

CAPSEC

Containerized Application Security
for Real Time Software Upgrade & Patching

Adrian Chavez

SANDIA NATIONAL LABORATORIES
DOE PACT VIRTUAL SHOWCASE

- Software patching disrupts uptime, resulting in increased costs and lost revenue.
- Many industries require near continuous uptimes, delaying patching.
- Delays in patching result in increased vulnerabilities to cyber attack

Thesis: Use of Containerization eliminates costs and downtime associated with updating/patching software.

- Increase resiliency to cyber attack
- Detect and isolate cyber attacks

Challenges:

- Accounts for Stateful (i.e.: ModBus) and Stateless (i.e.: Web Applications)
- The computing resource limitations of Stateful devices prevent use of IT continuous patching solutions
- Orchestration and containerization of existing software
- Fault tolerance

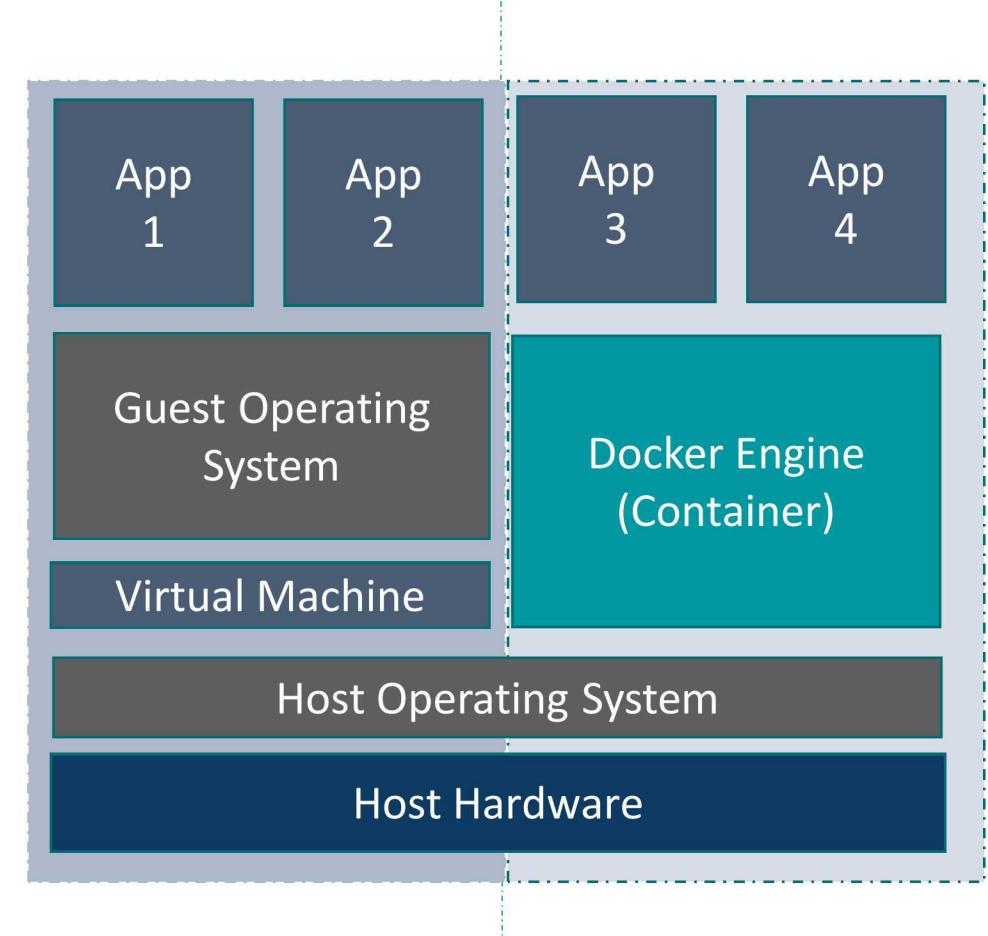
Containerization

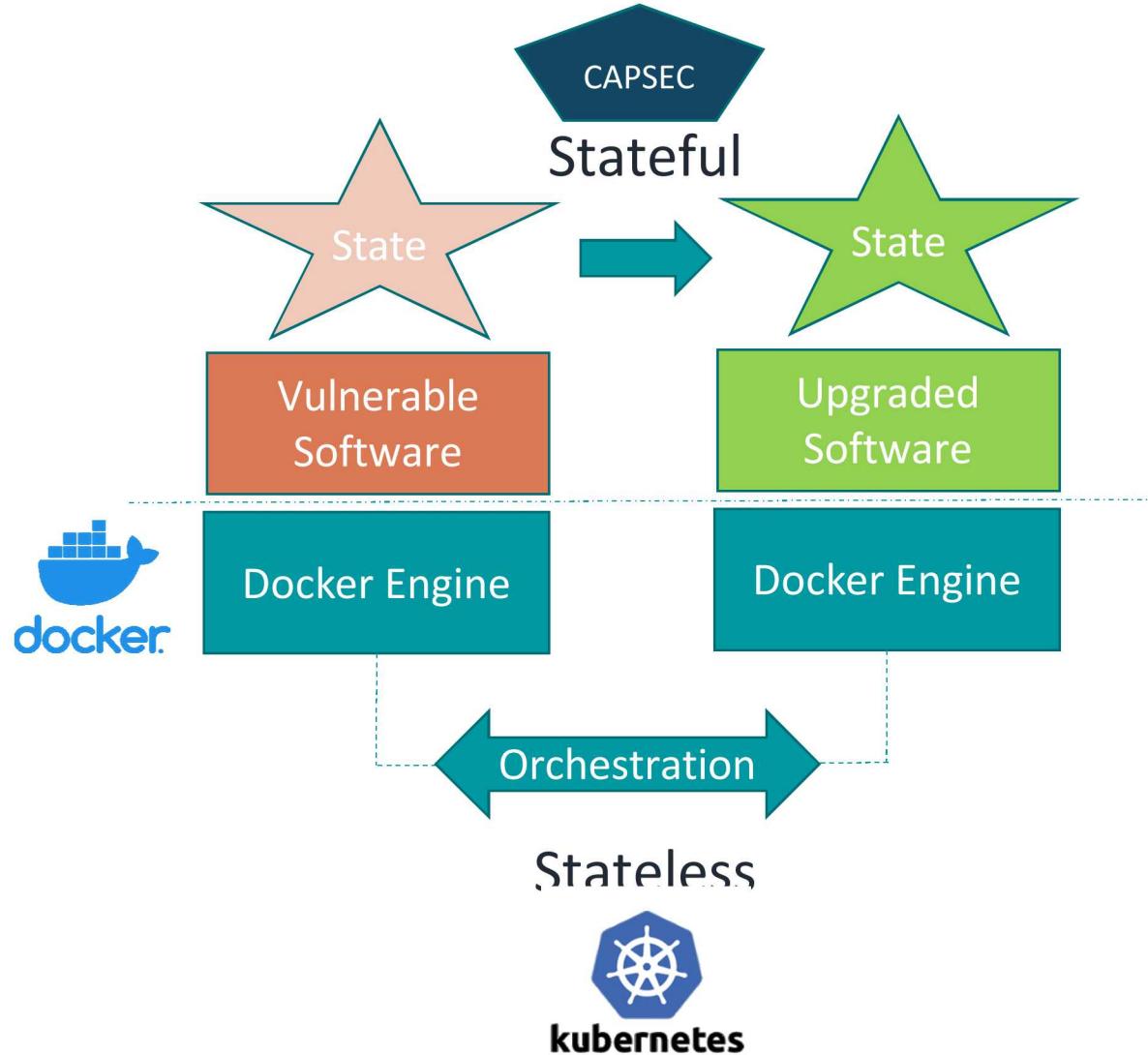
- Containers bundle all software, libraries, environment variables and configuration files

- Use less resources than VMs due to independence from full OS.

- Most common:

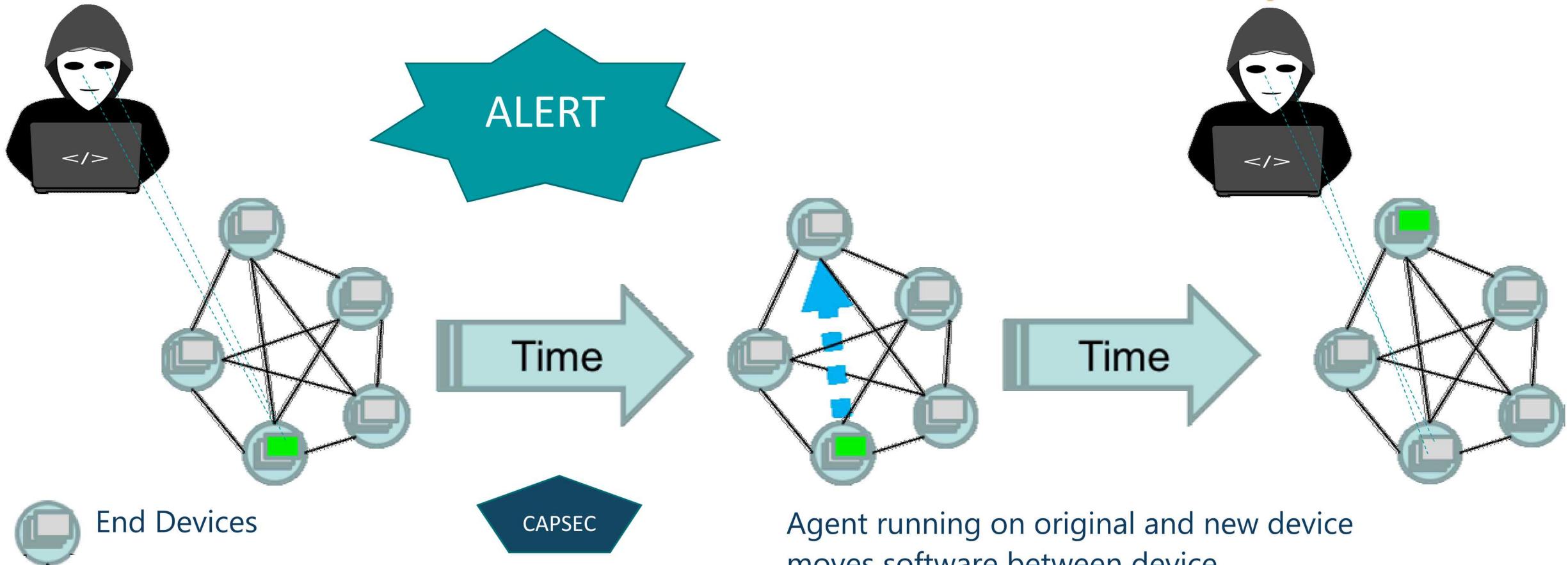
- Open source Docker Engine
- Open source container-orchestration system Kubernetes





- Containerized architecture
- Replicates Stateful environments
- TLS Agents on parallel containers

Migrate containers to implement moving target defense



Development History & Results

CAPSEC

Principal Investigator Adrian Chavez

- Ph.D. Computer Science,
University of California, Davis
- Principal Member of Technical Staff,
Cybersecurity R&D

Developmental History

2016 First filing (July)
2018 Project kick-off (May)
US patent granted (July)
2020 Established proof-of-concept (Feb)

Funding History: 2018-Present

\$2.5M, DOE CESER Office

Lead
 Sandia
National
Laboratories

Partners
 SEL SCHWEITZER
ENGINEERING
LABORATORIES



 GRIMM



Research Team
Ryan Birmingham
Jasenko Hasic
Jaykumar Patel
Kandy Phan
William Stout

Technology Readiness Level (TRL):
TRL 4 as of June 2020

IP Protection
US Patent No. 10,037,203
Real Time Software Upgrade

Market Validation

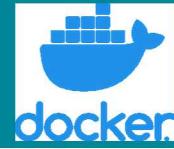


 GRIMM



TECHNICAL REQUIREMENTS

Live Upgrade & Migration Software



System of Multiple Units of Hardware

-Or- Software Replicates Nodes

BENEFITS

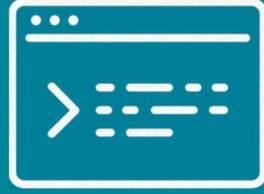
Improved resilience

Active patching & remediation during cyber attack

Decreased maintenance time



Expand application management support and fault tolerance



Test and evaluate CAPSec within representative environments



Interoperate CAPSec with vendor commercial products



Collect data and metrics

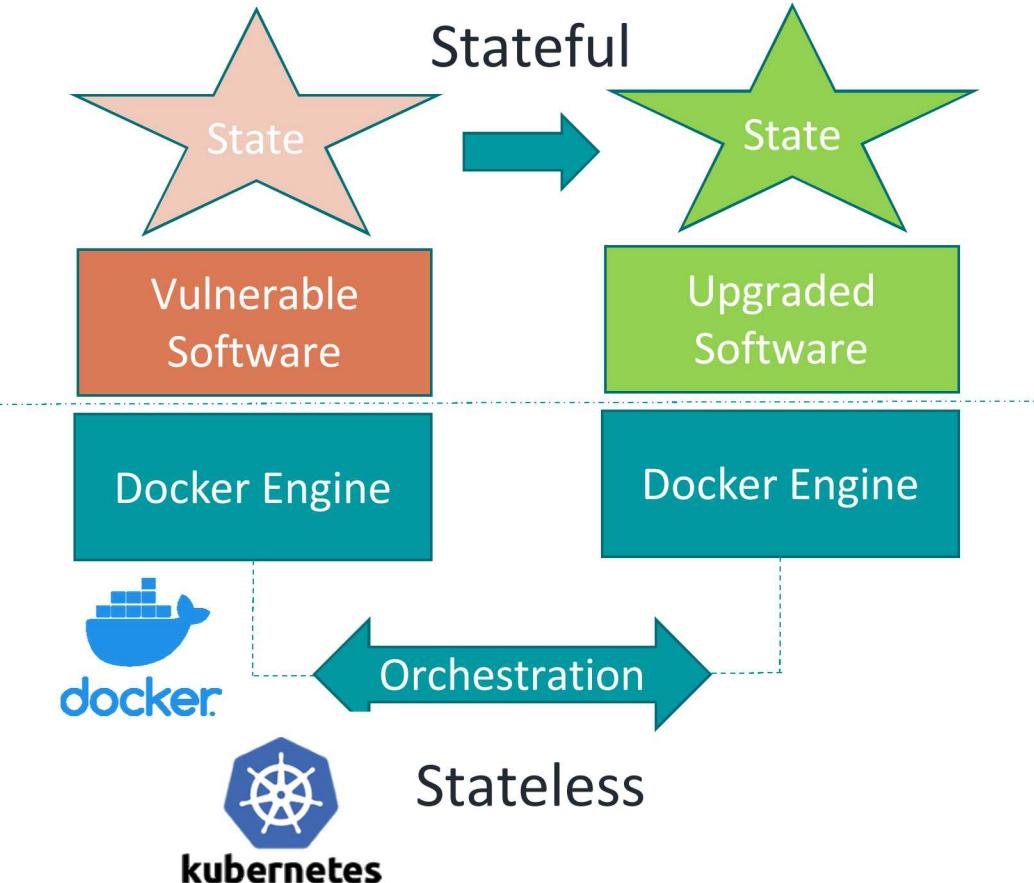


Pilot test CAPSec with OT partners

Bringing CAPSec to Market

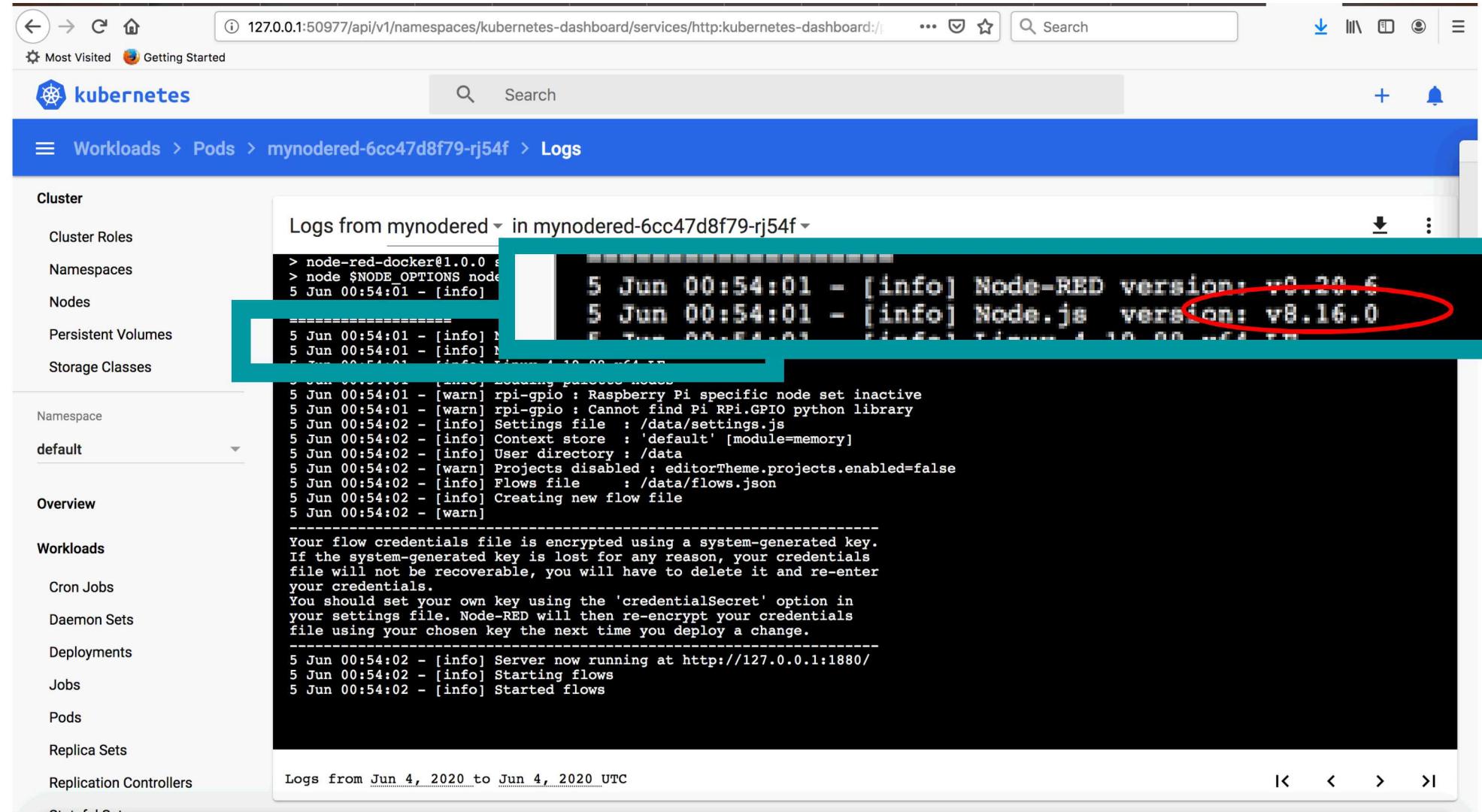
CAPSEC

DOE PACT VIRTUAL SHOWCASE
DEMONSTRATION

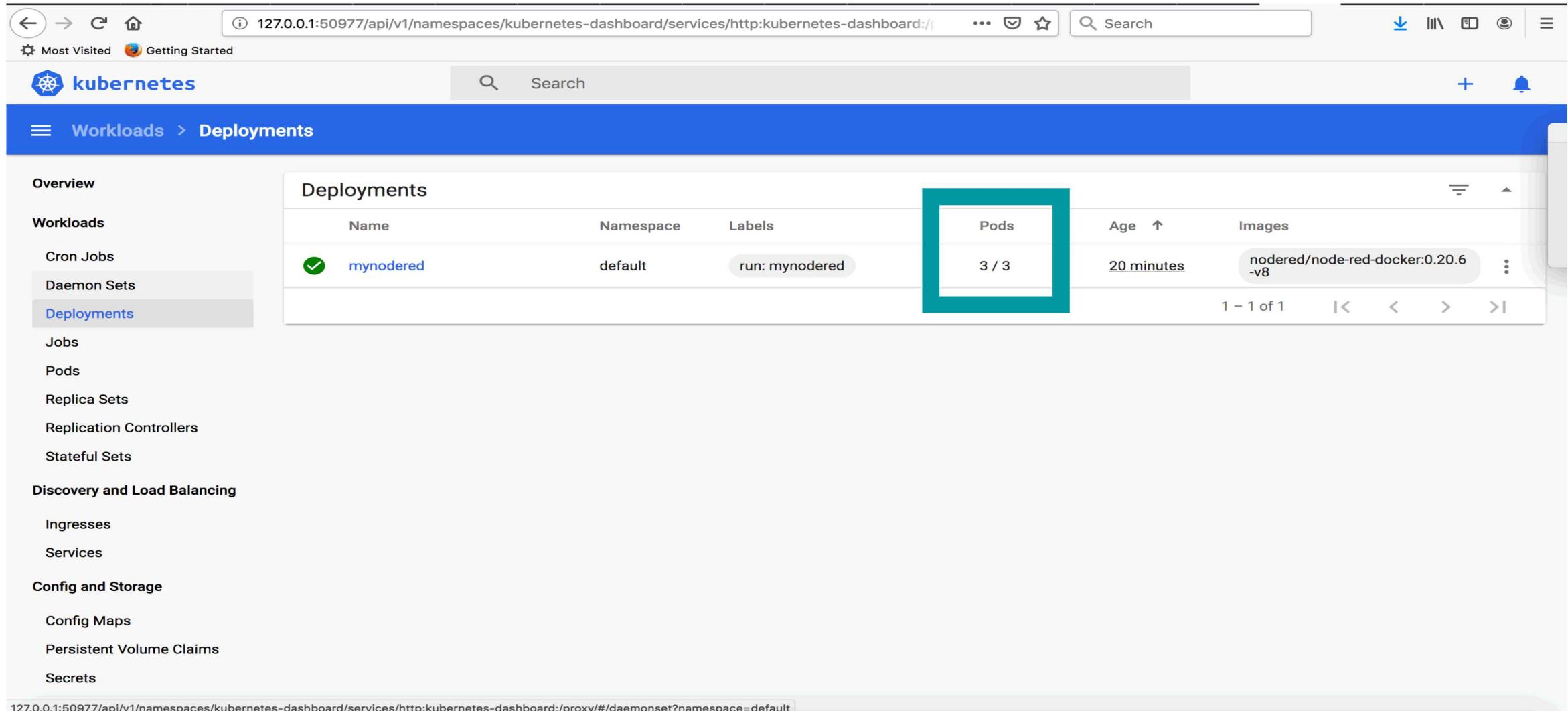


- Containerized architecture
- Replicates Stateful environments
- TLS Agents on parallel containers

NodeRed Software Version BEFORE Upgrade



NodeRed Running Multiple Instances to Upgrade Software without Downtime



The screenshot shows the Kubernetes Dashboard interface. The left sidebar is titled 'kubernetes' and contains a navigation menu with the following items:

- Overview
- Workloads
 - Cron Jobs
 - Daemon Sets
 - Deployments** (selected)
 - Jobs
 - Pods
 - Replica Sets
 - Replication Controllers
 - Stateful Sets
- Discovery and Load Balancing
 - Ingresses
 - Services
- Config and Storage
 - Config Maps
 - Persistent Volume Claims
 - Secrets

The main content area is titled 'Deployments' and shows a table with the following data:

Name	Namespace	Labels	Pods	Age	Images
mynodered	default	run: mynodered	3 / 3	20 minutes	nodered/node-red-docker:0.20.6 -v8

The 'Pods' column is highlighted with a teal border. The URL in the browser's address bar is 127.0.0.1:50977/api/v1/namespaces/kubernetes-dashboard/services/http:kubernetes-dashboard:/proxy/#/daemonset?namespace=default.

Upgrade NodeRed Software One Instance at a Time

```
S1006074:~ adrchav$ kubectl run mynodered --port=1880 --image=nodered/node-red-docker:0.20.6-v8
kubectl run --generator=deployment/apps.v1 is DEPRECATED and will be removed in a future
e version. Use kubectl run --generator=run-pod/v1 or kubectl create instead.
deployment.apps/mynodered created
S1006074:~ adrchav$
```

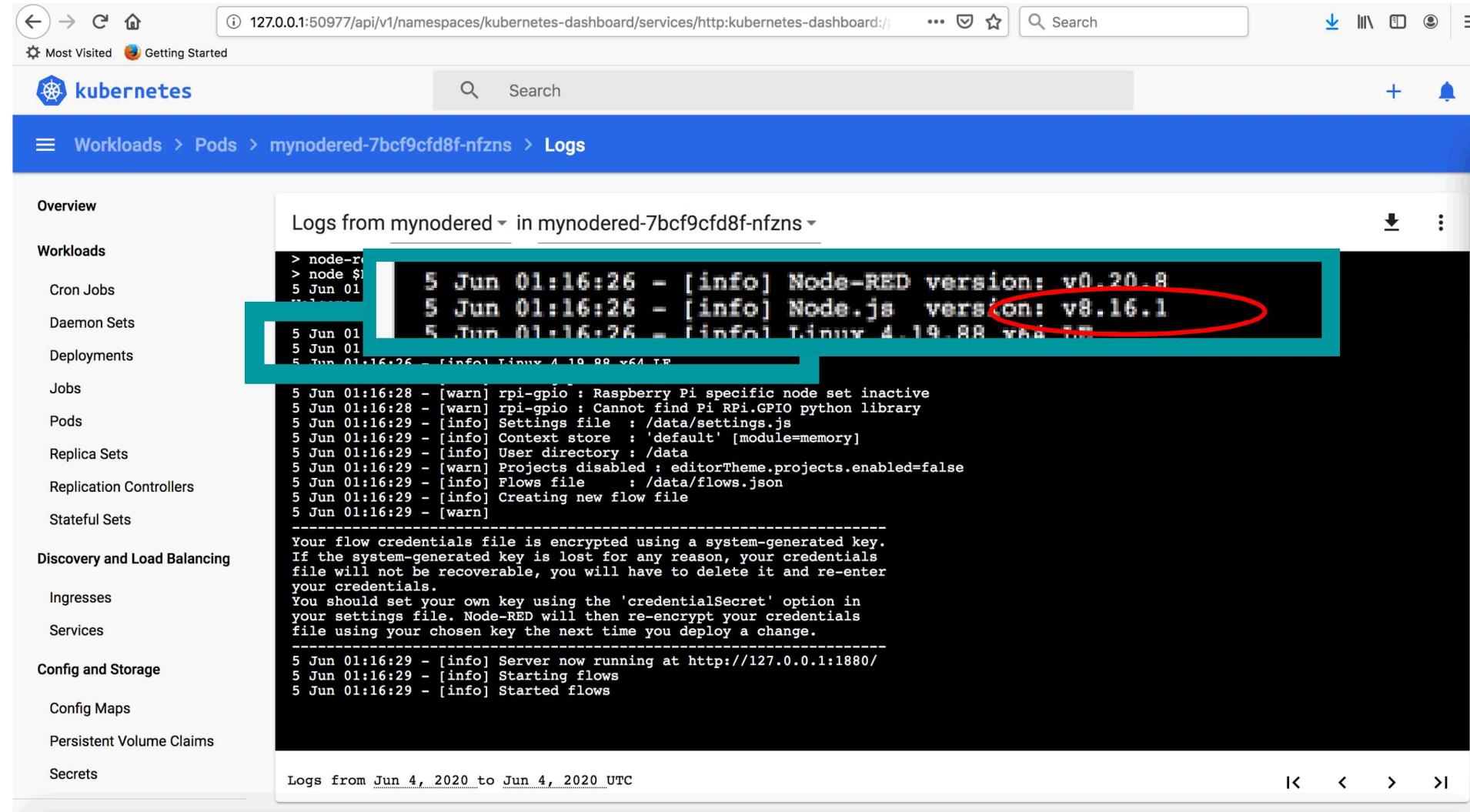


```
S1006074:~ adrchav$ kubectl set image deployment mynodered mynodered=nodered/node-red-docker:latest --record
deployment.apps/mynodered image updated
S1006074:~ adrchav$
```



```
S1006074:~ adrchav$ kubectl set image deployment mynodered mynodered=nodered/node-red-docker:latest --record
deployment.apps/mynodered image updated
S1006074:~ adrchav$ kubectl rollout status deployment mynodered
deployment "mynodered" successfully rolled out
S1006074:~ adrchav$
```

NodeRed Version Patched



Logs from mynodered in mynodered-7bcf9cf8f-nfzns

```
> node-red
> node $ 
5 Jun 01:16:26 - [info] Node-RED version: v0.20.8
5 Jun 01:16:26 - [info] Node.js version: v8.16.1
5 Jun 01:16:26 - [info] Linux 4.19.88-v7.0
5 Jun 01:16:28 - [warn] rpi-gpio : Raspberry Pi specific node set inactive
5 Jun 01:16:28 - [warn] rpi-gpio : Cannot find Pi RPi.GPIO python library
5 Jun 01:16:29 - [info] Settings file : /data/settings.js
5 Jun 01:16:29 - [info] Context store : 'default' [module=memory]
5 Jun 01:16:29 - [info] User directory : /data
5 Jun 01:16:29 - [warn] Projects disabled : editorTheme.projects.enabled=false
5 Jun 01:16:29 - [info] Flows file : /data/flows.json
5 Jun 01:16:29 - [info] Creating new flow file
5 Jun 01:16:29 - [warn]
-----
Your flow credentials file is encrypted using a system-generated key.
If the system-generated key is lost for any reason, your credentials
file will not be recoverable, you will have to delete it and re-enter
your credentials.
You should set your own key using the 'credentialSecret' option in
your settings file. Node-RED will then re-encrypt your credentials
file using your chosen key the next time you deploy a change.
-----
5 Jun 01:16:29 - [info] Server now running at http://127.0.0.1:1880/
5 Jun 01:16:29 - [info] Starting flows
5 Jun 01:16:29 - [info] Started flows
```

Logs from Jun 4, 2020 to Jun 4, 2020 UTC