How to Run Kubernetes Securely and Efficiently

September 2020

Fairwinds

Presenters

ROBERT BRENNAN

Director of Open Source - Fairwinds

Robert drives software initiatives at Fairwinds with a heavy focus on open source. He helps companies adopt and maintain Kubernetes clusters in a secure, scalable, and user-friendly way.

JOE PELLETIER

VP of Product - Fairwinds

Joe heads up product and strategy at Fairwinds. He is passionate about building great teams and solutions that bridge the gap between developers, security, and operations.

Fairwinds

Today’s agenda

- Why organizations choose Kubernetes
- Key challenges
- Technical implications for security and efficiency
- How Configuration Validation can help
- Q&A

Containers change how software is developed

1

Software packaging is shifting left

- Developers take responsibility for containerizing applications as part of CI process
- Developers also responsible for parts of the Kubernetes configuration

2

Container adoption is growing quickly – by 2023, 70% of global enterprises will be running two or more containerized applications, up from 20% last year*

3

Kubernetes is entering the mainstream – and in many cases, through development teams starting new projects

* Gartner, "3 Critical Mistakes That I&O Leaders Must Avoid With Containers. (Available by subscription-only)"

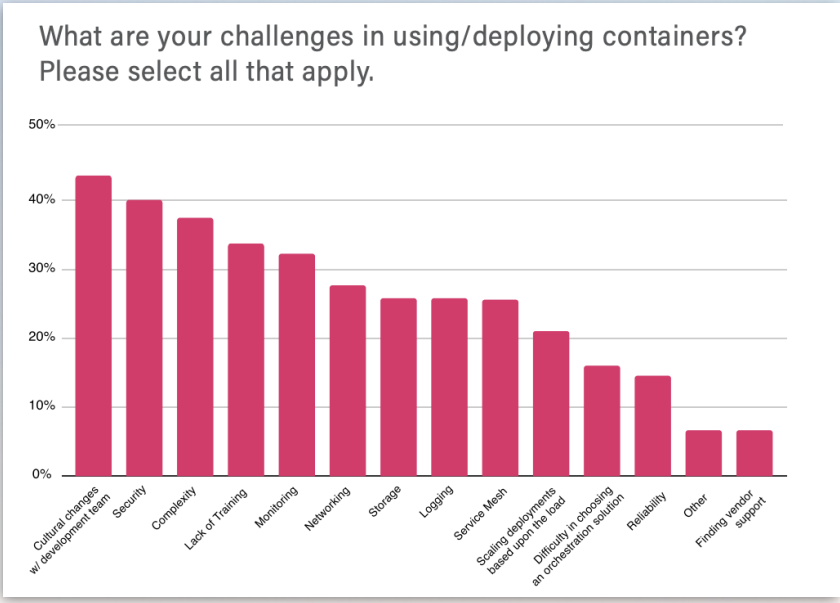
Fairwinds

Containers and Kubernetes enable teams to ship faster

Metric	Before	After	Improvement
Code deployment frequency	Weekly	Daily	**5x more frequently**
Time from commit to deploy to production	7 days	Daily	**86% faster**
Mean Time to Repair (MTTR) for security patches	3 days	15 minutes	**99x decrease in TTR**
Time from request to provision resources	14 days	10-15 mintues	**450x faster**

Source: Amazon Web Services (AWS), “Application Portability with Kubernetes”
https://www.slideshare.net/AmazonWebServices/application-portability-with-kubernetes-cmp310s-aws-reinvent-2018

Challenges with containers



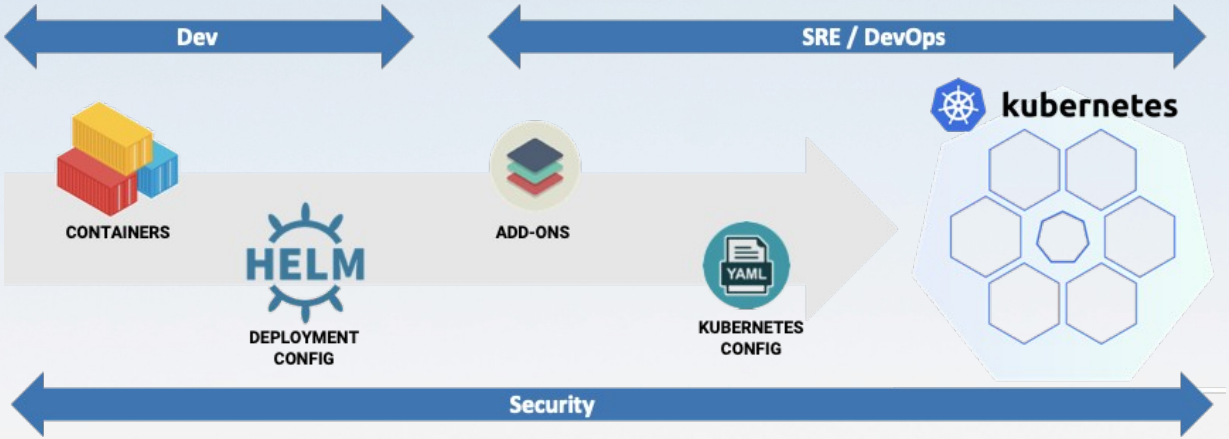
Credit: CNCF Survey 2019, Cloud Native Computing Foundation
https://www.cncf.io/wp-content/uploads/2020/03/CNCF_Survey_Report.pdf

Top 3 Challenges

- Cultural shift moving to containers and Kubernetes
- Security
- Complexity

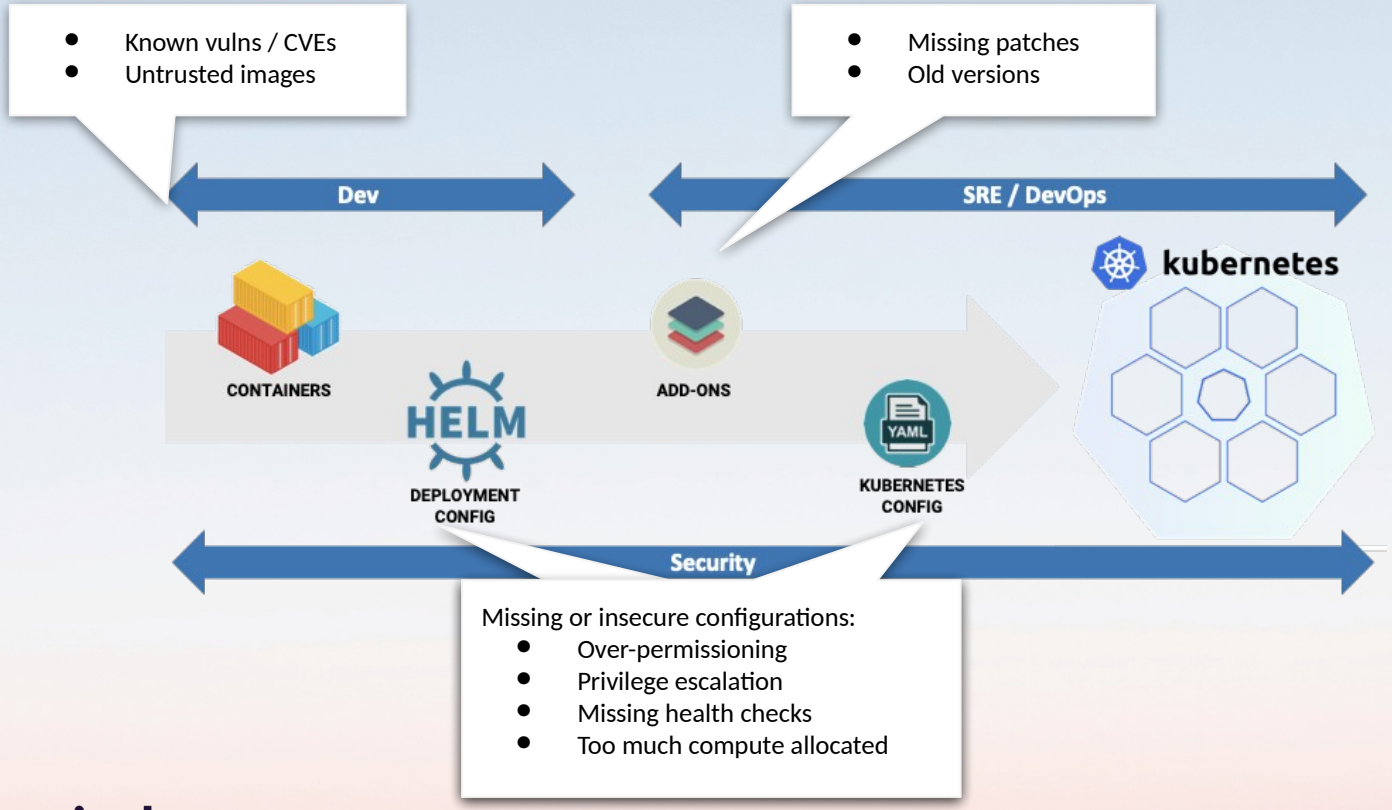
Fairwinds

Cultural shift: Changes to the development pipeline



Fairwinds

Cultural shift: Changes to the development pipeline



Fairwinds

Complexity: Different stakeholders, different needs

"We don't know how much CPU and memory to provision for our application."

The Developer

"Applications are missing critical Kubernetes health checks."

The SRE

"We need to know if we're running images with known vulnerabilities."

The Security Team

"We need reliable infrastructure for our next wave of growth."

The VPE

Fairwinds

Where security and efficiency challenges emerge

Containers

Security

- CVEs
- Image Pull Policies

Deployment Configurations

Security

- Configuration checks

Efficiency

- Health Checks
- Missing Requests and Limits

Kubernetes Clusters

Security

- Add-on Versions
- Risky RBAC Profiles
- Kubelet Config

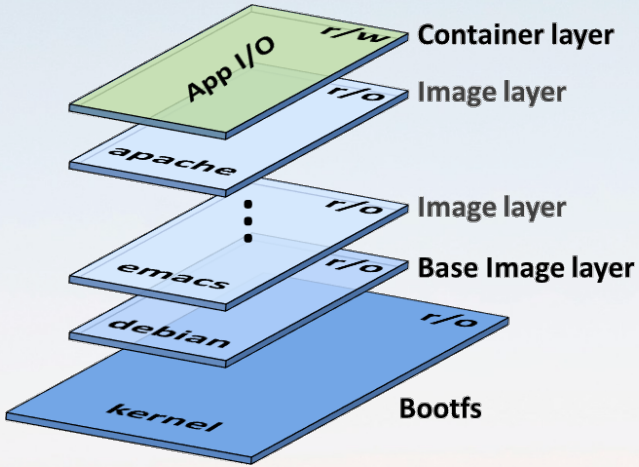
Efficiency

- Resource Optimization Recommendations

Fairwinds

Workload Security:
What are containers & kubernetes configurations?

CVE-201X-1234



Kubernetes Configuration

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: busybox-deployment
spec:
  template:
    spec:
      containers:
        - name: busybox
          image: busybox:latest
          securityContext:
            allowPrivilegeEscalation: true
            privileged: true
            readOnlyRootFilesystem: false
            runAsNonRoot: false
            capabilities:
              add:
              - CHOWN
              - SYS_ADMIN
              - SETUID
              - ALL
```

Fairwinds

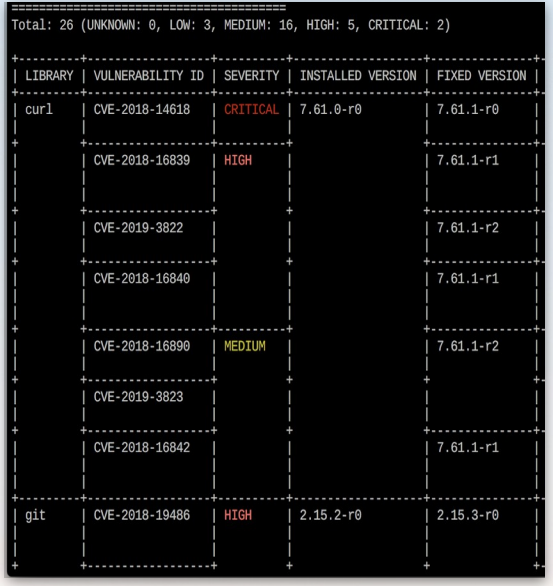
Example: Container vulnerabilities

Problem

- Docker images may use operating systems or libraries that contain vulnerabilities
- New vulnerabilities are being discovered daily

Impact

- Attackers could potentially access the running container, exfiltrate data, or trigger a DoS
- Zero-day attacks could affect running containers that have already passed CI/CD



```
==========================================
Total: 26 (UNKNOWN: 0, LOW: 3, MEDIUM: 16, HIGH: 5, CRITICAL: 2)

+---------+------------------+----------+-------------------+---------------+
| LIBRARY | VULNERABILITY ID | SEVERITY | INSTALLED VERSION | FIXED VERSION |
+---------+------------------+----------+-------------------+---------------+
| curl    | CVE-2018-14618   | CRITICAL | 7.61.0-r0         | 7.61.1-r0     |
+         +------------------+----------+                   +---------------+
|         | CVE-2018-16839   | HIGH     |                   | 7.61.1-r1     |
|         +------------------+          |                   +---------------+
|         | CVE-2019-3822    |          |                   | 7.61.1-r2     |
|         +------------------+          |                   +---------------+
|         | CVE-2018-16840   |          |                   | 7.61.1-r1     |
+         +------------------+----------+                   +---------------+
|         | CVE-2018-16890   | MEDIUM   |                   | 7.61.1-r2     |
|         +------------------+          |                   +               +
|         | CVE-2019-3823    |          |                   |               |
+         +------------------+          +                   +               +
|         | CVE-2018-16842   |          |                   | 7.61.1-r1     |
+---------+------------------+----------+-------------------+---------------+
| git     | CVE-2018-19486   | HIGH     | 2.15.2-r0         | 2.15.3-r0     |
+         +------------------+          +                   +               +
```

Fairwinds

Example: Over-permissioned containers

Problem

- Deployments may violate the principle of least privilege
- Developers may not be certain which permissions are truly needed

Impact

- Attackers can escalate to gain access to entire cluster or underlying nodes
- Honest mistakes can impact other workloads

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: busybox-deployment
spec:
  template:
    spec:
      containers:
        - name: busybox
          image: busybox:latest
          securityContext:
            allowPrivilegeEscalation: true
            privileged: true
            readOnlyRootFilesystem: false
            runAsNonRoot: false
            capabilities:
              add:
              - CHOWN
              - SYS_ADMIN
              - SETUID
              - ALL
```

Example: Health probes

Problem

- Kubernetes uses probes to test the health of an application
- But while highly recommended for production deployments, they are technically optional
- Developers may fail to define liveness and readiness probes

Impact

- Kubernetes may fail to detect a crashed/unresponsive application
- Workloads will not scale appropriately
- Workloads may experience downtime during releases

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: busybox-deployment
spec:
  template:
    spec:
      containers:
        - name: busybox
          image: busybox:latest
          livenessProbe: {}
          readinessProbe:
            httpGet:
              path: /index.html
              port: http
```

Example: Inappropriate resource requests and limits

Problem

- Developers need to specify the amount of CPU and memory their application needs
- But it's hard to know this ahead of time
- Often developers will omit resource specs, or will over-provision resources

Impact

- Kubernetes will fail to kill misbehaving applications
- Memory leaks can eat up all the resources in a cluster ("noisy neighbors")
- Over-provisioning can lead to huge cost overruns
- Under-provisioning can lead to instability

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: busybox-deployment
spec:
  template:
    spec:
      Containers:
        - name: busybox
          image: busybox:latest
          resources:
            requests:
              cpu: "4"
              memory: 20GB
            limits: {}
```

Let’s recap

Technical	Business Impact
Unresolved Container Vulnerabilities	Increased security risk
Over-permissioned Deployments	Increased security risk
Missing Health Probes	Potential for application downtime
Inappropriate Resources Requests and Limits	Potential for cloud cost overruns

Why Configuration Validation?

Kubernetes **configuration validation** enables DevOps teams to maintain consistent configurations for containers, deployments, and cluster infrastructure.

Configuration validation solutions provide organizations with findings, fix recommendations, and Policy-driven controls that reduce risk, save money, and avoid wasting time as applications move from development to operations.

Steps for implementing configuration validation

Get the Data	Analyze, Organize, and Prioritize the Data	Assign Ownership	Remediate
• Find the right tools • Script them to run • Normalize output	• Aggregate & de-dupe • Assign severity • Track and audit the findings • Summarize and prioritize what to do next	• Assign to owners • Integrate with downstream tools (monitoring, chat, ticketing systems)	• Investigate best practices • Document remediation guidance • Train engineers • Make and validate changes

Ways to implement Configuration Validation

1. Build your own tools
2. Use open source
3. Use a purpose-built platform

Build your own tools

Pros

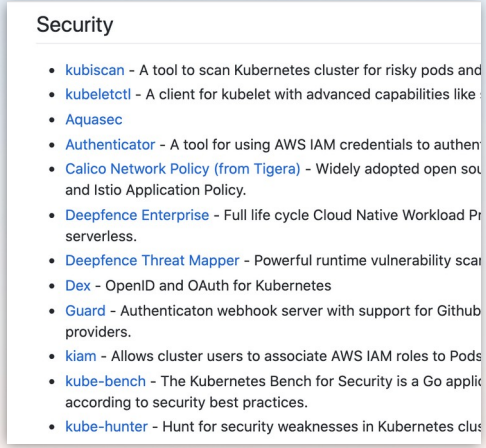
- Fully customized to your needs
- Satisfies engineers who want room to be creative

Cons

- Potentially huge effort, maintenance cost
- No competitive advantage to be gained
- Few needs are truly unique
- May fail to check important things - “unknown unknowns”

Use open source tools

Dozens of open source tools!



Security

- kubiscan - A tool to scan Kubernetes cluster for risky pods and
- kubeletctl - A client for kubelet with advanced capabilities like
- Aquasec
- Authenticator - A tool for using AWS IAM credentials to authen
- Calico Network Policy (from Tigera) - Widely adopted open sou and Istio Application Policy.
- Deepfence Enterprise - Full life cycle Cloud Native Workload Pr serverless.
- Deepfence Threat Mapper - Powerful runtime vulnerability scar
- Dex - OpenID and OAuth for Kubernetes
- Guard - Authenticaton webhook server with support for Github providers.
- kiam - Allows cluster users to associate AWS IAM roles to Pods
- kube-bench - The Kubernetes Bench for Security is a Go applic according to security best practices.
- kube-hunter - Hunt for security weaknesses in Kubernetes clus

Source: https://github.com/ramitsurana/awesome-kubernetes

Pros

- Free!
- Many best-in-class solutions are open source

Cons

- Each tool covers a single use case
- Hard to operationalize
  - different formats
  - need to run on a schedule
  - prioritization and tracking
- Tough to get support
- Still needs extensive research, DIY, and maintenance

Fairwinds

Use a Purpose-built Platform

Pros

- One platform can cover many different use cases
- Expertise is battle-tested across many organizations
- Integrates open source
- Enterprise-grade support

Cons

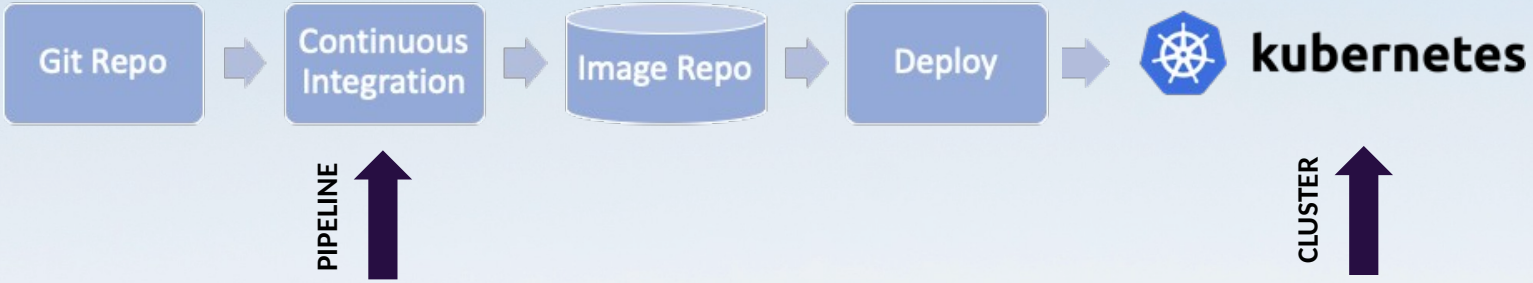
- Costs money
- May have proprietary or commercial components

Things to look for in a Purpose-built Platform

1. Prioritizes findings from multiple open source tools through single pane of glass
2. Offers out-of-the-box integrations with familiar DevOps tools
3. Addresses Containers, Deployment Configuration and Cluster risks
4. Offers remediation advice in-product and through support
5. Integrates Policy-driven features so you can automate enforcement and deployment rules

Fairwinds

Where should you implement Configuration Validation?



Fairwinds

Business benefits of Configuration Validation

Improve Security

- Gain continuous visibility into container, deployment and cluster risks.
- Prioritize and action the riskiest issues.

Reduce Costs

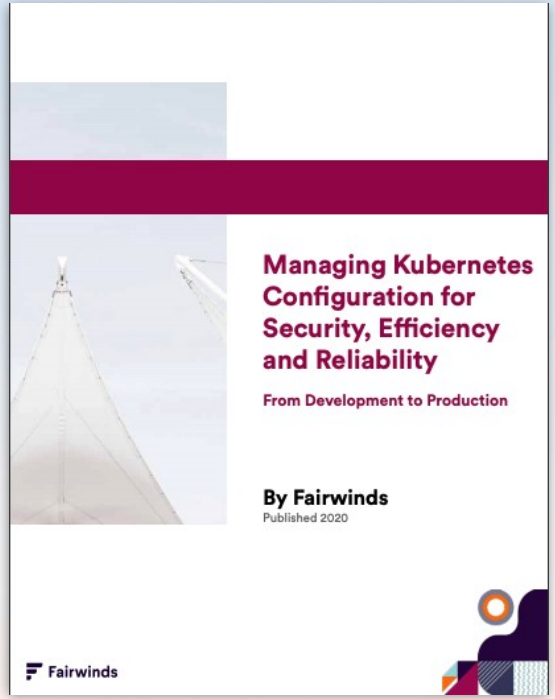
- Set the right amount of CPU and memory so your applications scale efficiently.

Save Time

- Avoid trial-and-error when configuring Kubernetes apps for security and efficiency gains.
- Provide a single-pane-of-glass for DevOps, security, and platform teams to collaborate from.
- Get feedback to app teams in the tools they use

Fairwinds

Fairwinds Guide to Managing Kubernetes Configuration



- Proactively identify Kubernetes security misconfigurations to prevent breaches
- Remove the guesswork when adjusting CPU and memory settings to save time and money
- Avoid downtime

*Learn more at **Fairwinds.com > Resources***

Ready to try Fairwinds Insights?

Free Trial

Fairwinds.com/insights

Fairwinds

Thank You

Questions?

Fairwinds