KubeCon | CloudNativeCon

North America 2024

Evolving Reddit's Infrastructure

Harvey Xia and Karan Thukral
Compute Infra @ Reddit

Agenda

01 Reddit Infrastructure in 2022

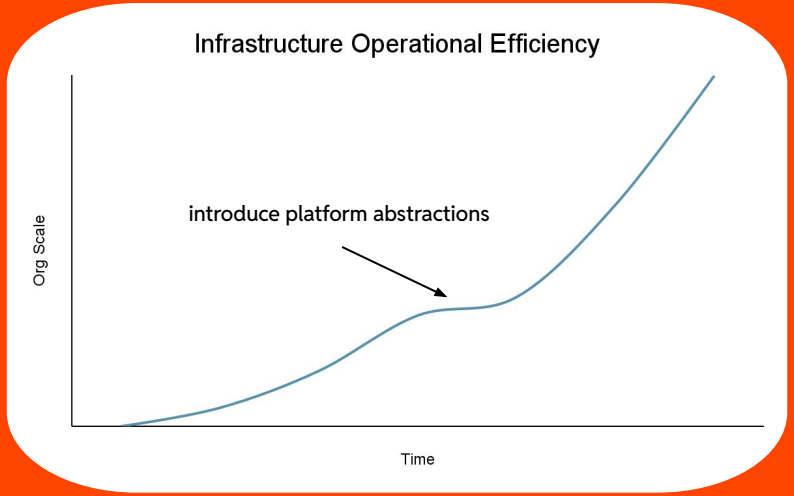
02 Platform Foundations

03 Results and Future Directions

04 Questions

reddit

tldr; When companies reach a certain maturity, they need platform abstractions to operate efficiently, especially as they grow.

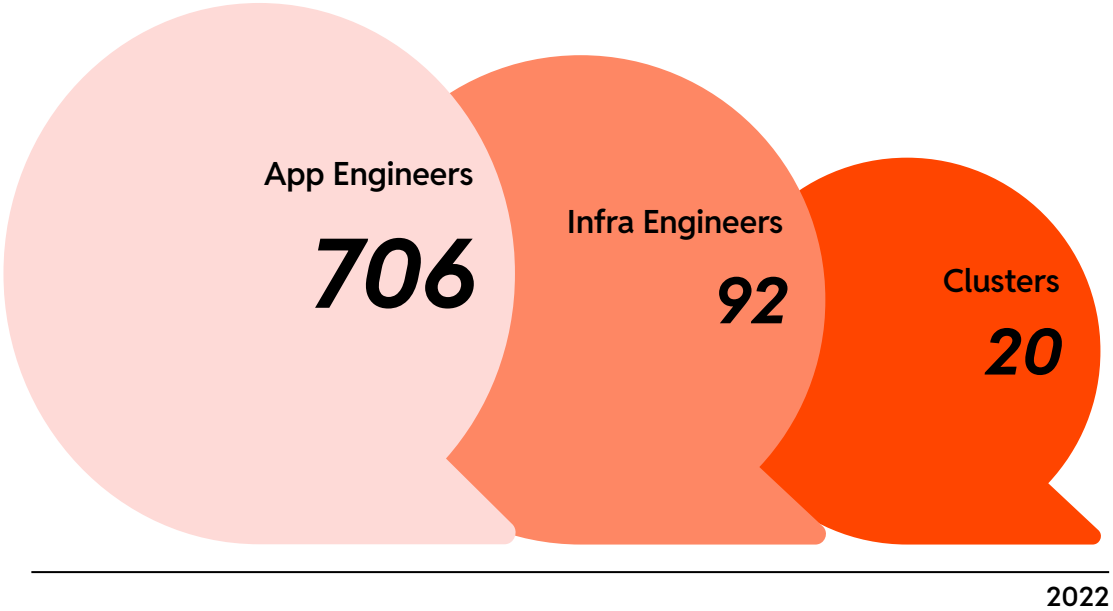


01

Reddit Infrastructure in 2022

State of Reddit in 2022

- Approaching IPO
- Expansion of serving stack to multi-region
  - Roadmap to expand… globally
- Growth of Ads and ML



Case Study: Kubernetes Namespaces

Namespaces and RBAC for applications

Copy and paste YAML

- Unfamiliar tooling
  - Helm charts or Kustomize manifests
- Unfamiliar concepts
  - Namespaces, Service Accounts, Roles, RoleBindings

Wait for Infra Review

- Blocking for up to 24 hours
- Subtle errors are often uncaught by the reviewer

Cross Fingers 🤞

- CI failures require Infra expertise to debug
- Failures can lead to waiting for another 24 hours

☹️

😓😡

Exposure to infra complexity

Blocking, white-glove review process

Case Study: Legacy Kubernetes Clusters

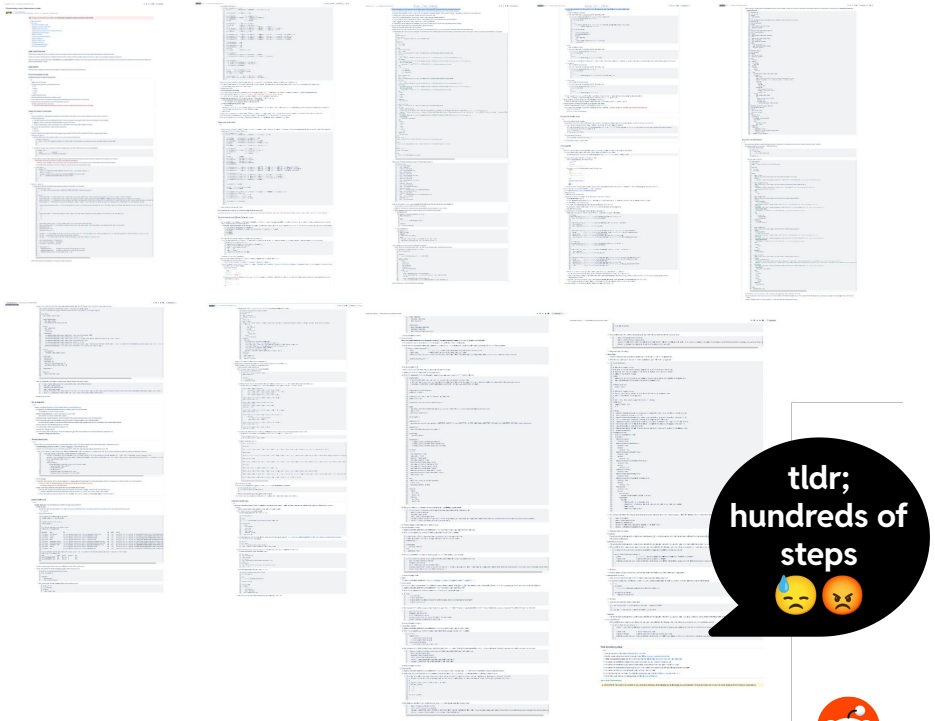
Artisanally crafted clusters

Lifecycle Management

- 30+ hours to create new clusters
- Dangerous, in-place cluster upgrades
- No standard process for decommissioning

Configuration Sprawl

- O(N) work for fleet-wide configuration
- Bespoke specialization for each cluster
  - eventually become “haunted”
  - e.g. Pi-Day Outage

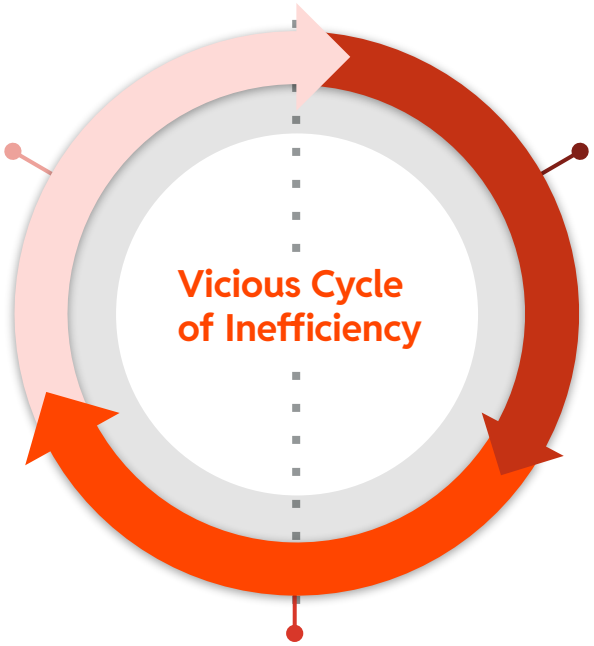


App Engineers

- Growing organizational demands
- New use cases
- New technology

Infra Engineers

- KTLO dominates engineering capacity
- Reactive, fire-fighting mindset



- White glove support for app engineers
- Manual workflows for managing infrastructure
- No standardization → haunted infrastructure

02

Platform Abstractions

Principled Platform Abstractions

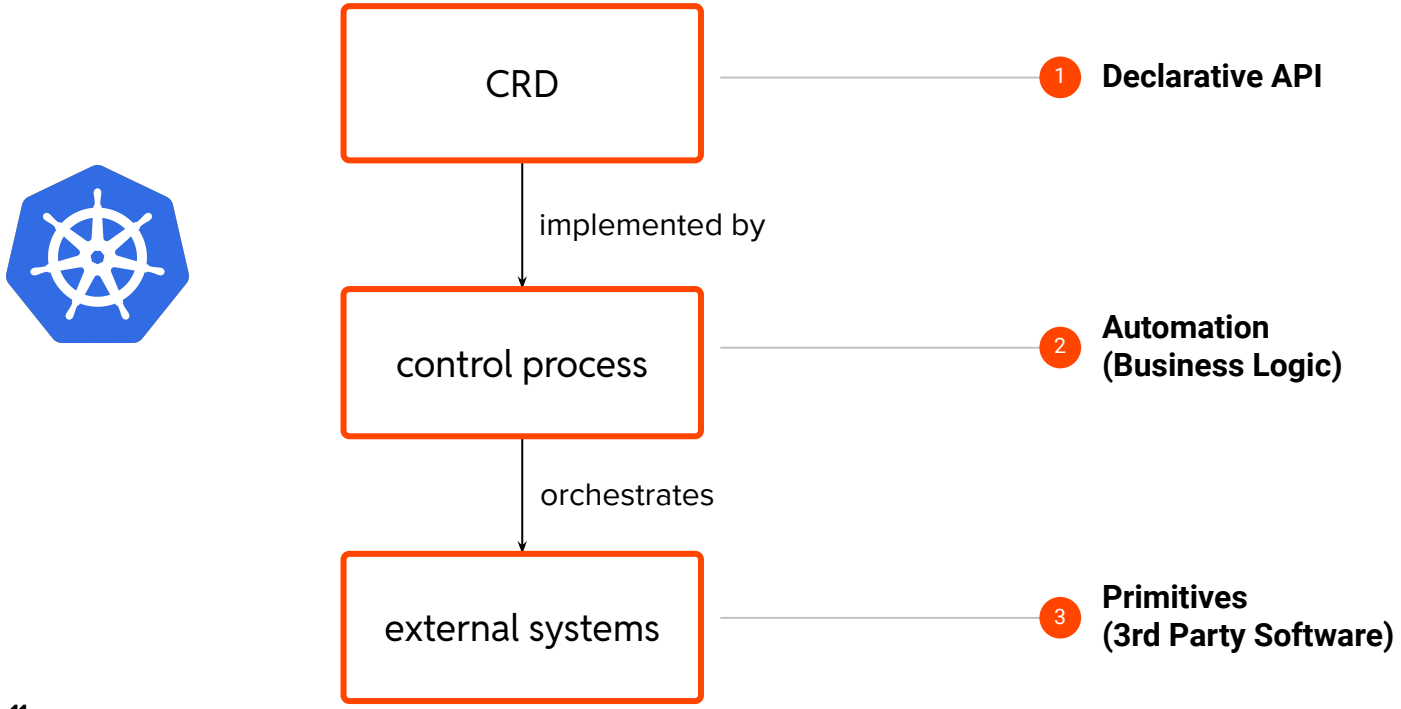
"Abstraction"
- Hides underlying complexity
- Separation of user concerns from implementation concerns

"Platform"
- Ecosystem of composable tools
- Ergonomic UX
- Safe and reliable
- Scalable (computational load + administration)

"Principled"
- Opinionated, not generic
  - Designed to solve specific user problems
  - Implements domain best practices
- Standardized
  - One problem, one solution

High Level Approach

Declarative APIs backed by Kubernetes control processes



Traditional IaC vs. k8s controllers

Pros

- Cheap upfront engineering cost
- Simpler mental model
- High granularity of control

- Continuous self healing, no state drift
- Arbitrarily complex behavior → high level APIs
- Lifecycle transitions automated by code
- Interfaces are programmatically consumable

tldr;
we made this
tradeoff

Cons

- Can't model arbitrarily complex behavior
- Interfaces + state are not programmatically consumable
- No dynamic behavior
- "Fire and forget" → state drift

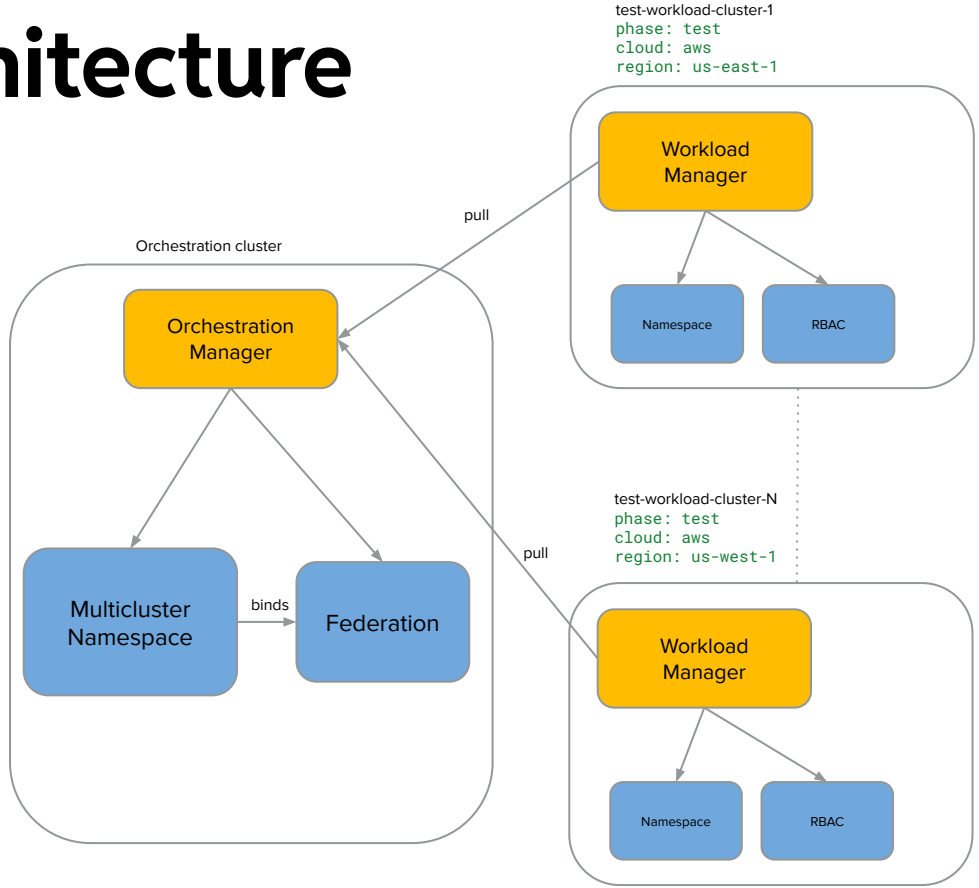
- Higher engineering cost
- More complex mental model
- Riskier failure modes

Orchestration Architecture

Multi-cluster Orchestration

- Orchestration cluster has centralized control over cluster fleet
- Fleet of workload clusters host apps
- Fleet-wide configuration via multi-cluster federation
- Orchestration designed to “fail static”

```
apiVersion: federation.infrared.reddit.com/v1alpha1
kind: Federation
metadata:
  name: all-test
spec:
  clusterSelector:
    labelSelectors:
      - matchLabels:
          "infrared.reddit.com/phase": "test"
```



3rd Party Software

Primitives Fronted by API Moats

FluxCD

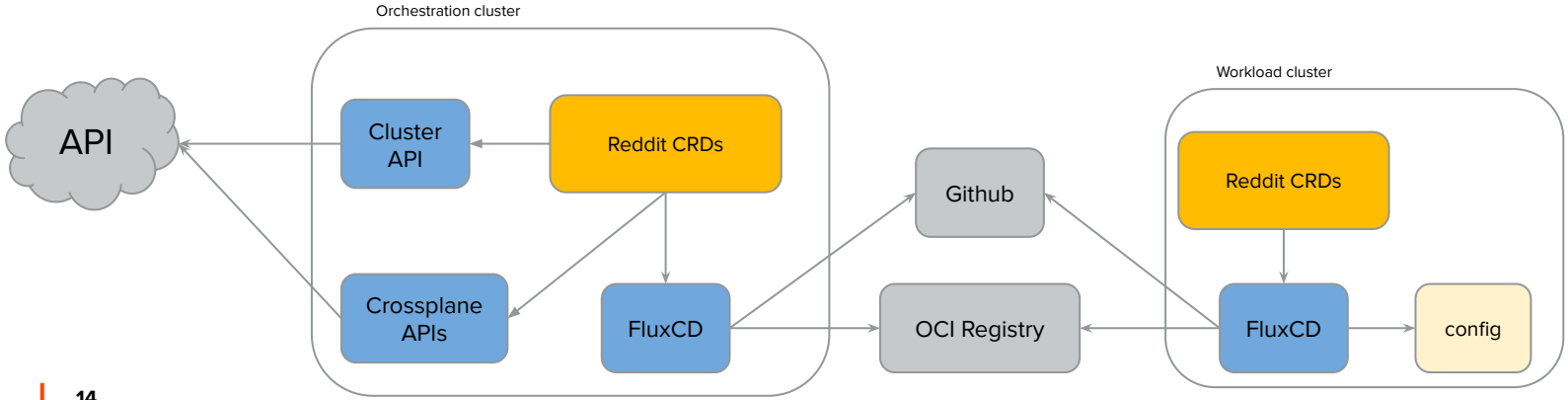
- Syncs config from source control to clusters
- Supports variety of sources (Git, S3, OCI)

Crossplane

- Kubernetes API into cloud vendors
- Support for various cloud vendors

Cluster API

- Kubernetes API for managing Kubernetes clusters
- Support for various cloud vendors



Case Study: Kubernetes Namespaces

Revisited

```yaml
apiVersion: app.infrared.reddit.com/v1alpha1
kind: FederatedRedditNamespace
metadata:
  name: my-new-app
spec:
  federationRefs:
    - all-prod
  redditNamespace:
    slackAlertsChannel: slack-chanel
    pagerdutyService: pd-service
    rbac:
      operators:
        - my-app-team
      readers:
        - other-service-team
    deploy:
      enabled: true
      repoAllowlist:
        - reddit/my-app-repo
```

Dynamically targets all production clusters

3rd party integrations

Human RBAC

Integration with deployment tooling

Case Study: Kubernetes Clusters

Revisited

```
apiVersion: cluster.infrared.reddit.com/v1alpha1
kind: RedditCluster
metadata:
  name: my-new-cluster
spec:
  cluster:
    managed:
      controlPlaneNodes: 3
      kubernetesVersion: 1.29.6
      networking:
        podSubnet: ${CIDR}
        serviceSubnet: ${CIDR}
      provider:
        aws:
          asgMachineProfiles:
            - id: standard-asg
              ref:
                name: standard-asg
                namespace: ""
          envRef: ${ENV_REF}
  labels:
    phase: test
    role: demo
  orchKubeAPIServerAddr: ${API_SERVER}
  vault:
    addr: ${VAULT}
    authPathOIDC: ${OIDC}
```

Control plane properties

Cloud provider properties

Integration with network environment

Integration with Hashicorp Vault

Achilles SDK

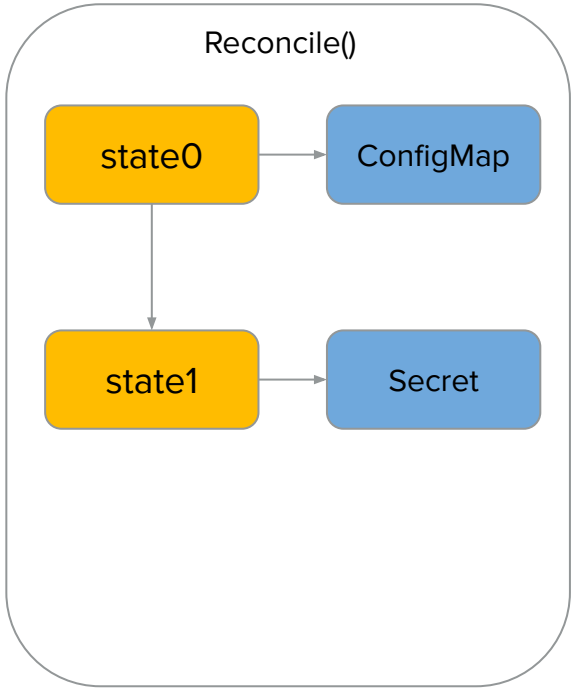
Lower the complexity of engineering Kubernetes controllers

- Built on top of controller-runtime
- Allow engineers to focus on modeling business logic and not controller internals
- Raises abstraction level by introducing conventions
  - Model `Reconcile()` as an FSM
  - OwnerRefs management
  - Status management
    - track managed child resources
    - status conditions tracking each FSM state
  - Finalizer management
  - Static tools for suspending/resuming
  - Opinionated logging and metrics

r/RedditEng

Achilles SDK

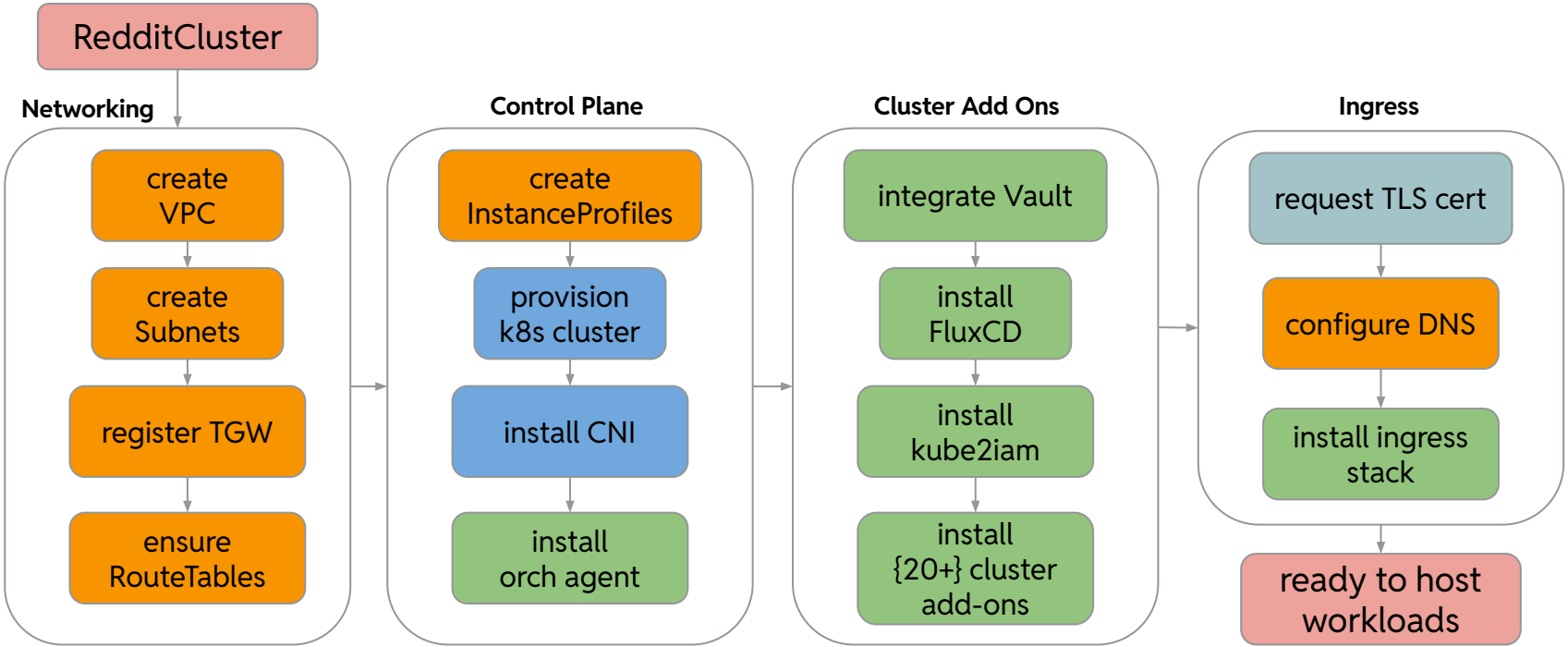
Model the control loop as a finite state machine




```
status:
  resourceRefs:
  - group: ''
    kind: ConfigMap
    name: ads-prod-usva-aws-1
    namespace: default
    version: v1
  - group: ''
    kind: Secret
    name: ads-prod-usva-aws-1-token
    namespace: ads-prod-usva-aws-1
    version: v1
  conditions:
  - type: State0
    message: State 0 succeeded.
    status: 'True'
  - type: State1
    message: State 1 succeeded.
    status: 'True'
...
```

Achilles SDK in Action

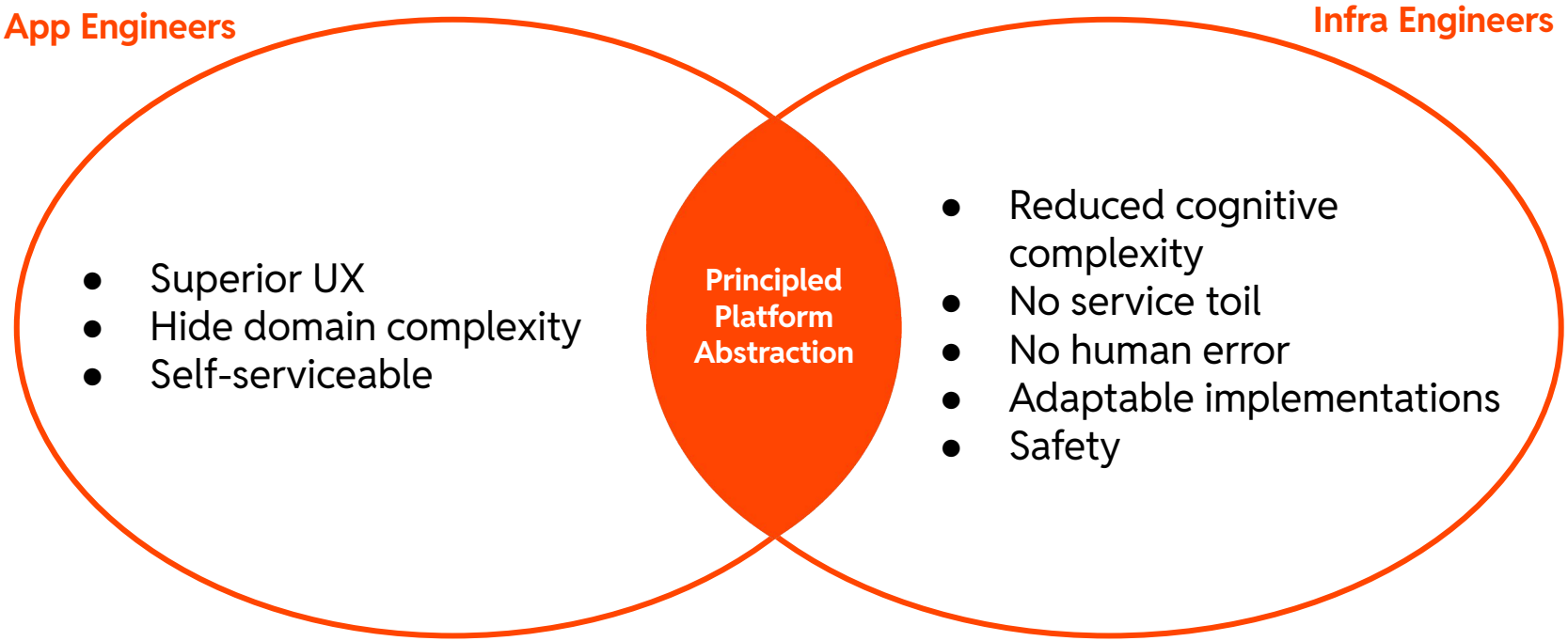
Taming Orchestration Complexity



03

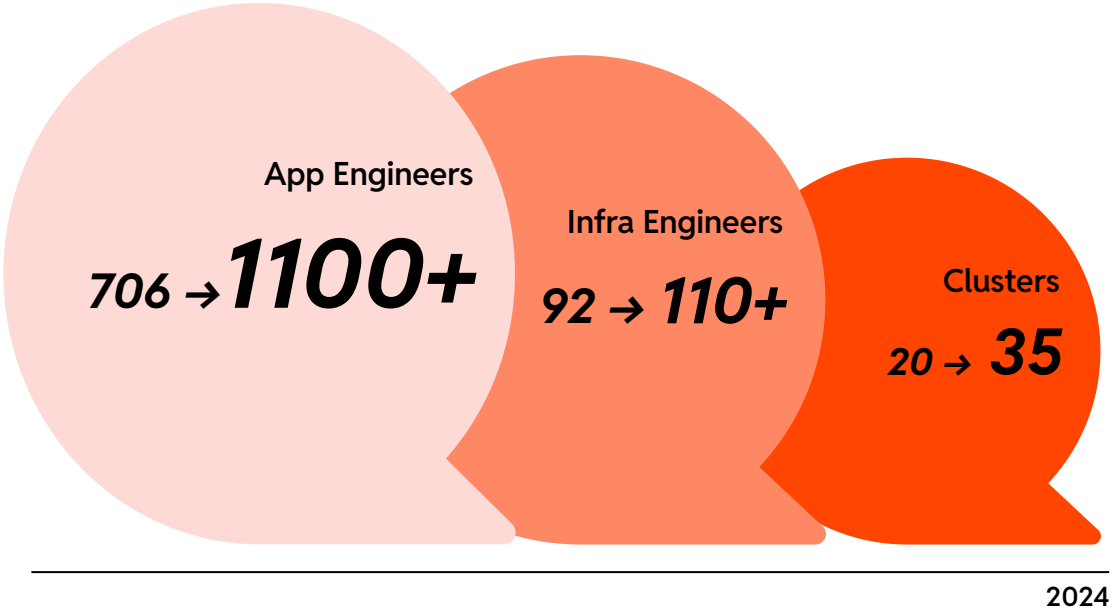
Results

A Better World for Everyone



State of Reddit in 2024

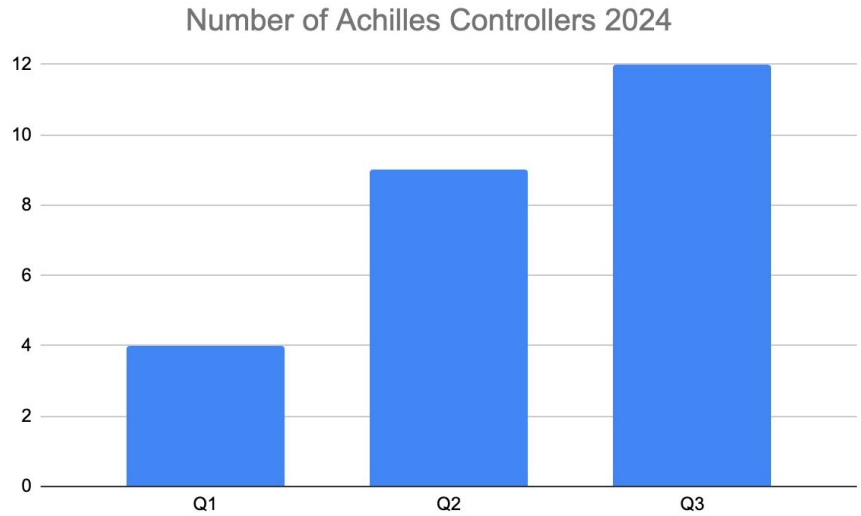
- App to infra engineer ratio grew from 7.6:1 → 10:1
- Kubernetes Clusters: 35
  - Projected growth to 100+



Achilles SDK Adoption

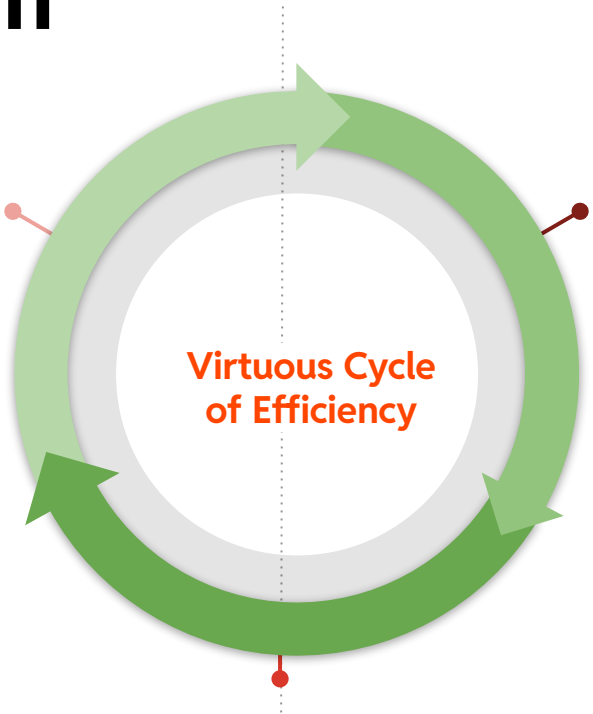
Empowering Infra Engineers

- Time to MVP: 2 weeks
- Number of controllers in production: 12
  - Managing infrastructure ranging from
    - Kubernetes clusters
    - Kubernetes Ingress stack
    - AWS Networking
    - Redis
    - Cassandra
    - Hashicorp Vault policies and roles



New Paradigm

Growing organization and product demands



Capacity for proactive, long-term engineering mindset

Safe, scalable, self-service interfaces.
Automation for complex, labor intensive workflows.

To automate or not to automate?

Automate

- Consolidated patterns
  - 80/20 rule
- Frequently used
- Highly complex

Don’t Automate

- Unconsolidated usage patterns
- Infrequently used
- Low complexity

tldr;

“When companies reach a certain maturity, they need platform abstractions to operate efficiently, especially as they grow.”

- Automation enables administrative and technical scale
- Platform abstractions empower both application engineers and infra engineers
- Kubernetes can act as a universal control plane

reddit

Thank you! Questions?