Kubernetes at Reddit: An Origin Story

Greg Taylor – EM, Reddit Infrastructure
/u/gctaylor


What is Reddit?

Something for everyone

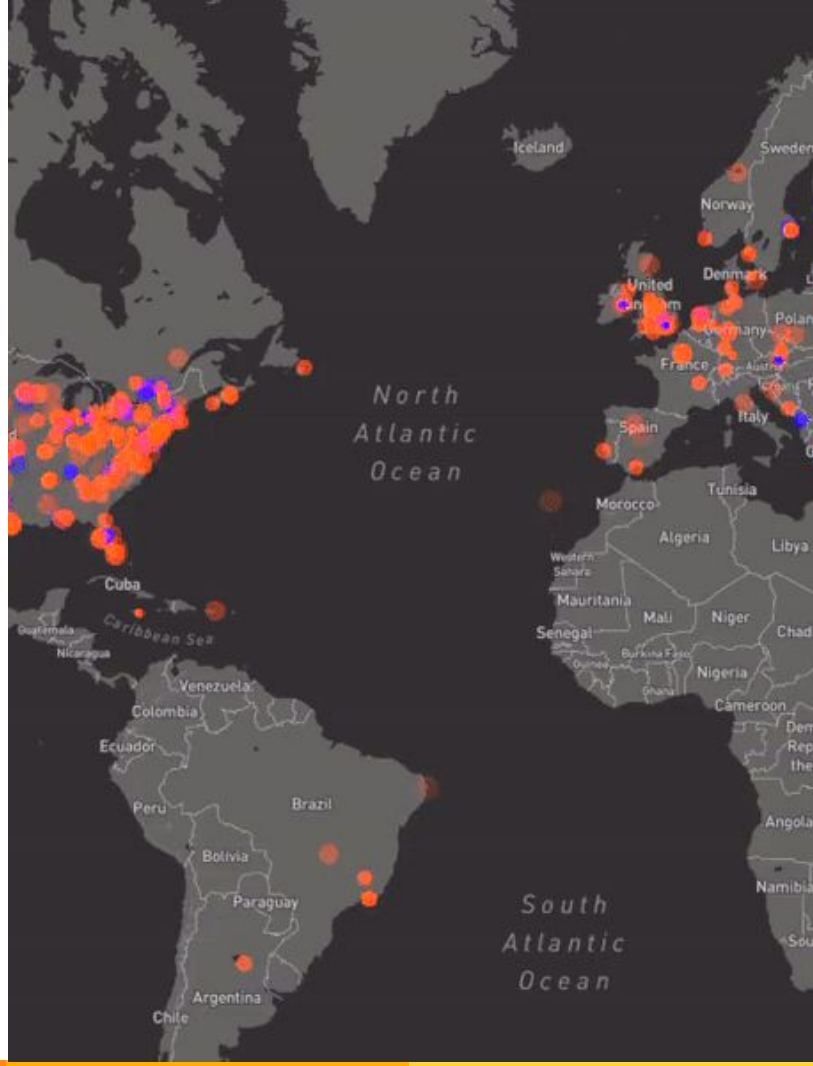
5th/20th Alexa Rank (US/World)

400M+ Monthly active users

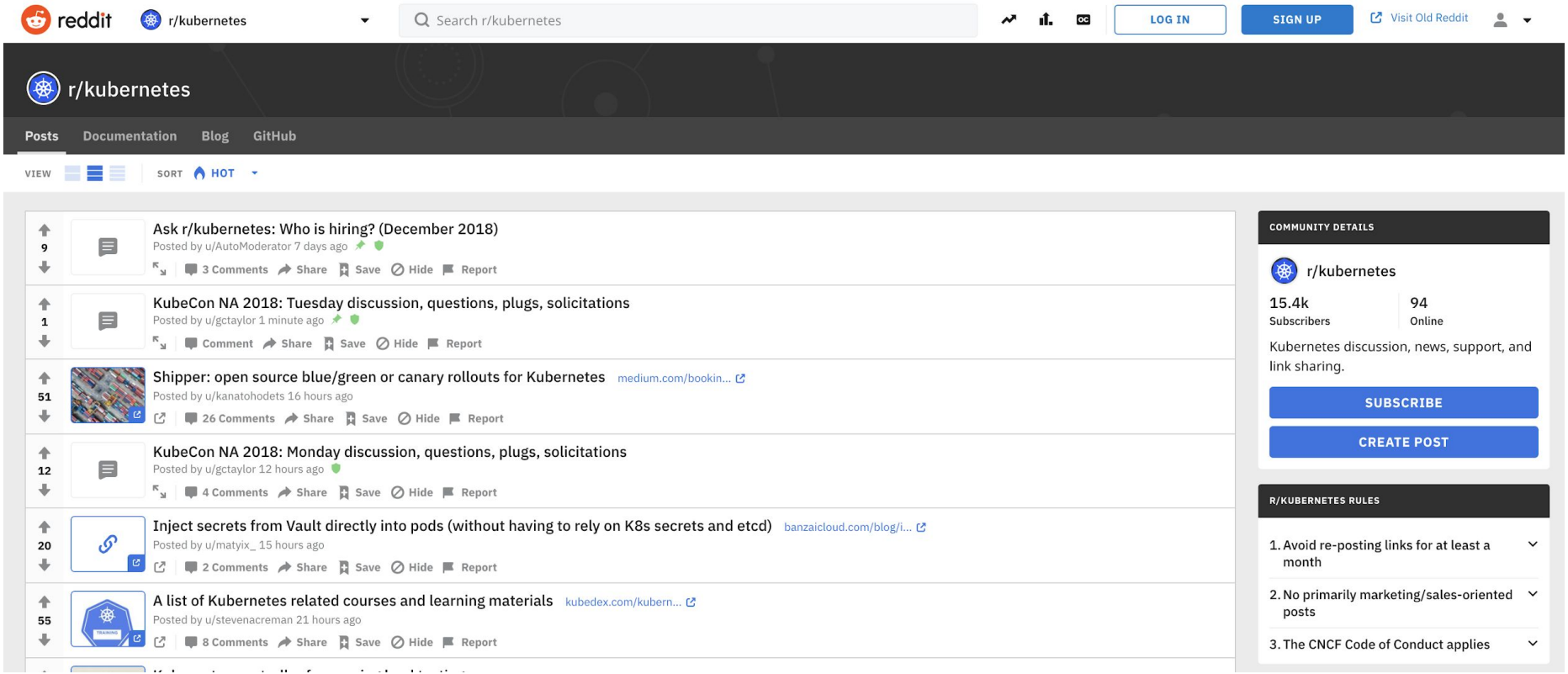
140K+ Communities

12M+ Posts per month

2B+ Votes per month

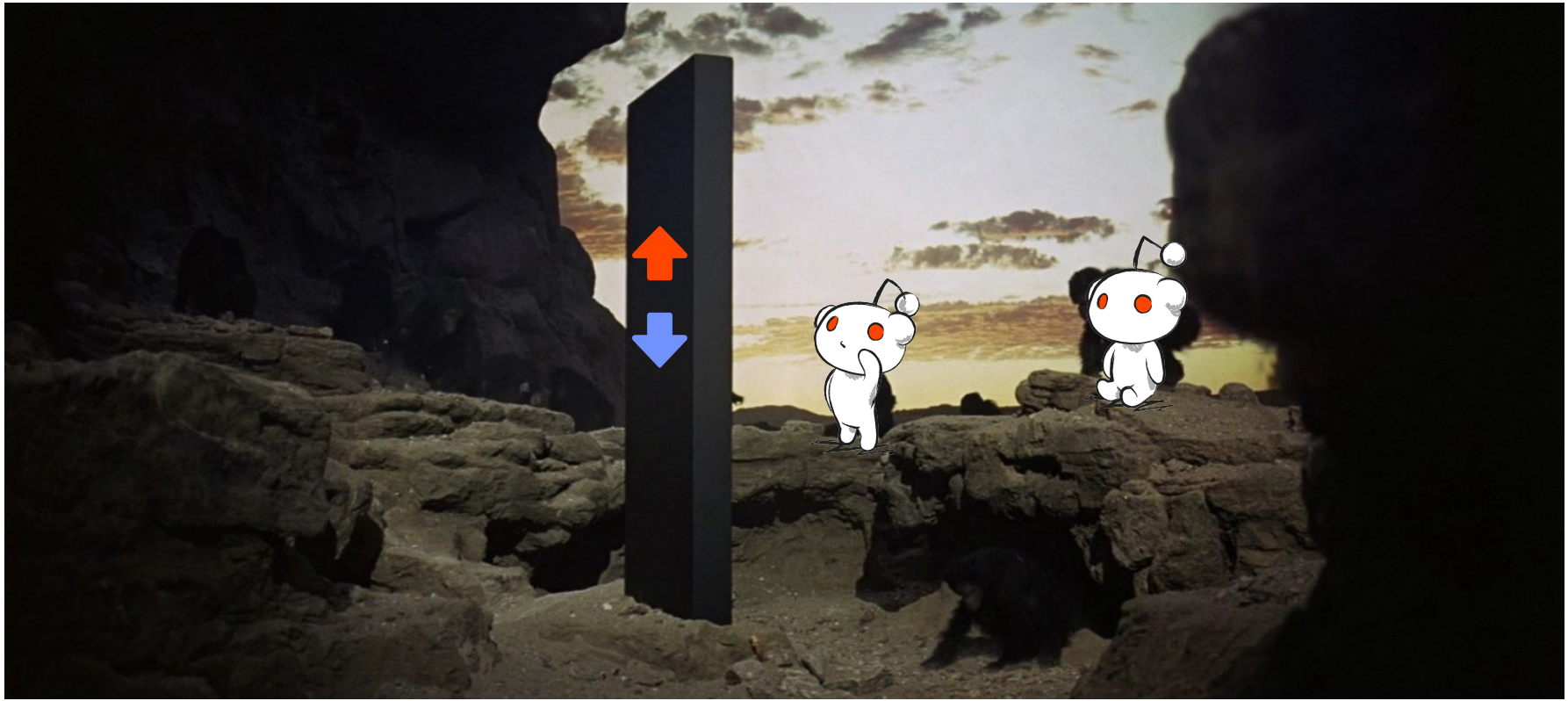


Example: /r/kubernetes




Get on with it!

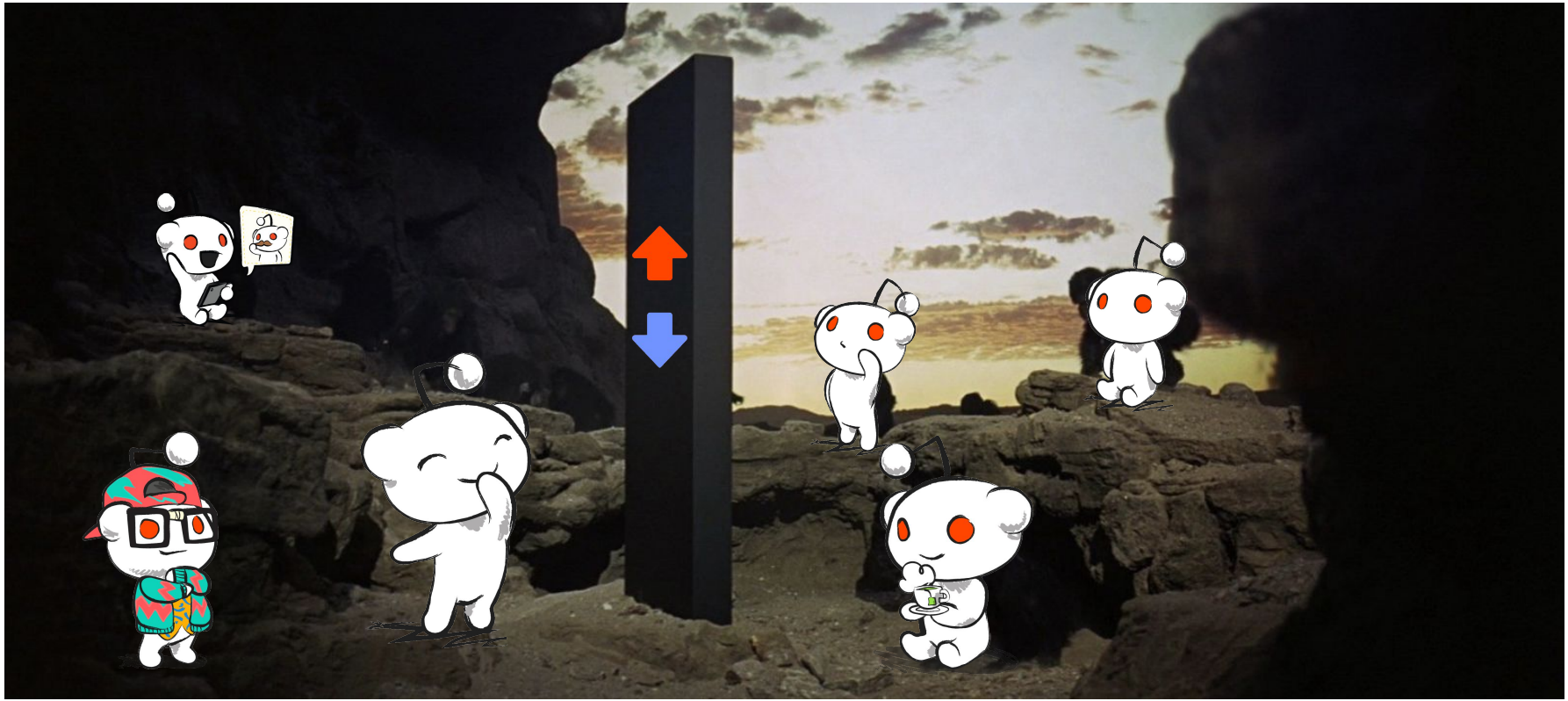
Welcome to early 2016



2016 - The Infrastructure Team

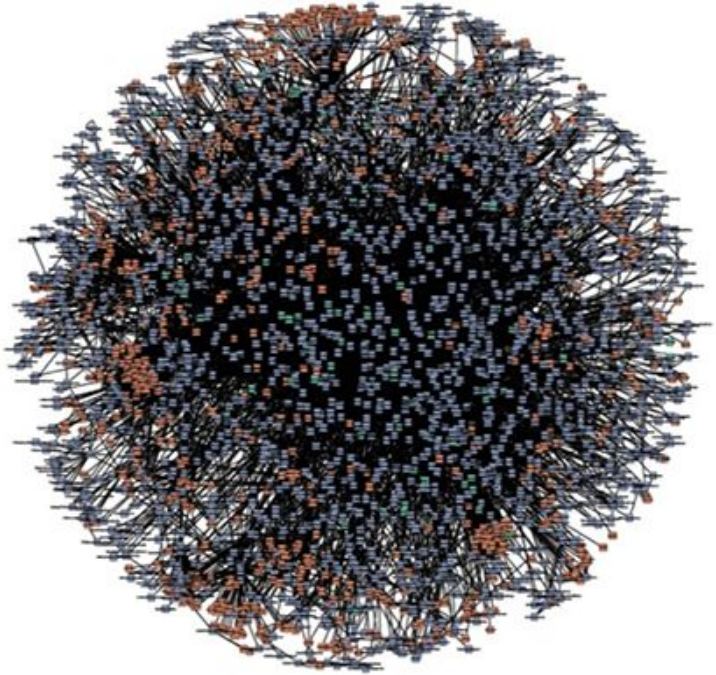
- Provisioned and configured all infrastructure
- Operated most of our systems
- Responded to most incidents

Mid-2016 and onward: The Great Embiggening



Determining the path forward

Service-oriented Architecture (SOA)



amazon.com

NETFLIX

Growing pains: Infra team as a bottleneck

- **Problem: Eng teams too dependent on Infra team**
  - Service provisioning
  - Ongoing operation
  - Debugging and performance work

- **Short-term “solution”: Train and deputize infrastructure-oriented teams**
  - Allows for more self-sufficiency
  - Only possible for some teams!

One size fits ~~all~~ some

Not all teams want to operate the full stack for their service

My toaster runs Docker

Don’t make me infrastructure!

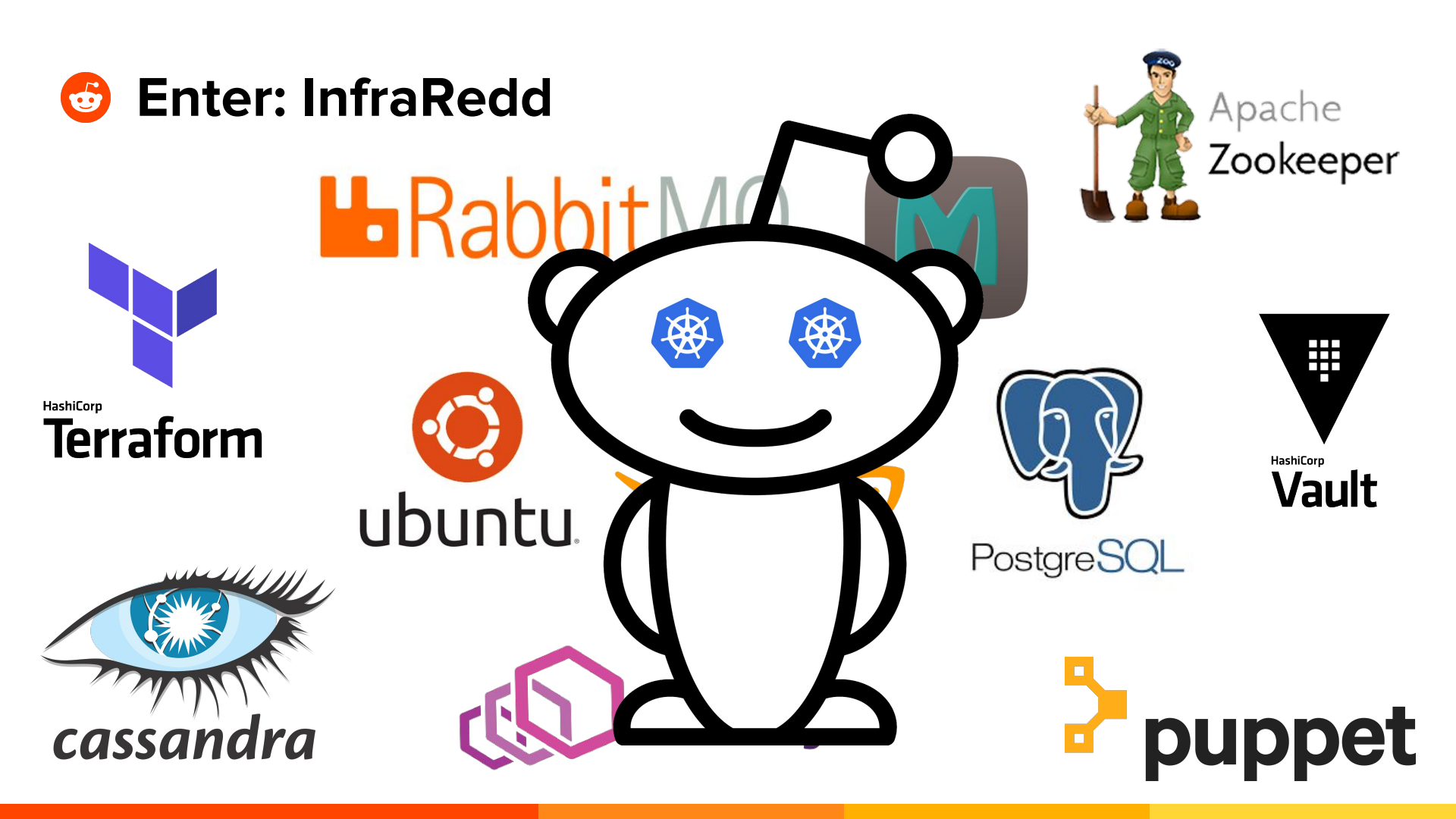
What do the engineering teams REALLY want?

Service ownership

A service owner is empowered and expected to:

- **Develop** their service from start to finish
- **Deploy** their service early and often
- **Operate** their service

Enter: InfraRedd

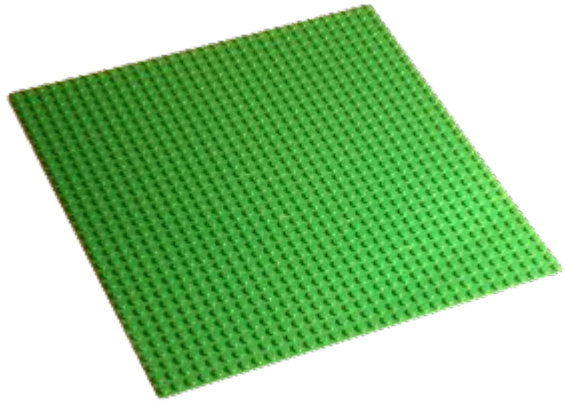


A service owner should be able to
develop, deploy, and operate their service.
Regardless of engineering background

Develop: Consistency in services

Regardless of language/toolset, the “shape” of each service should be consistent:

- RPC protocol
- Secrets fetching
- Metrics
- Tracing
- Log output format



Baseplate: https://baseplate.readthedocs.io

Develop: Service creation

Auto-generate starter material:

Service sources

- Python/Go/Node service stub
- Dockerfile
- CI configs

Helm Charts

- Friends don’t let friends write YAML!

Develop: “Local” development

Development is facilitated by Skaffold.

Major considerations:

- Accessible to those without deep Kubernetes experience
- As similar to production as possible
- Re-use our standard Charts + images
- Must not exhaust standard dev laptop’s resources

SKAFFOLD

A service owner should be able to develop, deploy, and operate their service.
Regardless of engineering background

Deploy: Tests, builds, deploys

- **CI runs through Drone**
  a. Tests
  b. Artifact builds

- **Spinnaker handles our deploys**
  a. Standardized pipeline templates
  b. Renders Helm Charts
  c. Applies rendered YAML
  d. Uses V2 Kubernetes provider

Deploy: Standard staging/production flow

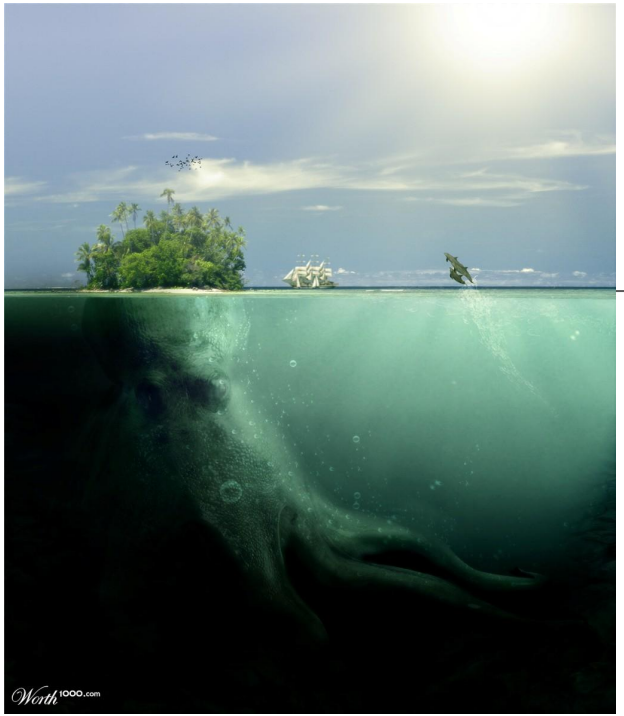
Staging and production deploy flow:

1. Developer pushes to canonical repo
2. Tests and builds run in CI
3. One of two flows are offered:
   a. CI triggers a deploy
   b. Eng manually triggers a deploy

A service owner should be able to develop, deploy, and operate their service.

Regardless of engineering background

Operate: Explicitly defined expectations



Service owners

- Learn some Kubernetes basics
- Deploy and operate own services

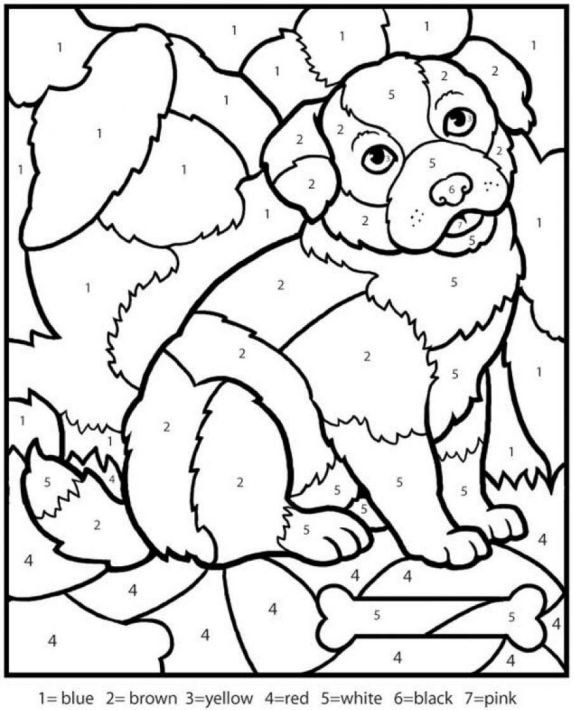
Infrastructure team

- Keep the Kubernetes clusters running
- Provision AWS resources, caches, DBs
- Support and advise Product Users

Operate: Paint-by-numbers

Enabling service ownership for all backgrounds:

- Take the guesswork out
- Document all the things
- You want to do X? Here's a guide for that
- Must be supported by training!



Operate: Service owner permissions

- Service owners auth via OpenID Connect
- RBAC policies are group-based
- Namespace per service
- Service owners have full access to their namespace(s)

Operate: Guardrails

Things that prevent or minimize damage

- Resource limits and Network Policies
  - Built into Kubernetes
- Throttling and circuit breaking
  - Envoy + Istio
- Object and Image policies
  - Open Policy Agent
- Finely scoped RBAC
  - Open Policy Agent

Operate: Oh no!

Something exploded!

Service owner:

1. Paged for service incident
2. Diagnoses + resolves issue
3. Can summon Infra if needed

Infrastructure team:

1. Paged for cluster issues
2. Those *never* happen. Yep.

Operate: Observe, Diagnose, Resolve

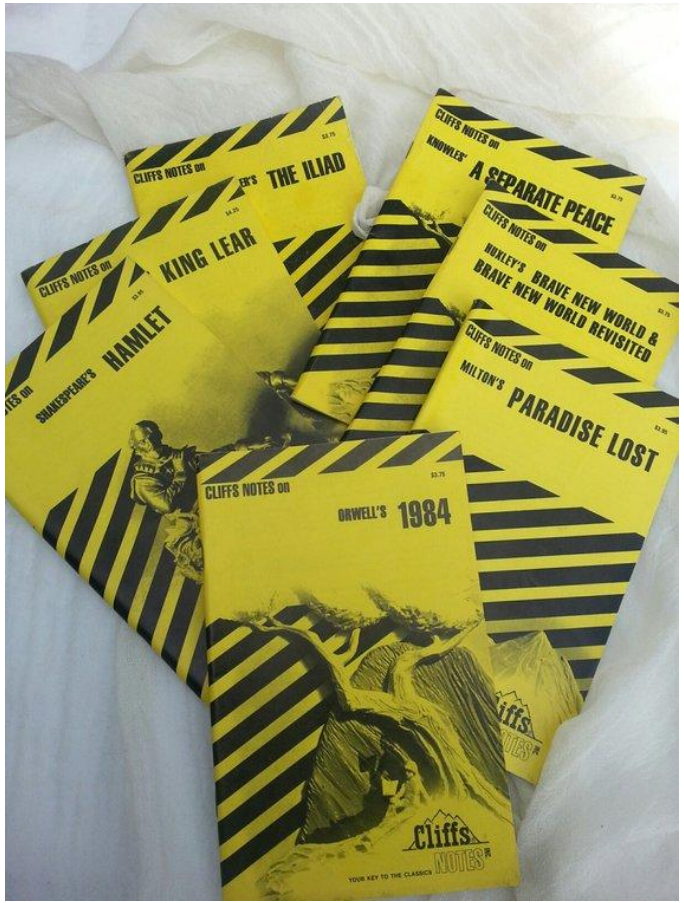
Observability *by default*:

- Metrics
  - Wavefront
- Alerting
  - PagerDuty
- Tracing
  - Zipkin
- Exception/error tracking
  - Sentry
- Central logging and analysis

Operate: Recap

Service owners have:

- Explicitly defined responsibilities
- Enough access to own their services
- Guardrails to prevent+limit damage
- Tools needed to respond to and diagnose issues



What does all of this buy us?

A service owner ~~should be~~ is able to develop, deploy, and operate their service.
Regardless of engineering background

Infra team: From Operators to Enablers

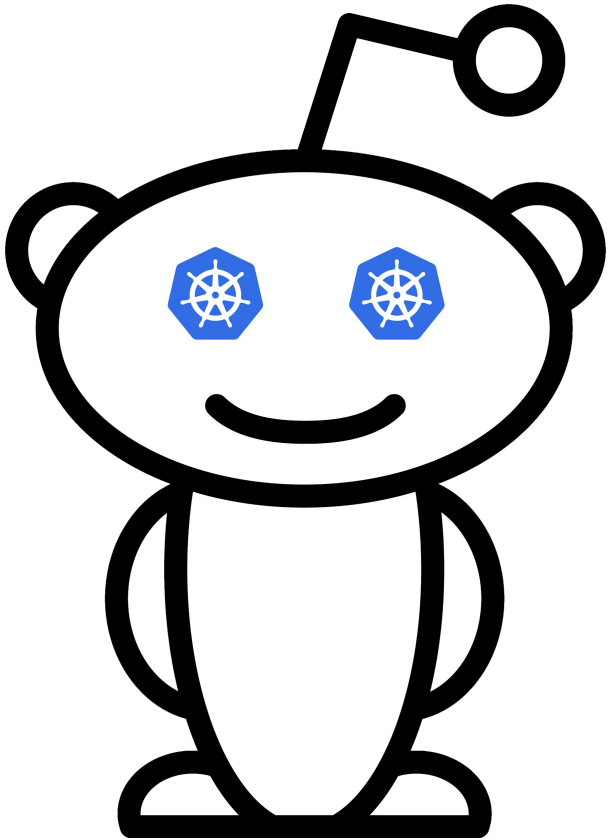
Kubernetes at Reddit

7 Kubernetes clusters

~30% Of our Engineering teams

~20 Production services

10-20 Deploys a day



New services on Kubernetes by default in Q1!

Closing Remarks

reddit.com/jobs

Presenter Info + Resources

- Greg Taylor - Reddit Infrastructure
- /u/gctaylor
- @gctaylor
- github.com/gtaylor

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