Security Practices in OpenShift

as experienced @ Amadeus

Nenad Bogojević
Amadeus S.A.S.
Diogenes Rettori
Red Hat
2017

Amadeus

In one slide

_Provides IT services for travel industry

_Operates e-commerce web sites, payment processing, b2b services in travel

_Using OpenShift 3 since 2 years

- In own datacenters, in public clouds

Why security

And not the one like in picture

_Protecting assets

- computing capacity, data

_Personal information

- General Data Protection Regulation (GDPR)

_e-Commerce & payment processing

- PCI/DSS

How?

To be better than the one like in picture

_OpenShift & Containers

- Lot of things are changing
- Old rules may not be applicable
- Risks are still out there

How?

To be better than the one like in picture

_OpenShift & Containers

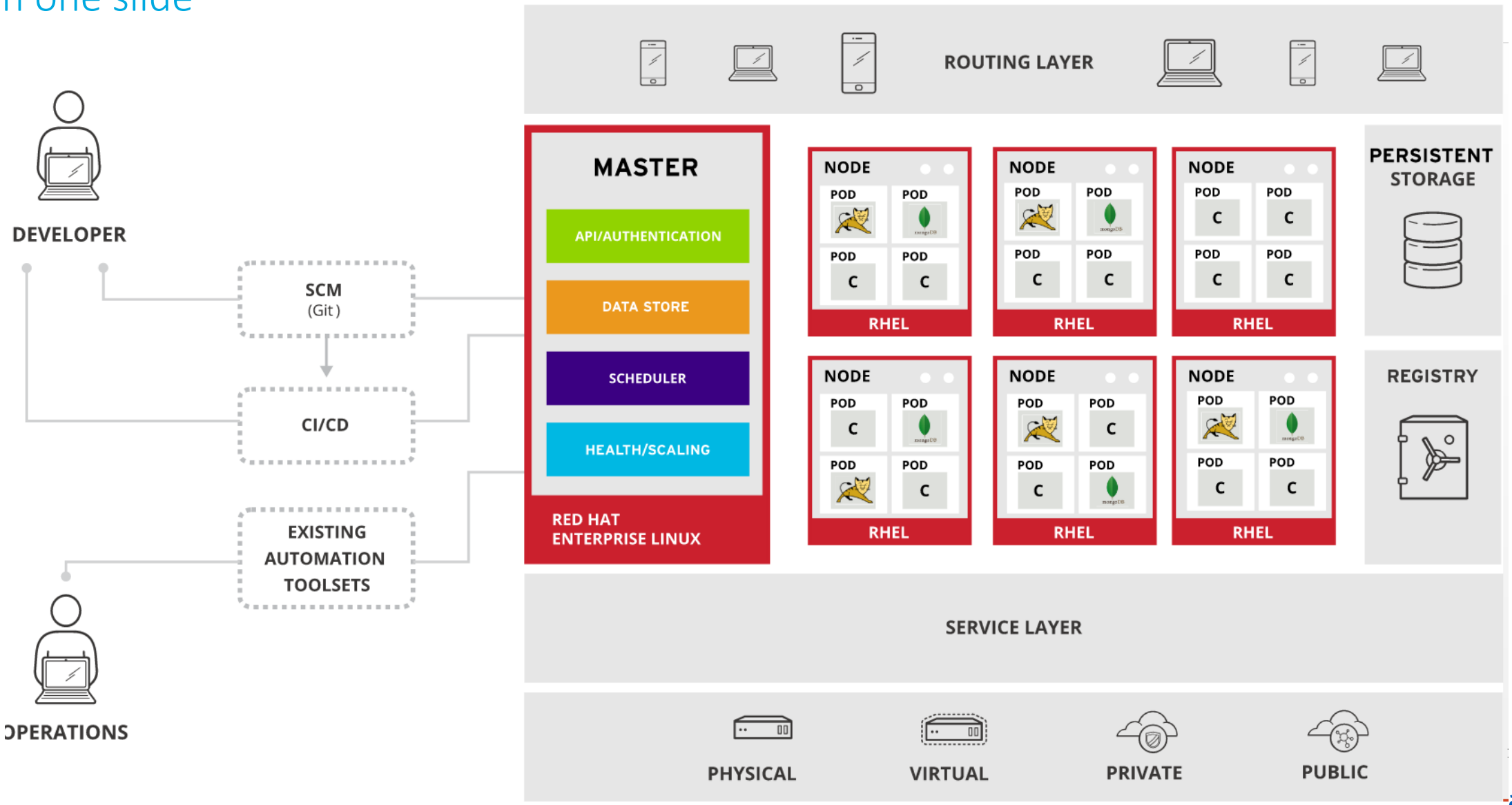
- Lot of things are changing
- Old rules may not be applicable
- Risks are still out there

EVERYONE ON BOARD

Infrastructure

OpenShift Architecture

In one slide



Preparing infrastructure

And security

Use OpenStack on our hardware

Or public cloud providers

_Pre-constructed VM images

- mirrored repositories & registries
- scanned using OpenSCAP

_Network design

- Where are DMZ and layered protection?
- OpenStack – security groups

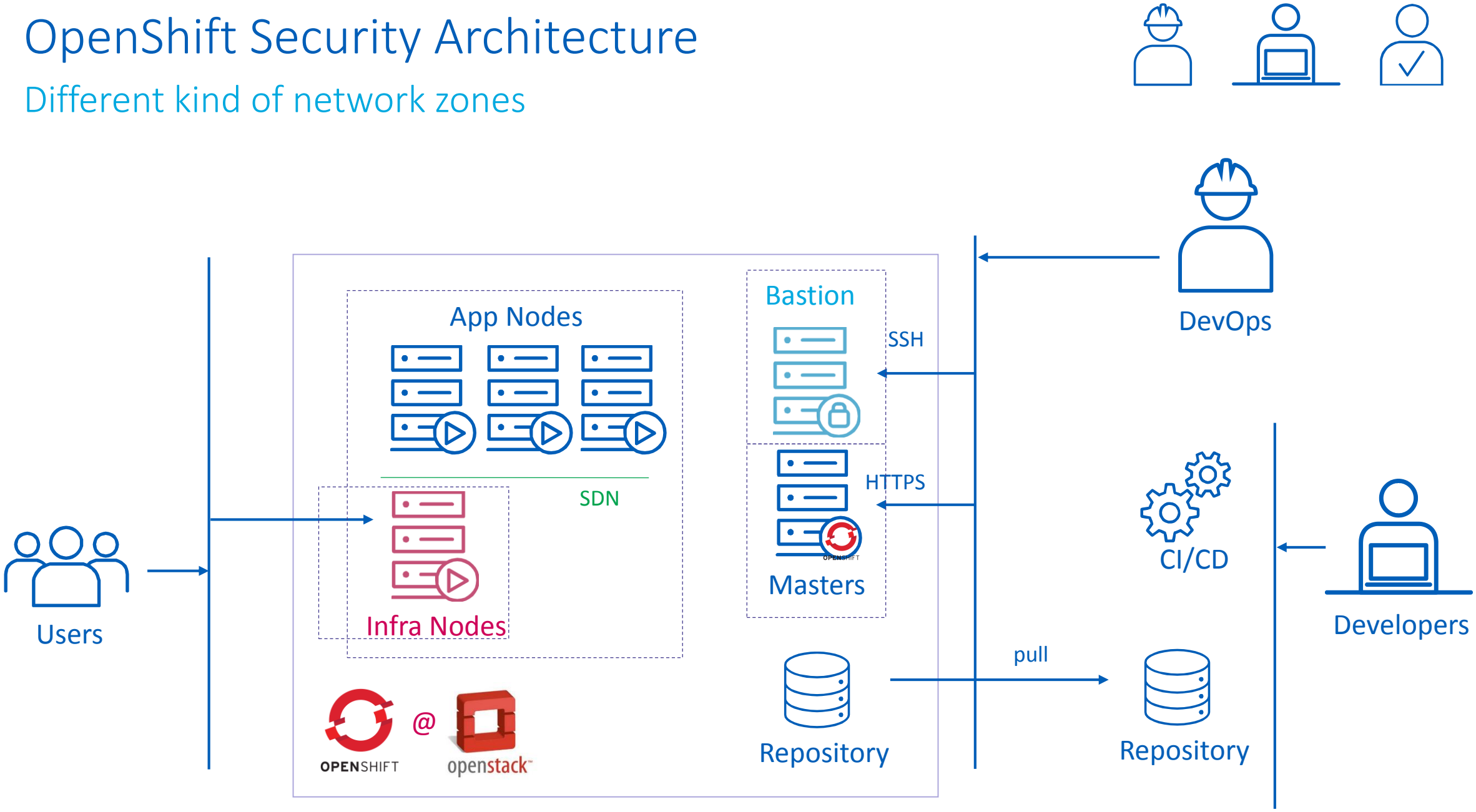
_Access control (bastion server)

_Upgrade policy

- Rebuild vs rolling
- Bi-weekly/monthly

OpenShift Security Architecture

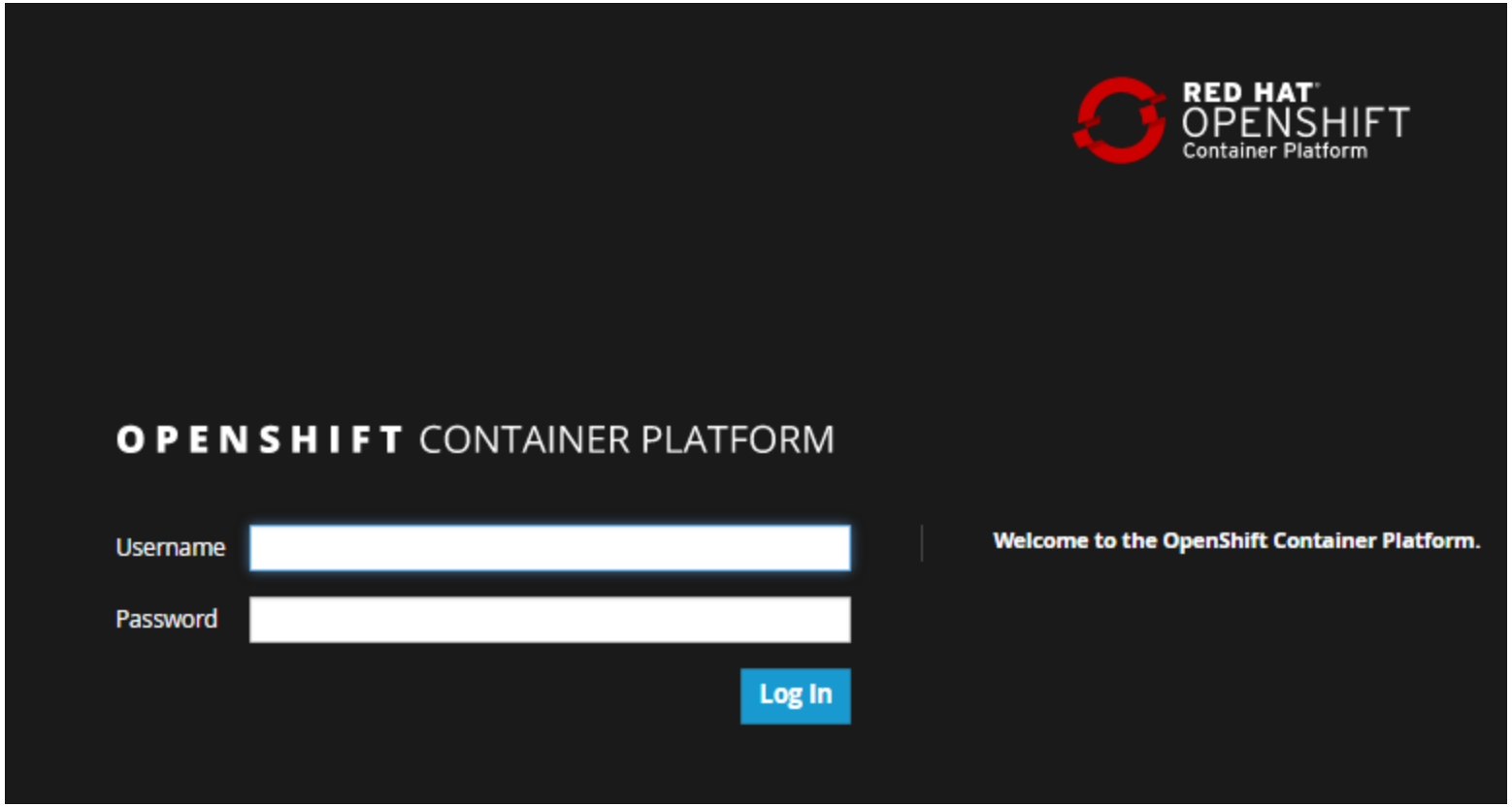
Different kind of network zones



OPENSHIFT

OpenShift

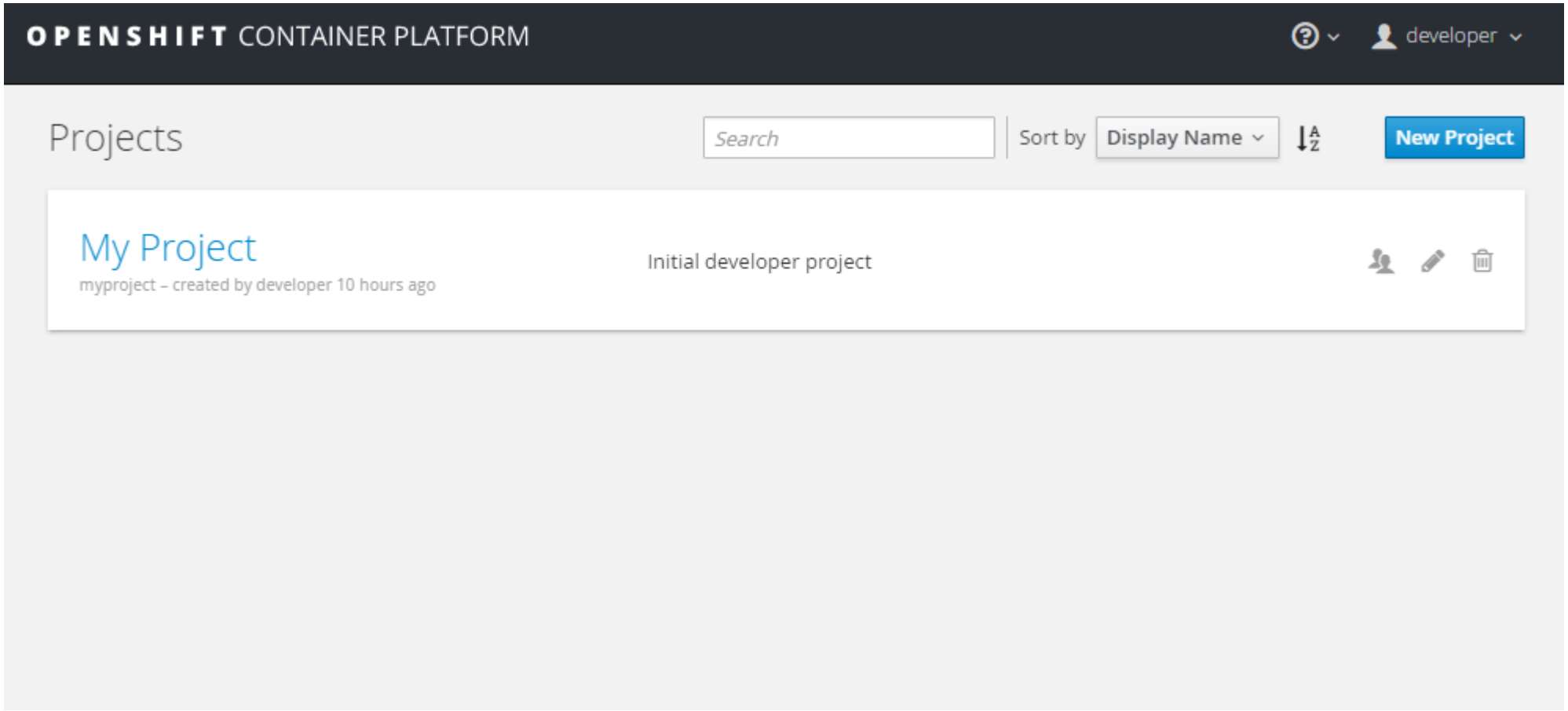
Let’s login!



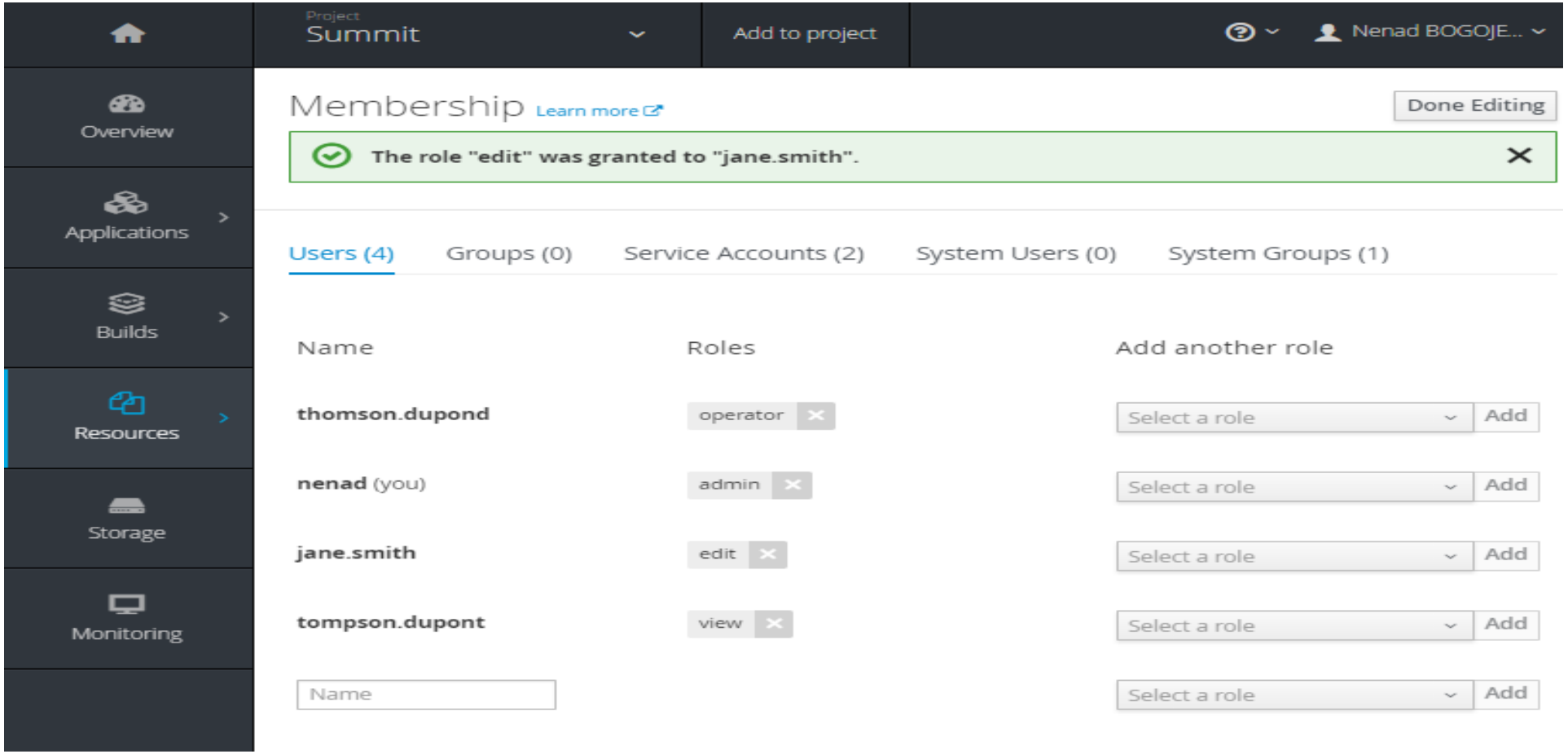
Let’s login!

```
oc login -u system:admin
```

Let’s login!



OpenShift Security Introduction



OpenShift Secrets

Decoupling sensitive information from applications

_Way of managing & distributing sensitive information

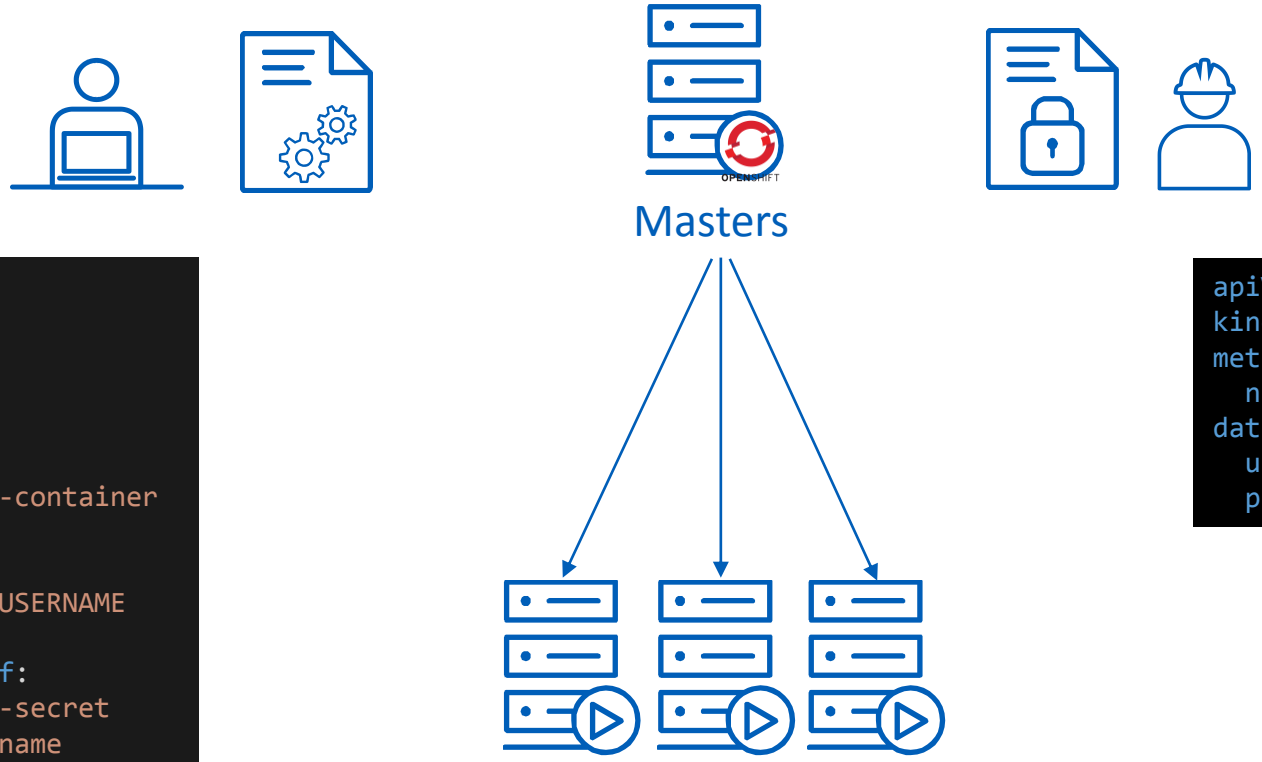
- keys, certificates, passwords, usernames

_Separate sensitive information management from application pods

- Secured delivery to nodes (TLS)
- Only present in memory on openshift nodes
- Centralized management
- Easy access from application
  - Environment variables
  - Volumes

Using Secrets

Security as code




```yaml
apiVersion: v1
kind: Pod
metadata:
  name: use-secret-pod
spec:
  containers:
    - name: secret-test-container
      image: myapp
      env:
        - name: SECRET_USERNAME
          valueFrom:
            secretKeyRef:
              name: top-secret
              key: username
  restartPolicy: Always
```

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: top-secret
data:
  username: bmVuYWQ=
  password: aWtuZXd5b3V3b3VsZHRyeXRoaXM=
```

OpenShift Secrets – „Less Great Things“

Handbrake for certification

_Stored in (almost) clear

- in etcd on masters
- on tmp storage on nodes
- accessible through API

_How about vaults?

Some solutions

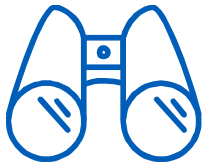
It’s not show-stopper

_You already have big issue if someone compromised your infrastructure

_Encrypt disks 

_Store in vault, with decryption service

- Side-car or init containers
- Security as a service

_Compensating controls 

OpenShift Audit Log

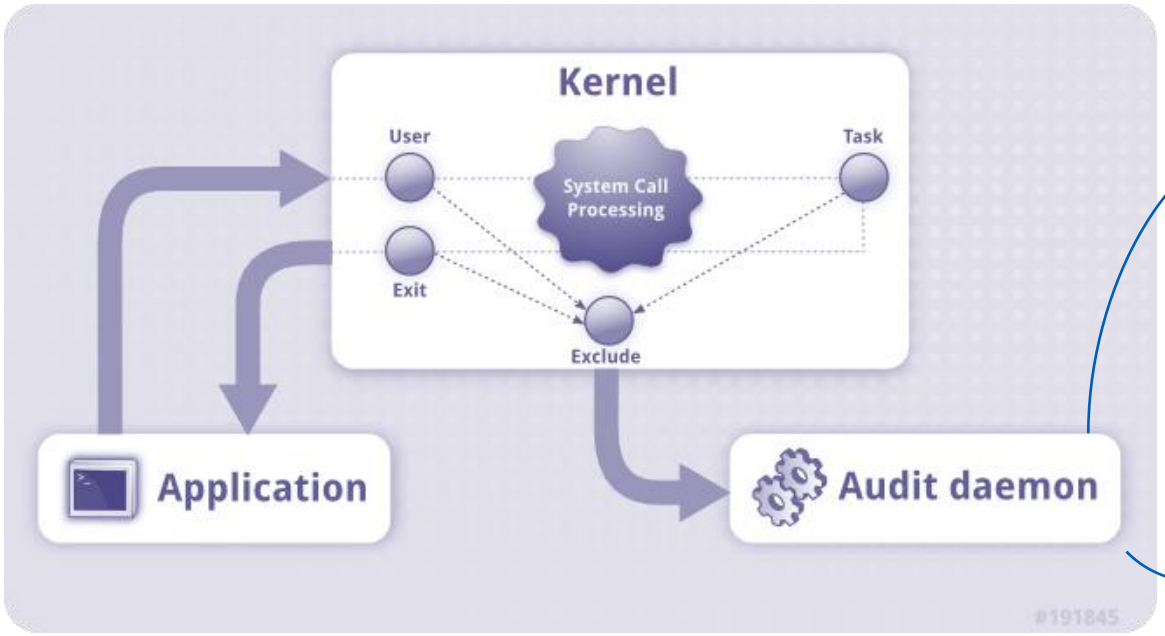
_OpenShift provides log of activities that have affected system by individual users, administrators, or other components of the system.

```
AUDIT: id="5c3b8227-4af9-4322-8a71-542231c3887b" ip="127.0.0.1"
method="GET" user="nenad" as="<self>" namespace="someproject"
uri="/api/v1/namespaces/someproject/secrets"
AUDIT: id="5c3b8227-4af9-4322-8a71-542231c3887b" response="401"
```

_Activate on master /etc/origin/master/master-config.yaml

```
auditConfig:
   enabled: true
```

auditd introduction



audit.log

rsyslog

Alerting system

auditd rules

```
-a always,exit -S <syscall>
-w <filename>
```

auditd rules for masters

Monitoring etcd

_OpenShift master - know if someone plays with etcd

```
-a always,exit -F arch=b64 -S creat -S open -S openat
    -S open_by_handle_at -S truncate -S ftruncate
    -F dir=/var/lib/etcd
    -k openshift_etcd
```

..and on nodes

Monitoring secret

_Secrets mounted as tmpfs inside /var/lib/opesnift.

_When new secret is mounted add it to auditd rules

- When new secret is unmounted remove it to from auditd rules

_All monitorable secrets must have certain string in name

- (e.g. secret~example)

_If you open or close secrets often, it may generate a lot of messages

```
findmnt --list --noheadings --types tmpfs --poll --output ACTION,TARGET |
grep secret~example |
awk ‘$1 == “mount” { print $2 }‘ |
xargs -L 1 -i auditctl --a always,exit -F arch=b64 -S creat -S open -S openat
-S open_by_handle_at -S truncate -S ftruncate -F dir={} -k openshift_secret
```

```
findmnt --list --noheadings --types tmpfs --poll --output ACTION,TARGET |
grep secret~example |
awk ‘$1 == “unmount” { print $2 }‘ |
xargs -L 1 -i auditctl --d always,exit -F arch=64 -S creat -S open -S openat
-S open_by_handle_at -S truncate -S ftruncate -F dir={} -k openshift_secret
```

More use of auditd

With the help of openscap

Compliance and Scoring

The target system did not satisfy the conditions of 23 rules! Please review rule results and consider applying remediation.

Rule results

51 passed | 23 failed | 1

Severity of failed rules

14 low | 6 medium | 3 high

Score

	Percent
	56.63%

Record attempts to alter time through adjtimex

Rule ID	xccdf_org.ssgproject.content_rule_audit_rules_time_adjtimex
Result	pass
Time	2016-08-30T10:57:30
Severity	low
Identifiers and References	identifiers: CCE-RHEL7-CCE-TBD references: AC-17(7), AU-1(b), AU-2(a), AU-2(c), AU-2(d), AU-12(a), AU-12(c), IR-5, 1487, 169

Description	If the `auditd` daemon is configured to use the `augenrules` program to read audit rules during daemon startup (the default), add the following line to a file with suffix `.rules` in the directory `/etc/audit/rules.d`:
	`-a always,exit -F arch=b32 -S adjtimex -k audit_time_rules`
	If the system is 64 bit then also add the following line:
	`-a always,exit -F arch=b64 -S adjtimex -k audit_time_rules`
	If the `auditd` daemon is configured to use the `auditctl` utility to read audit rules during daemon startup, add the following line to `/etc/audit/audit.rules` file:
	`-a always,exit -F arch=b32 -S adjtimex -k audit_time_rules`
	If the system is 64 bit then also add the following line:
	`-a always,exit -F arch=b64 -S adjtimex -k audit_time_rules`

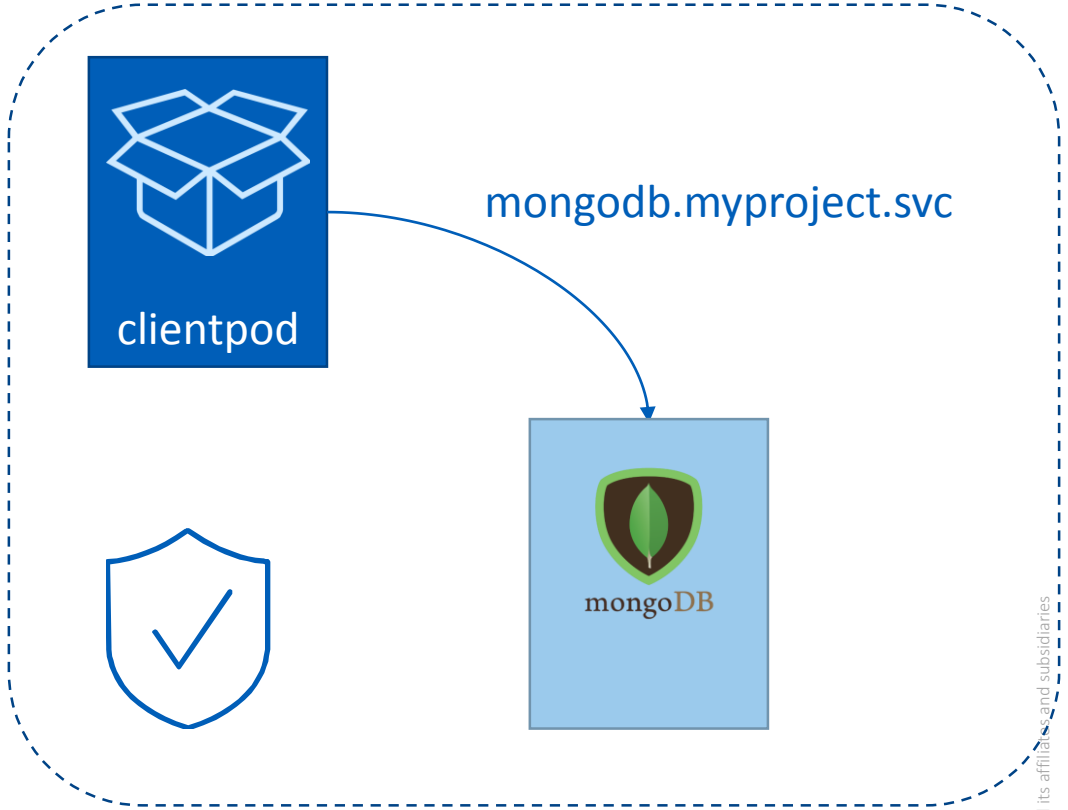
Service Signing Certificate

_Secure communication inside or outside your cluster

_Service annotated with service.alpha.openshift.io/serving-cert-secret-name=name

_Certificate automatically generated and provided as a secret to pod

_Clients can rely on automatically mounted CA /var/run/secrets/kubernetes.io/serviceaccount/service-ca.crt



Containers

Containers & Developers

- We want to empower developer

- Let’s be agile!
  - Run as root
    - Privileged containers – hostpath
    - port < 1000
  - Running old containers
    - FROM httpd:2.4.12
  - There’s this cool blackhat/jboss container on docker hub, let’s pull it

Containers & Developers

- We want to empower developer

- Let’s be agile!
  - Run as root
    - Privileged containers – hostpath
    - port < 1000
  - Running old containers
    - FROM httpd:2.4.12
  - There’s this cool blackhat/jboss container on docker hub, let’s pull it

Root Access

Not allowed

_Support arbitrary user ids

- Use root group
  `chown -R someuser:root /app && chmod -R g+rwX /app`

_Your application needs to listen on port 80?

- Can’t you change it?

```yaml
apiVersion: v1
kind: Service
metadata:
  name: httpd
spec:
  ports:
  - port: 80
    targetPort: 10080
    protocol: TCP
```

_Use SCC (Security Context Constraint)

- privileged containers, host paths, user id,
  FS Groups, selinux, capabilities

_seccomp if you want to restrict even more

Image control

Secured source

_All images come from internal registry

_Using RHEL as base images

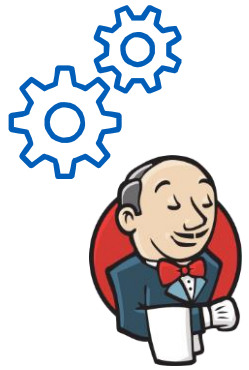
- RedHat repository mirrored into internal

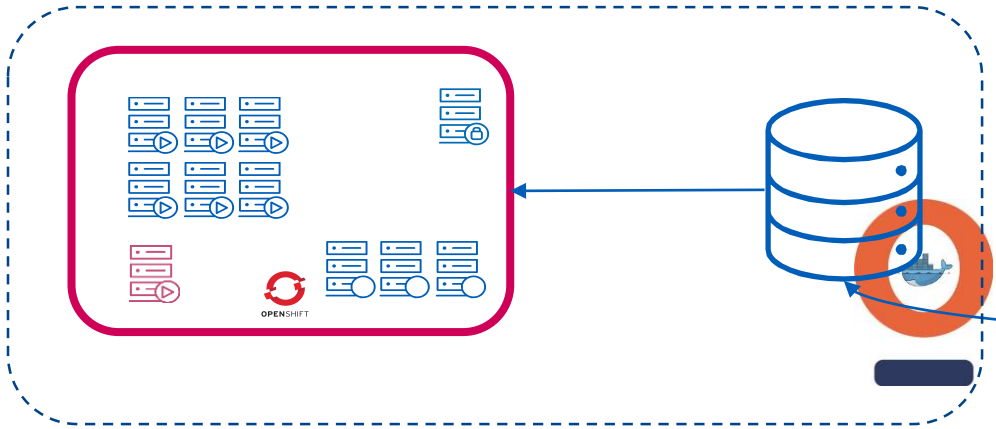
_Other images must be built internally from source code

_No automatic access to docker hub from build machines

_Production access it’s own repository with only validated images

access.redhat.com/containers





Old images and security vulnerabilities

image-inspector

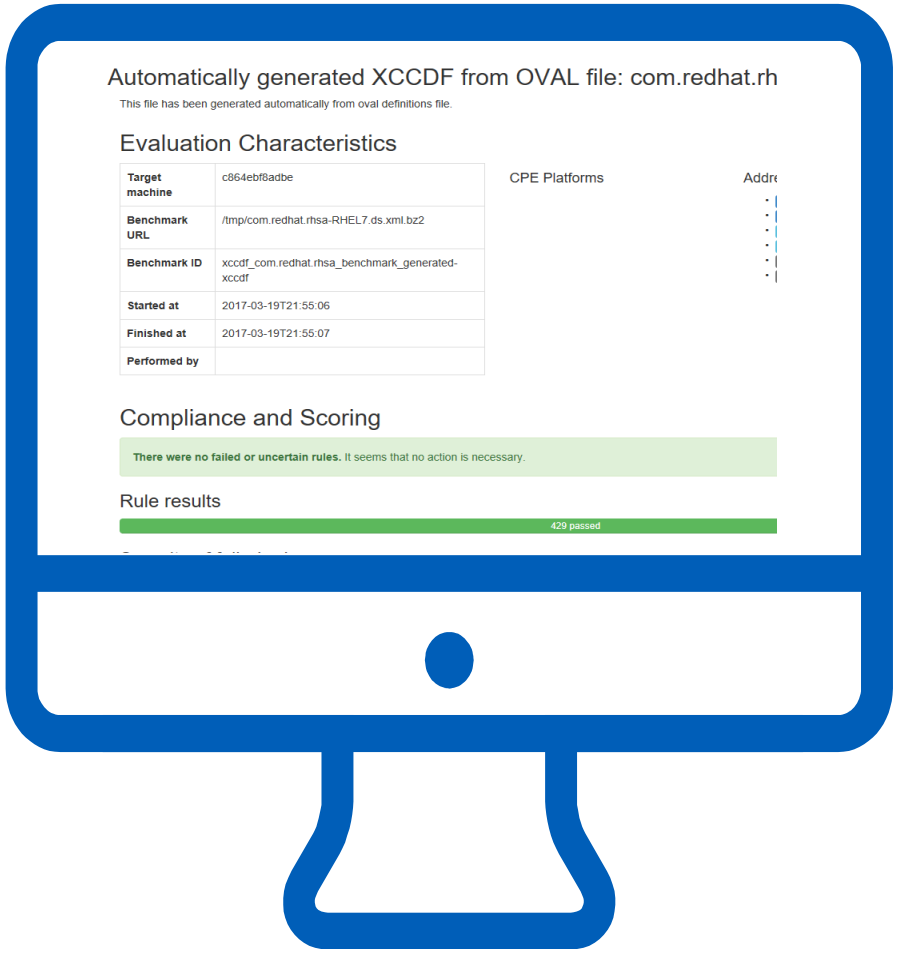
_Can we run security scan on image before it runs?

- image-inspector
- oscap-docker

_Run OpenSCAP on a docker image and serve result

```
docker run -ti --rm --privileged -p 8080:8080
 -v /var/run/docker.sock:/var/run/docker.sock
  openshift/image-inspector --image=some-application:20
  --path=/tmp/image-content --serve 0.0.0.0:8080 --scan-
type=openscap
```

_Used during build process



Guiding thoughts

_Platform can be secured from container vulnerabilities

- containers do bring risk, but it can be managed

_Platform will not solve application vulnerabilities

- but it can help
- true multitenancy is complex

_Start with the principle of least access

- grant new capabilities to applications only when needed

What we miss

This might be roadmap

Encryption of Secrets!

Network policies – internal and egress

Generic/pluggable image-inspector?

More fine-grained RBAC.

Thank you!

amadeus.com
amadeus.com/blog

You can follow us on:
AmadeusITgroup

amadeus