Leveraging the Serverless Architecture for Securing Linux Containers

*Nilton Bila, Paolo Dettori, **Ali Kanso**, Yuji Watanabe[*], Alaa Youssef*

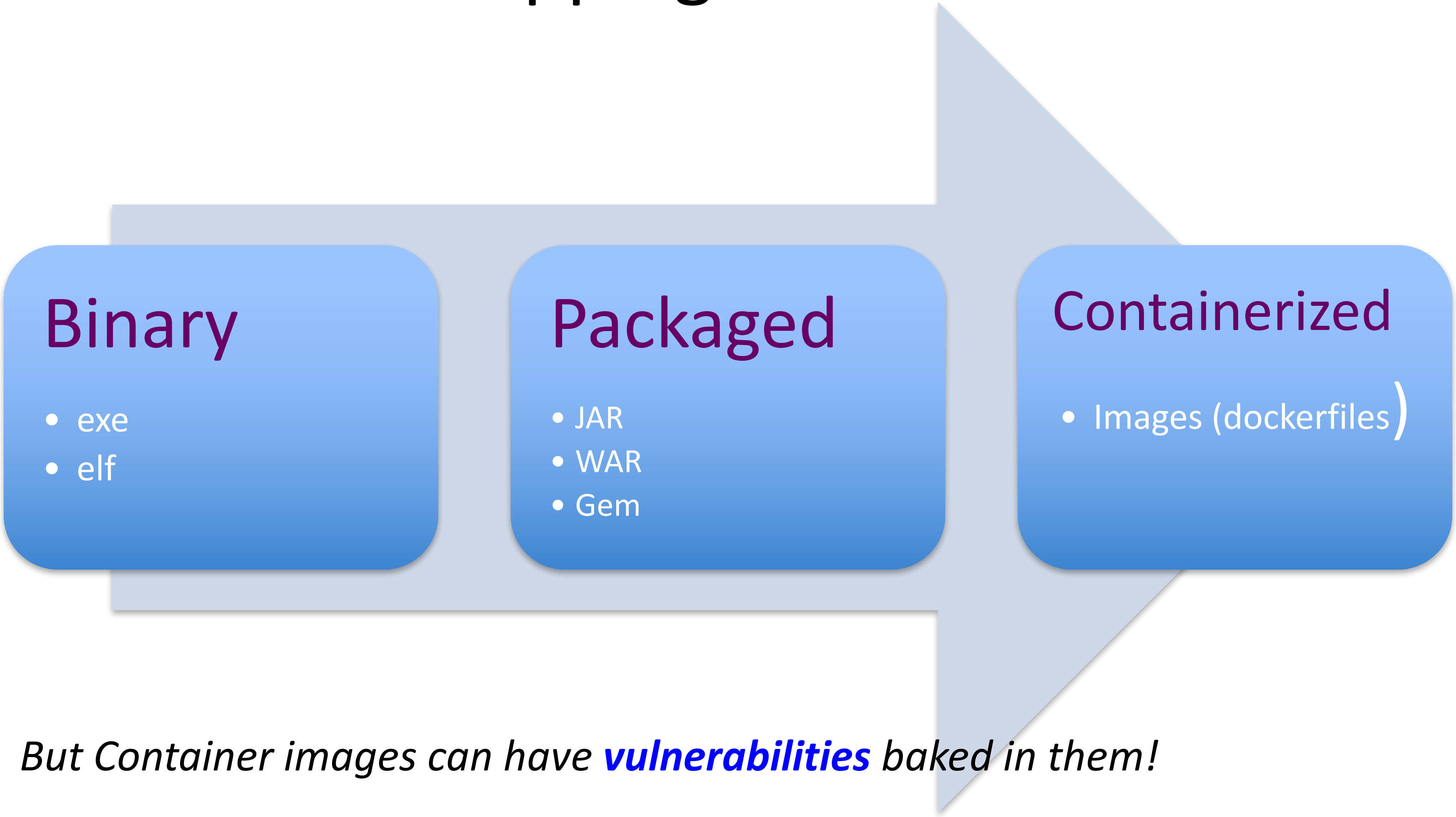
IBM T.J. Watson Research Center – New York

[*]IBM Research – Tokyo, IBM Japan Ltd.

Ali Kanso, PhD
Senior Cloud SW Engineer
IBM T.J. Watson, NY

Leveraging the Serverless Architecture for Securing Linux Containers

Shipping Code



Binary

- exe
- elf

Packaged

- JAR
- WAR
- Gem

Containerized

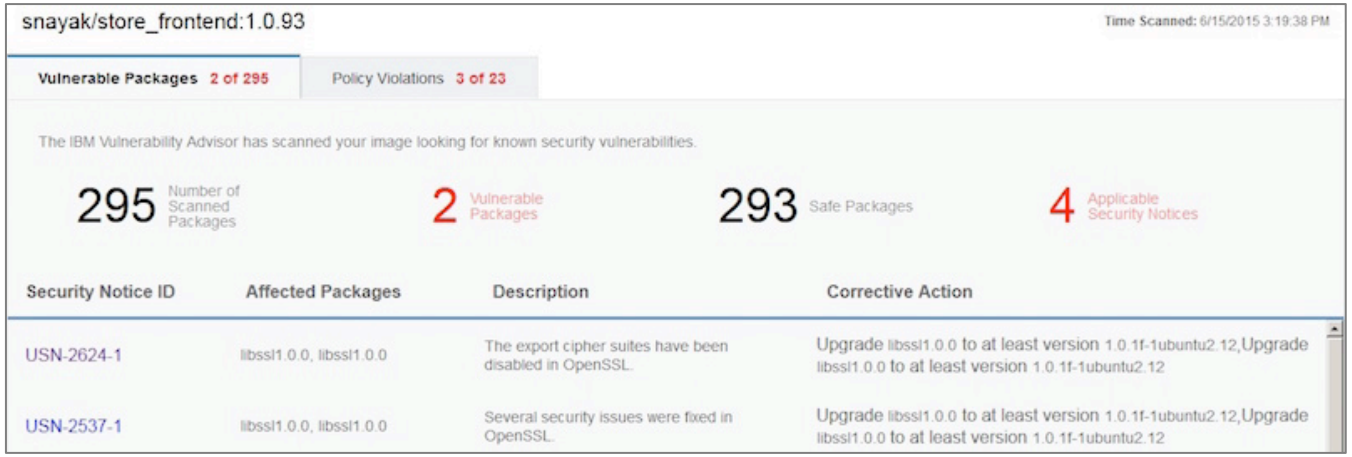
- Images (dockerfiles)

*But Container images can have **vulnerabilities** baked in them!*

Software Vulnerabilities

Scanning for Vulnerabilities

- ✓ Scan images and deployed containers
- ✓ Vulnerabilities in installed software packages
- ✓ Security configuration checks
- ✓ Malware signature detection



snayak/store_frontend:1.0.93

Time Scanned: 6/15/2015 3:19:38 PM

The IBM Vulnerability Advisor has scanned your image looking for known security vulnerabilities.

295 Number of Scanned Packages

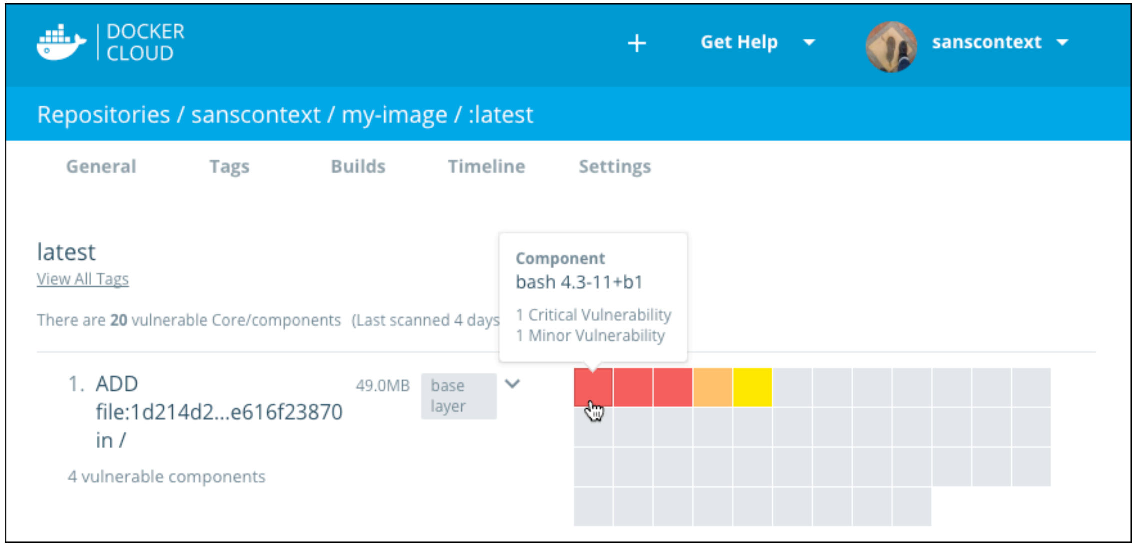
2 Vulnerable Packages

293 Safe Packages

4 Applicable Security Notices

Security Notice ID	Affected Packages	Description	Corrective Action
USN-2624-1	libssl1.0.0, libssl1.0.0	The export cipher suites have been disabled in OpenSSL.	Upgrade libssl1.0.0 to at least version 1.0.1f-1ubuntu2.12,Upgrade libssl1.0.0 to at least version 1.0.1f-1ubuntu2.12
USN-2537-1	libssl1.0.0, libssl1.0.0	Several security issues were fixed in OpenSSL.	Upgrade libssl1.0.0 to at least version 1.0.1f-1ubuntu2.12,Upgrade libssl1.0.0 to at least version 1.0.1f-1ubuntu2.12

IBM Vulnerability Advisor



Repositories / sanscontext / my-image / :latest

latest

There are **20** vulnerable Core/components (Last scanned 4 days

1. ADD file:1d214d2...e616f23870 in / 49.0MB

4 vulnerable components

Component
bash 4.3-11+b1

1 Critical Vulnerability
1 Minor Vulnerability

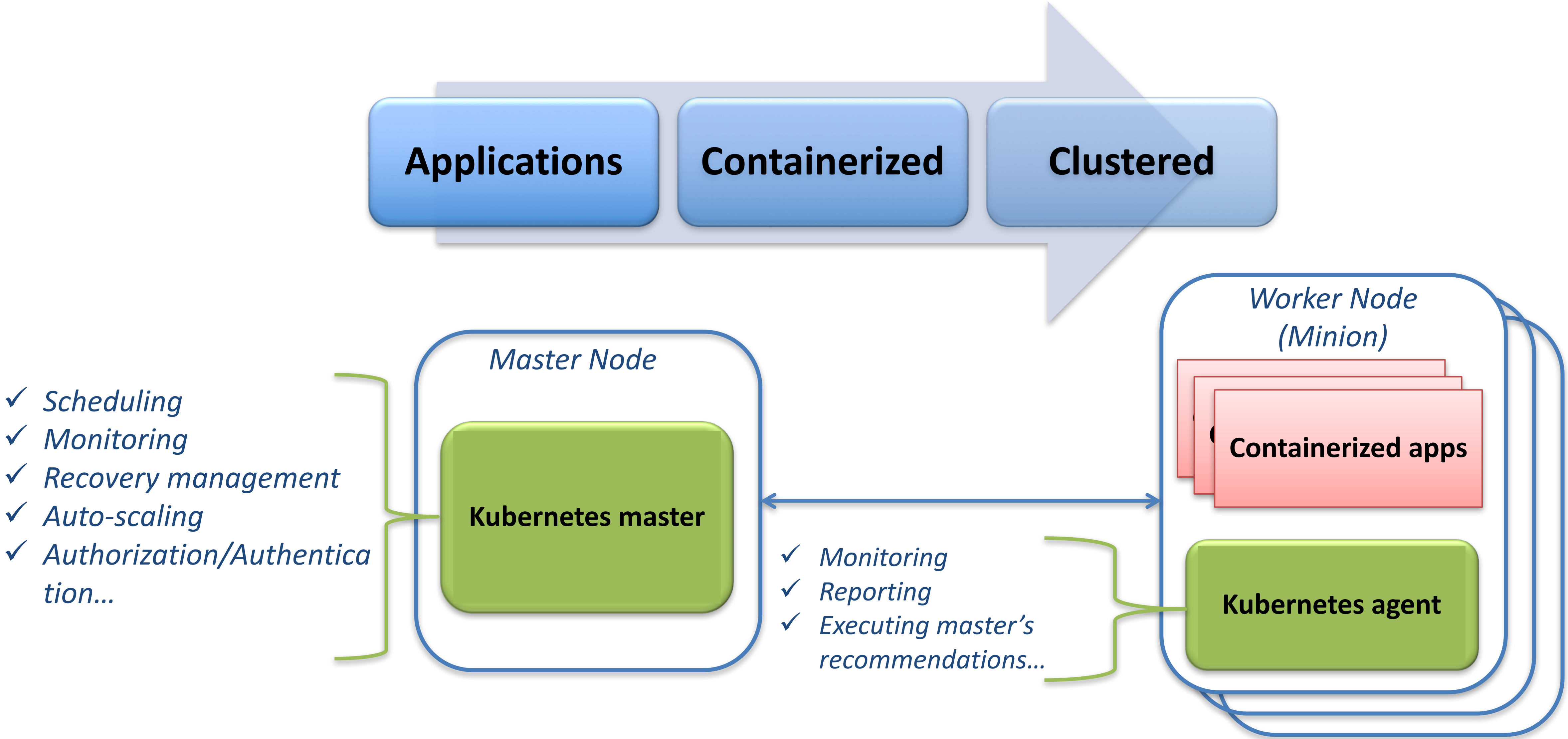
Docker Security Scanning

Clustering Containers

Clustering can be overwhelming

Kubernetes can help

What is Kubernetes?



Applications

Containerized

Clustered

- ✓ *Scheduling*
- ✓ *Monitoring*
- ✓ *Recovery management*
- ✓ *Auto-scaling*
- ✓ *Authorization/Authentication…*

Master Node

Kubernetes master

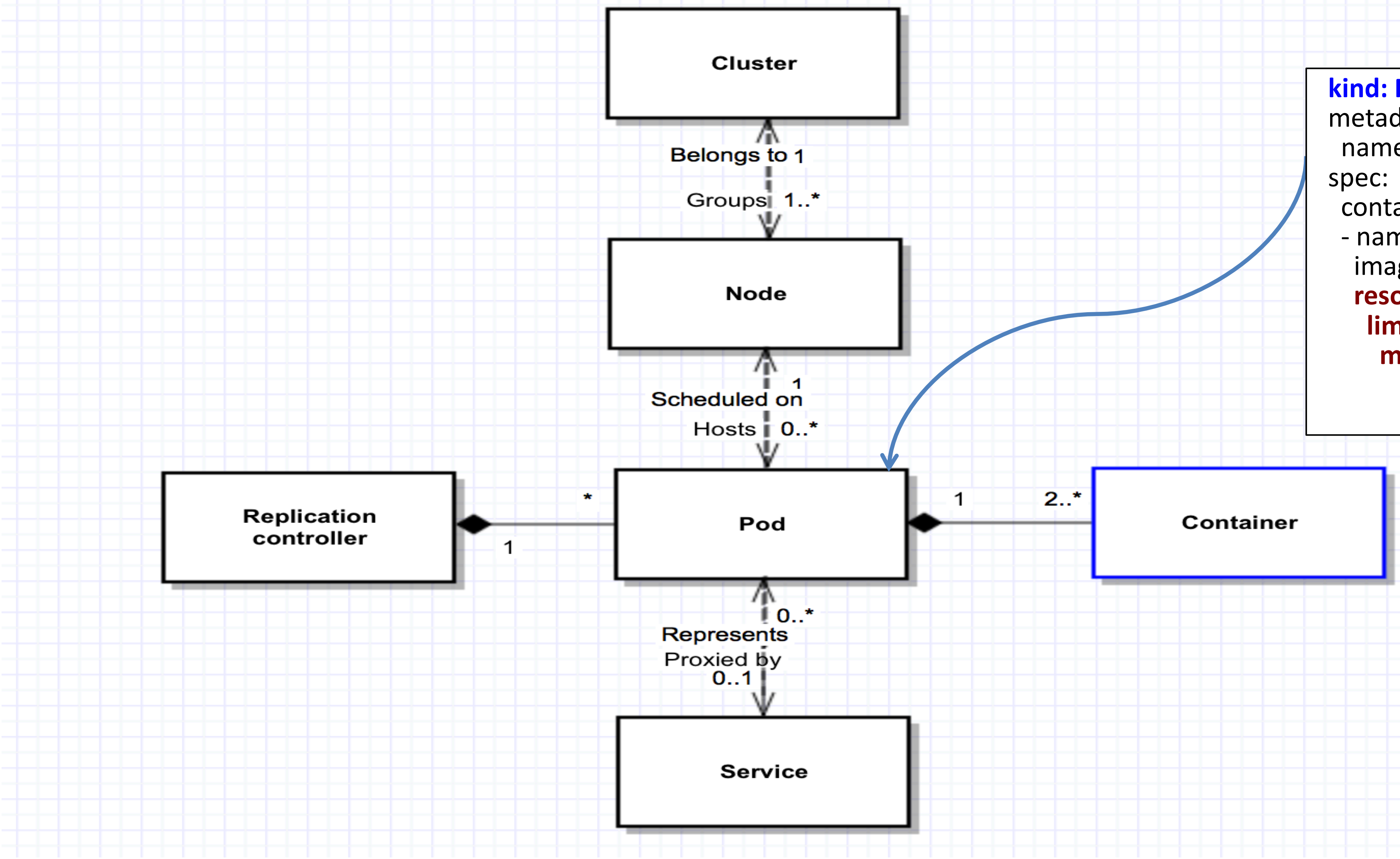
- ✓ *Monitoring*
- ✓ *Reporting*
- ✓ *Executing master's recommendations…*

Worker Node (Minion)

Containerized apps

Kubernetes agent

Kubernetes Resource Organization



Example pod description

```
kind: Pod
metadata:
  name: myPod
spec:
  containers:
  - name: sleep-forever
    image: pause:0.8.0
    resources:
      limits:
        memory: 1000Mi
```

K8s APIs

monolithic v1 API

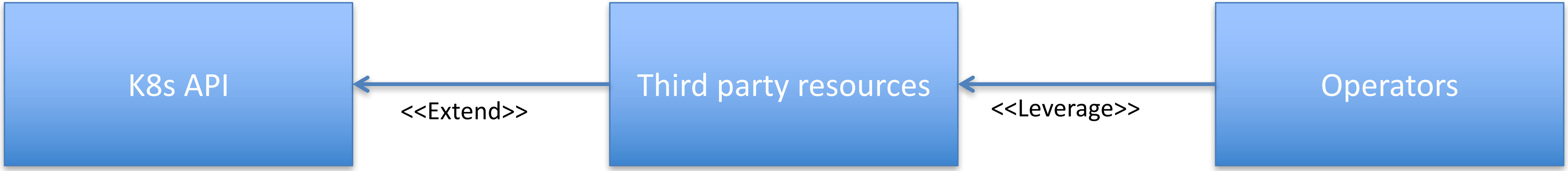
REST path /api/v1

- ✓ Pods
- ✓ Services
- ✓ Replication controllers
- ✓ Resource quotas
- ✓ Nodes
- ✓ Endpoints
- ✓ …

REST path /apis/extensions/$VERSION

- ✓ Deployments
- ✓ HorizontalPodAutoscalers
- ✓ Ingress
- ✓ Jobs
- ✓ DaemonSets
- ✓ Third party resources
- ✓ …

K8s Operators



K8s API

<<Extend>>

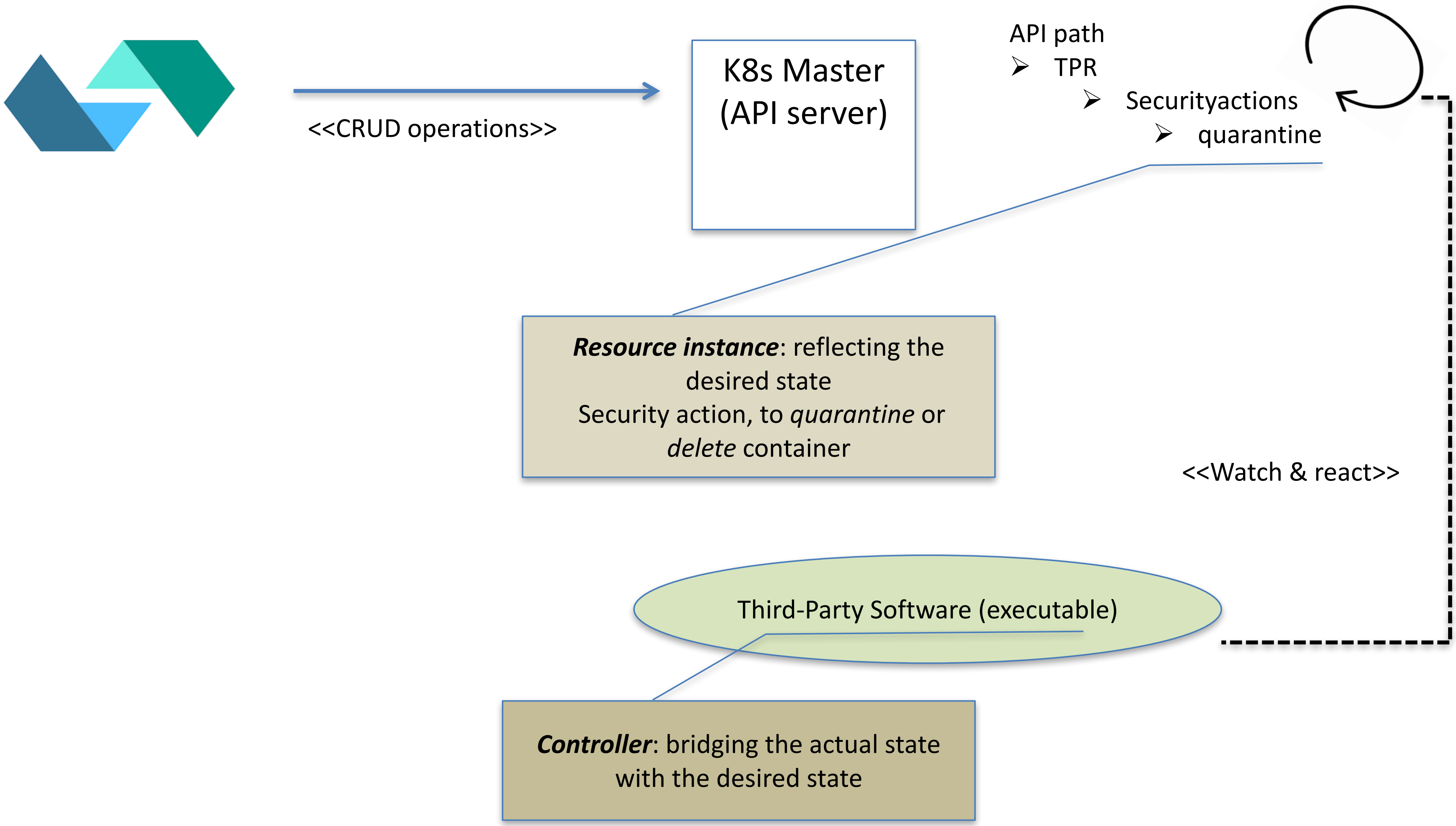
Third party resources

<<Leverage>>

Operators

K8s Third Party Resource (TPR)

*http://192.168.0.15:8080/apis/myorg.com/v1/namespaces/default/**securityactions**/**quarantine***



<<CRUD operations>>

K8s Master
(API server)

API path

- TPR
  - Securityactions
    - quarantine

Resource instance: reflecting the desired state
Security action, to *quarantine* or *delete* container

<<Watch & react>>

Third-Party Software (executable)

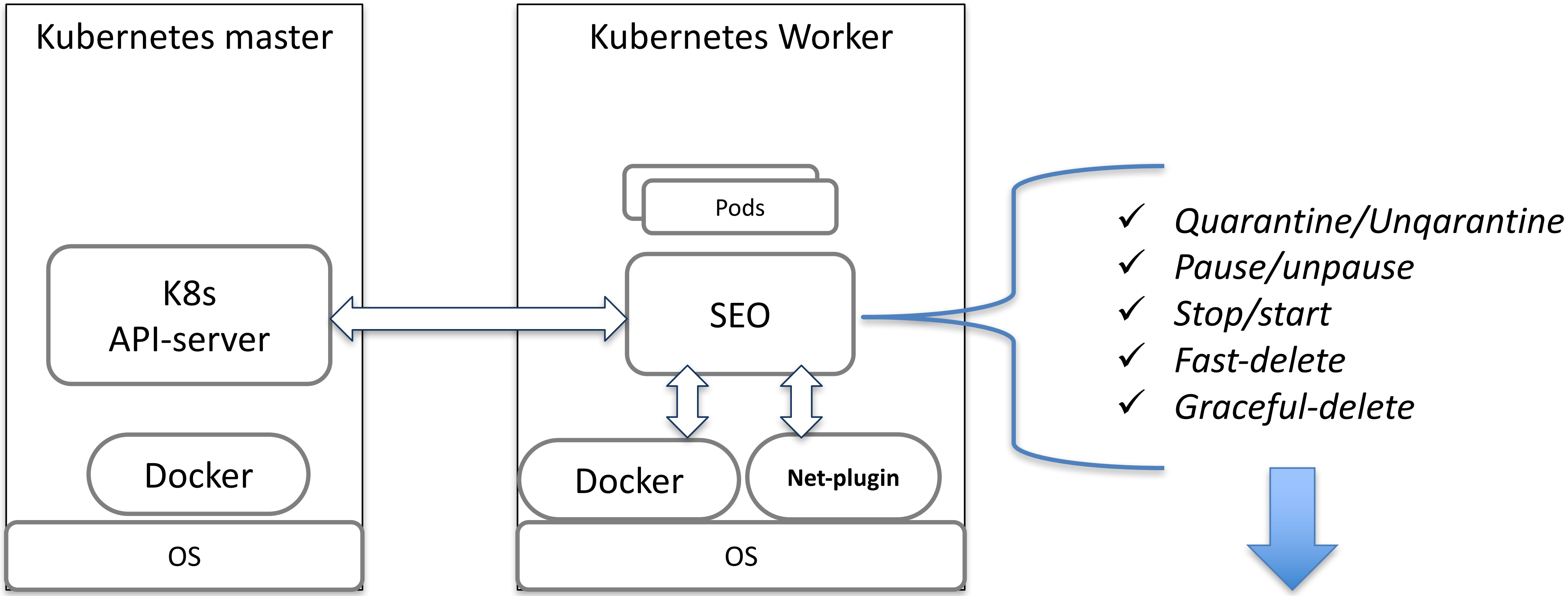
Controller: bridging the actual state with the desired state

Kubernetes Limitation

- K8s does not implement the needed range of actions to contain a threat
  - Limited to: Kill pod, Rolling-Upgrade (involves killing)

We need to have severity-based actions!

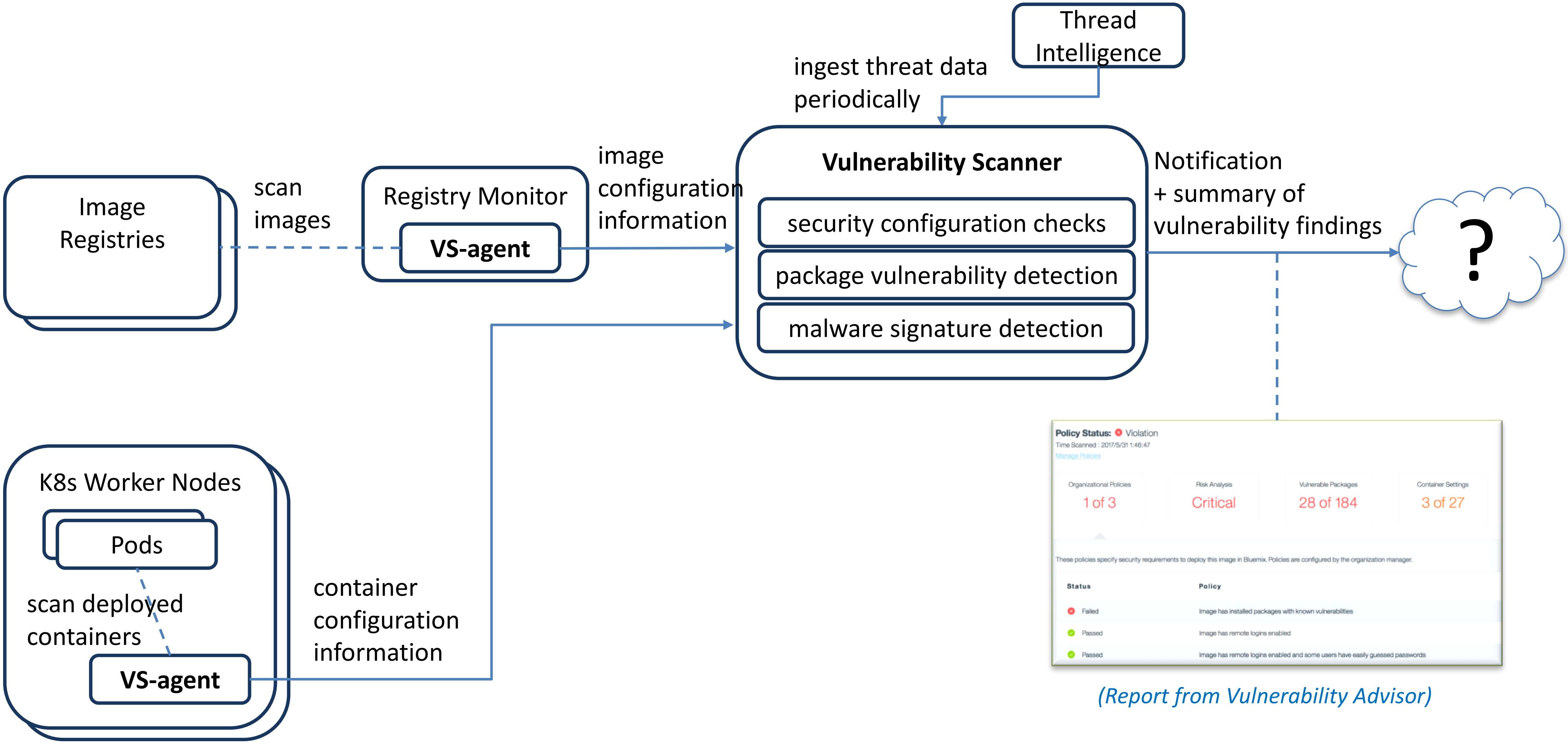
Introducing the Security Enforcement Operator



- ✓ *Quarantine/Unqarantine*
- ✓ *Pause/unpause*
- ✓ *Stop/start*
- ✓ *Fast-delete*
- ✓ *Graceful-delete*

Based on scanning results

Vulnerability Scanner


Thread Intelligence
ingest threat data periodically
Image Registries
scan images
Registry Monitor
VS-agent
image configuration information
Vulnerability Scanner
security configuration checks
package vulnerability detection
malware signature detection
Notification + summary of vulnerability findings
?
K8s Worker Nodes
Pods
scan deployed containers
VS-agent
container configuration information

Policy Status: Violation

Time Scanned : 2017/5/31 1:46:47

Manage Policies

Organizational Policies	Risk Analysis	Vulnerable Packages	Container Settings
1 of 3	Critical	28 of 184	3 of 27

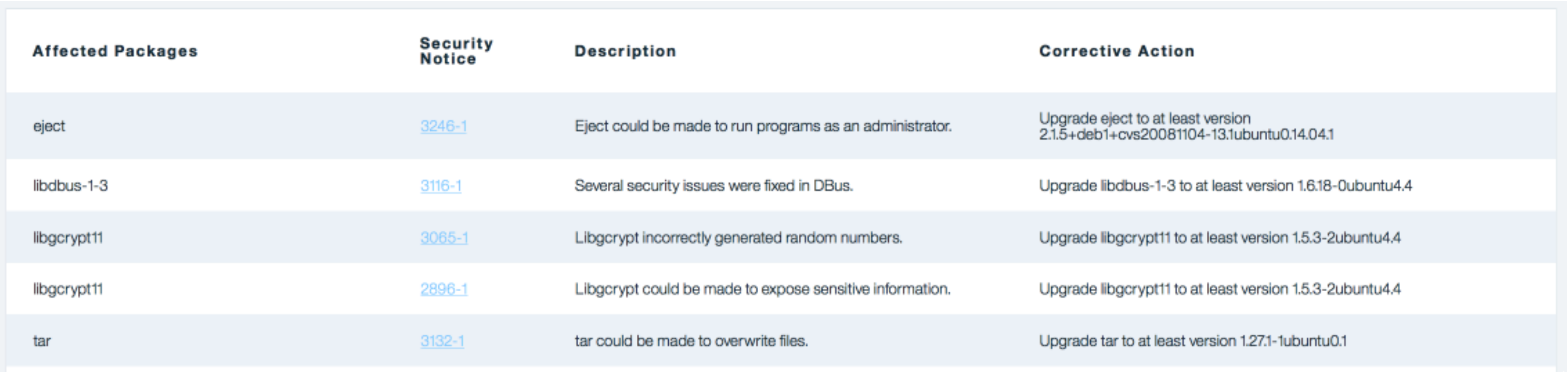
These policies specify security requirements to deploy this image in Bluemix. Policies are configured by the organization manager.

Status	Policy
Failed	Image has installed packages with known vulnerabilities
Passed	Image has remote logins enabled
Passed	Image has remote logins enabled and some users have easily guessed passwords

(Report from Vulnerability Advisor)

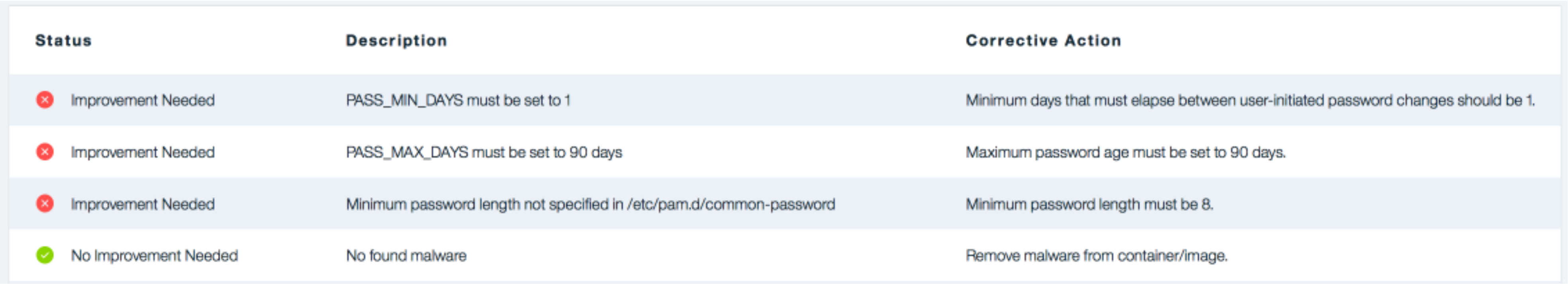
VS Report Example

- Identify specific software package versions in the container with disclosed vulnerabilities



Affected Packages	Security Notice	Description	Corrective Action
eject	3246-1	Eject could be made to run programs as an administrator.	Upgrade eject to at least version 2.1.5+deb1+cvs20081104-13.1ubuntu0.14.04.1
libdbus-1-3	3116-1	Several security issues were fixed in DBus.	Upgrade libdbus-1-3 to at least version 1.6.18-0ubuntu4.4
libgcrypt11	3065-1	Libgcrypt incorrectly generated random numbers.	Upgrade libgcrypt11 to at least version 1.5.3-2ubuntu4.4
libgcrypt11	2896-1	Libgcrypt could be made to expose sensitive information.	Upgrade libgcrypt11 to at least version 1.5.3-2ubuntu4.4
tar	3132-1	tar could be made to overwrite files.	Upgrade tar to at least version 1.27.1-1ubuntu0.1

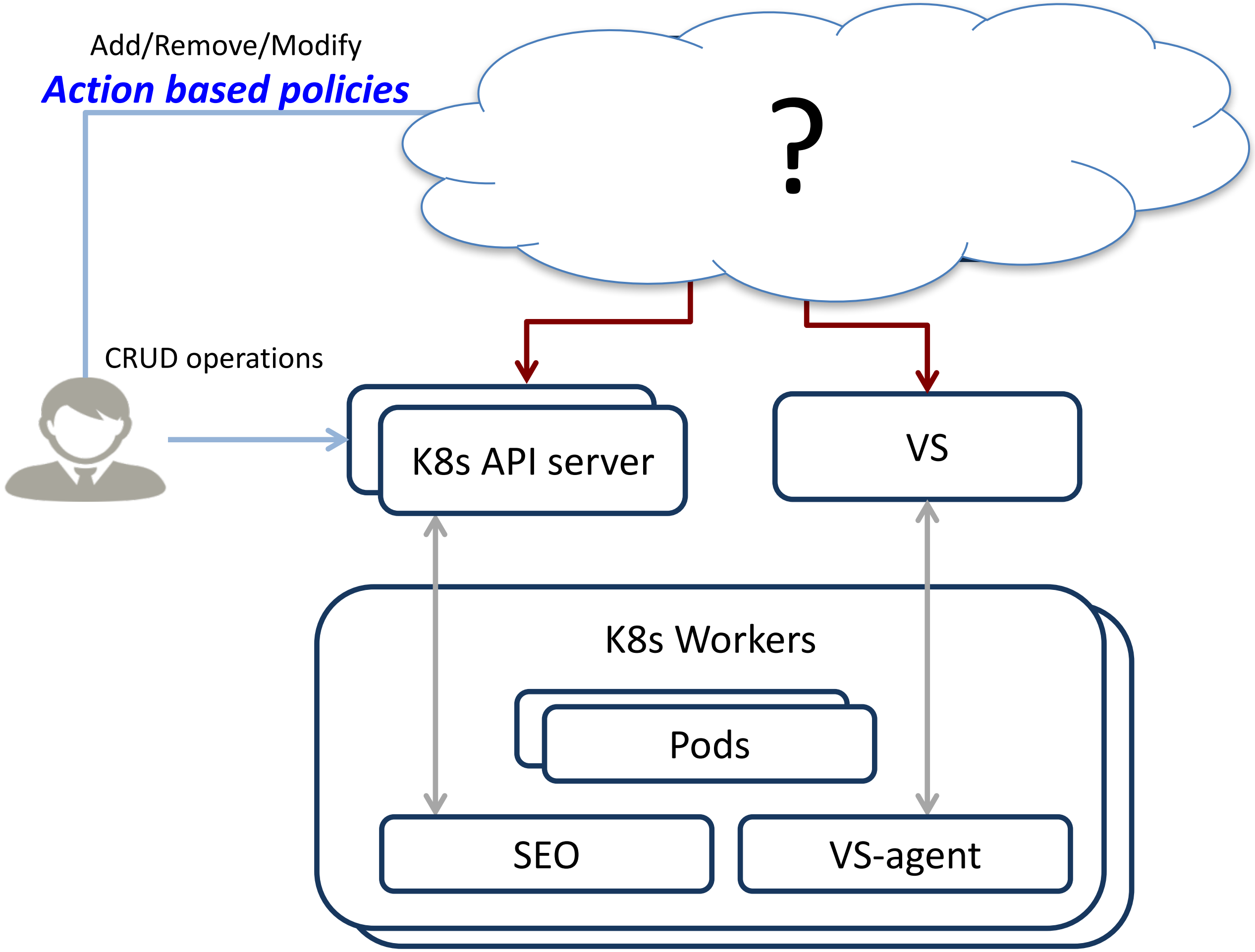
- Identify specific issues with the container configurations



Status	Description	Corrective Action
☒ Improvement Needed	PASS_MIN_DAYS must be set to 1	Minimum days that must elapse between user-initiated password changes should be 1.
☒ Improvement Needed	PASS_MAX_DAYS must be set to 90 days	Maximum password age must be set to 90 days.
☒ Improvement Needed	Minimum password length not specified in /etc/pam.d/common-password	Minimum password length must be 8.
✓ No Improvement Needed	No found malware	Remove malware from container/image.

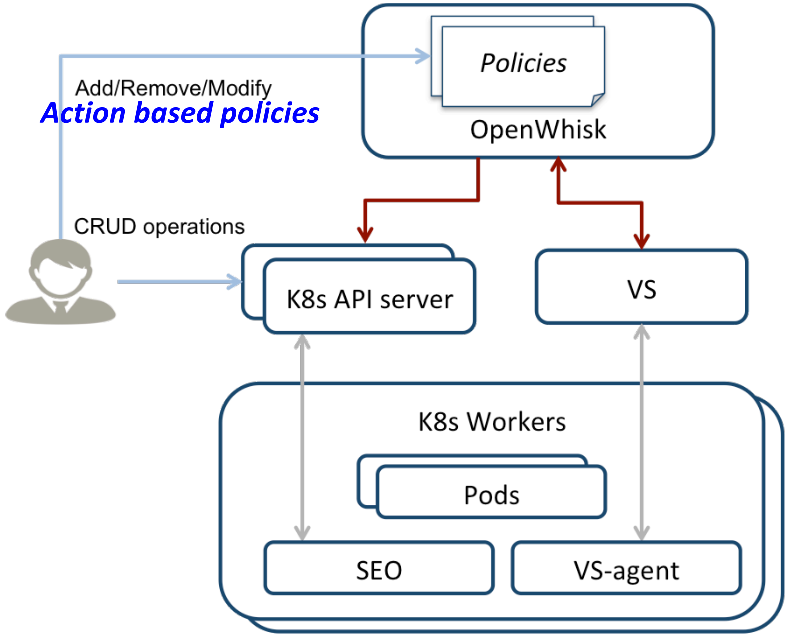
Leveraging the Serverless Architecture for Securing Linux Containers

Introducing OpenWhisk



Why OpenWhisk?

- OpenWhisk is the Glue between VS and K8s, it enables:
  - Different policies for different users
  - Multiple Clusters register to the same OpenWhisk deployment
  - Central point of policy management across clusters



Report API and Notifications on Vulnerability Scanner

- Supports scans for **multiple registered Kubernetes** clusters.
- Provide **RESTful APIs** for access to Vulnerability reports for each container
- Use authentication token to **restrict access** to cluster data at the granularity of Kubernetes namespaces.
- **Notify events** with new **vulnerability findings** to registered OpenWhisk API endpoints.
- Trigger **action invocations** to the **OpenWhisk API endpoints** registered for the Kubernetes cluster.

Notifications

__Per User Policy!__

- User creates action with known URL endpoint:
  - https://openwhisk.ng.bluemix.net/api/v1/web/<USER>/policy
- Vulnerability Scanner posts vulnerability notification to **policy endpoint**

```
{
  “clusterid”: “xyz”,
  “podid”: “nginx- 3382653011-3p4p0”,
  “vulnerability_type”: “package”,
  “vulnerability_status”: “vulnerable”
 }
```

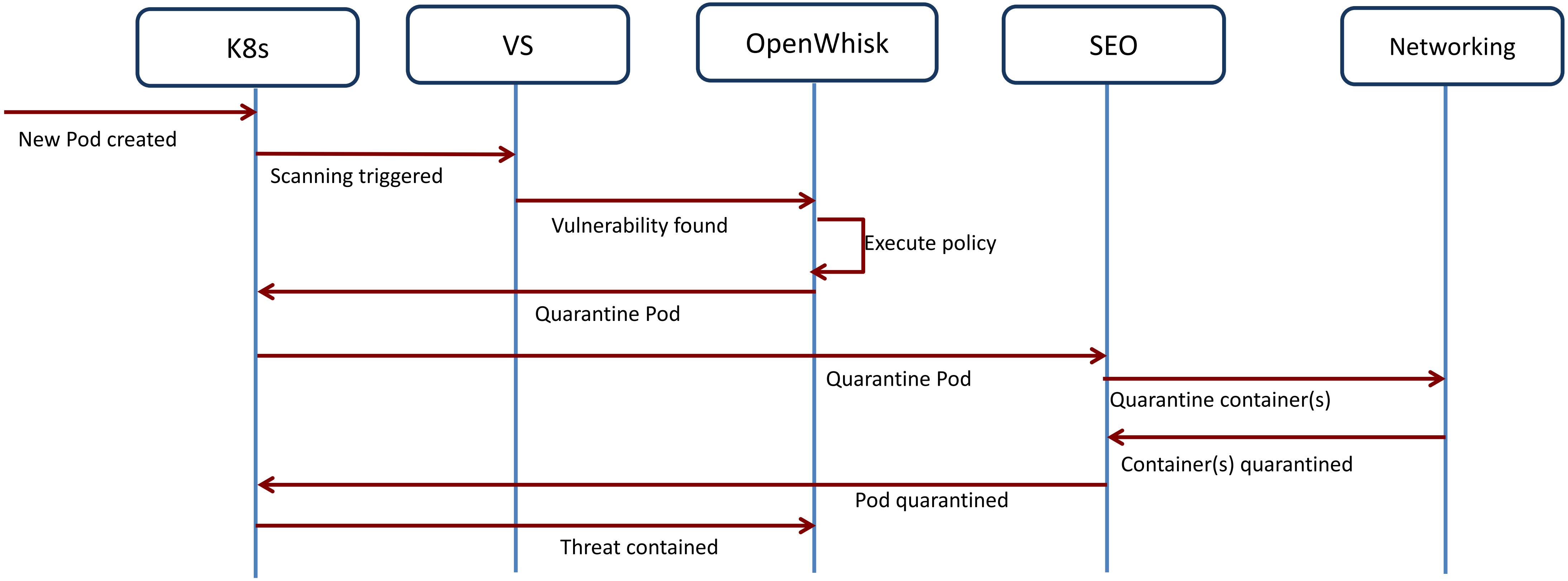
Serverless Policy

User1: marketing

```
import vs
import kubernetes

def main(params):
    findings = vs.get_findings(pod_id, timestamp)
    vulnerable_packages = findings['vulnerable_packages']
    insecure_configs = findings['insecure_configurations']

    if len(vulnerable_packages) > 0:
        kubernetes.snapshot(pod_id) kubernetes.terminate_graceful(pod_id)
        return {'text': 'Deleted pod ' + pod_id }

    if 'remote_shell_installed' in insecure_configs:
        kubernetes~~.quarantine(pod_id~~) Terminate_faste(pod_id)
        return {'text': ~~'Quarantined pod ~~' + pod_id}
                         Terminated pod
    return {'text': 'Container was not modified ' + pod_id}
```

User2: accounting

Interaction Summary



Related Work

Securing Containers

Starlight implements a kernel module that intercepts local operations on each host and passes them to a local agent which in turn passes them to an event processor that analyzes the event and determines whether or not to alert the admin.

LiCShield generates AppArmor profiles by tracing the container engine (Docker daemon) during the build and the execution of the containers.

Using Serverless in the Cloud

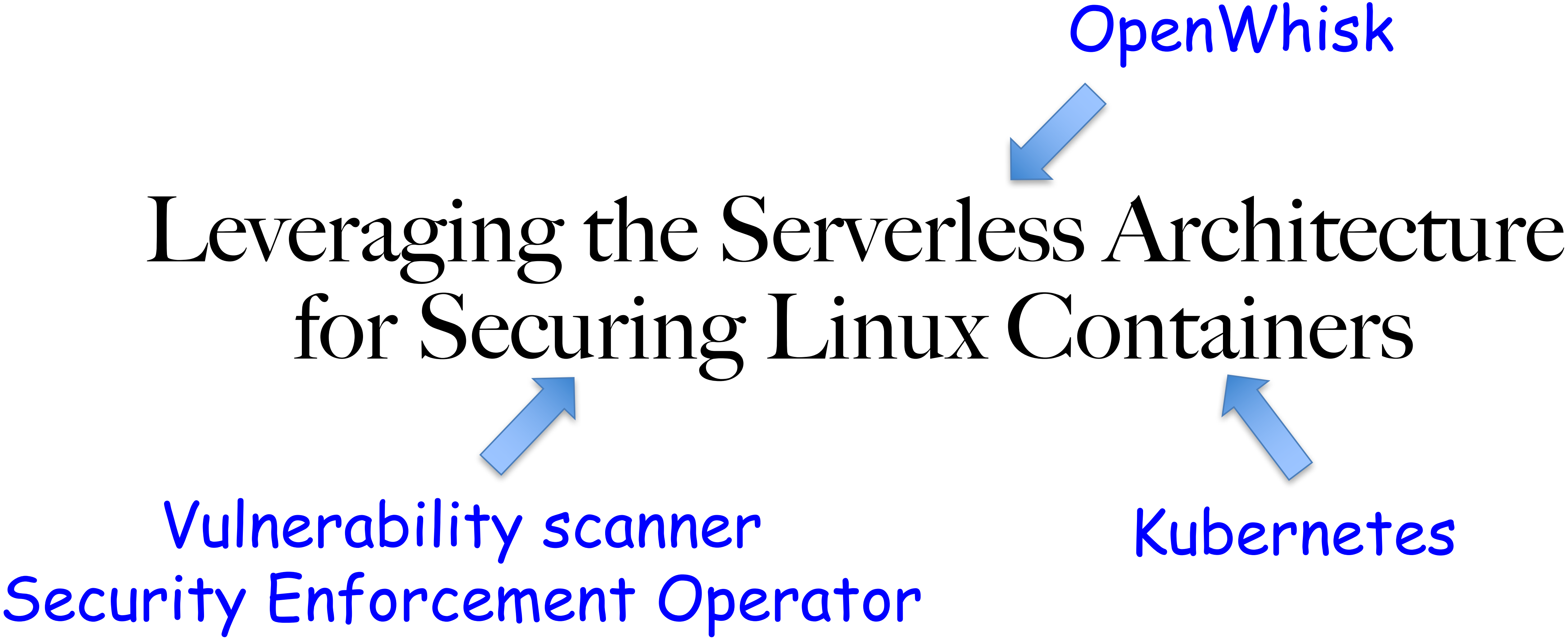
Lambdefy framework to demonstrate the differing requirements between applications deployed to IaaS and applications deployed as a cloud event, and Media Management System for showing high scalability of image resizing tasks on Lambda.

Container Scanners

OpenSCAP (Security Content Automation Protocol) searches for an appropriate fix element, resolves it, prepares the environment, and executes the fix script.

Docker Security Scanning can scan images in private repositories to verify that they are free from known security vulnerabilities or exposures, and report the results of the scan for each image tag

That's it! Questions?



OpenWhisk

Leveraging the Serverless Architecture for Securing Linux Containers

Vulnerability scanner
Security Enforcement Operator

Kubernetes