TEN LAYERS OF CONTAINER SECURITY

Tim Hunt
Kirsten Newcomer

May 2017

ABOUT YOU

Are you using containers?

What's your role?

- Security professionals
- Developers / Architects
- Infrastructure / Ops

Who considers security part of their job?

VALUE OF CONTAINERS

- Sandboxed application processes on a shared Linux OS kernel
- Simpler, lighter, and denser than virtual machines
- Portable across different environments

- Package my application and all of its dependencies
- Deploy to any environment in seconds and enable CI/CD
- Easily access and share containerized components

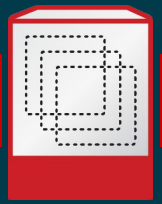

WHY ARE WE HAVING
THIS CONVERSATION?

SECURING CONTAINERS: THE TOP TEN LIST

1. Container Host & Multi-tenancy
2. Container Content
3. Container Registries
4. Building Containers
5. Deploying Containers
6. Container Platform
7. Network Isolation
8. Storage
9. API Management
10. Federated Clusters

1 CONTAINER HOST & MULTI-TENANCY THE OS MATTERS

RED HAT ENTERPRISE LINUX



RED HAT ENTERPRISE LINUX ATOMIC HOST

THE FOUNDATION FOR SECURE, SCALABLE CONTAINERS

A stable, reliable host environment with built-in security features that allow you to isolate containers from other containers and from the kernel.

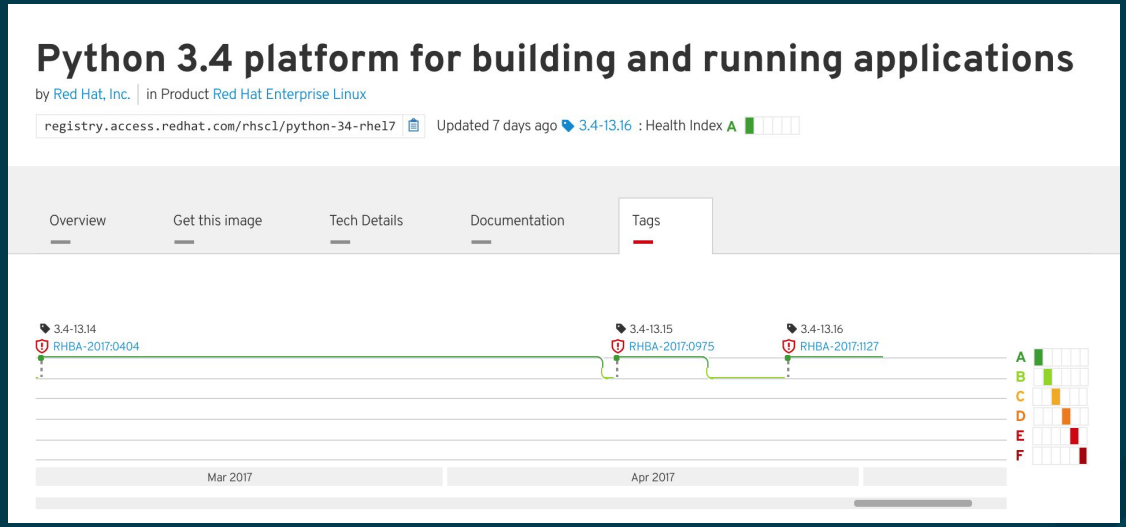
Minimized host environment tuned for running Linux containers while maintaining the built-in security features of Red Hat Enterprise Linux..

SELinux | Kernel namespaces | Cgroups | Capabilities | Seccomp

SECURITY FEATURES ON BY DEFAULT IN OPENSHIFT

2 CONTENT: USE TRUSTED SOURCES

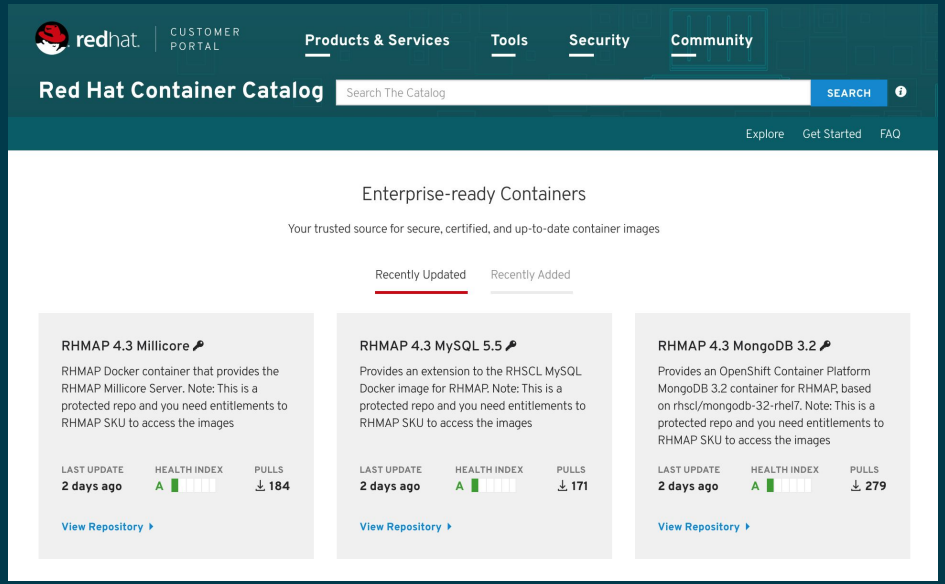
- Are there known vulnerabilities in the application layer?
- Are the runtime and OS layers up to date?
- How frequently will the container be updated and how will I know when it's updated?



Red Hat rebuilds container images when security fixes are released

2 CONTENT: USE TRUSTED SOURCES

Standardization makes security & ops work easier



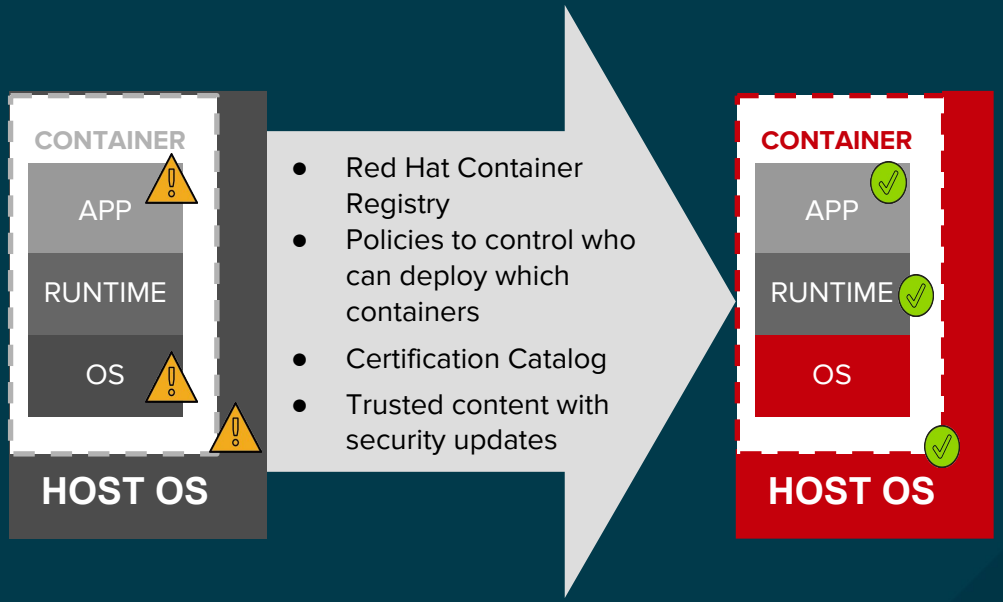
Developers want latest & greatest for best features

Consider breadth and diversity of your software content

3 PRIVATE REGISTRIES: SECURE ACCESS TO IMAGES

Image governance & private registries

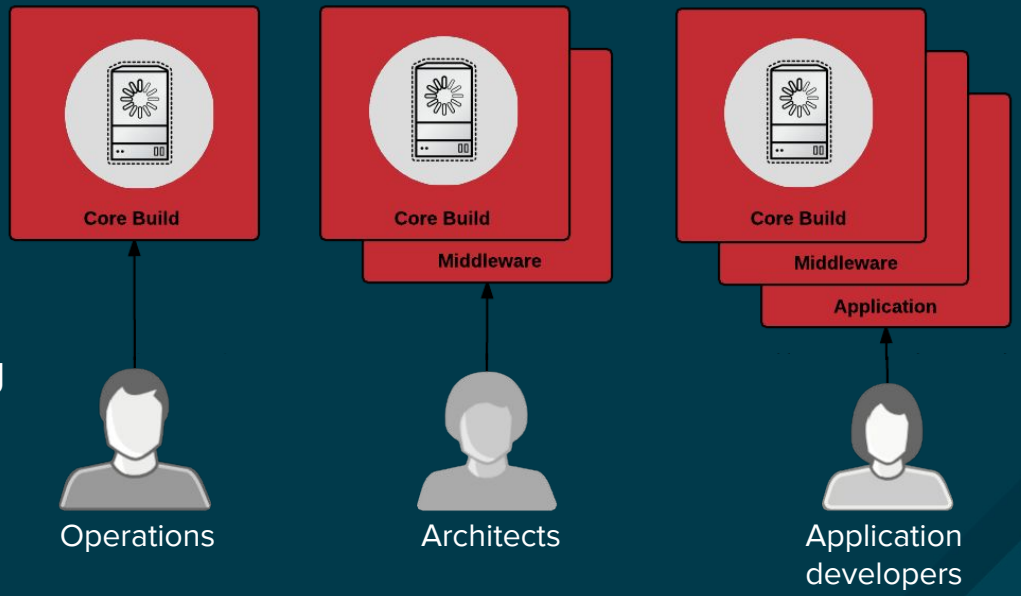
- Are there access controls on the registry? How strong are they?
- What security meta-data is available for your images?
- How is the data kept up-to-date?



4 MANAGING CONTAINER BUILDS

Security & continuous integration

- Layered packaging model supports separation of concerns
- Integrate security testing into your build / CI process
- Use automated policies to flag builds with issues
- Ensure builds always use the latest base image
- Trigger automated CI process



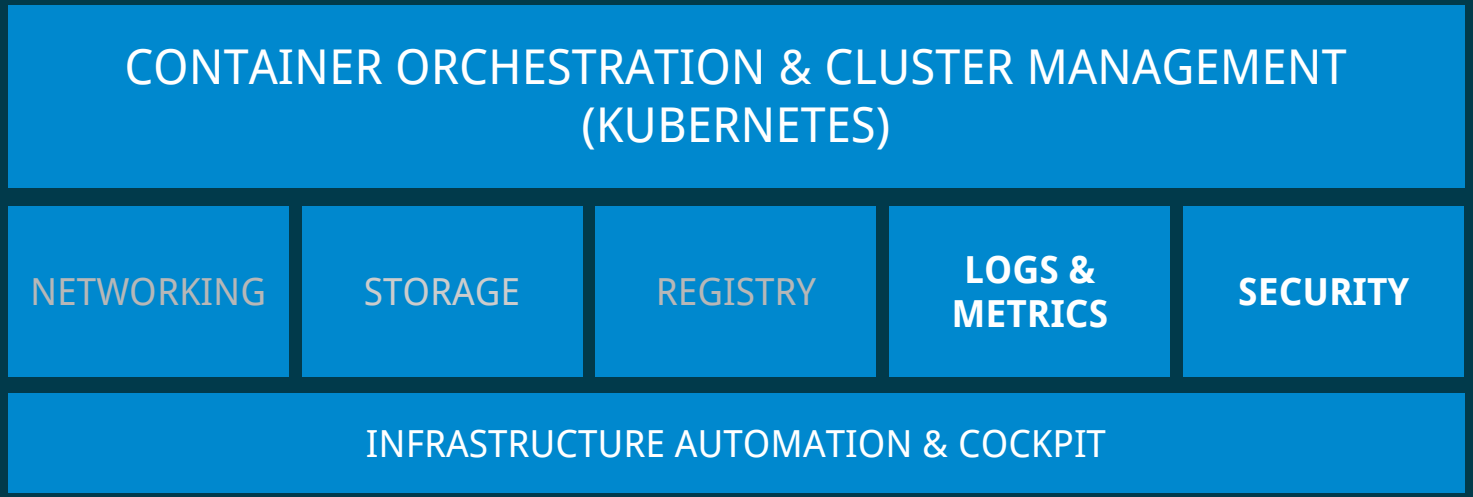
5 MANAGING CONTAINER DEPLOYMENT

Security & continuous deployment

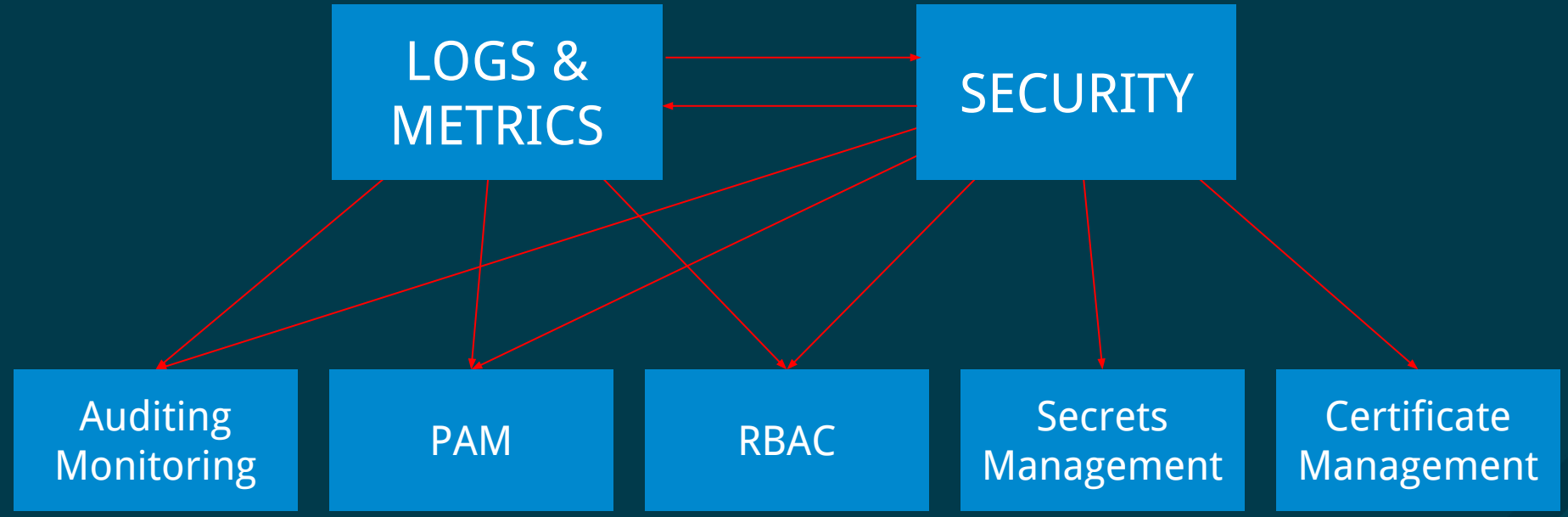
- Monitor image registry to automatically replace affected images
- Use policies to gate what can be deployed: e.g. if a container requires root access, prevent deployment
- Monitor application health & behavior

```
$ oc describe scc restricted
Name:                restricted
Priority:               <none>
Access:
  Users:             <none>
  Groups:               system:authenticated
Settings:
  Allow Privileged:        false
  Default Add Capabilities:      <none>
  Required Drop Capabilities:    KILL,MKNOD,SYS_CHROOT,SETUID,SETGID
  Allowed Capabilities:       <none>
  Allowed Volume Types:       configMap,downwardAPI,emptyDir,persistentVolumeClaim,secret
  Allow Host Network:         false
  Allow Host Ports:        false
  Allow Host PID:          false
  Allow Host IPC:          false
  Read Only Root Filesystem:     false
  Run As User Strategy:             MustRunAsRange
    UID:             <none>
    UID Range Min:           <none>
    UID Range Max:           <none>
  SELinux Context Strategy:         MustRunAs
    User:               <none>
    Role:               <none>
    Type:               <none>
    Level:              <none>
  FSGroup Strategy:                 MustRunAs
    Ranges:             <none>
  Supplemental Groups Strategy:     RunAsAny
    Ranges:             <none>
```

6 CONTAINER ORCHESTRATION & SECURITY



6 CONTAINER ORCHESTRATION & SECURITY



Integration with external logging systems in 3.4

\+

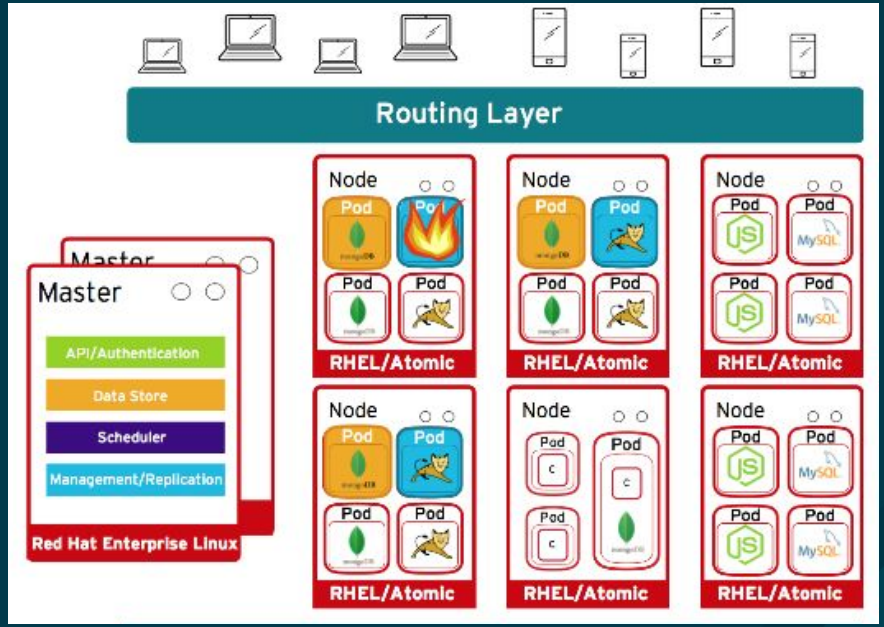
All access to master over TLS / API server is X.509 certificate or token based

Enhancements targeted for 3.6

Enhanced in 3.5

7 CONTAINER MULTITENANCY & NETWORK DEFENSE

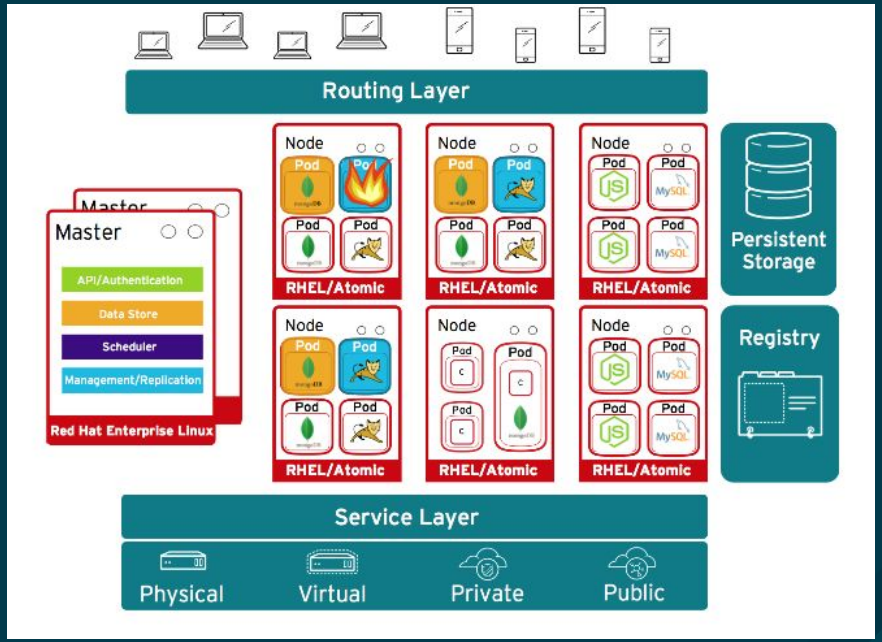
- Segment traffic to isolate users, teams, applications within a single cluster
- Manage egress traffic to meet existing firewall policies
- Tech-preview network policy plug-in allows isolation policies to be configured for individual pods



8 ATTACHED STORAGE

Secure storage by using

- SELinux access controls
- Secure mounts
- Supplemental group IDs for shared storage



9 API MANAGEMENT

Container platform & application APIs

- Authentication and authorization
- LDAP integration
- End-point access controls
- Rate limiting

FUTURE: FEDERATED CLUSTERS ROLES & ACCESS MANAGEMENT

10

Securing federated clusters across data centers or environments

- Authentication and authorization
- API endpoints
- Secrets
- Namespaces

Source: Building Globally Distributed Services using Kubernetes Cluster Federation. October 14, 2016

RED HAT® OPENSHIFT Container Platform

BRINGING IT ALL TOGETHER

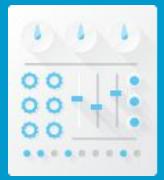
RED HAT® OPENSHIFT Online

Self-Service

Service Catalog
(Language Runtimes, Middleware, Databases)

Build Automation | Deployment Automation

OpenShift Application Lifecycle Management
(CI/CD)



Container Orchestration & Cluster Management
(kubernetes)

Networking | Storage | Registry | Logs & Metrics | Security

Infrastructure Automation & Cockpit

Enterprise Container Host

Container Runtime & Packaging
(Docker)

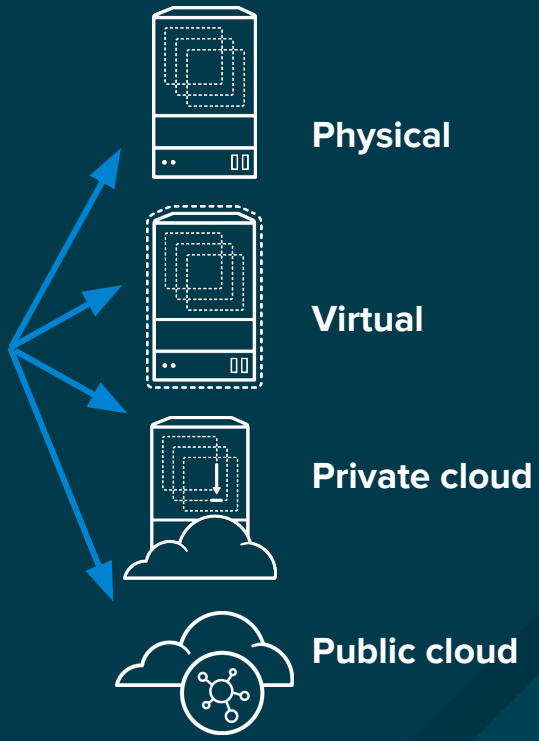
Atomic Host | Red Hat Enterprise Linux

Web & Mobile
Container

Data & Storage
Container

Integration
Container

Business Automation
Container



redhat.

READ THE WHITEPAPER

Ten Layers of Container Security

RED HAT SUMMIT

THANK YOU

plus.google.com/+RedHat

facebook.com/redhatinc

linkedin.com/company/red-hat

twitter.com/RedHatNews

youtube.com/user/RedHatVideos

LEARN. NETWORK.
EXPERIENCE
OPEN SOURCE.