An SLO-Driven Approach to Enhance Kubernetes Cluster Reliability

Qian Ding & Cong Chen, Ant Group

About Us

KubeCon CloudNativeCon North America 2020 Virtual

Qian Ding

sumang.dq@antgroup.com

Ant Group #Infra #SRE #K8S

Cong Chen

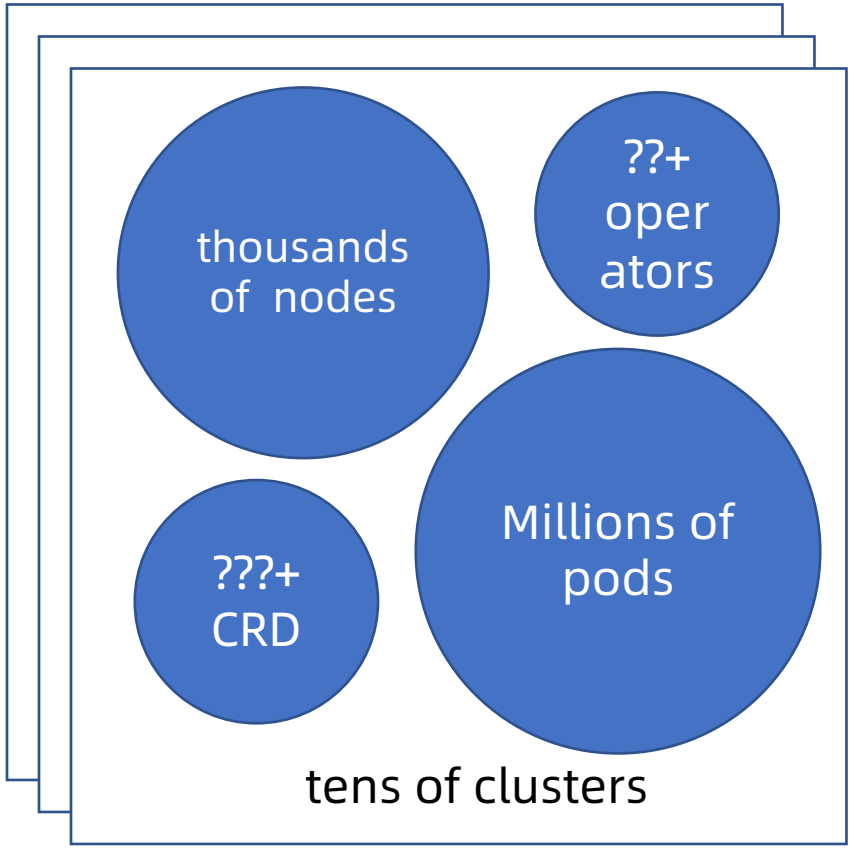
cc236361@antgroup.com

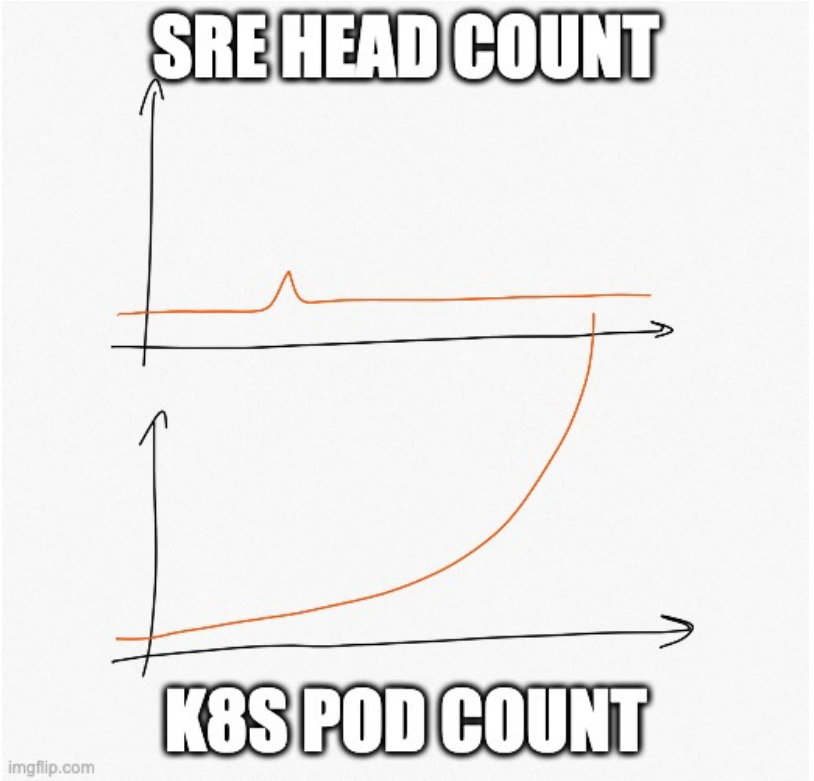
Ant Group #Infra #SRE #K8S

Outline

- Why
  - Motivation & Problem
- What
  - SLO Recap
  - SLO Design for Kubernetes Components
- How
  - SLO Alerting

Motivation



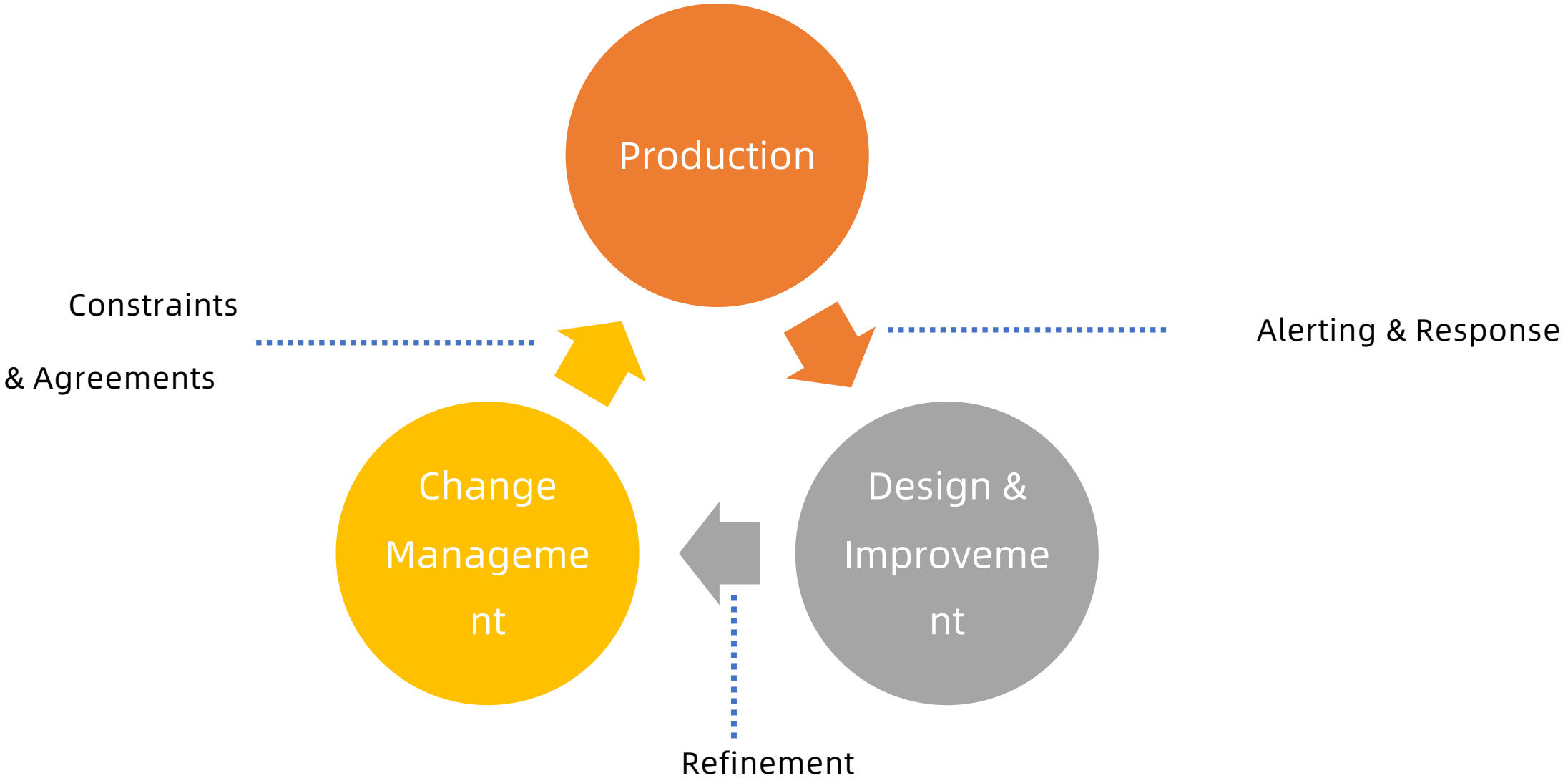


Cluster Management becomes increasingly challenging for K8S SRE when the cluster scales significantly.

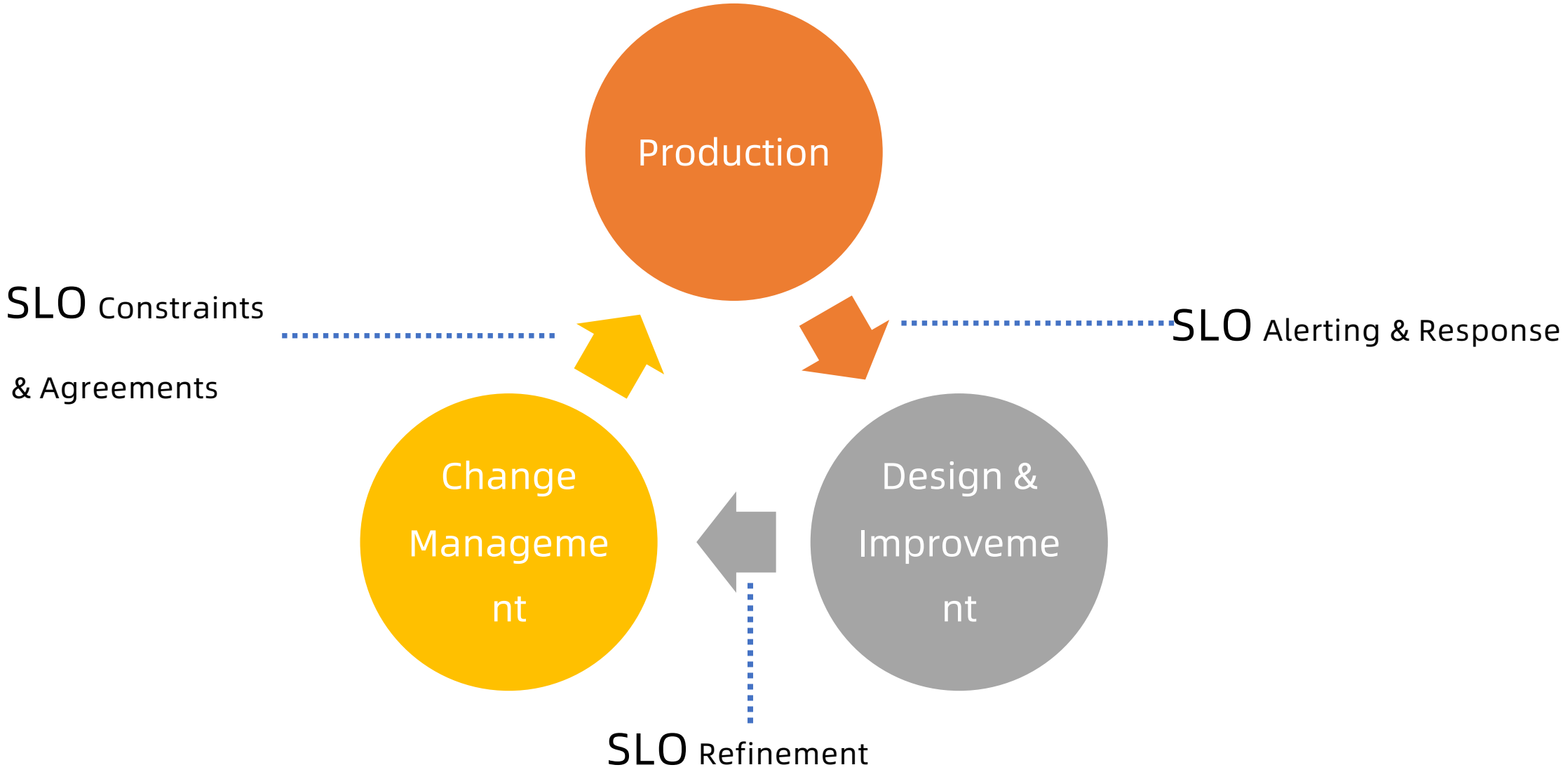
Fleet Management

- Release Velocity vs Reliability
- Master Components: complex interactions
- Node Components: delayed/scaled impact
- Node Capacity is tight: High-quality observability

General Approach



SLO Approach



SLO Design for Kubernetes

SLO Recap

SLI

Service Level Indicator

Metrics that use to measure the healthiness of a service

SLO

Service Level Objective

Quantitative targets set for the SLIs of the service to achieve

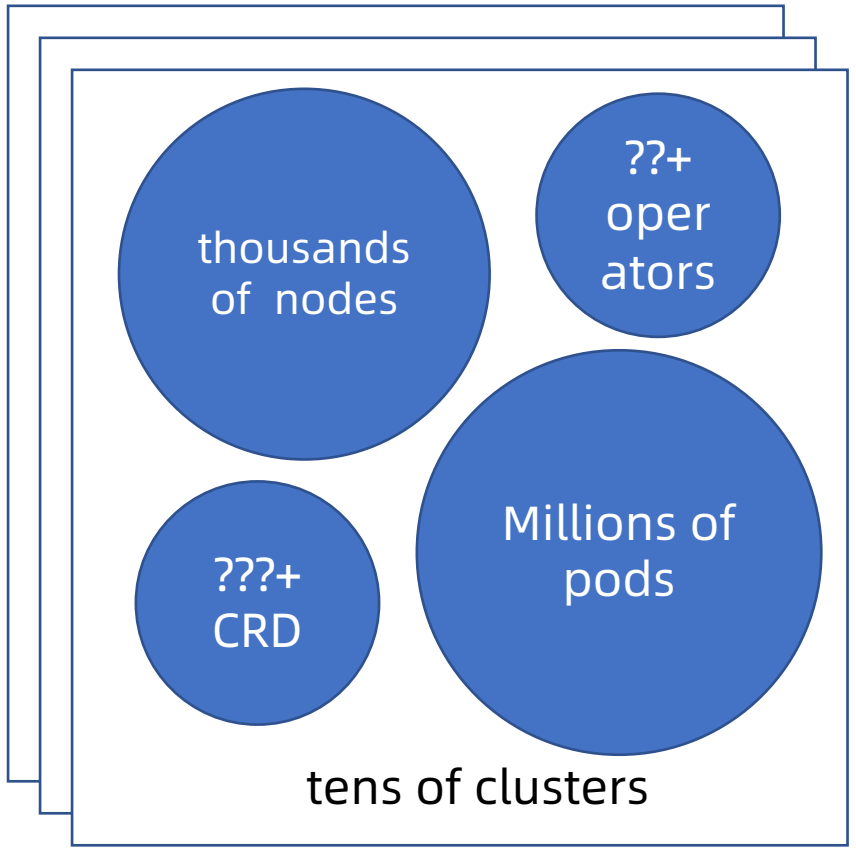
SLA

Service Level Agreement

Promises to make when failing to achieve the SLOs

tl;dr: SLIs drive SLOs which inform SLAs.

What SRE cares on K8S?



1 Resource delivery

- PAAS Level (e2e)
- Deployment / Workload
- Success Rate
- Delivery Latency

2 Pod Lifecycle SLO

- Single-Pod Lifecycle
- Creation, Update, Deletion
- Success Rate
- Latency

3 Pod health SLO

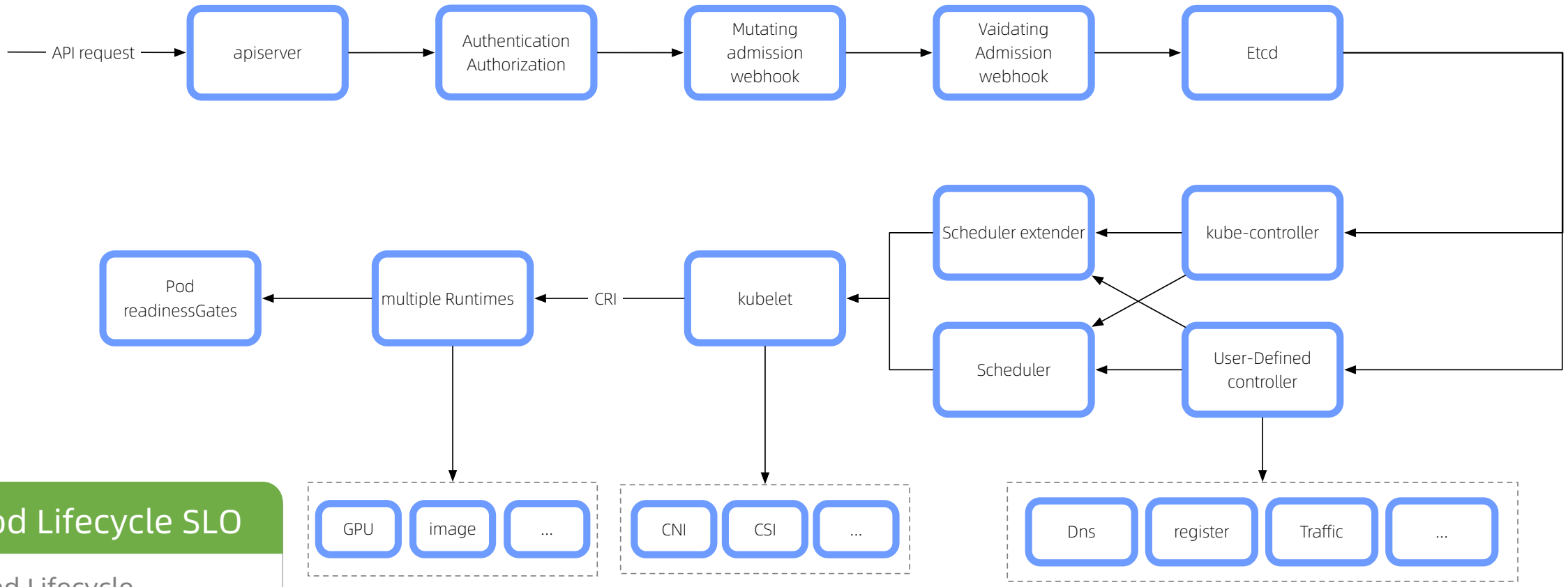
- Track Existing Pods
- Various Unhealthy Status
- Pod-level Up time

4 API-level SLO

- APIServer Availability
- Success Rate
- Latency

tl;dr: E2E SLOs represent real user experience

E2E SLO Design



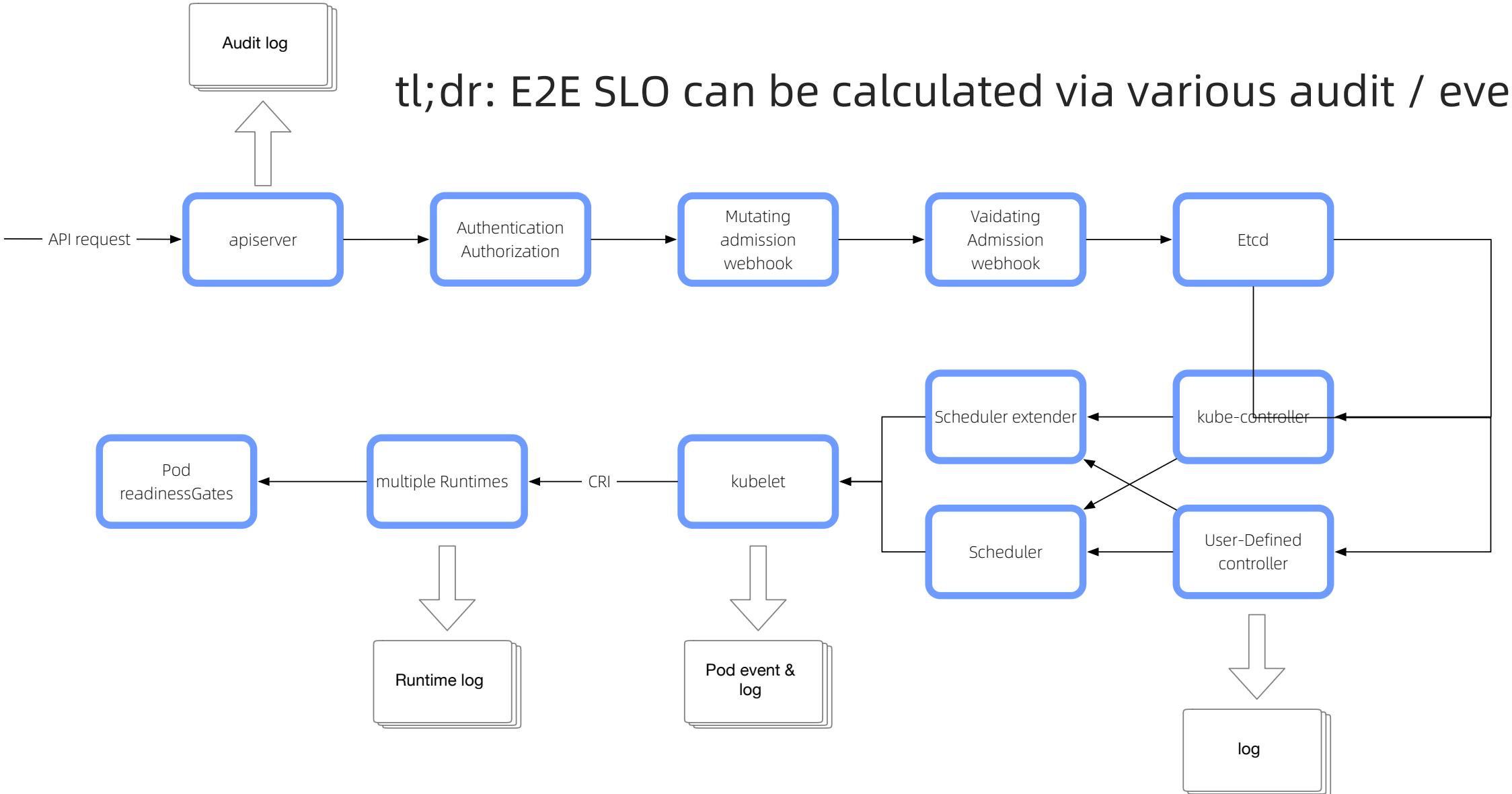
2 Pod Lifecycle SLO

- Single-Pod Lifecycle
- Creation, Update, Deletion
- Success Rate
- Latency

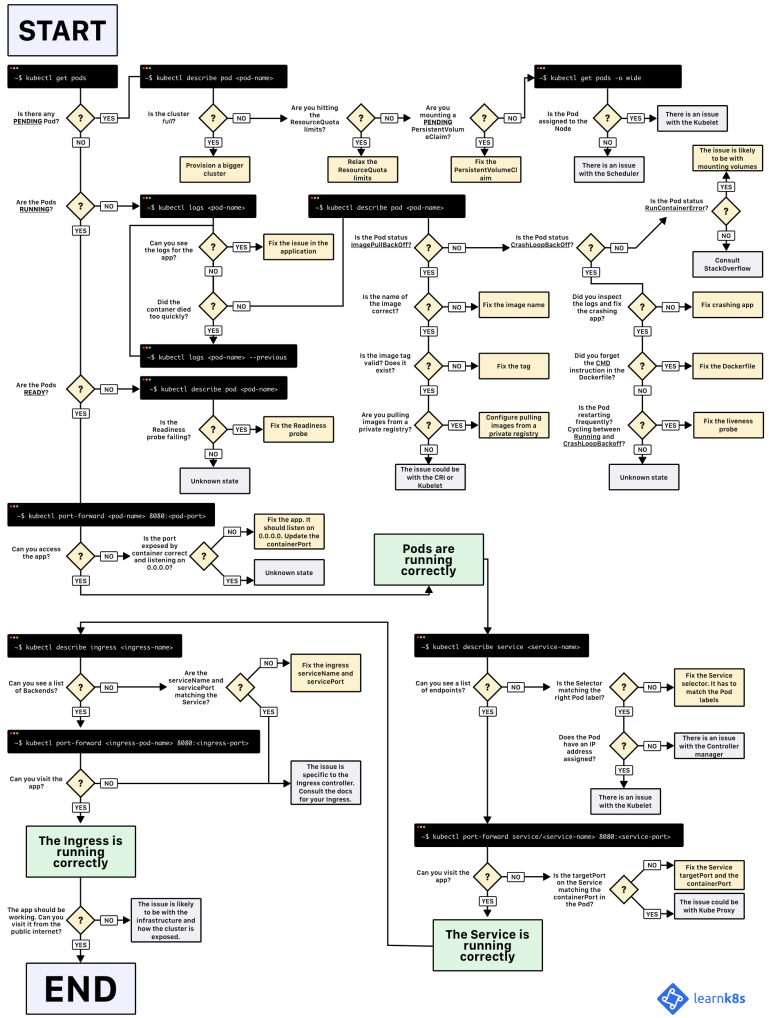
Example E2E SLO:
Monthly, 99% pods should be created successfully to reach the ready state within 5min.

E2E SLO Design

tl;dr: E2E SLO can be calculated via various audit / event logs.



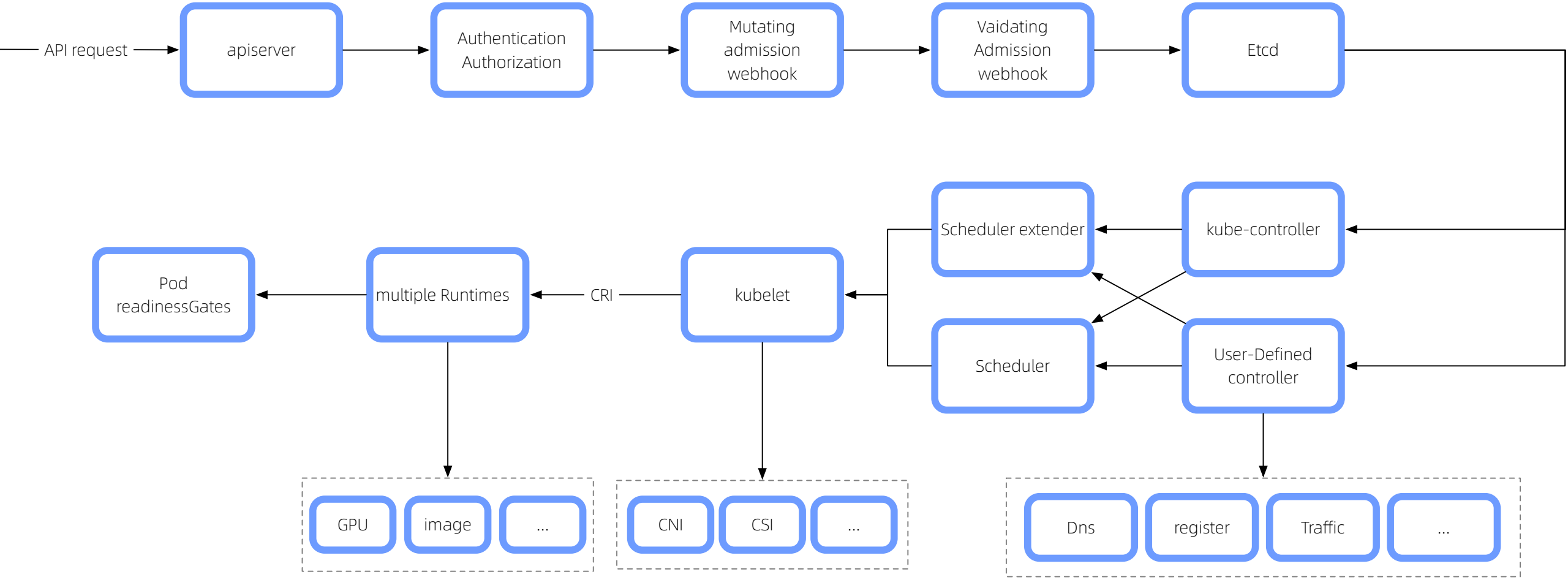
E2E SLO Design



demonstration of kubenetes flow chart

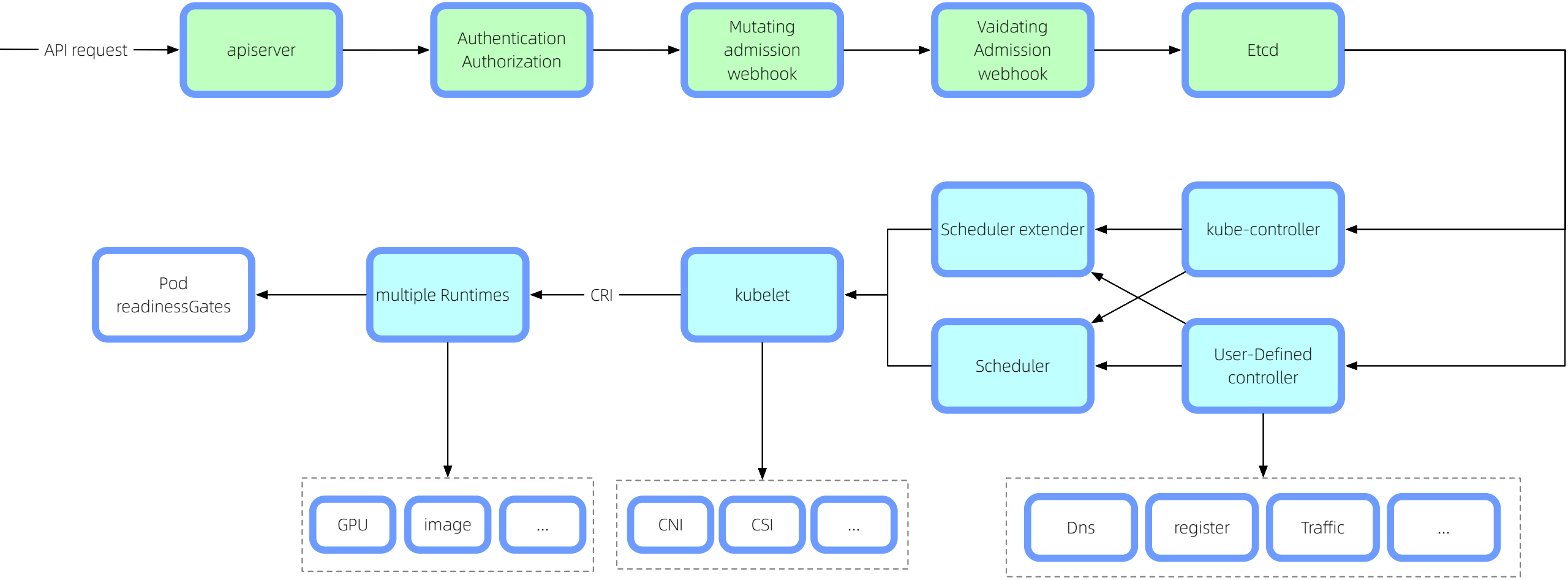
- Root causing is less obvious w/ E2E SLO metrics.
  - pod level, not component level
  - top-down approach
  - less noisy but may take longer time
  - rely on past expertise

Fine-grained SLO



Fine-grained SLO

- Synchronized Components: etcd, webhook, apiserver
- Asynchronized Components: controllers, scheduler, operators



Component SLO

- Stateless & synchronous
  - Success ratio and latency
  - Uptime
  - Traffic -> Client-side monitoring
  - ~~Saturation~~

Http request success rate

Http request latency

Component alive

Component SLO

- Stateful & Asynchronous
- Queueing system
- Reconciliation success and latency
- Leader-follower
- Internal & external dependencies

Reconcile success rate

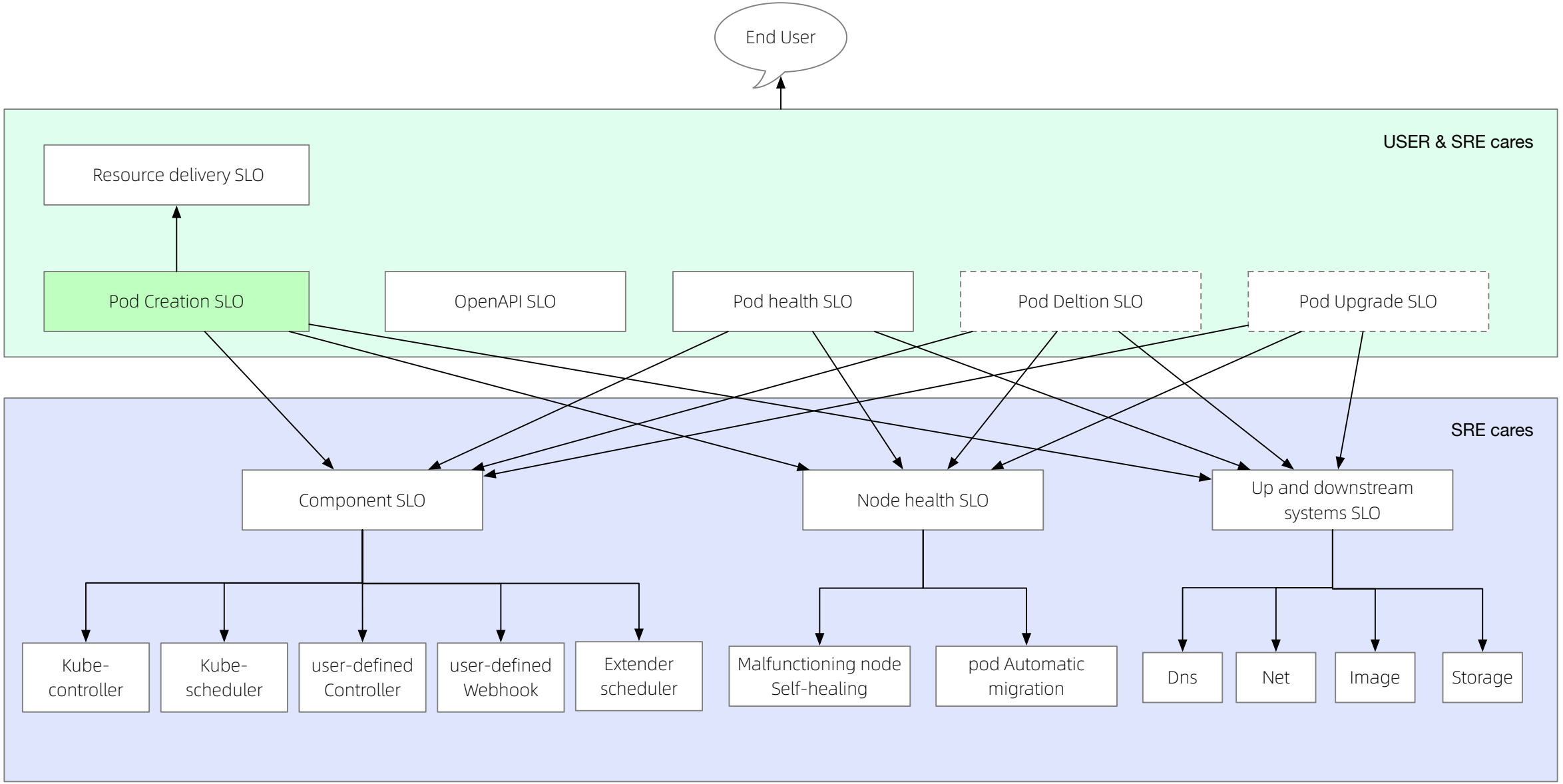
Reconcile latency

Component alive

Component leader select

Queue depth

Overall SLO Graph



SLO Alerting

Why RatioRate is bad?

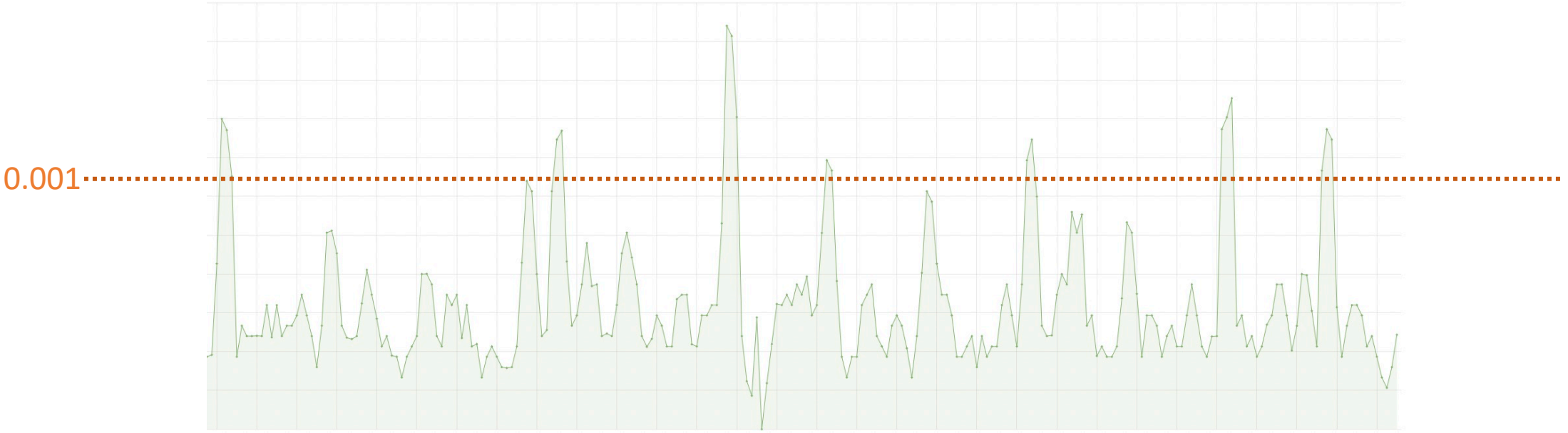
SLO: 99.9% of the requests to APIServer are successful* every month.

* Success means APIServer returns Status code < 500.

```
record: apiserver_request_sli_error_count

expr: sum(increase(apiserver_request_count{code=~"5.*"}[1m]))

record: apiserver_request_sli_total_count

expr: sum(increase(apiserver_request_count[1m]))

alert: high_apiserver_slo_failure_ratio

expr: apiserver_request_sli_error_count / apiserver_request_sli_total_count > 0.001
```

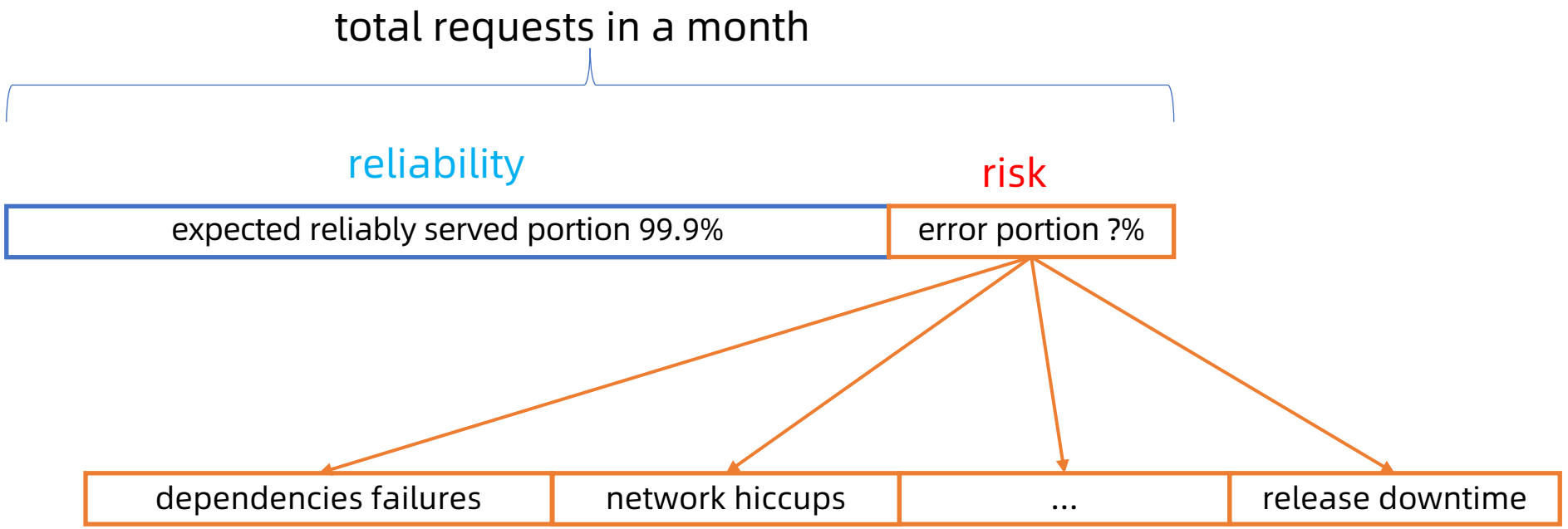
Why RatioRate is bad?

A Hypothetical Error Ratio Graph




```
alert: high_apiserver_slo_failure_ratio

expr: apiserver_request_sli_error_count / apiserver_request_sli_total_count > 0.001

for: 5m
```

Alerting Philosophy



tl;dr: SLO target should **NEVER** be 100%.

Alerting Philosophy

SLO: 99.9% of the requests to APIServer are successful* every month.

Figure out the maximum tolerable failed requests in a month.

Alert on a pre-defined portion of error budget being consumed in a short time window.

tl;dr: X% of monthly budget were consumed within a short time.

Alerting Philosophy

Page: # of SLI Failures in a Hour > 0.03 * (1-0.999) * (# of SLI Total in a month)

a pre-defined portion

Estimated monthly error budget

Ticket: # of SLI Failures in a day > 0.05 * (1-0.999) * (# of SLI Total in a month)

0.03 portion in a hour --> in 34 hours, the SLO will be broken;
0.05 portion in a day --> in 20 days, the SLO will be broken

Alert Practice

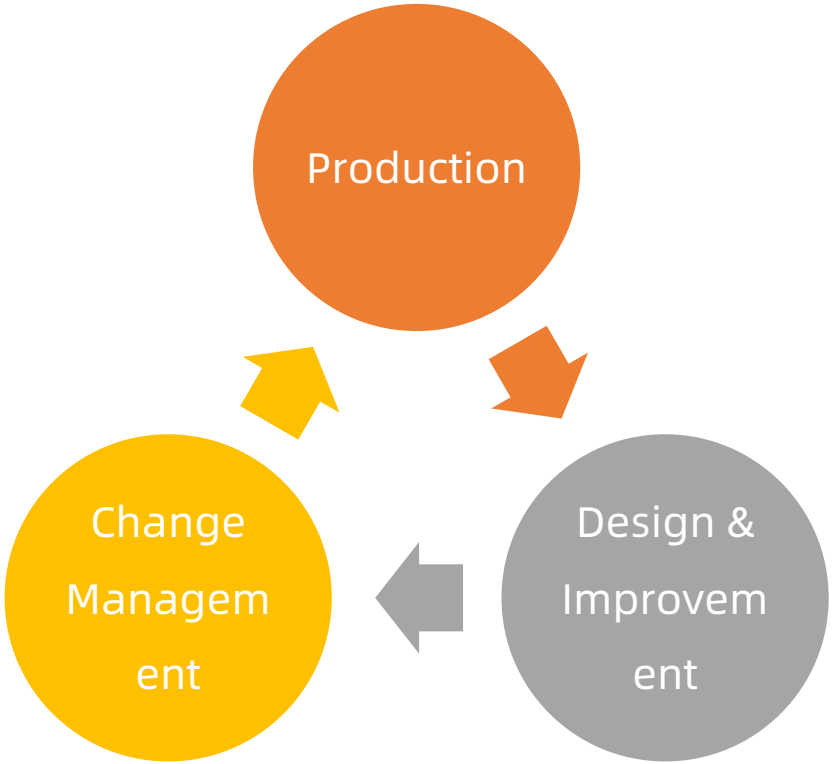
Page: # of SLI Failures in a Hour > 0.03 * (1-0.999) * (# of SLI Total in a month)

```
record: apiserver_request_sli_error_count
expr: sum(increase(apiserver_request_count{code=~"5.*"}[1m]))
record: apiserver_request_sli_total_count
expr: sum(increase(apiserver_request_count[1m]))
record: apiserver_request_sli_error_count_hourly
expr: sum_over_time(apiserver_request_sli_error_count[1h])
record: apiserver_request_sli_total_count_monthly
expr: sum_over_time(apiserver_request_sli_total_count[30d])
alert: high_apiserver_slo_failure_ratio
expr: apiserver_request_sli_error_count_hourly >
      0.03 * (1 - 0.999) * apiserver_request_sli_total_count_monthly
```

Summary

SLO Management

- Fined-grained SLO Definition
- Monitoring and Alerting
- SLO Documentation and Agreement
- Continuous Review and Report
- Automation on root-causing




KEEP CLOUD NATIVE
EVERYWHERE
KubeCon
CloudNativeCon
North America 2020
Virtual