



# Kubernetes Community Days

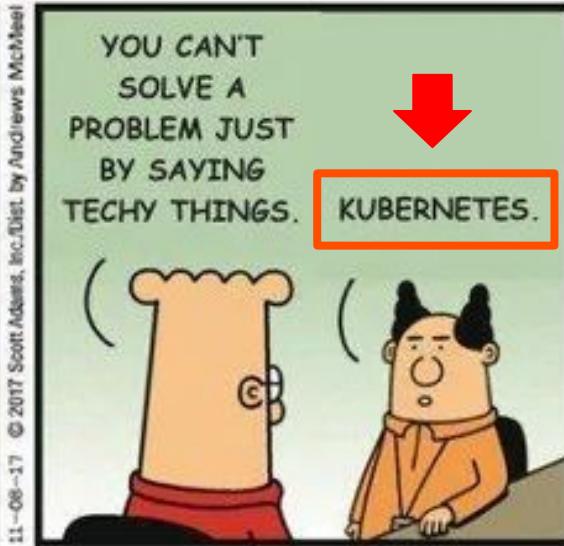


# Top 10 Metrics to Observe in Kubernetes (including stuffs around K8s)

**Steve Ng**  
Developer Relations Lead  
*New Relic APJ*

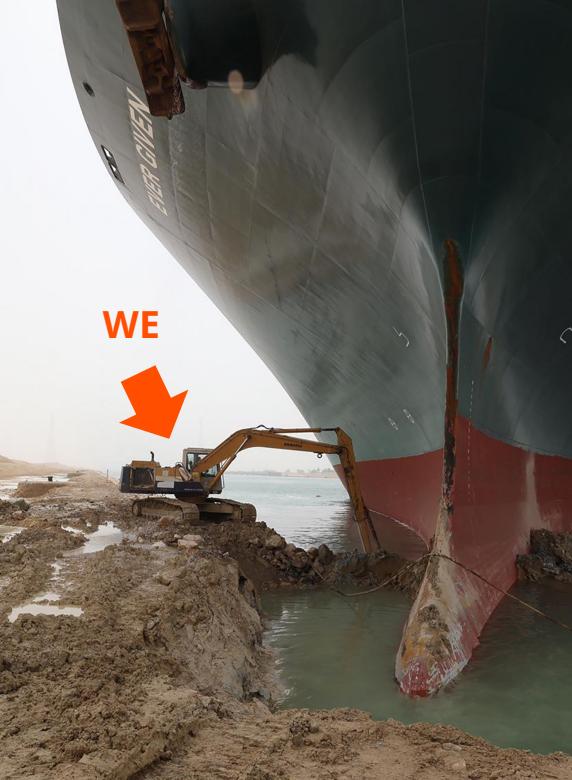


# Let's Talk Kubernetes



Kubernetes is awesome, and scary.

# Sounds Familiar?



# Where to get Telemetry ...



cAdvisor

Metric Server

API Server

Node Exporter

Kube State Metrics



**Observe Symptoms, not Problems**

Opinion - if only I can pick 10 ~

# Problem - Misconfiguration Oops!



## Why should you care?

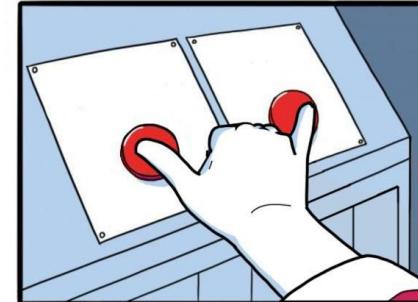
No magic solution to fix fat fingers, incorrect characters, missing requirements, invalid inputs, etc ...

**K8s** - A combination of source control + deployment pipeline with **smoke tests** for each stage (Launchdarkly).

Leverage existing tools such as Infrastructure as Code, and Configuration Management for config consistency.

**Tradeoff** - increased cost (environments to test) & good Git practice, but still better compared to Production outages.

Possible solution in 2022 - **GitOps with CD** aka CNCF ArgoCD.



Misconfigurations

# Problem - Misconfiguration Oops!



## Where to find them?

Status	Pipeline	Triggerer	Commit	Stages
running	#146411330		l'31649 -> dacc7ea3 Merge branch 'nicolasdular/sto...	
failed	#146410995		l'32306 -> 9a5d2aa1 Merge branch '12-10-stable-e...	
passed	#146410705		l'31801 -> 42738af2 Merge branch '210018-remove...	
passed	#146410223		l'master -> d635c709 Merge branch '22691-externall...	

**Important > Pipeline Status** (via [GitLab](#))  
[Analytics](#) coming based on DORA DevOps Research

Applications / guestbook

Synced ✓ To latest (d635c709) Authored by Alex Collins <alexei@users.noreply.github.com> Updated by Alex Collins <alexei@users.noreply.github.com> Update examples to better reflect hook usage today (4+)

guestbook-ui (service)

guestbook (application)

guestbook-ui (rev1 deployment)

guestbook-ui-65b878495d-wlxp2 (running 1/1 pod)

Uptime 1 day Applications Deployed 1094

Application Health: Healthy Progressing Suspended Degraded Missing Unknown

Application Sync Status: Synced OutofSync Unknown

Controller Stats: Beta Activity

Notification Activity

[ArgoCD](#) is promising with GitOps Guides for [Operators](#), and [Metrics](#) available

# Problem - I need MOORRREEEE!



## Why should you care?

Every system has a **DEFINITE** limit!

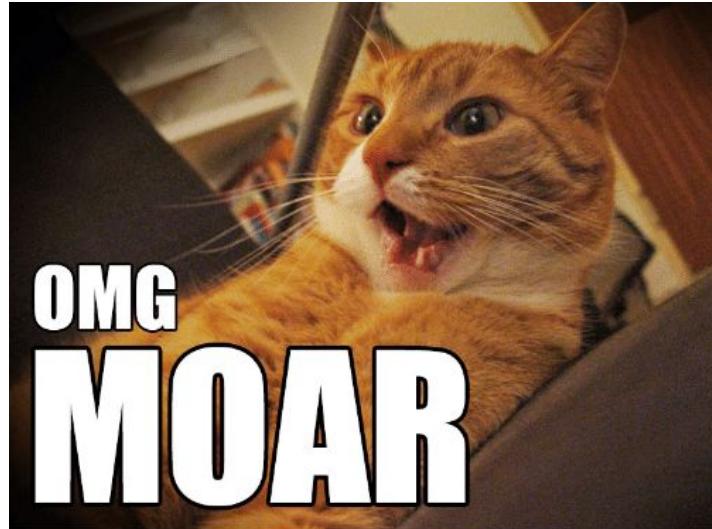
CPU limits - Throttling, Memory limits - OOMKilled.

The Debate of the Century - set quota or no quota!

Even Google recommends to set requests & limits!

Capacity planning is a lost art! How we plan,

- 30% free capacity - autoscale if needed.
- N+2 failover for services.
- Trend analysis - weekly, monthly & quarterly.
- Histogram heatmaps - anticipate spike.
- Forecast for features testing, and rollouts.

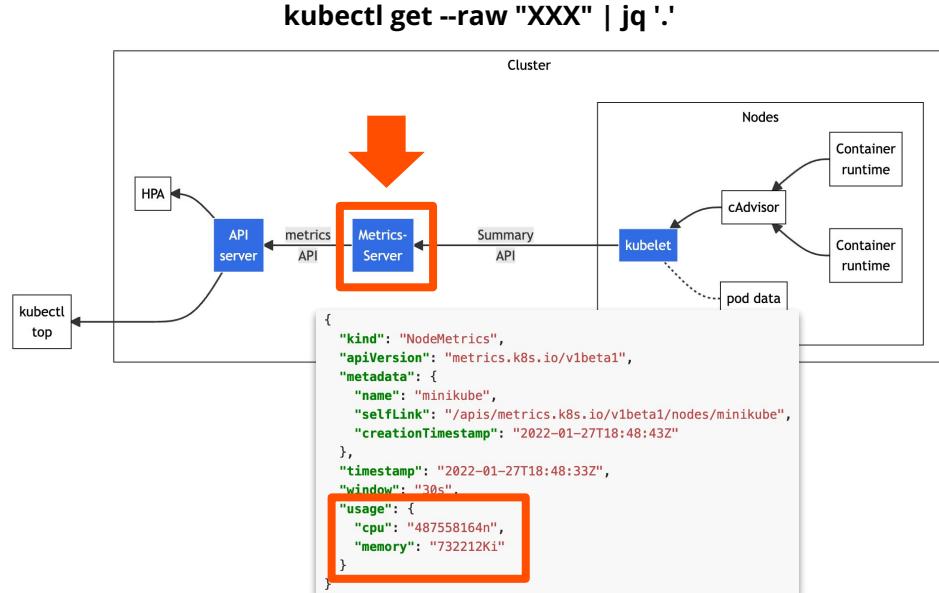


Give me MOAR, MOAR, MOAR!

# Problem - I need MOORRREEEE!



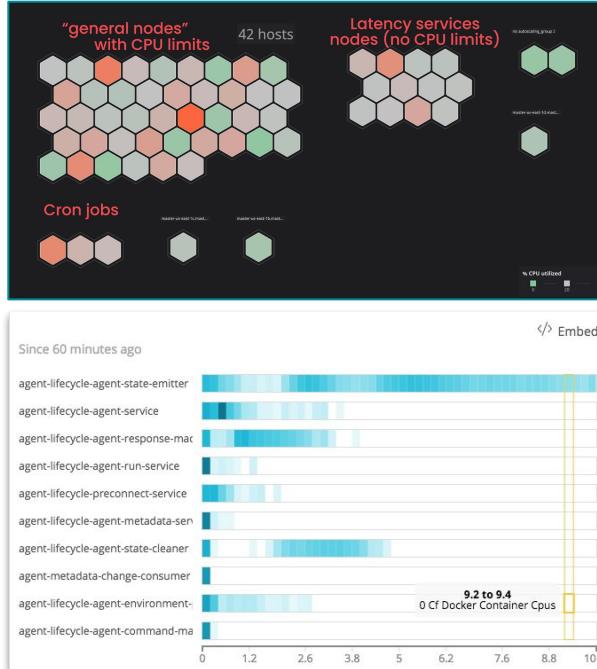
## Where to find them?



**Important > CPU, Important > Memory**

Metric Pipelines are important for Horizontal & Vertical Scaling

## Idea - Isolating “No CPU Limits” services



# Capacity Planning Made EASY

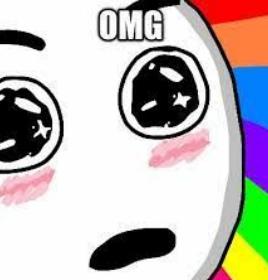


### KUBERNETES INSTANCE CALCULATOR

m5.4xlarge

AWS GCP Azure

NAME	MEM	CPU	EFFICIENCY	PODS
a1.2xlarge AWS	16GiB	8	82.98%	1.91\$
a1.4xlarge AWS	32GiB	16	57.27%	2.67\$
a1.large AWS	4GiB	2	95.21%	2.16\$
a1.medium AWS	2GiB	1	68.35%	6.12\$
a1.metal AWS	32GiB	16	57.27%	2.67\$
a1.xlarge AWS	8GiB	4	85.85%	1.99\$
A2 v2 AZURE	4GiB	2	92.55%	3.85\$
	16GiB	2	58.61%	5.04\$
	7GiB	4	94.60%	4.80\$
	14GiB	8	95.01%	4.55\$



POD REQUESTS

MEMORY CPU

128MB 100m

POD LIMITS

MEMORY CPU

128MB 100m

DAEMONSETS & AGENTS

MEMORY CPU

128MB 100m

OVERCOMMITMENT

66% 100%

EFFICIENCY %

66 100

MAX POD COUNT

77 110

Instance cost \$0.20 \$4.90 \$146.88

Cost per pod >\$0.01 \$0.06 \$1.91

Kubelet cost \$0.01 \$0.36 \$10.78

Unused costs \$0.03 \$0.74 \$22.27

HOUR DAY MONTH

a1.2xlarge

CPU MEM

100MiB or 0.6% Memory reserved to the eviction threshold

14.94GB or 89.1% Memory available to Pods

128MB or 0.8% Memory reserved for DaemonSets and agents

1.43GiB or 8.9% Memory reserved to the kubelet

100MiB or 0.6% Memory reserved to the operating system

SHARE CONFIGURATION

<https://learnk8s.io/kubernetes-instance-calculator>

# Problem - Network Headaches



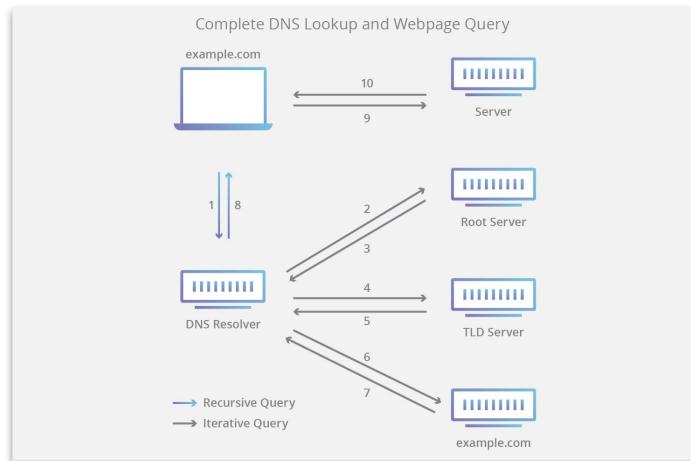
## Why should you care?

Troubleshooting network issue is **NO FUN AT ALL** :(

K8s v1.12 - **CoreDNS** is the recommended DNS Server, replacing **kube-dns**. K8s DNS schedules a DNS Pod and Service - tell containers to resolve DNS names.

Most common problems with DNS - **resolution errors**, or **network connectivity issues**. Will see **timeout** or **errors** with these problems.

Common problems with K8s Ingress, but remediation options are improving with providers.



What is a DNS Server?  
[CloudFlare](#)

# Problem - Network Headaches



## Where to find them? - DNS

The Corefile configuration includes the following [plugins](#) of CoreDNS:

- **errors**: Errors are logged to stdout.
- **health**: Health of CoreDNS is reported to `http://localhost:8080/health`. In this extended syntax `lameduck` will make the process unhealthy then wait for 5 seconds before the process is shut down.
- **ready**: An HTTP endpoint on port 8181 will return 200 OK, when all plugins that are able to signal readiness have done so.
- **kubernetes**: CoreDNS will reply to DNS queries based on IP of the services and pods of Kubernetes. You can find [more details](#) about that plugin on the CoreDNS website. `ttl` allows you to set a custom TTL for responses. The default is 5 seconds. The minimum TTL allowed is 0 seconds, and the maximum is capped at 3600 seconds. Setting TTL to 0 will prevent records from being cached. The `pods insecure` option is provided for backward compatibility with `kube-dns`. You can use the `pods verified` option, which returns an A record only if there exists a pod in same namespace with matching IP. The `pods disabled` option can be used if you don't use pod records.
- **prometheus**: Metrics of CoreDNS are available at `http://localhost:9153/metrics` in [Prometheus](#) format (also known as OpenMetrics).
- **forward**: Any queries that are not within the cluster domain of Kubernetes will be forwarded to predefined resolvers (`/etc/resolv.conf`).
- **cache**: This enables a frontend cache.
- **loop**: Detects simple forwarding loops and halts the CoreDNS process if a loop is found.
- **reload**: Allows automatic reload of a changed Corefile. After you edit the ConfigMap configuration, allow two minutes for changes to take effect.
- **loadbalance**: This is a round-robin DNS loadbalancer that randomizes the order of A, AAAA, and MX records in the response.

2 errors like '^read udp .\* i/o timeout\$' occurred in last 30s

- `coredns_health_request_duration_seconds{}` - duration to process a HTTP query to the local `/health` endpoint. As this is a local operation it should be fast. A (large) increase in this duration indicates the CoreDNS process is having trouble keeping up with its query load.
- `coredns_health_request_failures_total{}` - The number of times the internal health check loop failed to query `/health`.

### Important > [Health](#) (via CoreDNS)

### [Verify](#) CoreDNS is running well

```
.:53
2018/08/15 14:37:17 [INFO] CoreDNS-1.2.2
2018/08/15 14:37:17 [INFO] linux/amd64, go1.10.3, 2e322f6
CoreDNS-1.2.2
linux/amd64, go1.10.3, 2e322f6
2018/08/15 14:37:17 [INFO] plugin/reload: Running configuration MD5 = 24e6c59e83ce706f07bcc82c31b1ea1c
```

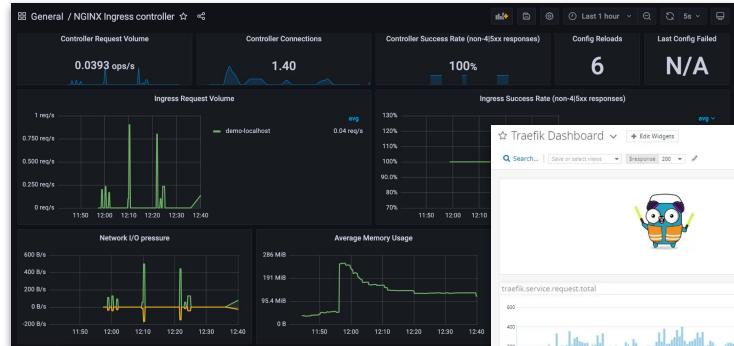
### [Kubernetes DNS](#) (via CoreDNS)

```
kubectl logs --namespace=kube-system -l k8s-app=kube-dns
```

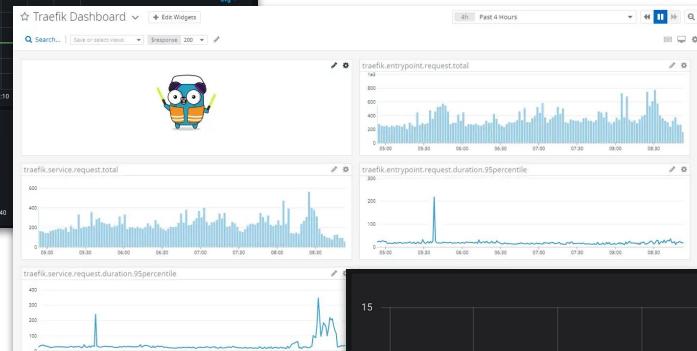
# Problem - Network Headaches



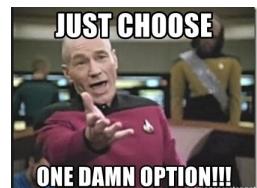
## Where to find them? - Ingress



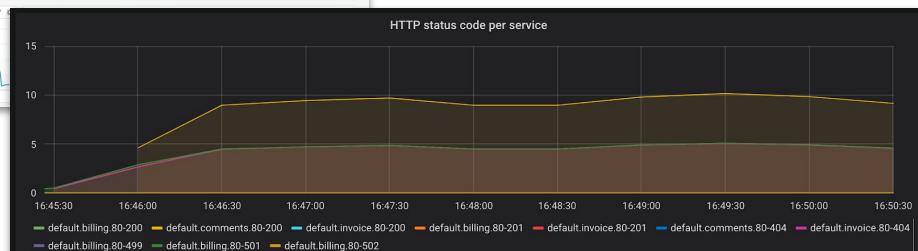
NGINX



Traefik



If I can only pick ONE,  
**Important > Error Rate (providers)**



Kong

# Problem - Help! My Pods are stuck!



## Why should you care?

Pretty common to see pods get stuck!

Issues such as insufficient resources, reached limits, incorrect image, misconfiguration, unable to resolve names, etc ... where should I start?

kubectl [describe](#) pod X is best to check what's going on.

K8s is getting better in errors, when things go wrong.  
Consider customizing your [termination message](#).

**Tip** - Only push **Errors** logs into a centralized service!



Pods need help too

# Problem - Help! My Pods are stuck!



## Where to find them?

Set up observers in **Kube State Metrics** - [pod-metric](#).

**Important** > `kube_pod_status_phase` - <Pending|Running|Succeeded|Failed|Unknown>

**Important** > `kube_pod_status_reason` - <Evicted|NodeAffinity|NodeLost|Shutdown|UnexpectedAdmissionError>

Events:						
FirstSeen	LastSeen	Count	From	SubobjectPath	Type	Reason
1m	48s	7	{default-scheduler }		Warning	FailedScheduling
				fit failure on node (kubernetes-node-6ta5):	Node didn't have enough resource: CPU, requested: 1000, used: 1420, capacity: 2000	
				fit failure on node (kubernetes-node-wul5):	Node didn't have enough resource: CPU, requested: 1000, used: 1100, capacity: 2000	

Conditions:						
Type	Status	LastHeartbeatTime	LastTransitionTime	Reason	Message	
NetworkUnavailable	False	Thu, 17 Feb 2022 17:09:13 -0500	Thu, 17 Feb 2022 17:09:13 -0500	WeaveIsUp	Weave pod has set this	
MemoryPressure	Unknown	Thu, 17 Feb 2022 17:12:40 -0500	Thu, 17 Feb 2022 17:13:52 -0500	NodeStatusUnknown	Kubelet stopped posting node status.	
DiskPressure	Unknown	Thu, 17 Feb 2022 17:12:40 -0500	Thu, 17 Feb 2022 17:13:52 -0500	NodeStatusUnknown	Kubelet stopped posting node status.	
PIDPressure	Unknown	Thu, 17 Feb 2022 17:12:40 -0500	Thu, 17 Feb 2022 17:13:52 -0500	NodeStatusUnknown	Kubelet stopped posting node status.	
Ready	Unknown	Thu, 17 Feb 2022 17:12:40 -0500	Thu, 17 Feb 2022 17:13:52 -0500	NodeStatusUnknown	Kubelet stopped posting node status.	

`kubectl describe pod XXX`

\* Need help? Great links here > Introspection [Apps](#), Troubleshoot [Apps](#), & Debug Running [Pods](#)

# Problem - Control Plane



## Why should you care?

**API server** - api for all things K8s. Sharing states, and communicate with other K8s components.

API servers went through many iterations of stability. Most problems now related to **inter-network** issues.

**etcd** as a key-value database - stores the configuration, actual state vs desired state of the system.

**Tip** - Don't forget [storage](#), and [backups](#) for etcd!

Good guide available by Kubernetes - [HERE](#).



K8s Control Plane aka Mission Control

# Problem - Control Plane



## Where to find them? - API Server & etcd

**Note** > API Server has 3 API endpoints (healthz, livez and readyz).

The healthz endpoint is deprecated (since Kubernetes **v1.16**)

Use the more specific livez and readyz endpoints instead.

```
[+]ping ok
[+]log ok
[+]etcd ok
[+]poststarthook/start-kube-apiserver-admission-initializer ok
[+]poststarthook/generic-apiserver-start-informers ok
[+]poststarthook/start-apiextensions-informers ok
[+]poststarthook/start-apiextensions-controllers ok
[+]poststarthook/crd-informer-synced ok
[+]poststarthook/bootstrap-controller ok
[+]poststarthook/rbac/bootstrap-roles ok
[+]poststarthook/scheduling/bootstrap-system-priority-classes ok
[+]poststarthook/start-cluster-authentication-info-controller ok
[+]poststarthook/start-kube-aggregator-informers ok
[+]poststarthook/apiservice-registration-controller ok
[+]poststarthook/apiservice-status-available-controller ok
[+]poststarthook/kube-apiserver-autoregistration ok
[+]autoregister-completion ok
[+]poststarthook/apiservice-openapi-controller ok
healthz check passed
```

**kubectl get --raw='/readyz?verbose'**

**Important** > readyz (API Server)

## HA with kubeadm

etcdctl --write-out=table --endpoints=\$ENDPOINTS endpoint status										ERRORS
ENDPOINT	ID	VERSION	DB SIZE	IS LEADER	IS LEARNER	RAFT TERM	RAFT INDEX	RAFT APPLIED INDEX		ERRORS
10.240.0.17:2379	4917a7ab173fabe7	3.5.0	45 KB	true	false	4	16726	16726		
10.240.0.18:2379	59796ba9cd1bcd72	3.5.0	45 KB	false	false	4	16726	16726		
10.240.0.19:2379	94df724b66343e6c	3.5.0	45 KB	false	false	4	16726	16726		

etcdctl --endpoints=\$ENDPOINTS endpoint health

```
10.240.0.17:2379 is healthy: successfully committed proposal: took = 3.345431ms
10.240.0.19:2379 is healthy: successfully committed proposal: took = 3.767967ms
10.240.0.18:2379 is healthy: successfully committed proposal: took = 4.025451ms
```

**Important** > health (etcd)

# Bonus - Problem - What about my App?



## Why should you care?

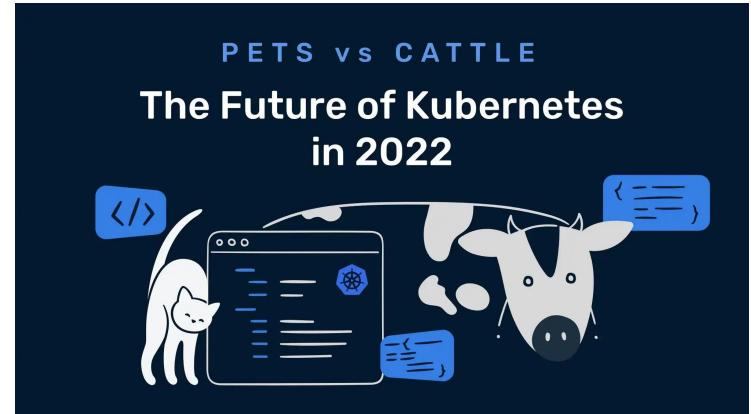
The ideal world in K8s - **stateless** applications.

Monitoring a complex application is a significant endeavor, and each language is unique.

**Cats** - Servers treated as indispensable or unique systems that can never be down.

**Cattle** - Servers using automated tools, and are designed for failure, where no one irreplaceable.

[Golden Signals](#) (Google SRE) can be a good start.  
Latency, Traffic, Errors, and Saturation.



Pets vs Cattle is still relevant in 2022  
Source - [2016, Traefik](#)

# Bonus - Problem - What about my App?



## Where to find them?

```
package main

import (
    "net/http"

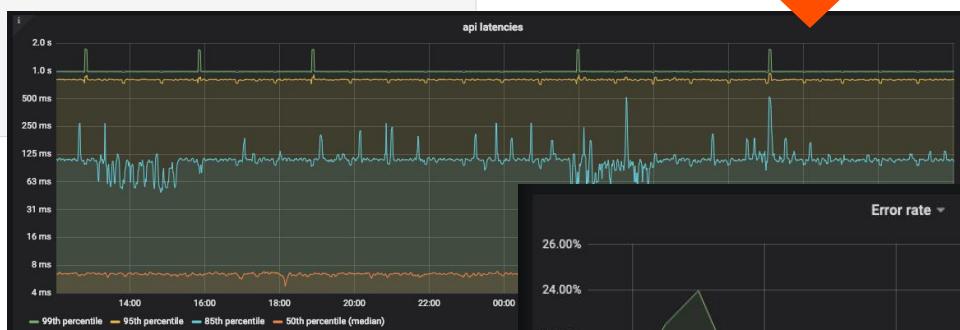
    "github.com/prometheus/client_golang/prometheus/promhttp"
)

func main() {
    http.Handle("/metrics", promhttp.Handler())
    http.ListenAndServe(":2112", nil)
}
```

Prometheus [Go client library](#) to instrument Go applications.

Note - Metrics don't provide deep context.

**Important > 95th percentile latency**  
The maximum latency, in seconds, for the fastest 95% of requests.



**Important > Error Rate %**  
The rate of requests that fail



# Recap - what we covered so far ...



- 1 Misconfiguration Oops! > Pipeline **Status**.
- 2 I need MOORRREEEE! > Resources **CPU** and **Memory**.
- 3 Network Headaches > DNS **Error**, **Health**, and Ingress **Error Rate**.
- 4 Help! My Pods are stuck! > Status **Phase**, and Status **Reasons**.
- 5 Control Plane > API Server **Readyz**, and etcd **Health**.
- 6 What about my Apps? > **P95** and **Error Rate**.

# Recap - what we covered so far ...



Post Mortem is a key process, to understand, mitigate, and prepare.

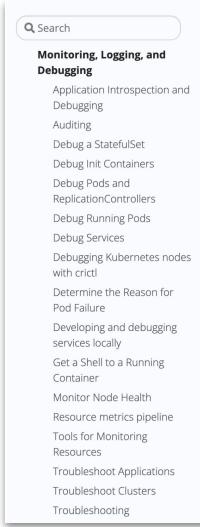
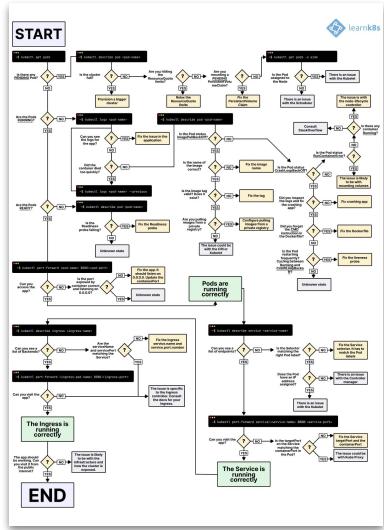
It's **ALRIGHT** for things to fail. Redirection (Plan B) is needed when things fail.

The community is getting better with Kubernetes adoption, and operational flow.

Tempting, but managed K8s isn't perfect. Fundamental understanding is still important!

Trying to improve context relationship between Apps, K8s, and related Pods.

# Recap - K8s Troubleshooting Guide



Kubernetes Documentation / Tasks / Monitoring, Logging, and Debugging

## Monitoring, Logging, and Debugging

### Application Introspection and Debugging

#### Auditing

#### Debug a StatefulSet

#### Debug Init Containers

#### Debug Pods and ReplicationControllers

#### Debug Running Pods

#### Debug Services

#### Debugging Kubernetes nodes with crictl

#### Determine the Reason for Pod Failure

#### Developing and debugging services locally

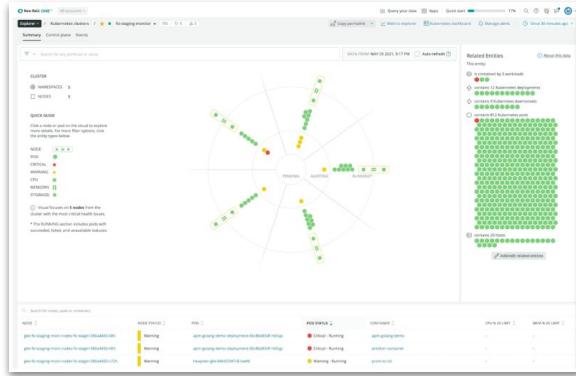
#### Get a Shell to a Running Container

Troubleshooting in Kubernetes can be intimidating if you don't know where to start.

See [HERE](#), by **learnk8s**.

See [HERE](#), by **Kubernetes.io**.

# Kubernetes Observability



[Kubernetes Cluster Explorer](#)

**Integrate your Prometheus data**

Using the Prometheus Remote Write Integration, you can point your Prometheus servers at New Relic to store and visualize your data. You can explore other ways to [integrate Prometheus data](#) in our docs.

- 1 Generate your Prometheus `remote_write` URL  
Enter some details below to help us generate your `remote_write` URL.  
NEW RELIC ACCOUNT  
RPM/LSD Demotom Q2  
Add a name to identify your data source  
`prometheus-data`  
This will add an attribute to your metrics that you can use to query them: `label{implementation,souce = "prometheus-data"}`  
[Generate url](#)
- 2 Add this `remote_write` URL to your main Prometheus configuration file
- 3 Reload or restart your Prometheus server
- 4 Check for your data  
Click the button and, in 3 minutes or less, we'll let you know when we've received your data and where to go to see it.  
[See your data](#)

[Managed Prometheus Data](#)



[Grafana & PromQL Support](#)

# QR Code Here



A copy of this deck  
(including reference links)

[Special page for KCD](#)



Kubernetes  
Community Days

## Reference Links

- [cAdvisor](#)
- [Metric Server](#)
- [API Server](#)
- [Node Exporter](#)
- [Kube State Metrics](#)

## Get started in Kubernetes with New Relic

- [Introduction to K8s Integration](#)
- [Install & Configure](#)
- [Link application to K8s](#)
- [Understand K8s Cluster Explorer](#)
- [K8s Monitoring Guide](#)