https://github.com/wagoodman/dive

master · Go to file · Code

wagoodman remove meetup callout ... ✓ on Jul 2, 2021 412

.circleci	Bring Go current for CircleCI	2 years ago
.data	add ci integration test with kaniko image	2 years ago
.github	Bring Go current for GH build	2 years ago
.scripts	replace travis with gitlab; linting fixes	3 years ago
cmd	Add wrapping tree key	2 years ago
dive	docker-archive: Add support for kaniko	2 years ago
runtime	Merge branch 'master' into wrap-tree	2 years ago
utils	Update gocui module	2 years ago

Implementing DevSecOps

Container Administration with Portainer

Docker Container Administration

Administration with Portainer

Lack of control

Production

Scale containers

Vulnerability

Requirements

- **Development**: application codes and libraries.
- **Application release**: managers coordinate the automation of application environments
- **IT operations**: the containers are deployed in production

Business

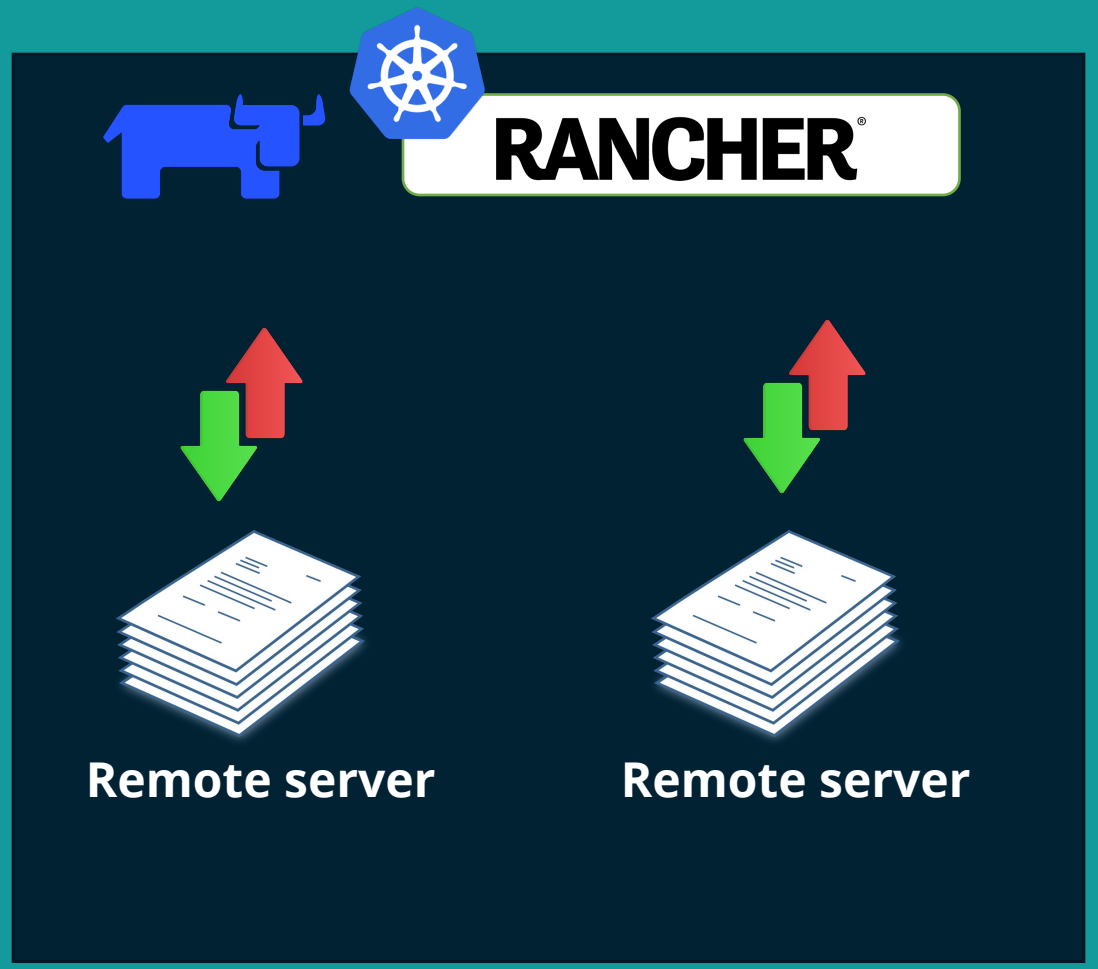
Implementing DevSecOps

Container Administration with Rancher

Docker Container Administration

Administration with Rancher

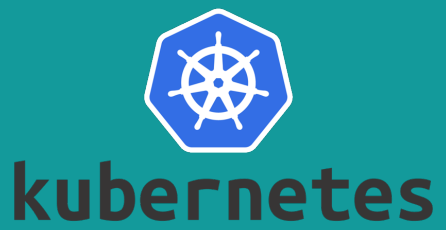
- It allows you to create as many **environments** as you need.
- It allows you to select the container orchestrator.
- There is a public catalog called **Rancher Community.**
- It makes **single-cluster and multi-cluster** deployments easy
- It facilitates cluster provisioning



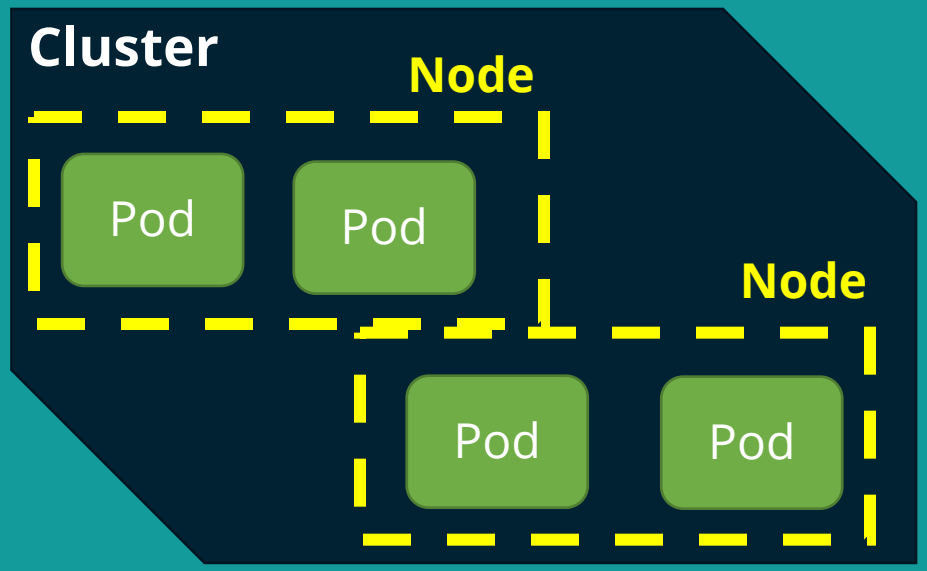
Kubernetes Architecture

Kubernetes Architecture

Kubernetes architecture



- Avoids “vendor lock-in
- Manages container clusters
- Performs container monitoring



Secrets

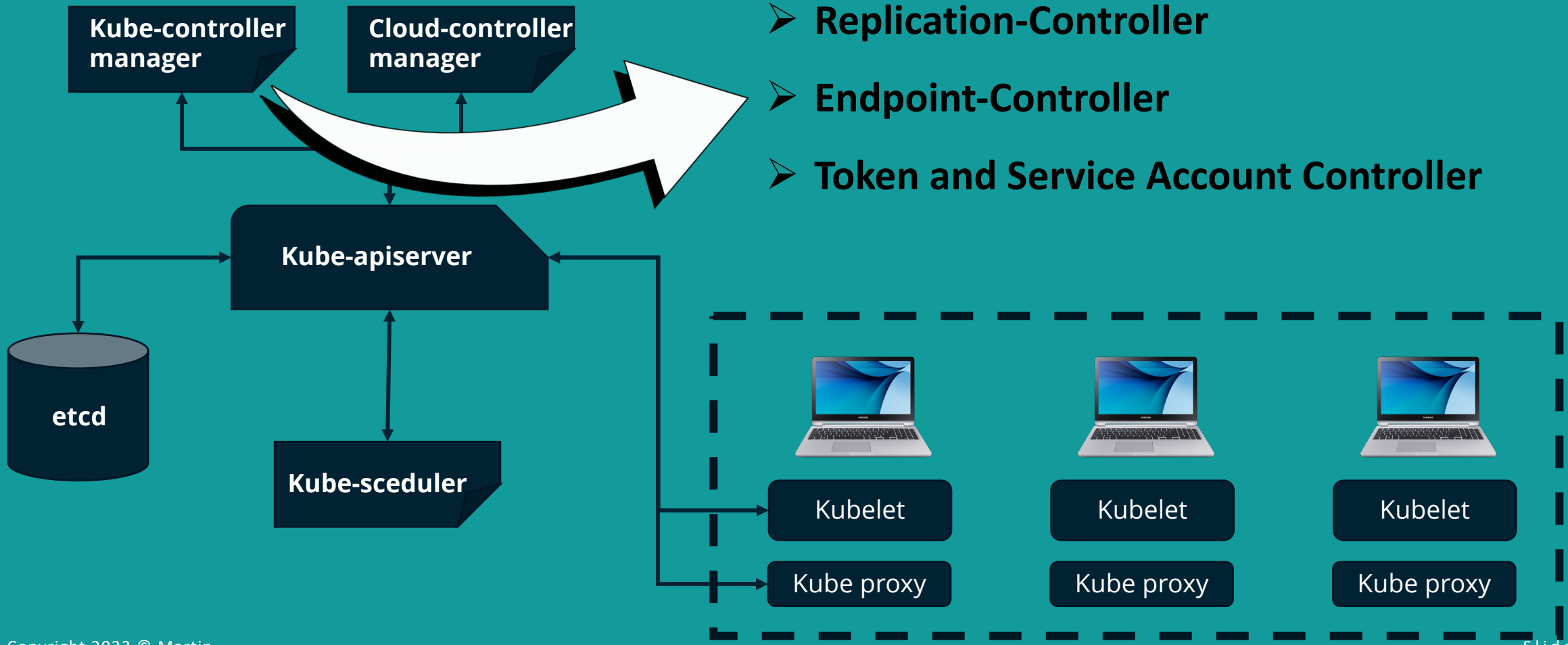
Scaling

Regeneration

Load balance

Deployment

Kubernetes Architecture

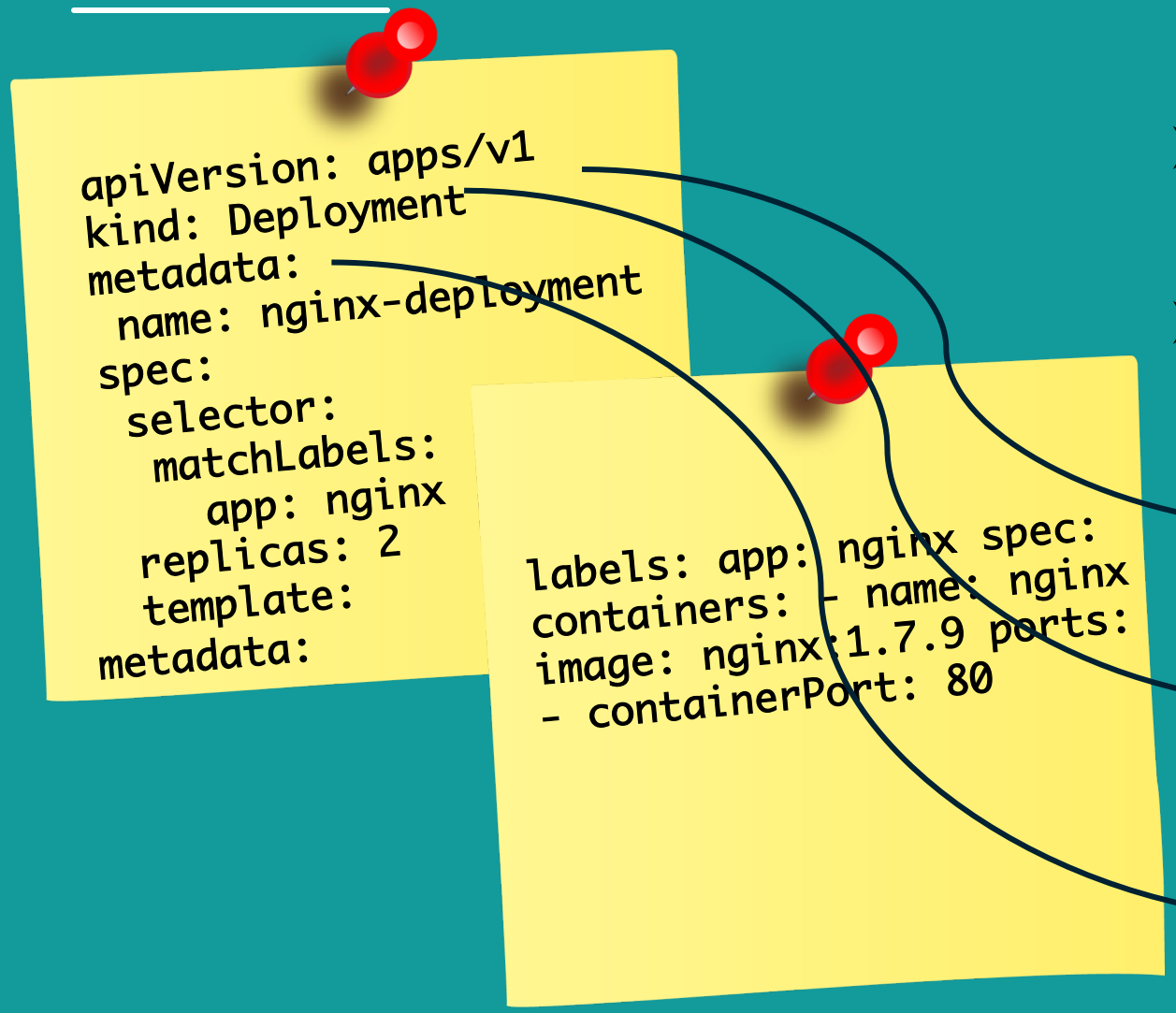


- Node-Controller
- Replication-Controller
- Endpoint-Controller
- Token and Service Account Controller

Kubernetes Objects

Kubernetes Architecture

Kubernetes Objects



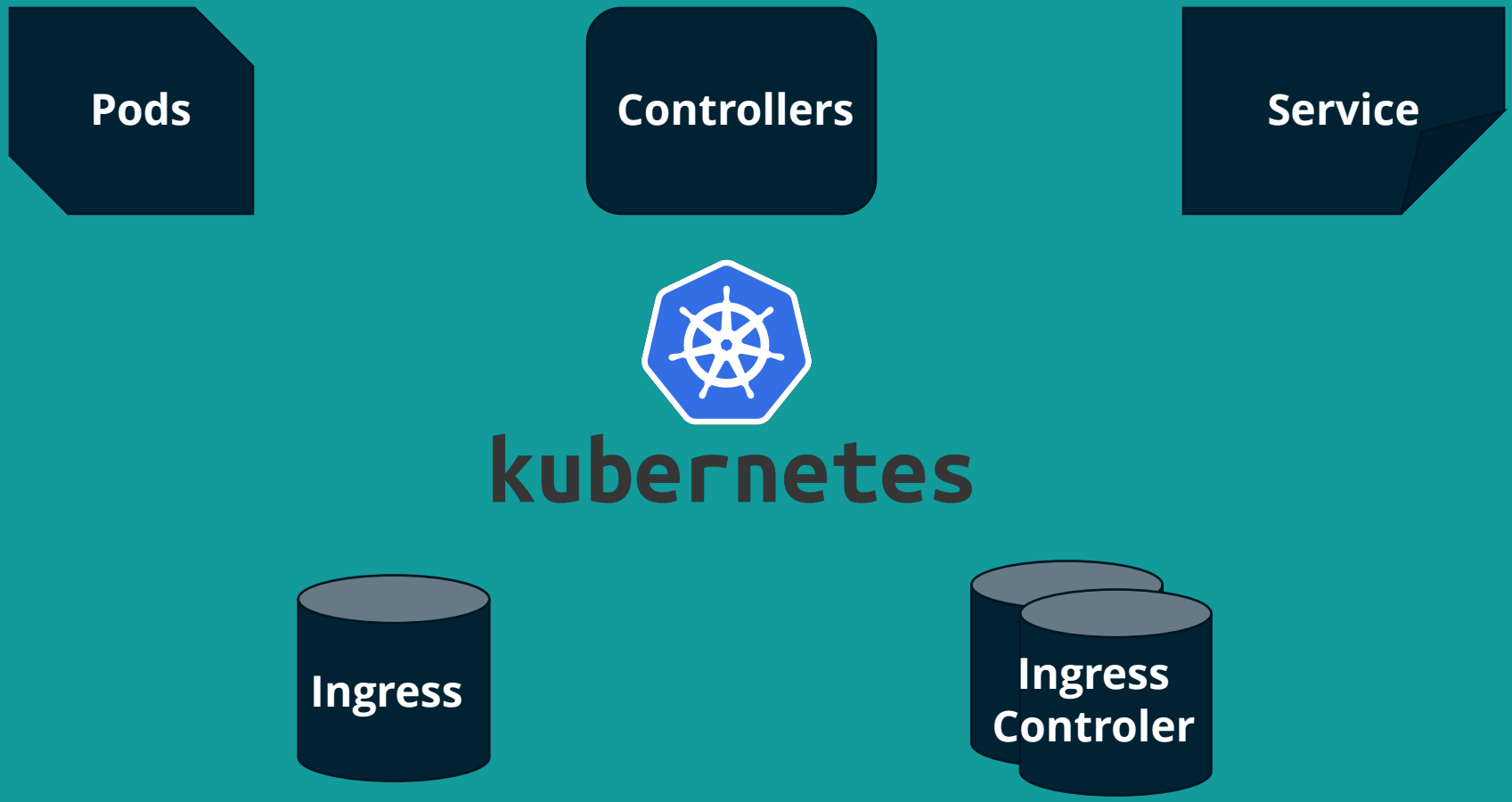
- **What applications are running in containers and the node they are running on**
- **The resources available for those applications**
- **The policies and rules associated with those applications**

apiVersion: Specifies which version of the Kubernetes API to use to create the object

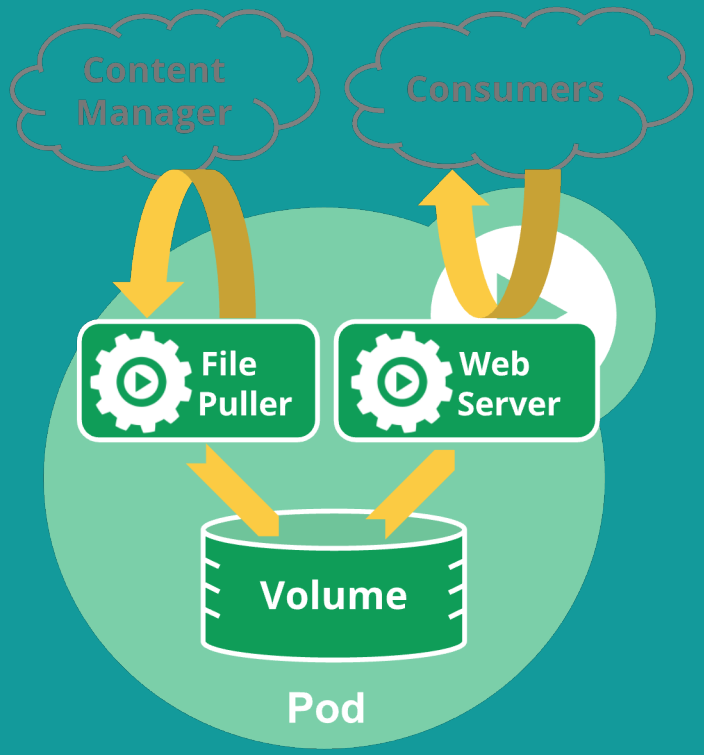
kind: Specifies what type of object you want to create

metadata: A single piece of data that allows the object to be differentiated

Kubernetes Objects



Pods

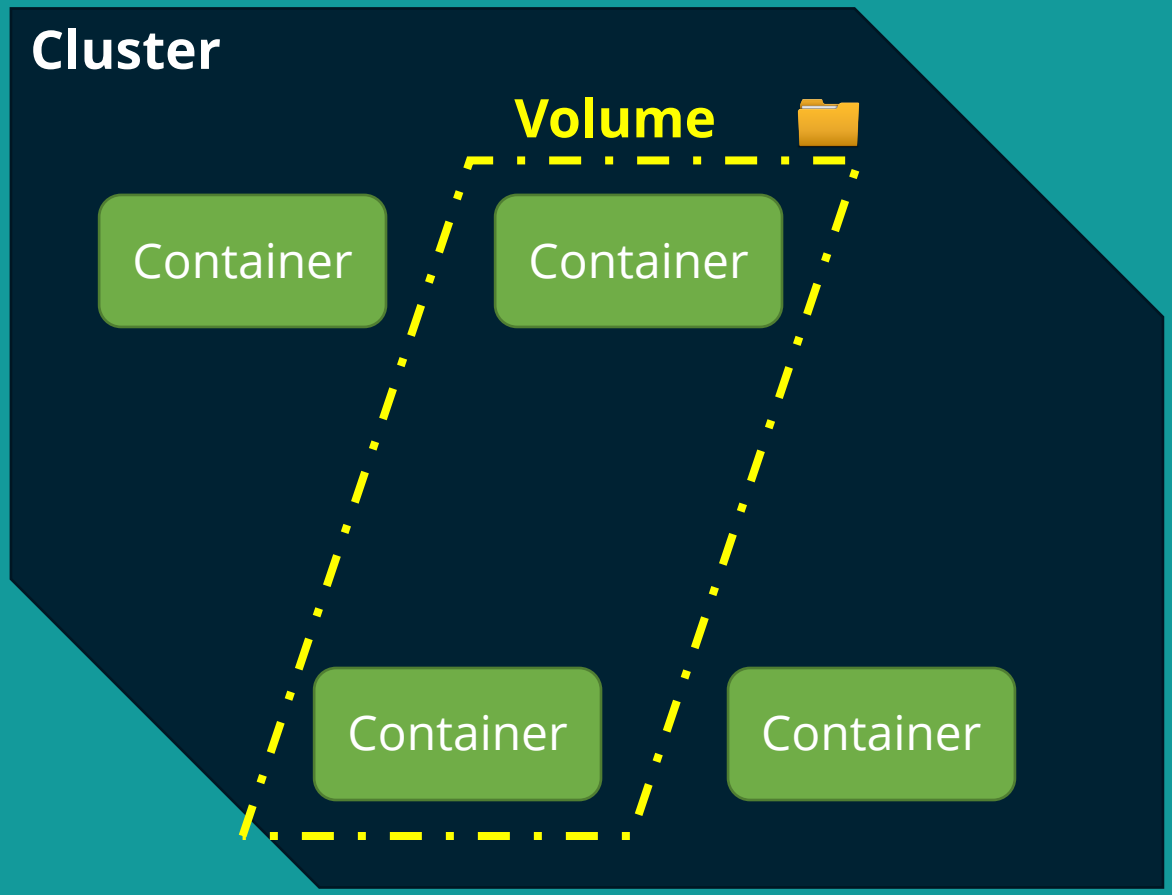


- Pod Name
- Container Image
- Variables
- Ports
- Pull Policy
- Resource Limits
- Readiness

```
apiVersion: v1
kind: Pod
metadata:
      name: nginx
      namespace: default
      labels:
            app: nginx

 spec:
      containers:
            - image: nginx
            name: nginx
```

Volumes



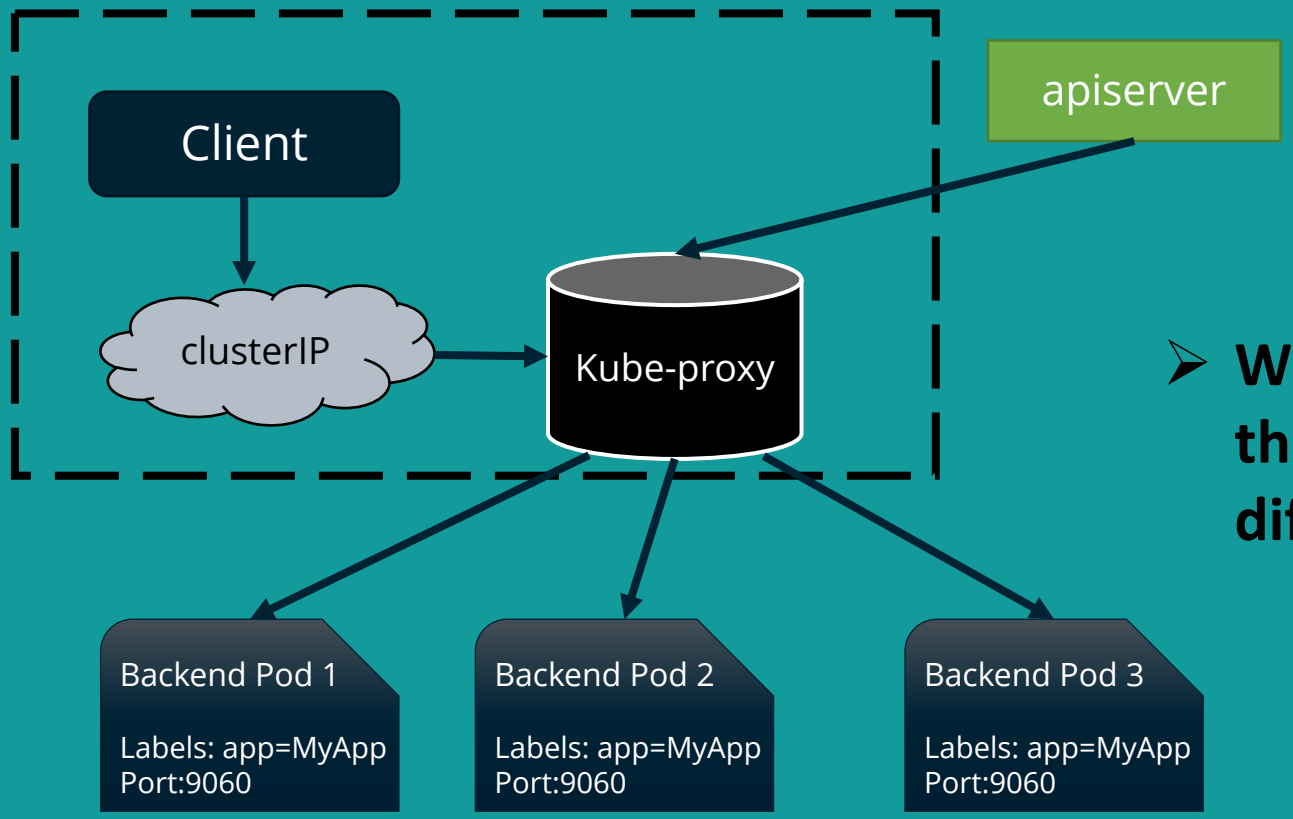
emptyDir

nfs

persistantVolumeClaim

Secret

Kubernetes Services



Stateful Sets

- **What happens if one of the pods replicates the application deployed in it and has a different status than the rest ?**

Kubernetes Networking Model

Container to container

- Network namespace space for each container
- Network namespace is called when a Pod is started

Pod to Pod

- Pods are assigned to nodes in a random way
- Each Pod receives an IP address.

External

- Services are used to provide connectivity
- Services become accessible using virtual IP

Implementing DevSecOps

Tools for Deploying Kubernetes

Kubernetes Architecture

Tools for Deploying Kubernetes

Minikube

Kubeadm

Kops

MicroK8s

Microk8s

K3s

kind

Kind

K3D

K3d

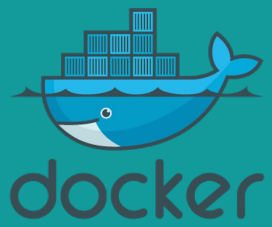
kubernetes

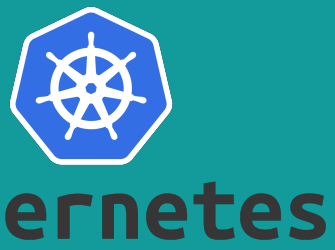
Implementing DevSecOps

Introduction to Kubernetes Security

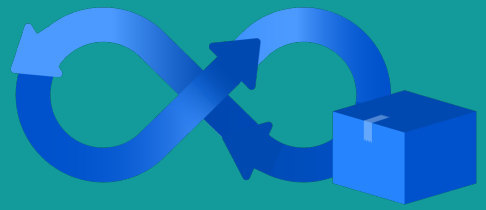
Kubernetes Security

Introduction to Kubernetes Security





REVOLUTION



- Operating in the DevOps model
- Creation of common service sets
- Data-center pre-configuration

Configuring Kubernetes Security

Multiple Nodes

Replication

Services

Pod

Kubernetes controller

Systemd, kube-api-server, kube-controller-manager

Kubernetes nodes

Kube-proxy, kubelet

Kubectl command

kubectl

Resource files

Json, yaml

Implementing DevSecOps

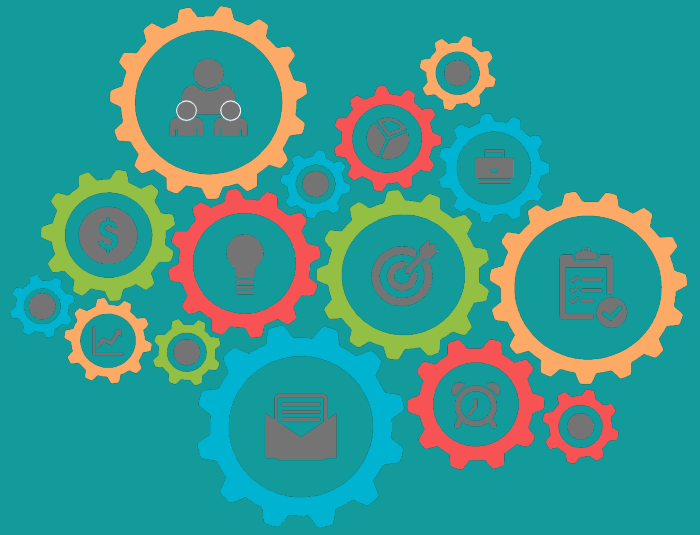
Kubernetes Security Best Practices

Kubernetes Security

Using Secrets

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
    name: cluster-role
rules:
- apiGroups: [“”]
  resources: [“pods”]
  verbs: [“get”, “list”]
```

Secret



K8s Built-In Security Features

Security Features

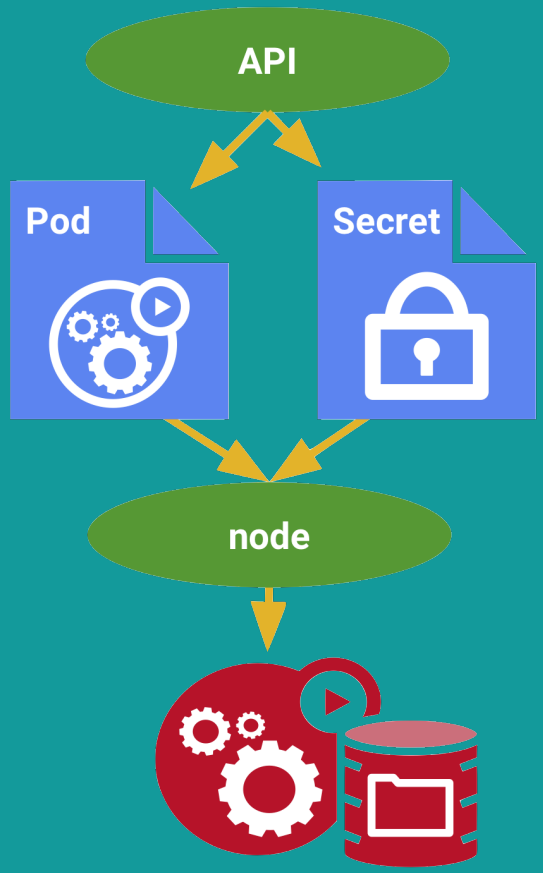
- **Role-Based Access Control (RBAC)**
- **Pod security policies and network policies**
- **Network encryption**

No Protection Against

- **Malicious code inside containers or images**
- **Security vulnerabilities in host operating systems**
- **Container runtime vulnerabilities**
- **Kubernetes API abuse**
- **Tools vulnerabilities or configuration errors**

Managing Kubernetes Secrets

Secret



- Secrets are objects with namespaces, they exist in the context of a namespace
- You can access them through a volume or environment variable from a container running in a pod.

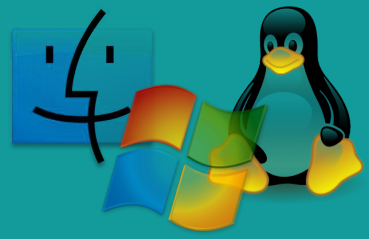
Managing Kubernetes Secrets

Integrate security from the early stages of development

Consider a commercial platform of Kubernetes

Do not trust your old tools and practices

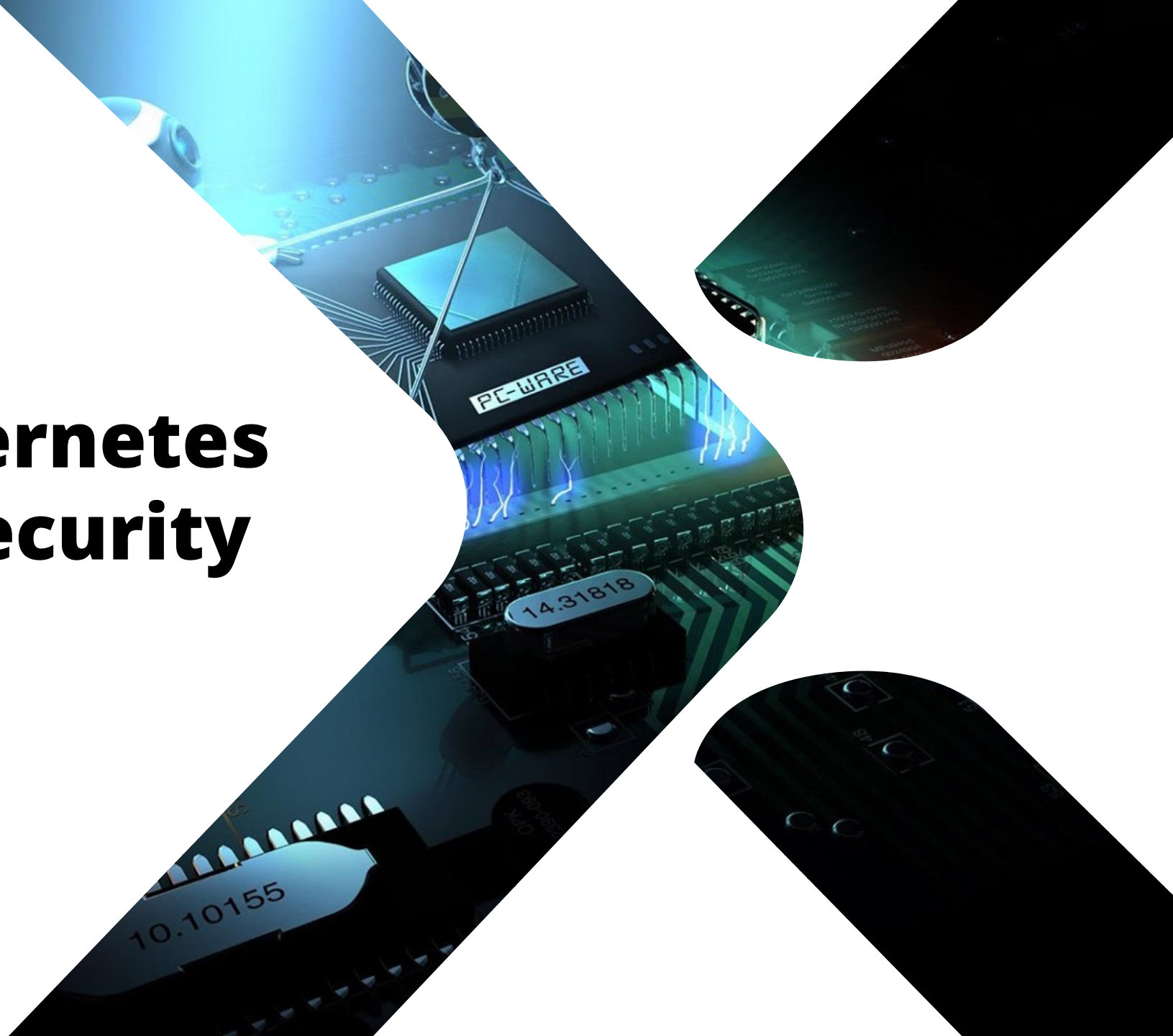
Containers



Operating System

Container Runtime

Network Layer

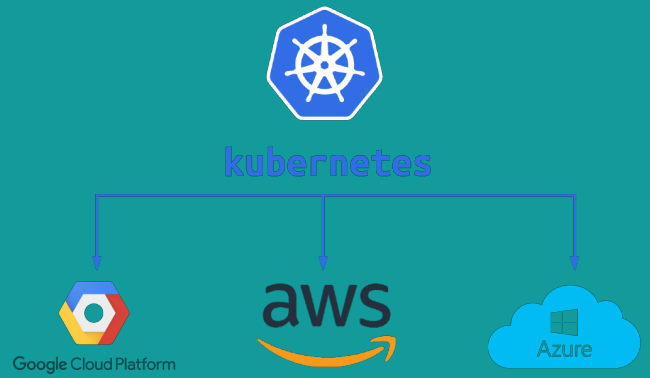


Analyzing Kubernetes Components Security

Kubernetes Security

Kubernetes Components Security

```
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
  name: permissive
spec:
  privileged: true
  hostNetwork: true
  hostIPC: true
  hostPID: true
  seLinux:
   rule: RunAsAny
```




```
supplementalGroups:
   rule: RunAsAny
runAsUser:
   rule: RunAsAny
fsGroup:
   rule: RunAsAny
hostPorts:
- min: 0
   max: 65535
volumes:
- '*'
```

Security Projects

Containers in privileged mode

- Host namespace
  - HostPID
  - HostIPC
  - HostNetwork
  - HostPorts

Volumes and filesystems

- Volumes
- FSGroup
- AllowedHostPaths
- ReadOnlyRootFilesystem

Users and groups

- RunAsUser
- RunAsGroup

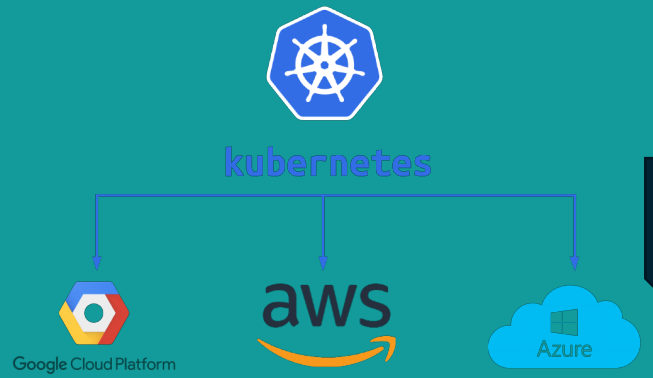
Privilege & Capabilities

- Privilege escalation
- allowPrivilegeEscalation
- DefaultAllowPrivilegeEscalation
- Capabilities
- AllowedCapabilities
- RequiredDropCapabilities

Kubernetes Components Security

```
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
  name: permissive
spec:
  privileged: true
  hostNetwork: true
  hostIPC: true
  hostPID: true
  seLinux:
   rule: RunAsAny
```

```
supplementalGroups:
   rule: RunAsAny
runAsUser:
  rule: RunAsAny
fsGroup:
  rule: RunAsAny
hostPorts:
- min: 0
  max: 65535
volumes:
- '*'
```



livenessProbe

readinessProbe

Implementing DevSecOps

KubeBench Security

Auditing and Analyzing Kubernetes Vulnerabilities

CIS Benchmarks

https://hub.docker.com/r/aquasec/kube-bench

Internet Security

Security Projects

Auditing and Analyzing Kubernetes Vulnerabilities

Security Projects

- Relies on known attack vectors
- It allows remote, internal, or CIDR scanning

https://github.com/aquasecurity/kube-hunter

- Helps you quantify the risk for Kubernetes resources
- Runs against your Kubernetes applications

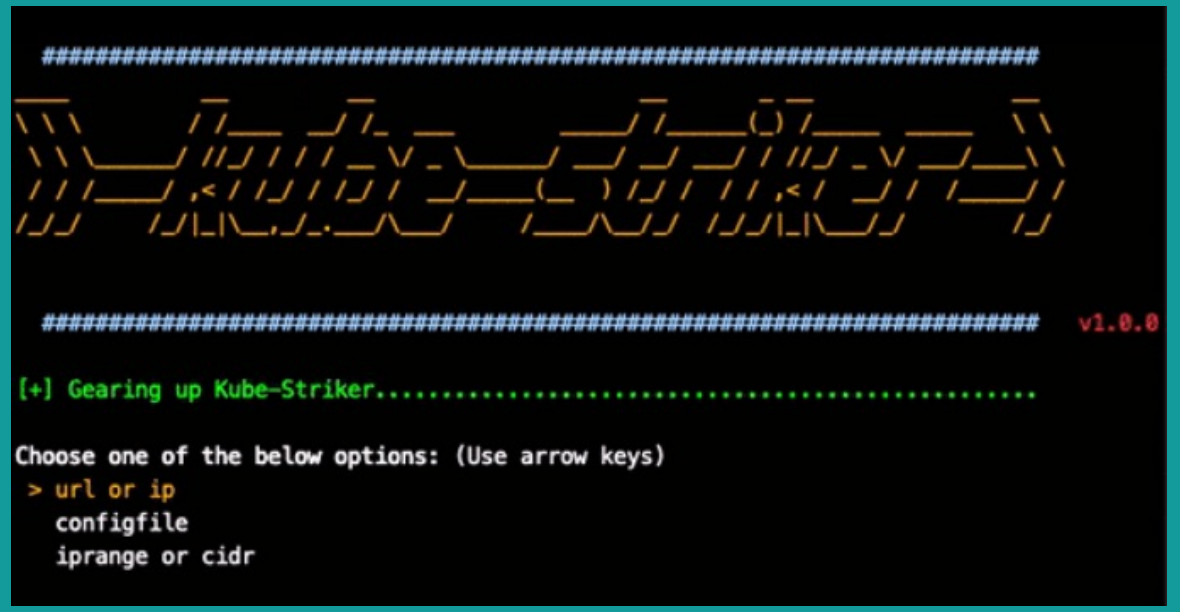
https://kubesec.io/

- Getts the information from a Kubernetes cluster

https://github.com/sysdiglabs/kubectl-dig

Kubestriker

Performs numerous in-depth checks on a range of services and open ports

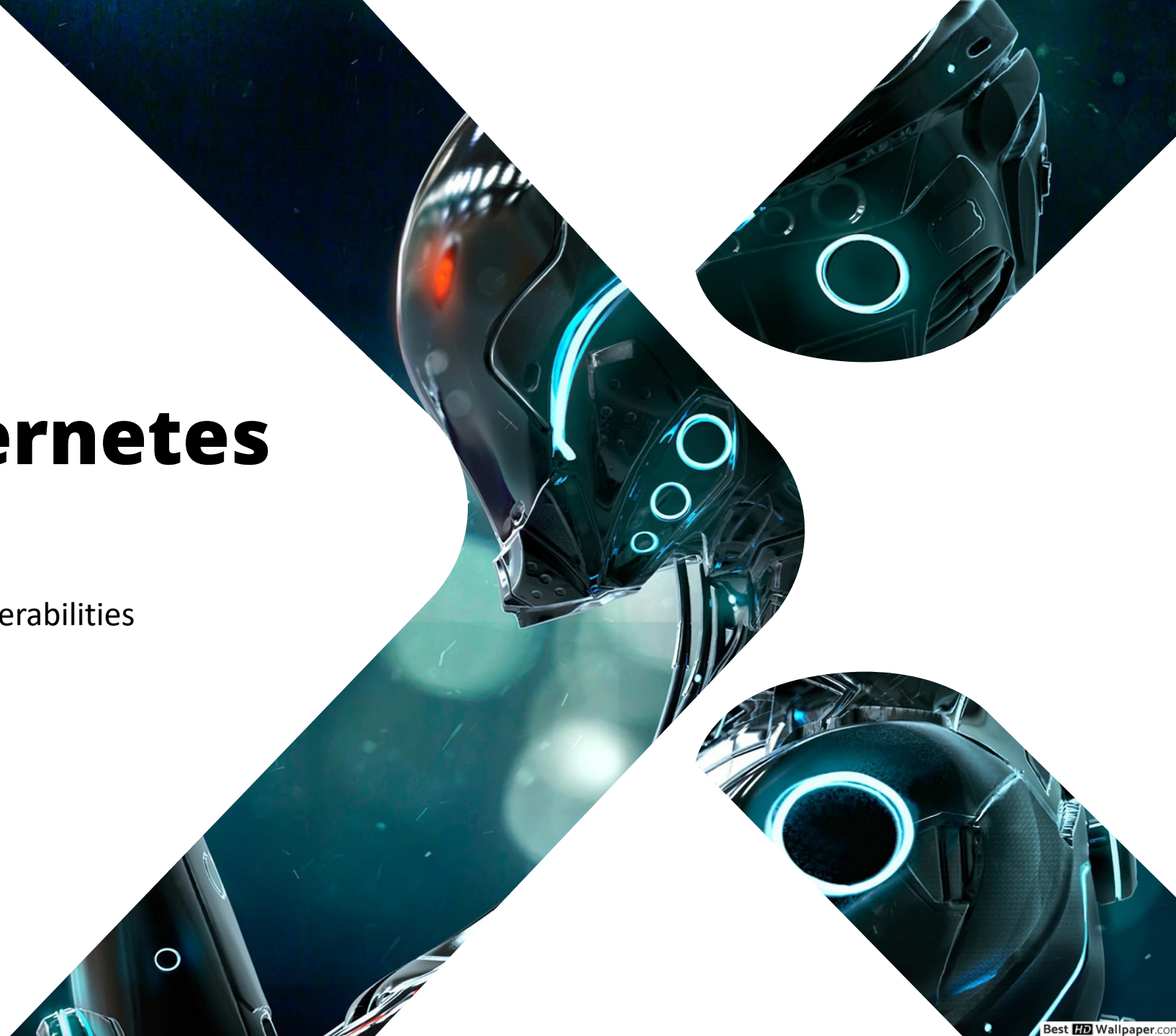


https://github.com/vchinnipilli/kubestriker

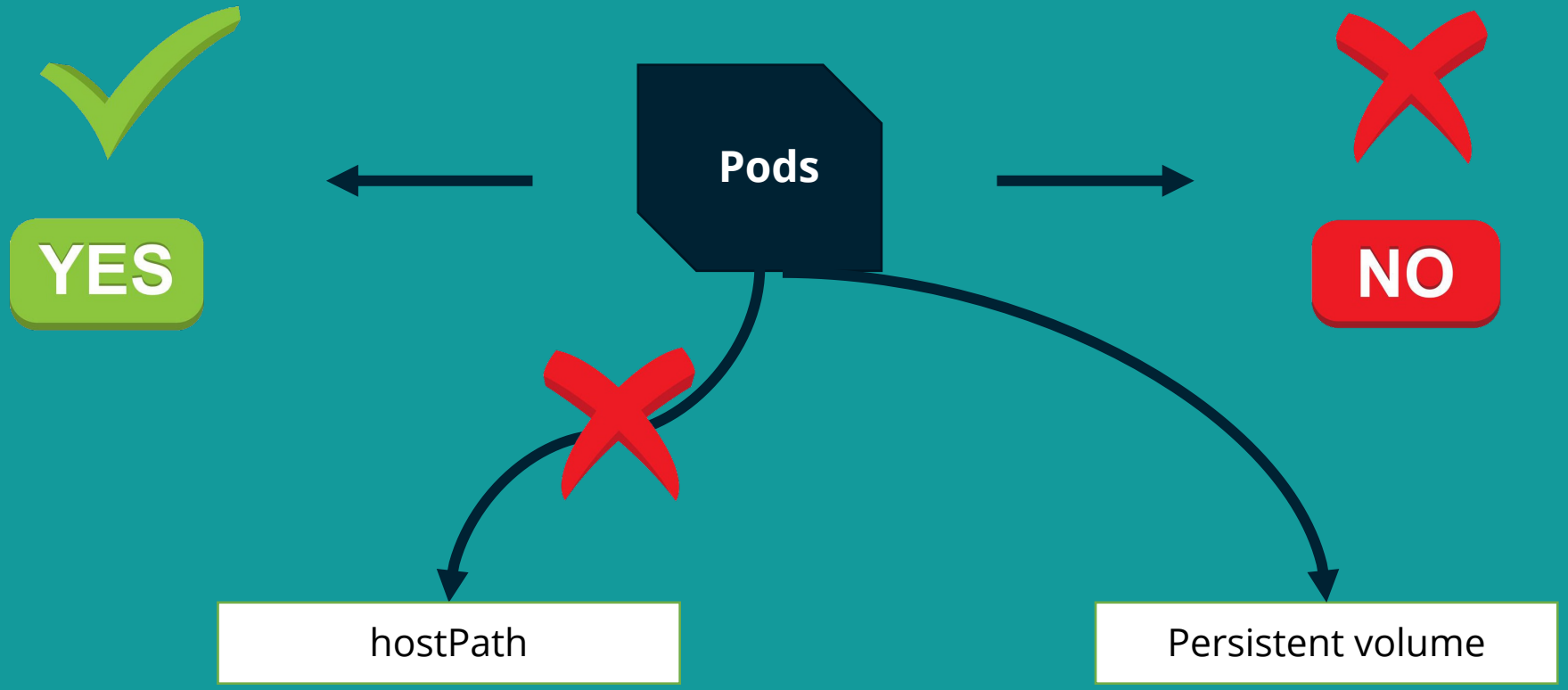
Implementing DevSecOps

Analyzing Kubernetes Vulnerabilities

Auditing and Analyzing Kubernetes Vulnerabilities



Pod Security Policy



Kubernetes Dashboard and Cluster

Monitoring Kubernetes

Observability and Monitoring

Monitoring

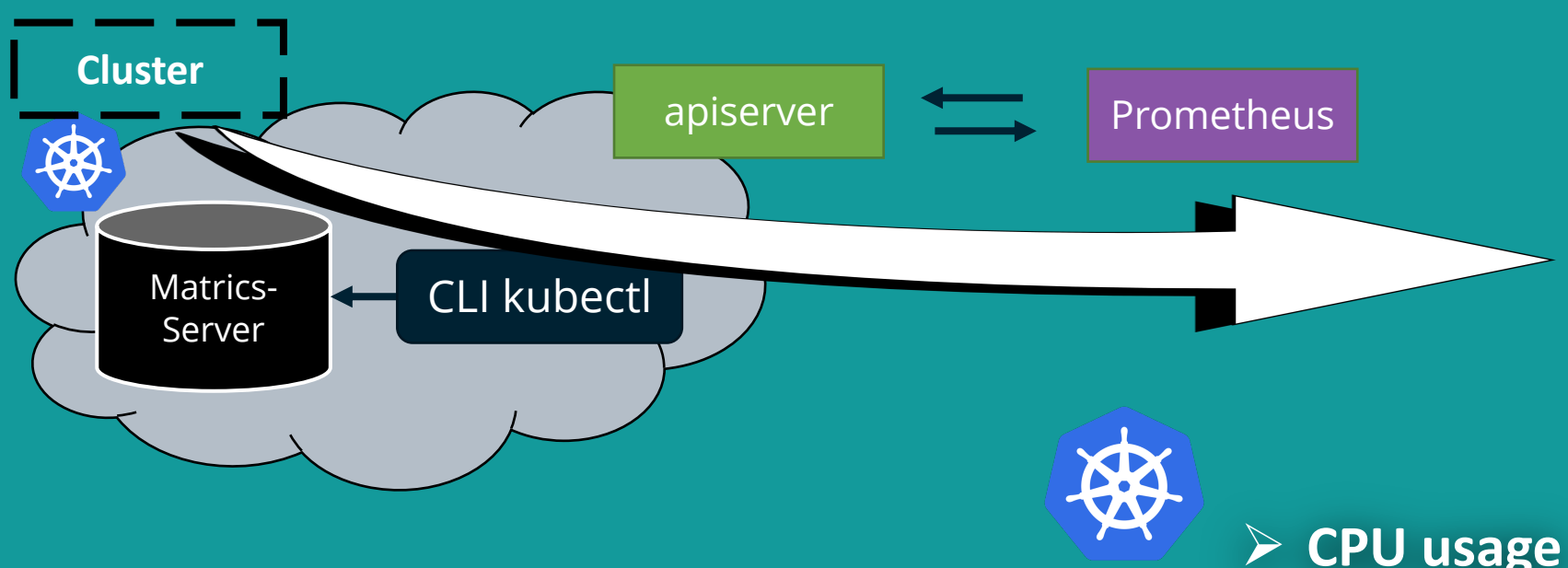
Open instrumentation

Correlation and context

Programmability

APOps

Observability in a Cluster



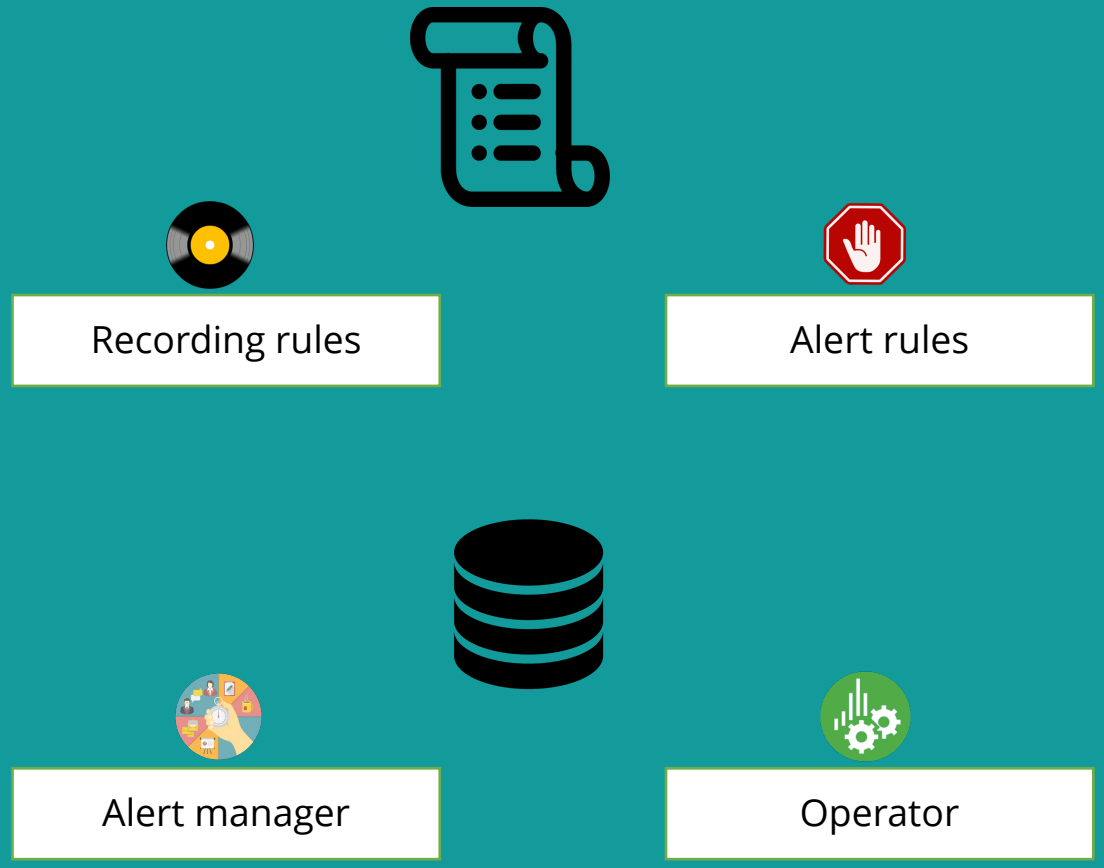
- CPU usage
- Memory usage
- Disk usage
- Network bandwidth
- Pods resources

Prometheus Tools

Monitoring Kubernetes

Prometheus

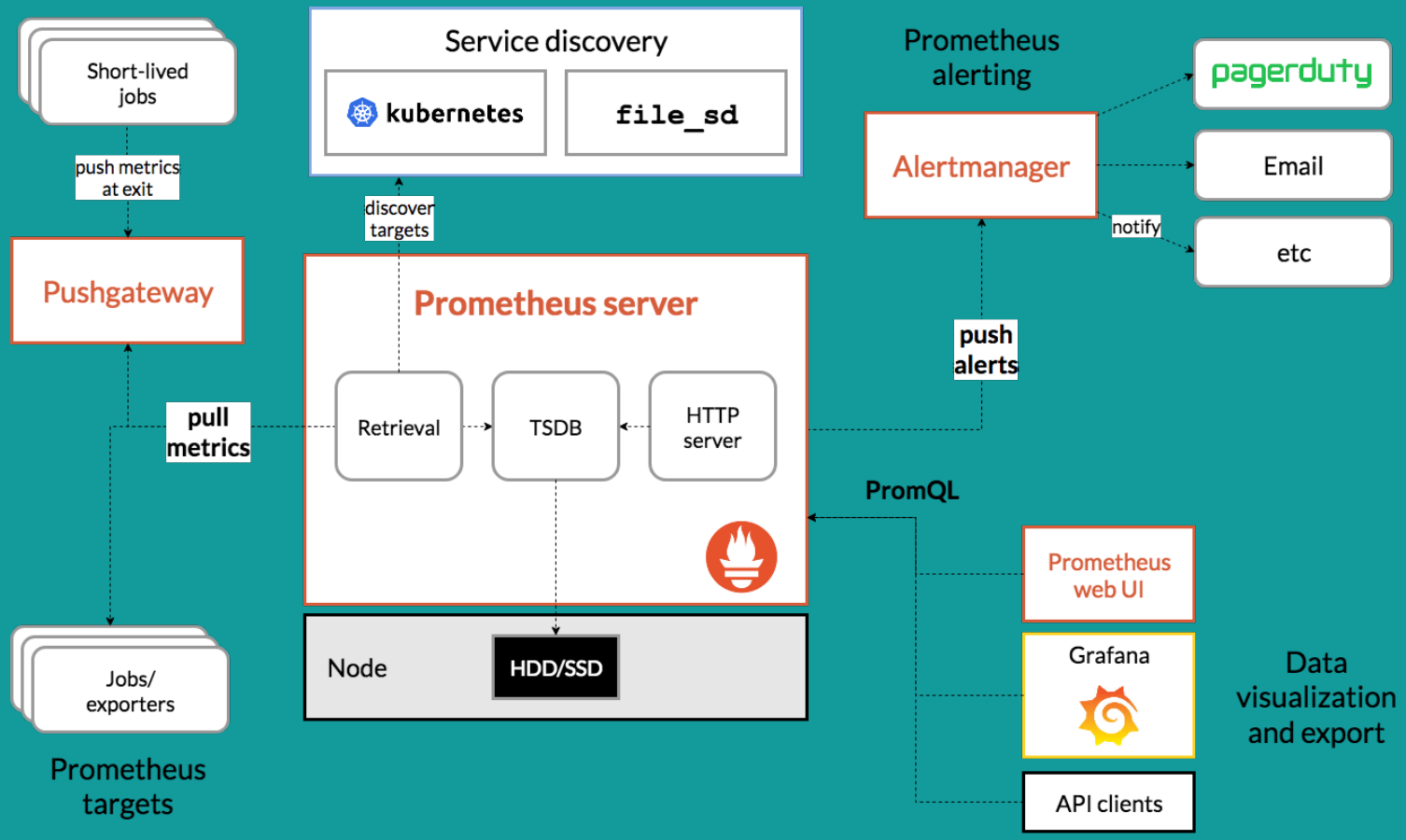
- Managing alarms
- Grouping alarms
- Sending alarms to other applications



https://github.com/prometheus/prometheus

Prometheus Architecture

- Provides a multidimensional data model and a powerful query language
- Collects metrics information automatically
- Offer four types of metrics: counter, gauge, histogram, and summary.



Implementing DevSecOps

Collecting and Exploring Matrix with Grafana

Monitoring Kubernetes

Grafina

https://grafana.com

- Display graphs of collected data
- Metrics include the utilization of CPU/memory/disk of the K8s master and workers.
- The cluster metrics include data at the container level and K8s cAdvisor endpoints.

CPU usage per node

Unavailable nodes

RAM usage per node

Undesired PODs

File usage per node

Other Analyzing Tools

https://www.datadoghq.com

https://www.influxdata.com

https://newrelic.com

https://www.splunk.com