



# Improving Resilience Through Service Mesh Metrics

DevSecOps Days Pittsburgh 2021

Software Engineering Institute  
Carnegie Mellon University  
Pittsburgh, PA 15213

Copyright 2021 Carnegie Mellon University.

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

The view, opinions, and/or findings contained in this material are those of the author(s) and should not be construed as an official Government position, policy, or decision, unless designated by other documentation.

References herein to any specific commercial product, process, or service by trade name, trade mark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by Carnegie Mellon University or its Software Engineering Institute.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

This material may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other use. Requests for permission should be directed to the Software Engineering Institute at [permission@sei.cmu.edu](mailto:permission@sei.cmu.edu).

Carnegie Mellon® is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

DM21-0567

# This talk will cover an introduction to...

- Resilience
- Metrics
- Service Mesh
- Kubernetes automation
- Prometheus
- Basic scripting concepts

We have a lot of territory to cover.

# Resilience

*“an ability to recover from or adjust easily to misfortune or change” –Merriam-Webster*

- Resilience is also about survival
- Recovery is variable
- Adjustment is variable
- “Easily” is the measure of how resilient something is

Resilience is not easy.



*warhistoryonline.com– B-17 bombers that miraculously made it home*

# Metrics

*“a standard of measurement” –Merriam-Webster*

**Metrics are how we measure progress.**

Things we can measure in DevSecOps: Disk IOPS, Network throughput, IO Latency, CPU Utilization, Memory Utilization, requests per second, response latency, authorizations/failed auths/s, bad requests/s, uptime, MTTR, commits per user, deploys per day, bugs discovered/fixed, security vulns discovered/fixed, protocol overhead, and an unlimited number of additional derivative metrics.

We can measure anything, but what should you be measuring and why and for whom?

What metrics are useful for measuring and improving resilience?

# Service Mesh

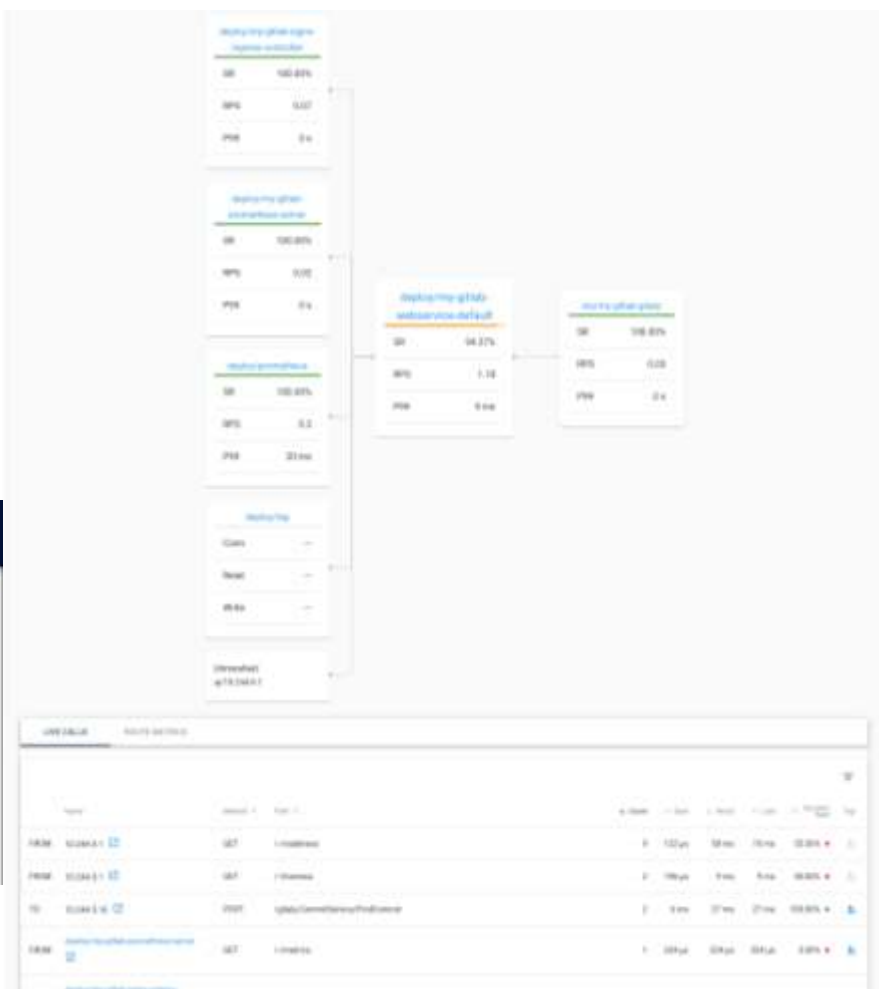
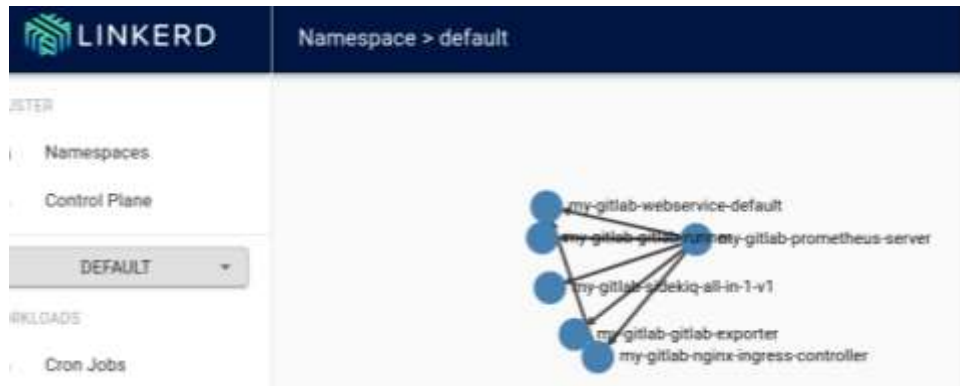
*“... makes running services easier and safer by giving you runtime debugging, observability, reliability, and security...” –LinkerD*

- The service mesh exists at the interface between co-operating services, providing all of the above properties to your existing applications. For us, they provide a lot useful data. (for creating metrics)
- Technology enablers:
  - Kubernetes’ “pod” concept
  - Transparent, inject-able proxies that modify communications pathways at the edge of the pod

Service mesh proxies perform useful transformations on your data, including encryption, service instance routing, HTTP trace logging, and many others.

# Service Mesh with Kubernetes

- Install your application services as normal
- Install and configure your mesh
- “Inject” the mesh into your application
- Gain insights into your operations



# Where does the data come from?

- For the purposes of this discussion, the data comes from the individual services and service mesh proxies, and is reported to a central logging service, called Prometheus.
- Novel metrics can be created from data sources that are not normally collected by the mesh.
  - Application Service logs
  - Security/Access logs
  - Git Commit logs
  - Many others

For further information about a vulnerability, search its ID on: <https://www.exploit-db.com/>

ID	LOCATION	CATEGORY	VULNERABILITY	DESCRIPTION	REFERENCE
48988	10.0.0.1:443	Information Stacktrace	Access to API using service account token	The api server port is accessible. According to pod name, attacker this could expose access to a control of pod cluster.	"/bin/kubectl get pods --namespace=default --output=json --field-selector=status.phase!=Running --no-headers --raw"
48989	Local to Pod (kube- hunter-zmq)	Information Stacktrace	Access to Pod's kube- config	Access to the kube- config file exposes information about the machine associated with the cluster.	10.0.0.1:443
48990	10.0.0.1:443	Information Stacktrace	K8s Version Stacktrace	The Kubernetes version could be obtained from the function endpoint.	10.0.0.1
None	Local to Pod (kube- hunter-zmq)	Access Risk	Cap_Sys_Sem enabled	Cap_Sys_Sem is enabled by default for pods. If an attacker manages to compromise a pod, they could potentially take advantage of this capability to perform network attacks on other pods running on the same node.	
None	Local to Pod (kube- hunter-zmq)	Access Risk	Access to pod's secrets	Accessing the pod's secrets within a compromised pod might disclose sensitive data to a potential attacker.	"/var/run/secrets/kubernetes.io/serviceaccount/token", "/var/run/secrets/kubernetes.io/serviceaccount/namespace"
48991	Local to Pod (kube- hunter-zmq)	Access Risk	Read access to pod's service account token	Accessing the pod service account token gives an attacker the ability to use the service API.	"/var/run/secrets/kubernetes.io/serviceaccount/token"

*"Kube Hunter" – Kubernetes' threat report*

The data comes from wherever it needs to come from, in order to generate useful metrics.

# Plenty of data – How to use it?

## You need useful user stories.

*“As a security professional, I need to know how many failed authorizations we are seeing each day, so that I know when we are facing a specific kind of threat actor.”*

*“As a developer, I need to know what vulnerabilities exist in the dependencies my code relies on, so that we deliver safe software.”*

*“As an automation engineer, I need to know when the system is experiencing abnormal loads, so I can decide whether or not to scale the infrastructure.”*

```
130 ~ /p/p/tales | grype docker:isd.io/buoyantio/emo|voto-emo|l-vcv:11
✓ Vulnerability DB [updated]
✓ Parsed Image
✓ Cataloged packages [132 packages]
✓ Scanned Image [128 vulnerabilities]

NAME          INSTALLED          FIXED-IN          VULNERABILITY          SEVERITY
apt           1.8.2-1
apt           1.8.2-1           1.8.2-2
bash         5.8-4
bind9-host   1:9.11.5.P4+dfsg-5.1+deb10u2  1:9.11.5.P4+dfsg-5.1+deb10u3  CVE-2020-8625      Medium
bind9-host   1:9.11.5.P4+dfsg-5.1+deb10u2  1:9.11.5.P4+dfsg-5.1+deb10u3  CVE-2021-25215     Medium
bind9-host   1:9.11.5.P4+dfsg-5.1+deb10u2  1:9.11.5.P4+dfsg-5.1+deb10u3  CVE-2021-25214     Medium
bind9-host   1:9.11.5.P4+dfsg-5.1+deb10u2  1:9.11.5.P4+dfsg-5.1+deb10u3  CVE-2021-25218     Medium
coreutils    8.38-3
coreutils    8.38-3           (won't fix)
              CVE-2016-2781      Low
curl         7.64.0-4+deb10u1
              (won't fix)
              CVE-2017-10818     Negligible
curl         7.64.0-4+deb10u1
              (won't fix)
              CVE-2020-12298     Unknown
curl         7.64.0-4+deb10u1
              (won't fix)
              CVE-2020-8169     Medium
curl         7.64.0-4+deb10u1
              (won't fix)
              CVE-2020-8177     Medium
curl         7.64.0-4+deb10u1
              (won't fix)
              CVE-2020-8231     Medium
curl         7.64.0-4+deb10u1
              (won't fix)
              CVE-2020-8284     Medium
curl         7.64.0-4+deb10u1
              (won't fix)
              CVE-2020-8285     Medium
curl         7.64.0-4+deb10u1
              (won't fix)
              CVE-2020-8286     Medium
curl         7.64.0-4+deb10u1
              (won't fix)
              CVE-2021-2287a    Medium
curl         7.64.0-4+deb10u1
              (won't fix)
              CVE-2021-2289a    Medium
dnstools     1:9.11.5.P4+dfsg-5.1+deb10u2  1:9.11.5.P4+dfsg-5.1+deb10u3  CVE-2020-6625     Medium
dnstools     1:9.11.5.P4+dfsg-5.1+deb10u2  1:9.11.5.P4+dfsg-5.1+deb10u3  CVE-2021-25215     Medium
dnstools     1:9.11.5.P4+dfsg-5.1+deb10u2  1:9.11.5.P4+dfsg-5.1+deb10u3  CVE-2021-25214     Medium
dnstools     1:9.11.5.P4+dfsg-5.1+deb10u2  1:9.11.5.P4+dfsg-5.1+deb10u3  CVE-2021-25218     Medium
gcc-8-base   8.3.0-6
              (won't fix)
              CVE-2018-1288a    Medium
gcc-8-base   8.3.0-6
              (won't fix)
              CVE-2019-15847    Medium
ppgv         2.2.12-1+deb10u1
              (won't fix)
              CVE-2019-14855     Low
iptables     1.8.2-4
iptables     1.8.2-4
libapt-pkg5.8 1.8.2-1
libapt-pkg5.8 1.8.2-1           1.8.2-2
              CVE-2020-8625     Medium
libbind9-161 1:9.11.5.P4+dfsg-5.1+deb10u2  1:9.11.5.P4+dfsg-5.1+deb10u3  CVE-2020-8625     Medium
libbind9-161 1:9.11.5.P4+dfsg-5.1+deb10u2  1:9.11.5.P4+dfsg-5.1+deb10u3  CVE-2021-25215     Medium
libbind9-161 1:9.11.5.P4+dfsg-5.1+deb10u2  1:9.11.5.P4+dfsg-5.1+deb10u3  CVE-2021-25214     Medium
libbind9-161 1:9.11.5.P4+dfsg-5.1+deb10u2  1:9.11.5.P4+dfsg-5.1+deb10u3  CVE-2021-25218     Medium
libc-bin     2.28-10
              (won't fix)
              CVE-2019-9192     Negligible
```

Anchore's Grype tool, reporting known package vulnerabilities in a running container instance.

# Automation with Kubernetes

Containers  $\subseteq$  Pods  $\subseteq$  Services  $\subseteq$  Deployments  $\subseteq$  Charts  $\subseteq$  Applications

- Kubernetes is known for adding the concepts of Pods, Services, and Deployments on top of Containers
  - Containers are the building blocks of the Application
  - Pods are the smallest deployable unit in Kubernetes (unit of organization of containers)
  - Services abstract features of your containers, such as the Service IP or endpoint
  - Deployments allow abstract reasoning about the runtime state of Services for the purposes of scaling
  - Helm Charts (and related technologies) allow you to reason about your applications, as if they were a set of packages that work together in a specific configuration
  - Applications are made up of potentially numerous Charts, configured to provide a full user experience or capability

**Jobs are the Kubernetes janitor**

# The basic unit of cluster work is the Job

When the cluster needs to run some meta-task that isn't part of the application, it deploys a Job.

As containers are the building blocks of the Application/Cluster, Jobs are built on top of Containers

Jobs can be built for any purpose. We will use them for collecting and reporting metrics.

```
---
apiVersion: batch/v1
kind: Job
metadata:
  name: kube-hunter
spec:
  template:
    spec:
      containers:
      - name: kube-hunter
        image: aquasec/kube-hunter
        command: ["kube-hunter"]
        args: ["--pod"]
        restartPolicy: Never
      backoffLimit: 4
```

*You had one job.*

```
kubectl -n kubeconfig ./kubeconfig describe jobs kube-hunter
Name:         kube-hunter
Namespace:    default
Selector:     controller-uid=9ddbdc51-a225-4eab-b8a5-9ce48cb7d900
Labels:       controller-uid=9ddbdc51-a225-4eab-b8a5-9ce48cb7d900
              job-name=kube-hunter
Annotations:  <none>
Parallelism:  1
Completions:  1
Start Time:   Tue, 15 Jun 2021 02:34:54 +0000
Pod Statuses: 1 Running / 0 Succeeded / 0 Failed
Pod Template:
  Labels:  controller-uid=9ddbdc51-a225-4eab-b8a5-9ce48cb7d900
          job-name=kube-hunter
  Containers:
    kube-hunter:
      Image:   aquasec/kube-hunter
      Port:   <none>
      Host Port: <none>
      Command:
        kube-hunter
      Args:
        --pod
      Environment: <none>
      Mounts: <none>
      Volumes: <none>
  Events:
Type     Reason          Age   From          Message
----     -
Normal   InjectionSkipped 19s   linkerd-proxy-injector Linkerd sidecar proxy injection skipped
Normal   SuccessfulCreate 19s   job-controller   Created pod: kube-hunter-znms
```

*This Job produced the Kube-Hunter logs from slide 7.*

# Jobs can be repeated (Cron Jobs)

```
apiVersion: batch/v1
kind: CronJob
metadata:
  name: hello
spec:
  schedule: "0 * * * *"
  jobTemplate:
    spec:
      template:
        spec:
          containers:
            - name: hello
              image: busybox
              imagePullPolicy: IfNotPresent
              command:
                - /bin/sh
                - -c
                - date +"%H:%M"; echo "0'clock and all is well."
          restartPolicy: OnFailure
```

A simple cron job illustrates the point nicely. We can use the same approach to collect and report metrics

or not. In the day the watchmen looked forth from towers, when aught was expected. And, though clouds might lower at midnight, and thunder mutter in the distance, and sometimes marauding bands threaten, a sense of security and of safety came when the watchman, faithful and true, called aloud: "Twelve o'clock, and all's well!" It behooves us sometimes, when perplexities or troubles gather, or when danger seems to stalk abroad, to call out to him who watches: "Watchman, what of the night?" And you, gentlemen who will watch and guard the future of our state and country, will have to answer the question, whether all is well.

# Creating custom automations

```
~/D/p/R/tales p ? gype docker.io/buoyantio/emoji:voto-emoji-svc:v11 -o json | jq '.matches[].vulnerability | select(.severity == "High")' > results.json
✓ Vulnerability DB [no update available]
✓ Parsed image
✓ Cataloged packages [152 packages]
✓ Scanned image [129 vulnerabilities]
```

JSON Data is extremely common. Get used to working with it.

- A popular data source for basic metrics reporting for cluster applications today is Prometheus. (Next slide)
- Prometheus provides a number of clients for connecting to the service, reporting and querying data.
- Clients are available for various languages and can be used to create various forms of metrics and automation jobs.

```
{
  "id": "CVE-2021-3520",
  "dataSource": "https://security-tracker.debian.org/tracker/CVE-2021-3520",
  "namespace": "debian:10",
  "severity": "High",
  "uris": [
    "https://security-tracker.debian.org/tracker/CVE-2021-3520"
  ],
  "cvss": [],
  "fix": {
    "versions": [
      "1.8.3-1+deb10u1"
    ],
    "state": "fixed"
  },
  "advisories": [
    {
      "id": "DSA-4919-1",
      "link": "https://security-tracker.debian.org/tracker/DSA-4919-1"
    }
  ]
}
~/D/p/R/tales p ? cat results.json | jq '.id' | count
14
```

Creating a metric to count current “High” CVEs.

# ConfigMaps, Custom Containers, and Scripts

- ConfigMaps are a useful place to store simple scripts that can be executed from an off-the-shelf container.
- Custom containers provide ultimate flexibility for running your jobs.
  - Implement a Prometheus client metric in the language of your choice
  - Place it in a container or a ConfigMap
  - Create a job to run it
  - Create a Cron job to schedule it
  - Report results and/or push log files to your collection point.

```
from prometheus_client import start_http_server, Summary
import random
import time

# Create a metric to track time spent and requests made.
REQUEST_TIME = Summary('request_processing_seconds', 'Time spent processing request')

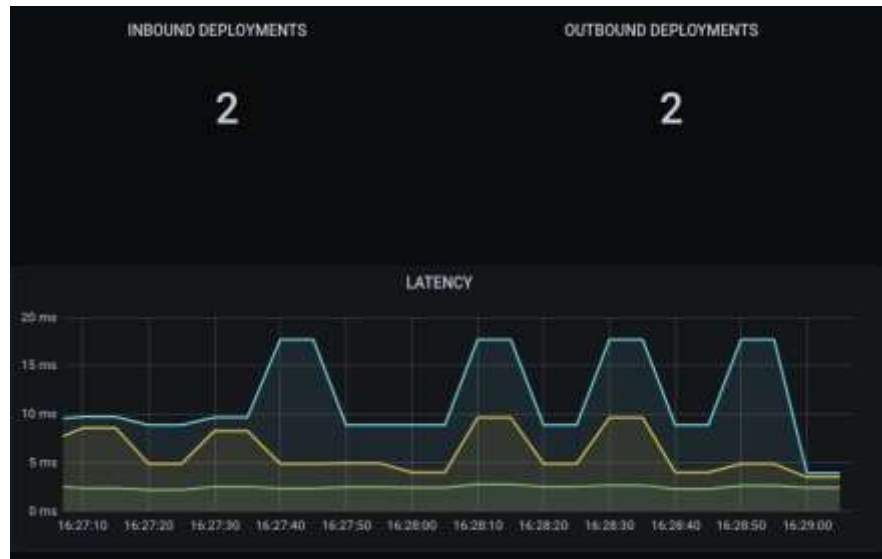
# Decorate function with metric.
@REQUEST_TIME.time()
def process_request(t):
    """A dummy function that takes some time."""
    time.sleep(t)

if __name__ == '__main__':
    # Start up the server to expose the metrics.
    start_http_server(8000)
    # Generate some requests.
    while True:
        process_request(random.random())
```

A sample script, a starting point for developing a custom metric.

# Displaying Metrics

- User engagement is crucial in developing a visual form for your data that gets the users the information they need, in a format that has meaning and facilitates decisions.
- Many tools to choose from for displaying information.
- The “dashboard” is a common tool.



You could visualize your metrics like this, but does it convey the necessary message to your user?

Never lose sight of the target for the metrics and the automations you create.

The end