GO, GO, KUBERNETES

Building a Kubernetes Powered Central Go Modules Repository

JFrog

LET'S GO ON A TRIP THROUGH HISTORY

A QUICK HISTORY OF GO

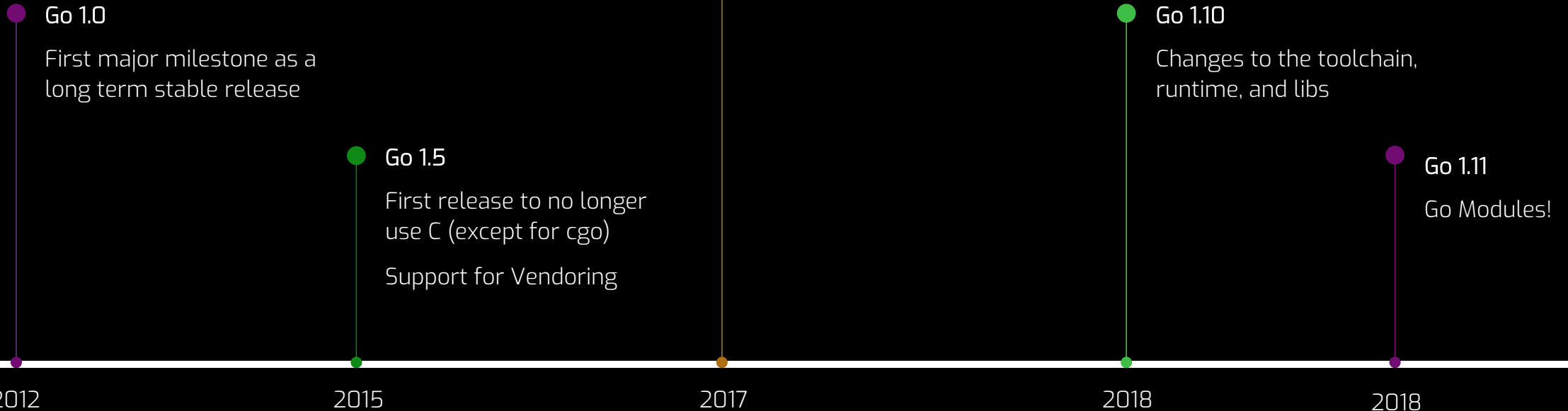**Go 1.8**

Introduction of Go plugins

Go 1.0

First major milestone as a
long term stable release

Go 1.10

Changes to the toolchain,
runtime, and libs

Go 1.5

First release to no longer
use C (except for cgo)

Support for Vendoring

2012   2015   2017   2018
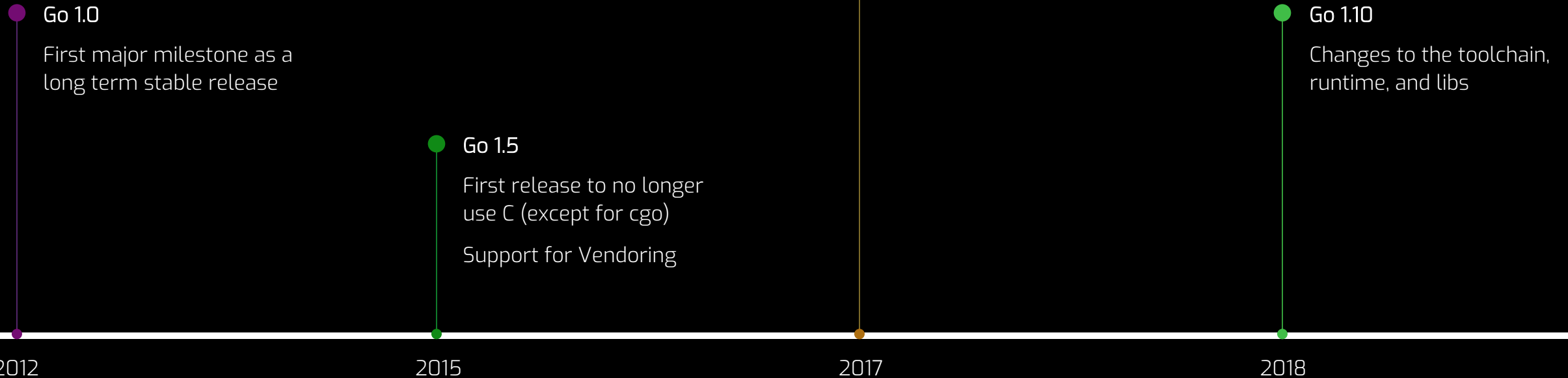
JFrog

Dependency Management…

"*Tis impossible to be sure of anything but Death and Taxes*"

- Christopher Bullock

A QUICK HISTORY OF GO

Go 1.8

Introduction of Go plugins

Go 1.0

First major milestone as a
long term stable release

Go 1.10

Changes to the toolchain,
runtime, and libs

Go 1.5

First release to no longer
use C (except for cgo)

Support for Vendoring

Go 1.11

Go Modules!

2012   2015   2017   2018   2018

SO, WHAT ARE MODULES?

Official Dependency Management

No more GOPATH

In one single tool

JFrog

FORKING IS NOT ALWAYS GOOD, BAD FORKING IS ALWAYS BAD

And that's what vendoring is

IN SOFTWARE IMMUTABLE AND REPEATABLE ARE REALLY IMPORTANT

Immutable dependencies

The best way to guarantee issues is force push

Lost Dependencies

Who doesn't remember left-pad with Node.js?

Trust

Do you trust your colleagues? What about the rest of the Internet?

WHO AM I?
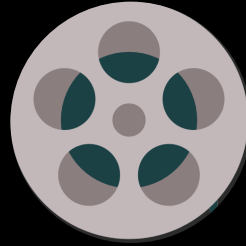

Baruch Sadogursky, Chief Sticker Officer

🐦 @jbaruch

- Head of Developer Relations

- Chief Sticker Officer

SHOWNOTES

Slides

Video

Links

Comments

Ratings

Raffle!

https://jfrog.com/shownotes

#1

WHAT IS GOCENTER ACTUALLY?

JFrog

WHAT IS GOCENTER ACTUALLY?

#2 HOW WE GOT TO KUBERNETES

JFrog

WHY DID WE CHOOSE KUBERNETES?

- Make use of a proven orchestration platform

- Leverage out-of-the-box features

RUNNING ON GOOGLE CLOUD PLATFORM

Most mature
Kubernetes
offering

Easy to deploy
using existing tools
and techniques

Lots of
troubleshooting
tools like BigQuery
and StackDriver

Google Cloud Platform

WHO NEEDS STATE ANYWAY?

1. Codebase
2. Dependencies
3. Config
4. Backing services
5. Build, release, run
6. Processes

7. Port binding
8. Concurrency
9. Disposability
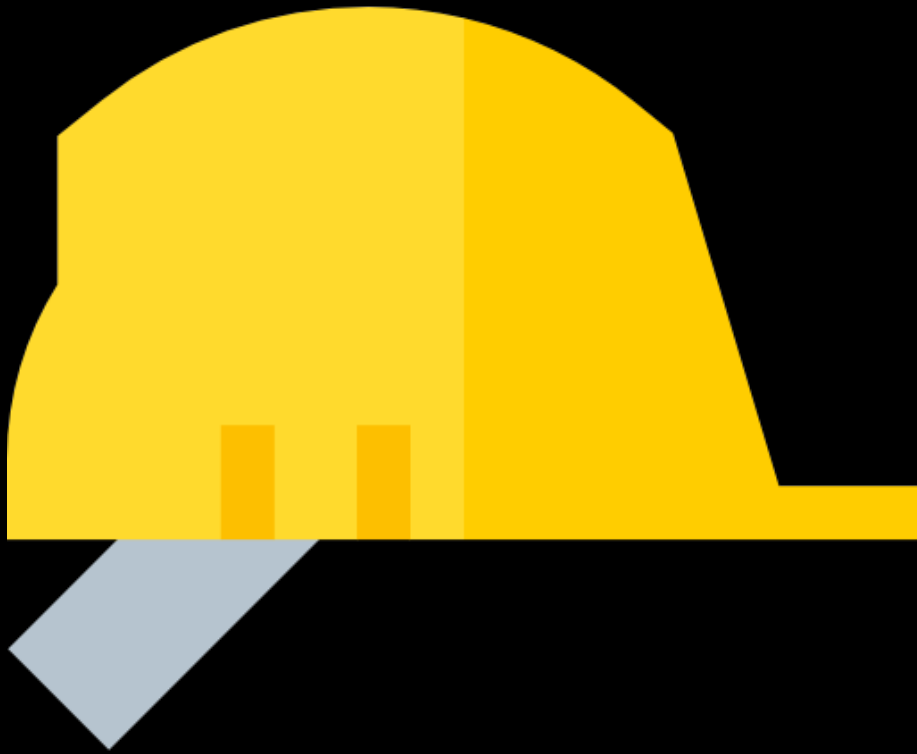10. Dev/Prod parity
11. Logs
12. Admin processes

THE TWELVE-FACTOR APP

BUT WE DIDN'T BUILD EVERYTHING OURSELVES

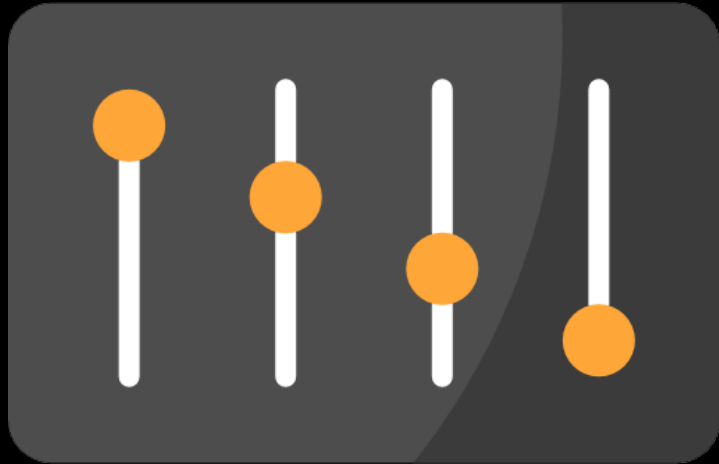@jbaruch | jfrog.com/shownotes | gocenter.io

Dependencies

DEPLOYMENTS USING HELM

- Helm Charts are versioned

- Separate value files per environment

- Having one of the co-founders of Helm at our company also doesn't hurt

Config

SOME BEST PRACTICES WE LEARNED WHILE BUILDING

Limits & Labels

Secrets

Namespaces & Clusters

Portbinding, Concurrency & Disposability

JFrog

#3 HOW WE BUILT OUR INFRASTRUCTURE

WE'RE BIG ON THOSE PIPELINES AND PROMOTIONS CONCEPTS

CI SERVER

IF QUALITY REQUIRMENTS ARE HIT

IF QUALITY REQUIRMENTS ARE HIT

IF QUALITY REQUIRMENTS ARE HIT

INTEGRATION

SYSTEM TESTING

STAGING

PRODUCTION

- QUALITY GATES -

JFrog

5
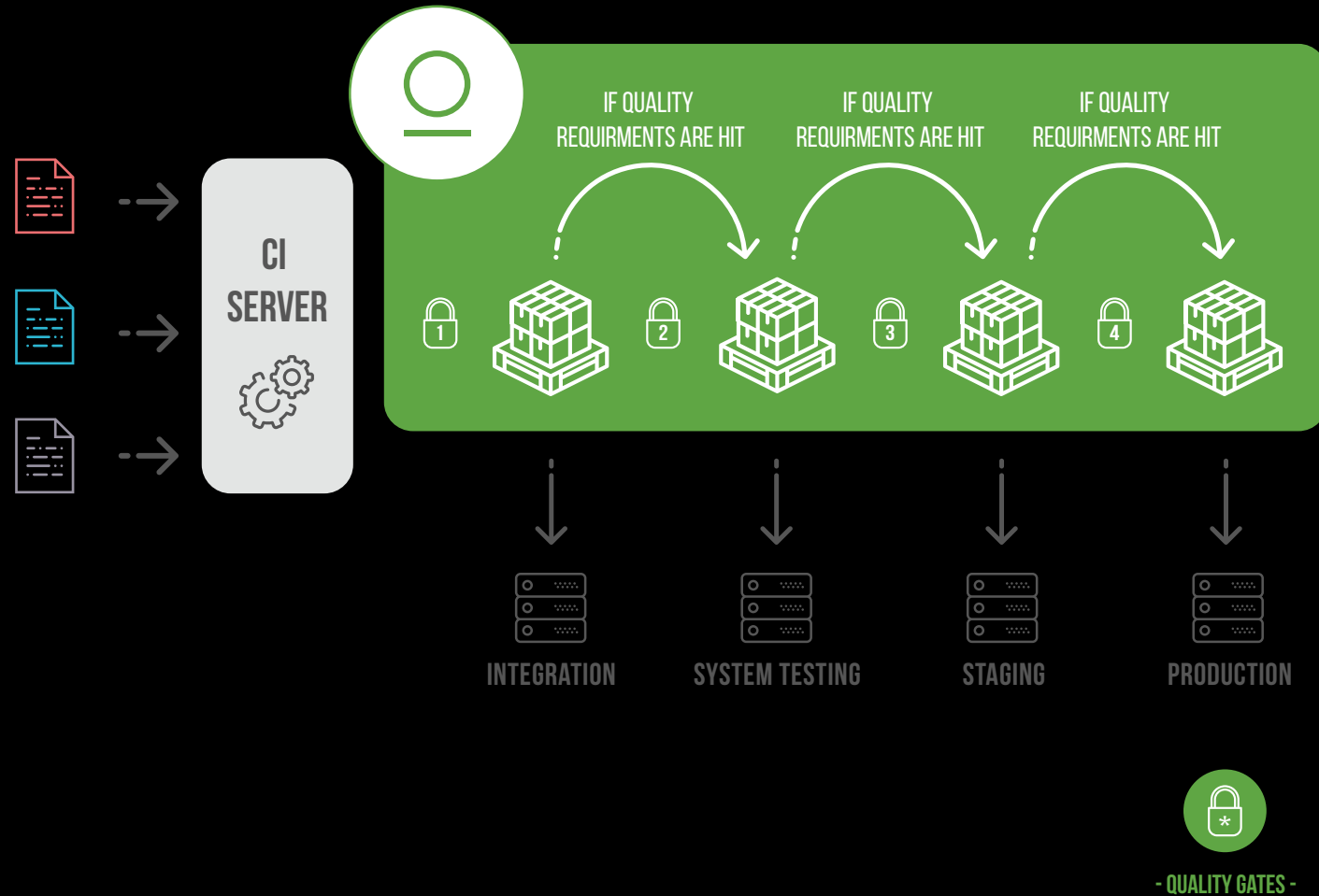
NUMBER OF CLUSTERS CURRENTLY RUNNING
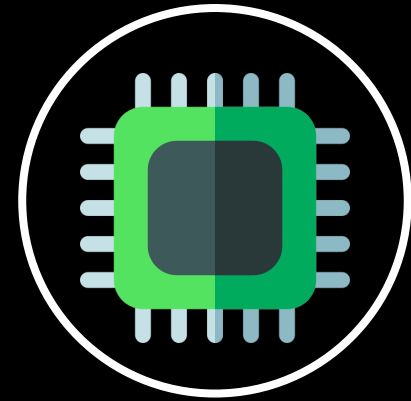
Dev/Prod Parity

JFrog

RUNNING A FEW MICROSERVICES

Discovery

Notifier

Processor

RUNNING A FEW MICROSERVICES

Validator
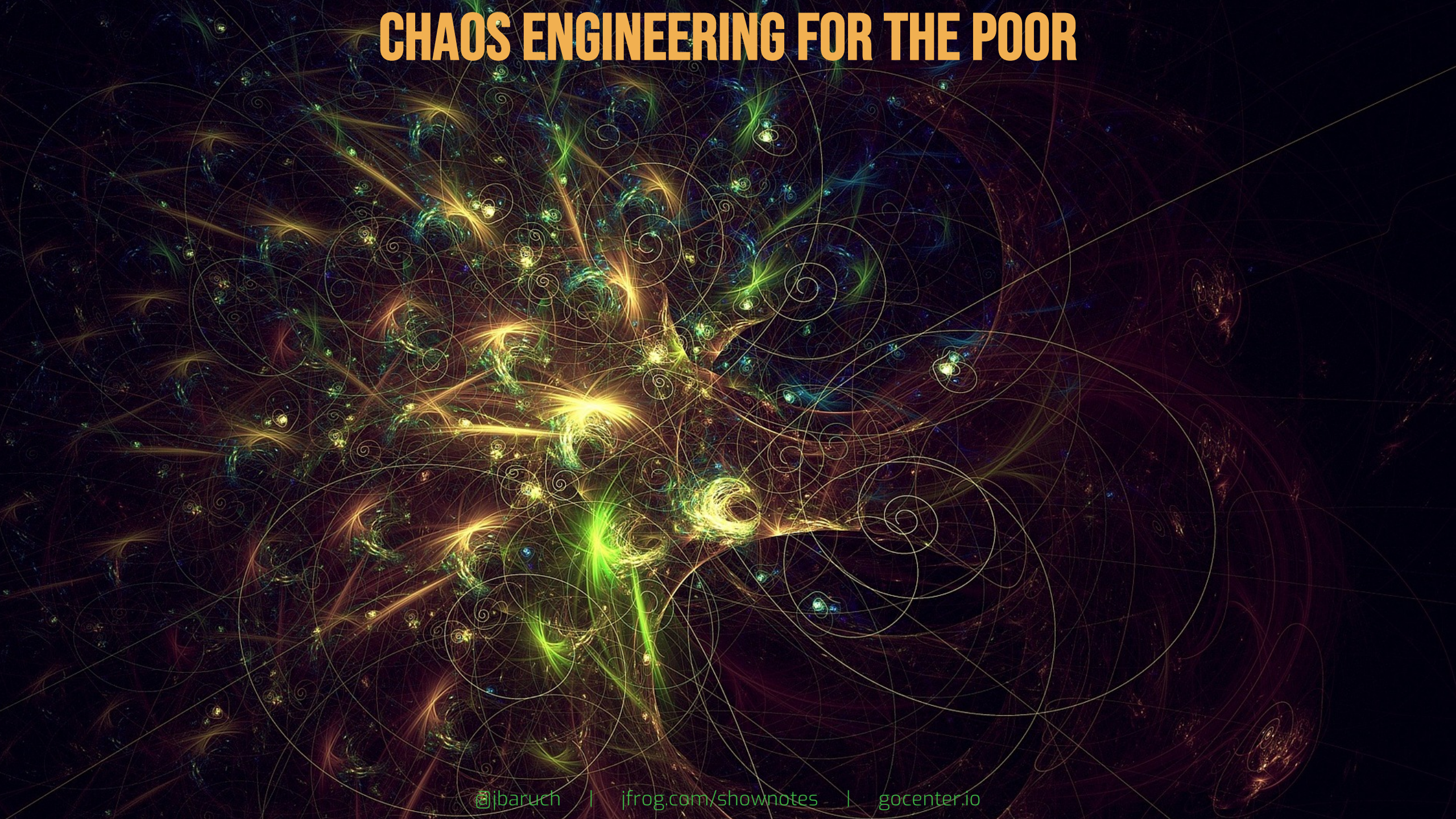
UI Backend

UI

THIS IS WHAT OUR CLUSTER LOOKS LIKE

GoCenter Kubernetes Cluster

Google Cloud

SQL

PubSub

Artifactory

Prd

Stg

Dev
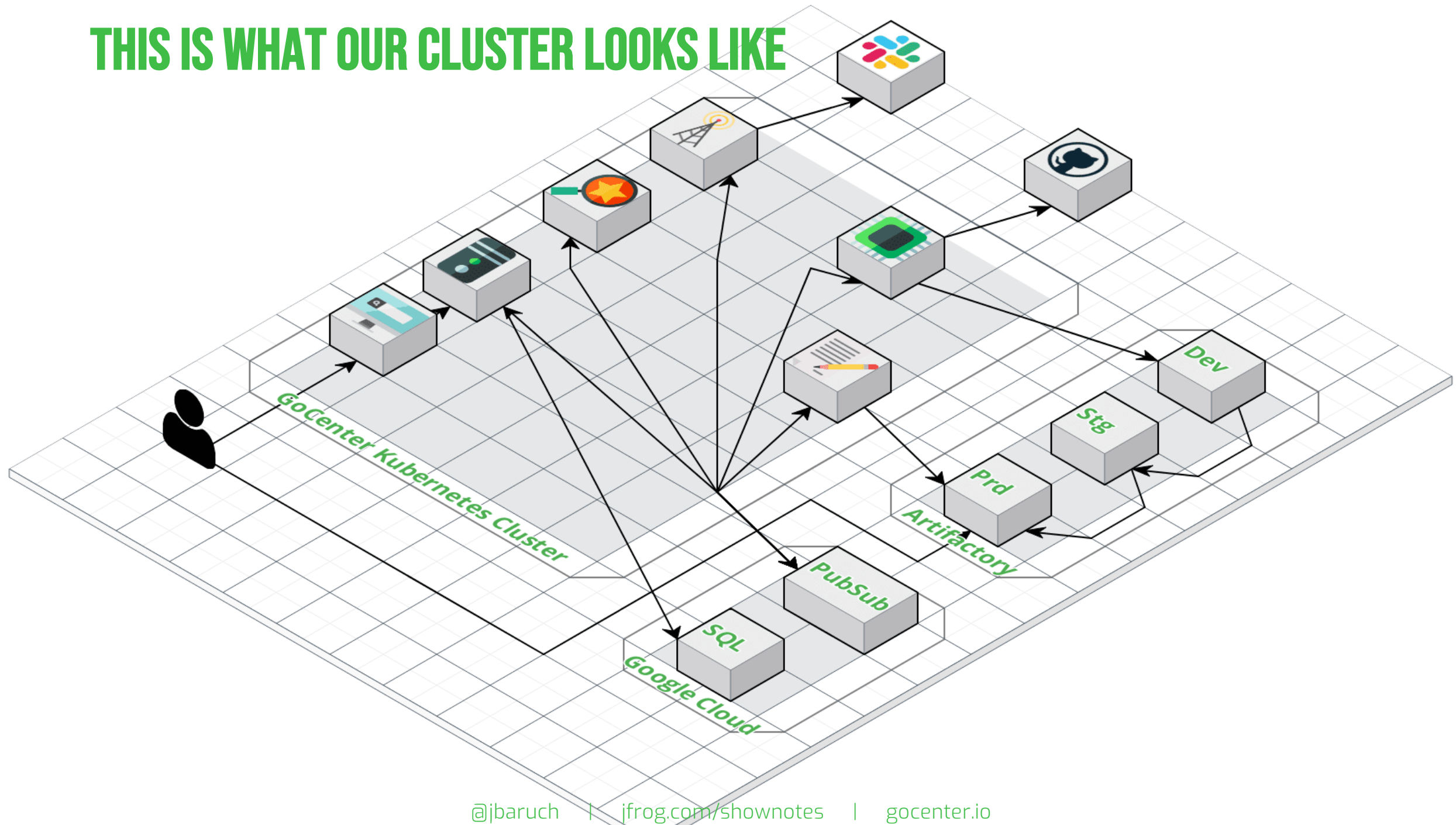
LOAD TESTING

- UI: 10K users with 100 requests per second

- Backend: Load 10K modules... twice

GoCenter

Filter...

Kubernetes Nodes - CPU utilization
1 | 1 hr interval (mean)

100%
80%
60%
40%
20%

Mon 07 Wed 09 Fri 11 Sun 13 Tue 15 Thu 17 Sat 19 Mon 21 Wed 23 Fri 25 Sun 27 Tue 29 Feb 01 Sun 03 Tue 05

Container - CPU limit utilization for caddyshack.*
1 | 1 hr interval (mean)

120%
100%
80%
60%
40%
20%
0

Mon 07 Wed 09 Fri 11 Sun 13 Tue 15 Thu 17 Sat 19 Mon 21 Wed 23 Fri 25 Sun 27 Tue 29 Feb 01 Sun 03 Tue 05

Container - Log entries for ERROR, caddyshack.*
2 | 1 hr interval (rate)

0.8/s
0.6/s
0.4/s
0.2/s
0

Mon 07 Wed 09 Fri 11 Sun 13 Tue 15 Thu 17 Sat 19 Mon 21 Wed 23 Fri 25 Sun 27 Tue 29 Feb 01 Sun 03 Tue 05

PubSub - Number of unacknowledged messages by Cloud region
1 hr interval (mean)

5,000
4,000
3,000
2,000
1,000
0

Mon 07 Wed 09 Fri 11 Sun 13 Tue 15 Thu 17 Sat 19 Mon 21 Wed 23 Fri 25 Sun 27 Tue 29 Feb 01 Sun 03 Tue 05

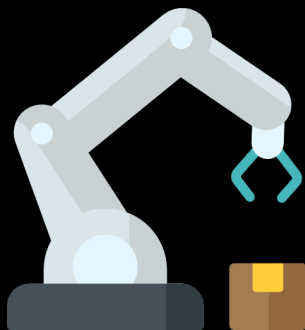# CHAOS ENGINEERING FOR THE POOR

#4   RUNNING AT SCALE

NODE POOLS, CAUSE EVEN NODES HAVE TO RELAX?

DEPLOYMENTS USING HELM

```
resources:
  limits:
    cpu: 800m
    memory: 2Gi
  requests:|
    cpu: 800m
    memory: 1Gi
```

```
spec:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: node-pool
            operator: In
            values:
            - pvm-pool
    podAntiAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
      - labelSelector:
          matchExpressions:
          - key: app
            operator: In
            values:
            - discovery
        topologyKey: kubernetes.io/hostname
```
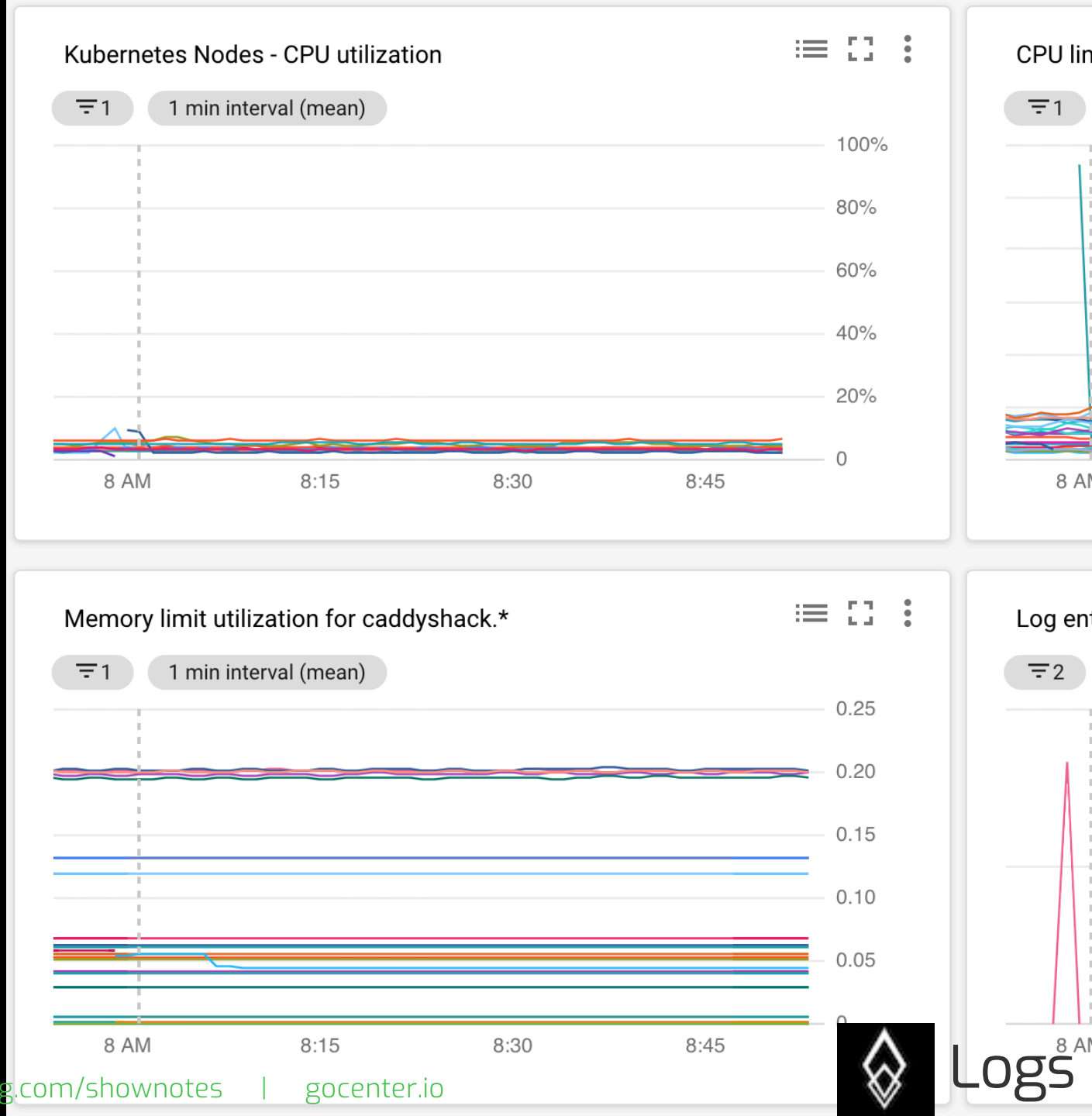
https://hub.helm.sh

Build, release, run

KEEPING A CLOSE EYE ON CONTAINER HEALTH

- FluentD

- Prometheus

- Grafana

[Live Dashboard](#)

Kubernetes Nodes - CPU utilization

1   1 min interval (mean)

100%
80%
60%
40%
20%
0

8 AM   8:15   8:30   8:45

CPU lin

1

8 AM

Memory limit utilization for caddyshack.*

1   1 min interval (mean)

0.25
0.20
0.15
0.10
0.05
0

8 AM   8:15   8:30   8:45

Log en

2

8 AM

Logs

SOMETIMES IT'S GOOD TO BRING A FRIEND ALONG

WE KEEP IN TOUCH WITH OUR NODES

Kube-Slack

jenkins APP 7:24 AM
Started Job 'SolDev/caddyshack_deploy_prod' to deploy '0.1.147-2' on prod.

gocenter APP 7:25 AM
containers with unready status: [caddyshack-proxy]

caddyshack/caddyshack-webui-85d58dd497-t66lk: ContainersNotReady
containers with unready status: [caddyshack-proxy]

Pod is ready

caddyshack/caddyshack-webui-85d58dd497-t66lk: Pod is ready
Pod is ready

containers with unready status: [caddyshack-ui caddyshack-proxy]

caddyshack/caddyshack-webui-85d58dd497-kjbs8: ContainersNotReady
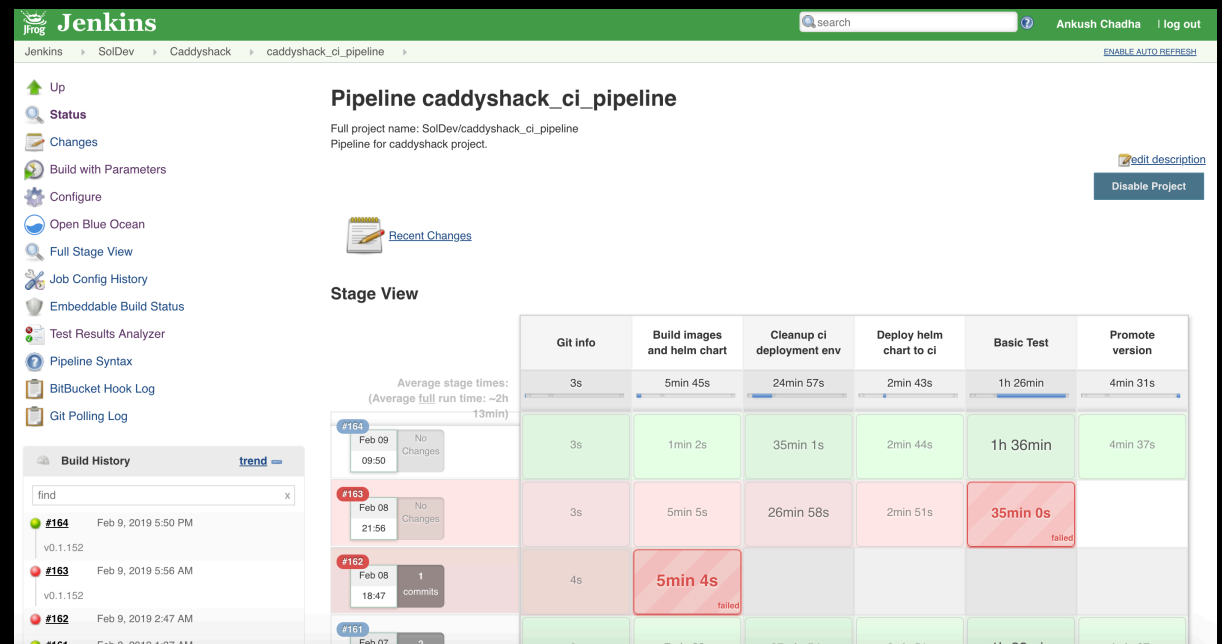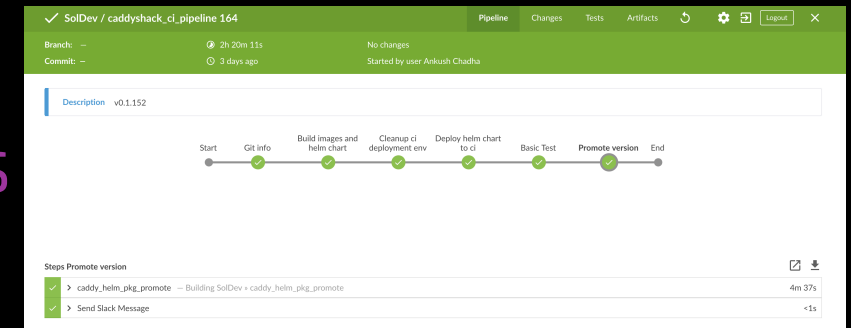containers with unready status: [caddyshack-ui caddyshack-proxy]

Pod is ready

caddyshack/caddyshack-webui-85d58dd497-kjbs8: Pod is ready
Pod is ready

jenkins APP 7:28 AM
Finished Job 'SolDev/caddyshack_deploy_prod'. Deployment '0.1.147-2' on prod.

Jenkins

Logs & Build, release, run

JFrog

SCALING ON MANY LEVELS

Node
Provision nodes when needed

Compute
Run more microservices

Messaging
Scaling based on queue depth

Up and Down
What goes up must come down... usually

Concurrency

#5 DARE TO DREAM

WHAT'S NEXT FOR GOCENTER

ChatOps
Use Slack to let teams take more control

In-Cluster Artifactory
Making upgrades more seamless, but…

Smart dependency selection
Ratings, filters, etc

JFrog

TWITTER ADS AND Q&A

jfrog.com/
shownotes

@jbaruch

gocenter.io