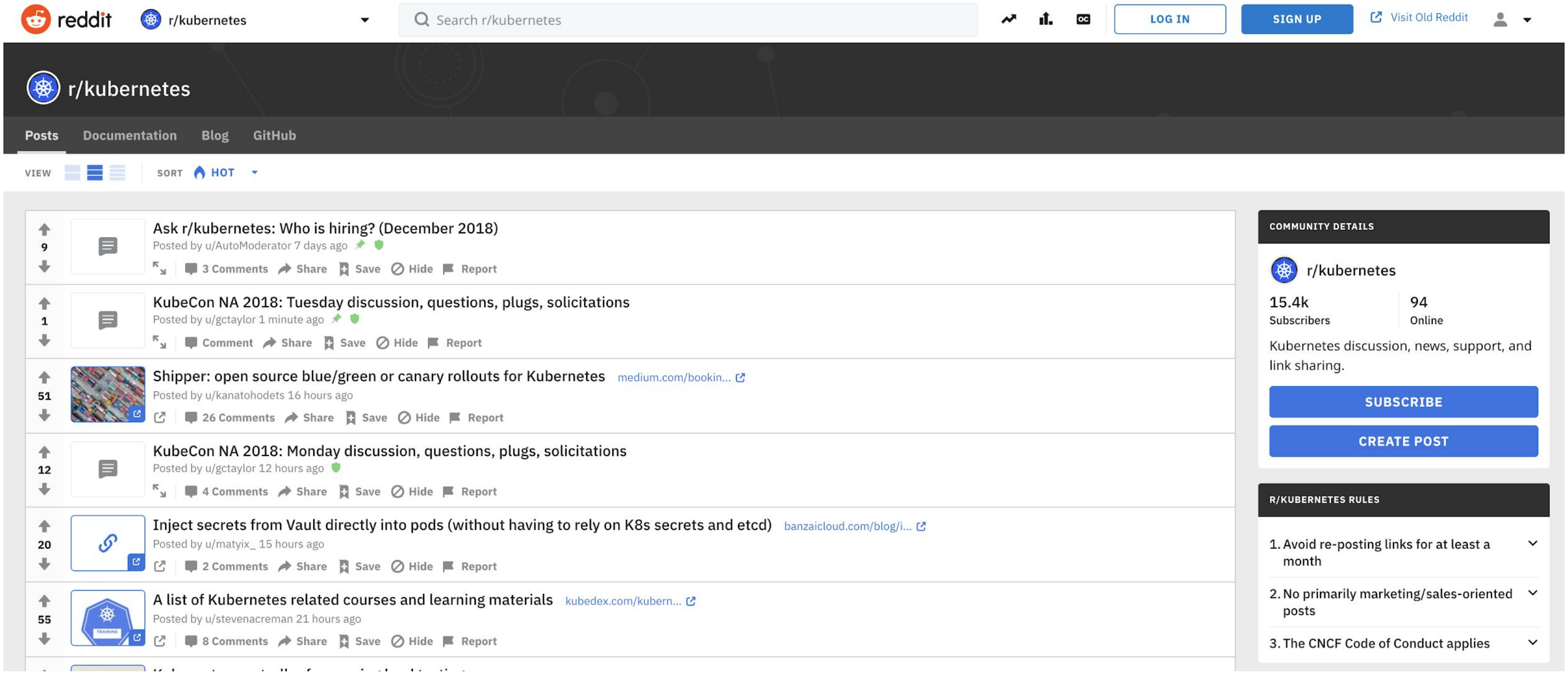
Kubernetes at Reddit: Tales from Production

Greg Taylor – EM, Reddit Infrastructure
/u/gctaylor

What is Reddit?

Example: /r/kubernetes



9 Ask r/kubernetes: Who is hiring? (December 2018)
Posted by u/AutoModerator 7 days ago
3 Comments Share Save Hide Report

1 KubeCon NA 2018: Tuesday discussion, questions, plugs, solicitations
Posted by u/gctaylor 1 minute ago
Comment Share Save Hide Report

51 Shipper: open source blue/green or canary rollouts for Kubernetes medium.com/bookin...
Posted by u/kanatohodets 16 hours ago
26 Comments Share Save Hide Report

12 KubeCon NA 2018: Monday discussion, questions, plugs, solicitations
Posted by u/gctaylor 12 hours ago
4 Comments Share Save Hide Report

20 Inject secrets from Vault directly into pods (without having to rely on K8s secrets and etcd) banzaicloud.com/blog/i...
Posted by u/matyix_ 15 hours ago
2 Comments Share Save Hide Report

55 A list of Kubernetes related courses and learning materials kubedex.com/kubern...
Posted by u/stevenacreman 21 hours ago
8 Comments Share Save Hide Report

COMMUNITY DETAILS

r/kubernetes

15.4k Subscribers

94 Online

Kubernetes discussion, news, support, and link sharing.

SUBSCRIBE

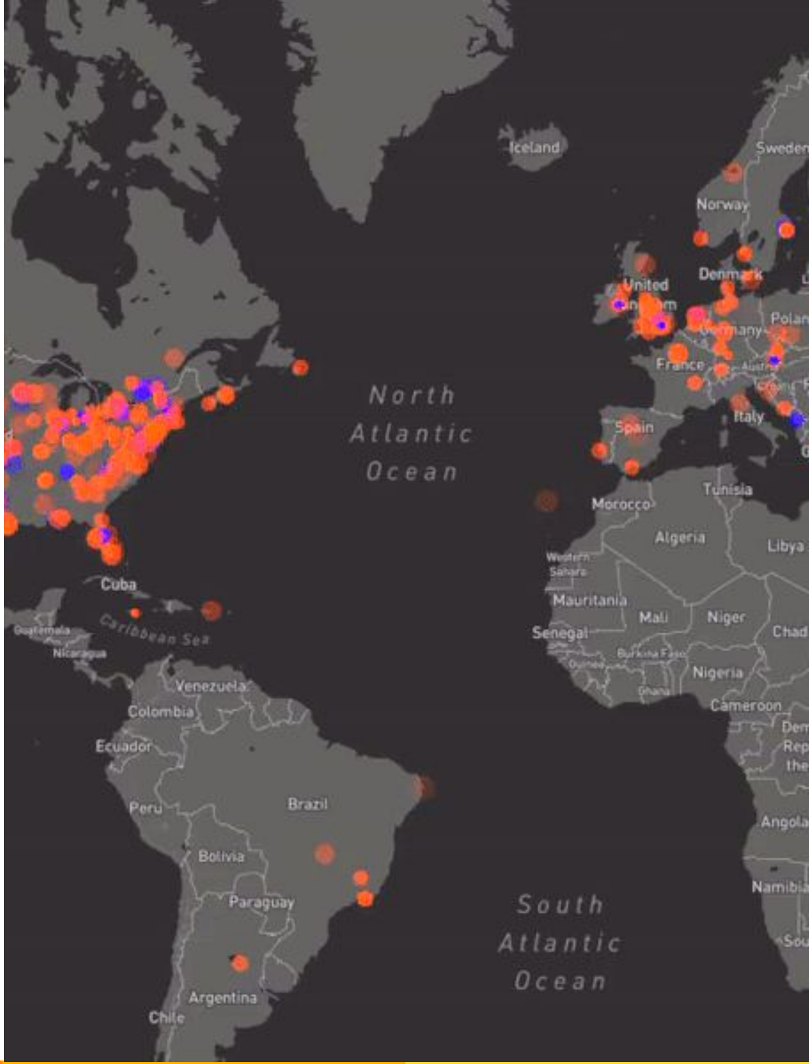
CREATE POST

R/KUBERNETES RULES

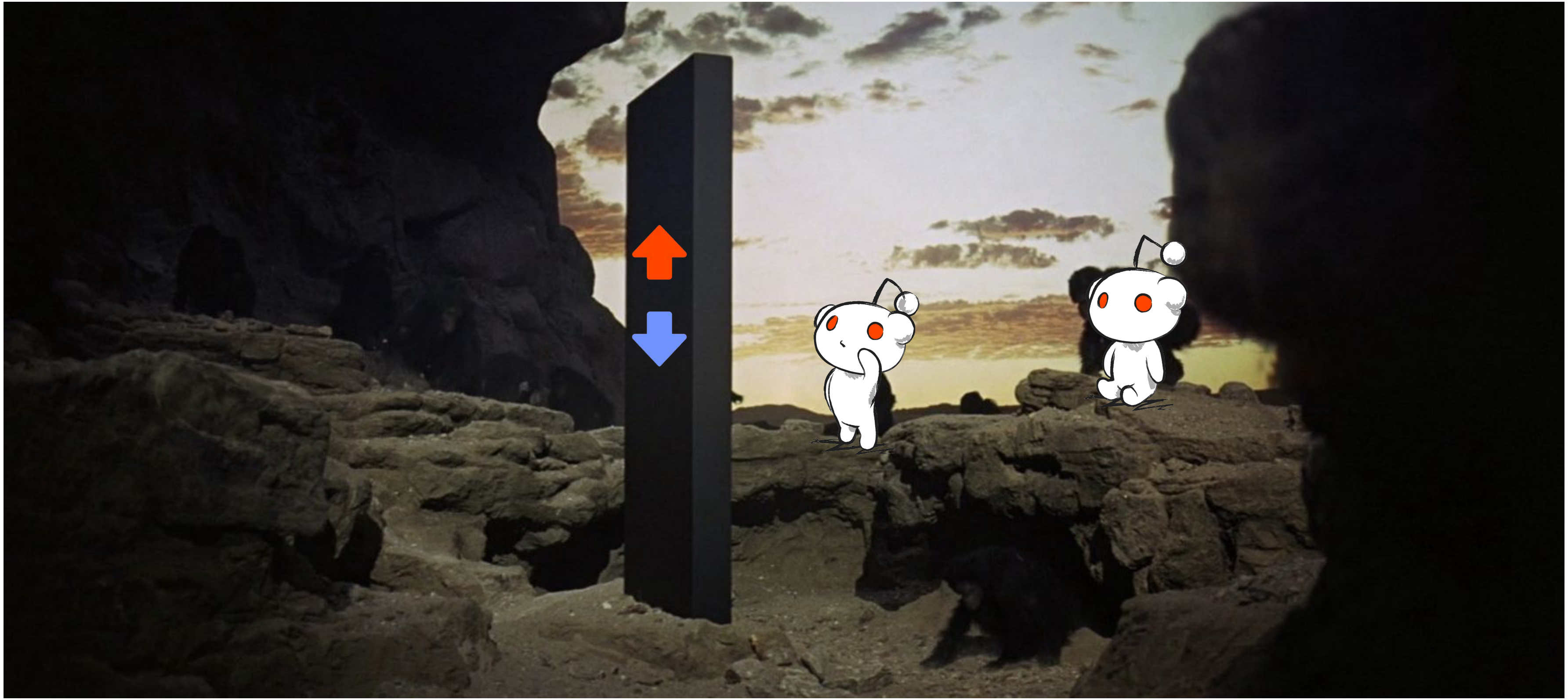
1. Avoid re-posting links for at least a month
2. No primarily marketing/sales-oriented posts
3. The CNCF Code of Conduct applies

By the numbers

5th/20th	Alexa Rank (US/World)
500M+	Monthly active users
140K+	Communities
16M+	Posts per month
2.8B+	Votes per month



~~Long ago~~ 2016

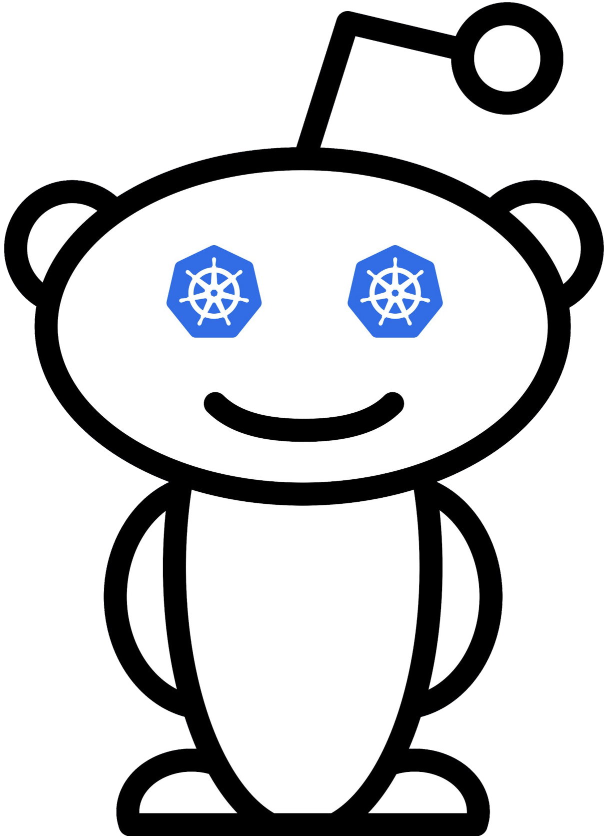


Rapid growth

Enter: Microservices

Microservices without empowerment

Enter: InfraRed



InfraRed Development and Launch

Org-wide Onboarding

Org-wide Onboarding: The Plan

1. Introduce a launch schedule with pre-allocated dates.
2. Allocate two weeks of hands-on time per service launch.
3. Expect to spend at least half of that in training.
4. Address gaps in docs, process, and automation as we go.
5. We’ll eventually reach an organizational critical mass.

Org-wide Onboarding: What went well

Critical mass is real!

Org-wide Onboarding: What went well

Empowered service owners.

Org-wide Onboarding: What didn’t go well

Tough sledding until critical mass.

Org-wide Onboarding: What didn’t go well

Uneven post-launch support for new service owners.

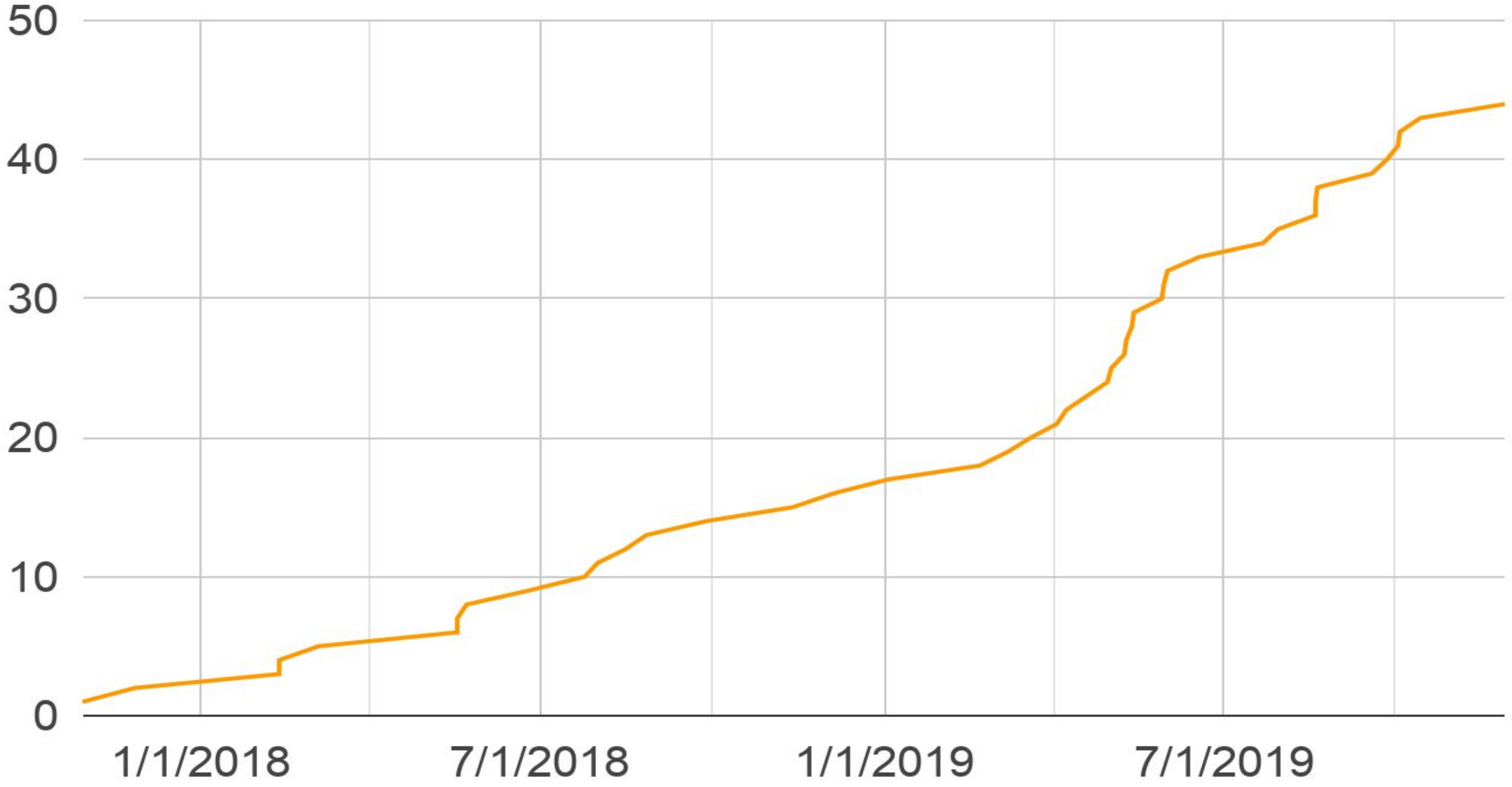
Org-wide Onboarding: What we'd do differently

Build central SRE org sooner.

Org-wide Onboarding: Where we are today

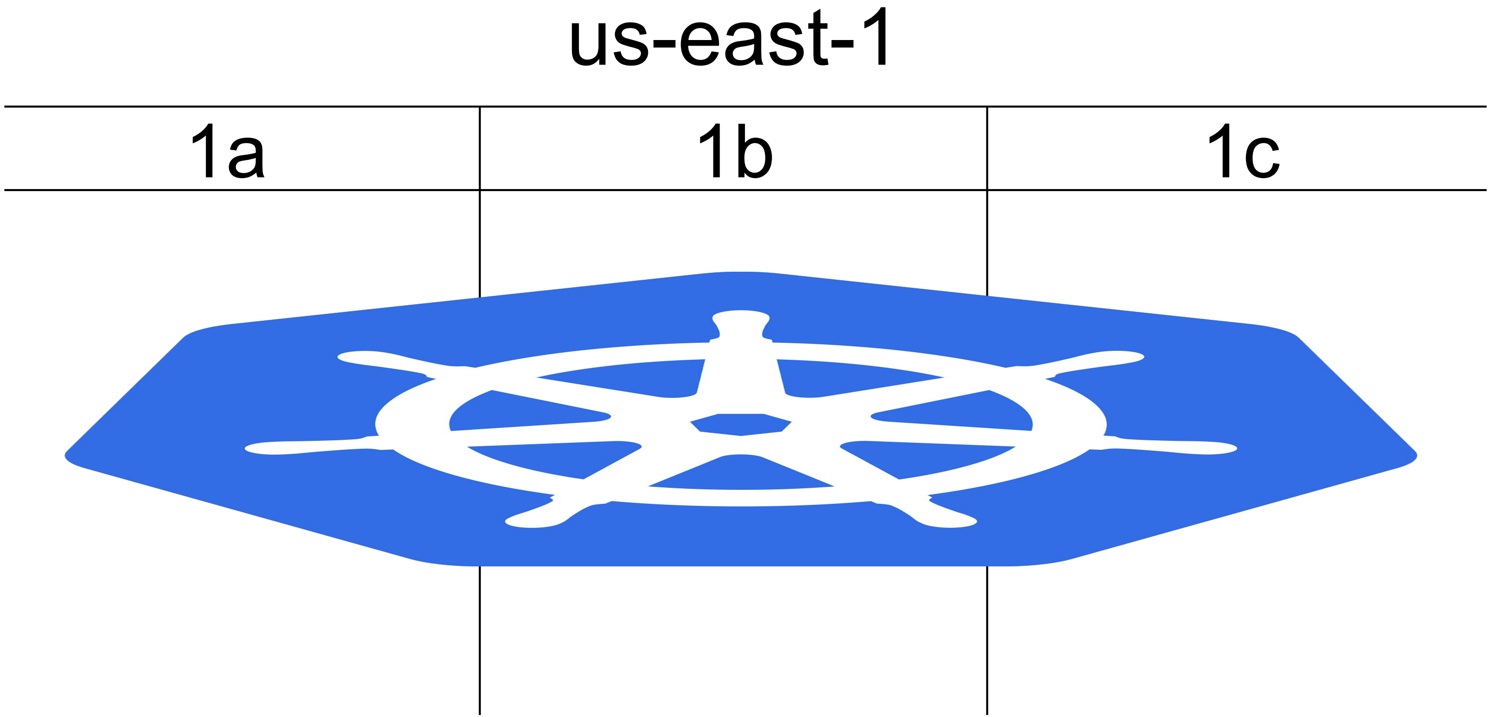
- Most engineering teams have adopted InfraRed/Kubernetes.
- Embedded SREs in most prolific service-owning divisions.
- Support load is manageable.
- More work to be done on launch automation, docs, tooling, and training.

Reddit-authored services in InfraRed/Kubernetes over time



Managing our Clusters.

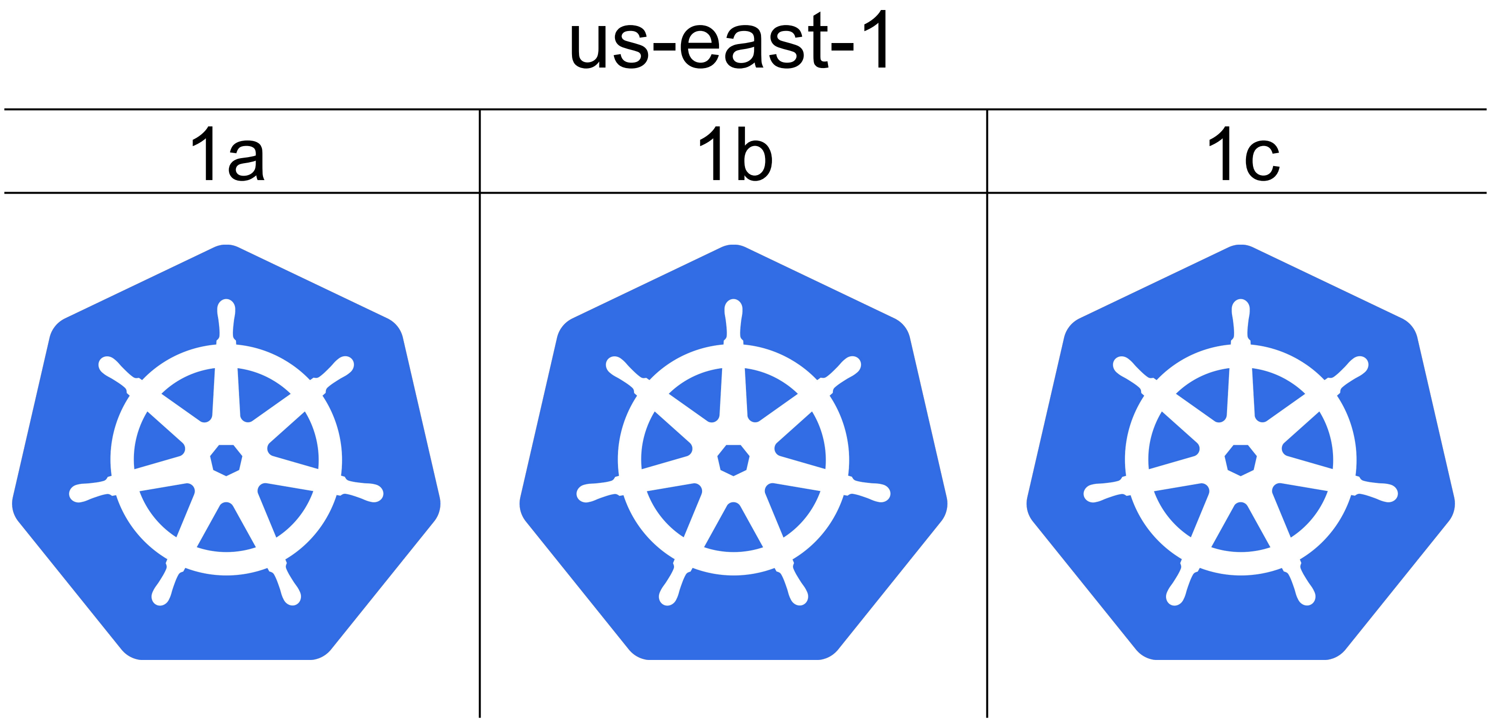
Managing our clusters: Prior reality



us-east-1

1a	1b	1c

Managing our clusters: The Plan (Cluster groups)



us-east-1

1a

1b

1c

Managing our clusters: What went well

Mirrored clusters have prevented outages.

Managing our clusters: What went well

Cost and latency savings from silo’d AZs.

Managing our clusters: What didn’t go well

More clusters, more admin overhead.

Managing our clusters: What we’d do differently

Start with single-AZ clusters for critical environments.

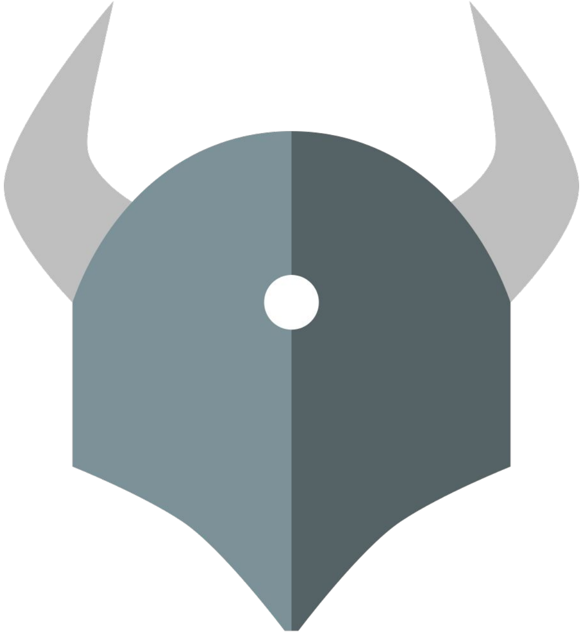
Managing our clusters: Where we are today

- Essential environments use the cluster group model.
- Everything else stays multi-AZ.
- We operate 19 clusters.
- Spinnaker takes the drudge work out of multi-cluster deploys.

Cluster Policy

Cluster policy: The Plan

1. Needed to protect against mistakes or malicious activity.
2. Ex: Missing cost tags, duplicate Ingress hosts.
3. RBAC can’t catch these kinds of issues.
4. Planned to use Open Policy Agent to introduce guard rails.



```
# Deny external-facing load balancer whose name isn’t whitelisted.
deny[explanation] {
    input.request.kind.kind == "Service"
    input.request.operation == "CREATE"
    loadbalancer.is_external_lb
    namespace := input.request.namespace
    name := input.request.object.metadata.name
    not whitelisted[{"namespace": namespace, "name": name}]
    explanation = sprintf(
        "Service %v/%v is an external load balancer but has not
         Been whitelisted", [namespace, name])
}

# Simple white list.
whitelisted[{"namespace": "retail", "name": "payment_lb"}]
```

Cluster policy: What went well

Nice balance of empowerment and safety.

Cluster policy: What didn’t go well

Death spiral on one of our busiest clusters.

Cluster policy: What we’d do differently

Test OPA on a stressed control plane.

Cluster policy: Where we are today

- Open Policy Agent runs on all of our clusters.
- Policies are in place for the scariest of possibilities.
- OPA’s monitor mode has been handy for audits and experimentation.

YAML

YAML: The Plan

1. Needed to minimize service owner YAML drudgery.

2. Wanted to avoid premature abstraction.

3. Decided to stay close to the community and use Helm.

4. Aimed to auto-generate Helm Charts for Reddit (baseplate) services.

5. Planned to pass Helm Charts to Spinnaker to be rendered/deployed.

YAML: What went well

Auto-generated Helm was a great starting point.

YAML: What went well

Spinnaker has complemented Helm well.

YAML: What didn’t go well

Overhead of managing divergent Helm Charts.

YAML: What we’d do differently

Resource generator for Baseplate services instead of Helm.

```
conf = release.config(
    name = app_name,
    filename = "test.yml",
)
app = release.app(
    name = app_name,
    config = conf,
    ports = [http],
    requests = bp.resources(cpu = "500m"),
)
http_svc = release.service(
    name = app_name,
    ports = [http],
)
```

YAML: Where we are today

- Good understanding of usage cases. Best practices established.
- Baseplate makes our services look and act the same.
- Starlark-powered resource generator instead of charts or megachart.
- Helm Charts will stick around for non-Reddit services.

Dev Environment

Dev Environment: The Plan

1. Needed a dev environment that looked like production.
2. Wanted to use the same Helm Chart for local dev, staging, and prod.
3. Needed to be able to develop multiple microservices in parallel.
4. Something something service dependencies.
5. Planned on using Skaffold paired with local minikube.

Dev Environment: What went well

When everything worked, it was nice.

Dev Environment: What didn’t go well

Constant breakage and flakiness.

Dev Environment: What didn’t go well

Multi-service dev woes.

Dev Environment: What we’d do differently

Focus on developing against remote clusters.

Dev Environment: Where we are today

- Shifted from Minikube to development against a remote dev clusters.
- Swapped Skaffold out for Tilt.
- Starlark resource generator instead of service Helm Charts.
- Master branch of dependencies auto-deploys to dev cluster.
- Multi-service dev is possible with a minor config tweak.

Next challenges

Next challenges

- Fully self-serve service launches.
- Istio all of the things.
- Flesh out of dev environment story.
- Start optimizing for cost and density.
- Continue to build out our SRE organization.

Presenter Info + Resources

- Greg Taylor - Reddit Infrastructure
- /u/gctaylor
- @gctaylor

- reddit.com/r/kubernetes
- redditblog.com/topic/technology

reddit.com/jobs