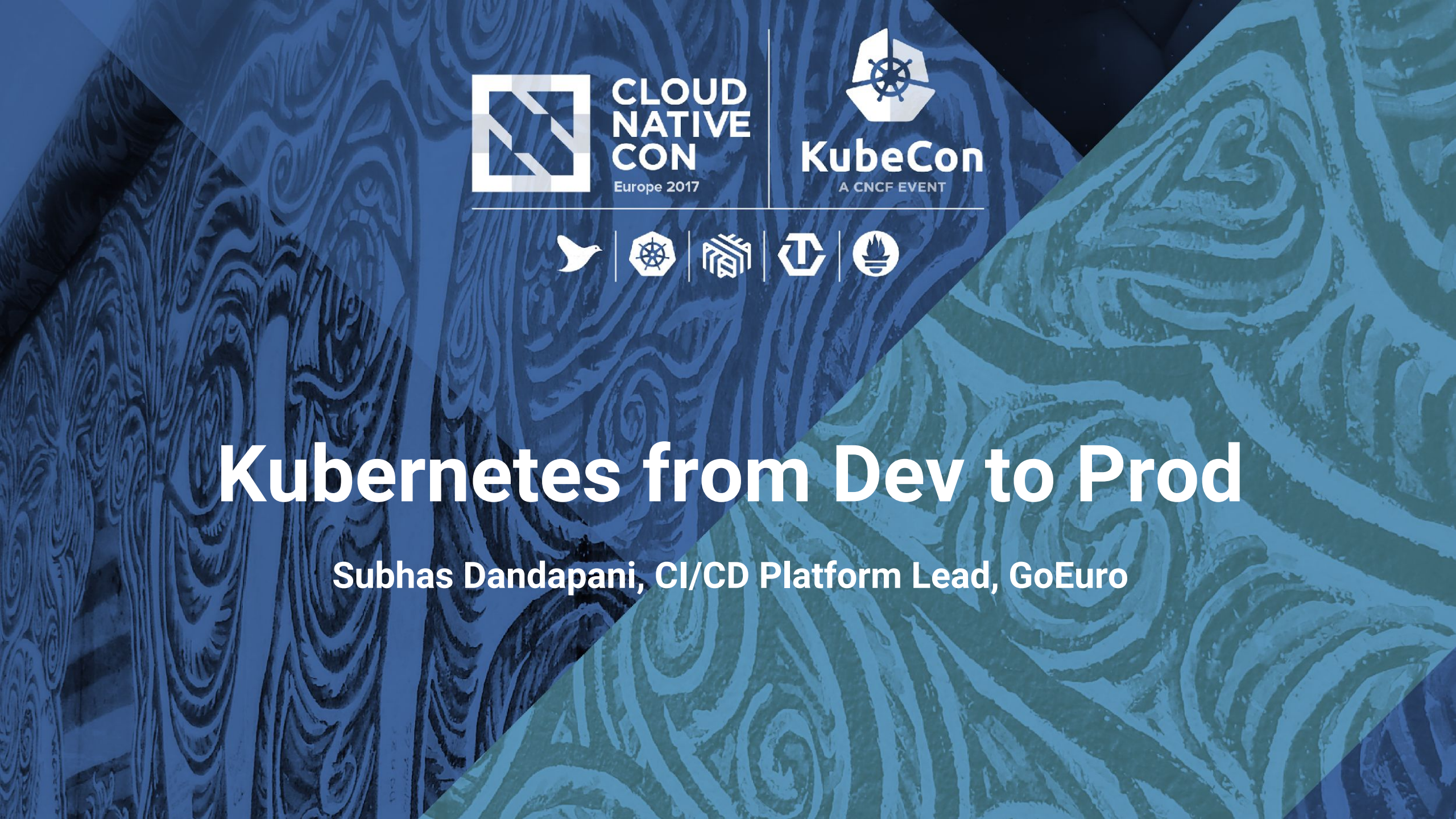



# Kubernetes from Dev to Prod

Subhas Dandapani, CI/CD Platform Lead, GoEuro

CLOUD NATIVE CON
Europe 2017

KubeCon
A CNCF EVENT

GOEURO

12 countries
30k+ stations
200+ airports
10+ million monthly searches

Subhas Dandapani

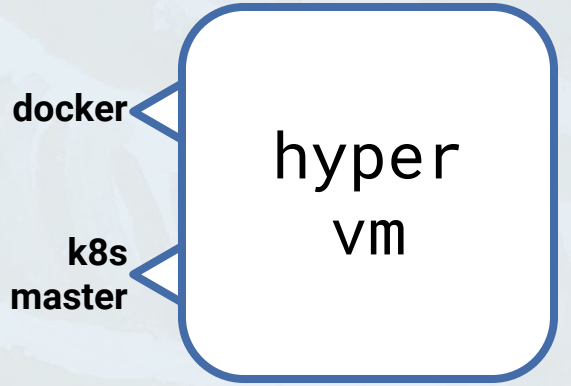
CI/CD Platform Lead
medium.com/@rdsubhas
github.com/rdsubhas

# Transition Journey

- 50 engineers, 10 services → 150 engineers, 50 services
- 10 coordinated releases → 150+ independent releases per week
- Legacy → CI/CD/Docker/K8s
  Excluding persistent services
- Through all kinds of legacy problems
  Integration patterns: REST, File, MQ, DB-driven, Metrics-driven, Compile-time, Ancillary artifacts
  Dev hacks for ops issues, Ops hacks for dev issues
- Transition in 4 months across company
  (ground work laid before in a pilot project)

CLOUD
NATIVE
CON
Europe 2017
KubeCon
A CNCF EVENT
Development
GOEURO

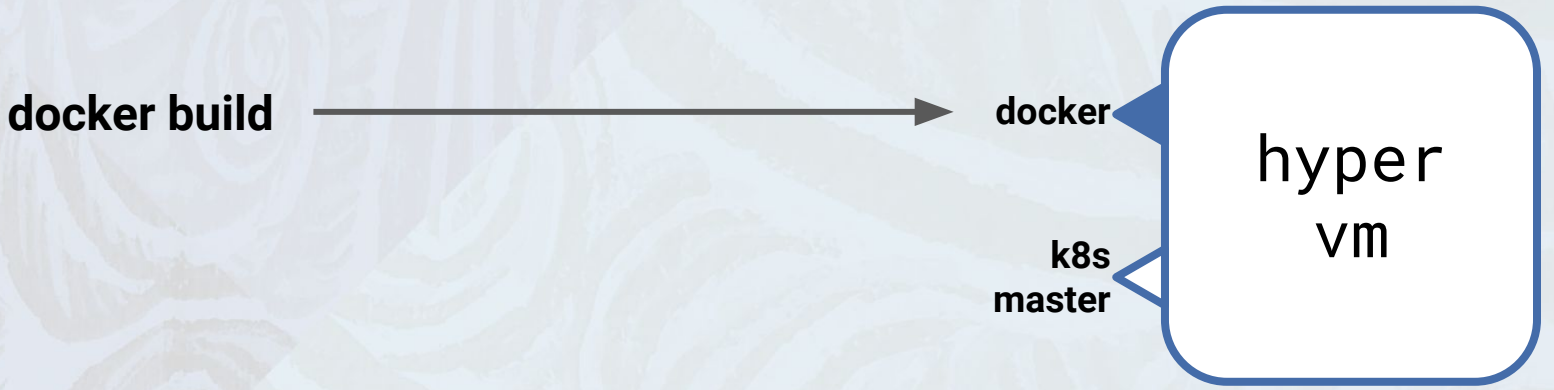
# From the bottom up: hyper-vm




- Single-node k8s+docker VM for development
- docker build & kubectl apply on same node
- Metadata based kubernetes customization
  (e.g. installing services at bootup)
- Optimized for development & testing in the cloud
  docker bind mounting (nested resolv, etc), integrate with internal network, build-arg-activated docker server patches (build caching), etc
- First point to upgrade kubernetes from dev to prod

# Development

```
service-repo
├ ops
├─ Dockerfile
├─ helm chart
├─ ci.yaml
├─ Makefile
```




- Every service has "ops" folder
- Self-contain most dev+ops requirements

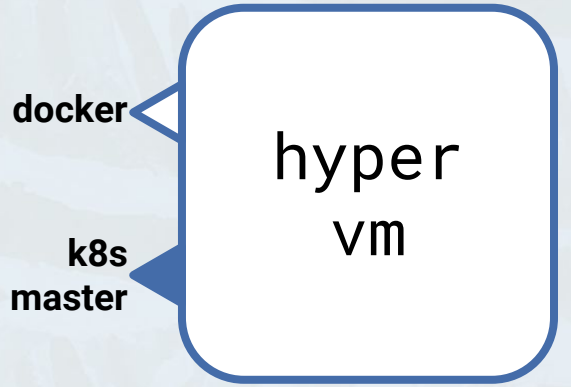
# y8s (yaml-k8s-sharing)

```
service-repo
├ ops
├─ Dockerfile
├─ helm chart
├─ ci.yaml
```

- Adopt Helm Chart standard/layout/templating
- But Render-as-a-service
- Additional simplification:
  - 1 input: “env”, configure everything else out of the box
  - env=dev: use local image
  - env=test: use stubs (provider-driven stubs/mocks)
  - env=qa/preprod/prod: pin revision
  - Optional params (overrides, etc)

# y8s (yaml-k8s-sharing)

```
kubectl apply -f
  http://y8s.int/repos/any-service-repo
    ?env=<test|dev|qa|...>
     &values[foo]=bar&filters[kind]=Service&...
```




- kubectl apply **any repo**
- DevOps services also applied/deployed same way (router, ingress, core services, database, couchbase, etc)
- Additional REST options (values, filters, validate, dry run, json/yaml content types, etc) and generated metadata (for tracing back to source)

# Landscape

| Develop | Integrate | Deploy |
| --- | --- | --- |
| ops/chart in repo<br>self contained repos | | |
| y8s<br>consume any repo/service | | |
| hyper-vm<br>develop in the cloud | | |

CLOUD
NATIVE
CON
Europe 2017
KubeCon
A CNCF EVENT
Integration
GOEURO

# CI Contract

```
service-repo
├ ops
├─ Dockerfile
├─ helm chart
├─ ci.yaml
```

- Adopt GitLab CI.yaml syntax/design
- But parse/implement it ourselves
- Automate all CI pipelines/jobs
- We orchestrate all triggers, do all plumbing

  pr checks, manual checkpoints, cron jobs, distributed caching, nested docker builds, environments, retry/reassignment rules, slack notifications, emails, release annotations/graphs, etc

# ci.yaml

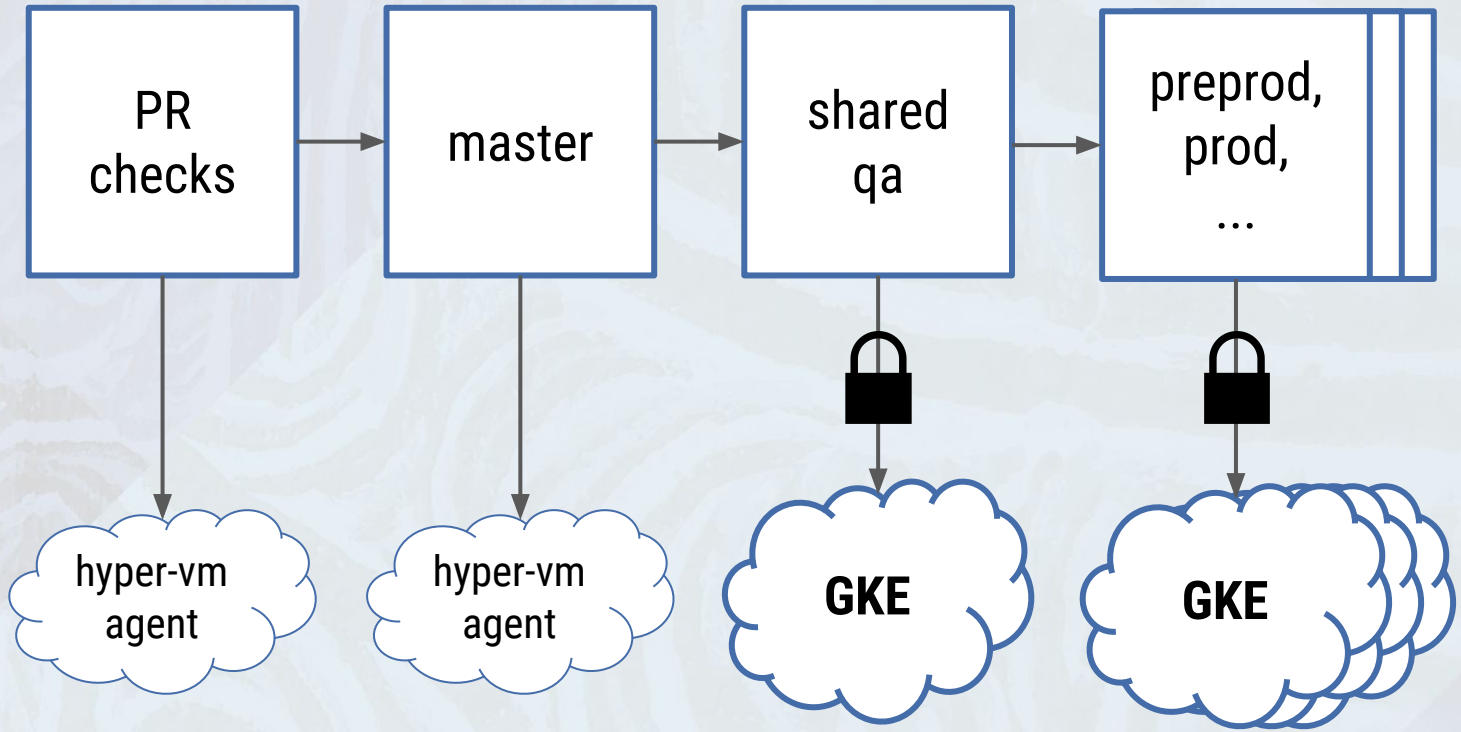
```
image: jenkins-plain:latest

unit-tests:
  stage: test
  script: test.sh

master:
  script: release.sh

deploy-qa:
  environment:
    name: qa
    credentials: "..."
  script: deploy.sh

deploy-preprod:
deploy-prod:
   ...
```

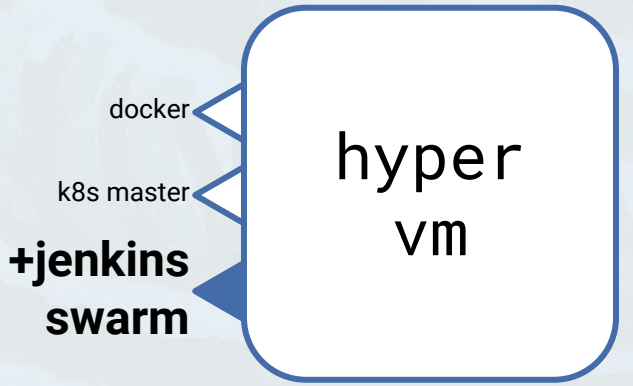


# CI Contract

- Every agent is hyper-vm + jenkins slave
- docker/kubectl readily configured
- Jobs run inside configurable build images
- Recreate from source every 2 hours

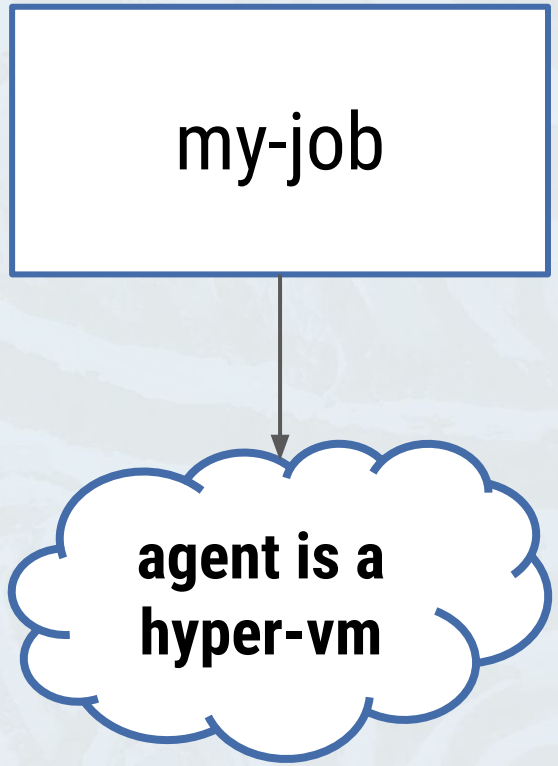
  with no downtime, no possibility of snowflake CI
- Encourage early integration tests

  `kubectl apply -f y8s.g.int/repos/anything and test`
- **Take Jobs all the way to Prod** (safety in upcoming slides)



# Job Environment - Closer Look #1

```
image: jenkins-plain:latest
my-job:
  script: make build
```

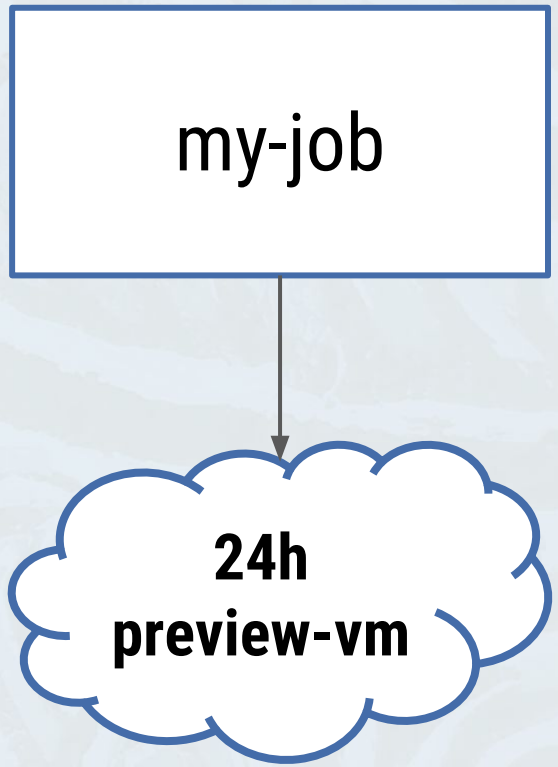



- Start using docker/kubectl right away
- Cleaned up after every build
- Early Integration/E2E tests

```
kubectl apply -f "y8s.int/...dependencies..."
kubectl run / docker run <e2e tests>
```

# Job Environment - Closer Look #2

```
my-job:
  script: make deploy
  environment:
    name: preview-24h
```




- Provision on-the-fly ephemeral hyper-vm

  Same build contract, kubectl/docker readily pointed to that VM
- But lives after job completes
- Show demos, connect for development/debugging

# Job Environment - Closer Look #3

```
my-job:
  script: make deploy
  environment:
    name: qa|preprod|prod-region
    credentials: <credentials>
```

- Apply to Deploy! Including prod regions
- We'll talk about security in later slides

# Composition

```
my-job:
   stage: pr
   when: manual
   environment:
      name: preview-24h
```

Run on PRs

when: manual
on pipelines: manual checkpoint
on PRs: comment “trigger my-job”

And boot an ephemeral preview VM

Product demos on demand,
GitHub Flow (deploy from PR to prod), etc

# Patterns

- Database migrations
  - `kubectl run --overrides=$(curl -H ...json y8s.g.int/...migration)`
- End to end testing
  - Shared E2E tests treated as services, have their own ops folder as well
  - `kubectl apply y8s.g.int/repos/... && kubectl run <e2e-tests>`
- Additional Services
  - helm templates for services like selenium, couchbase, mysql, etc
  - `kubectl apply y8s.g.int/...services…`
  - No need for additional abstraction (like travis-ci services, for example)

# Landscape

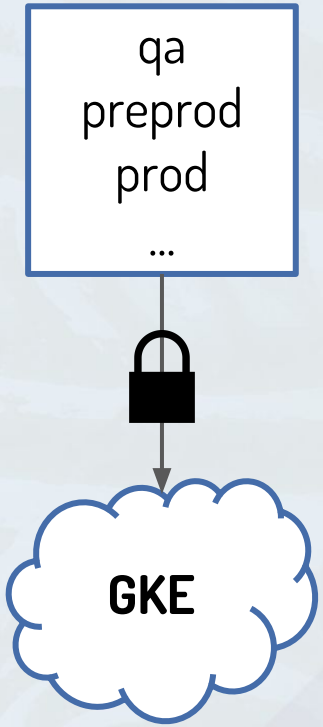
| Develop | Integrate | Deploy |
| --- | --- | --- |
| ops/chart in repo<br>self contained repos | ci.yaml<br>pipelines & patterns | |
| y8s<br>consume any repo/service | hyper-vm + ci agent<br>early E2E in CI | |
| hyper-vm<br>develop in the cloud | ready environments<br>preview, qa, preprod, prod, ... | |

CLOUD
NATIVE
CON
Europe 2017
KubeCon
A CNCF EVENT
Deployment
GOEURO

# auth8s

- Kubernetes API gateway/proxy
  - Deployed in every k8s cluster, proxies to kubernetes master
- Security
  - Each namespace gets its own username/password

    k8s 1.6 RBAC is in “beta”, this security part can be offloaded to k8s when its stable
- Compliance
  - Validate and enforce CPU/Memory Limits, Live/Readiness probes, ...
  - Necessary annotations on every resource, whitelisted resource kinds, ...
  - Much more using HTTP interception, Necessary for Governance and Standards
  - Supports most features (except interactive shells)



# Value additions: Cluster services

- Auto-register DNS for independent clusters
- Metrics: heapster, restart counters, etc
- Logging: Ship structured container logs using filebeat
- Routing: nodePort, containerizing current router, etc
  - Moving to: Ingress
  - Automatically instrument external health checks (pingdom-style)

# Value additions: Sidecar and Regional services

- Sidecar container for big artifacts for legacy services
- Region-specific external services
  - All devops-managed external resources (db, logstash, etc) have their own pipelines and yamls
  - Provide containerized/test in dev/qa/preprod/etc
  - Endpoints pointing to persistent resources in prod
- Multi-region focused from the beginning

# Value additions: Annotation driven

- Coming up: More annotation-driven cluster services
  - Capture periodic thread-dumps by declaring annotations
  - Autoscale by declaring metric queries in annotations
  - Automated external uptime checks and status pages for ingresses
  - etc

# Landscape

| Develop | Integrate | Deploy |
| --- | --- | --- |
| ops/chart in repo<br>self contained repos | ci.yaml<br>generated pipelines & jobs | gke<br>best kubernetes experience |
| y8s<br>consume any repo/service | hyper-vm + ci agent<br>early E2E in CI | auth8s<br>security, compliance |
| hyper-vm<br>develop in the cloud | ready environments<br>preview, qa, preprod, prod, ... | value added services<br>routing, logs, metrics, dns, ... |

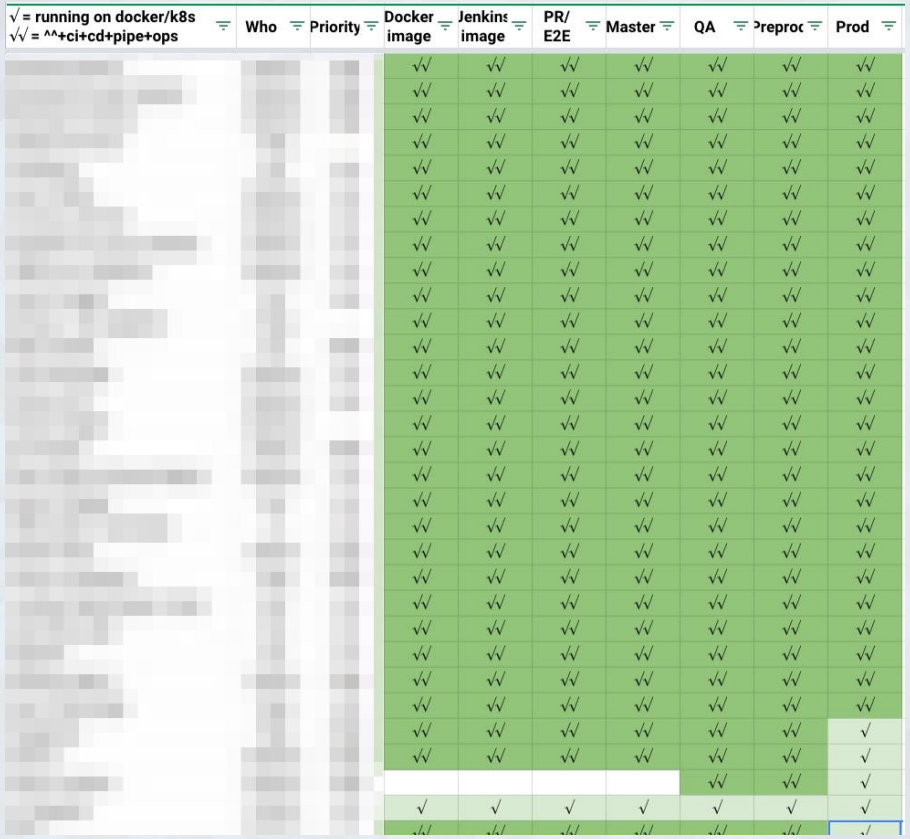
CLOUD
NATIVE
CON
Europe 2017
KubeCon
A CNCF EVENT
Learnings
GOEURO

# What we learned?

- Containers are a cultural change
  - Lots of presentations, PoCs, consulting, lobbying, “we needed this yesterday”
  - We did at same time: CI, CD, Unversioning, Dockerizing, Kubernetizing
  - Couldn’t have done without support from teams, managers, product, business
  - Don’t try too much: Changing monitoring, routing, building, processes, etc.
- Migrate as-it-is
  - Services closer to 12factor, With small dep-graph, Hard legacy challenges
  - 5 minute bootup, 400 MB images, Dev vs Ops hacks, etc
  - Run with 0% traffic, Shadow comparison, QA/Preprod, Traffic switching
  - Establish the pipe, Optimize later

# What we learned?

- Work with feature teams
  - Get out of the DevOps desk/room/...
  - Be part of their journey
  - Let them own their success
  - Avoid same mistake of dev vs ops silos
- Embrace Standards, Validate Subset
  - Too many DevOps tool ~micro-JS explosion
  - Choose wisely, Draw boundaries, Be compatible
- Less tools, More APIs, More Dogfooding

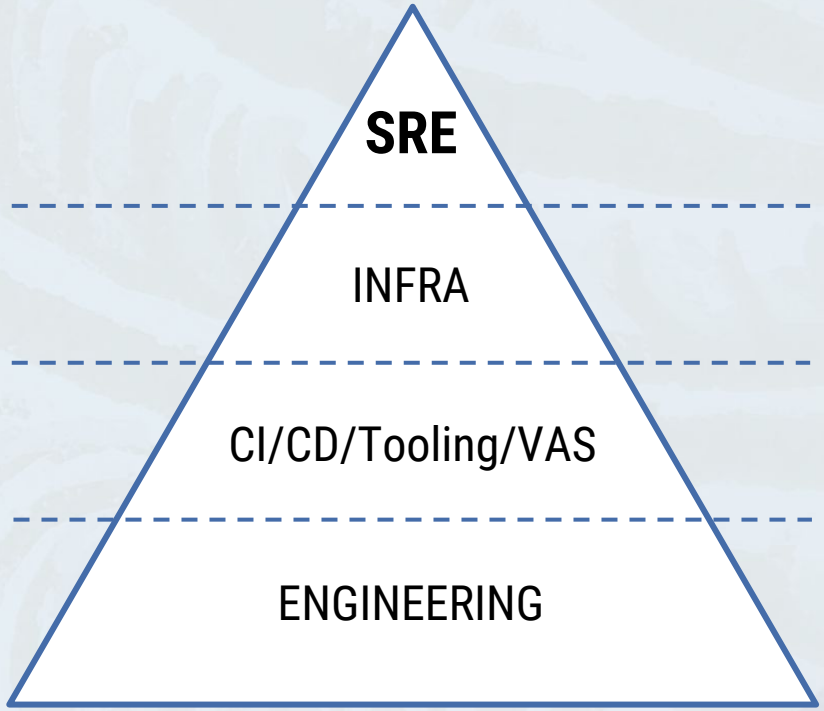


# What Kubernetes Enabled Us

- Transition from Cloud resources => Container resources
  - From Cloud: Disk Image + InstanceGroup/ASG + LoadBalancer
  - To Container: Docker Image + K8s Deployment + K8s Service/Ingress
  - From: packer build, terraform apply
  - To: docker build, kubectl apply
- Allow teams to deliver from Dev to Prod
  - Balance safety, compliance, automation and ease-of-use
- Transition legacy and modern services
  - Integrate and iterate fast, next 1000 releases per week

# Stepping Forward

- Site Reliability Engineering
  - Teams deliver up to prod-x%
  - Rollout to rest automatically based on metrics
- Better State Management
- Secret Management
  - Multiple possibilities, ci.yaml, auth8s, etc
- Ingress, Federation
- Namespace Quotas
  - Virtual data center, Team budgets, etc



CLOUD
NATIVE
CON
Europe 2017
KubeCon
A CNCF EVENT
Questions
GOEURO