A DevOps State of Mind: Continuous Security with DevSecOps + Containers

Chris Van Tuin
Chief Technologist, NA West / Silicon Valley
cvantuin@redhat.com

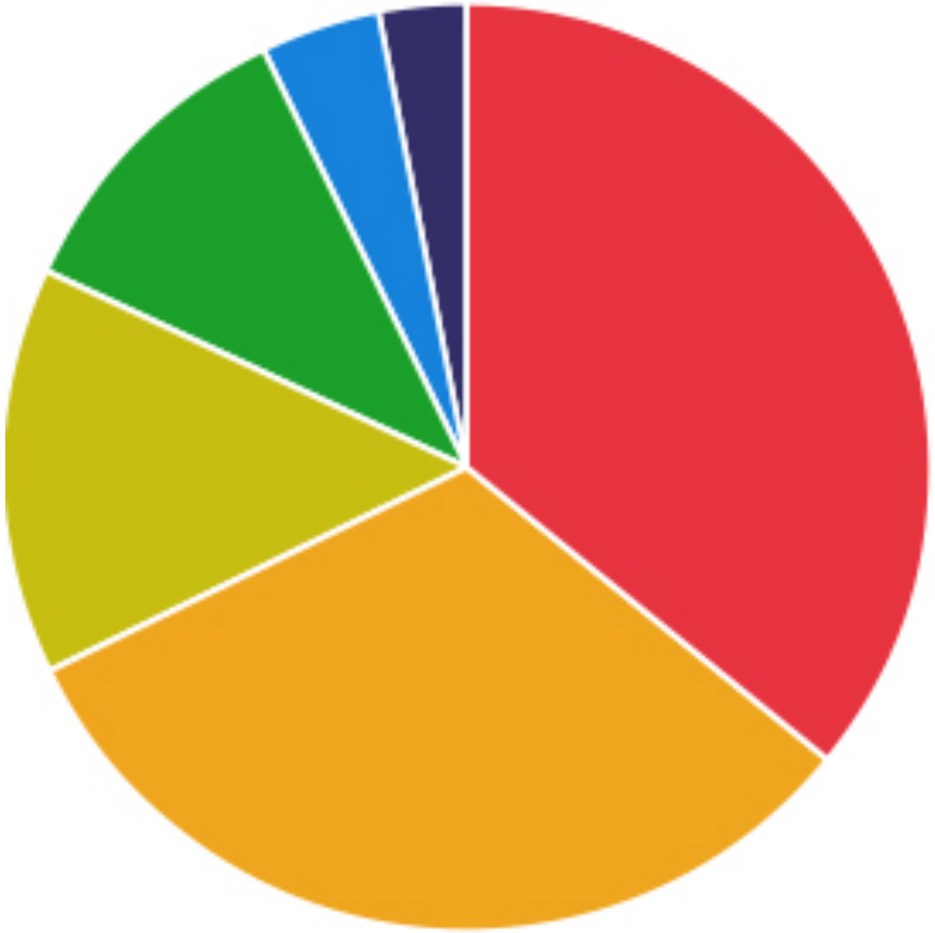
redhat.

SECURITY BREACH: BILLION DATA RECORDS

http://www.informationisbeautiful.net/visualizations/worlds-biggest-data-breaches-hacks/

WHAT IS THE GREATEST SECURITY RISK?

% of Respondants



- 36% - Employees not taking proper security measures
- 32% - Outside breach
- 14% - Unpatched or unpatchable
- 11% - Internal attack by an employee
- 4% - Shadow IT
- 3% - Bring your own device/mobile

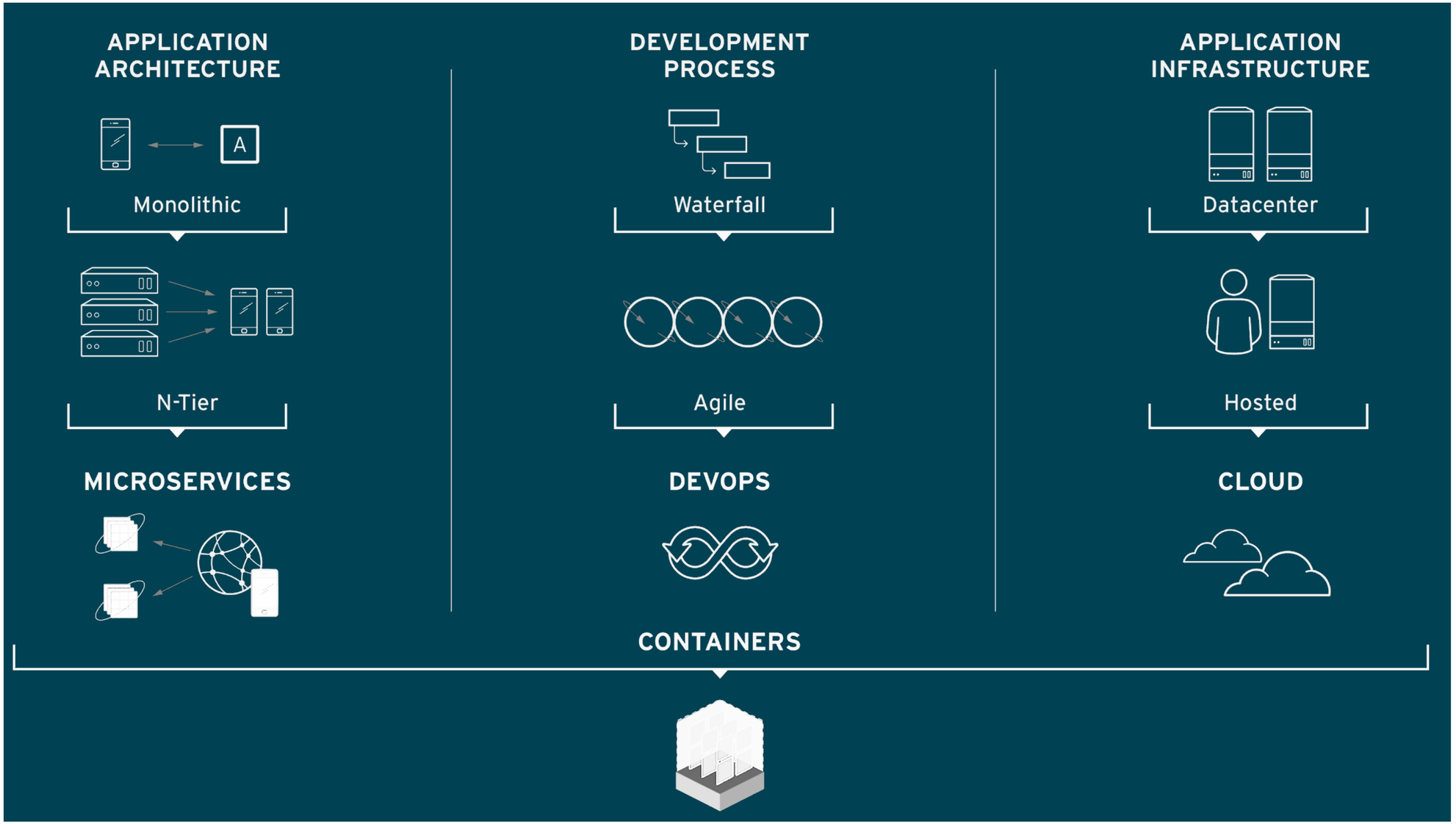
Source: Techvalidate/Red Hat

redhat.

“Only the paranoid survive”

- Andy Grove, 1996

SECURITY MUST EVOLVE & KEEP UP

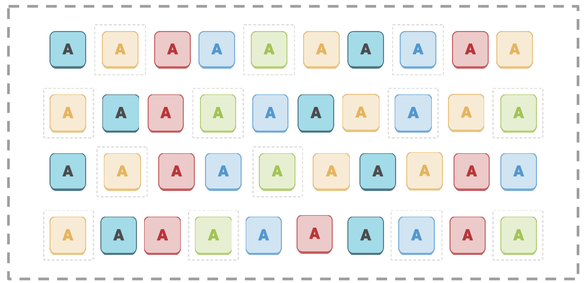


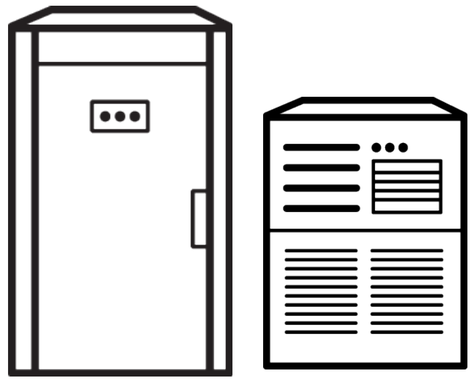
redhat.

HYBRID CLOUD ENVIRONMENTS

ANY COMBINATION, WHETHER TRADITIONAL OR CONTAINERIZED

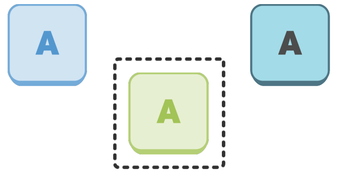
LEGACY APPS
(1,000+)



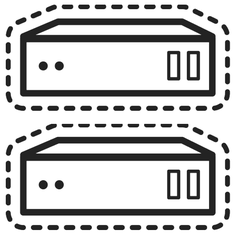


PRODUCTION

DEV/TEST

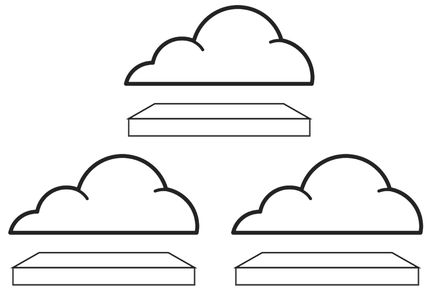


BARE METAL



VIRTUAL

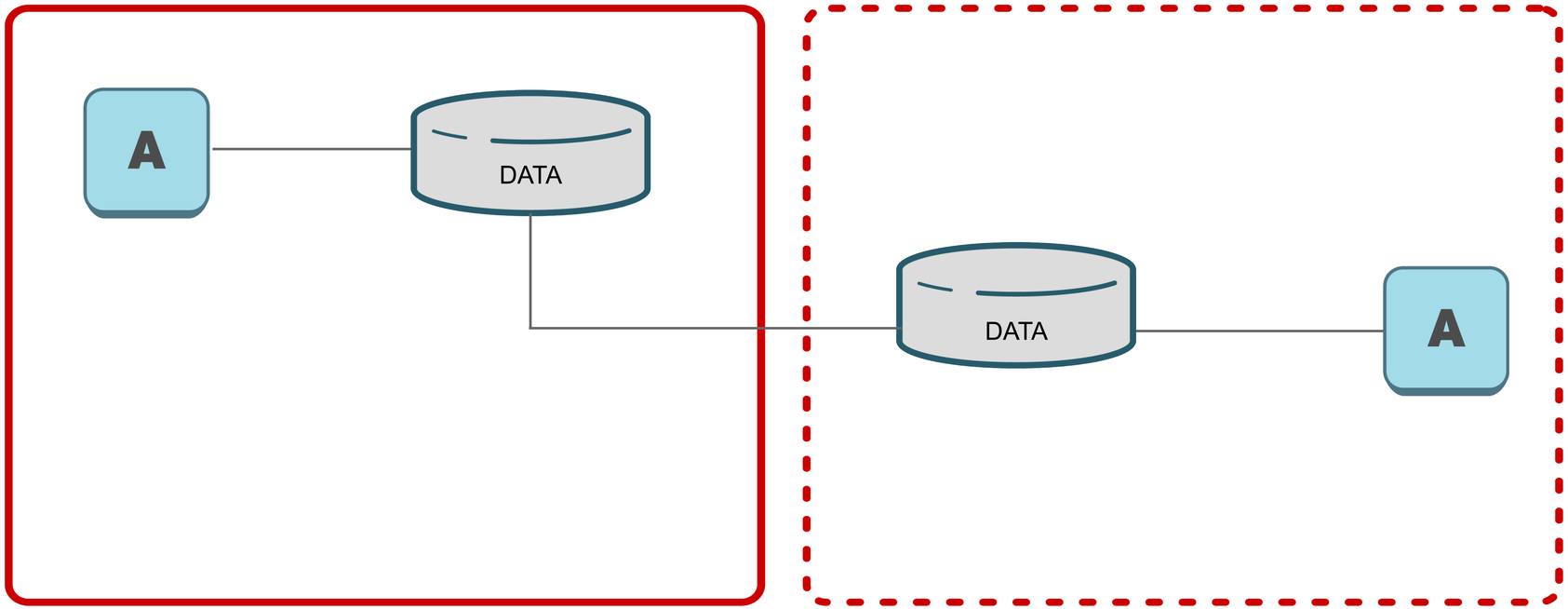
PRIVATE CLOUD



PUBLIC CLOUD

redhat.

DISTRIBUTED APPLICATIONS



ON-PREMISE

OFF-PREMISE

BARE METAL

VIRTUAL

PRIVATE CLOUD

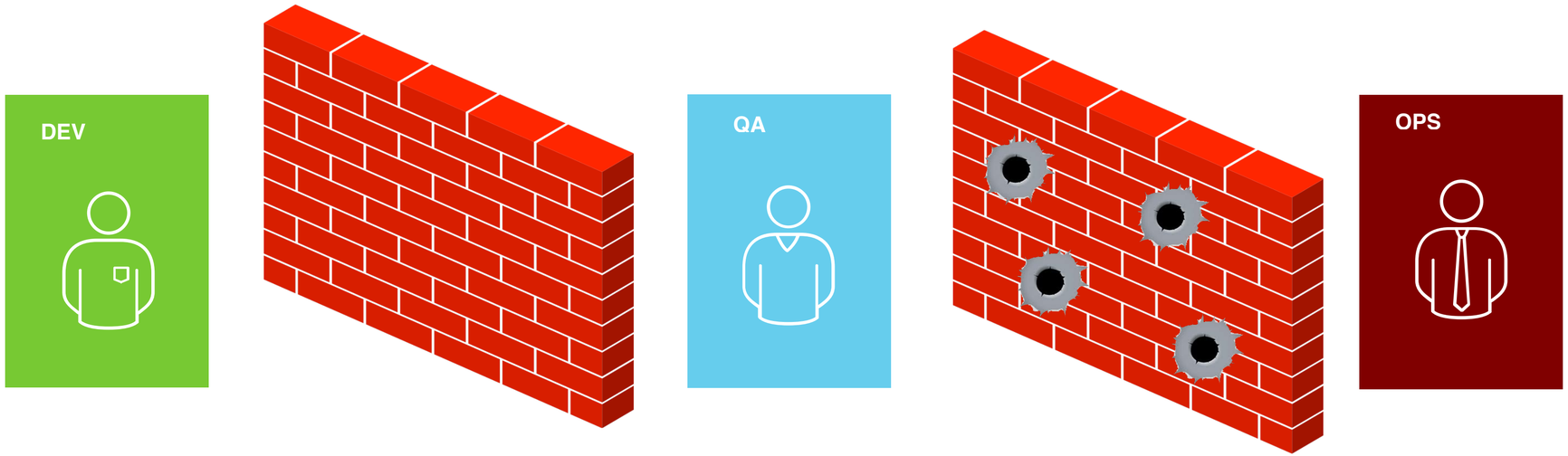
PUBLIC CLOUD

redhat.

SECURITY IS AN AFTERTHOUGHT

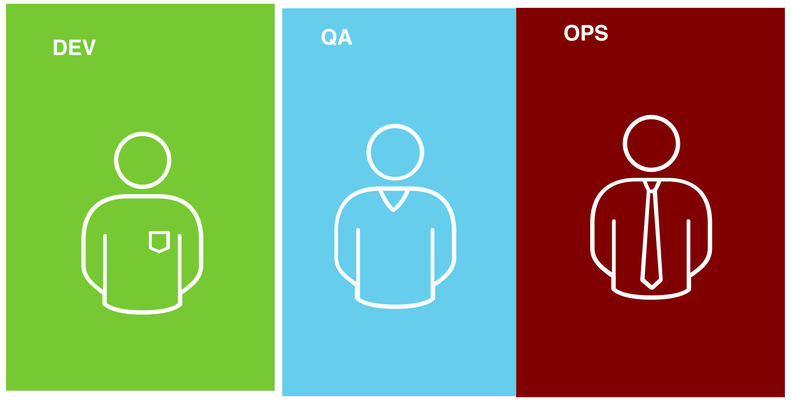
"Patch?
The servers are behind the firewall."

- Anonymous (far too many to name), 2005 - …



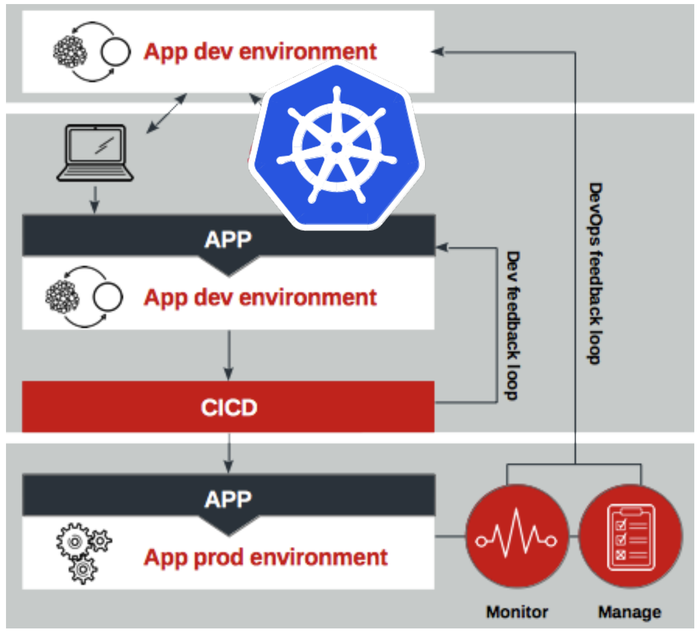
DEVSECOPS

Culture



+

Process



+

Technology

Linux + Containers
IaaS
Orchestration
CI/CD
Source Control Management
Collaboration
Build and Artifact Management
Testing
Frameworks

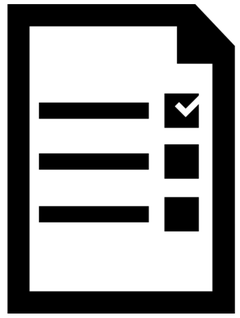
Open Source

End to End Security

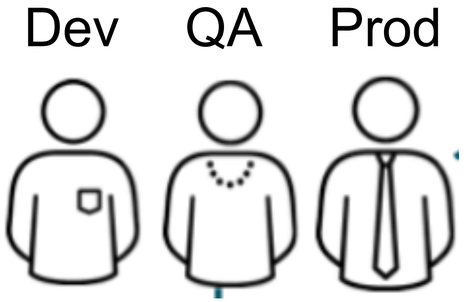
redhat.

DEVSECOPS

Reduce Risks, Lower Costs, Speed Delivery, Speed Reaction



Security Automation

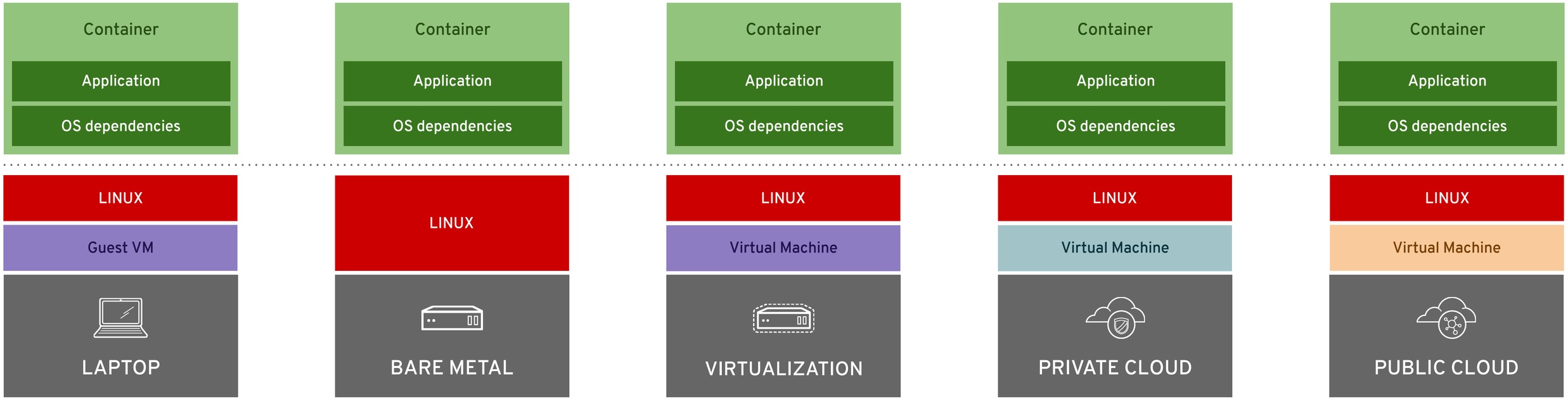


Process Optimization

Continuous Security Improvement

CONTAINERS

APPLICATION PORTABILITY WITH CONTAINERS

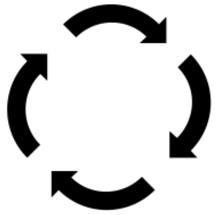


redhat.

CONTAINERS AT SCALE

MORE THAN CONTAINERS…

Scheduling



Lifecycle & health

Discovery

Monitoring

Scaling

Persistence

Aggregation

Security

kubernetes

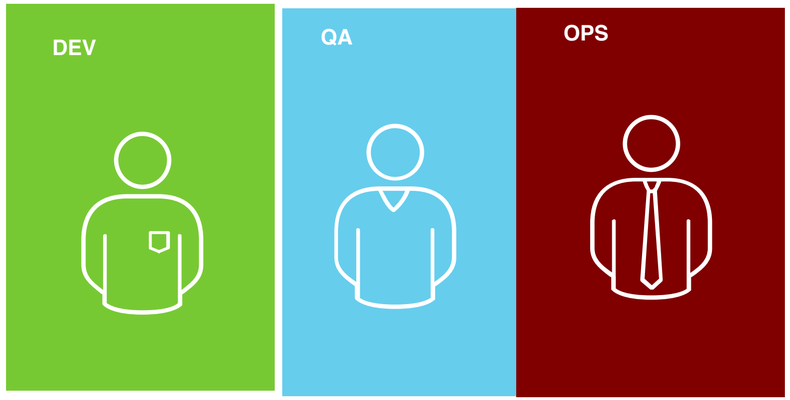
BARE METAL

VIRTUAL

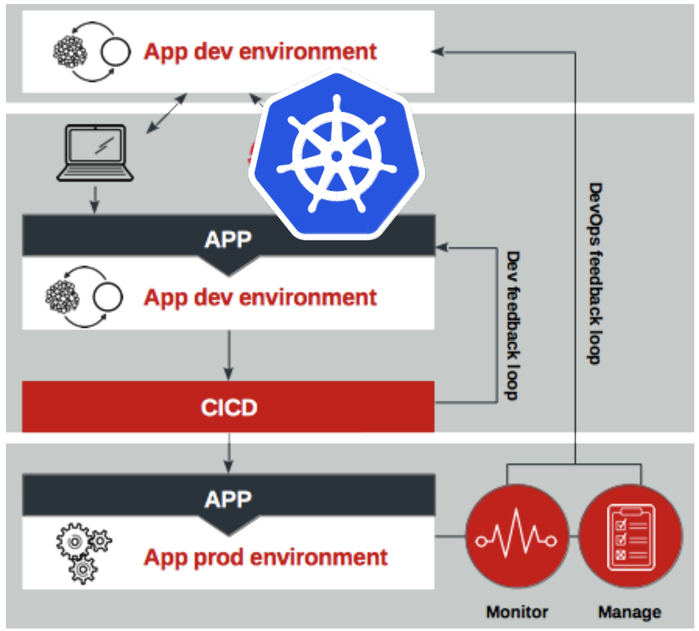
PRIVATE CLOUD

PUBLIC CLOUD

DEVSECOPS
End to End Security



+



+

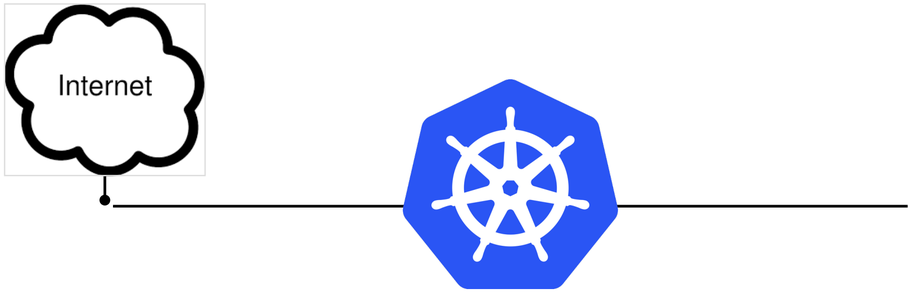
kubernetes by Google

SECURITY

redhat.

ORCHESTRATION

Deployment, Declarative



Controller
Manager
&
Data Store
(etcd)

Web
replicas=2,
role=web

Database
replicas=1,
role=db

Nodes

redhat.

ORCHESTRATION

Schedule + Provision Pods (Compute/Storage/Network)

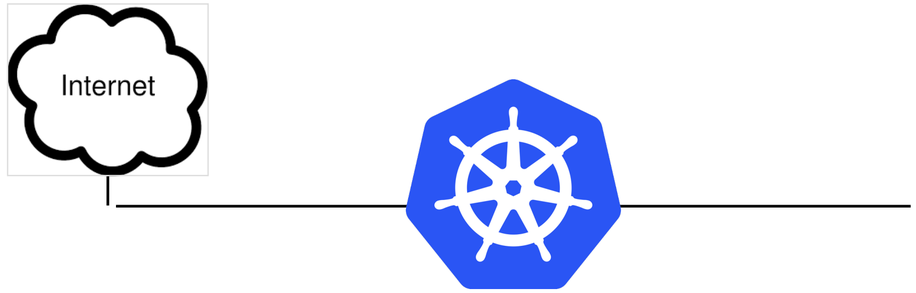


Image Registry

Web
replicas=2,
role=web
ReplicaSet

Pods

role=web

role=web

Nodes

redhat.

ORCHESTRATION

Schedule + Provision Pods (Compute/Storage/Network)

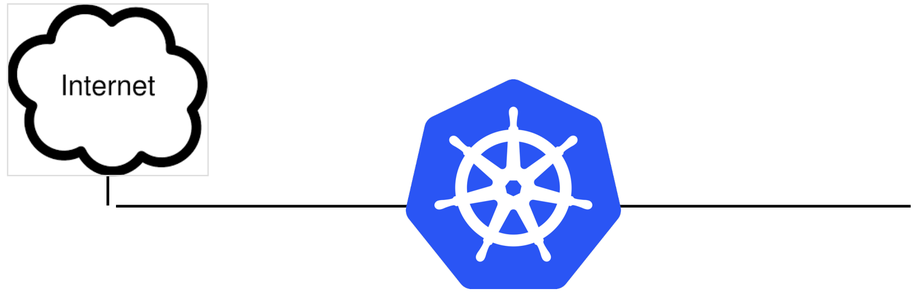
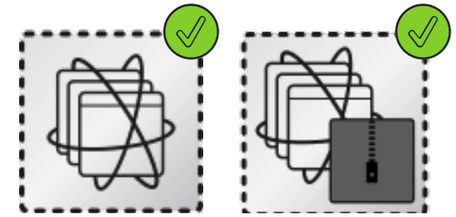


Image Registry

Web
replicas=2,
role=web
ReplicaSet

Database
replicas=1,
role=db
StatefulSet

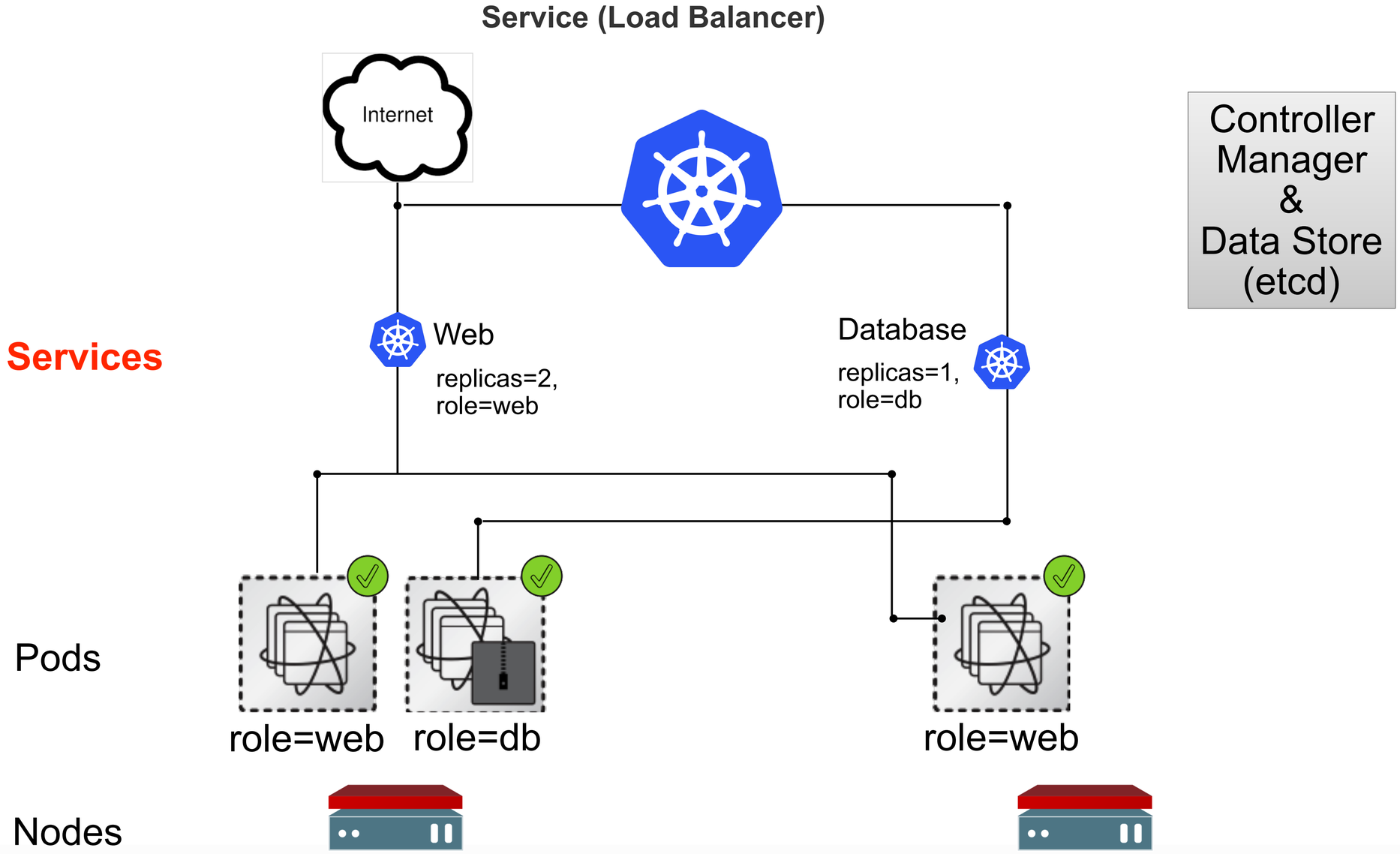
Pods



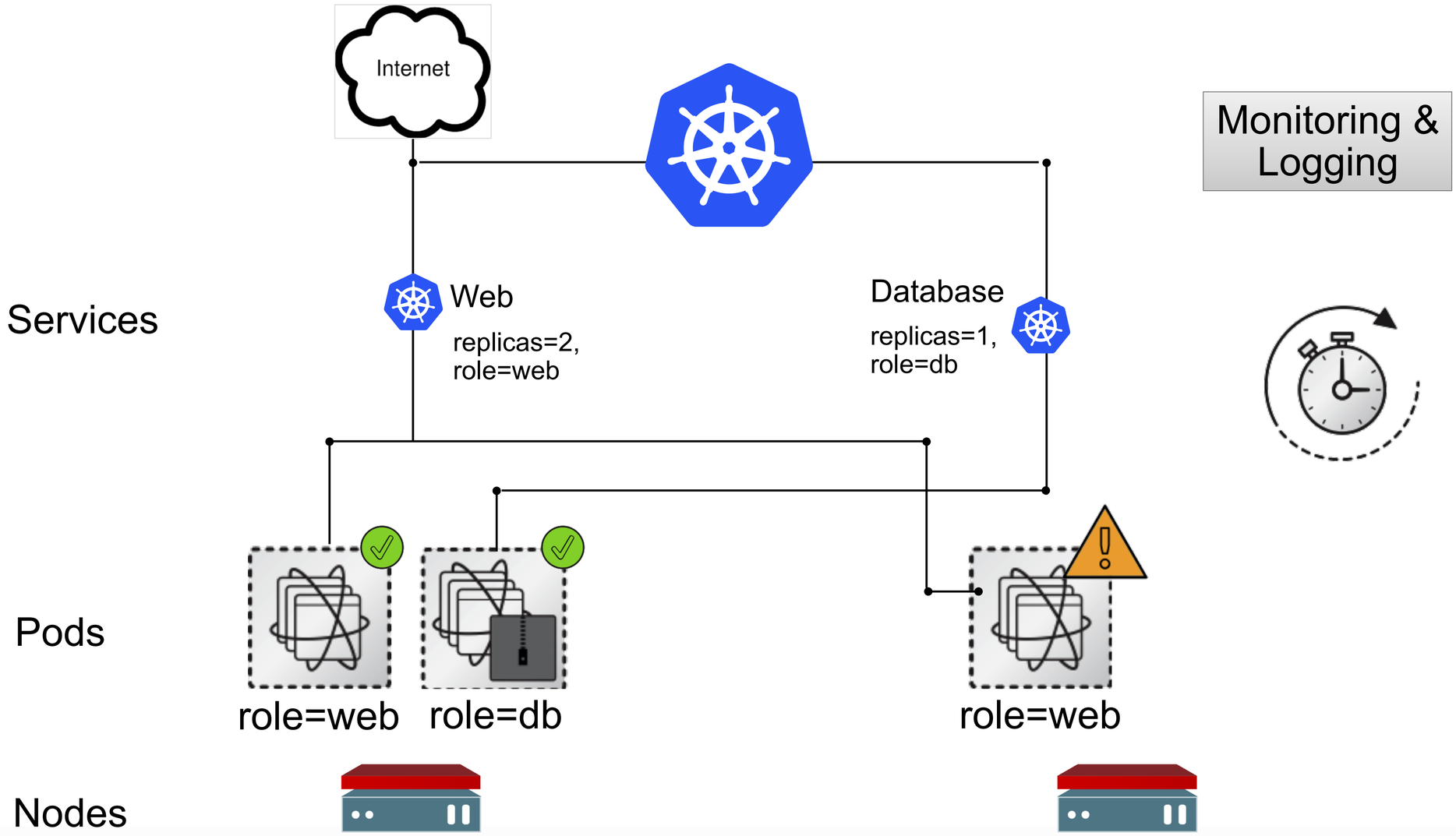
Nodes

redhat.

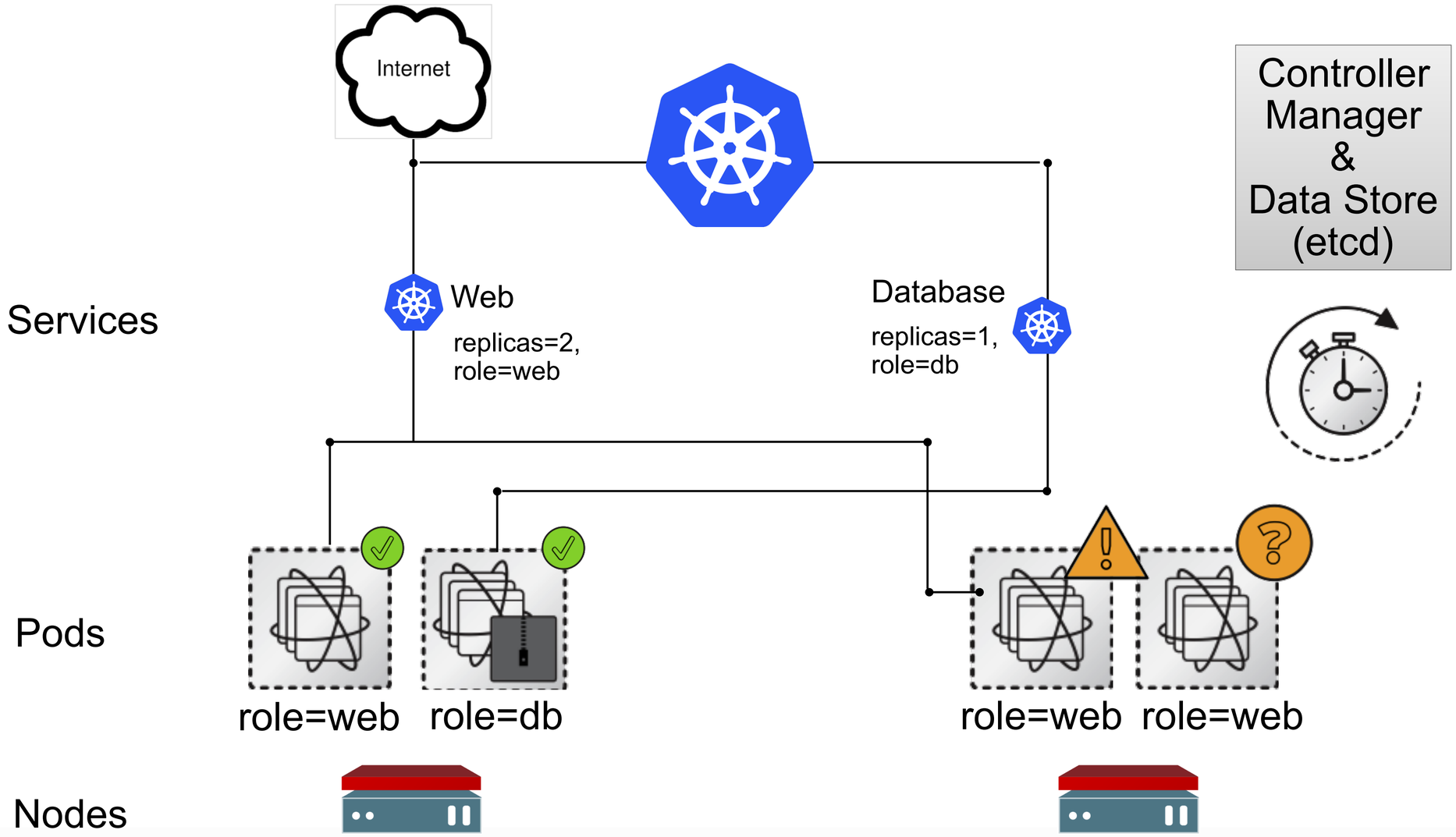
ORCHESTRATION



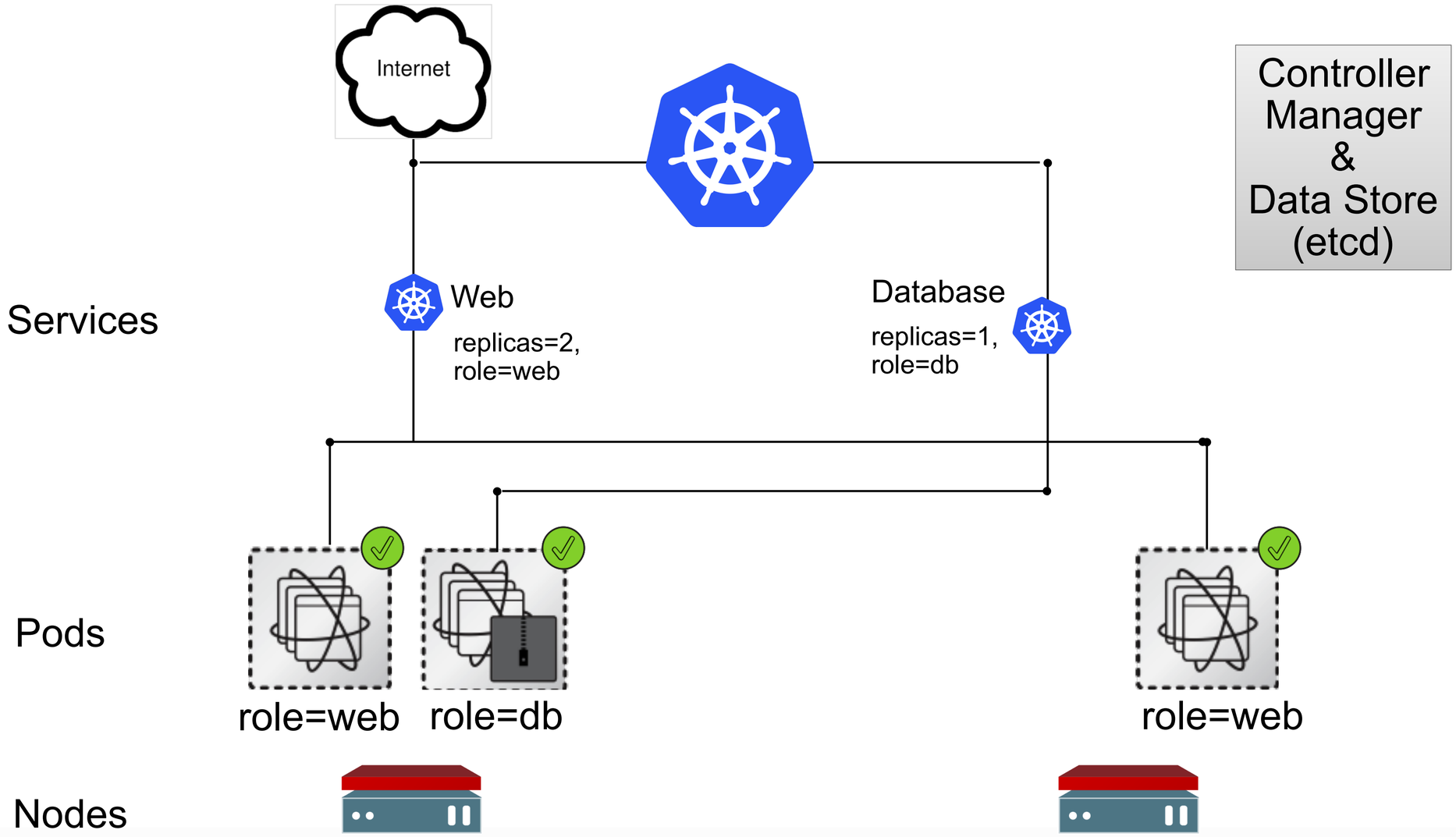
HEALTH CHECK



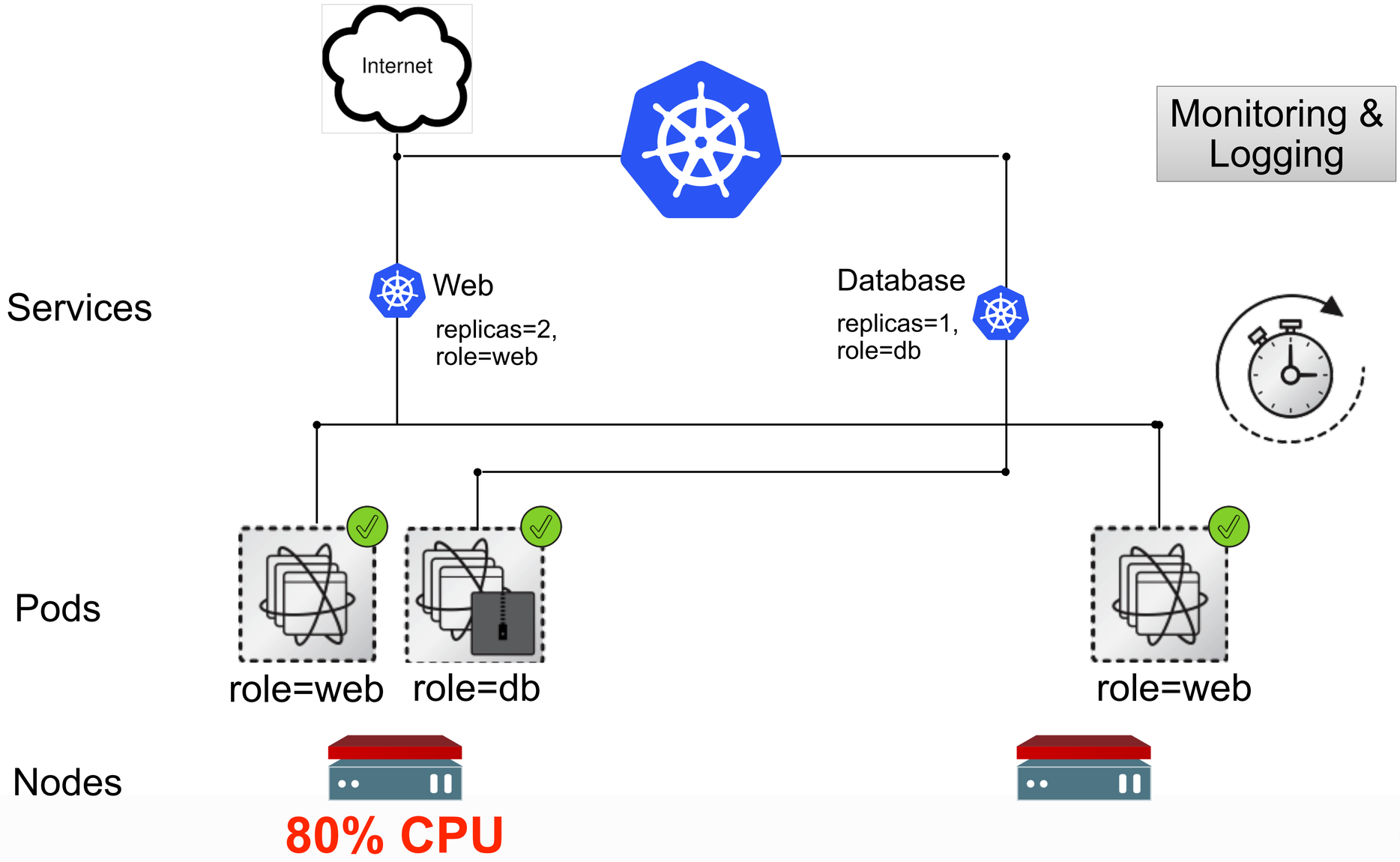
HEALTH CHECK



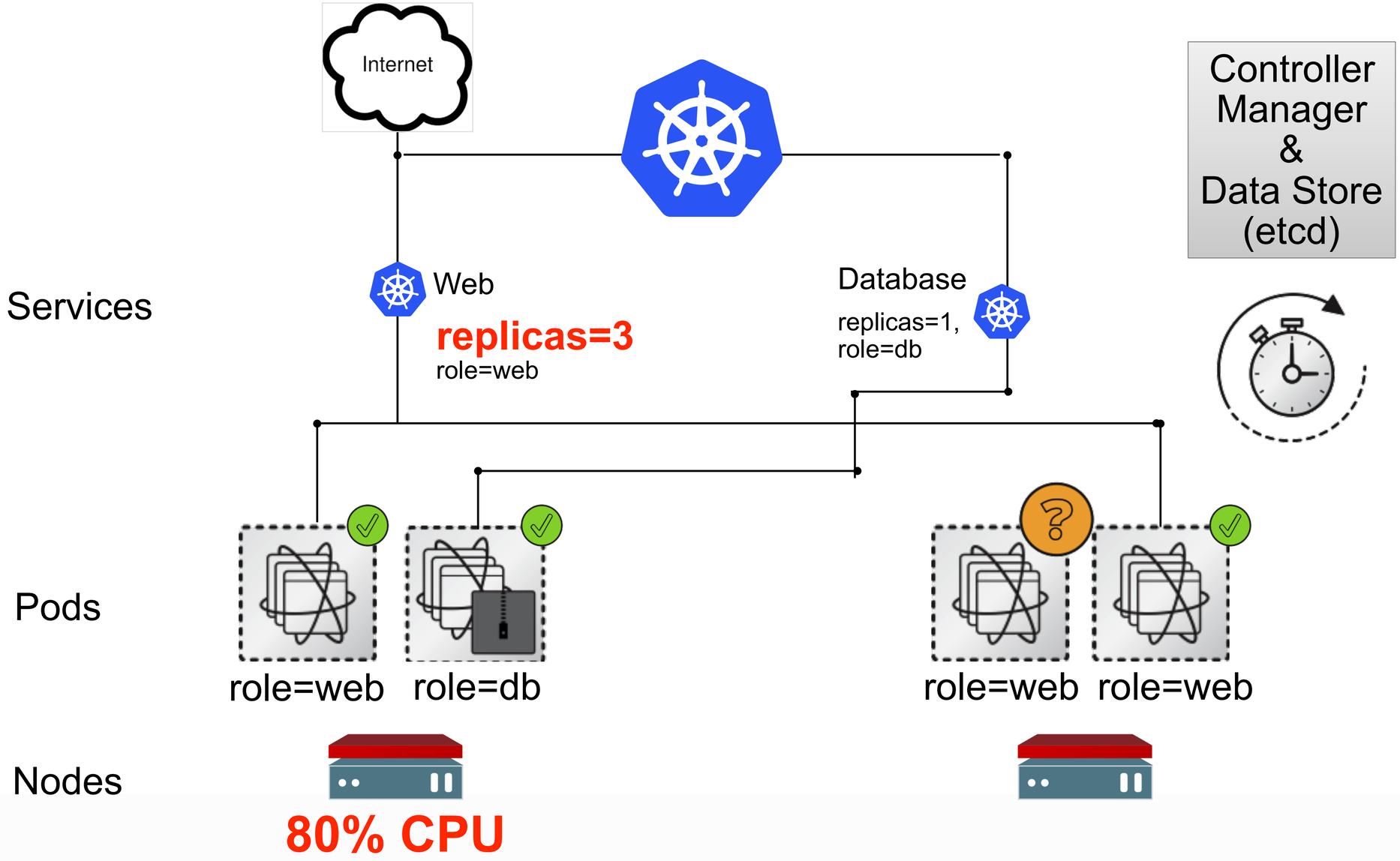
HEALTH CHECK



AUTO-SCALE

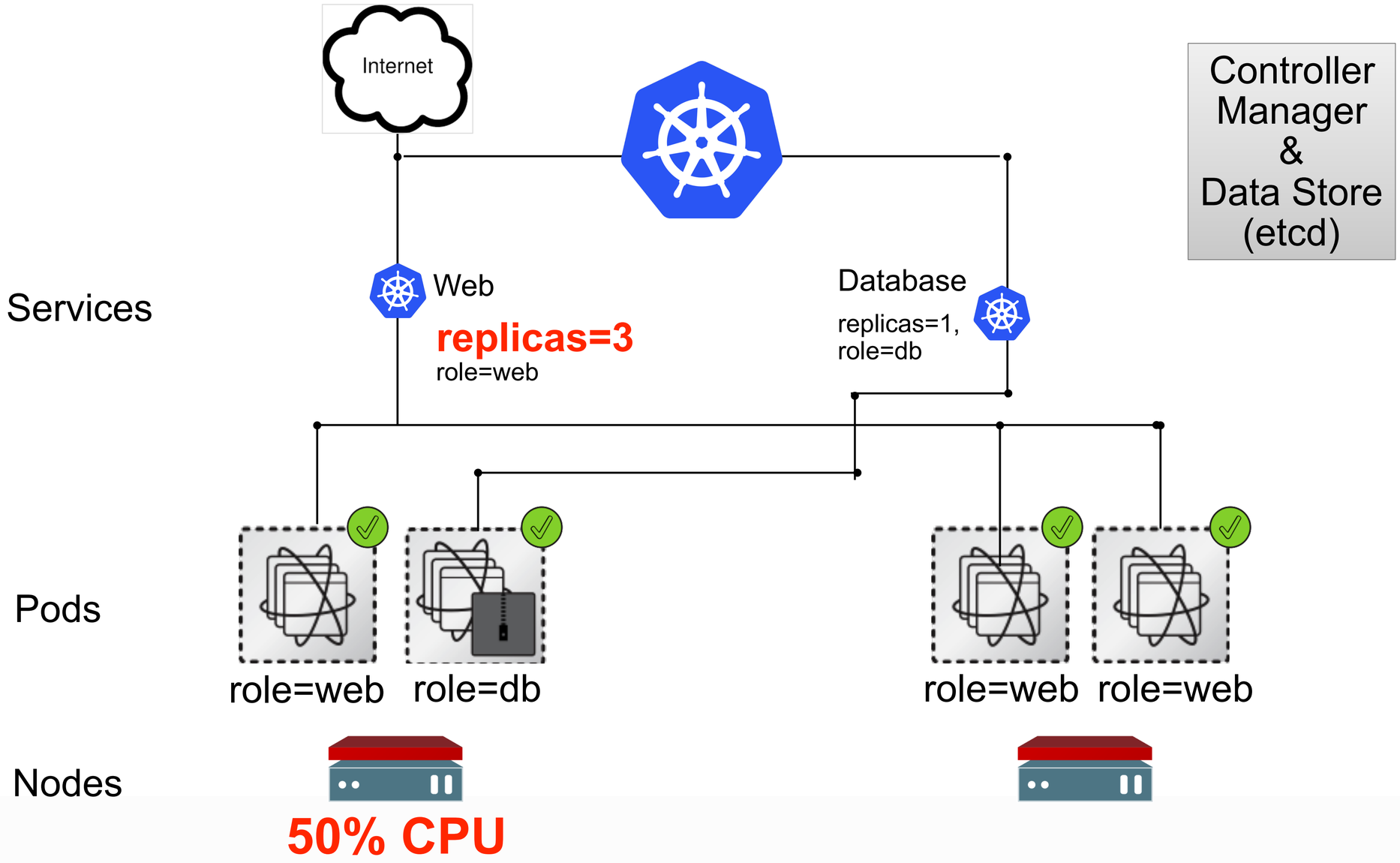


AUTO-SCALE



redhat.

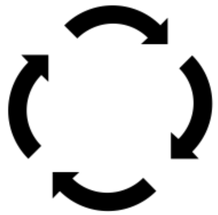
AUTO-SCALE



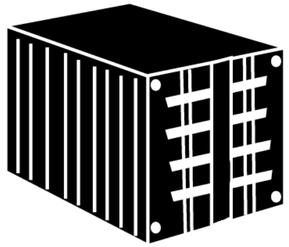
redhat.

CONTAINER SECURITY

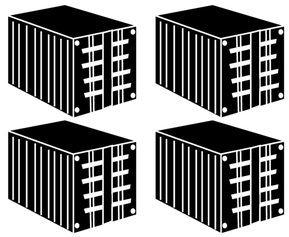
SECURING CONTAINERS



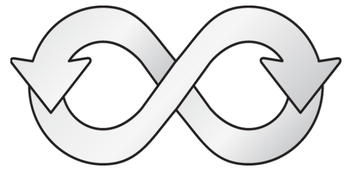
Builds



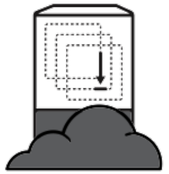
Images



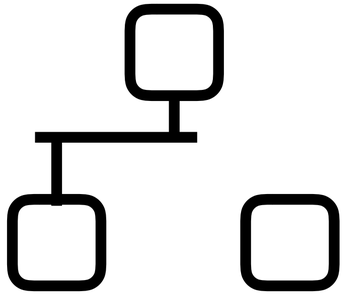
Registry



CI/CD

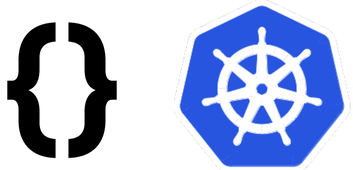


Container host



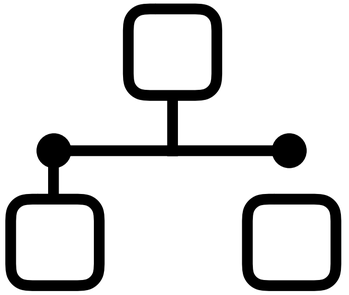
Network isolation

Storage



API & Platform access

Monitoring & Logging

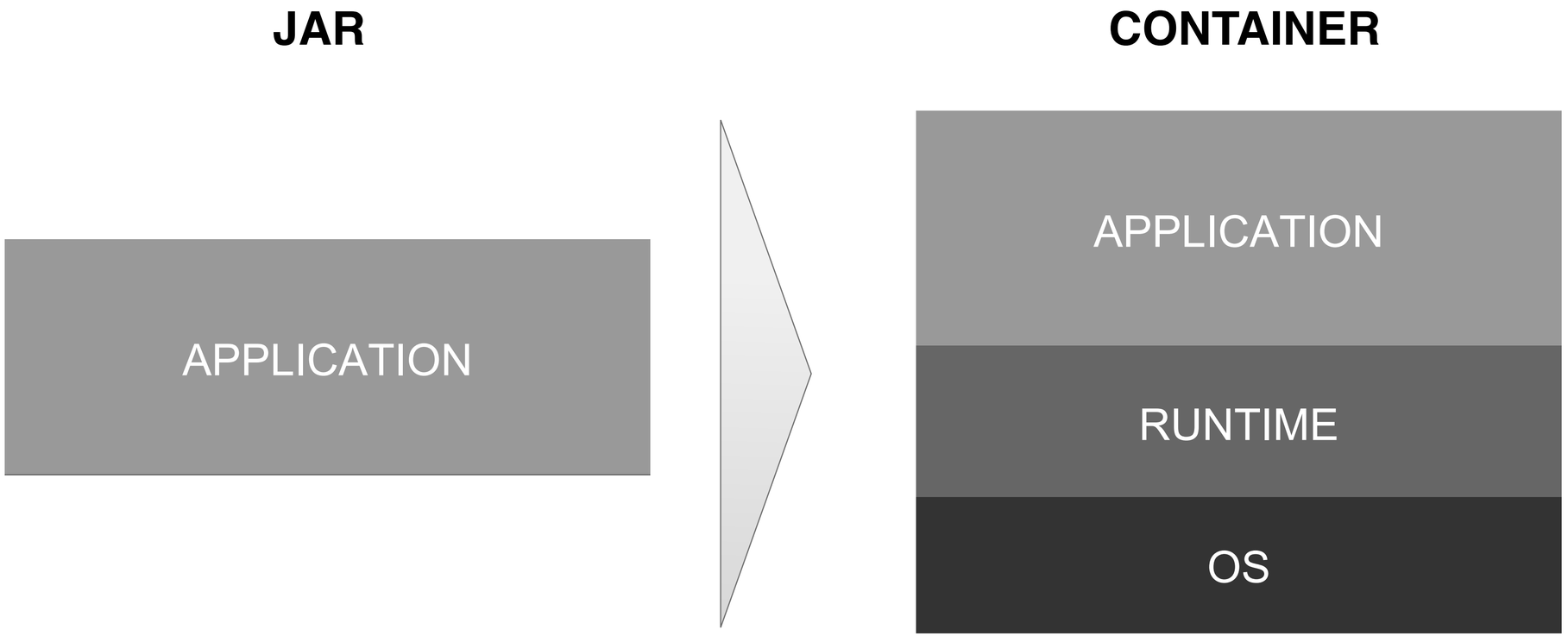


Federated clusters

redhat.

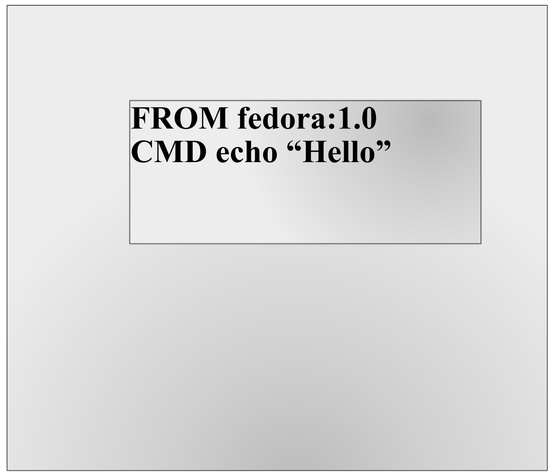
CONTAINER BUILDS

CONTENT: EACH LAYER MATTERS



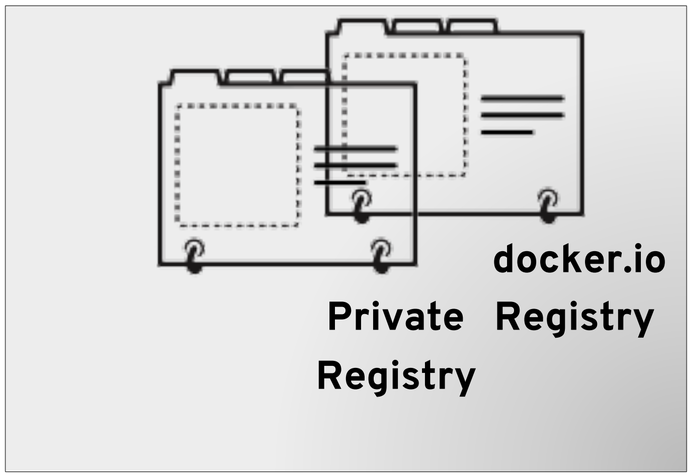
CONTAINER BUILDS

Build



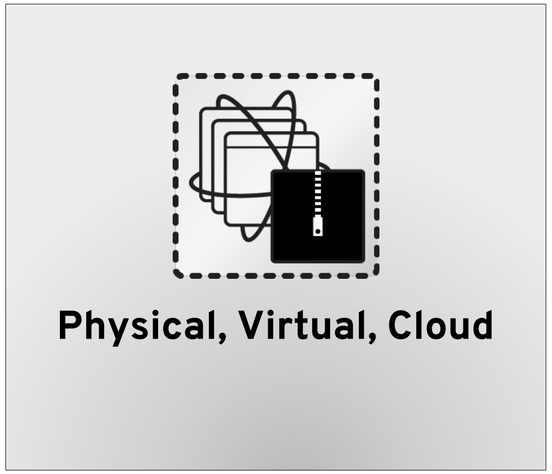
Build file

Ship



Image

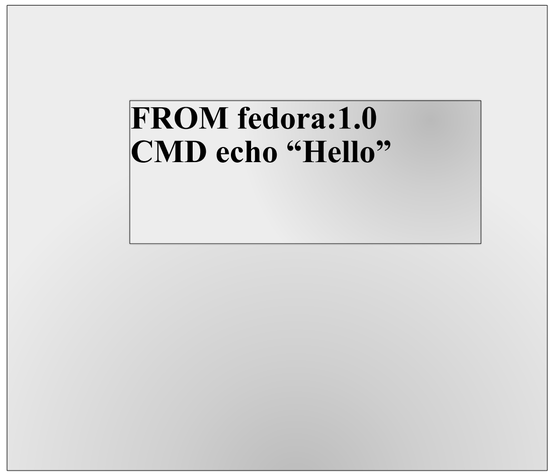
Run



Container

CONTAINER BUILDS

Build

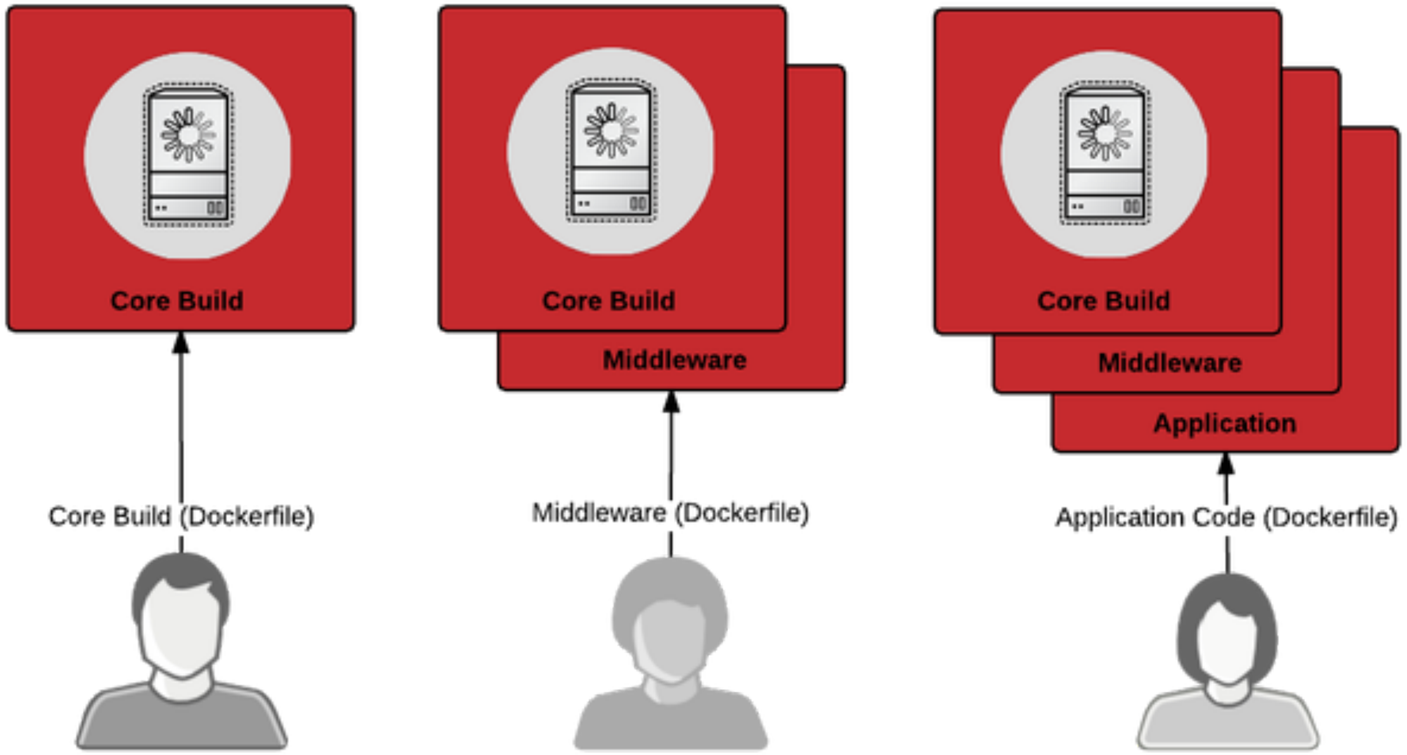


Build file

Best Practices

- Treat as a Blueprint
- Specify a user, defaults to root
- Don’t login to build/configure
- Version control build file
- Be explicit with versions, not latest
- Each Run creates a new layer

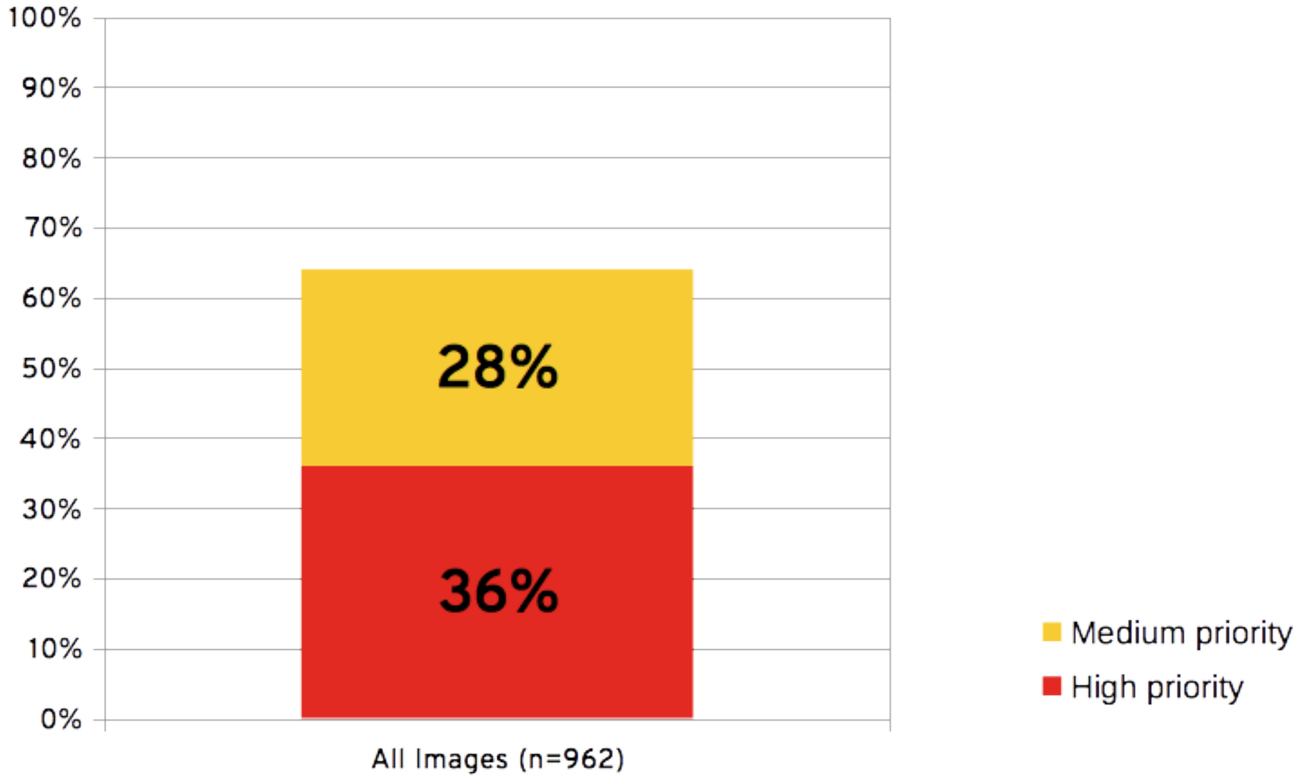
A CONVERGED SOFTWARE SUPPLY CHAIN



CONTAINER IMAGE SECURITY

WHAT'S INSIDE THE CONTAINER MATTERS

64% of official images in Docker Hub
contain high priority security vulnerabilities



examples:

ShellShock (bash)

Heartbleed (OpenSSL)

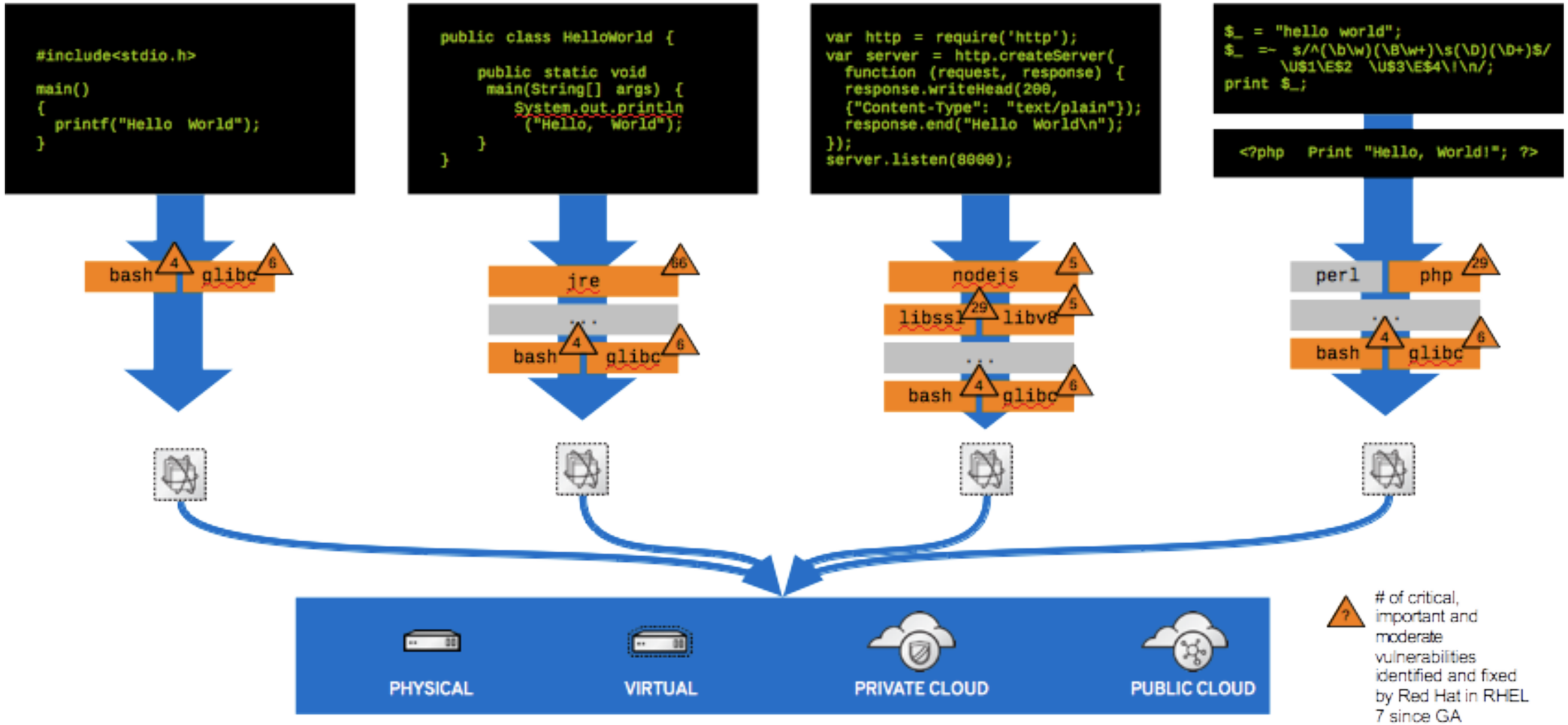
Poodle (OpenSSL)

Source: *Over 30% of Official Images in Docker Hub Contain High Priority Security Vulnerabilities*, Jayanth Gummaraju, Tarun Desikan, and Yoshio Turner, BanyanOps, May 2015 (http://www.banyanops.com/pdf/BanyanOps-AnalyzingDockerHub-WhitePaper.pdf)

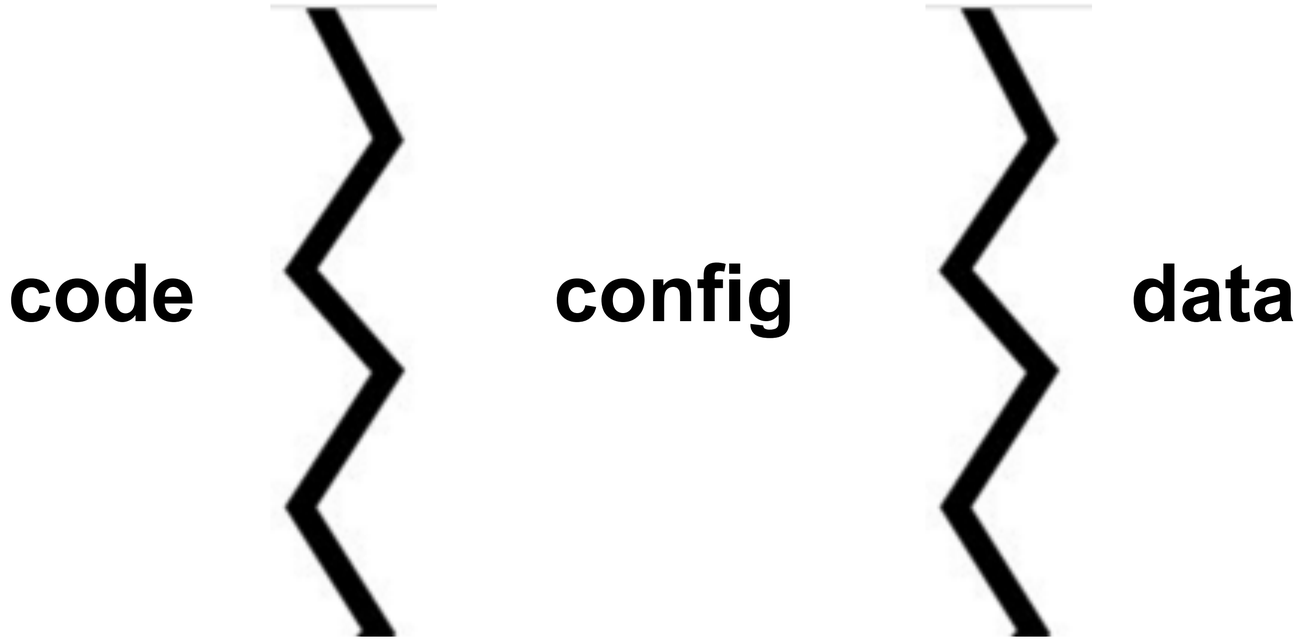
redhat.

SECURITY IMPLICATIONS

What's inside matters…



TREAT CONTAINERS AS IMMUTABLE



Container image

Kubernetes configmaps secrets

Traditional data services, Kubernetes persistent volumes

redhat.

CONTAINER REGISTRY SECURITY

PRIVATE REGISTRY

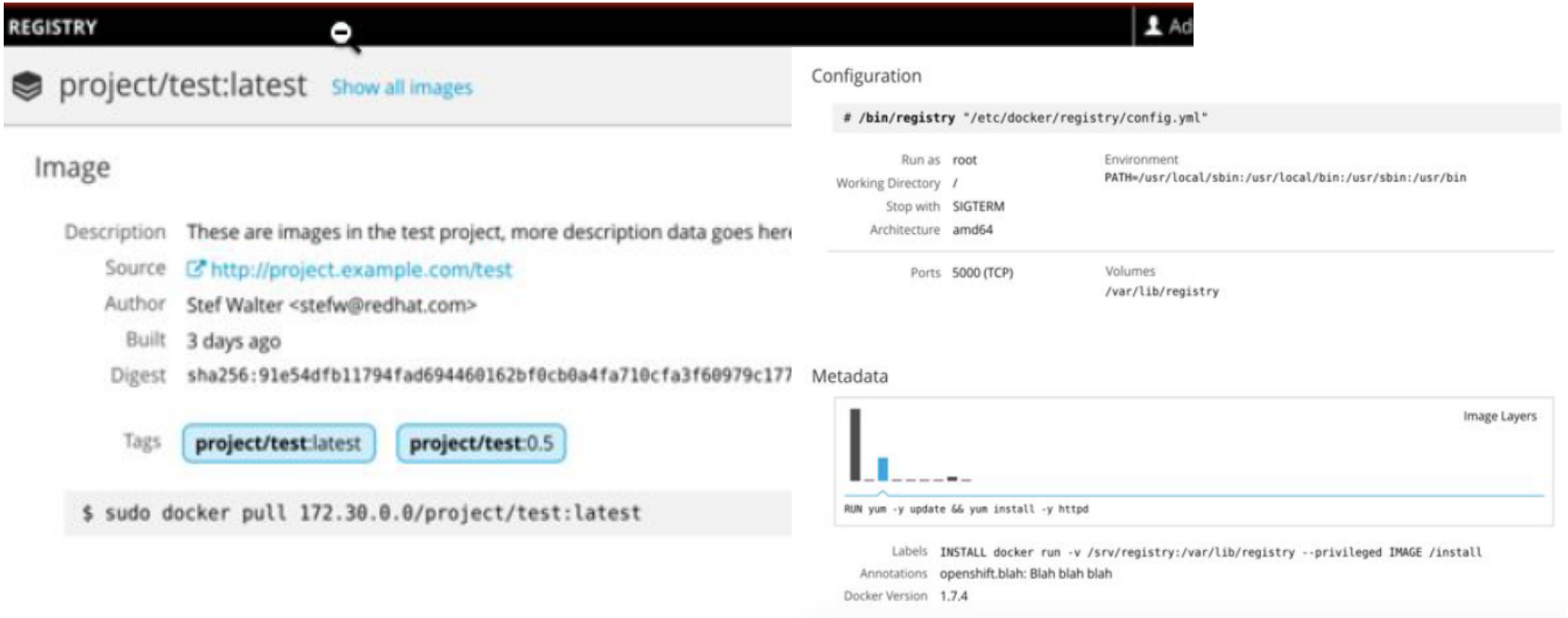
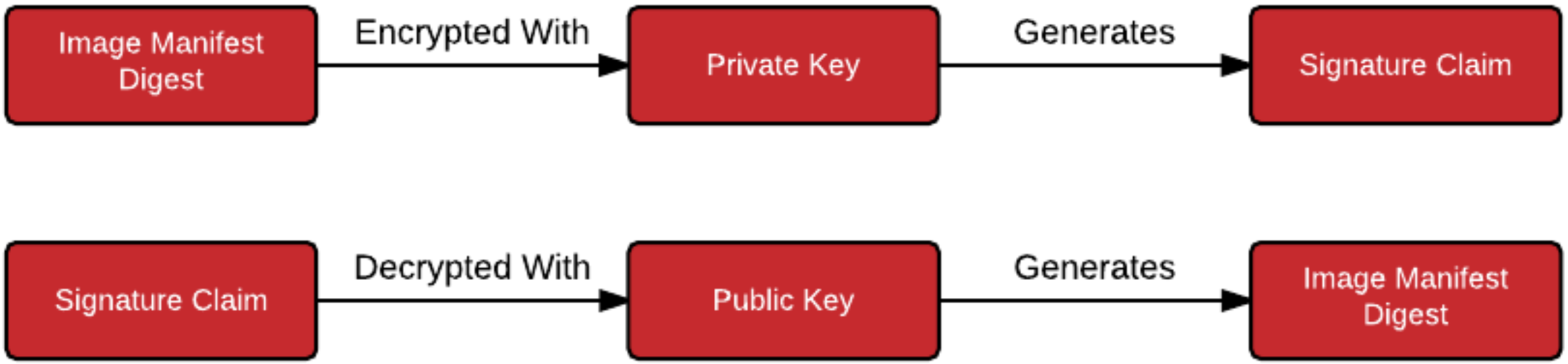


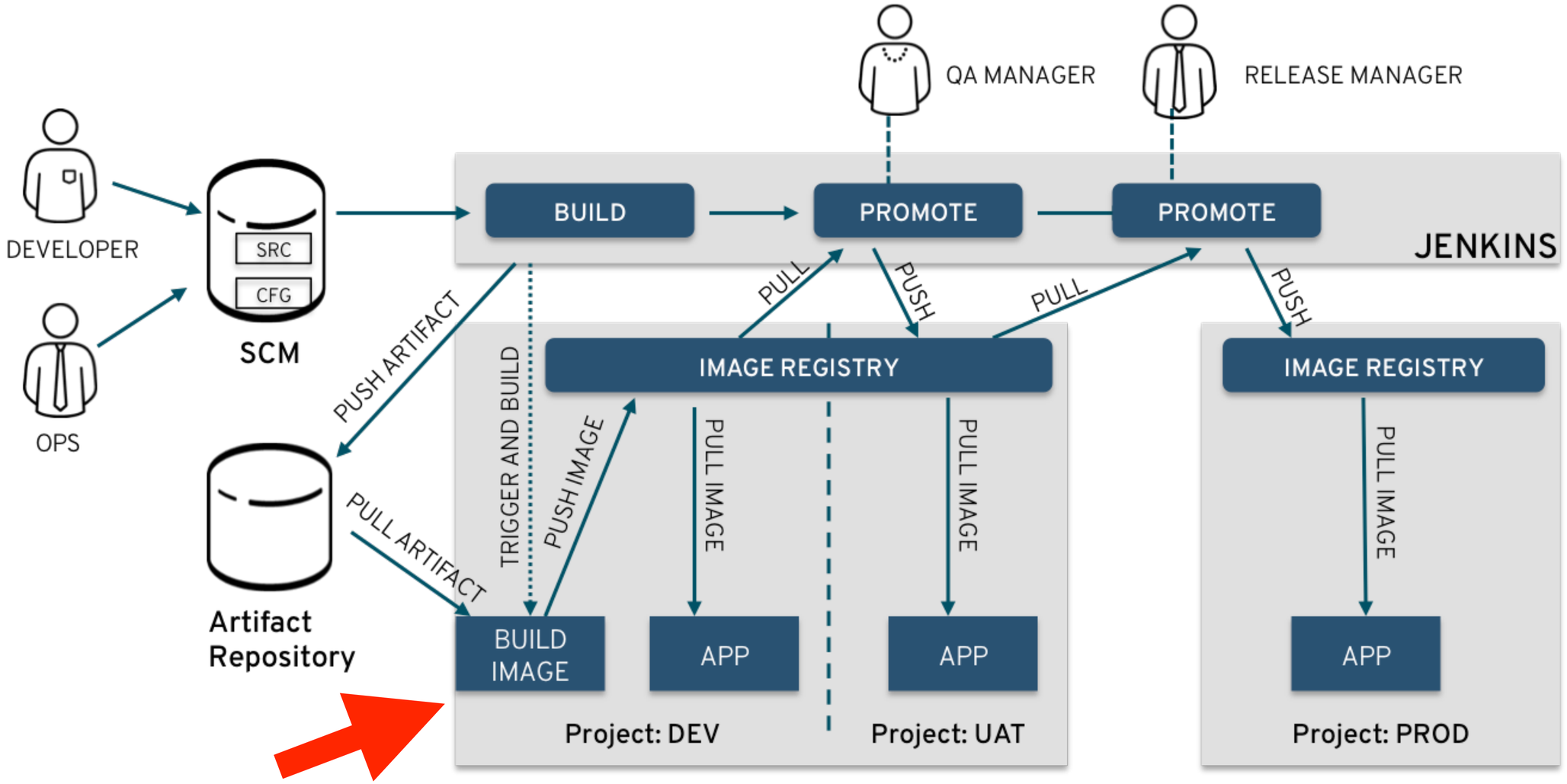
IMAGE SIGNING

Validate what images and version are running



CONTINUOUS INTEGRATION WITH CONTAINERS

CONTINUOUS INTEGRATION + SECURITY



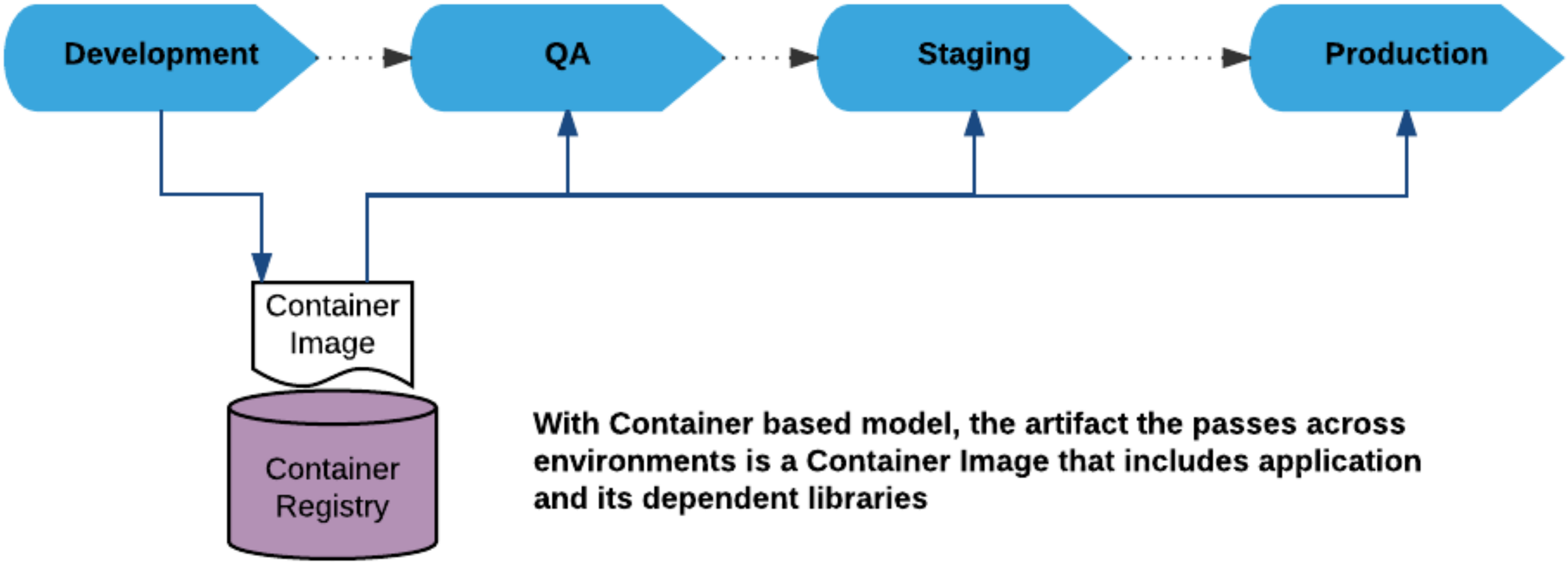
redhat.

CONTINUOUS INTEGRATION WITH SECURITY SCAN

redhat.

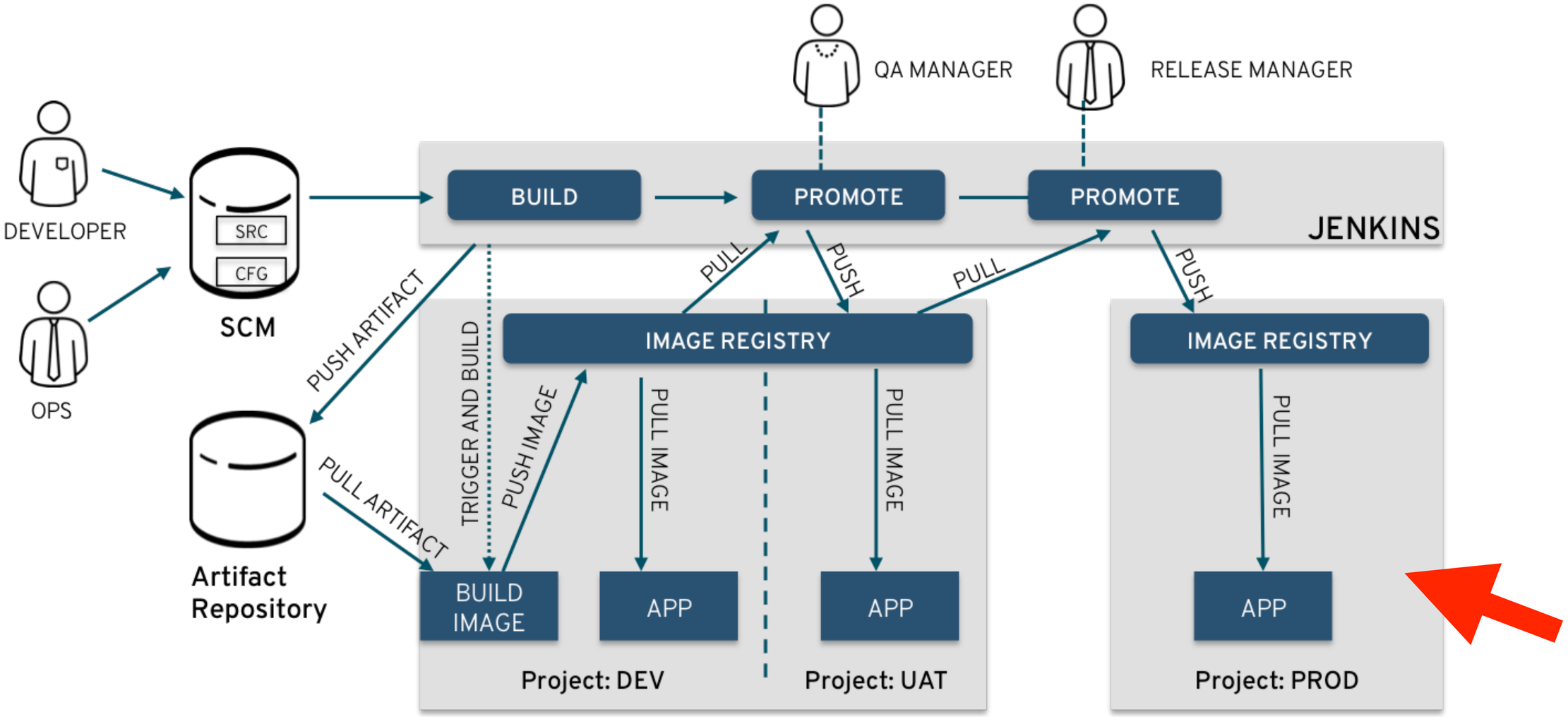
CONTINUOUS DELIVERY WITH CONTAINERS

CONTINUOUS DELIVERY WITH CONTAINERS



With Container based model, the artifact the passes across environments is a Container Image that includes application and its dependent libraries

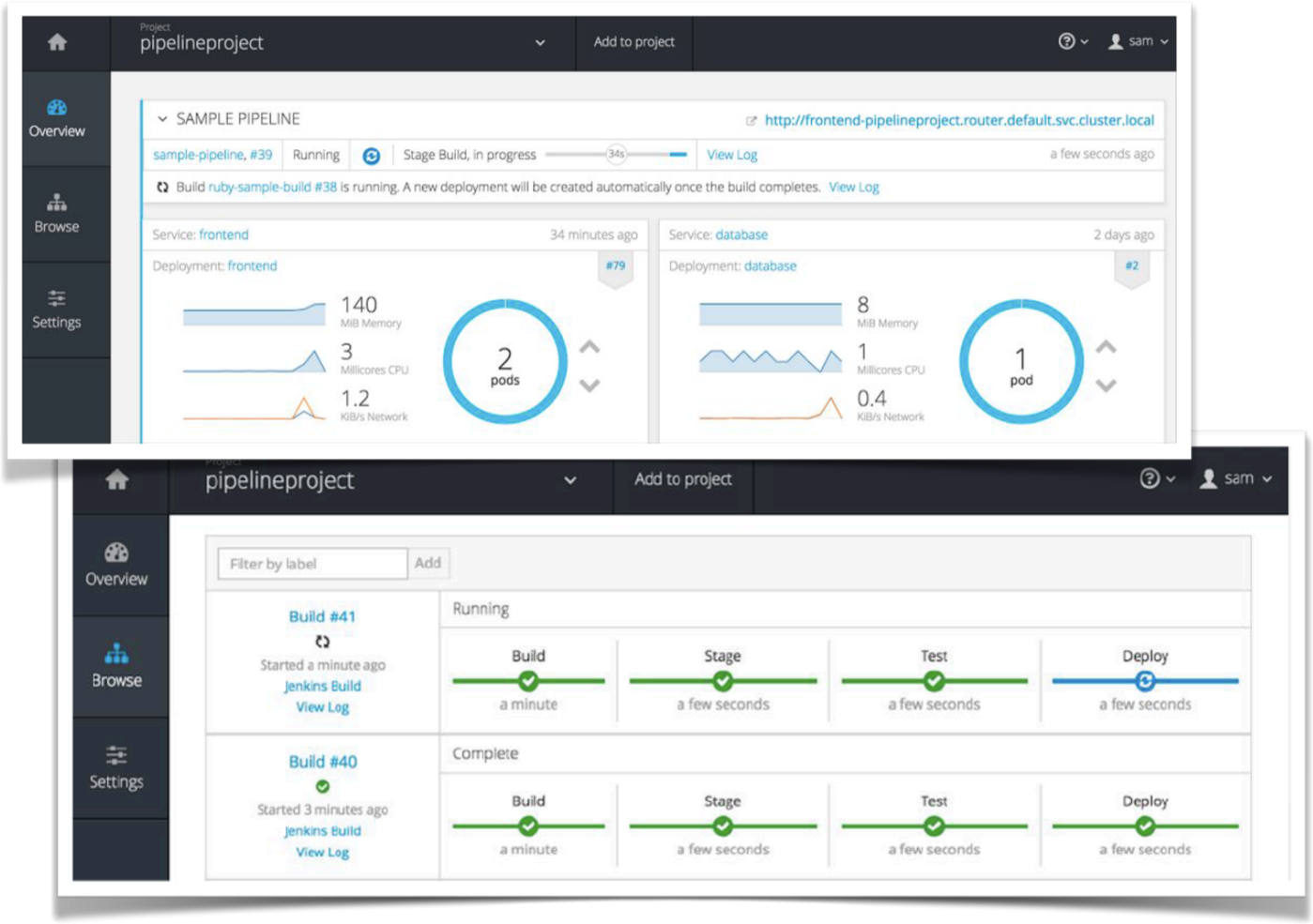
CONTINUOUS DELIVERY + SECURITY



redhat.

CONTINUOUS DELIVERY: DEPLOYMENT STRATEGIES

CONTINUOUS DELIVERY DEPLOYMENT STRATEGIES



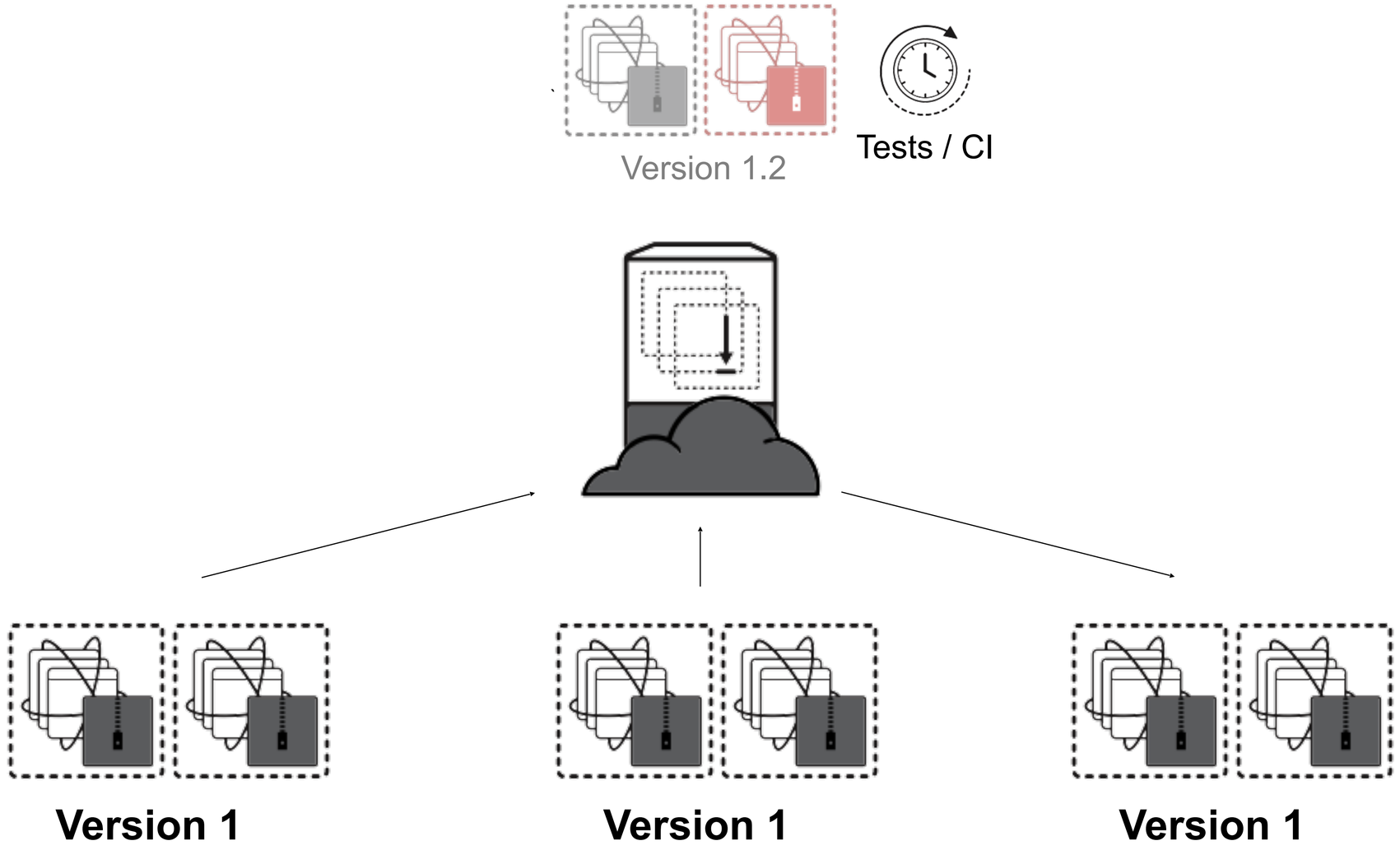
DEPLOYMENT STRATEGIES

- Recreate
- Rolling updates
- Blue / Green deployment

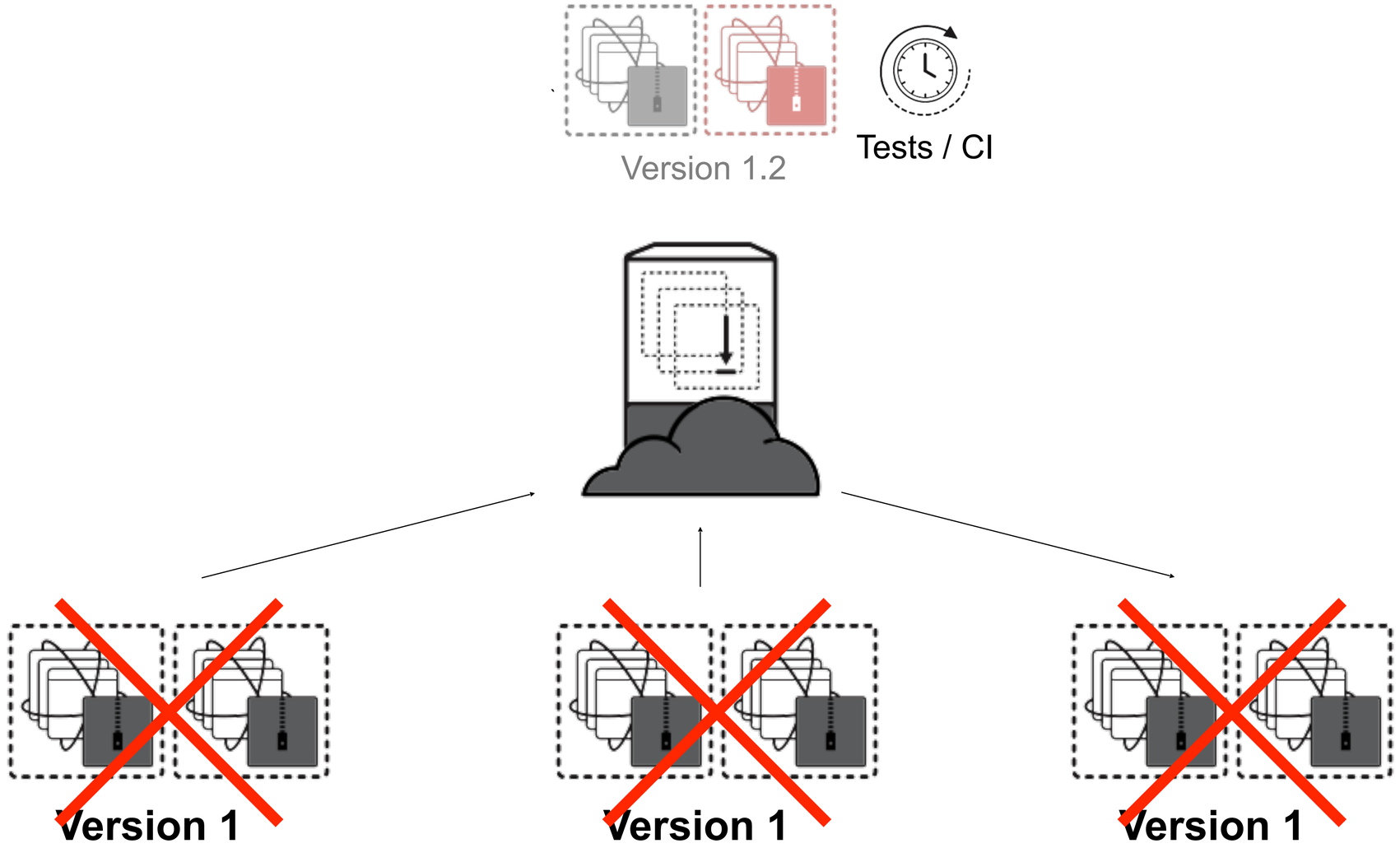
redhat.

Recreate

RECREATE WITH DOWNTIME



RECREATE WITH DOWNTIME

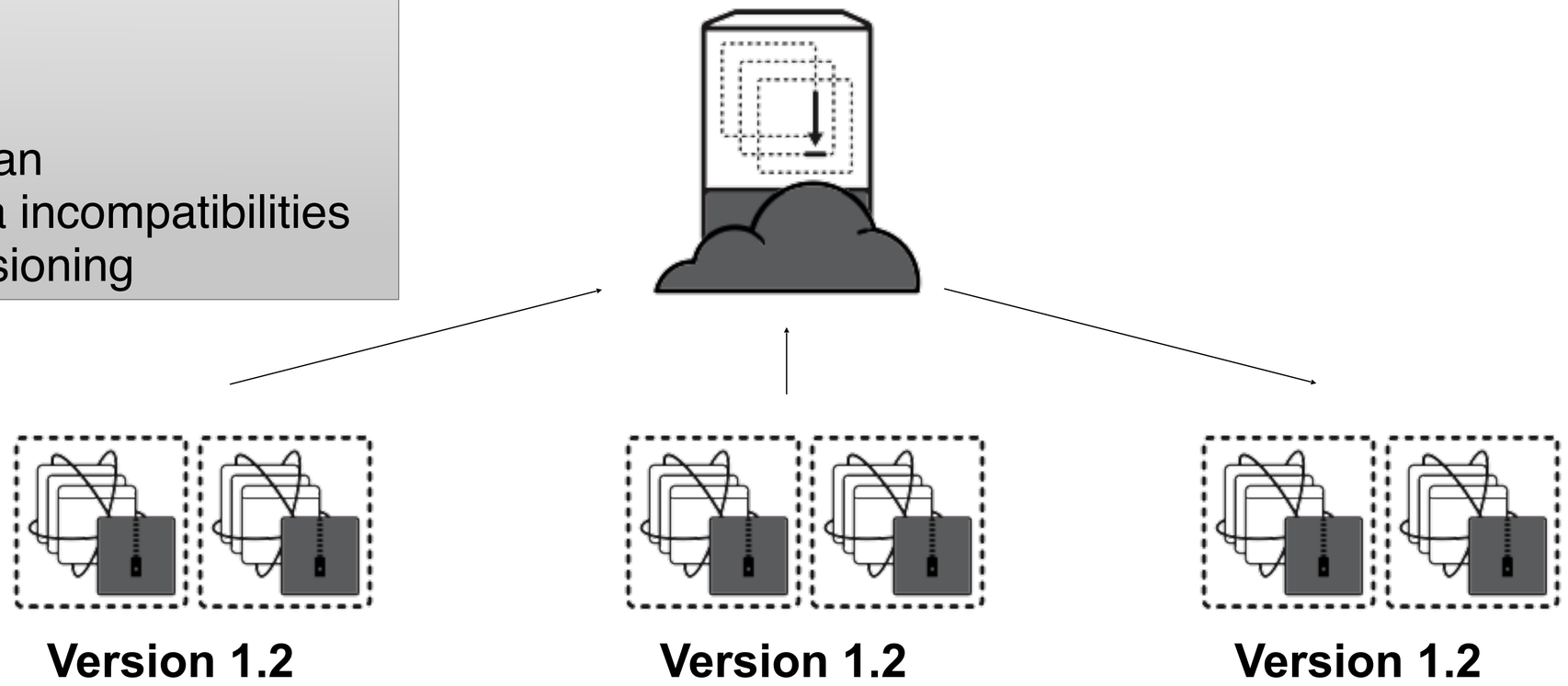


RECREATE WITH DOWNTIME

Use Case
- Non-mission critical services

Cons
- Downtime

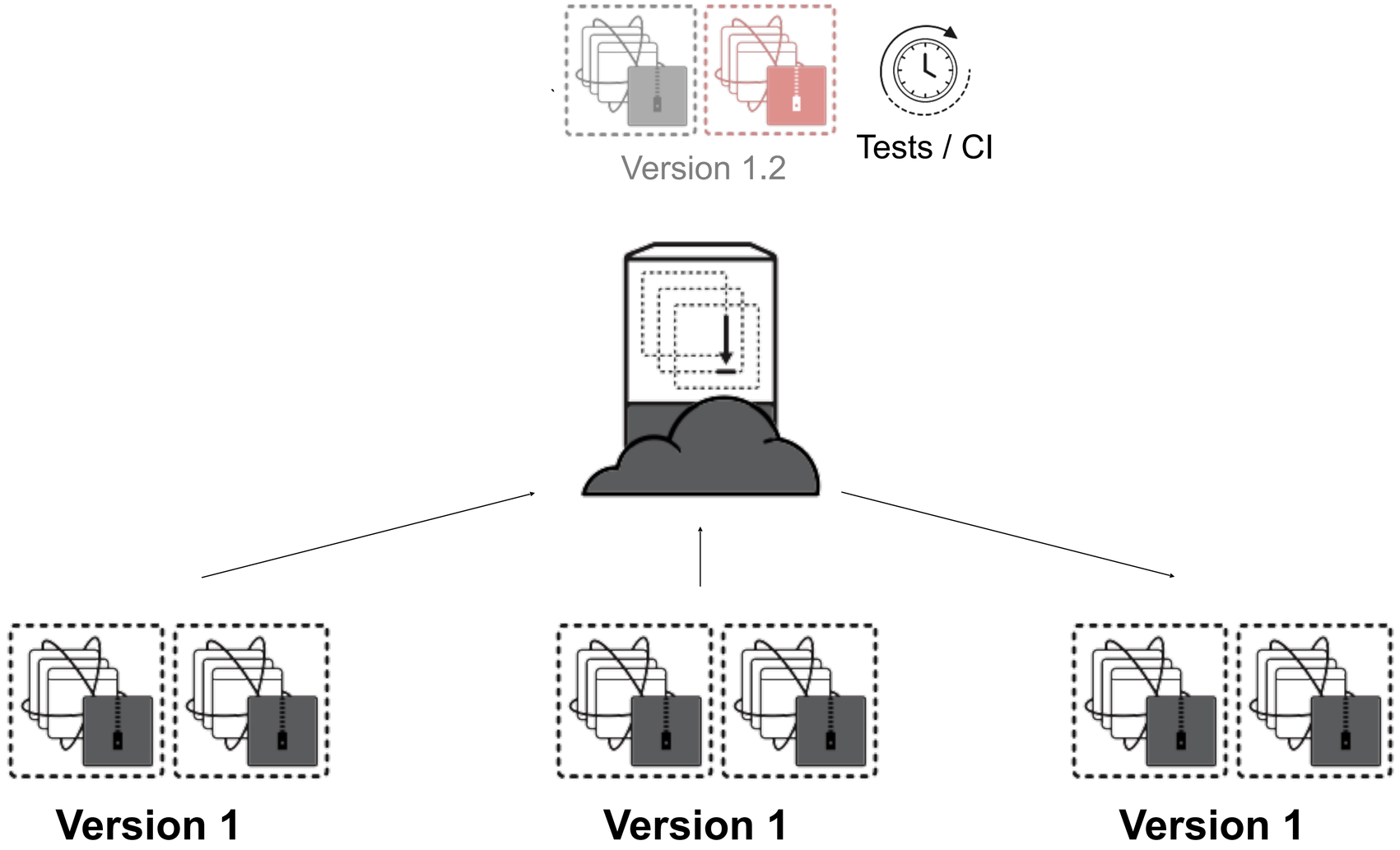
Pros
- Simple, clean
- No Schema incompatibilities
- No API versioning



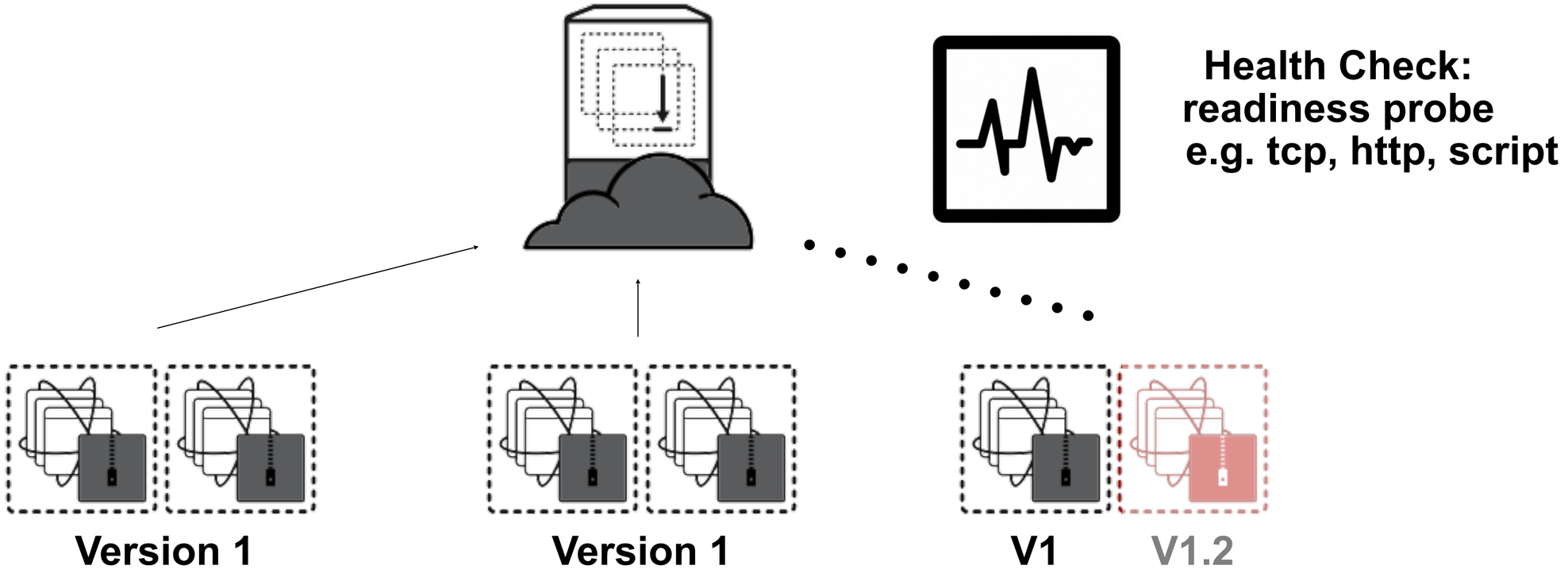
redhat.

Rolling Updates

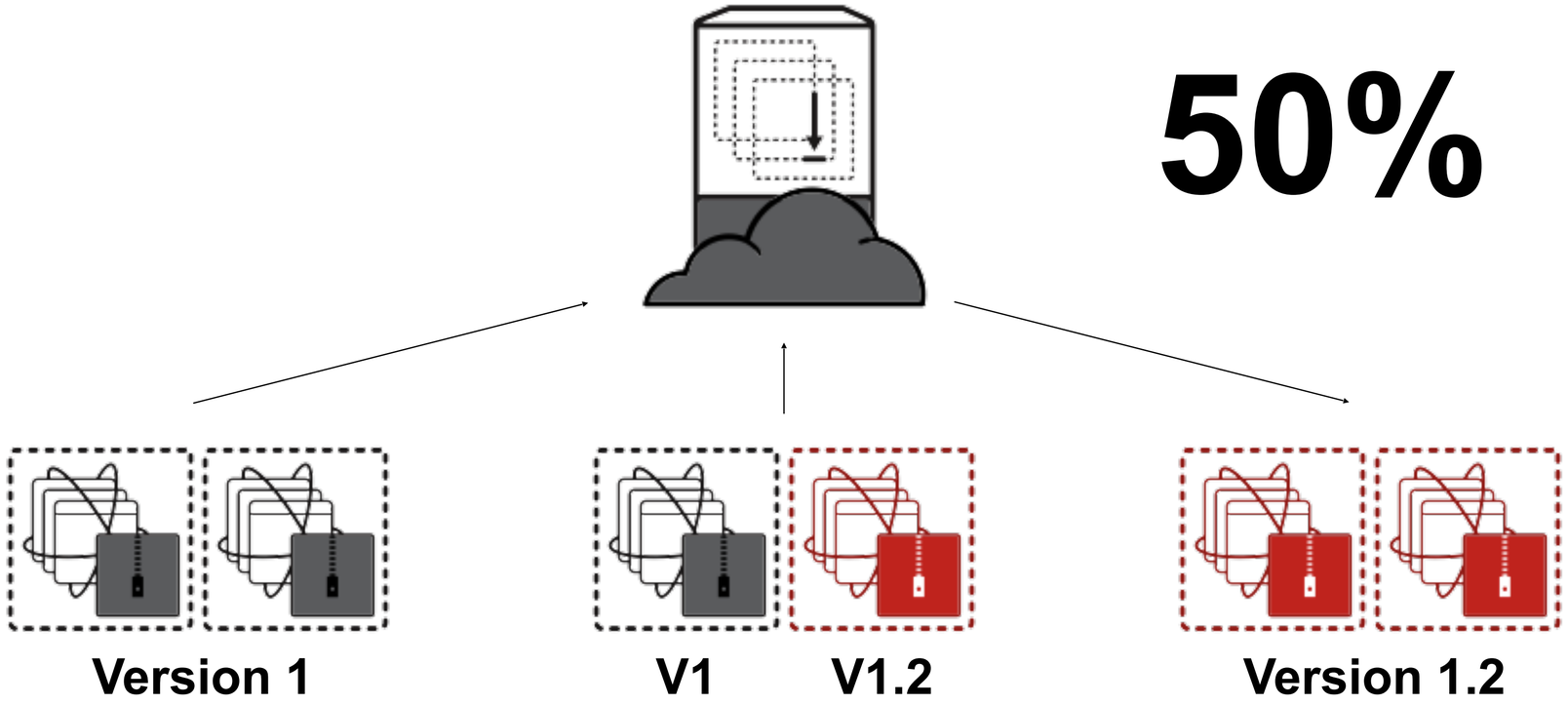
ROLLING UPDATES with ZERO DOWNTIME



Deploy new version and wait until it’s ready…



Each container/pod is updated one by one

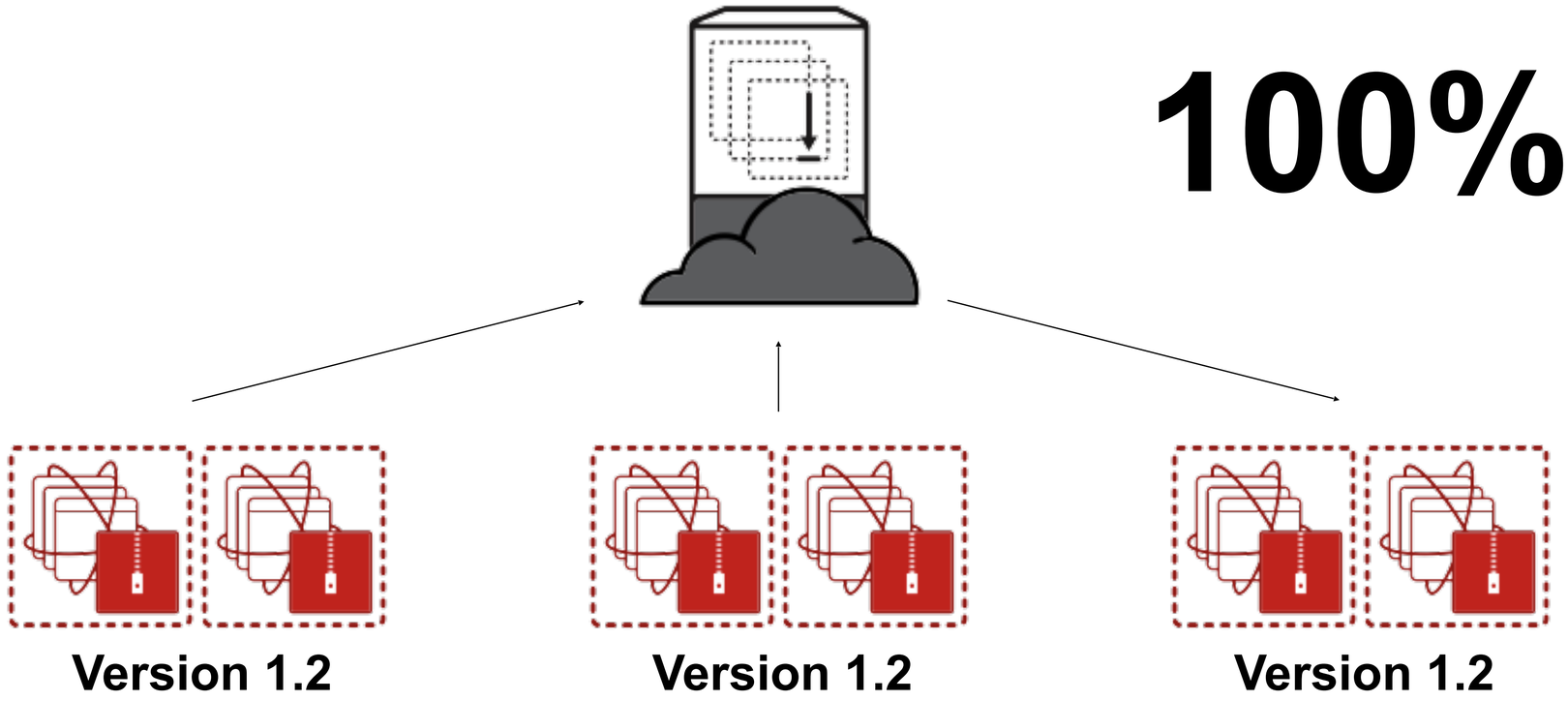


Each container/pod is updated one by one

Use Case
- Horizontally scaled
- Backward compatible API/data
- Microservices

Cons
- Require backward compatible APIs/data
- Resource overhead

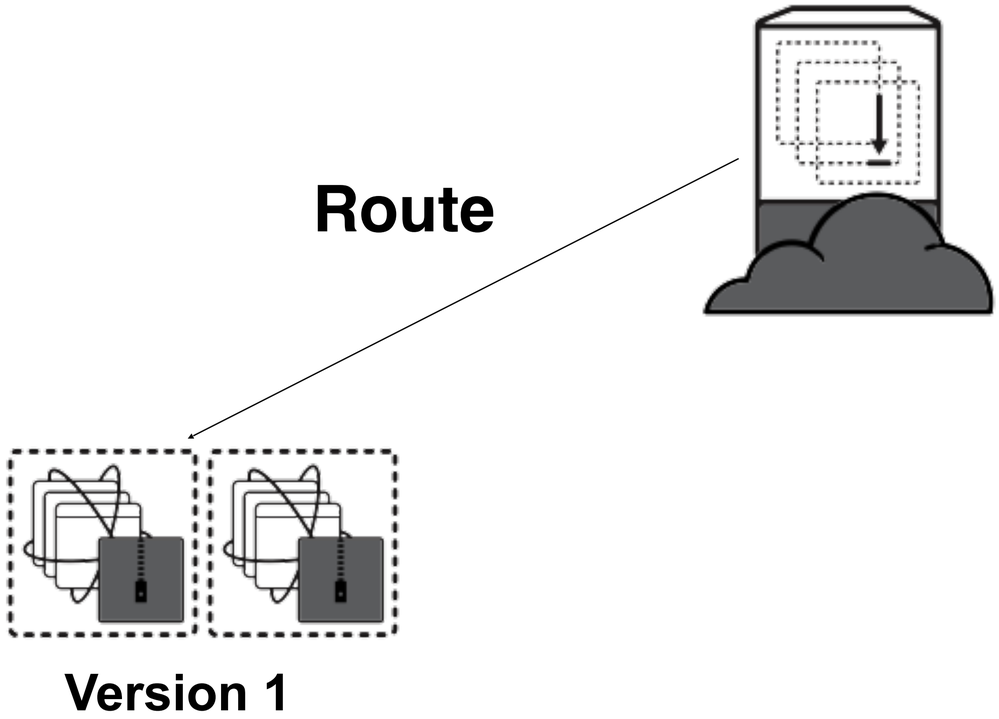
Pros
- Zero downtime
- Reduced risk, gradual rollout w/health checks
- Ready for rollback



redhat.

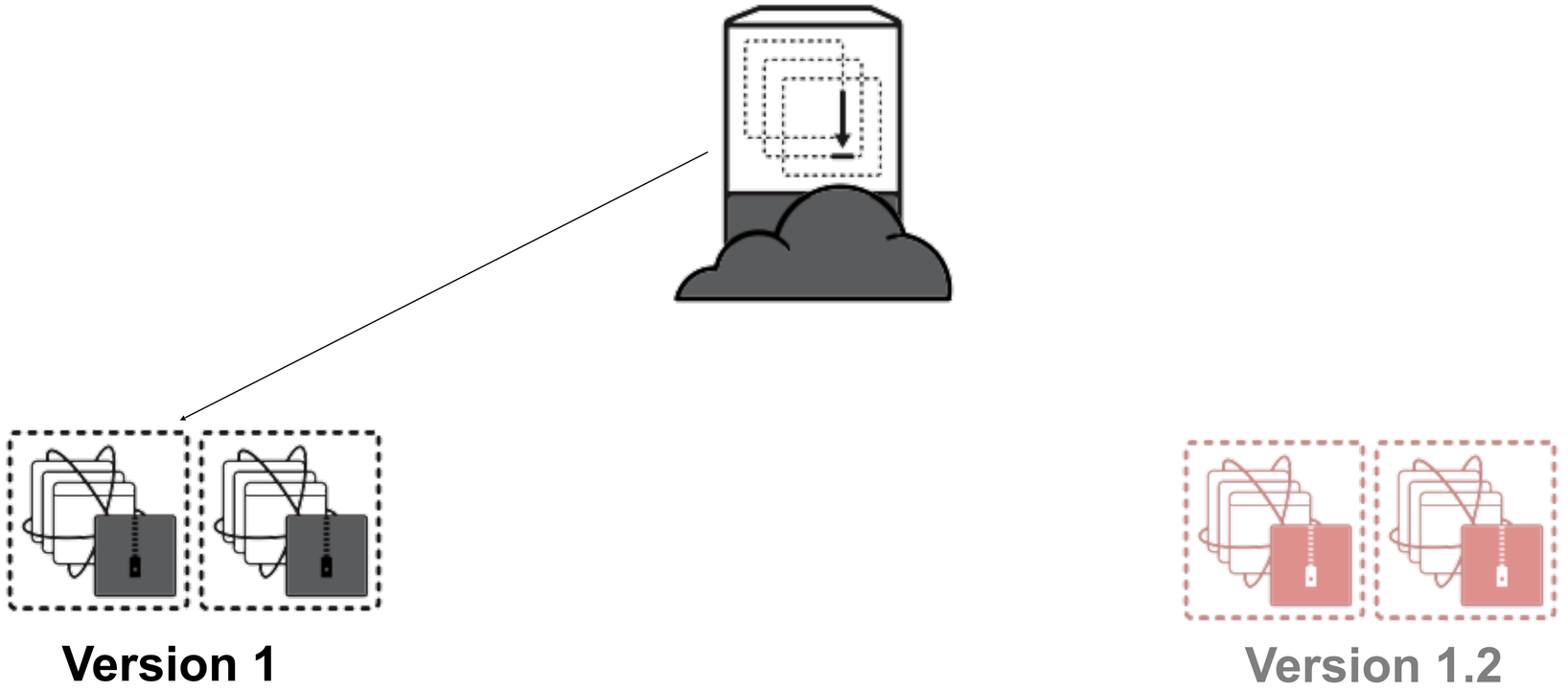
Blue / Green Deployment

BLUE / GREEN DEPLOYMENT



BLUE

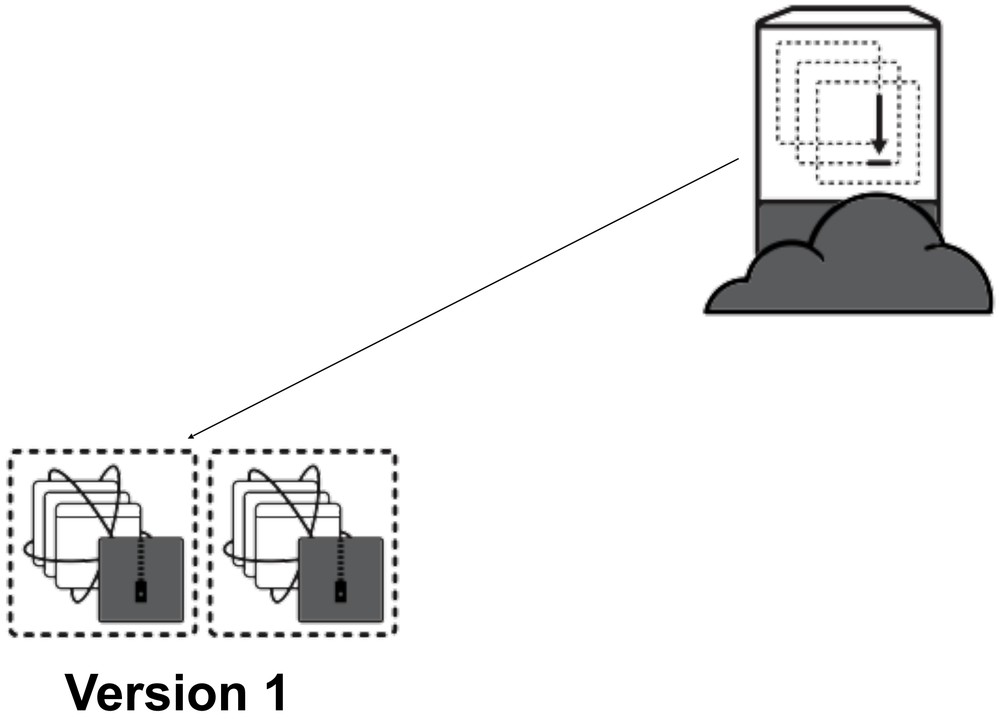
BLUE / GREEN DEPLOYMENT



BLUE

GREEN

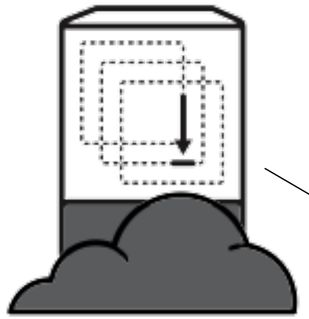
BLUE / GREEN DEPLOYMENT



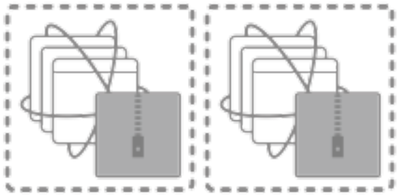
BLUE

GREEN

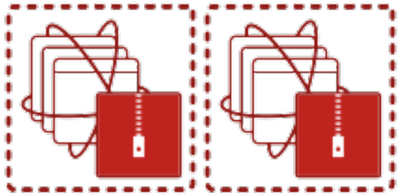
BLUE / GREEN DEPLOYMENT



Route



Version 1

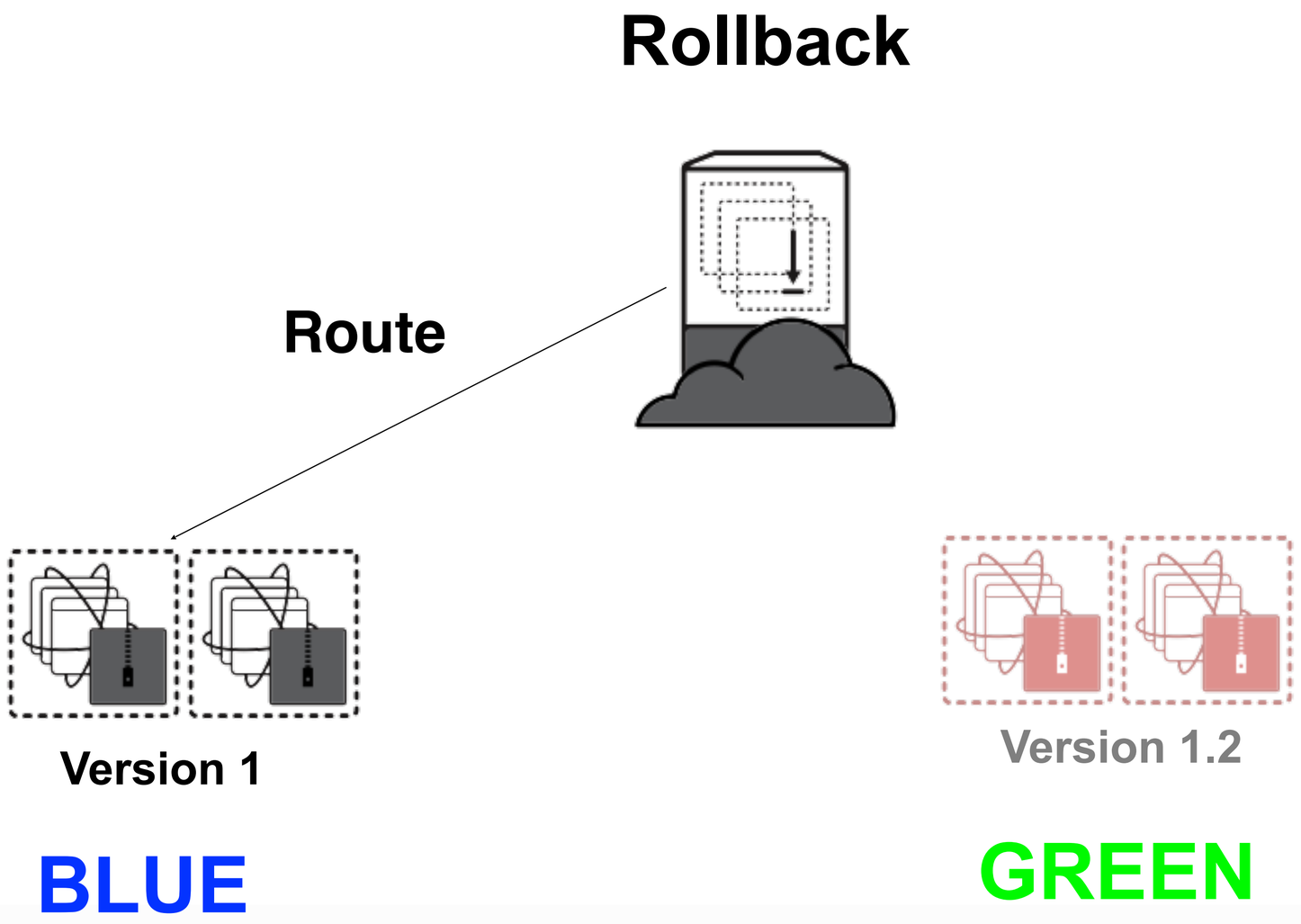


Version 1.2

BLUE

GREEN

BLUE / GREEN DEPLOYMENT



Use Case
- Self-contained micro services (data)

Cons
- Resource overhead
- Data synchronization

Pros
- Low risk, never change production
- No downtime
- Production like testing
- Rollback

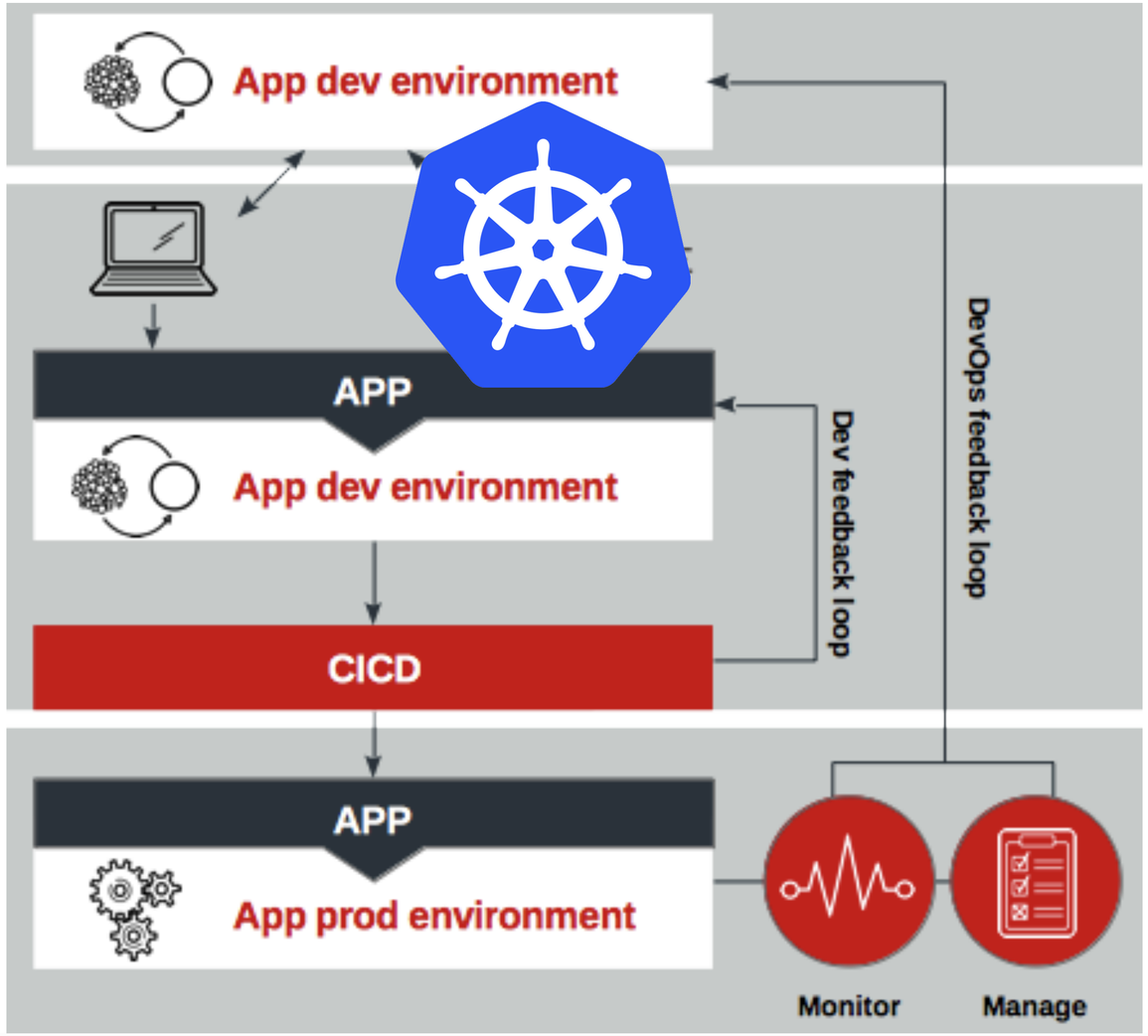
RAPID INNOVATION & EXPERIMENTATION

MICROSERVICES
RAPID INNNOVATION & EXPERIMENTATION

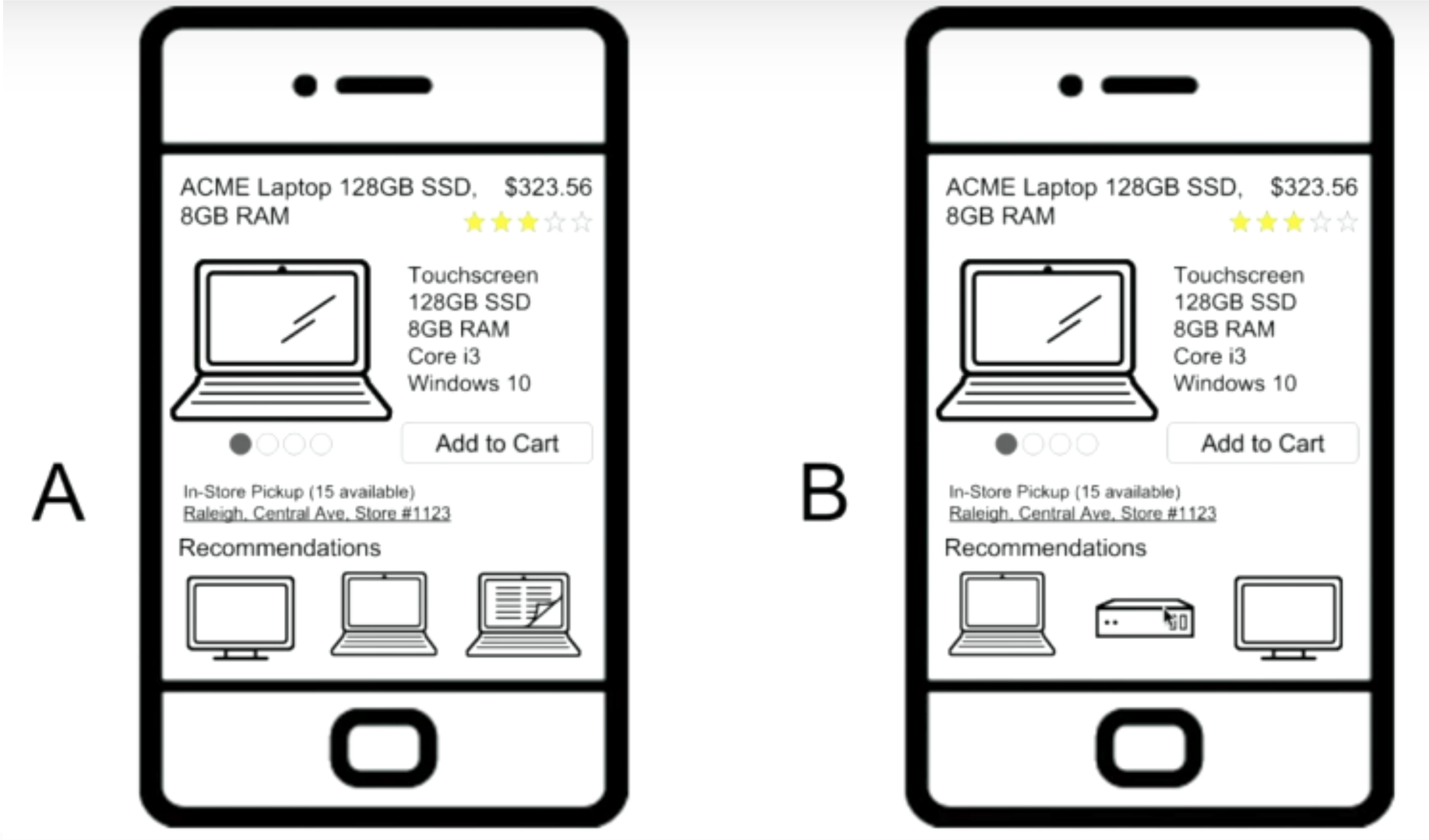
”only about 1/3 of ideas improve the metrics they were designed to improve.”

Ronny Kohavi, Microsoft (Amazon)

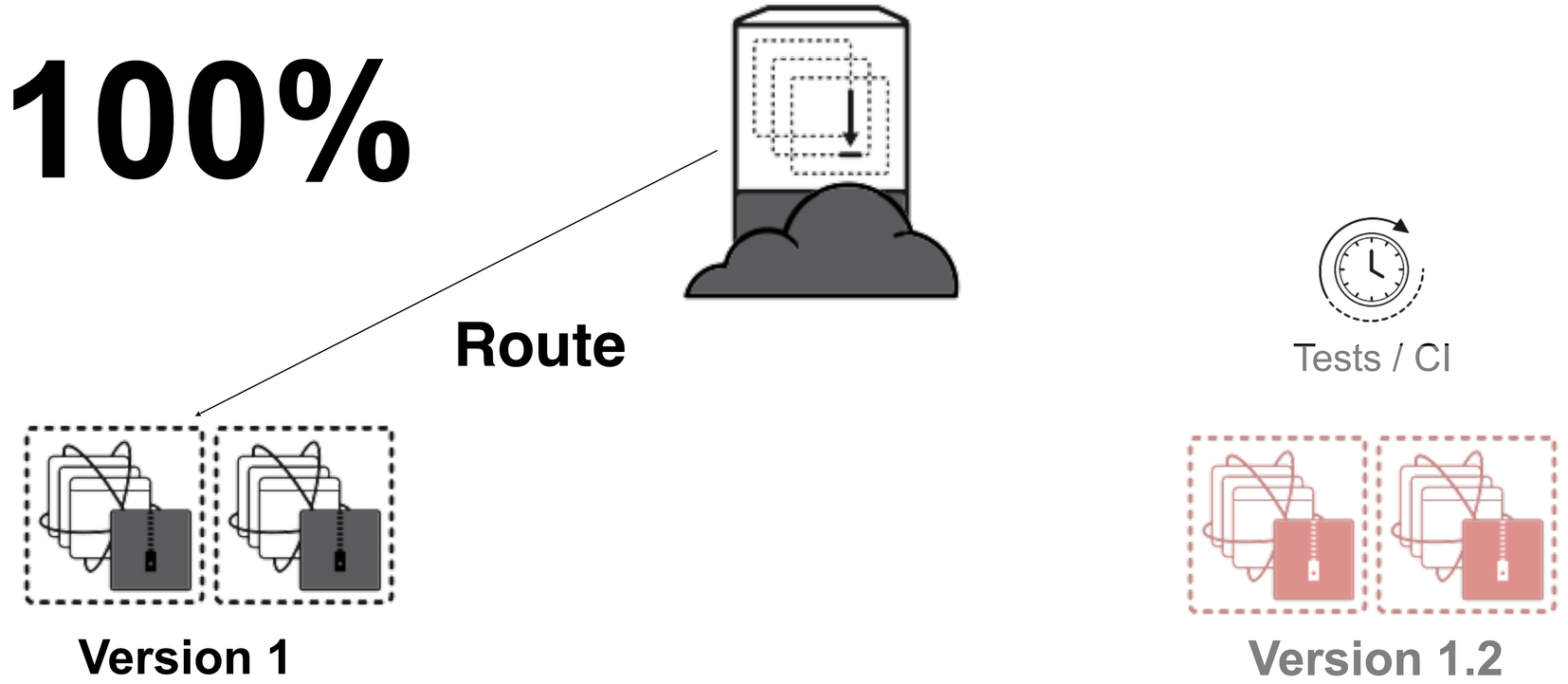
CONTINUOUS FEEDBACK LOOP



A/B TESTING USING CANARY DEPLOYMENTS



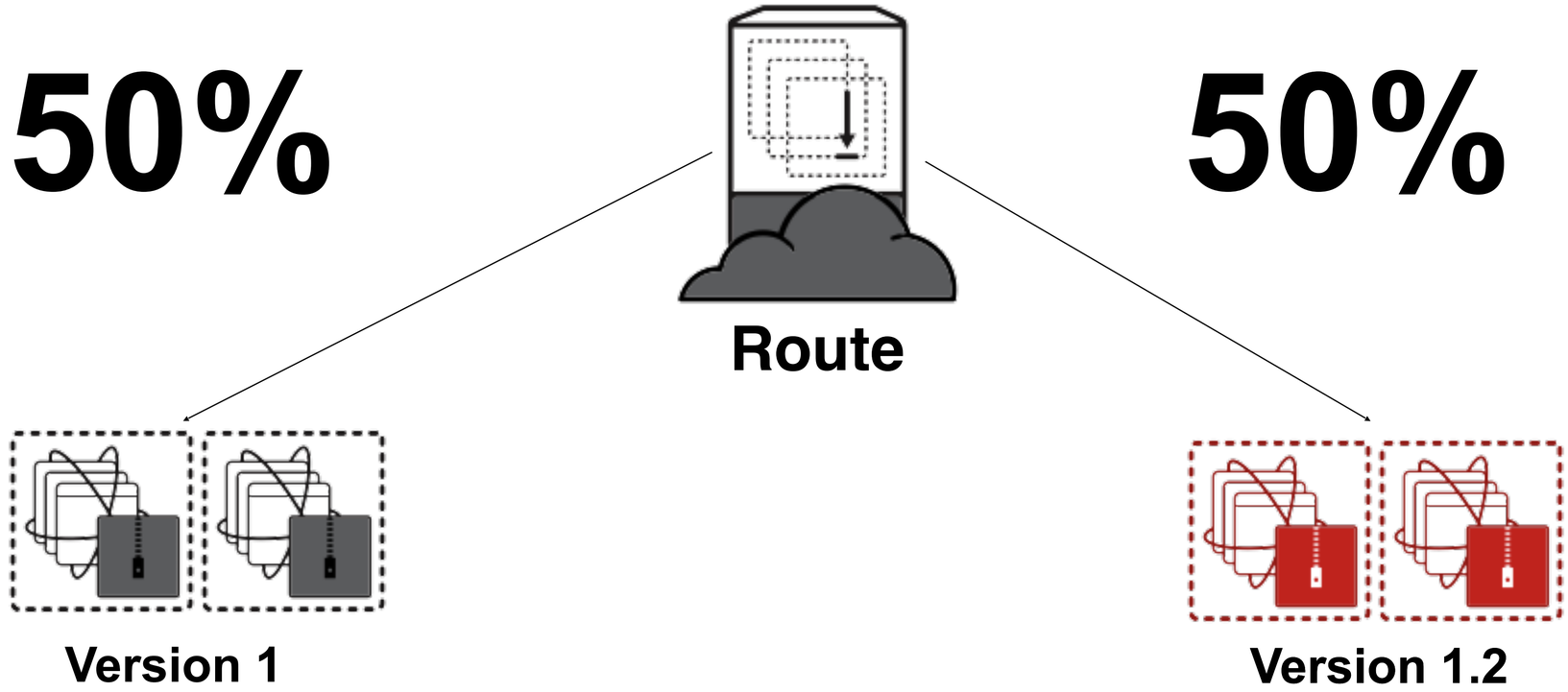
CANARY DEPLOYMENTS



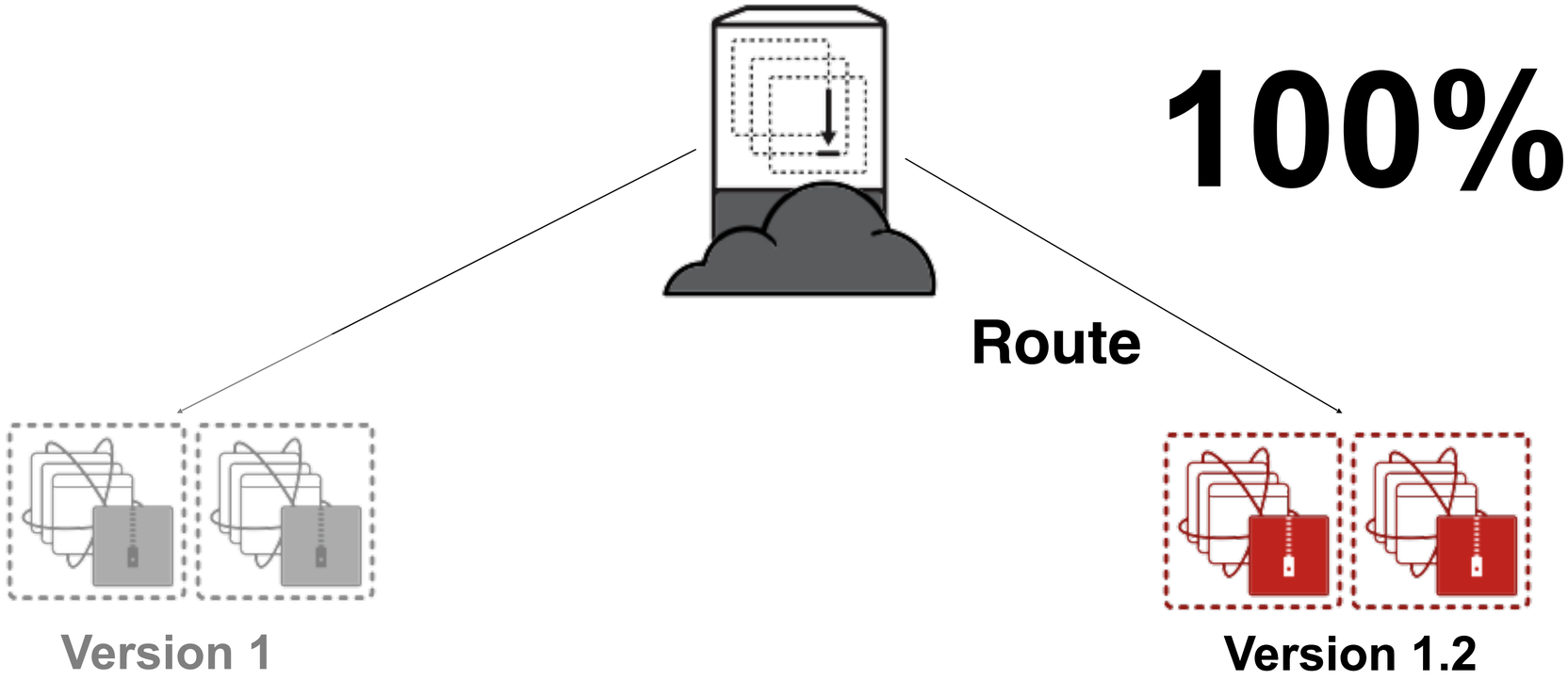
25% Conversion Rate

?! Conversion Rate

CANARY DEPLOYMENTS



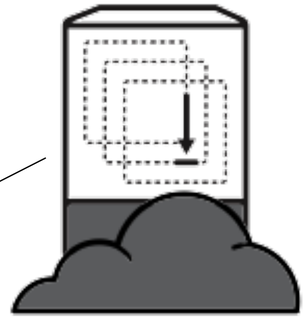
CANARY DEPLOYMENTS



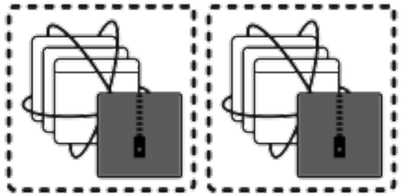
CANARY DEPLOYMENTS

Rollback

100%

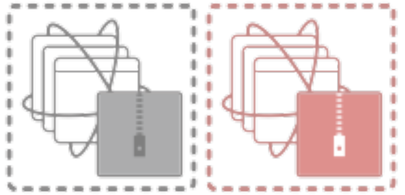


Route



Version 1

25% Conversion Rate

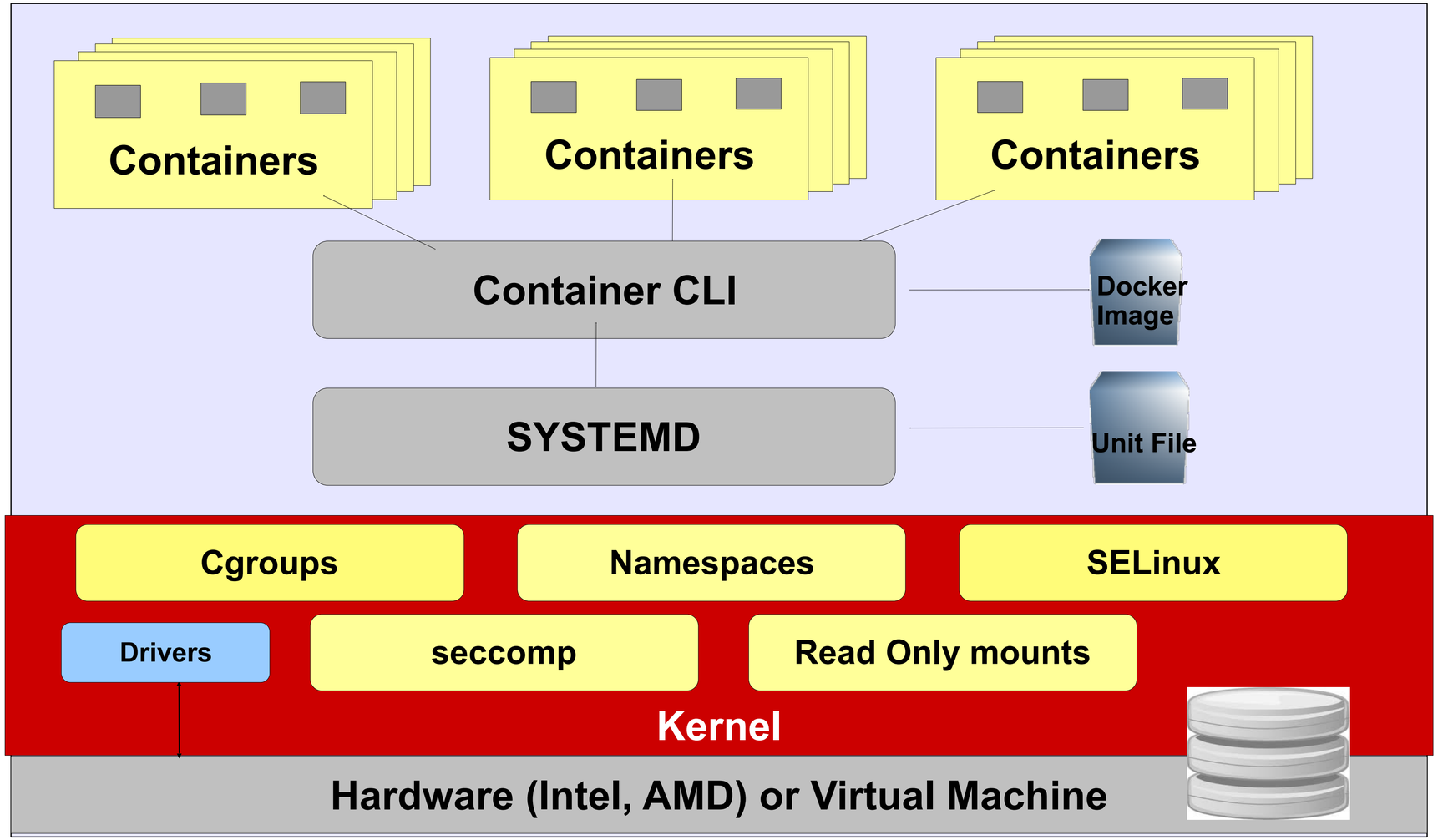


Version 1.2

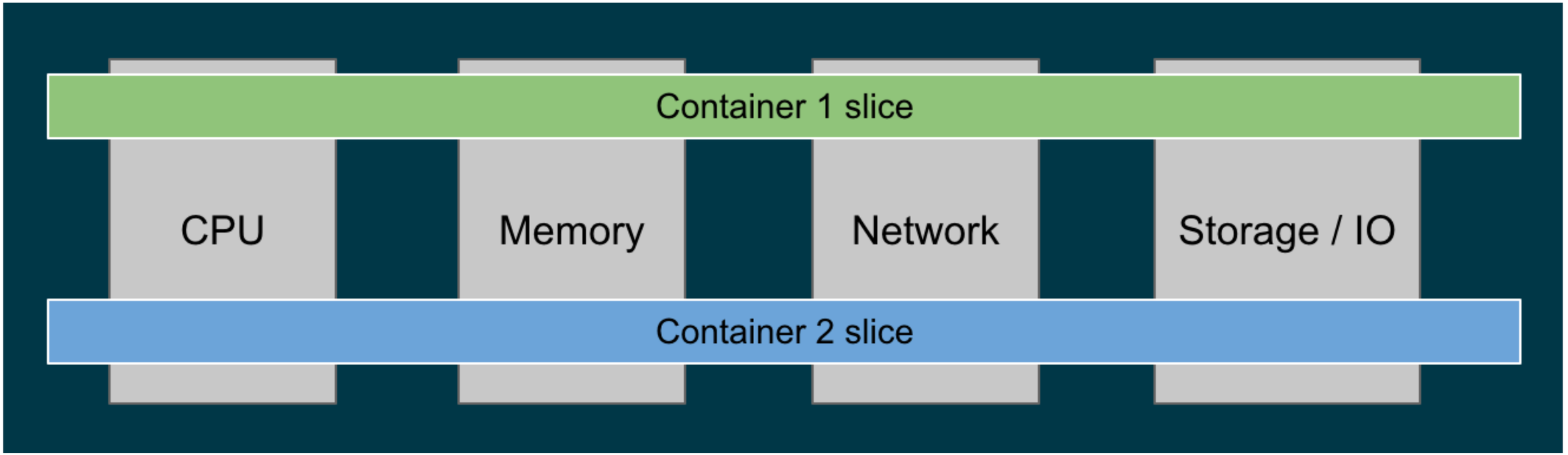
20% Conversion Rate

CONTAINER HOST SECURITY

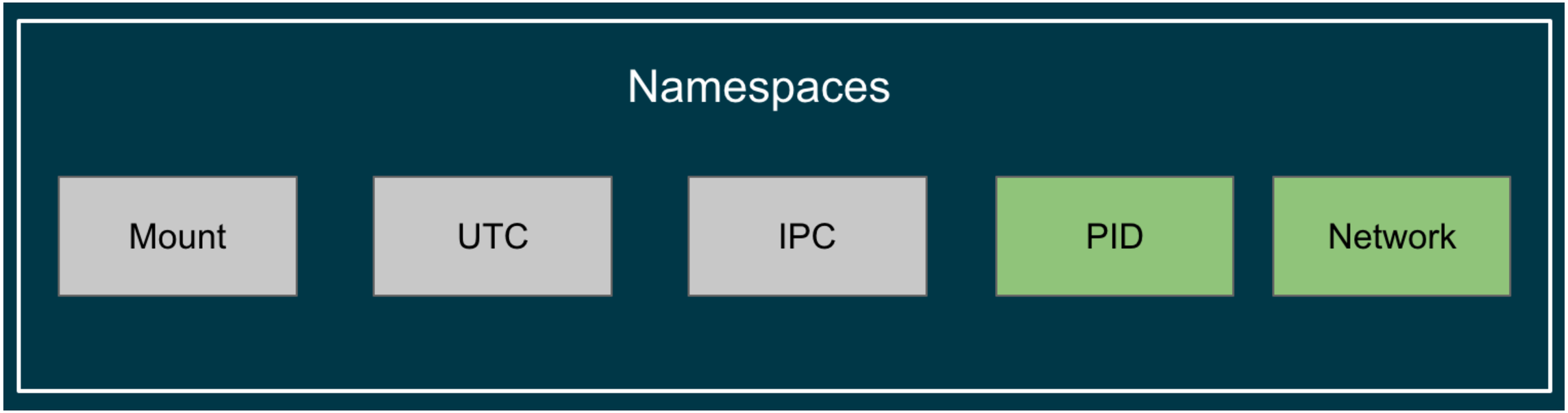
CONTAINERS ARE LINUX



CGROUPS - RESOURCE ISOLATION



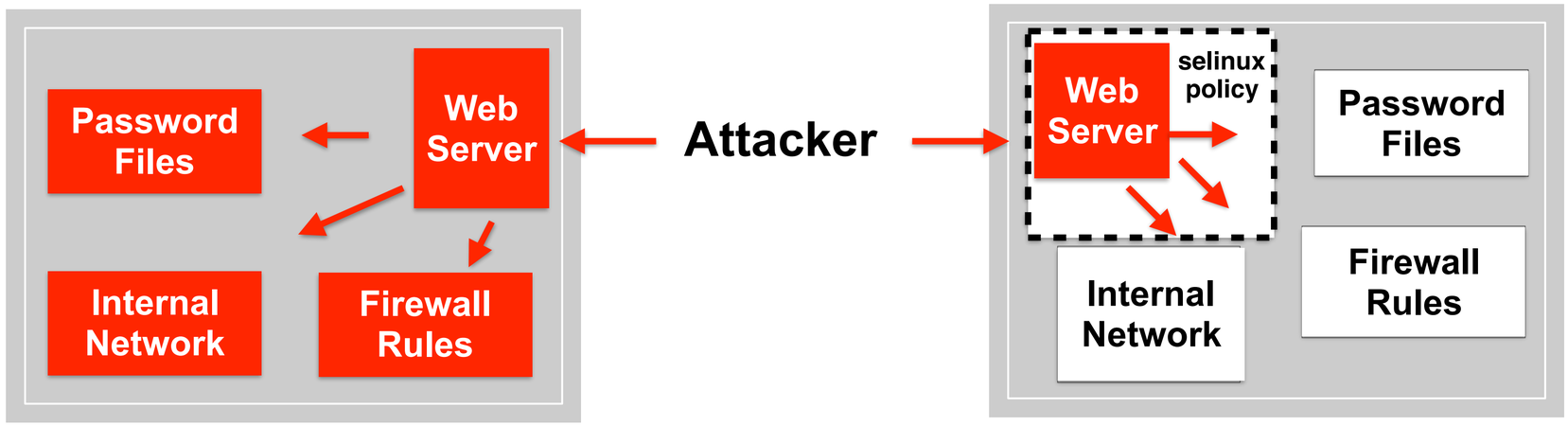
NAMESPACES - PROCESS ISOLATION



SELINUX - MANDATORY ACCESS CONTROLS

Discretionary Access Controls (file permissions)

Mandatory Access Controls (selinux)



redhat.

SECCOMP - DROPPING PRIVILEGES

CAP_SETPCAP	Modify process capabilities
CAP_SYS_MODULE	Insert/Remove kernel modules
CAP_SYS_RAWIO	Modify Kernel Memory
CAP_SYS_PACCT	Configure process accounting
CAP_SYS_NICE	Modify Priority of processes
CAP_SYS_RESOURCE	Override Resource Limits
CAP_SYS_TIME	Modify the system clock
CAP_SYS_TTY_CONFIG	Configure tty devices
CAP_AUDIT_WRITE	Write the audit log
CAP_AUDIT_CONTROL	Configure Audit Subsystem
CAP_MAC_OVERRIDE	Ignore Kernel MAC Policy
CAP_MAC_ADMIN	Configure MAC Configuration
CAP_SYSLOG	Modify Kernel printk behaviour
CAP_NET_ADMIN	Configure the network: - Setting the hostname/domainname - mount(),unmount() - nfsservctl - ….
CAP_SYS_ADMIN	

READ ONLY MOUNTS

/sys

/proc/sys

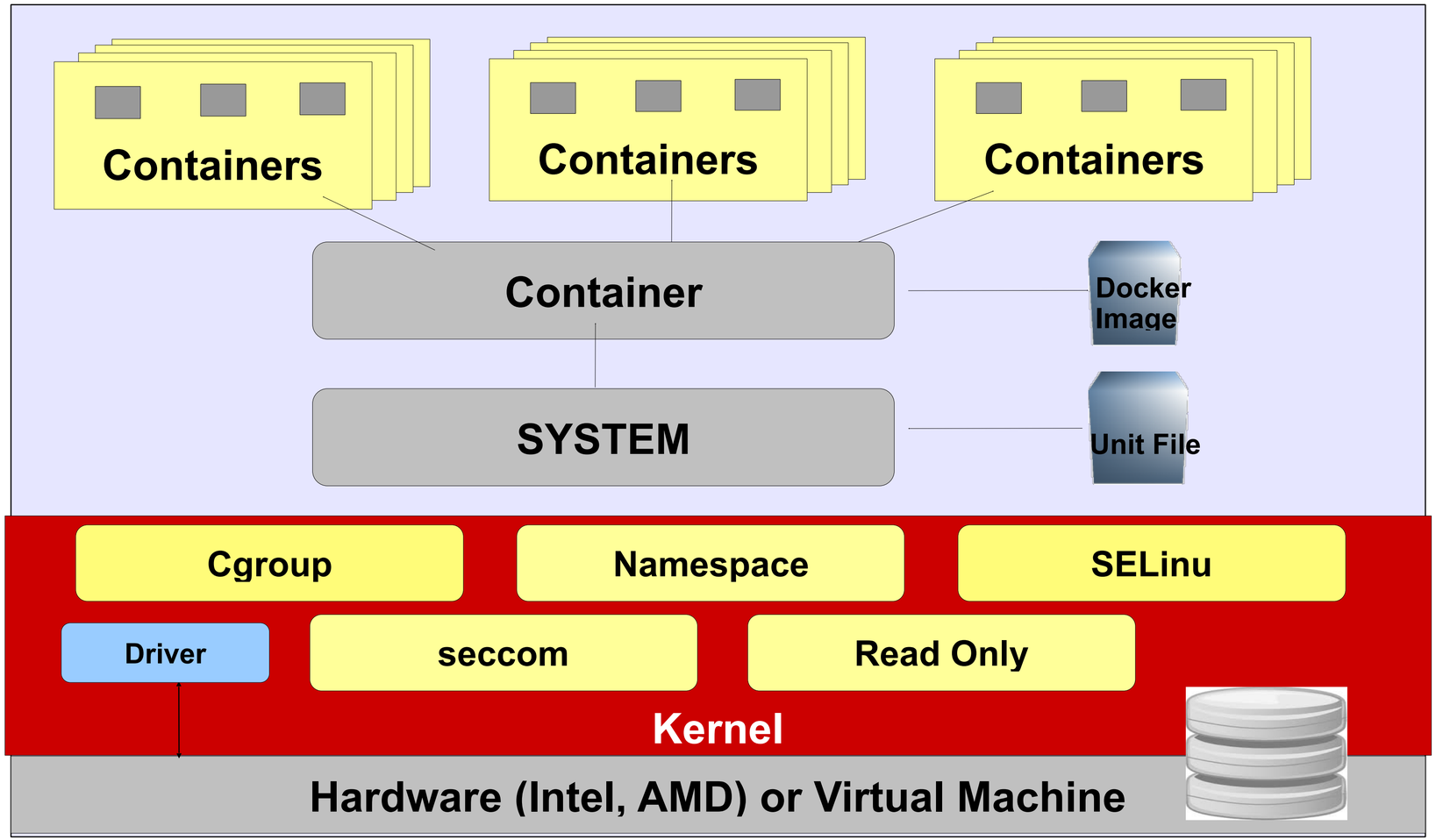
/proc/sysrg-trigger

/proc/irq

/proc/bus

R/O

CONTAINER HOST SECURITY



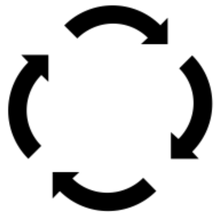
Best Practices

- Don’t run as root
- Limit SSH Access
- Use namespaces
- Define resource quotas
- Enable logging
- Apply Security Errata
- Apply Security Context and seccomp filters

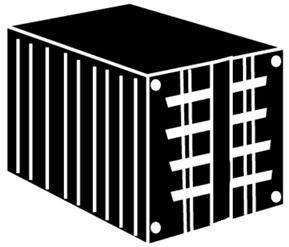
http://blog.kubernetes.io/2016/08/security-best-practices-kubernetes-deployment.html

redhat.

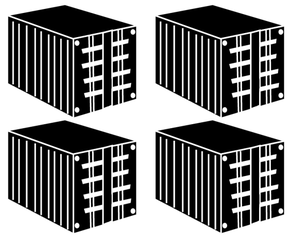
SECURING CONTAINERS



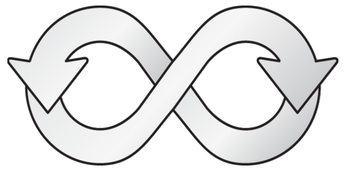
Builds



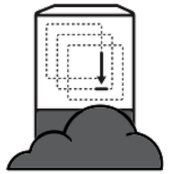
Images



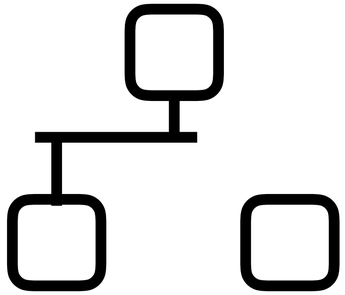
Registry



CI/CD

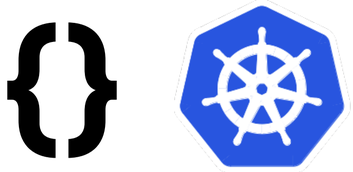


Container host



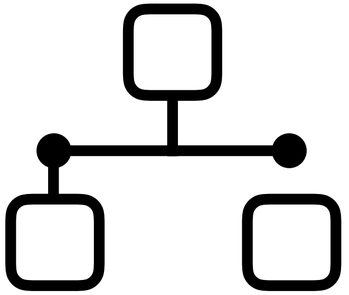
Network isolation

Storage



API & Platform access

Monitoring & Logging



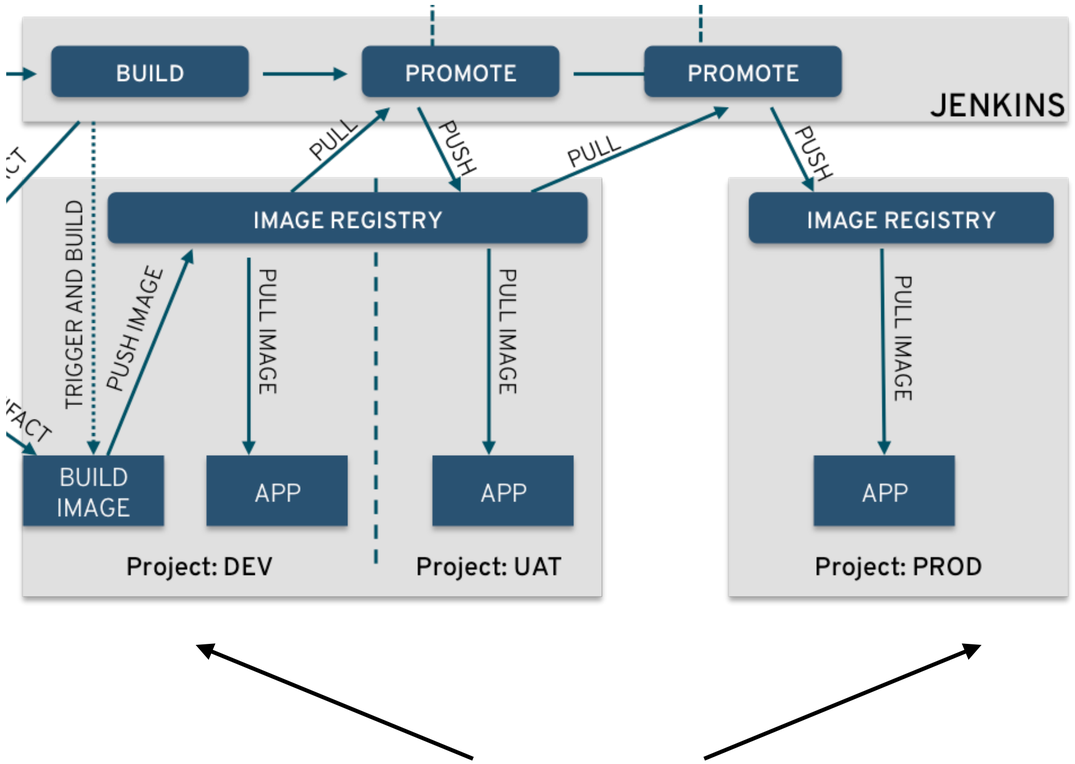
Federated clusters

redhat.

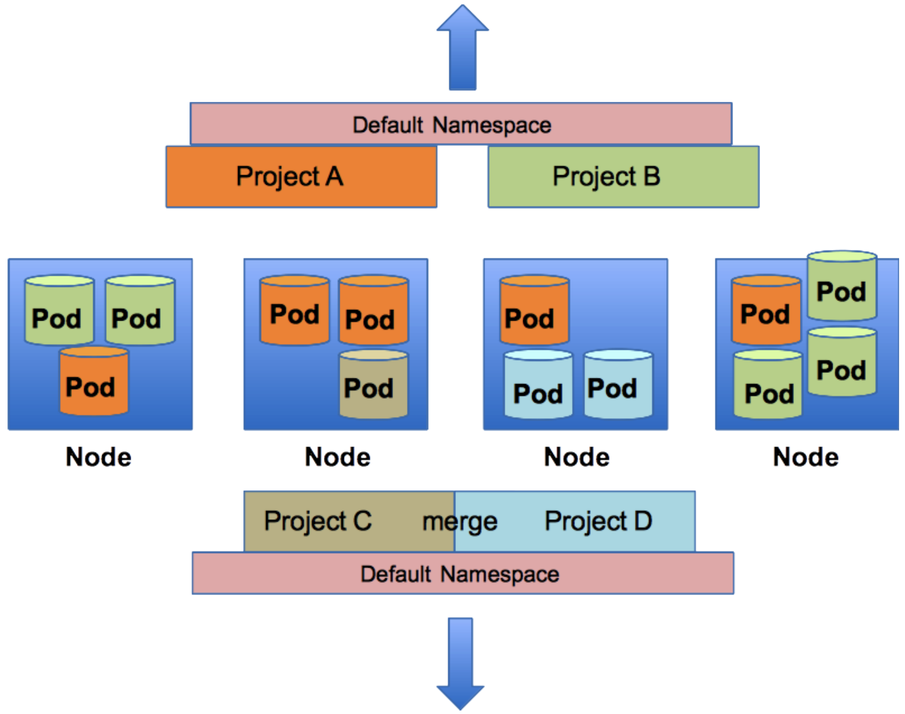
NETWORK ISOLATION

NETWORK ISOLATION

Multi-Environment



Multi-Tenant

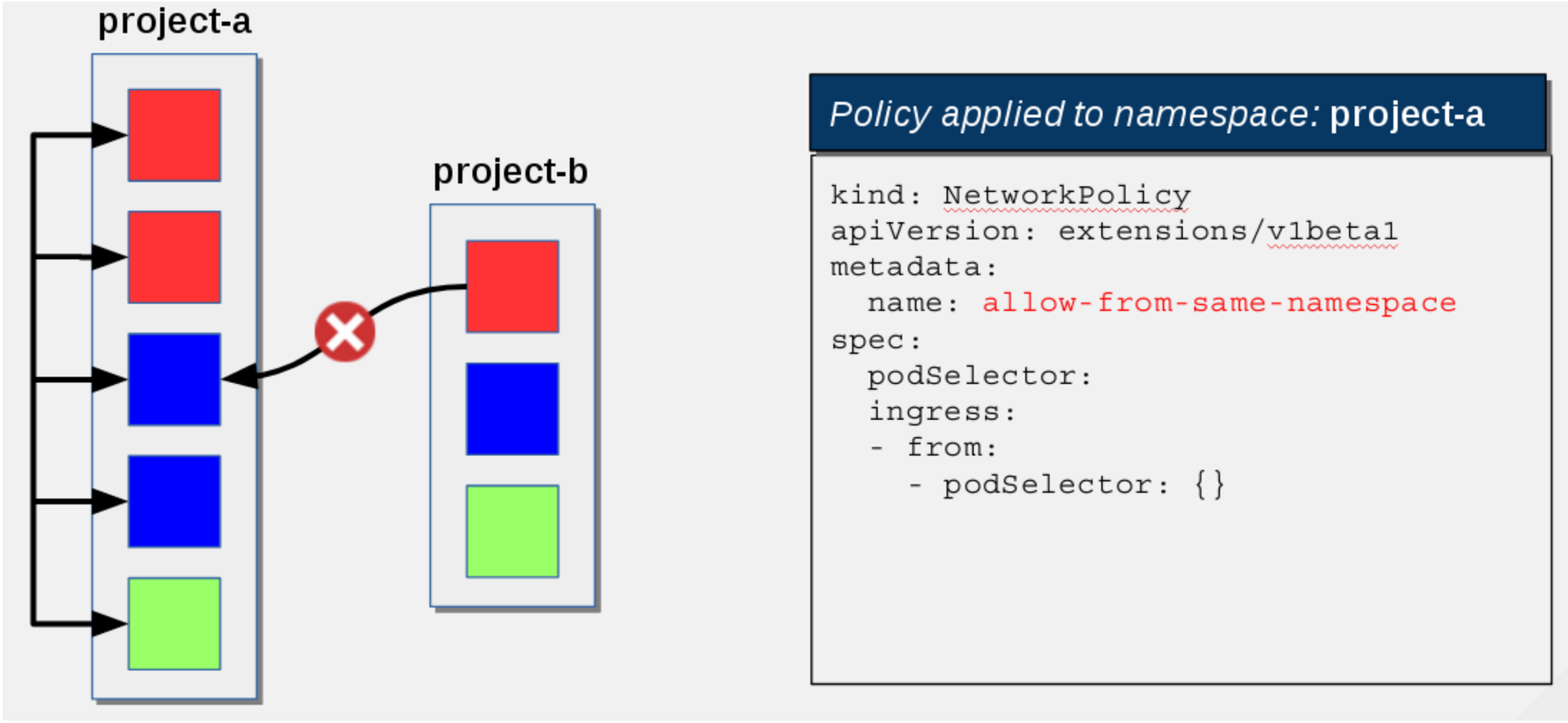


Network Namespace provides resource isolation

redhat.

NETWORK POLICY

example:
all pods in namespace ‘project-a’ allow traffic
from any other pods in the same namespace.”

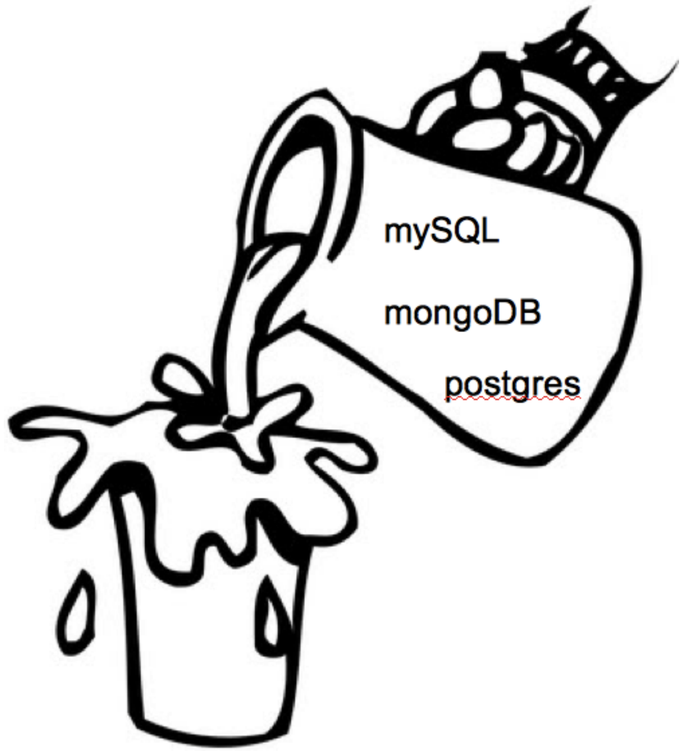


STORAGE SECURITY

STORAGE SECURITY

Security Context Constraints

Local Storage Quota



API & PLATFORM ACCESS

API & PLATFORM ACCESS

Authentication
via
OAuth tokens and
SSL certificate

Authorization
via
Policy Engine
checks
User/Group
Defined Roles

MONITORING & LOGGING

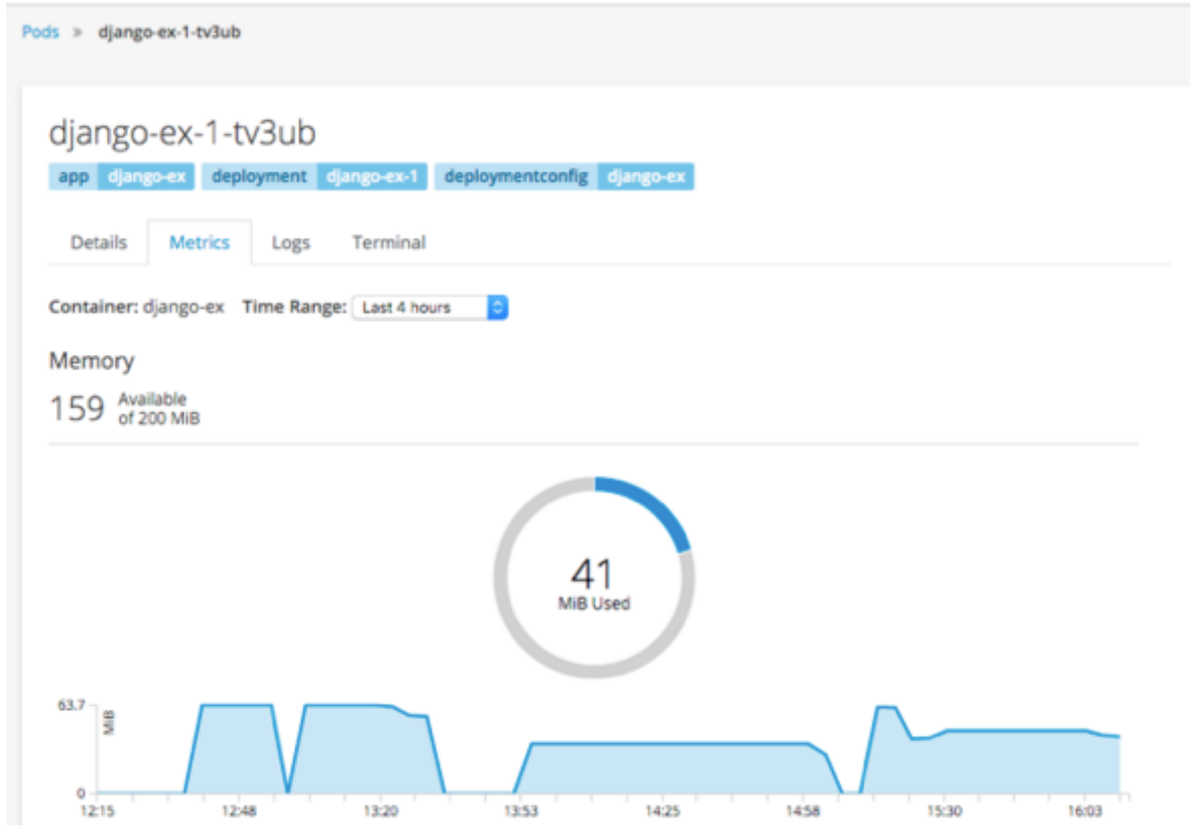
LOGGING

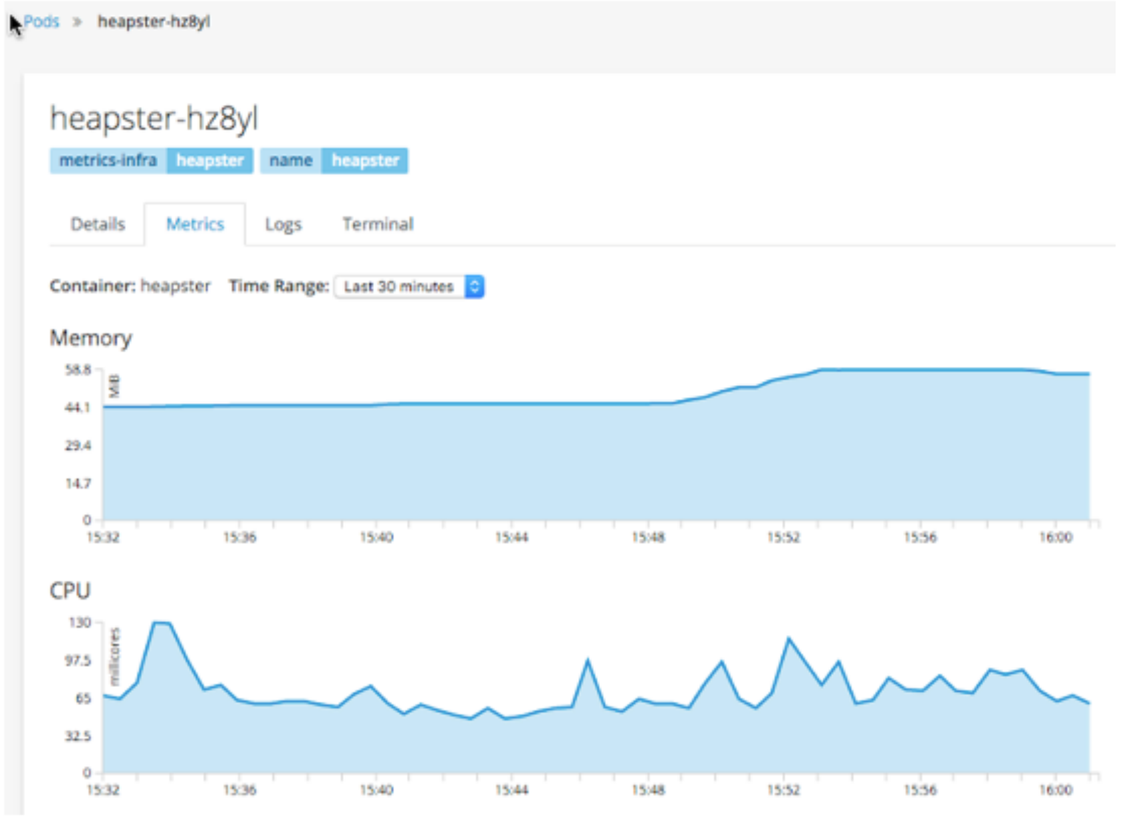
Aggregate platform and application log access via Kibana + Elasticsearch

redhat.

MONITORING

Historical CPU and Memory usage



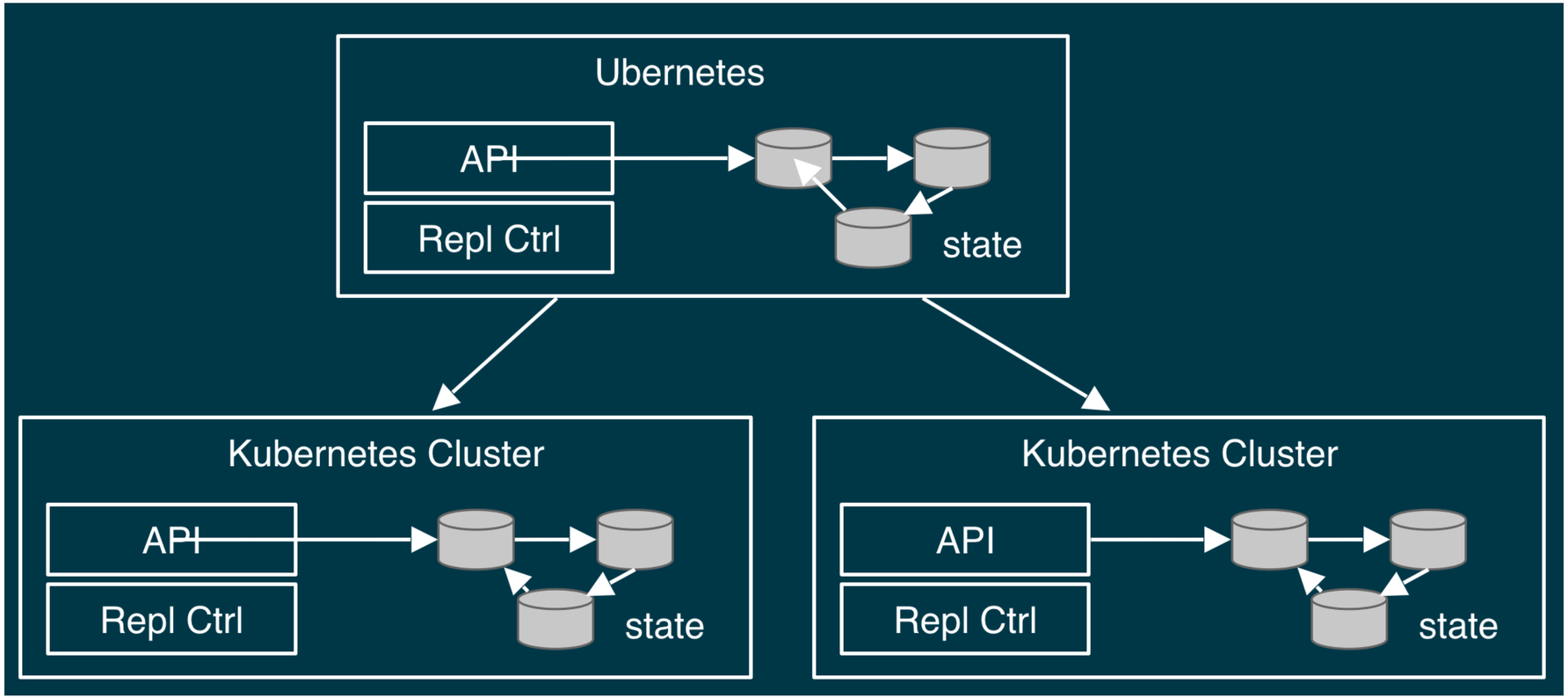


redhat.

FEDERATION

FEDERATED CLUSTERS

Roles & access management (in-dev)



Amazon East

OpenStack

MICROSERVICES

SERVICE MESH

Monitoring & Metrics
-prometheus (logs)
-grafana (visual)

Traffic routing
- pilot
- circuit breaker
- a/b testing
- traffic mirroring

Access Control
& usage policies
-mixr (policy decisions)

Encryption & Auth
-citadel
-service 2 service
-user auth

Fault injections
-envoy
corner cases: abort & delays

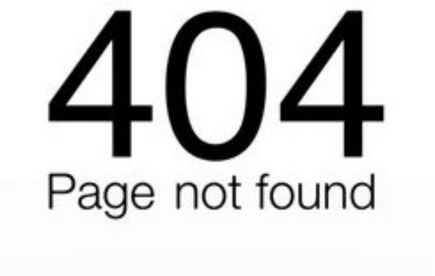
redhat.

DEVSECOPS METRICS

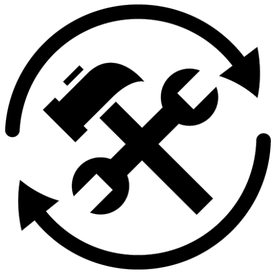
Compliance Score

Deployment Frequency

Lead Time



Deployment Failure Rate



Mean Time to Recover

99.999

Service Availability

THANK YOU

linkedin: Chris Van Tuin
email: cvantuin@redhat.com
twitter: @chrisvantuin