Build a Low-Cost, High-Performance Analytic Platform with Kubernetes and Open Source*

Robert Hodges
Altinity

*And ClickHouse, too!

A brief message from our sponsor…

Robert Hodges

Database geek with 30+ years on DBMS. Kubernaut since 2018. Day job: Altinity CEO

Altinity Engineering

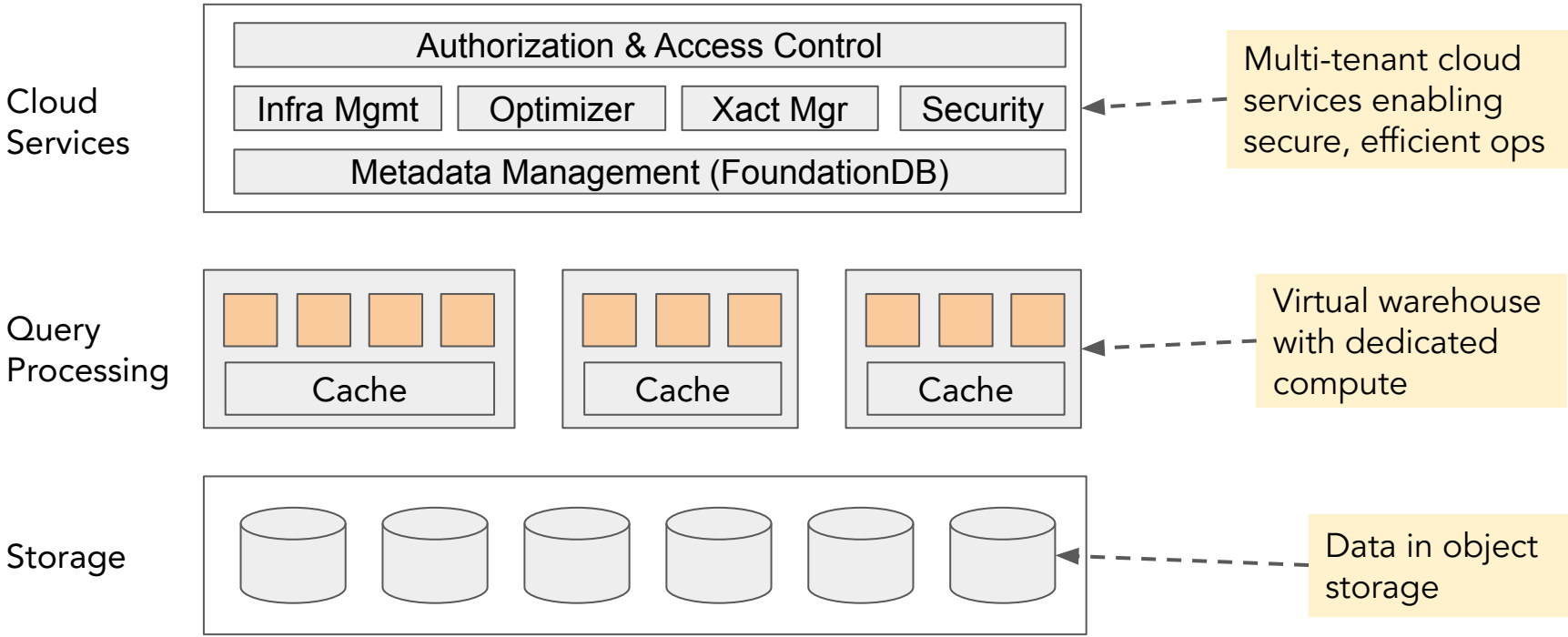
Database geeks with centuries of experience in DBMS and applications

ClickHouse support and services: Altinity.Cloud and Altinity Stable Builds
Authors of Altinity Kubernetes Operator for ClickHouse

How do cloud analytic databases work?

And what are some of the tradeoffs?

Snowflake database architecture



What's great about Snowflake?

- ✔ General purpose
- ✔ Serverless operation
- ✔ Handles large numbers of tenants with completely different applications
- ✔ Standards-compliant SQL
  - Complete implementation with ACID transactions
  - Sophisticated query optimizer
  - Efficient columnar storage with self-tuning partitioning and compression
  - Big table joins
- ✔ UI with built-in SQL editing and management

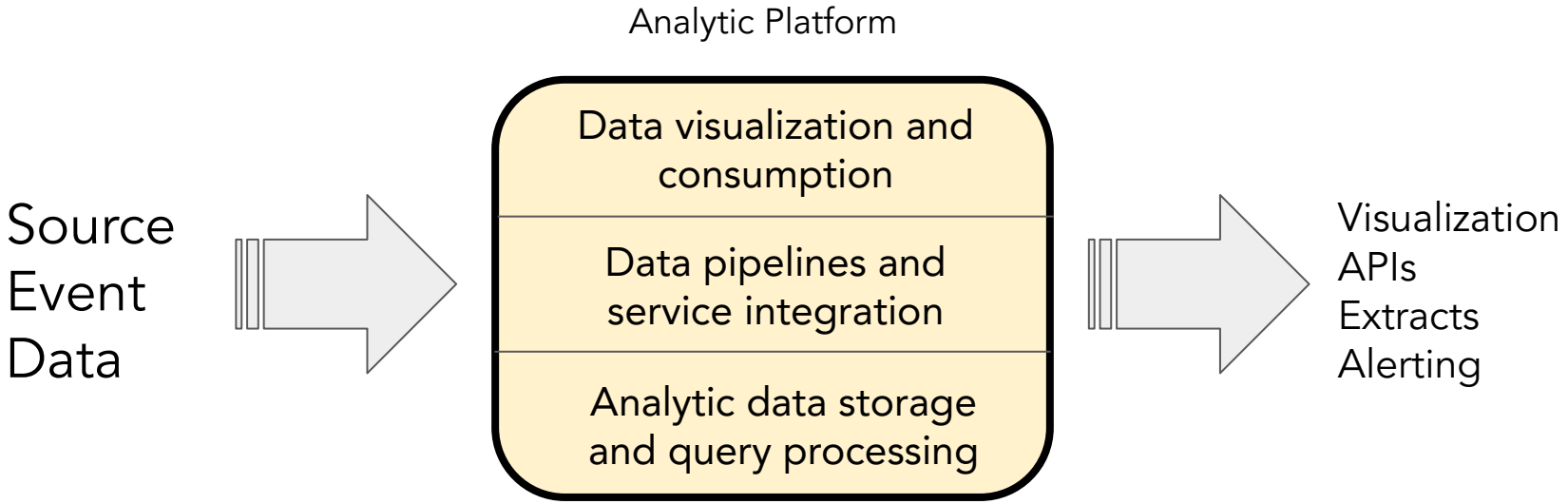
What Snowflake does not do

- ✘ Keep data in customer cloud account
- ✘ Minimize costs, especially for 24x7 analytics
- ✘ Deliver stable real-time response
- ✘ Handle SaaS user-facing analytics
- ✘ No vendor lock-in

Let’s design an analytic service with open source

Focus on a specific problem

Deliver a GDPR-compliant replacement for Google Analytics



First, scope the requirements

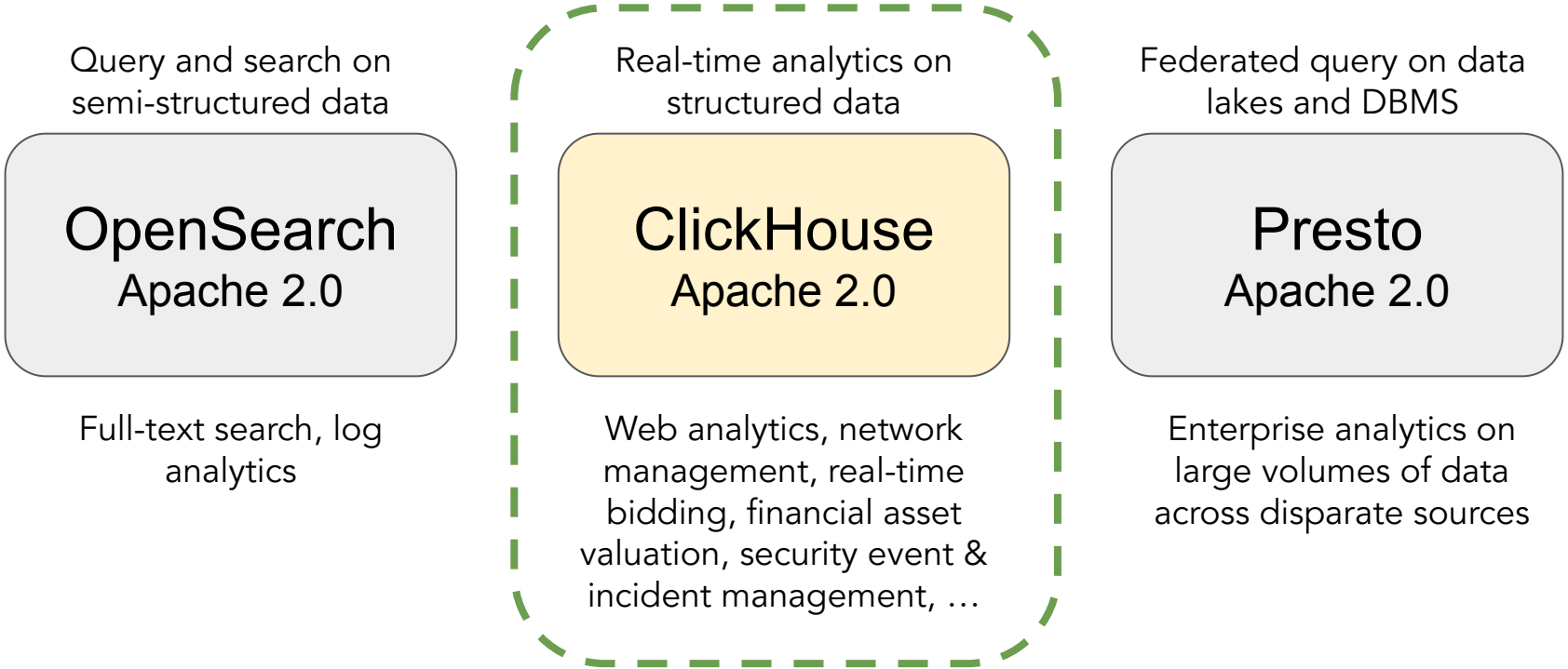
Snowflake strengths

- ✘ General purpose
- ✔ Serverless operation
- ✘ Handle wide range of applications
- ✘ Standards-compliant SQL
- ✔ UI with SQL editing & management

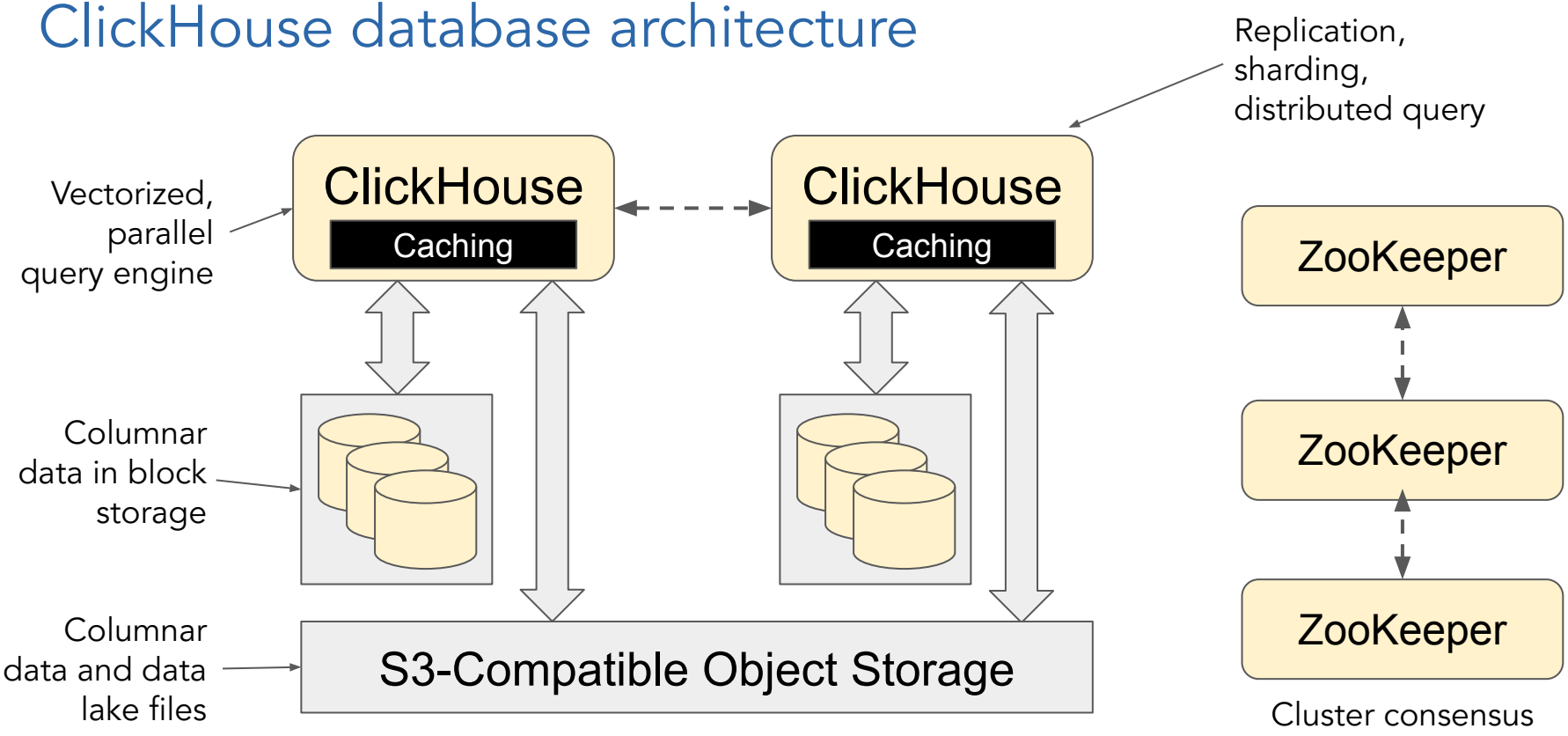
Snowflake weaknesses

- ✔ Keep data in your own cloud account
- ✔ Minimize costs for 24x7 systems
- ✔ Deliver stable real-time response
- ✔ Handle SaaS user-facing analytics
- ✔ No vendor lock-in

Second: pick an open source analytic database



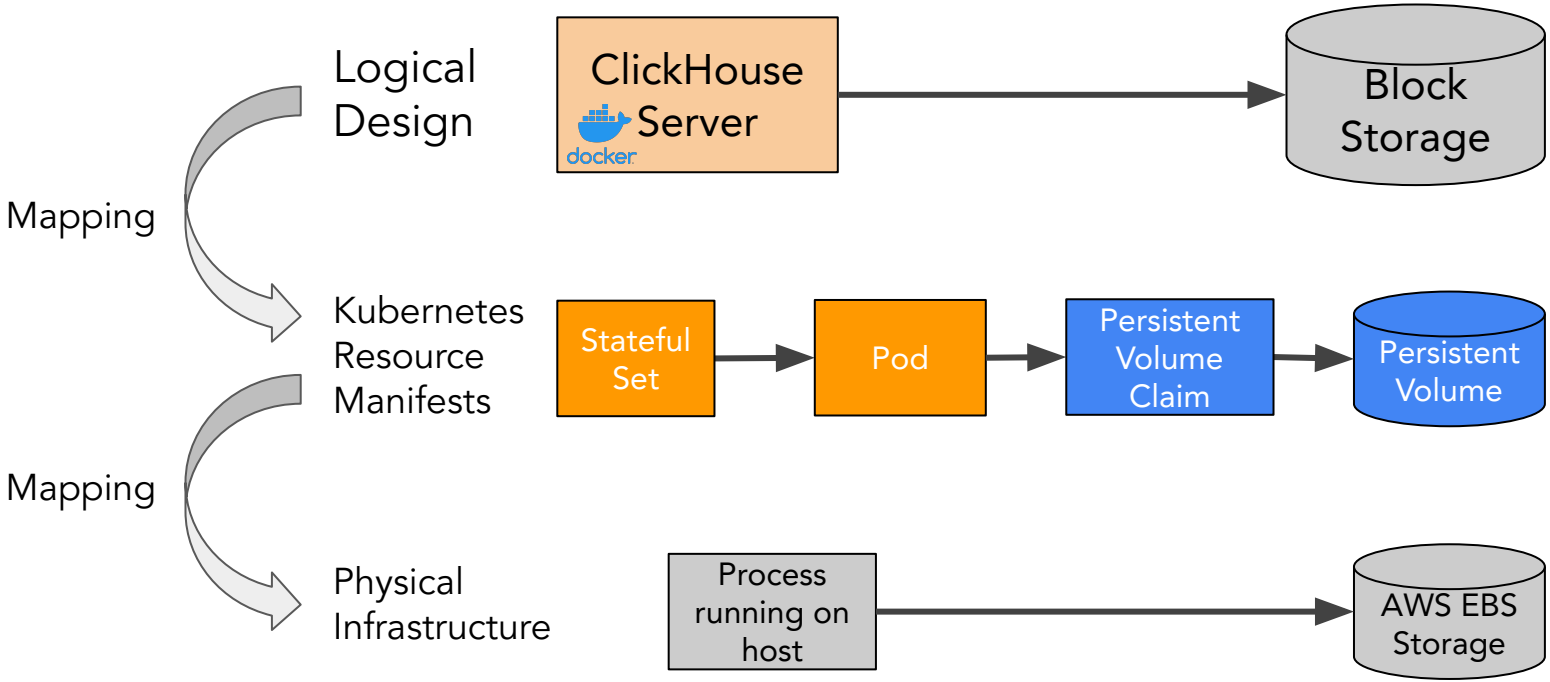
ClickHouse database architecture



Third: lay out the analytic platform logical design

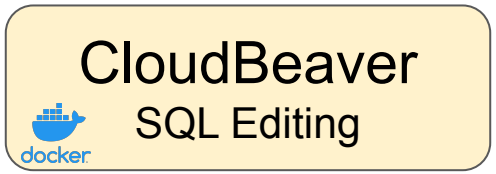
Implementing on Kubernetes with ArgoCD

Kubernetes orchestrates container-based applications



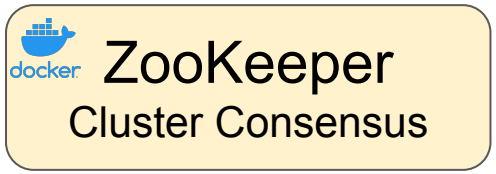
Map the logical design to Kubernetes resources

Install using Altinity ClickHouse operator

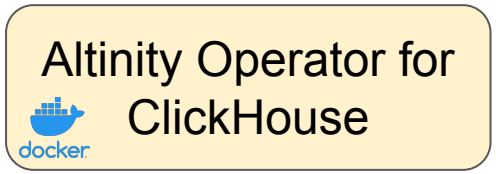


Install using manifest

Install using manifest



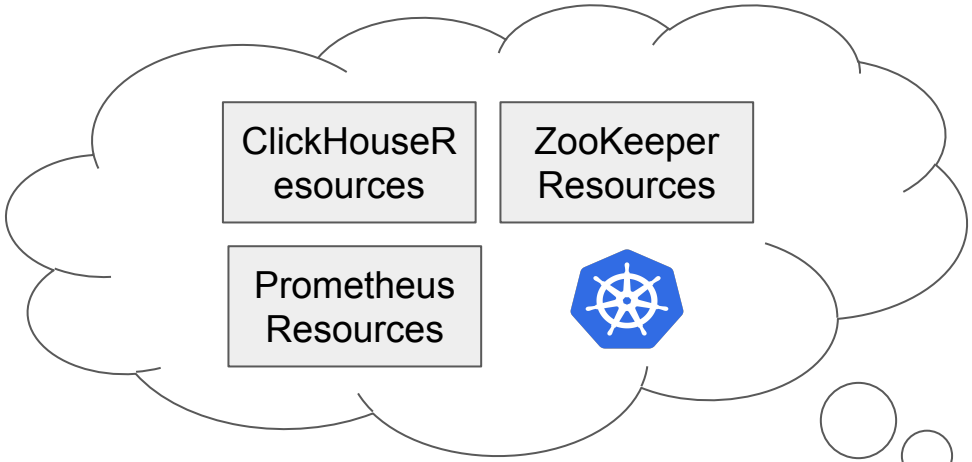
Install using manifest



Install using community Helm chart

Install using community Helm chart

How can we deploy the stack in a cloud native way?



Working Kubernetes Cluster

Definitions in Git

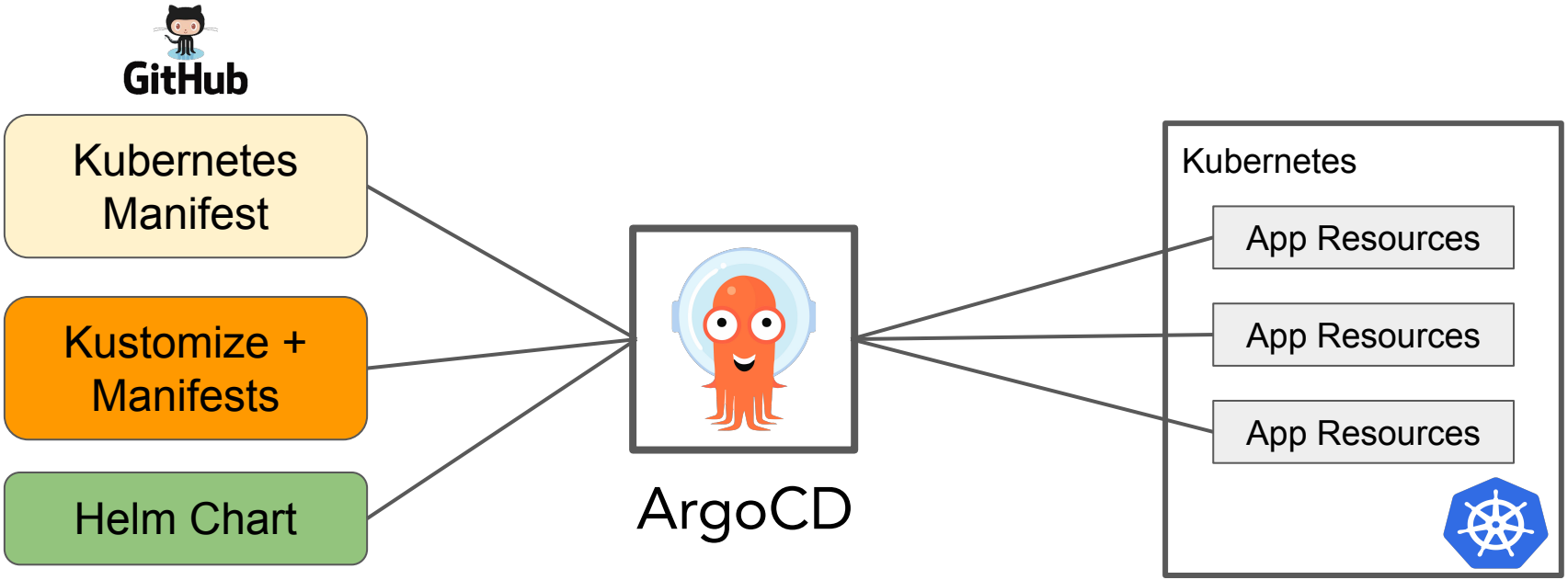
GitHub

ClickHo
Analytic

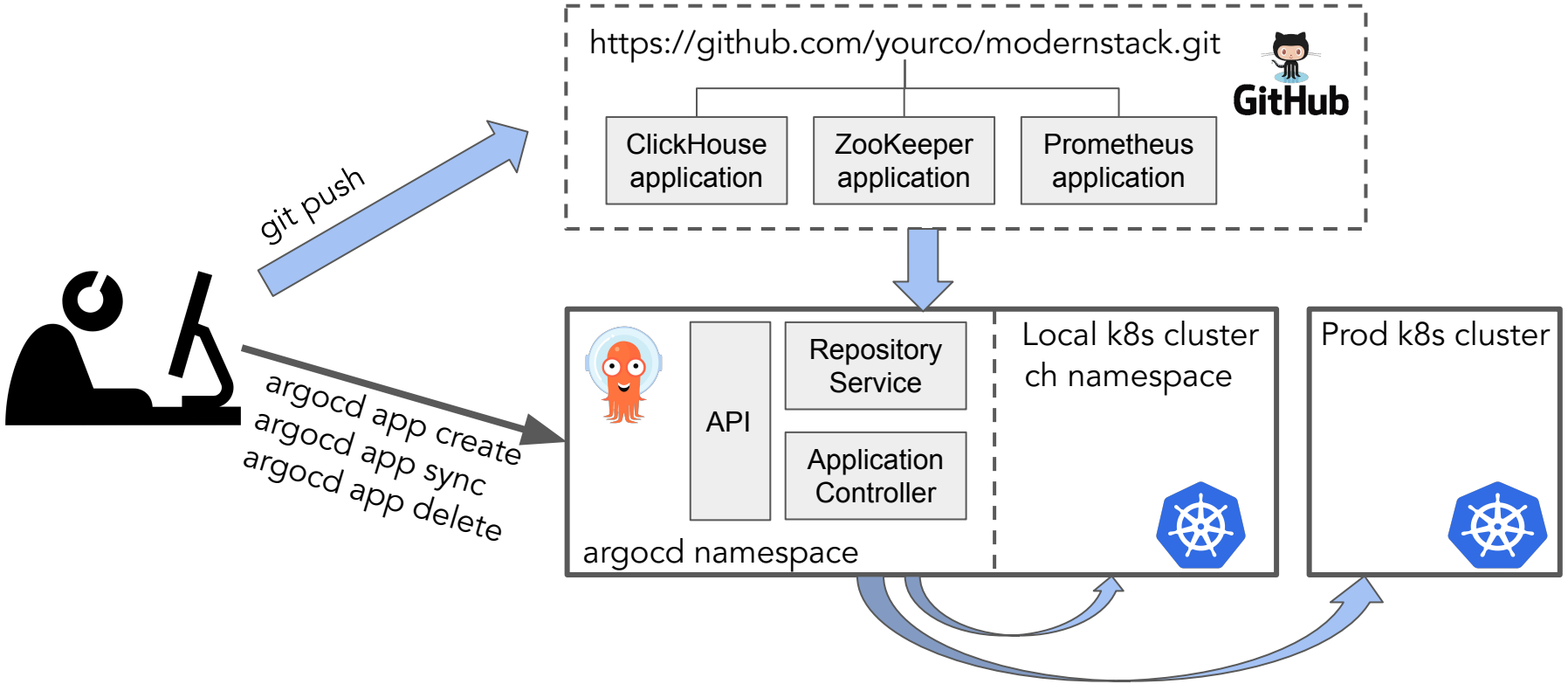
Prometheus
Operational Metrics

Cluster Consensus

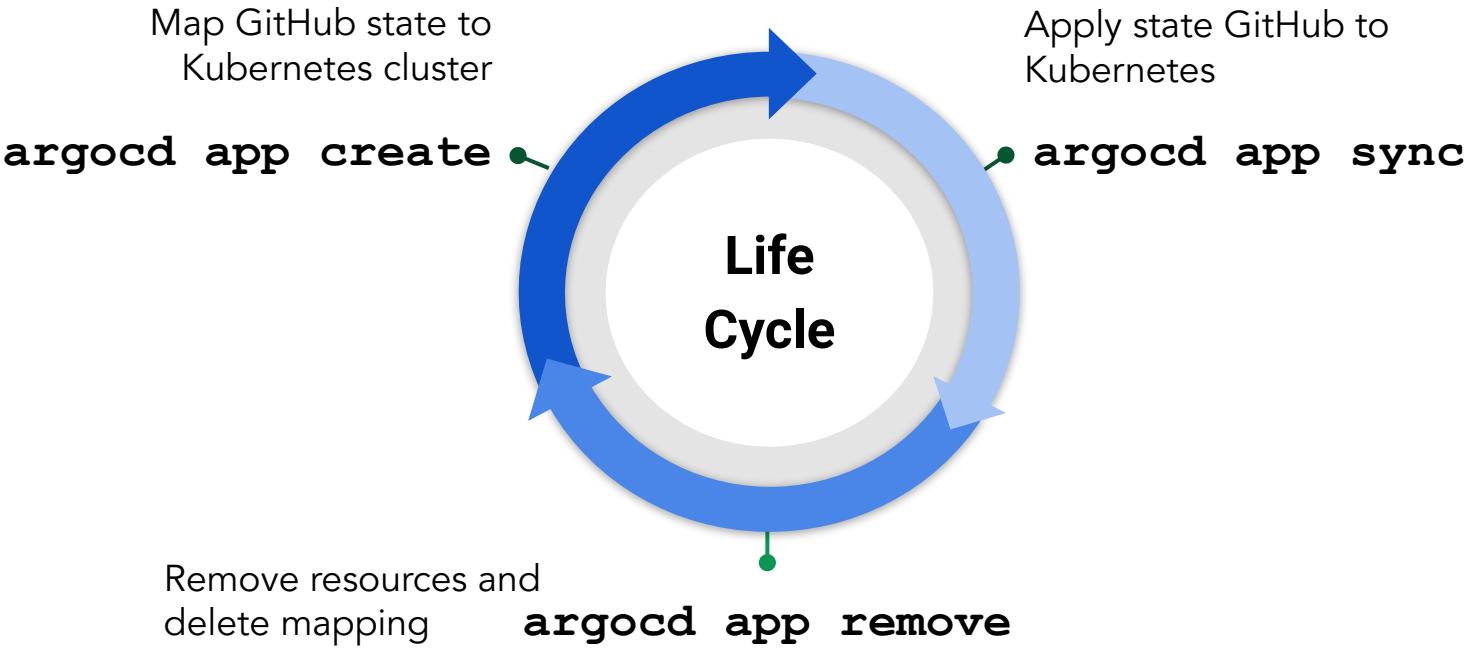
ArgoCD maps deployments from Git[Hub] to K8s



Basic GitOps using GitHub, ArgoCD, and Kubernetes



Life cycle for ArgoCD applications

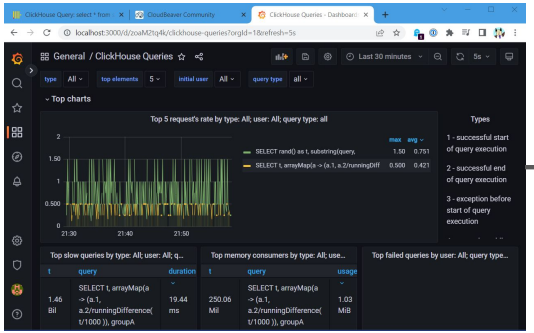


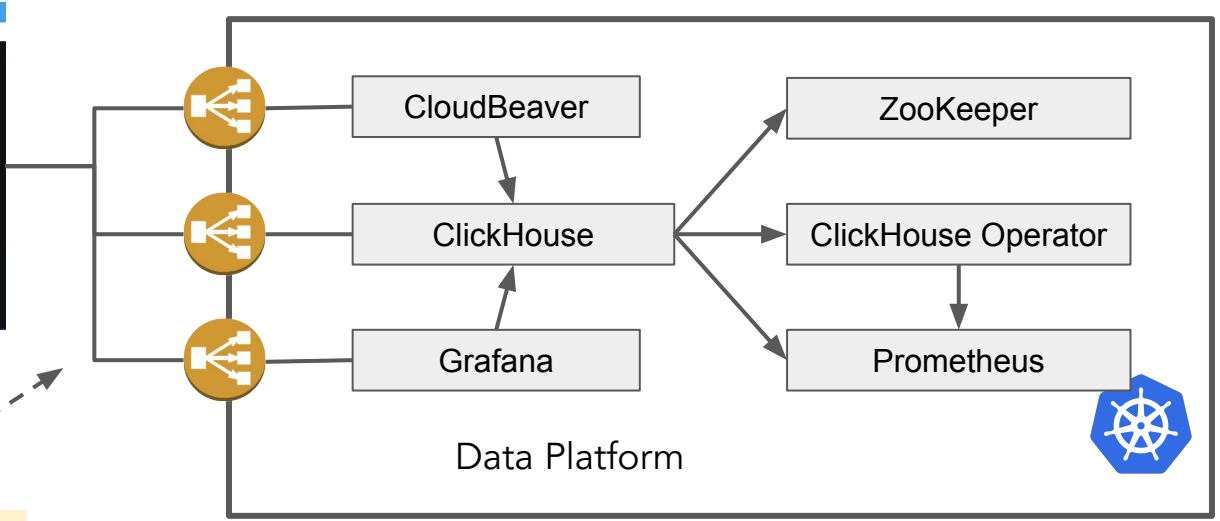
Managing Kubernetes applications with ArgoCD

DEMO TIME!

Wiring and dependencies in the stack

Applications





Forwarding, peering, or VPN

ArgoCD Assessment

Strengths

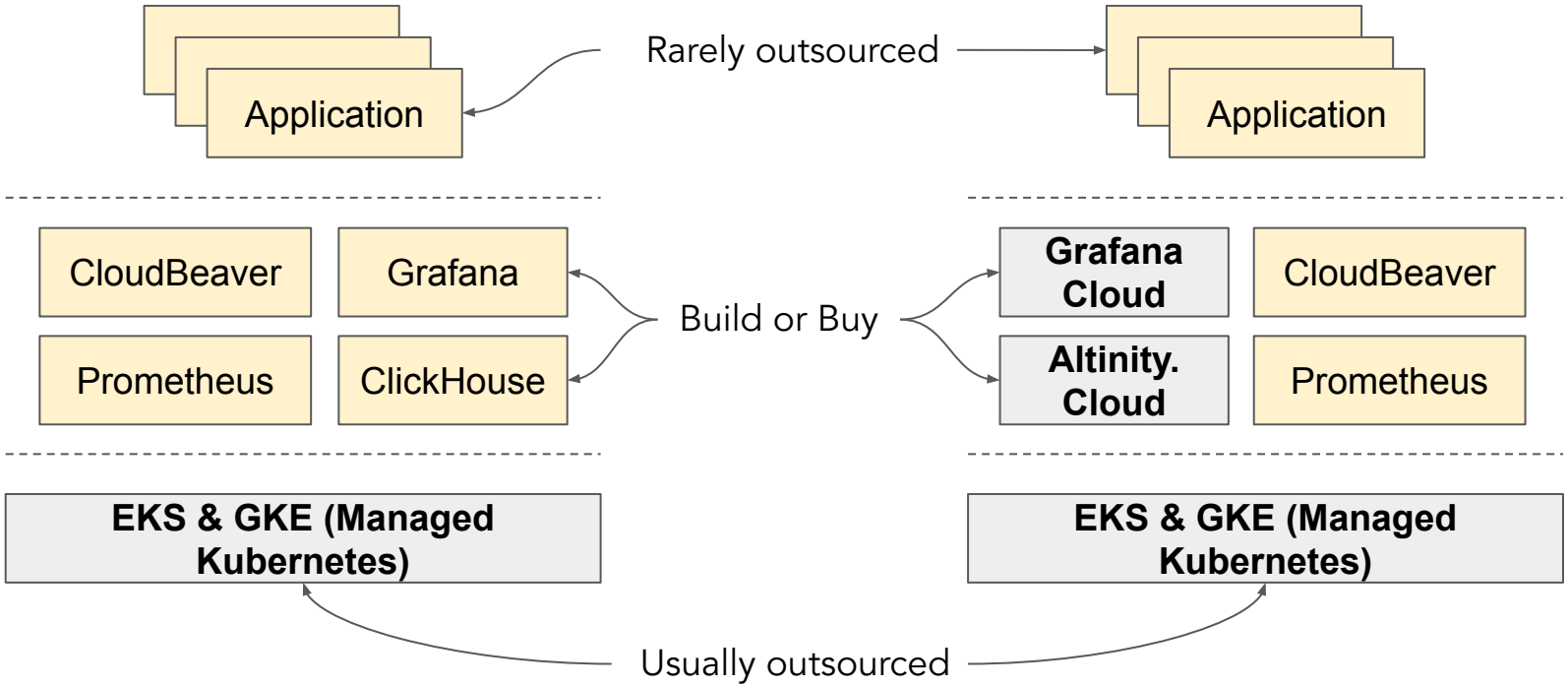
- Enables infrastructure as code - your configuration lives in Git
- Can map configuration to multiple environments
- Very adaptable–you can usually get things to install
- Exchange components to evolve the stack

Weaknesses

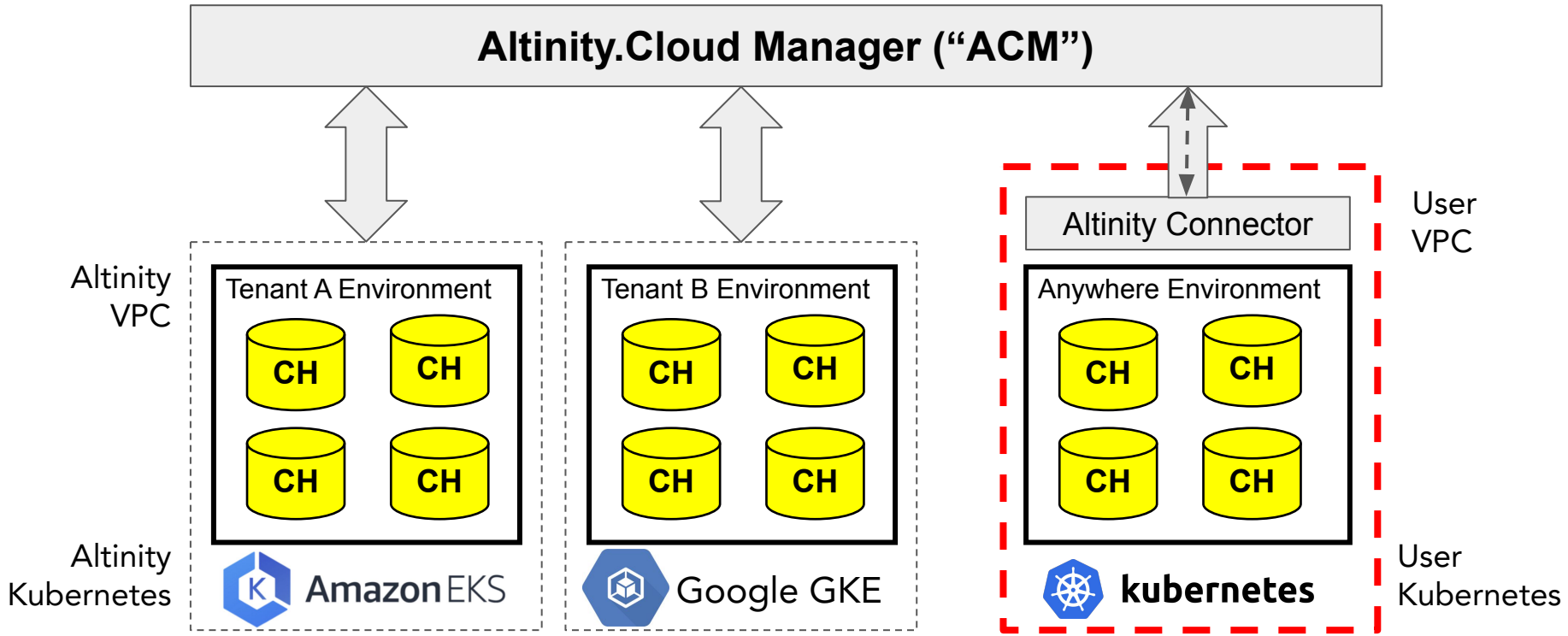
- Have to understand Kubernetes to understand ArgoCD
- Not all features are mature
- Full GitOps automation is complex
- Does not handle deployment outside of Kubernetes

Getting to a production analytic stack

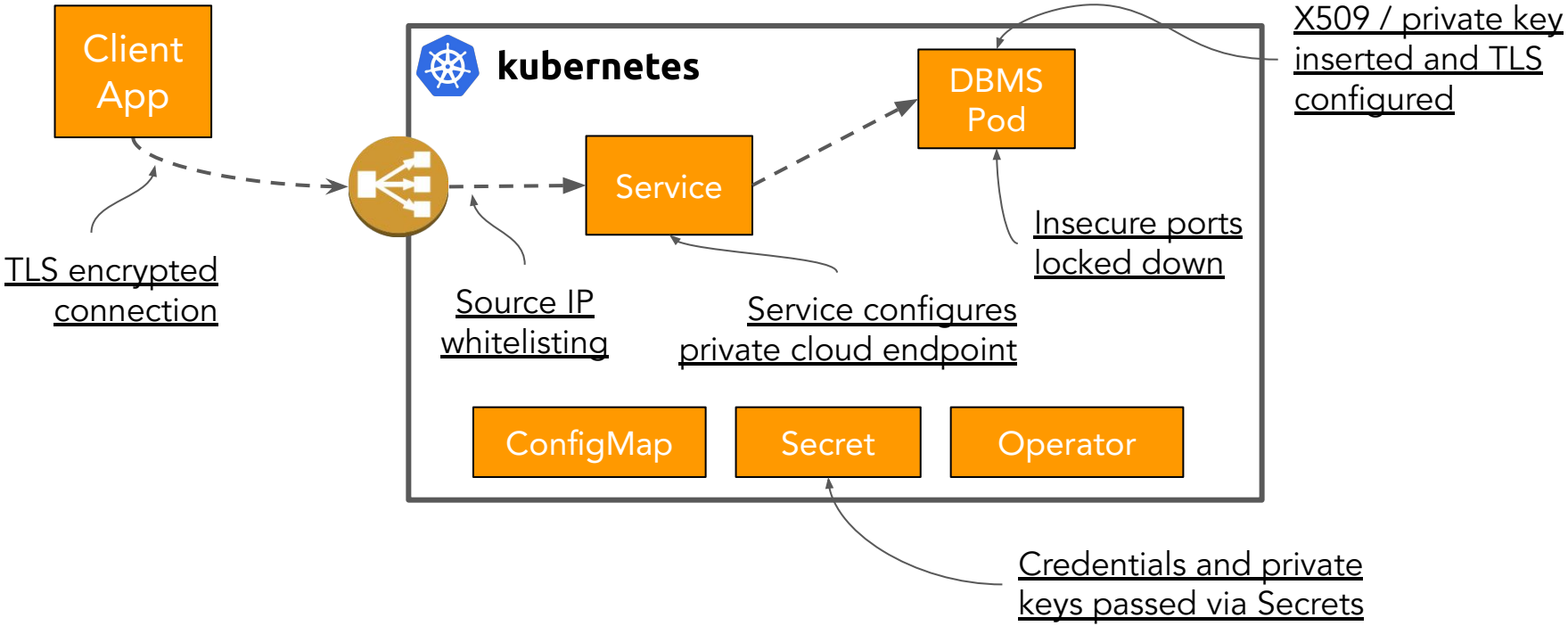
Buy vs. build, aka “Pick your battles”



Kubernetes enables more flexible managed services



Security in analytic platforms requires work



Look for operators and hardening guides for components

```
apiVersion: "clickhouse.altinity.com/v1"
kind: "ClickHouseInstallation"metadata:
  name: "prod"
spec:
  templates:
    serviceTemplates:
      - generateName: clickhouse-{chi}
        metadata:
          annotations:
            service.beta.kubernetes.io/aws-load-balancer-internal: "true"
        name: default-service-template
        spec:
          ports:
            - name: https
              port: 8443
            - name: secureclient
              port: 9440
          type: LoadBalancer
```

Vendor specific config for internal load balancer without public IP address

Only permit secure protocols

More tasks to deploy the analytic stack

- What other services do you need?
  - Airflow, Flink, Spark, …
- Adding hooks to synchronize Git fully with ArgoCD
- Building a dev/staging/prod pipeline
  - Or blue/green deployments
- Capacity planning and performance scaling
- Backup
- Monitoring

And of course, building your applications.

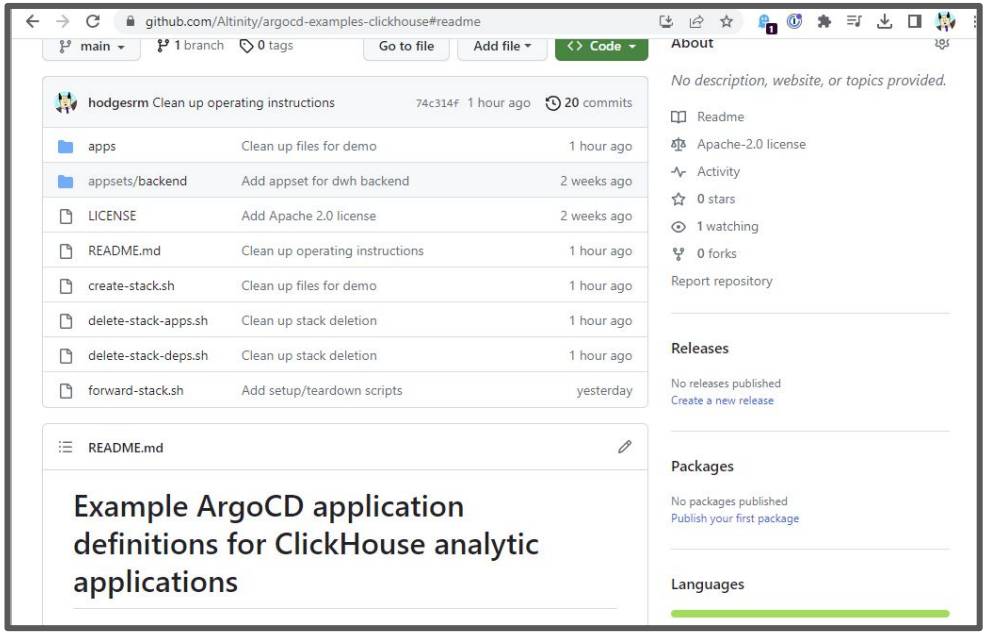
Final notes and more to come

Tips for building your own analytics platform

1. Open source stacks beat proprietary services for specific problems
2. Keep the problem small
3. Kubernetes offers state-of-the art platform for constructing the stack
4. ArgoCD maps Git state flexibly to Kubernetes resources
    a. Papers over installation differences
    b. Enables infrastructure as code for the entire stack
5. Production systems require expertise and careful design
6. Outside Kubernetes you need other options: Terraform or Ansible

How to get started with the example application

```
git clone https://github.com/Altinity/argocd-examples-clickhouse
```



Projects that went into the stack

- ArgoCD: https://argo-cd.readthedocs.io/en/stable/
- Altinity Projects
  - ArgoCD Examples
  - Altinity Kubernetes Operator for ClickHouse
  - Altinity Stable Builds for ClickHouse
- The rest of the stack
  - ClickHouse: https://github.com/ClickHouse/ClickHouse
  - Prometheus: https://github.com/prometheus-community/helm-charts
  - Grafana: https://github.com/grafana/grafana
  - CloudBeaver: https://github.com/dbeaver/cloudbeaver

Thank you and good luck!

Any Questions?

Robert Hodges
https://altinity.com

Altinity.Cloud
Altinity Stable Builds for ClickHouse
Altinity Kubernetes Operator for ClickHouse