Overview: Infrastructure as Code to support automation and workload migration Part #2D

Note: Template background, notes, references, identification and other details, and some content, have been removed from this version.

Jeff Barnes

Feb 10, 2018 (v 0.4)

For information only. We acknowledge incomplete attribution after the notes were stripped from his version

Agenda

Definition - Cloud Native

1. Single-stage versus Multi-stage builds in Docker
    - Single-Stage Build (Vulnerability Scan #1)
    - Single-Stage Build (Vulnerability Scan #2)
    - Multi-Stage Builds
    - Monitor and Protect
2. Application Deployment
    - Application Deployment
    - Local deploy using Kubernetes
    - Kubernetes on other platforms
    - Containers on Public Clouds

Questions/Additional Slides

Cloud Native

A cloud-native application is a distributed, elastic and horizontal scalable system composed of (micro)services which isolates state in a minimum of stateful components. The application and each self-contained deployment unit of that application is designed according to cloud-focused design patterns and operated on a self-service elastic platform.

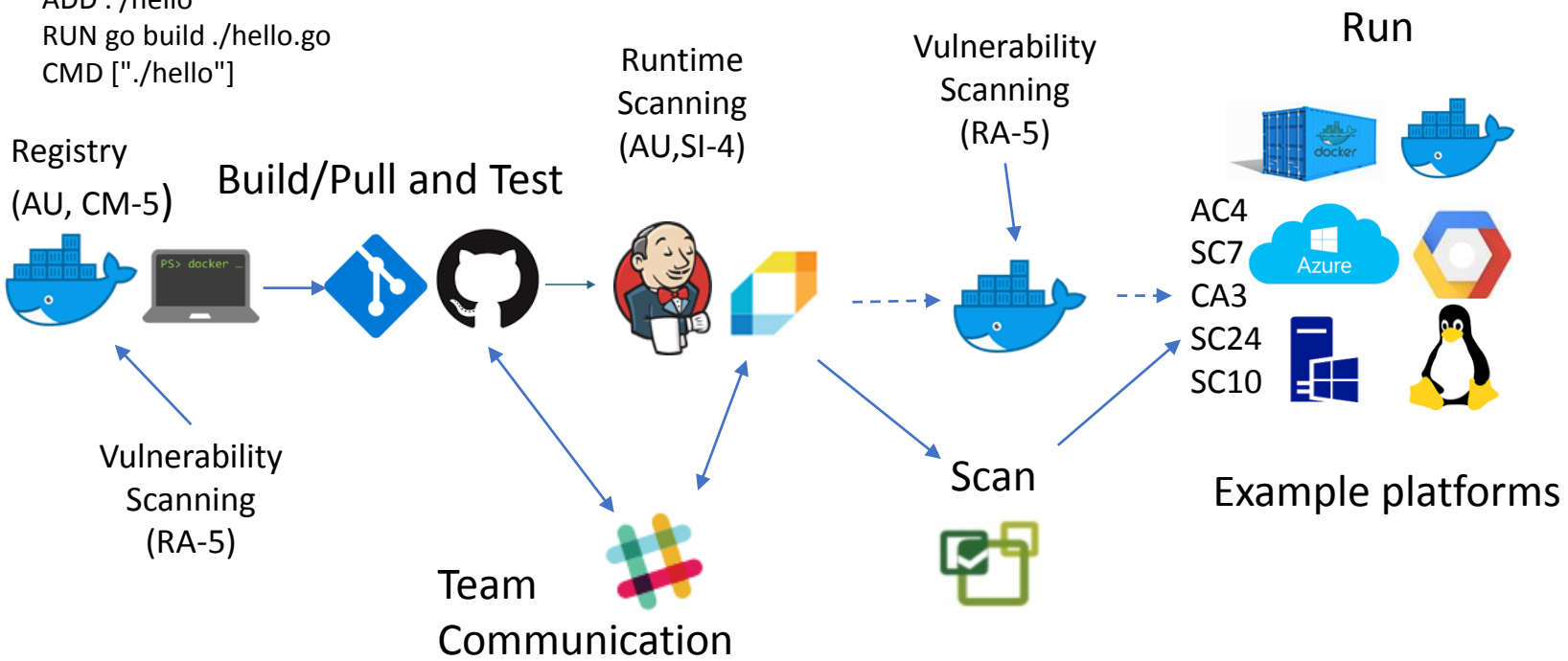
Even though the specific tools and patterns may differ, cloud-native organizations and applications follow a fairly consistent pattern

Kratzke, N., Quint, P.C.: Understanding Cloud-native Applications after 10 Years of Cloud Computing - A Systematic Mapping Study. Journal of Systems and Soft-ware 126(April), 1–16 (2017)

Single-Stage Build

Dockerfile
FROM golang:alpine
RUN mkdir -p /hello
WORKDIR /hello
ADD . /hello
RUN go build ./hello.go
CMD ["./hello"]

Docker Single-stage build, includes CI/D tools Git, GitHub, Slack, Jenkins, Docker Hub and deployment stack

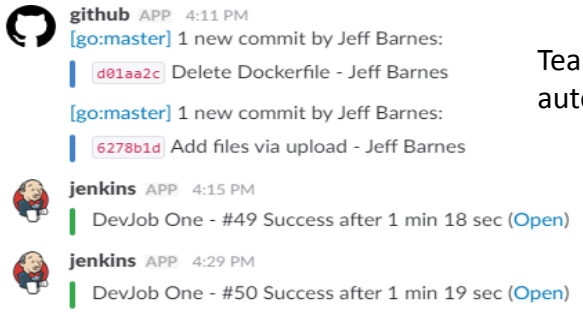


portainer.io Container Management

SP800-53 controls taken from: Love and agony: containers in government https://speakerdeck.com/aidanfeldman/love-and-agony-containers-in-government

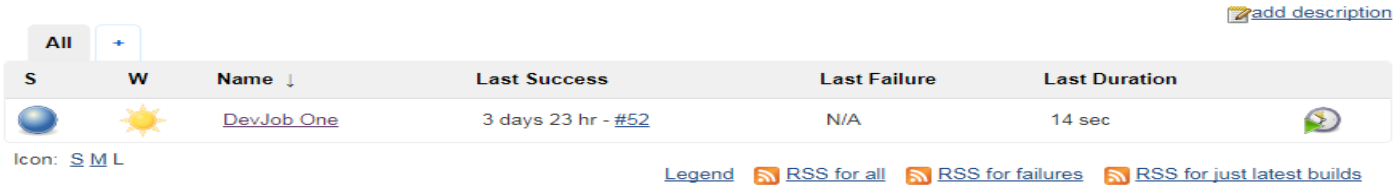
Single-Stage Build

\GitHub\go [master = +0 ~2 -0 !]> git add hello.go
\GitHub\go [master = +0 ~1 -0 | +0 ~1 -0 !]> git commit



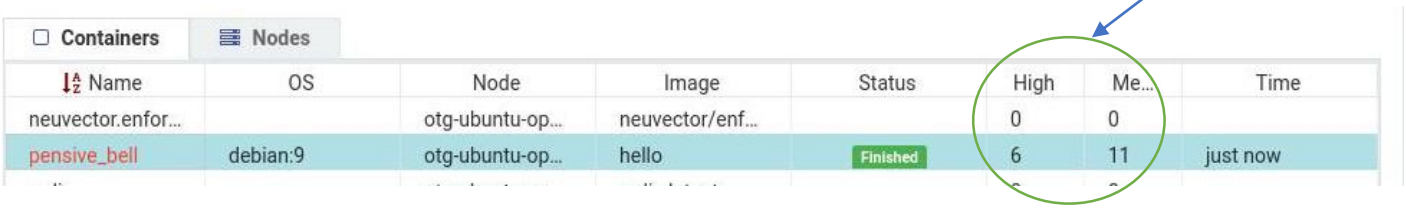
Team update sent to Slack Chanel. Can also do versioning control and add a Webhook to automatically trigger Jenkins build on Github update.

Build runs in Jenkins, can include QA, testing, Blue/Green deployment, etc. In this example, compiles GO binary, builds and runs in Docker containers

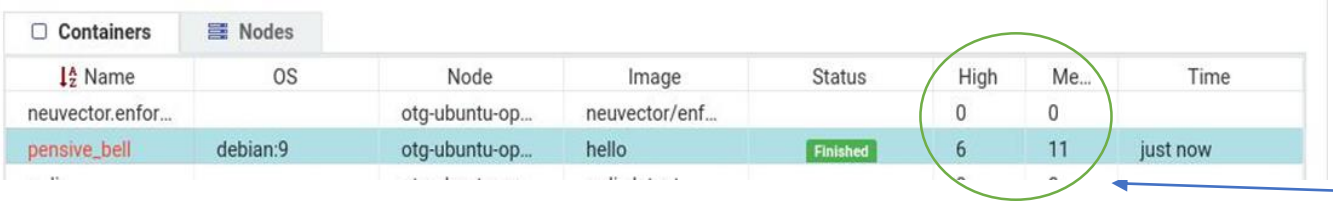


Manually verify/validate images using Neuvector against CIS database and other controls. Can also be automated in the build pipeline

Contains CIS vulnerabilities! Automated verification would fail the build in Jenkins



Vulnerability Scan #1



Single-stage build includes libraries and OS, which contain vulnerabilities!

Name	Urgency	Package	Version
CVE-2017-8804	High	glibc	2.24-11+deb9u1
CVE-2017-8831	High	linux	4.9.51-1
CVE-2013-7445	High	linux	4.9.51-1
CVE-2017-7487	High	linux	4.9.51-1
CVE-2017-8890	High	linux	4.9.51-1
CVE-2016-2779	High	util-linux	2.29.2-1
CVE-2017-9038	Medium	binutils	2.28-5
CVE-2017-9043	Medium	binutils	2.28-5
CVE-2017-9040	Medium	binutils	2.28-5
CVE-2017-9042	Medium	binutils	2.28-5

Single-stage Dockerfile

```
FROM golang
RUN mkdir -p /hello
WORKDIR /hello
ADD . /hello
RUN go build ./hello.go
CMD ["./hello"]
```

Post build scan using Neuvector Enforcer Container Security Broker
Can also run docker-bench

Plus the build size is 735 MB for a Hello World program

```
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
hello               v2                  aa9b89ac96cf        2 minutes ago       735MB
hello               v0                  17cbe44ed49a        10 minutes ago      5.86MB
```

Vulnerability Scan #2 (Same code)

Single-stage build includes libraries and OS, which contain vulnerabilities! Scan occurs during build, can log vulnerabilities or fail build

Single-stage Dockerfile

```
FROM golang
RUN mkdir -p /hello
WORKDIR /hello
ADD . /hello
RUN go build ./hello.go
CMD ["./hello"]
```

Plus the build size is 735 MB for a Hello World program

Vulnerability Scan during build with Aqua Container Security Broker, log or fail build on vulnerabilities

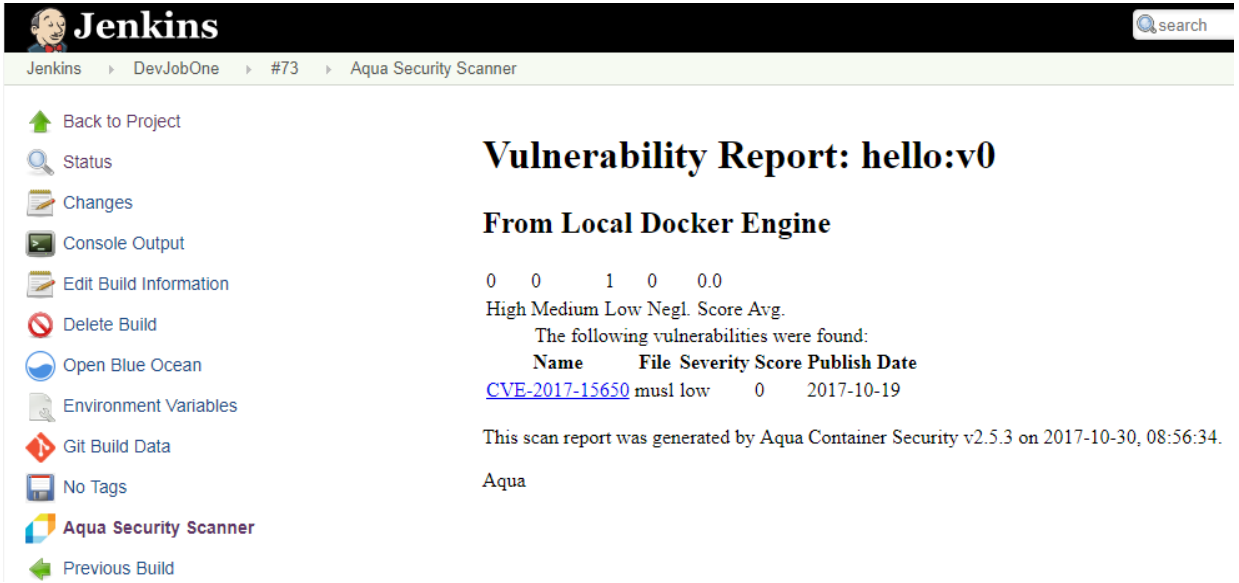
```
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
hello               v2                  aa9b89ac96cf        2 minutes ago       735MB
hello               v0                  17cbe44ed49a        10 minutes ago      5.86MB
```

Multi-Stage build (Same code)

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
hello	v2	aa9b89ac96cf	2 minutes ago	735MB
hello	v0	17cbe44ed49a	10 minutes ago	5.86MB

Multi-stage build contains fewer libraries and OS, minimal vulnerabilities

Program contains minimal vulnerability (inherently more secure) plus only 5.8 MB (versus 735 MB) in size



Multi-Stage Dockerfile

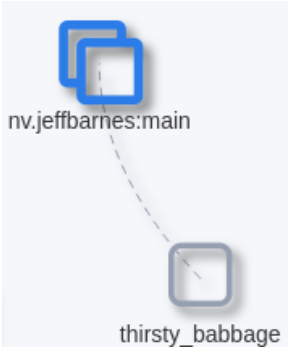
```
FROM golang AS build-env
RUN mkdir -p /app
WORKDIR /app
ADD . /app
RUN go build ./hello.go

# final stage
FROM alpine
WORKDIR /app
COPY --from=build-env /app /app
RUN cd /app
CMD ./hello
```

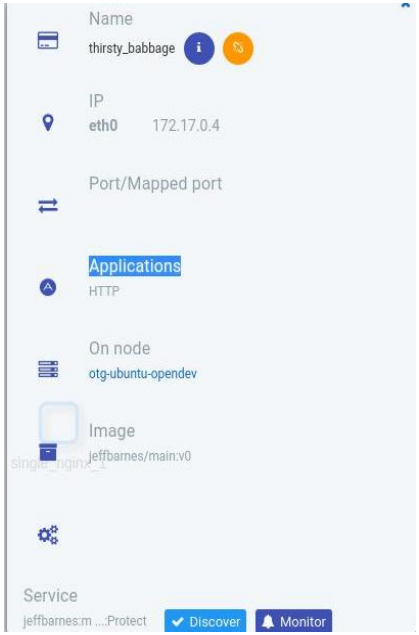
Monitor and Protect Code once in Containers

Container Security Broker scans, monitors and can protect running containers on host/platform

Application	Client	Server	Client bytes	Server byt...	Rule	Action	Duration
		850(	140 bytes	74 bytes	0	Open	00:00:14
		850(	140 bytes	74 bytes	0	Open	00:00:14
		850(	140 bytes	74 bytes	0	Open	00:00:14
HTTP		850(	3.3 KB	1.3 KB	0	Open	00:00:14

Similar services available from CSP, both managed and DIY





Application maps, container security

Neuvector

#2 Application Deployment

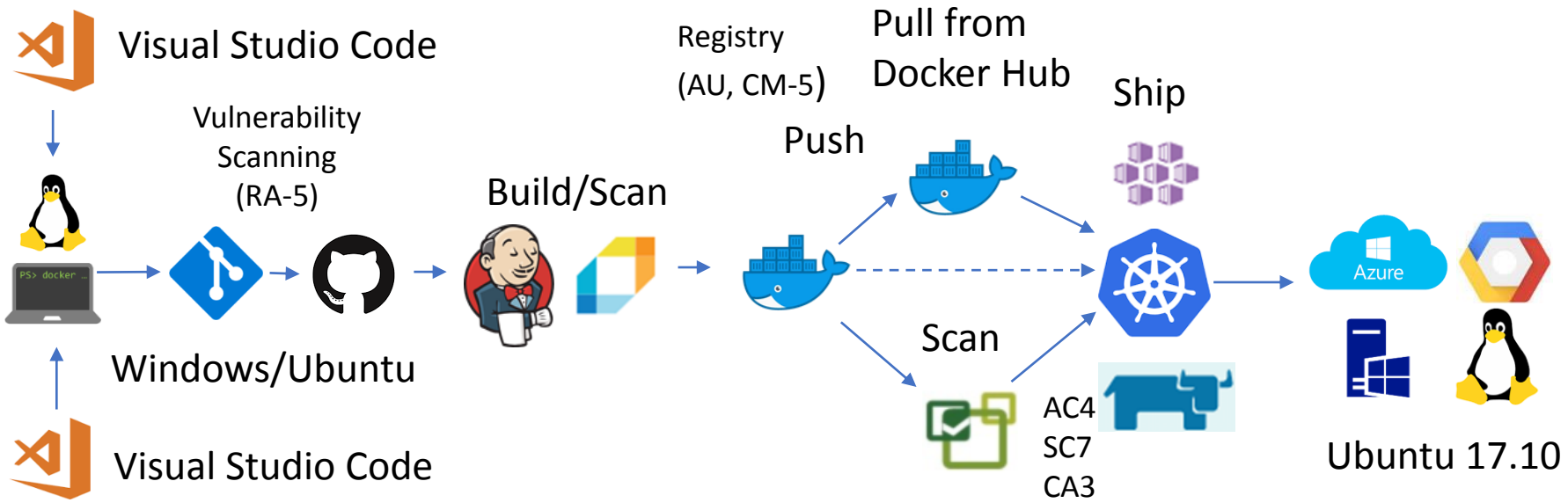


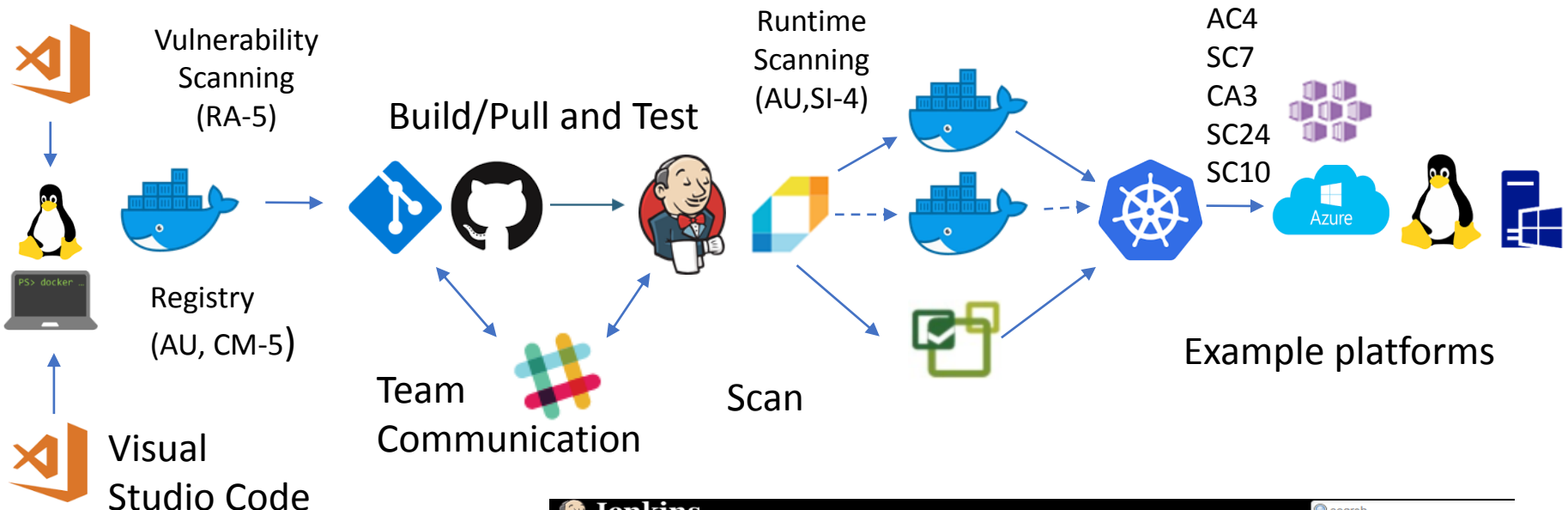
Image is only 10 MB

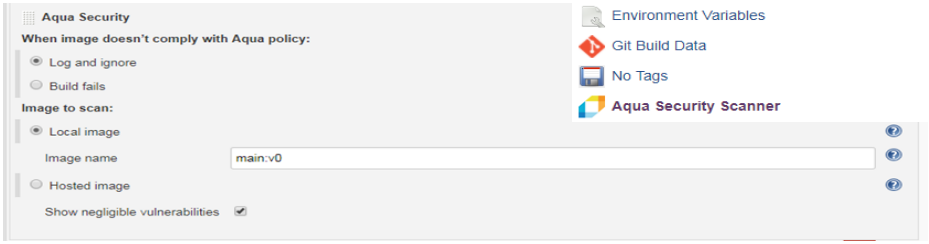
```
>docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
main                v0                  XXXXXXXafe64        50 seconds ago      10.4MB
```

Note: example dev build deploys Kubernetes cluster on a local host. Automate testing and deployment to public cloud using the same patterns. Deploy through CI/CD, such as Jenkins, to cloud service providers or deploy over multiple clouds

Multi-Stage build

Build/Pull and Test





AquaSec

Local deploy using Kubernetes

```
kubectl run go-main --image=jeffbarnes/main:v0 --port=8500
kubectl expose deployment go-main --type="LoadBalancer"
```

```
minikube service go-main --url
http://192.168.99.100:30079
```

Application/service deployed and available

Scale deployment to 5 manually

```
kubectl scale deployment go-main --replicas=5
deployment "go-main" scaled
```

go-main-1774950723-8drth	minikube	Running	0	7 minutes
go-main-1774950723-m5094	minikube	Running	0	7 minutes
go-main-1774950723-pt19n	minikube	Running	0	7 minutes
go-main-1774950723-wdlkk	minikube	Running	0	7 minutes
go-main-1774950723-g5mlx	minikube	Running	0	16 minutes

Minikube

Local deploy to Kubernetes

via YAML

```
kubectl apply -f deploy.yaml
```

deploy.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: main
spec:
 selector:
    matchLabels:
      app: main
 replicas: 3
 template:
    metadata:
     labels:
      app: main
    spec:
     containers:
     - name: main
       image: jeffbarnes/main:v0
       ports:
        - containerPort: 8500
       securityContext:
         readOnlyRootFilesystem: true
         allowPrivilegeEscalation: false
---
apiVersion: v1
kind: Service
metadata:
  name: main
  labels:
    app: main
spec:
  type: LoadBalancer
  selector:
    app: main
  ports:
  - protocol: TCP
    port: 8500
    targetPort: 8500
    name: http
```

via Helm

```
helm create main
$ tree main

main
├── charts
├── Chart.yaml
├── templates
│   ├── deployment.yaml
│   ├── _helpers.tpl
│   ├── ingress.yaml
│   ├── NOTES.txt
│   └── service.yaml
└── values.yaml


$ helm install --name main /home/(user)/main
```

Local deploy using Kubernetes

Now let's delete a pod and see what happens

`kubectl delete pods go-main-1774950723-m5094`

Kubernetes automatically re-adds it, can also provision by resources

Replicas restored

Local deploy using Kubernetes

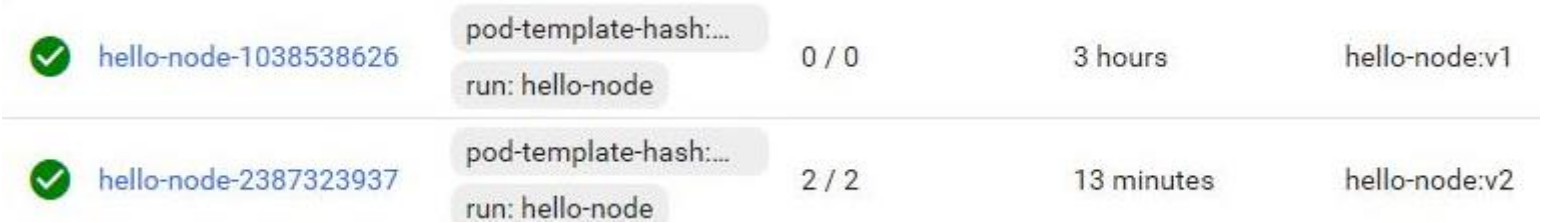
Pods – scale deployment manually or automatically. Scale from 5 to 2

```
kubectl scale deployment go-main --replicas=2
```

Replica Sets

Upgrading the Deployment: various ways to do Rolling Updates/Rollouts

```
kubectl set image deployment/hello-node hello-node=hello-node:v2
```




```
kubectl rolling-update hello-node --image=hello-node:v2
kubectl rolling-update hello-node --rollback
```

Containers on Public Clouds

Deploy image to Azure Container Instance via Jenkins

15 seconds from a build in Jenkins to publically accessible application in Azure

Azure Container Service

```
az acs create --name XXX --dns-prefix XXX --resource-group XXX --orchestrator-type kubernetes --generate-ssh-keys \
```

Or any number of other private, public or hybrid cloud services: GKE, AKS, EKS, IBM Cloud

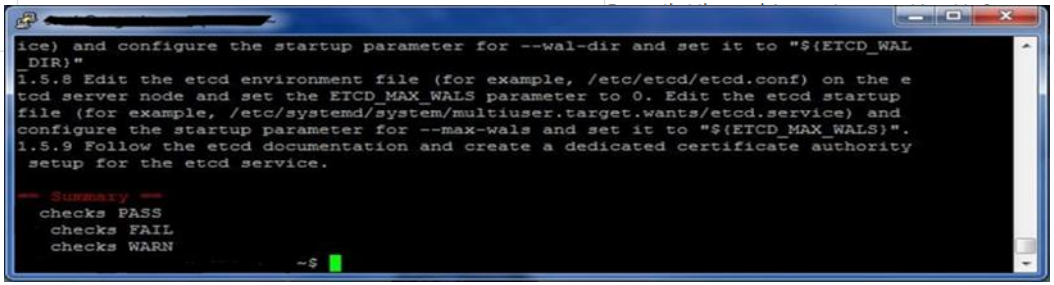
Kubernetes Controls Matrix (WIP)

Mapping NIST SP800 53 controls to Kubernetes via the CIS Benchmark
Automate checking via Kube-bench or CASB

	A	B	C	D
1	NIST 800-53 rev4	Critical Security Control	Kubernetes Master (APIServer)	Kubernetes Worker (Kublet)
	CA-7: Continuous Monitoring SC-39: Process Isolation SC-44: Detonation Chambers SI-3: Malicious Code Protection SI-4: Information System Monitoring	Critical Security Control #8: Malware Defenses		
9	SI-8: Spam Protection		Ensure that the admission control policy is set to EventRateLimit	
	AC-4: Information Flow Enforcement CA-7: Continuous Monitoring CA-9: Internal System Connections CM-2: Baseline Configuration CM-6: Configuration Settings CM-8: Information System Component Inventory SC-20: Secure Name /Address Resolution Service (Authoritative Source) SC-21: Secure Name /Address Resolution Service (Recursive or Caching Resolver) SC-22: Architecture and Provisioning for Name/Address Resolution Service SC-41: Port and I/O Device Access	Critical Security Control #9: Limitation and Control of Network Ports	Ensure --insecure-bind-address argument is not set Ensure --insecure-port argument is set to 0 Ensure --authorization-mode argument is not set to AlwaysAllow Ensure that the --authorization-mode argument is set to Node Ensure that the --etcd-certfile and --etcd-keyfile arguments are set as appropriate Ensure that the --max-wals argument is set to 0 Ensure that a unique Certificate Authority is used for etcd	Ensure that the --read-only-port argument is set to 0 Ensure that the --streaming-connection-idle-timeout argument is not set to 0 Ensure that the --make-iptables-util-chains argument is set to true
10	SI-4: Information System Monitoring			

```
$./kube-bench master
```

Run ./kube-bench against master or nodes to validate Kubernetes install against CIS Benchmark



https://github.com/aquasecurity/kube-bench

Basic Hygiene

- Use a non-root user inside containers

Dockerfile

```
RUN groupadd -r nodejs
RUN useradd -m -r -g nodejs nodejs
USER nodejs
```

YAML File

```
securityContext:
 runAsNonRoot: true
```

- Make the filesystem read-only

YAML File

```
securityContext:
 readOnlyRootFilesystem: true
```

- Use the “record” option for easier rollbacks
- Use Labels, Tags, (Not LATEST)
- RBAC, ABAC, Namespace, Sidecars