KubeCon



Not Just for Infrastructure

Rebuilding an OSS Product with Kubernetes-Style Architecture

Artur Wierzbicki

Senior Engineering Manager · Grafana Labs

Stephanie Hingtgen

Principal Software Engineer · Grafana Labs

25M+ users,
and **10+** years
of workflows
they depend on.

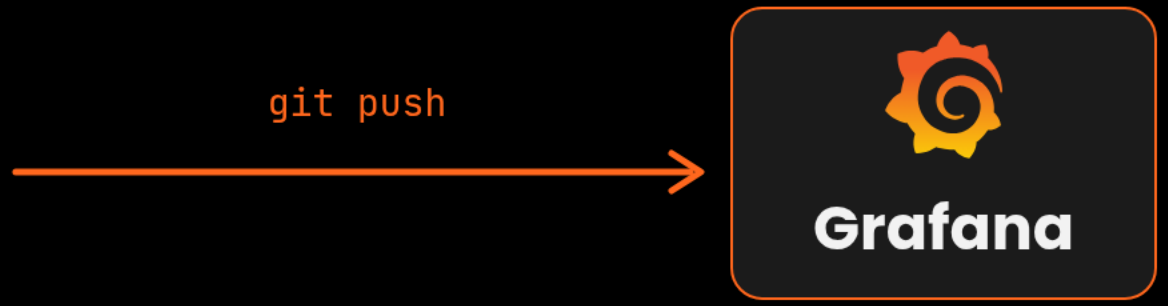


Grafana v1.0, January 2014

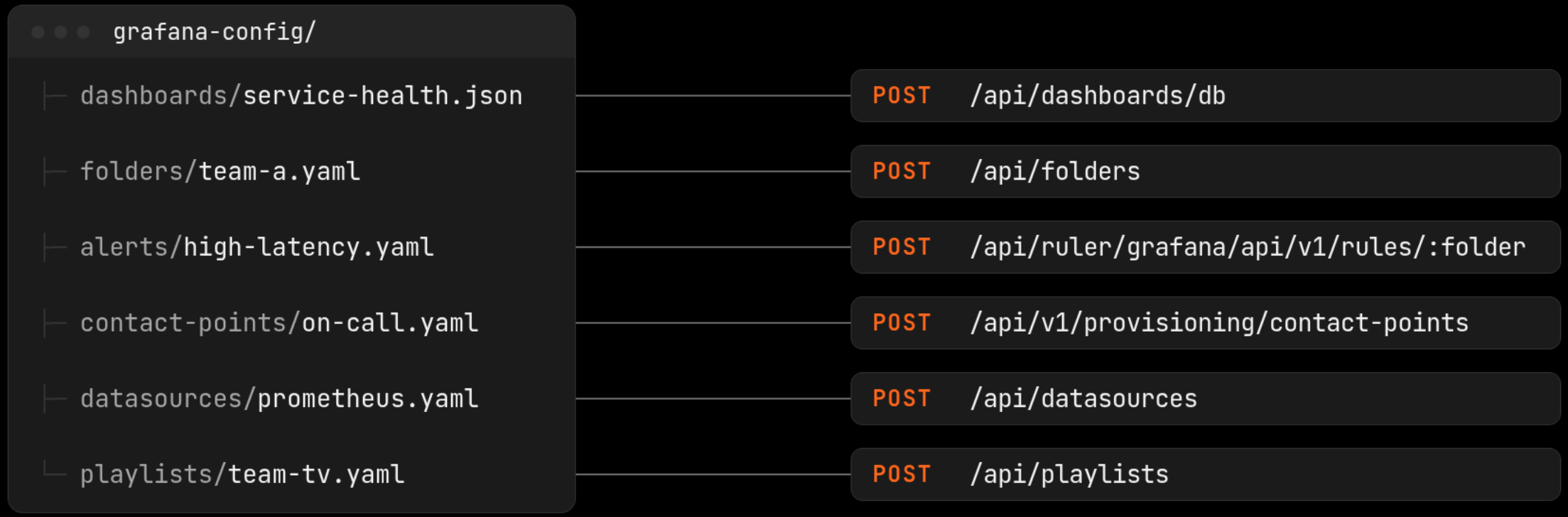
We wanted everything as code

Every new feature, manageable as code, with no extra work

```
grafana-config/
├─ dashboards/service-health.json
├─ folders/team-a.yaml
├─ alerts/high-latency.yaml
├─ contact-points/on-call.yaml
├─ datasources/prometheus.yaml
└─ playlists/team-tv.yaml
```

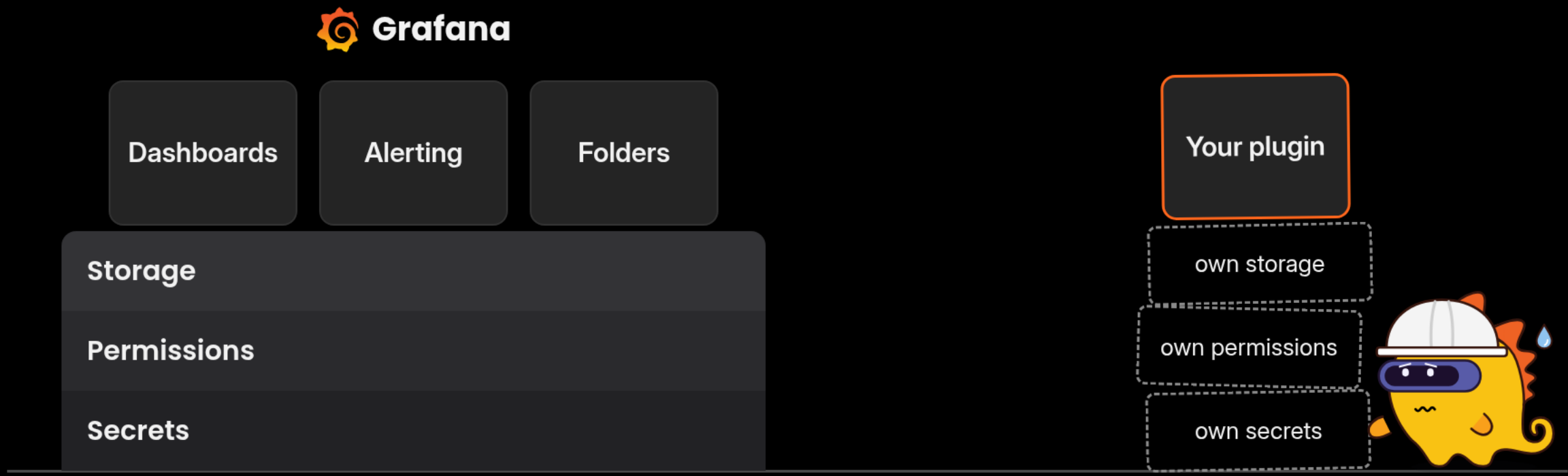


But every feature shipped its own API

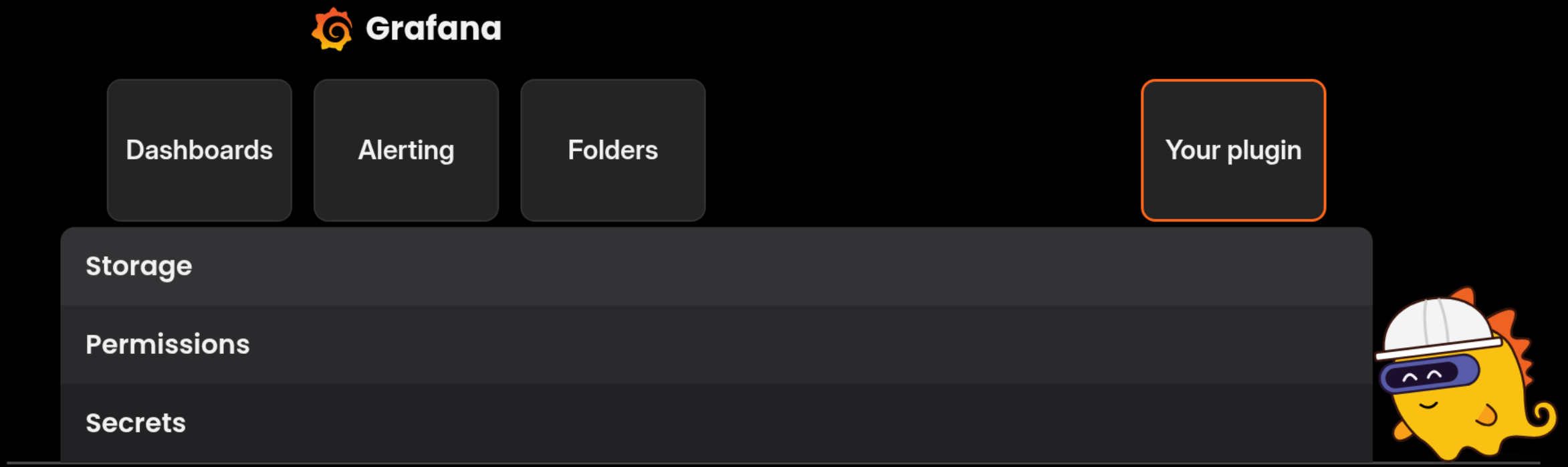


many ways to version, validate and authorize

We wanted a bigger plugin ecosystem

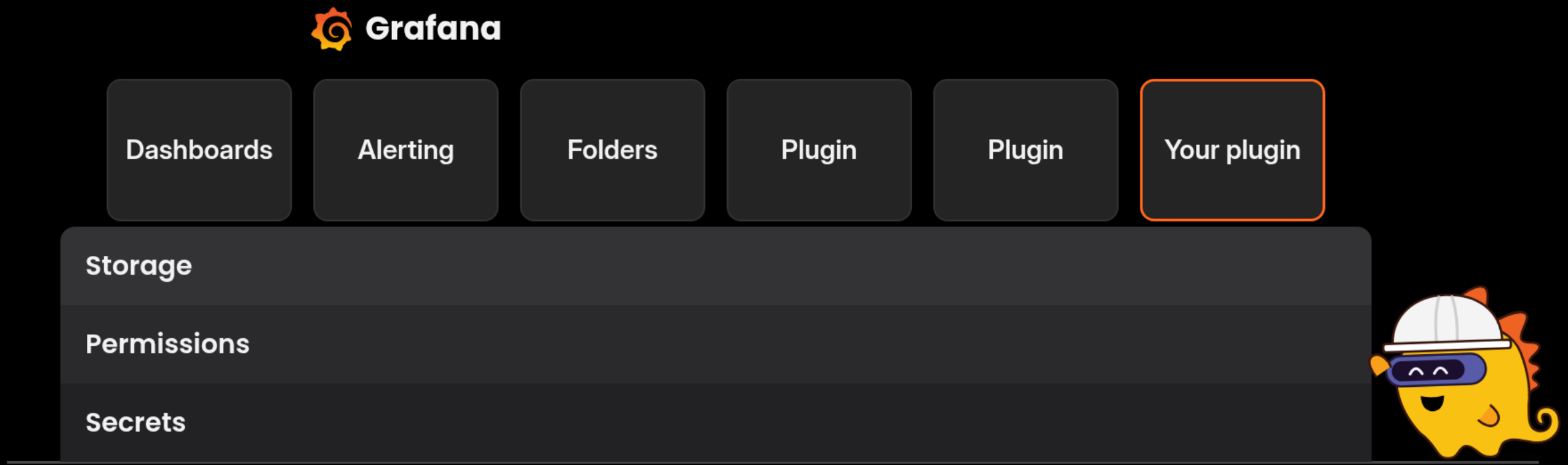


We wanted a bigger plugin ecosystem



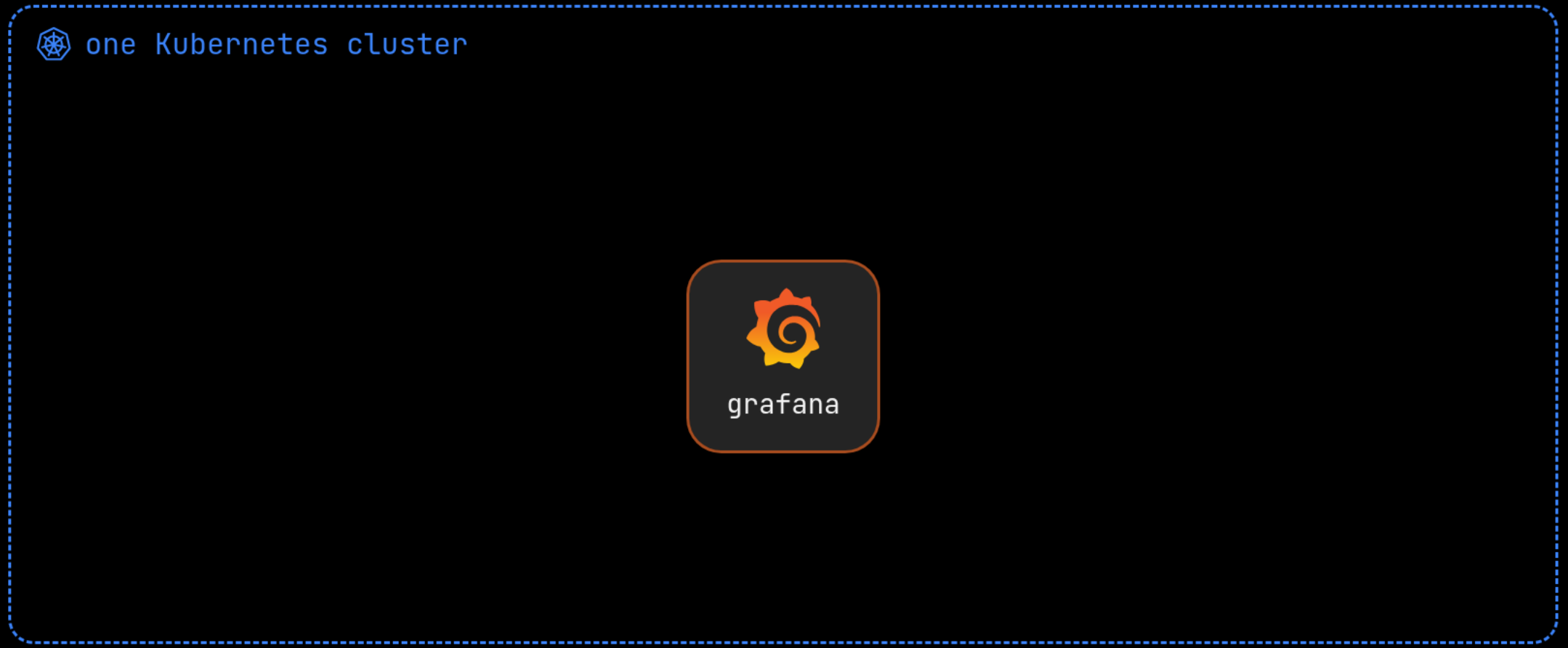
We had to make it simple to build powerful plugins that feel like core Grafana.

We wanted a bigger plugin ecosystem



We had to make it simple to build powerful plugins that feel like core Grafana.

We needed to go multi-tenant



One customer, one Grafana

We needed to go multi-tenant



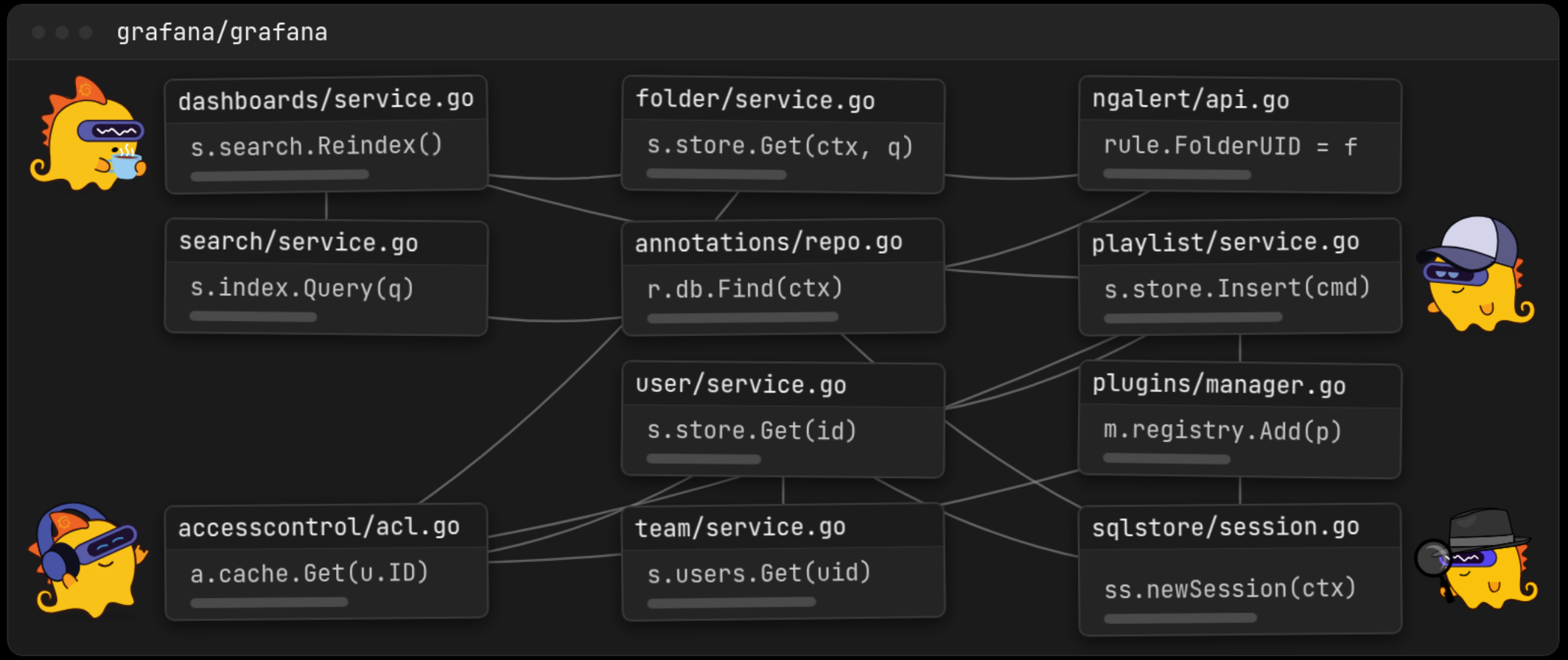
Every new customer, another Grafana

We needed to go multi-tenant

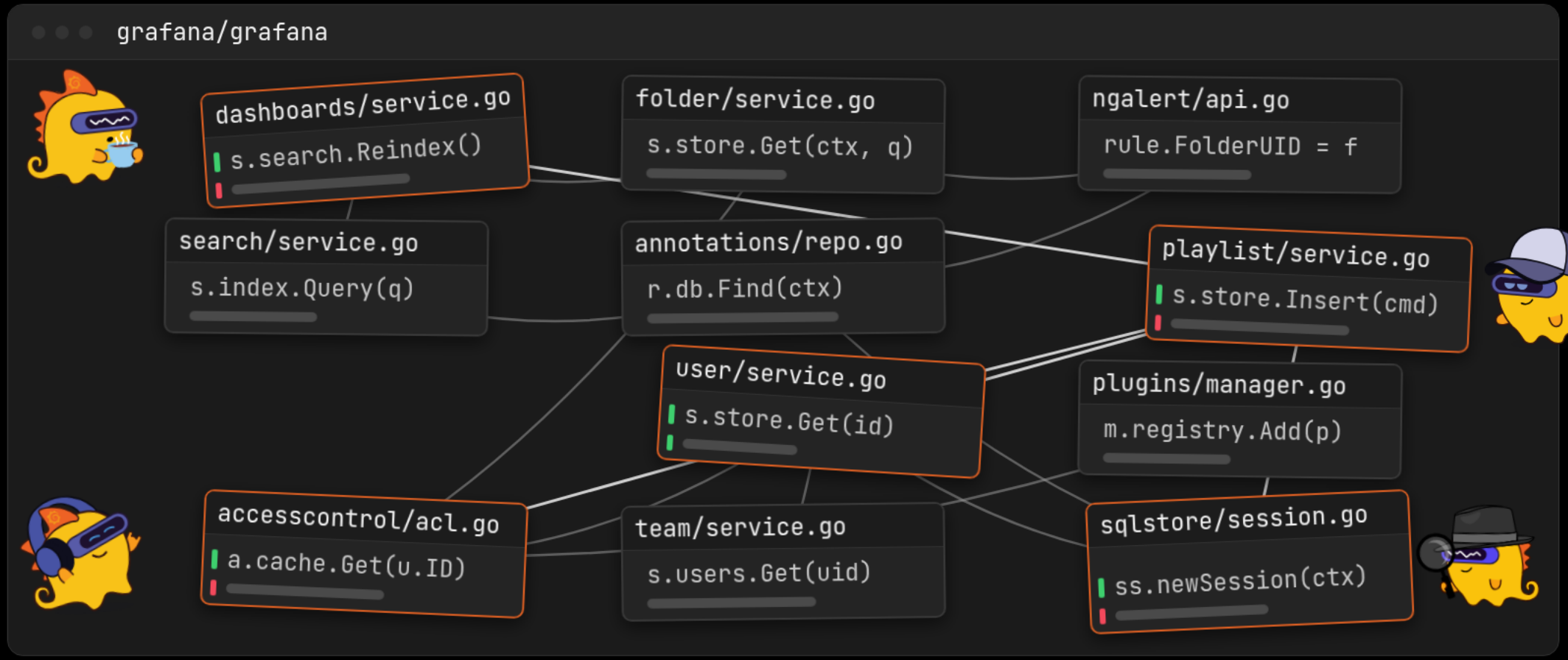


Running all of them got extremely hard

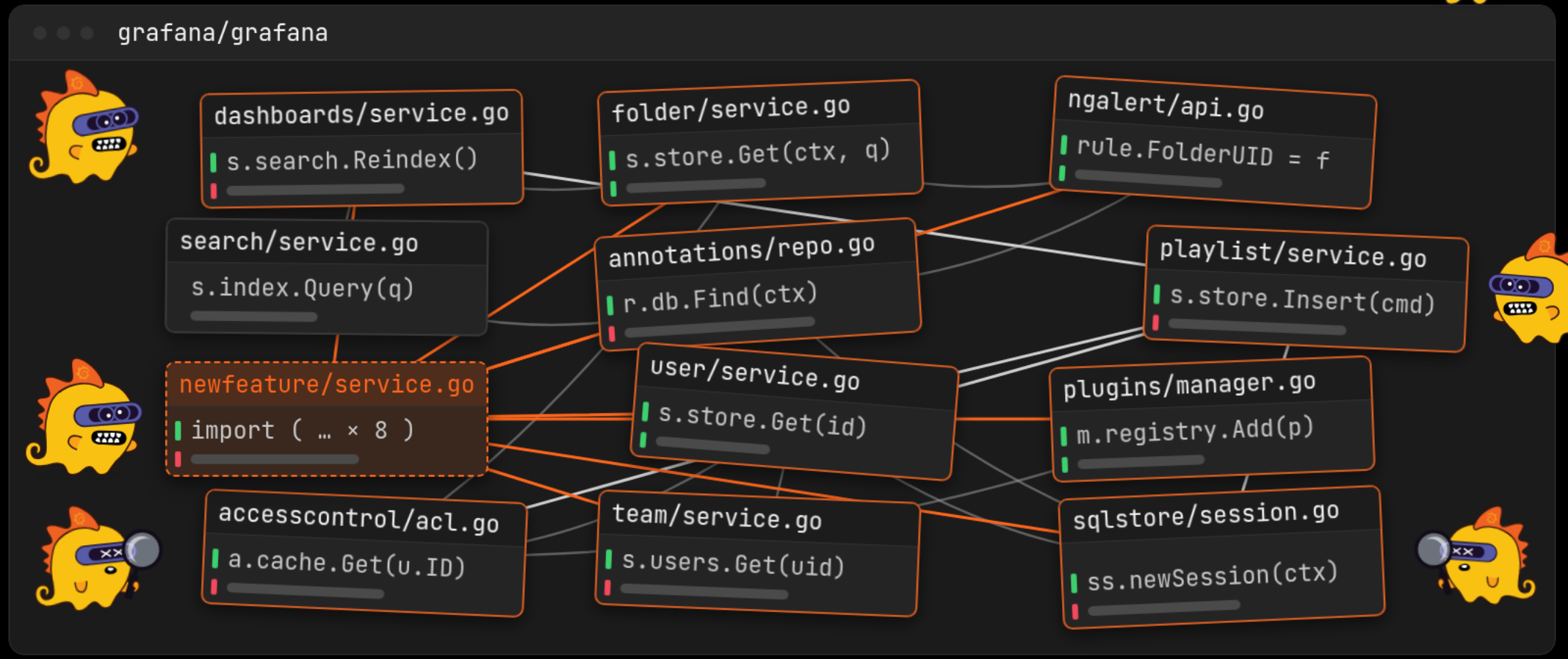
We had to accelerate



We had to accelerate



We had to accelerate



Adding a feature meant touching everything

How do we get there?

As code | Plugin ecosystem | Multi-tenancy | Moving faster



① No framework

Steer clear from gin and similar, they are absolutely not idiomatic.

Just f*ing use Go**

Stop pretending you need a framework.

[redacted] · 4y ago

GO has no framework... GO needs no framework...



① No framework

Steer clear from gin and similar, they are absolutely not idiomatic.

Just f***ing use Go

Stop pretending you need a framework.

· 4y ago

GO has no framework... GO needs no framework...

② A Go framework

- Goravel
- Encore
- GoFrame
- Buffalo



① No framework

Steer clear from gin and similar, they are absolutely not idiomatic.

Just f*ing use Go**

Stop pretending you need a framework.

[redacted] · 4y ago

GO has no framework... GO needs no framework...

③ Embed Kubernetes

grafana/grafana
pkg/services/apiserver/service.go

```go
package apiserver

import (
    "k8s.io/apiserver/pkg/server"
    "k8s.io/apimachinery/pkg/runtime"
    "k8s.io/client-go/rest"
)
```

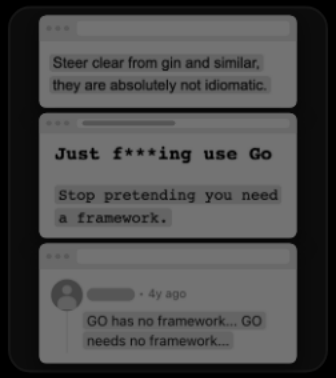


② A Go framework

- Goravel
- Encore
- GoFrame
- Buffalo

③ Embed Kubernetes

① No framework



② A Go framework

- Goravel
- Encore
- GoFrame
- Buffalo

grafana/grafana

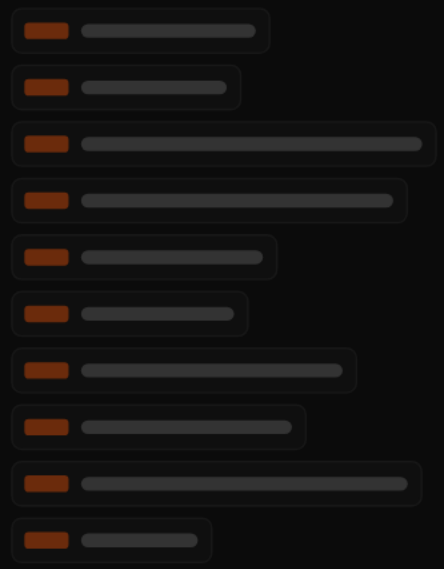
pkg/services/apiserver/service.go

```
package apiserver

import (
    "k8s.io/apiserver/pkg/server"
    "k8s.io/apimachinery/pkg/runtime"
    "k8s.io/client-go/rest"
)
```



Kubernetes already had it all



As code



Plugin ecosystem



Multi-tenancy



Moving faster

Kubernetes already had it all

As code

```
apiVersion: …/v1
kind: Dashboard
                {}schema
```

```
apiVersion: …/v1beta1
kind: AlertRule
                {}schema
```

A versioned schema for every resource



Plugin ecosystem



Multi-tenancy



Moving faster



Kubernetes already had it all

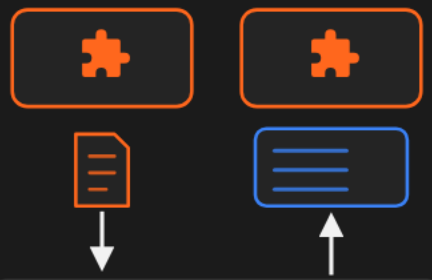
As code

```
apiVersion: …/v1
kind: Dashboard
```
{} schema

```
apiVersion: …/v1beta1
kind: AlertRule
```
{} schema

A versioned schema for every resource

Plugin ecosystem



CRD Aggregation

CRDs and API aggregation



Multi-tenancy



Moving faster



Kubernetes already had it all

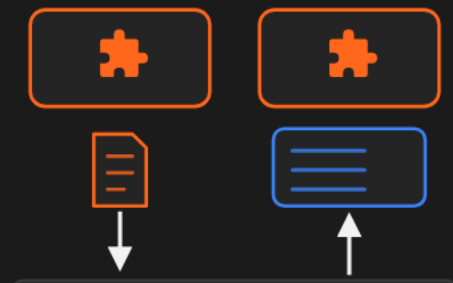
As code

```
apiVersion: …/v1
kind: Dashboard
```
schema

```
apiVersion: …/v1beta1
kind: AlertRule
```
schema

A versioned schema for every resource

Plugin ecosystem



CRDs and API aggregation

Multi-tenancy

Grafana

- stacks-1
- stacks-2
- stacks-3
- stacks-4

Namespaces



Moving faster

Kubernetes already had it all

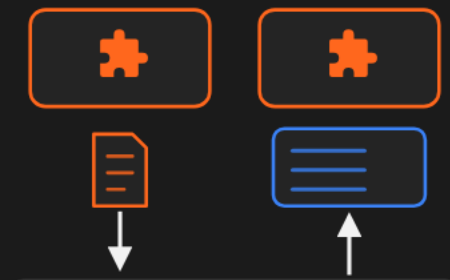
As code

```
apiVersion: …/v1
kind: Dashboard
```
{} schema

```
apiVersion: …/v1beta1
kind: AlertRule
```
{} schema

A versioned schema for every resource

Plugin ecosystem



CRDs and API aggregation

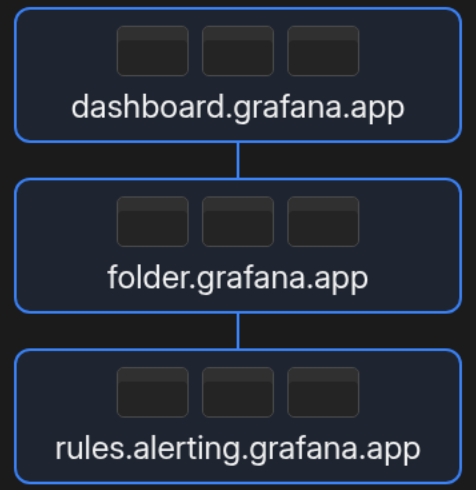
Multi-tenancy

Grafana

- stacks-1
- stacks-2
- stacks-3
- stacks-4

Namespaces

Moving faster



Team-owned API groups





One consistent API model

Before

```
POST  /api/dashboards/db
POST  /api/folders
POST  /api/ruler/grafana/api/v1/rules/:folder
POST  /api/v1/provisioning/contact-points
POST  /api/datasources
POST  /api/playlists
```

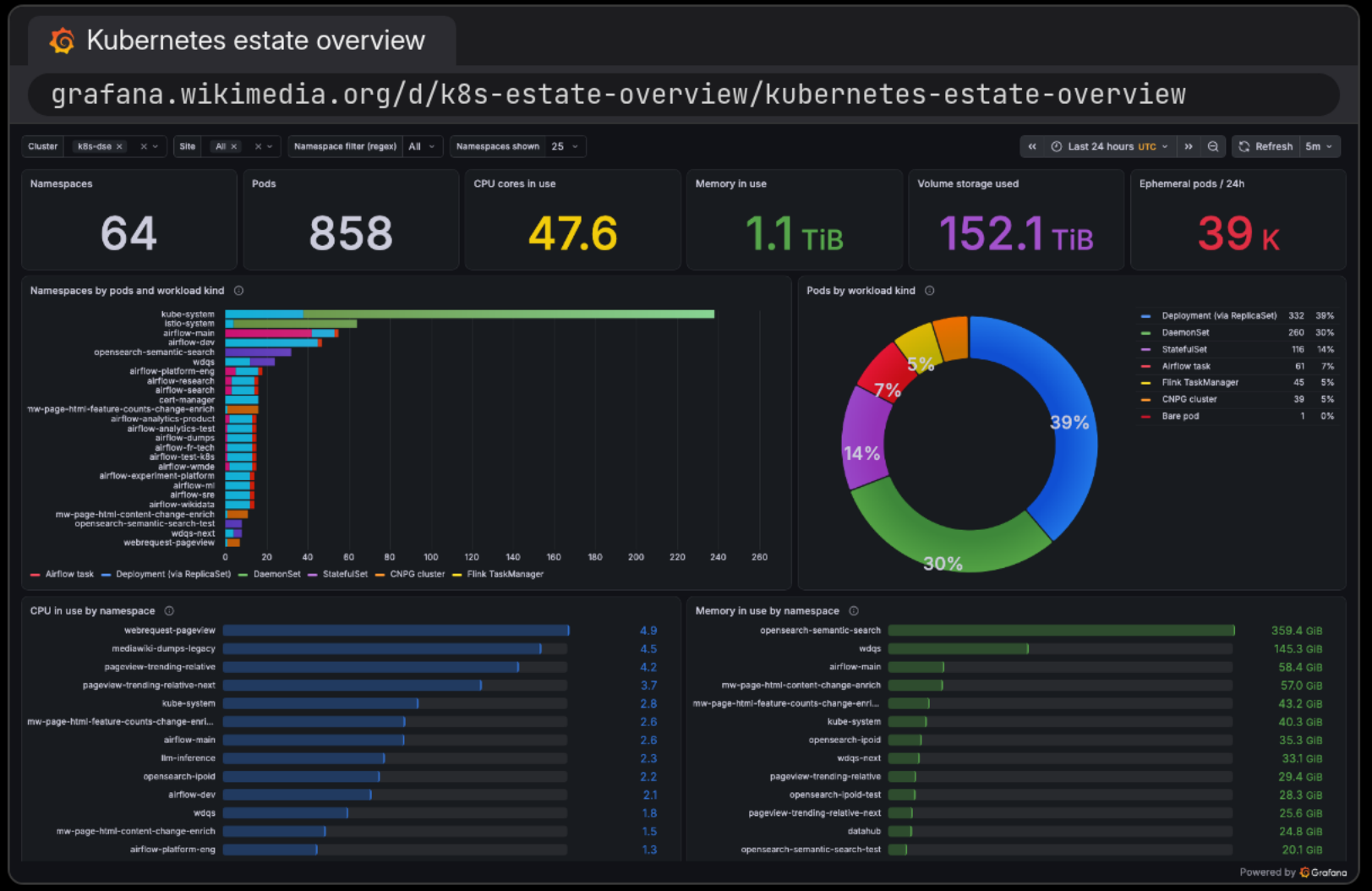
many ways to version, validate and authorize

One consistent API model

/apis/	<group>/	<version>/	namespaces/	<namespace>/	<resource>/	<name>
/apis/	dashboard.grafana.app/	v1/	namespaces/	default/	dashboards/	:uid
/apis/	folder.grafana.app/	v1/	namespaces/	default/	folders/	:uid
/apis/	playlist.grafana.app/	v1/	namespaces/	default/	playlists/	:uid

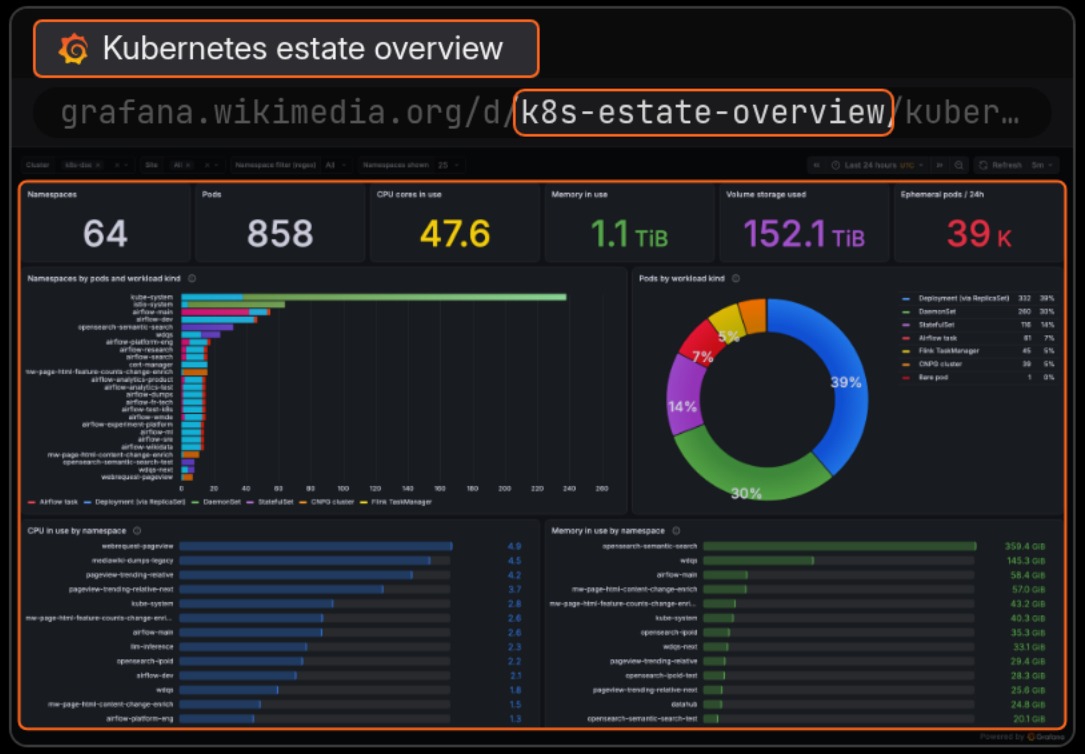
one way to version, validate and authorize

Every dashboard became a resource



Dashboard: grafana.wikimedia.org

Every dashboard became a resource




```yaml
apiVersion: dashboard.grafana.app/v1

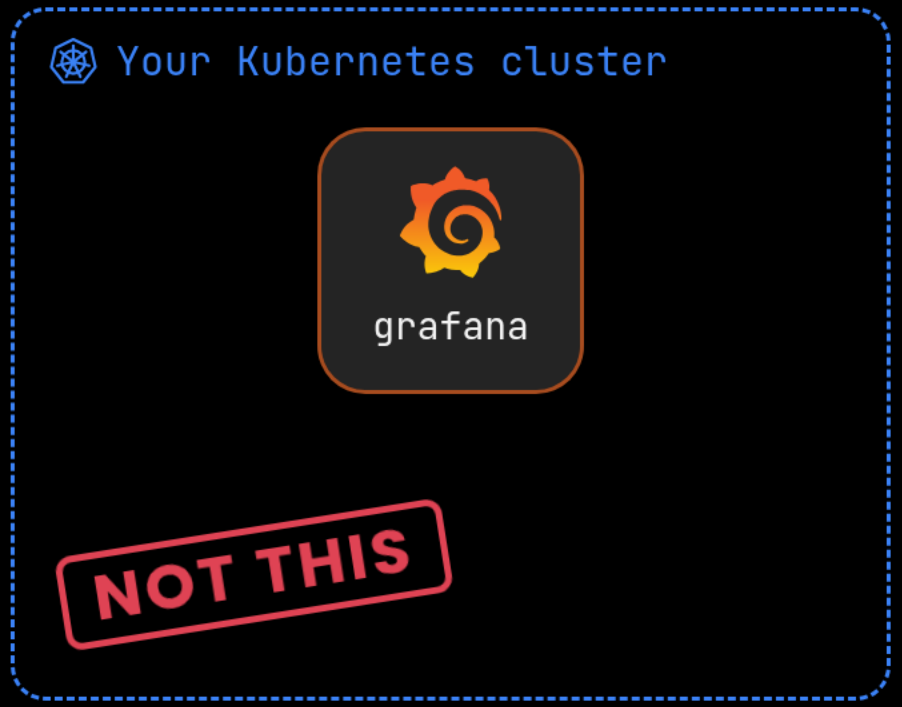
kind: Dashboard

metadata:
  name: k8s-estate-overview
  namespace: default

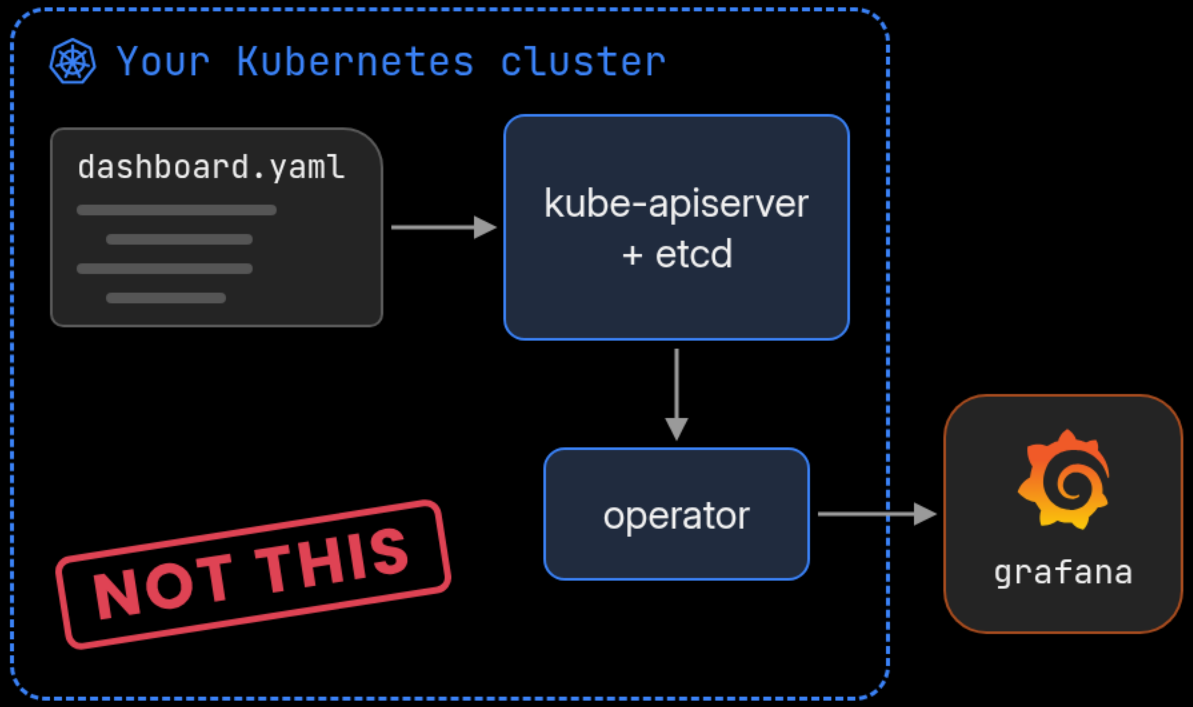
spec:
  title: Kubernetes estate overview
  panels: [ … ]   # 16 panels
```

Dashboard: grafana.wikimedia.org

Not on a cluster. Not an operator.

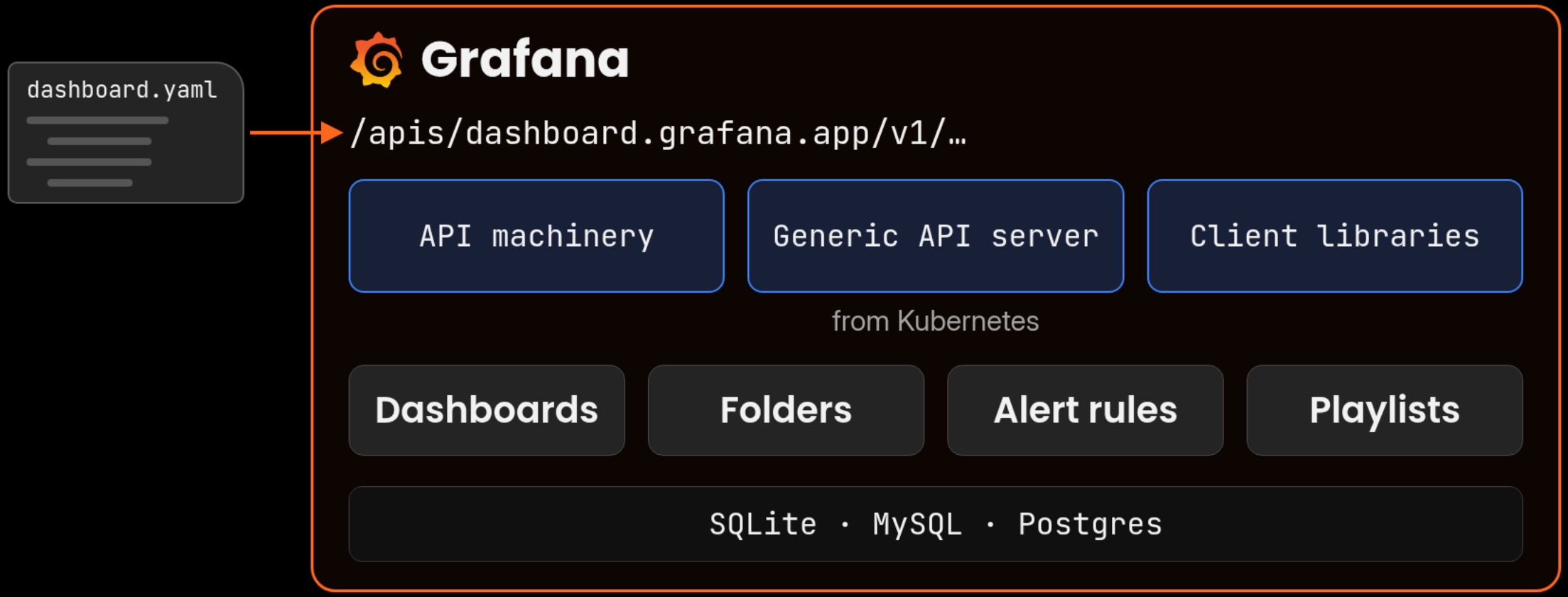


Grafana running on Kubernetes



An operator syncing CRDs into Grafana

Not on a cluster. Not an operator.



We have company

NAME	APIVERSION	NAMESPACED	KIND
cmds	tilt.dev/v1alpha1	false	Cmd
filewatches	tilt.dev/v1alpha1	false	FileWatch
uiresources	tilt.dev/v1alpha1	false	UIResource

Tilt



We have company

NAME	APIVERSION	NAMESPACED	KIND	
cmds	tilt.dev/v1alpha1	false	Cmd	Tilt
filewatches	tilt.dev/v1alpha1	false	FileWatch	
uiresources	tilt.dev/v1alpha1	false	UIResource	
workspaces	tenancy.kcp.io/v1alpha1	false	Workspace	kcp
routes	route.openshift.io/v1	true	Route	OpenShift



We have company

NAME	APIVERSION	NAMESPACED	KIND	
cmds	tilt.dev/v1alpha1	false	Cmd	Tilt
filewatches	tilt.dev/v1alpha1	false	FileWatch	
uiresources	tilt.dev/v1alpha1	false	UIResource	
workspaces	tenancy.kcp.io/v1alpha1	false	Workspace	kcp
routes	route.openshift.io/v1	true	Route	OpenShift
dashboards	dashboard.grafana.app/v1	true	Dashboard	Grafana
folders	folder.grafana.app/v1	true	Folder	
playlists	playlist.grafana.app/v1	true	Playlist	



What you'll leave with

A rubric for which Kubernetes pattern fits each part of an app.

If this part of your app…	Use…
Resource with desired state	?
CRUD with custom storage, AuthZ or secrets	?
Isn't a Resource	?



Part 1

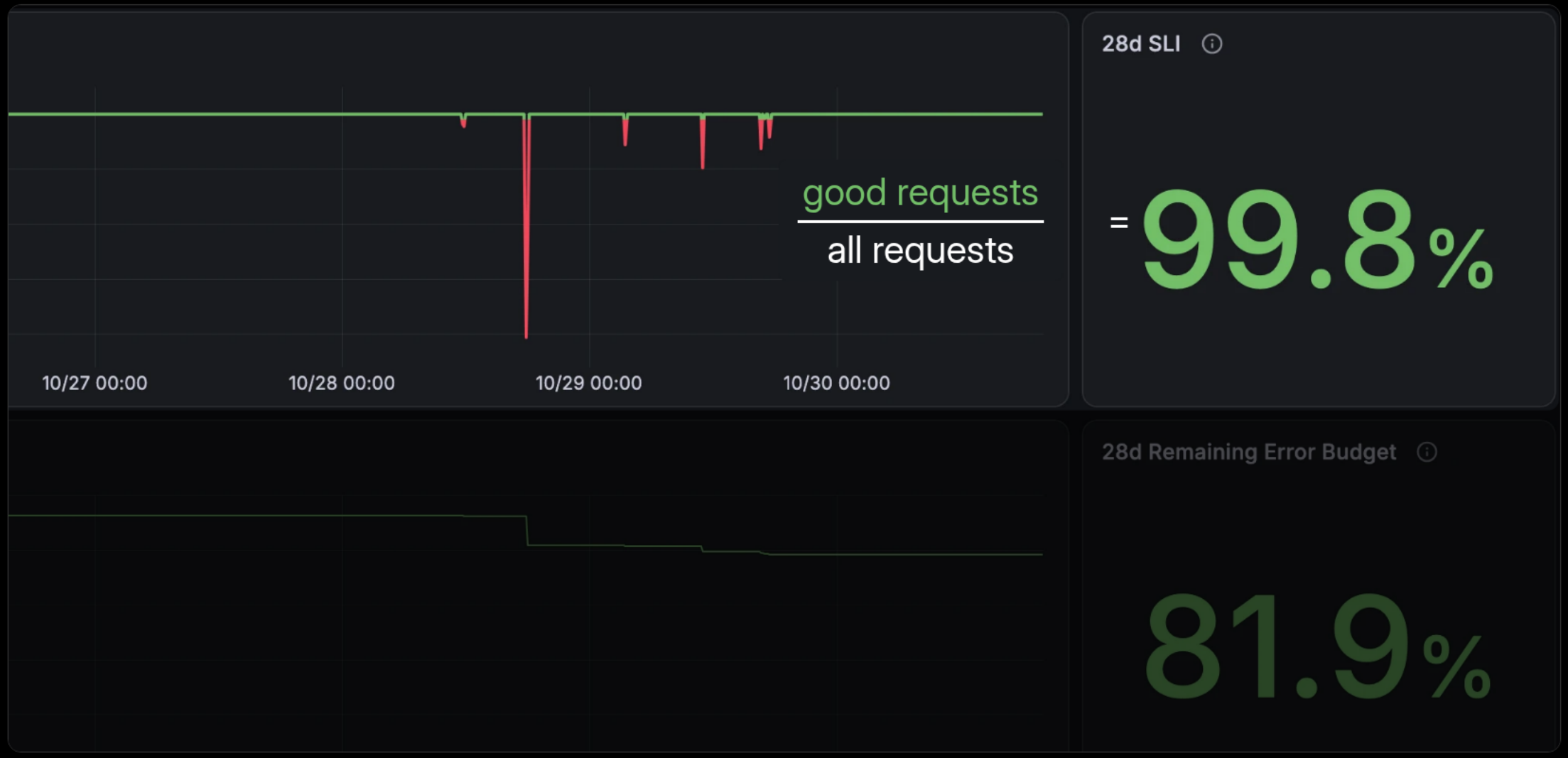
Starting small

Building a new app on vanilla Kubernetes

What is an SLO?



What is an SLO?



What is an SLO?



What is an SLO?



Grafana SLO

Query

```
sum(rate(checkout_ok_total[$__rate_interval]))
  / sum(rate(checkout_total[$__rate_interval]))
```

Objective

99% over 28 days

Grafana SLO

Query

```
sum(rate(checkout_ok_total[$__rate_interval]))
  / sum(rate(checkout_total[$__rate_interval]))
```

Objective

99% over 28 days

Dashboard



Grafana SLO

Query

```
sum(rate(checkout_ok_total[$__rate_interval]))
  / sum(rate(checkout_total[$__rate_interval]))
```

Objective

99% over 28 days

Dashboard



Alert rules

Checkout availability - Error Budget Burn Rate is Very High

Grafana SLO › Checkout availability (Grafana SLO) | grafanacloud-play-prom

Checkout availability - Error Budget Burn Rate is High

Grafana SLO › Checkout availability (Grafana SLO) | grafanacloud-play-prom

An SLO is a resource with a desired state

Query

```
sum(rate(checkout_ok_total[$__rate_interval]))
  / sum(rate(checkout_total[$__rate_interval]))
```

Objective

99% over 28 days

Dashboard



Alert rules

- Checkout availability - Error Budget Burn Rate is Very High
  Grafana SLO › Checkout availability (Grafana SLO) | grafanacloud-play-prom
- Checkout availability - Error Budget Burn Rate is High
  Grafana SLO › Checkout availability (Grafana SLO) | grafanacloud-play-prom

An SLO is a resource with a desired state

```yaml
apiVersion: grafana-slo-app.plugins.grafana.com/v0-0
kind: Slo
metadata:
  name: checkout-availability
spec:
  description: Customers can always check out
  query:
    type: freeform
    freeform:
      query: >-
        sum(rate(checkout_ok_total[$__rate_interval]))
        / sum(rate(checkout_total[$__rate_interval]))
  objectives:
    - value: 0.99
      window: 28d
```

Dashboard



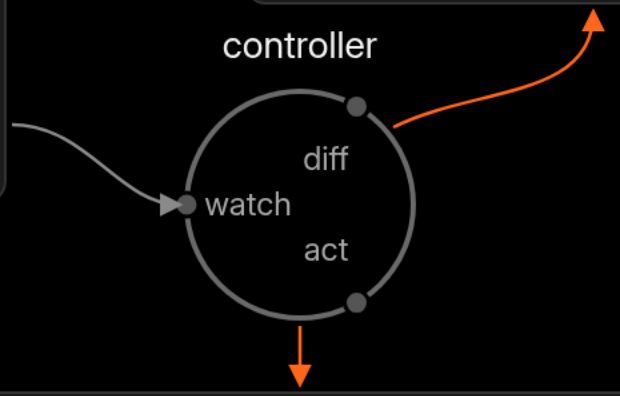
Alert rules

Checkout availability - Error Budget Burn Rate is Very High
Grafana SLO › Checkout availability (Grafana SLO) | grafanacloud-play-prom

Checkout availability - Error Budget Burn Rate is High
Grafana SLO › Checkout availability (Grafana SLO) | grafanacloud-play-prom

An SLO is a resource with a desired state

```yaml
apiVersion: grafana-slo-app.plugins.grafana.com/v0-0
kind: Slo
metadata:
  name: checkout-availability
spec:
  description: Customers can always check out
  query: { … }
  objectives:
    - value: 0.99
      window: 28d
```




```yaml
apiVersion: dashboard.grafana.app/v1
kind: Dashboard
spec:
  title: Checkout availability SLO
  panels: [ … ]
```

```yaml
apiVersion: rules.alerting.grafana.app/v1beta1
kind: AlertRule
spec:
  title: Checkout availability - Error Budget Burn Rate is Very High
  trigger:
    interval: 1m
  expressions: { … }
  noDataState: NoData
  execErrState: Error
```

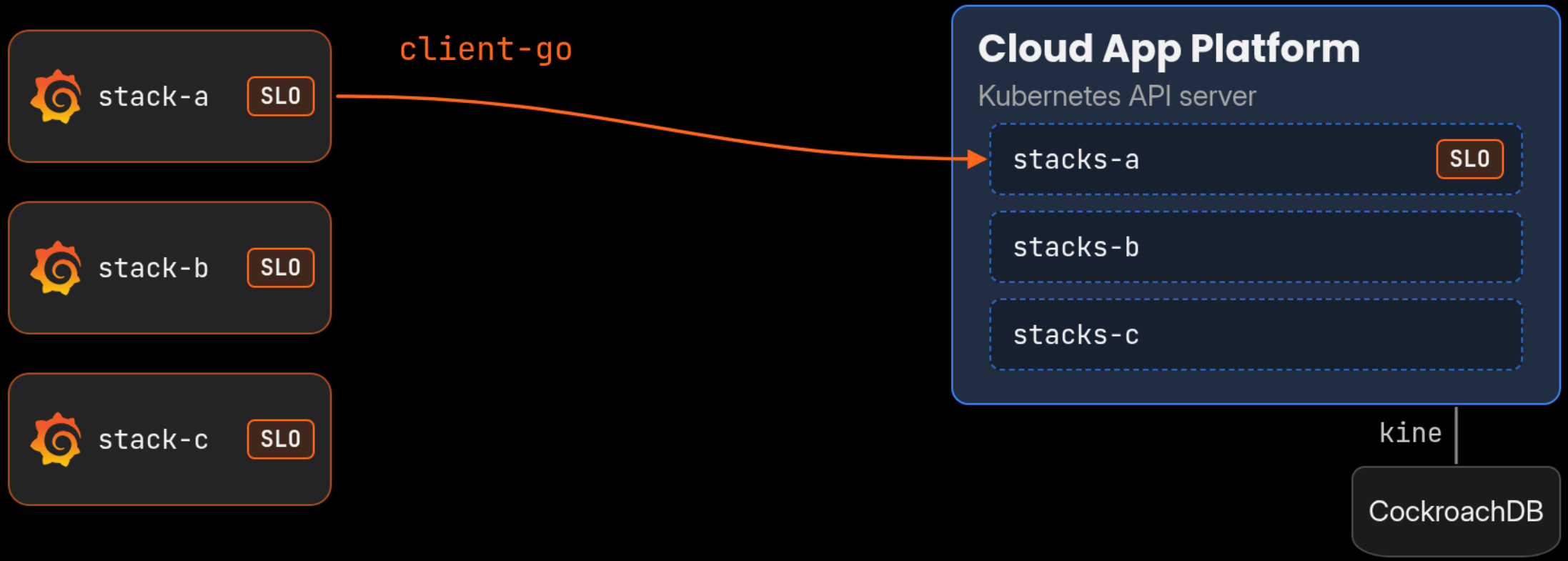
Desired state? A CRD and a controller.

If this part of your app…	Use…
Resource with desired state	CRD + Controller
CRUD with custom storage, AuthZ or secrets	?
Isn't a Resource	?

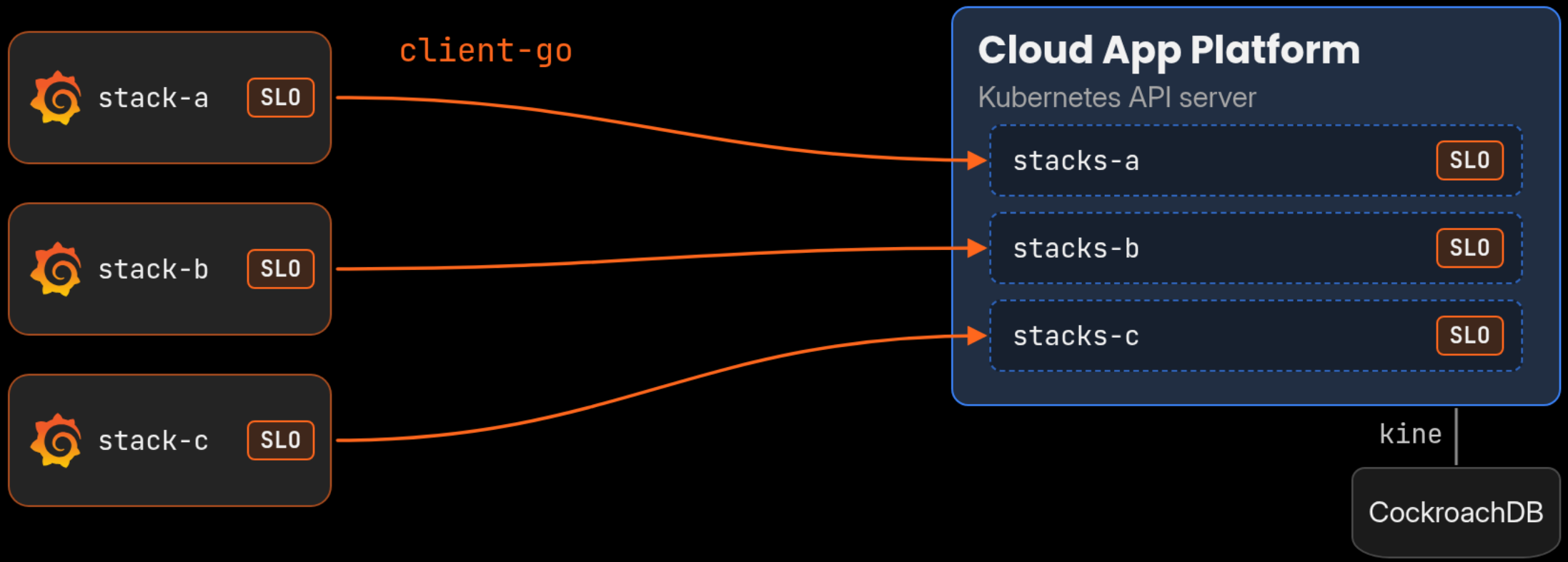
Cloud App Platform



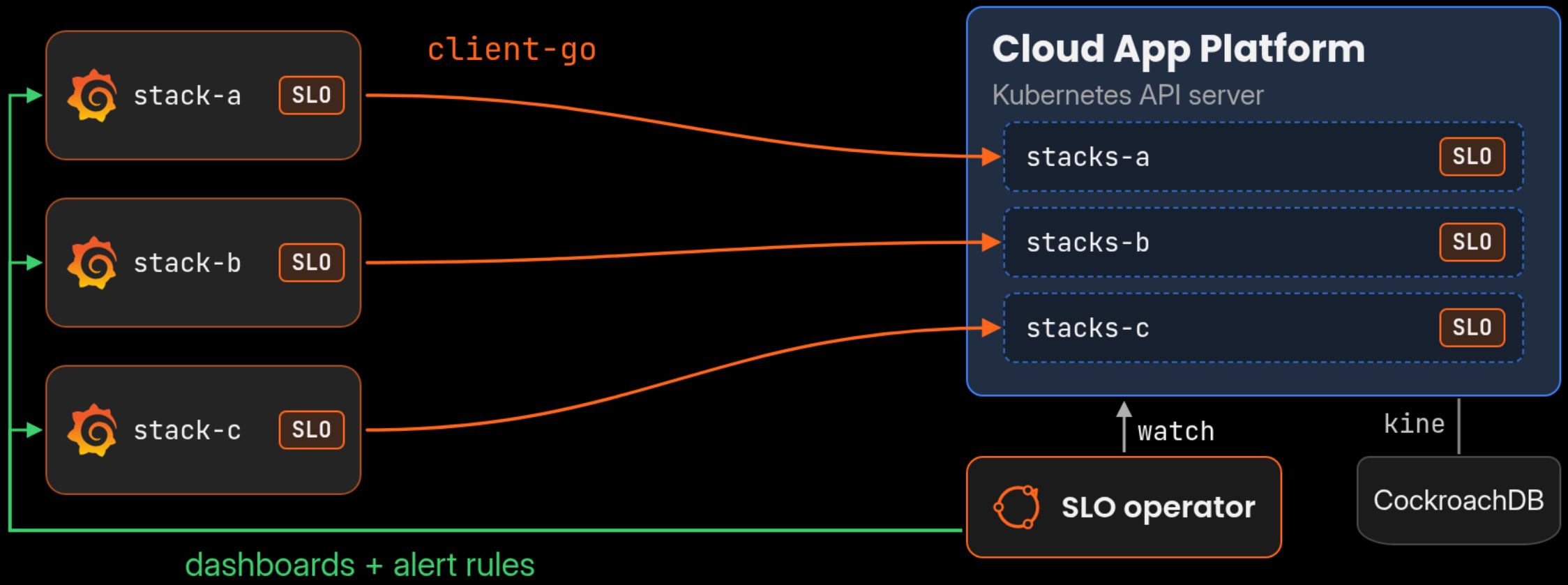
Cloud App Platform



Cloud App Platform



Cloud App Platform



Well, now there are two.

Cloud App Platform		Grafana backend
Grafana Cloud only	Ships to	OSS, Enterprise, Cloud
Viewer · Editor · Admin	Permissions	Grafana RBAC
Not built in	Folders	Built in, with permissions
Managed by the platform	Storage	Custom SQL tables per feature



PART 2

Bringing it into Grafana

The road to dashboards v1

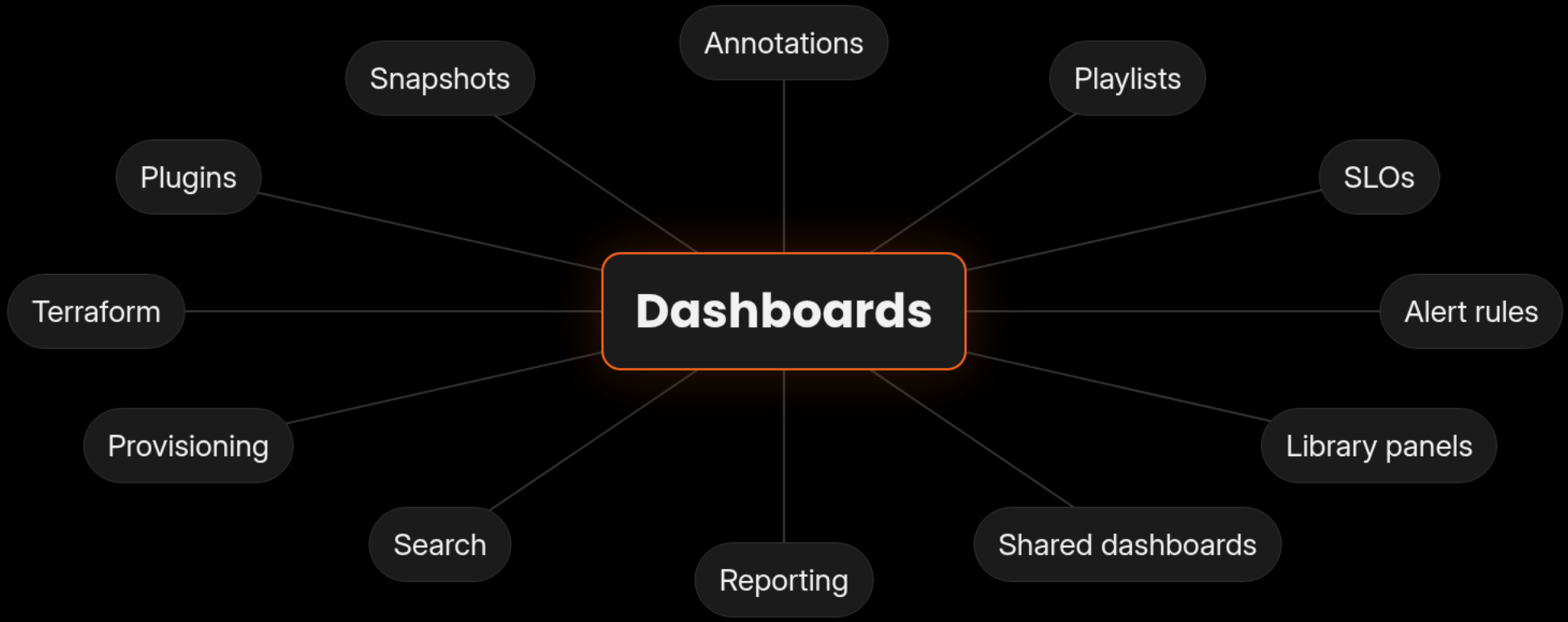
42 schema versions, piling up

Every old version still has to open.

v1	v2	v3	v4	v5	v6	v7	v8	v9	v10	v11	v12	v13	v14	v15
v16	v17	v18	v19	v20	v21	v22	v23	v24	v25	v26	v27	v28	v29	v30
v31	v32	v33	v34	v35	v36	v37	v38	v39	v40	v41	v42			

CODE grafana/grafana › apps/dashboard/…/schemaversion ↗

Everything depends on dashboards



CODE grafana/grafana › apps/dashboard/…/schemaversion ↗

One goal: dashboards v1

1 · REHEARSE

Playlists

Smallest resource, low blast radius

HAD TO REPLACE

Storage

2 · SHIP

Dashboards v1

The resource everything depends on

HAD TO REPLACE

Authorization

3 · DELIVER

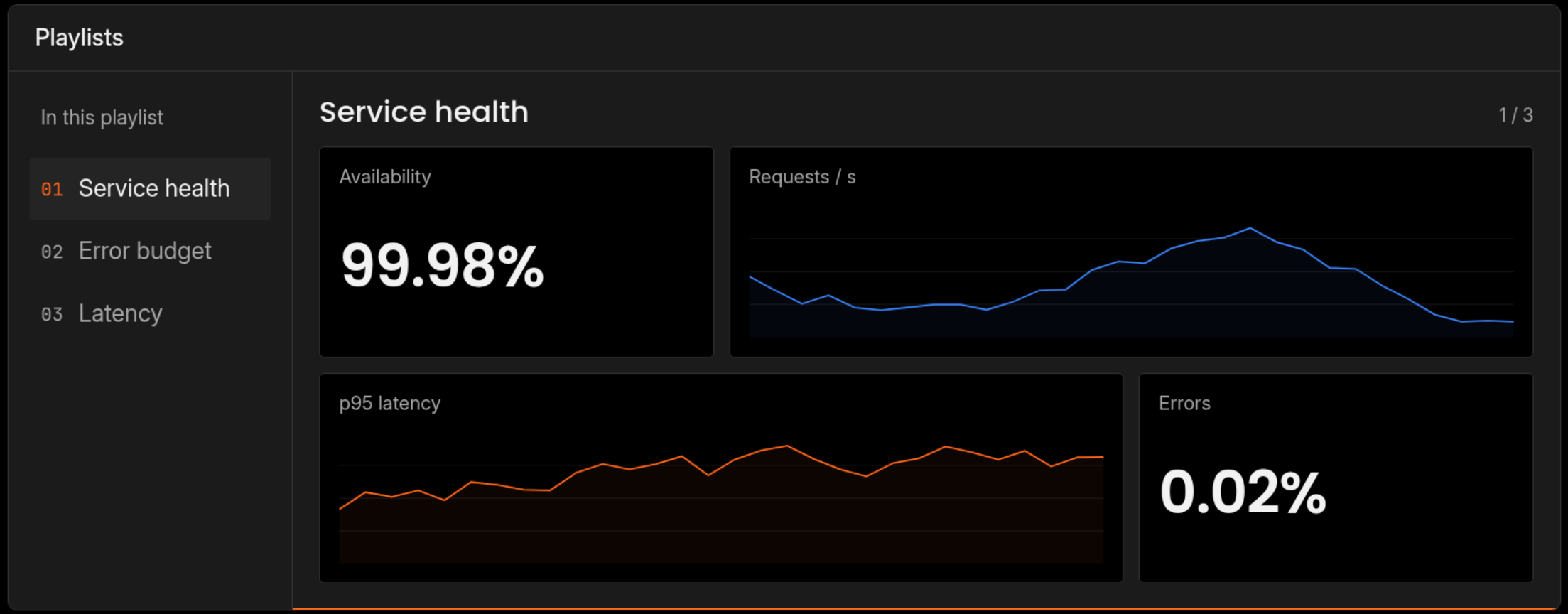
Git Sync

Our most requested feature

HAD TO REPLACE

Secrets

Proving an end-to-end path



CODE grafana/grafana › apps/playlist ↗

Why not etcd?

etcd

Lists by key prefix only

```
/registry/…/playlists/stack-1/*
```

No secondary indexes

Grafana needs

- Filter by folder
- Filter by permission
- Sort and page
- 100,000s of objects

So we built unified storage

Unified storage

Indexed tables on Grafana's own databases

SQLite · MySQL · Postgres

Same Kubernetes API in front

Grafana needs

- ✓ Filter by folder
- ✓ Filter by permission
- ✓ Sort and page
- ✓ 100,000s of objects

Swapping out etcd, in code

pkg/storage/unified/apistore/store.go

```go
var _ storage.Interface = (*Storage)(nil)

type Storage struct {
  gr           schema.GroupResource
  keyFunc      func(obj runtime.Object) (string, error)
  newFunc      func() runtime.Object
  newListFunc  func() runtime.Object
  getAttrsFunc storage.AttrFunc
  // ...

  store          resourceclient.ResourceClient
  getKey         func(string) (*resourcepb.ResourceKey, error)
  // ...
}
```

Step 1 of 2

Implement storage.Interface

The same interface the etcd3 store implements.

CODE grafana/grafana › apistore/store.go ↗ resource.proto ↗

Swapping out etcd, in code

pkg/storage/unified/apistore/store.go

```go
func (s *Storage) GetList(ctx context.Context, key string,
  opts storage.ListOptions, listObj runtime.Object) error {
  ctx, span := tracer.Start(ctx, "apistore.Storage.GetList")
  defer span.End()
  k, err := s.getKey(key)
  // ...
  req, predicate, err := toListRequest(k, opts)
  // ...
  rsp, err := s.store.List(ctx, req)
  if err := resourceutil.ErrorFromResponse(rsp.GetError(), err);
    err ≠ nil {
    return statusErrorFromStorageError(err)
  }
  // ...
}
```

Step 2 of 2

A list becomes a gRPC call

ListOptions turn into a ListRequest.

What is actually in the database

resource · current state

group	resource	namespace	name	resource_version	folder	value
playlist.grafana.app	playlists	stacks-1234	**on-call-wall**	1793178843482117	''	{"spec":{"interval":"5m",…}}
dashboard.grafana.app	dashboards	stacks-1234	**service-health**	1793116975913840	team-sre	{"spec":{"title":"Service health",…}}
folder.grafana.app	folders	stacks-1234	**team-sre**	1792583300245519	''	{"spec":{"title":"Team SRE",…}}

CODE grafana/grafana › migrations/resource_mig.go ↗ sql/backend.go ↗ resource_trash.sql ↗

What is actually in the database

resource · current state

group	resource	namespace	name	resource_version	folder	value
playlist.grafana.app	playlists	stacks-1234	**on-call-wall**	1793179241007353	''	{"spec":{"interval":"10m",…}}
dashboard.grafana.app	dashboards	stacks-1234	**service-health**	1793116975913840	team-sre	{"spec":{"title":"Service health",…}}
folder.grafana.app	folders	stacks-1234	**team-sre**	1792583300245519	''	{"spec":{"title":"Team SRE",…}}

resource_history · every version

group	resource	namespace	name	resource_version	action	value
playlist.grafana.app	playlists	stacks-1234	**on-call-wall**	1793178843482117	1 · created	{"spec":{"interval":"5m",…}}
playlist.grafana.app	playlists	stacks-1234	**on-call-wall**	1793179241007353	2 · updated	{"spec":{"interval":"10m",…}}

CODE grafana/grafana › migrations/resource_mig.go ↗ sql/backend.go ↗ resource_trash.sql ↗

What is actually in the database

resource · current state

group	resource	namespace	name	resource_version	folder	value
~~playlist.grafana.app~~	~~playlists~~	~~stacks-1234~~	~~on-call-wall~~	~~1793179241007353~~	~~''~~	~~{"spec":{"interval":"10m",…}}~~
dashboard.grafana.app	dashboards	stacks-1234	**service-health**	1793116975913840	team-sre	{"spec":{"title":"Service health",…}}
folder.grafana.app	folders	stacks-1234	**team-sre**	1792583300245519	''	{"spec":{"title":"Team SRE",…}}

resource_history · every version

group	resource	namespace	name	resource_version	action	value
playlist.grafana.app	playlists	stacks-1234	**on-call-wall**	1793178843482117	1 · created	{"spec":{"interval":"5m",…}}
playlist.grafana.app	playlists	stacks-1234	**on-call-wall**	1793179241007353	2 · updated	{"spec":{"interval":"10m",…}}
playlist.grafana.app	playlists	stacks-1234	**on-call-wall**	1793179872660284	3 · deleted	{"spec":{"interval":"10m",…}}

CODE grafana/grafana › migrations/resource_mig.go ↗ sql/backend.go ↗ resource_trash.sql ↗

What is actually in the database

resource · current state

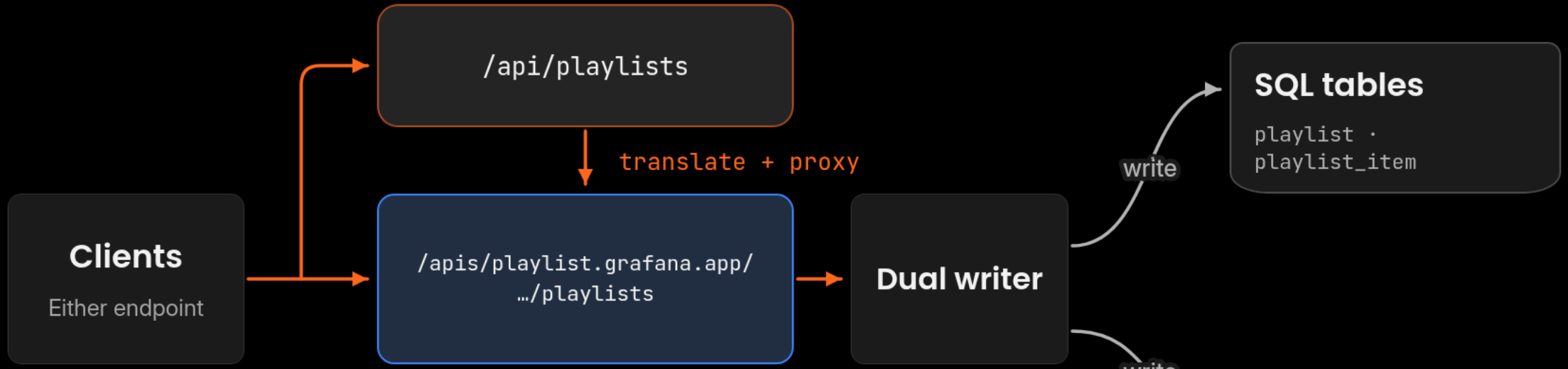
group	resource	namespace	name	resource_version	folder	value
~~playlist.grafana.app~~	~~playlists~~	~~stacks-1234~~	~~on-call-wall~~	~~1793179241007353~~	~~''~~	~~{"spec":{"interval":"10m",…}}~~
dashboard.grafana.app	dashboards	stacks-1234	**service-health**	1793116975913840	team-sre	{"spec":{"title":"Service health",…}}
folder.grafana.app	folders	stacks-1234	**team-sre**	1792583300245519	''	{"spec":{"title":"Team SRE",…}}
provisioning.grafana.app	repositories	stacks-1234	**git-sync**	1793180557118902	''	{"spec":{"type":"github",…}}

resource_history · every version

group	resource	namespace	name	resource_version	action	value
playlist.grafana.app	playlists	stacks-1234	**on-call-wall**	1793178843482117	1 · created	{"spec":{"interval":"5m",…}}
playlist.grafana.app	playlists	stacks-1234	**on-call-wall**	1793179241007353	2 · updated	{"spec":{"interval":"10m",…}}
playlist.grafana.app	playlists	stacks-1234	**on-call-wall**	1793179872660284	3 · deleted	{"spec":{"interval":"10m",…}}
provisioning.grafana.app	repositories	stacks-1234	**git-sync**	1793180557118902	1 · created	{"spec":{"type":"github",…}}

CODE grafana/grafana › migrations/resource_mig.go ↗ sql/backend.go ↗ resource_trash.sql ↗

Same API, new storage

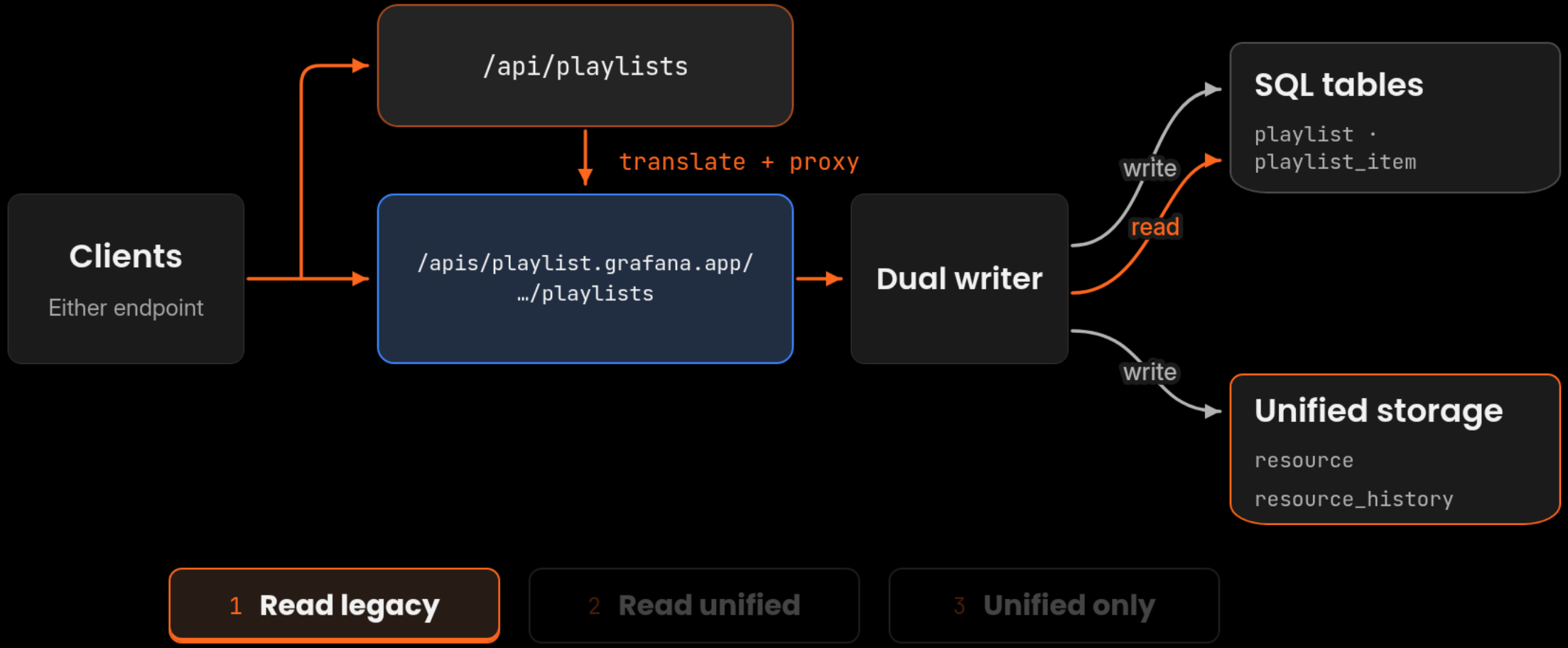


1 Read legacy

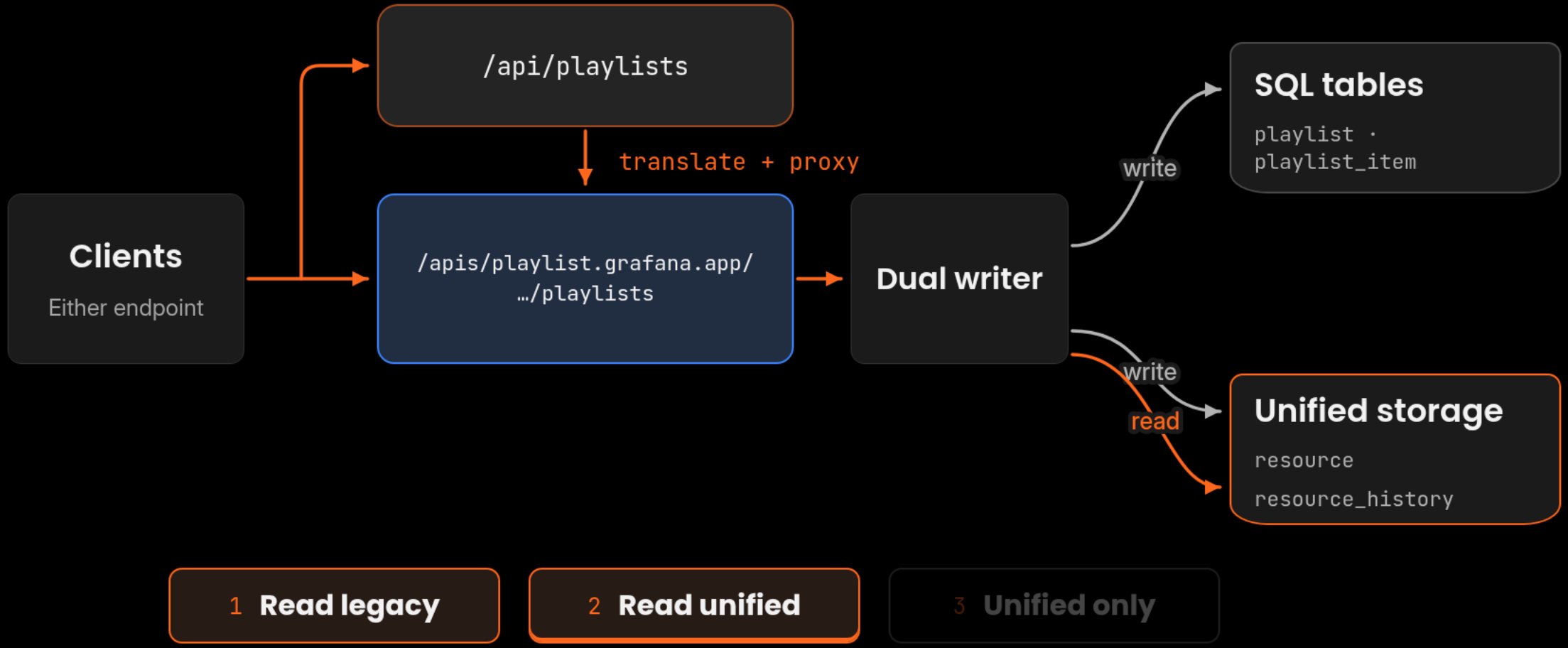
2 Read unified

3 Unified only

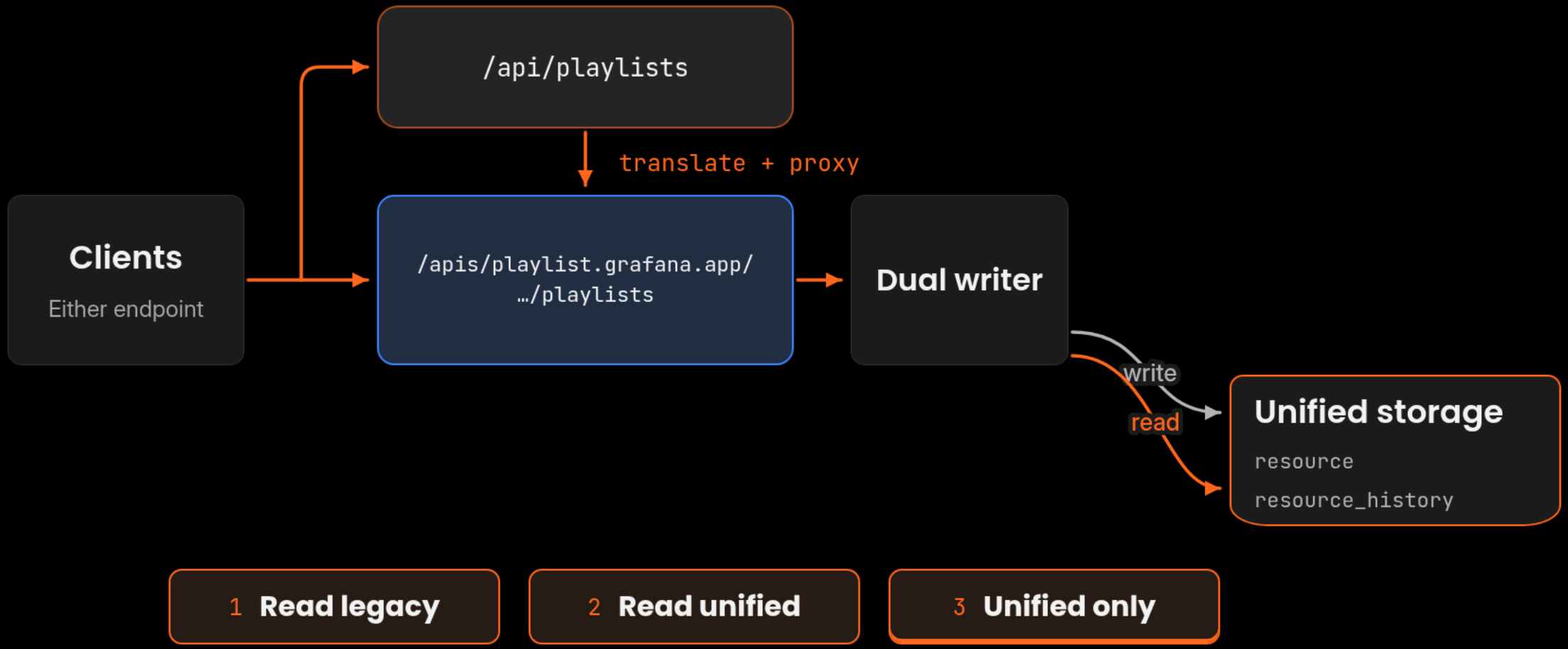
Same API, new storage



Same API, new storage



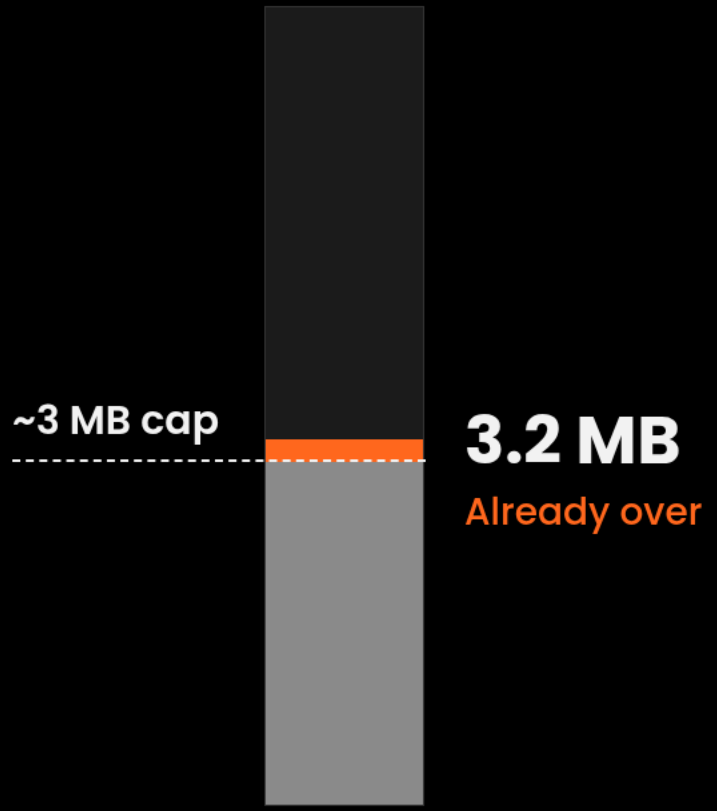
Same API, new storage



Where we hit Kubernetes limits

service-health.json

```
{
  "apiVersion": "dashboard.grafana.app/v1",
  "kind": "Dashboard",
  "metadata": { "name": "service-health" },
  "spec": {
    "title": "Service health",
    "panels": [
      {
        "id": 1,
        "type": "timeseries",
        "title": "p95 latency · api",
        "gridPos": { "h": 8, "w": 8, "x": 0, "y": 0 },
        "targets": [{ "expr": "histogram_quantile(…)" }],
        "fieldConfig": { "defaults": { "unit": "s", "thresholds"
```



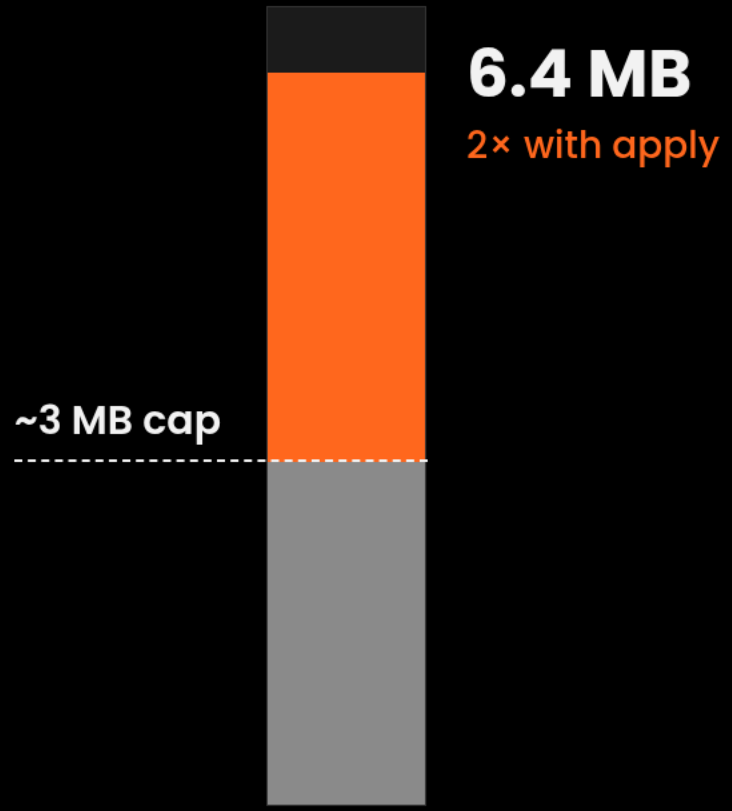
Where we hit Kubernetes limits

service-health.json

```
metadata:
  annotations:
    kubectl.kubernetes.io/last-applied-configuration: |
      {"apiVersion":"dashboard.grafana.app/v1", …
       every panel, again … }
```

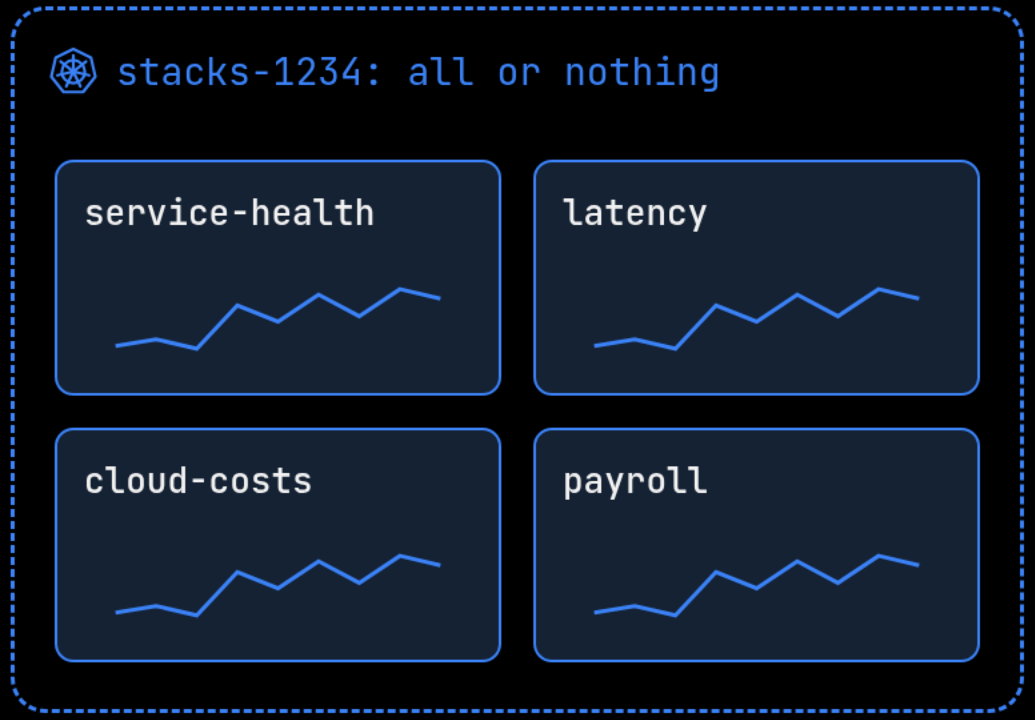
```
{
  "apiVersion": "dashboard.grafana.app/v1",
  "kind": "Dashboard",
  "metadata": { "name": "service-health" },
  "spec": {
    "title": "Service health",
    "panels": [
      {
```

```
$ kubectl apply -f service-health.json
```



Kubernetes RBAC doesn't know about folders

Kubernetes: namespace access



Grafana: Alice's folders

- ▾ team-sre/ — Alice: view
  - ▾ api/
    - · latency.json
    - · service-health.json
- ▾ finance/
  - · cloud-costs.json
  - · payroll.json

CODE grafana/grafana › pkg/services/apiserver/auth/authorizer ↗ pkg/services/authz/zanzana ↗

Storage checks every row against its folder

pkg/storage/unified/resource/server.go

```go
func listAuthorizationItem(key *resourcepb.ResourceKey,
  name, folder string) authz.BatchCheckItem {
  return authz.BatchCheckItem{
    Name:      name,
    Folder:    folder,
    Verb:      utils.VerbGet,
    Group:     key.Group,
    Resource:  key.Resource,
    Namespace: key.Namespace,
  }
}

// ... inside List:
for item, err := range authz.FilterAuthorized(ctx, s.access,
  candidates, extractFn, authz.WithTracer(tracer)) {
```



Authz deep dive

One check per row

`name + folder + verb`

Only allowed rows

`FilterAuthorized`

CODE grafana/grafana › unified/resource/server.go ↗

One resource, many versions

GET …/v1/…/dashboards/service-health

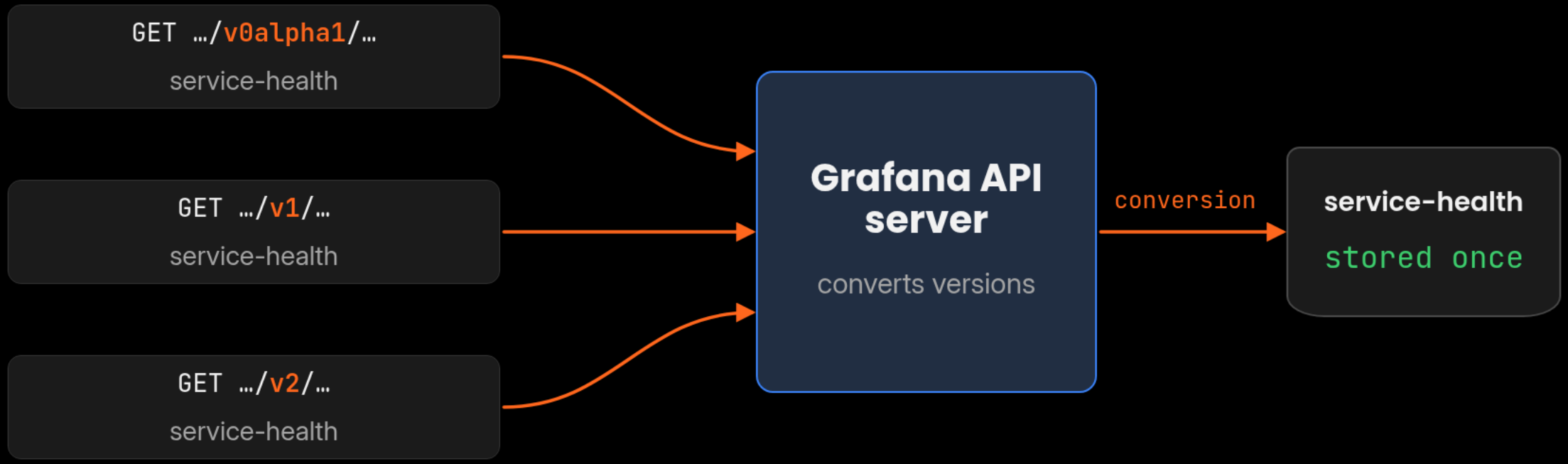
```
{
  "apiVersion": "dashboard.grafana.app/v1",
  "kind": "Dashboard",
  "spec": {
    "title": "Service health",
    "panels": [
      { "id": 1, "type": "timeseries",
        "gridPos": {"x":0,"y":0,"w":12,"h":8},
        "targets": [ … ] }
    ]
  }
}
```

GET …/v2/…/dashboards/service-health

```
{
  "apiVersion": "dashboard.grafana.app/v2",
  "kind": "Dashboard",
  "spec": {
    "title": "Service health",
    "elements": {
      "panel-1": { "kind": "Panel", … }
    },
    "layout": {
      "kind": "GridLayout",
      "items": [{ "element": "panel-1" }]
    }
  }
}
```

CODE grafana/grafana › apps/dashboard ↗

One resource, many versions



Before: 42 one-way migrations hidden inside the JSON

CODE grafana/grafana › apps/dashboard ↗

Kubernetes Secrets aren't secret

secret.yaml

```yaml
apiVersion: v1
kind: Secret
data:
  token: czNjcjN0
```

```
$ echo czNjcjN0 | base64 -d
s3cr3t
```



Blog post: how we built it

CODE grafana/grafana › apps/secret ↗ pkg/registry/apis/secret ↗

Kubernetes Secrets aren't secret

secret.yaml

```yaml
apiVersion: v1
kind: Secret
data:
  token: czNjcjN0
```

```
$ echo czNjcjN0 | base64 -d
s3cr3t
```



Blog post: how we built it

securevalue.yaml

```yaml
apiVersion: secret.grafana.app/v1beta1
kind: SecureValue
spec:
  value: ••••••••  write-only
```

repository.yaml

```yaml
kind: Repository
secure:
  token:
    name: repo-token  a reference
```

CODE grafana/grafana › apps/secret ↗ pkg/registry/apis/secret ↗

Three years to a stable v1

Year 1

Design

Schematize everything?

Year 2

Alpha

Dashboards follow
the playlist migration

Year 3

v1

A stable contract,
v2 right behind

Three years to a stable v1



Year 1

Design

Schematize everything?

Year 2

Alpha

Dashboards follow
the playlist migration

Year 3

v1

A stable contract,
v2 right behind

A migration is never a straight line.

It wasn’t all smooth sailing

Pulled

v13.0.0

Available

v13.0.1

Regression

grafana/grafana #122804 ↗

Dashboards reverted after the 13.0.0 upgrade

The GA migration cleared unified-storage state used by early Git Sync. Months of changes remained in Git; recovery required manual steps.

Fix

#122866 ↗

Migration no longer clears Git Sync state.



It wasn't all smooth sailing

Pulled

v13.0.0

Available

v13.0.1

Regression — `grafana/grafana #122804` ↗

Dashboards reverted after the 13.0.0 upgrade

The GA migration cleared unified-storage state used by early Git Sync. Months of changes remained in Git; recovery required manual steps.

Fix — `#122866` ↗

Migration no longer clears Git Sync state.

What we learned



Chaos-test your flags

It wasn't all smooth sailing

Pulled

v13.0.0

Available

v13.0.1

Regression

`grafana/grafana #122804` ↗

Dashboards reverted after the 13.0.0 upgrade

The GA migration cleared unified-storage state used by early Git Sync. Months of changes remained in Git; recovery required manual steps.

Fix

`#122866` ↗

Migration no longer clears Git Sync state.

What we learned



Chaos-test your flags

Clean up legacy early

It wasn't all smooth sailing

Pulled

v13.0.0

Available

v13.0.1

Regression `grafana/grafana #122804` ↗

Dashboards reverted after the 13.0.0 upgrade

The GA migration cleared unified-storage state used by early Git Sync. Months of changes remained in Git; recovery required manual steps.

Fix `#122866` ↗

Migration no longer clears Git Sync state.

What we learned

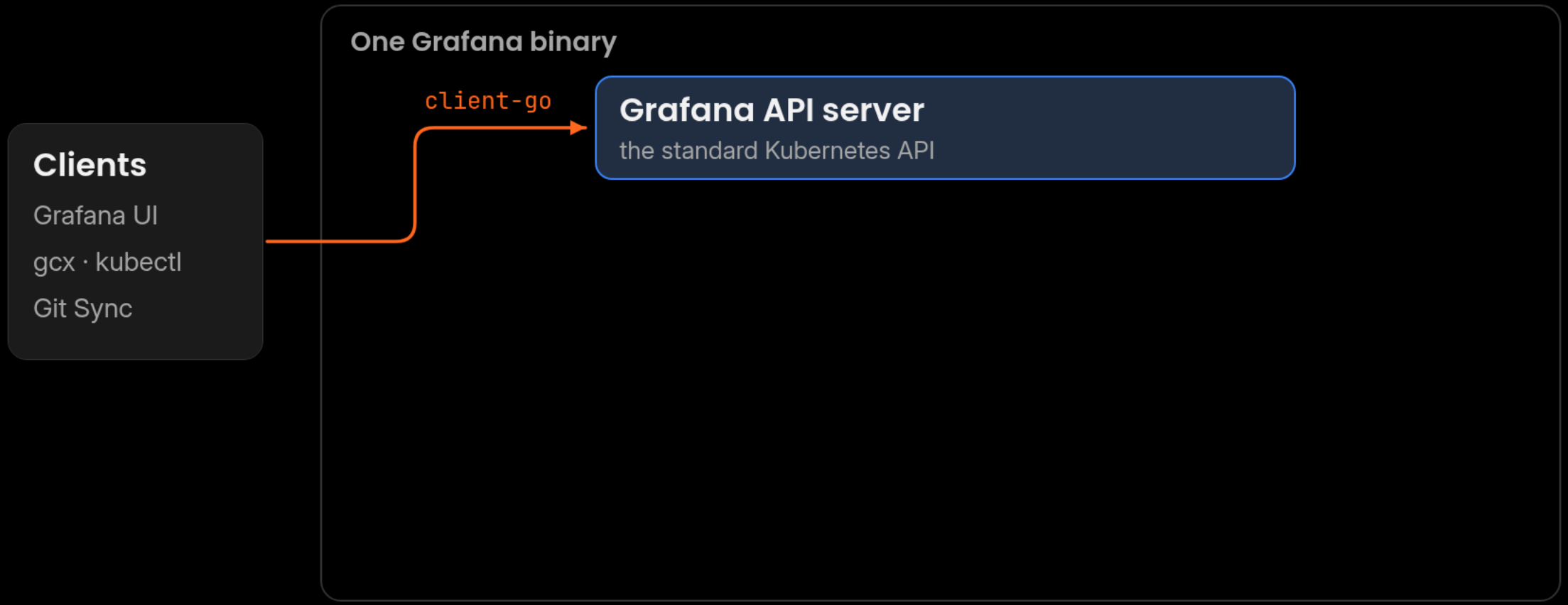


Chaos-test your flags

Clean up legacy early

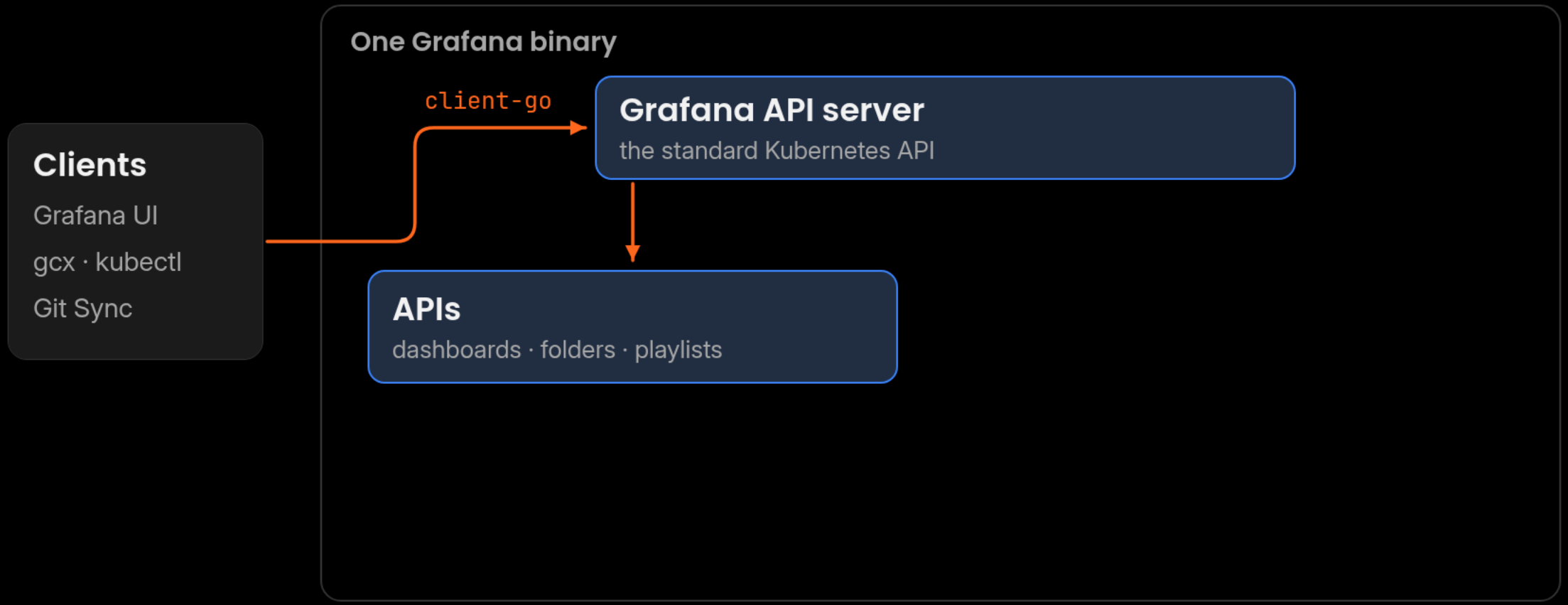
Black-box end-to-end tests

Grafana App Platform



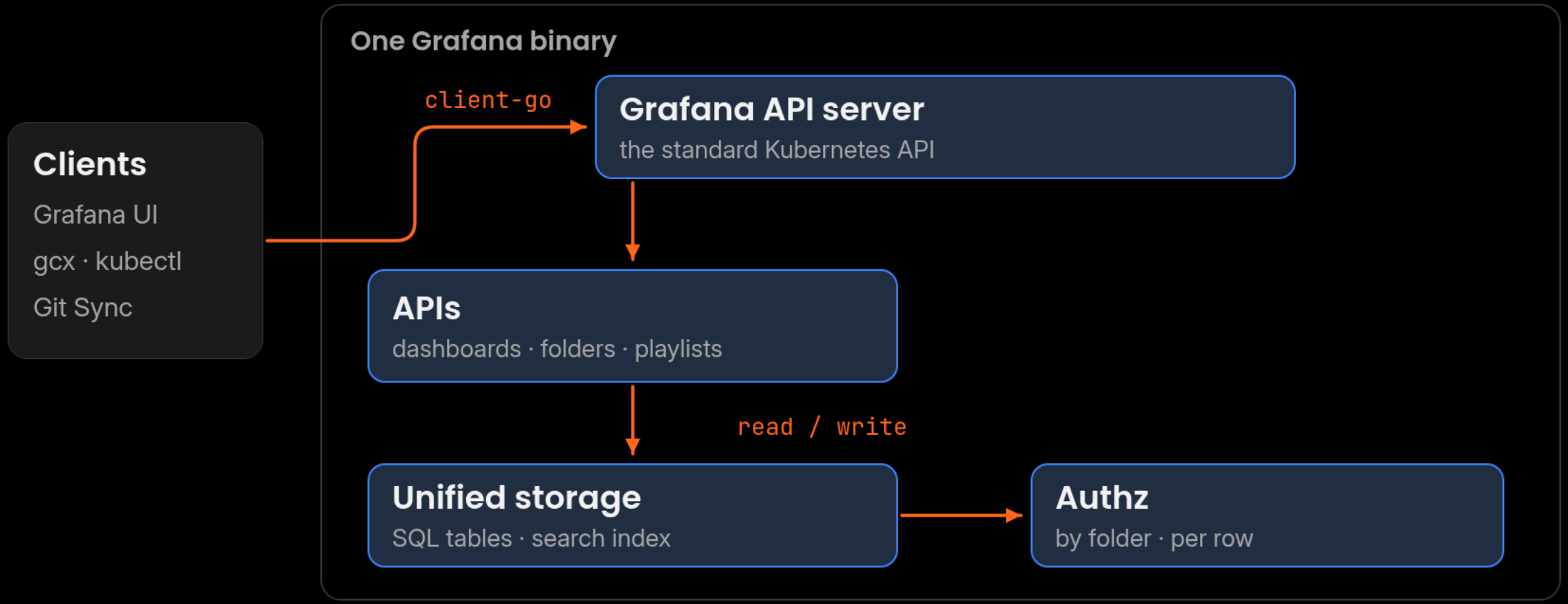
CODE grafana/grafana › pkg/services/apiserver ↗

Grafana App Platform



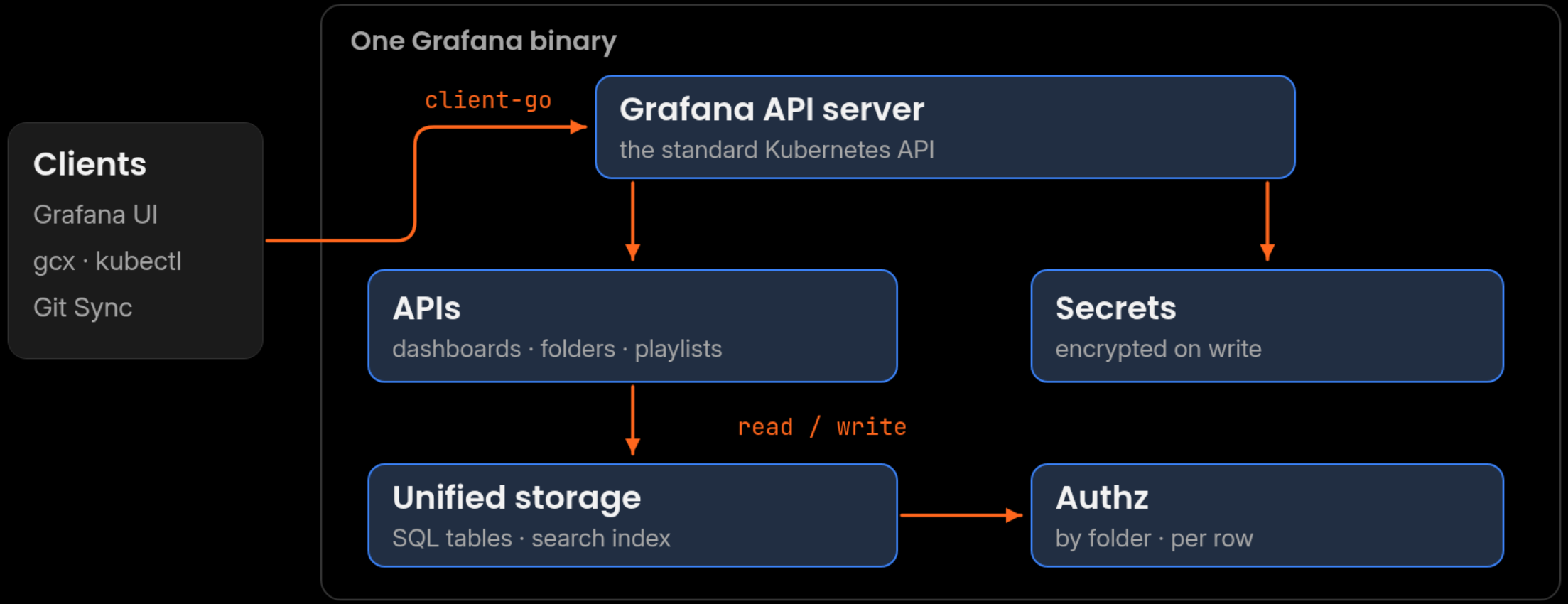
CODE grafana/grafana › pkg/services/apiserver ↗

Grafana App Platform



CODE grafana/grafana › pkg/services/apiserver ↗

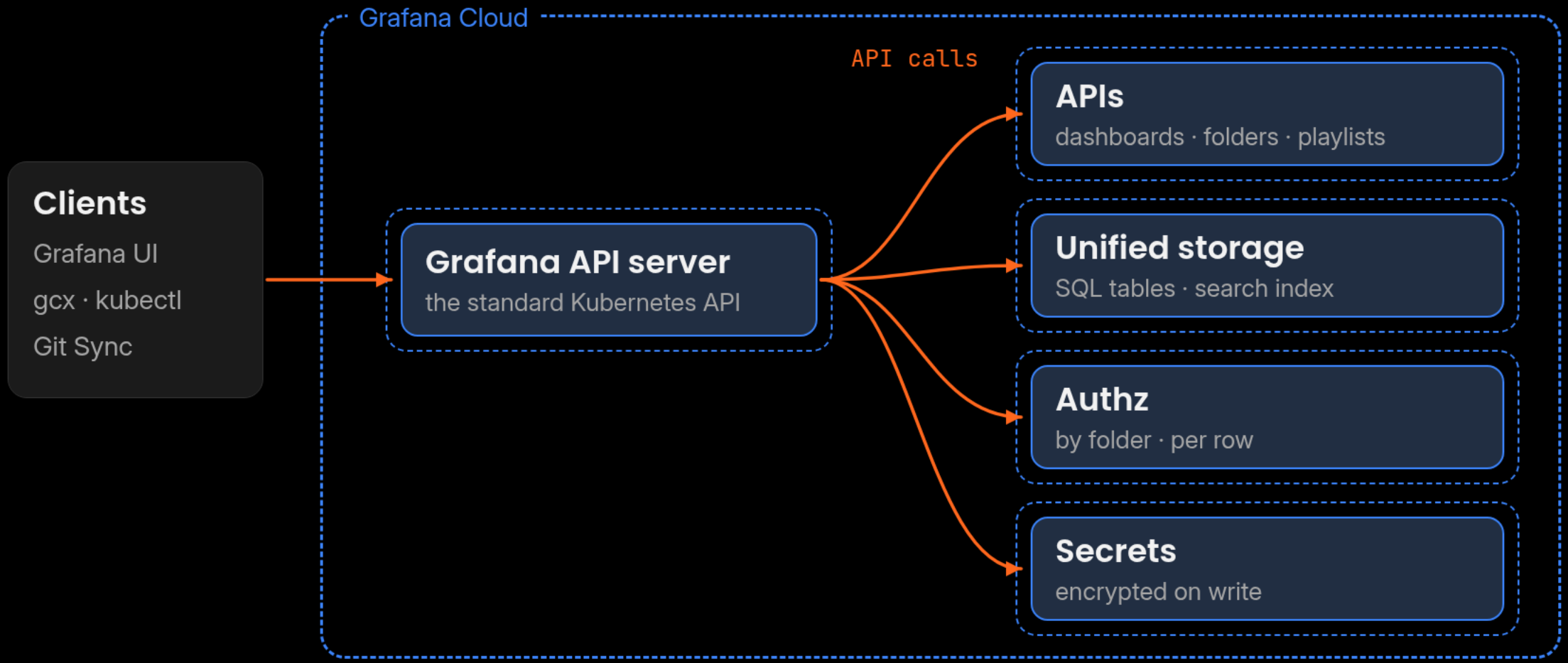
Grafana App Platform



One API. One binary.

CODE grafana/grafana › pkg/services/apiserver ↗

Grafana App Platform



CODE grafana/grafana › pkg/services/apiserver ↗

Custom storage, auth or secrets? Standard API.

If this part of your app…	Use…
Resource with desired state	CRD + Controller
CRUD with custom storage, AuthZ or secrets	Standard k8s APIs, custom backend behind them
Isn't a Resource	?

Things that don't fit nicely

resource	group	get	list	watch	create	update	delete
dashboards	dashboard.grafana.app	●	●	●	●	●	●
folders	folder.grafana.app	●	●	●	●	●	●
playlists	playlist.grafana.app	●	●	●	●	●	●

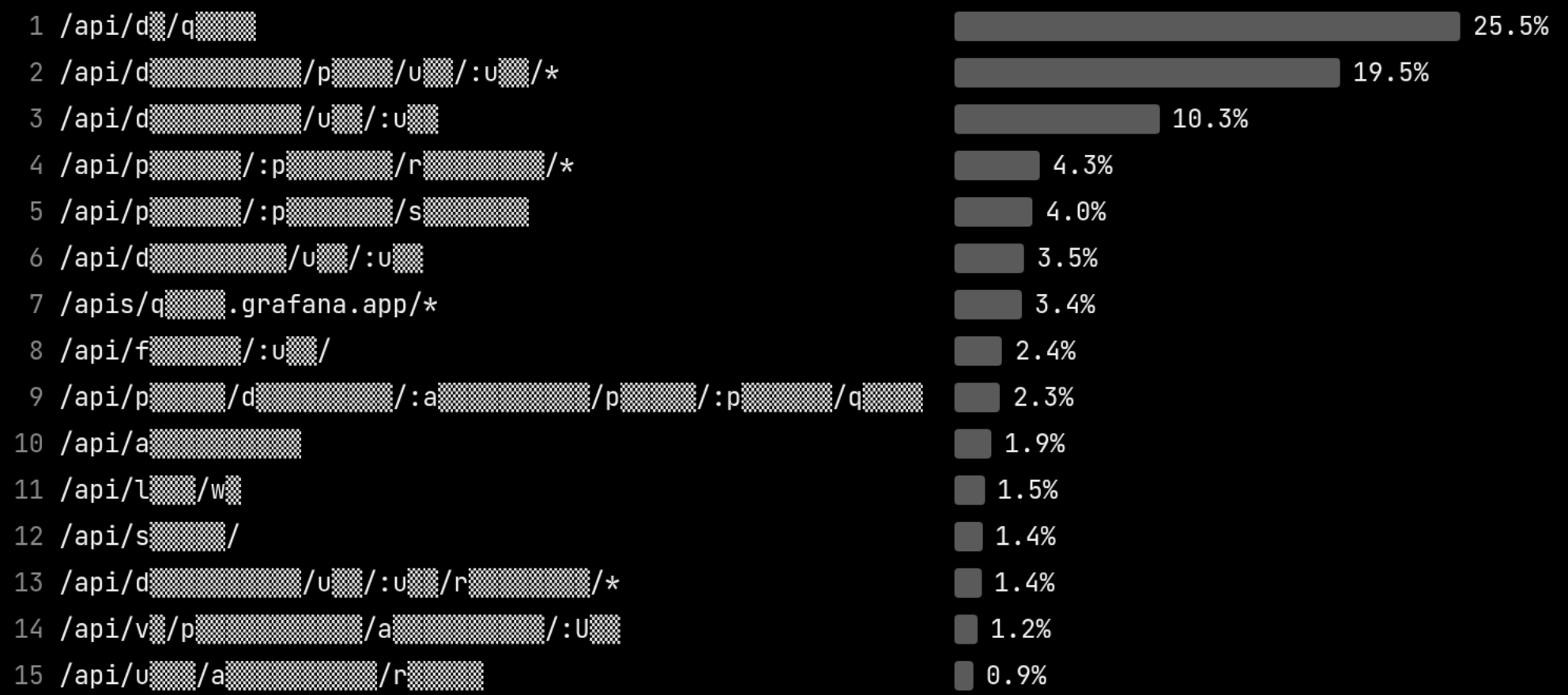


Things that don't fit nicely

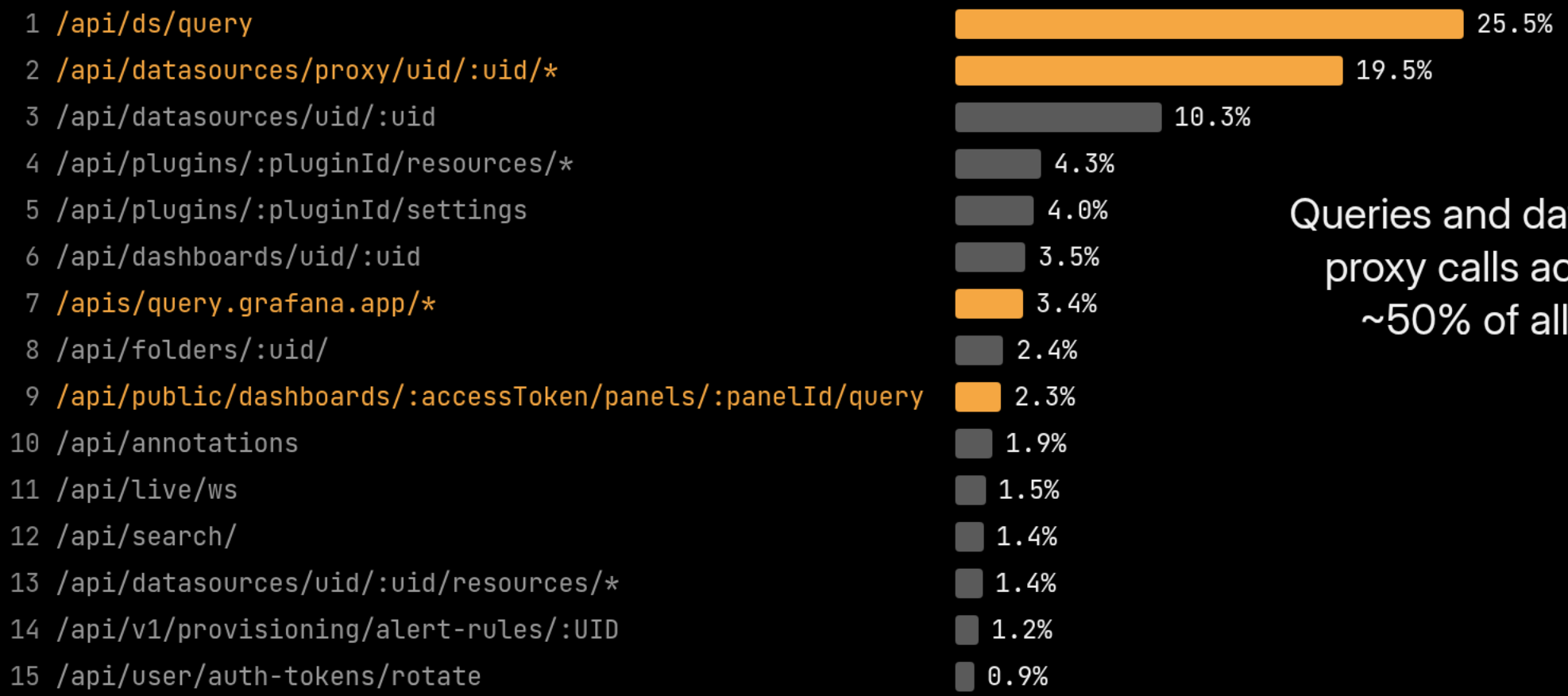
resource	group	get	list	watch	create	update	delete	execute?
dashboards	dashboard.grafana.app	●	●	●	●	●	●	
folders	folder.grafana.app	●	●	●	●	●	●	
playlists	playlist.grafana.app	●	●	●	●	●	●	
[redacted]	[redacted].grafana.app							●



Which API does Grafana serve the most?



Which API does Grafana serve the most?



Queries and data source proxy calls account for ~50% of all requests

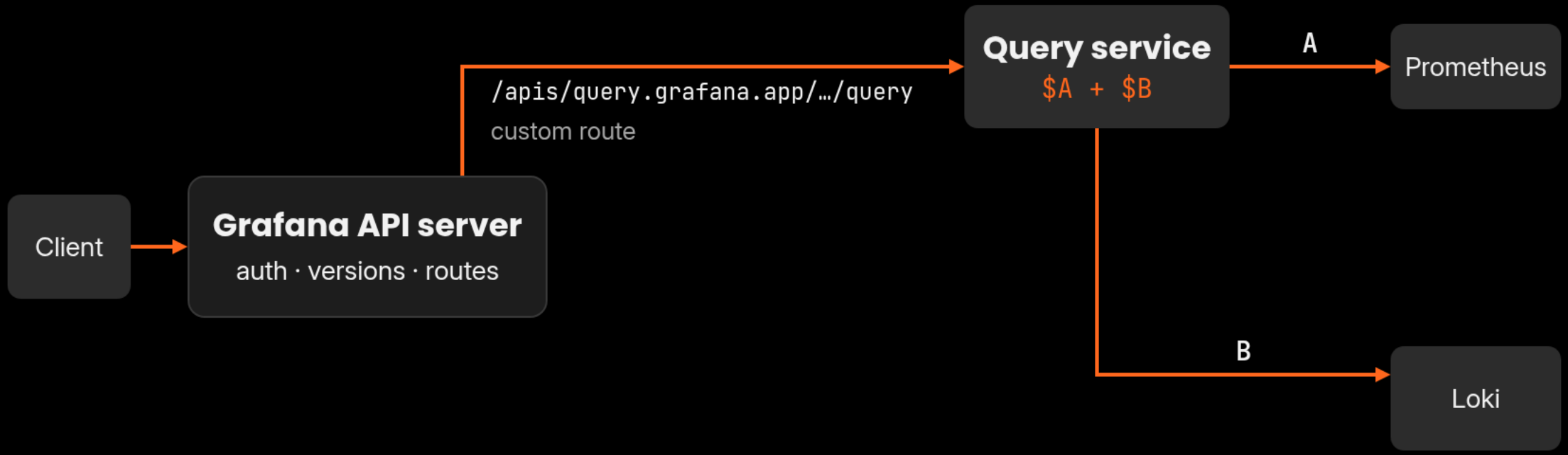
Query isn’t a resource, so it gets custom routes

Client

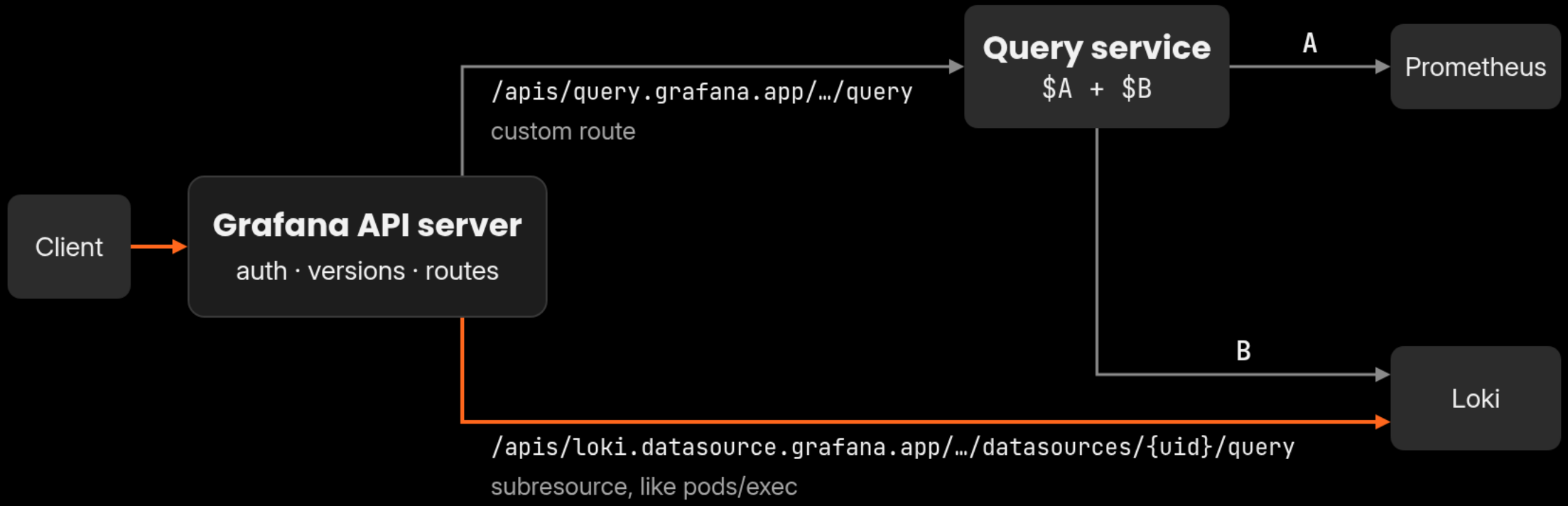
Grafana API server

auth · versions · routes

Query isn’t a resource, so it gets custom routes



Query isn't a resource, so it gets custom routes



Not a resource? Use a custom route.

If this part of your app…	Use…
Resource with desired state	CRD + Controller
CRUD with custom storage, AuthZ or secrets	Standard k8s APIs, custom backend behind them
Isn’t a Resource	Custom routes

Building a feature shouldn’t require becoming a Kubernetes expert

Building a feature shouldn't require becoming a Kubernetes expert

"...all feeling very complicated to just expose a single endpoint making a single network request."

Engineer adding one endpoint



Building a feature shouldn't require becoming a Kubernetes expert

"...I can't seem to get a good response from any aggregated APIs."

Engineer debugging an outage



Building a feature shouldn't require becoming a Kubernetes expert

"Sorry, I am so confused."

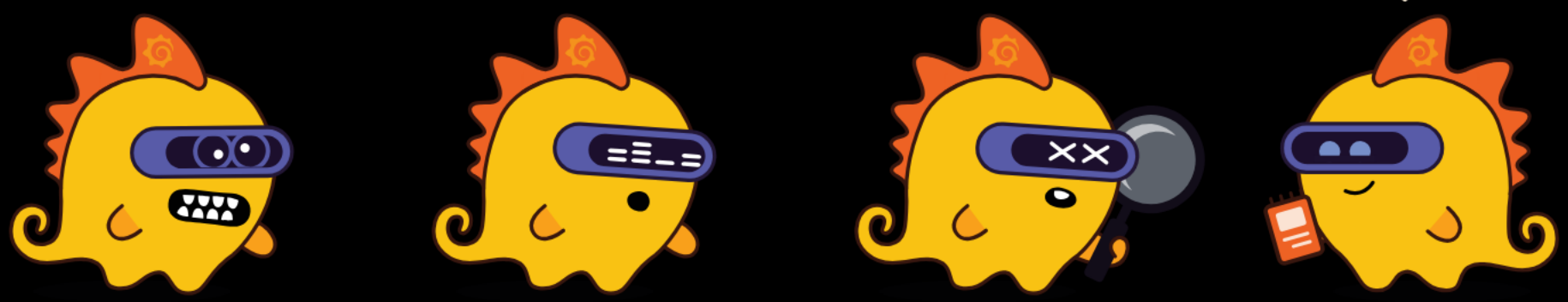
Engineer, mid-incident



Building a feature shouldn't require becoming a Kubernetes expert

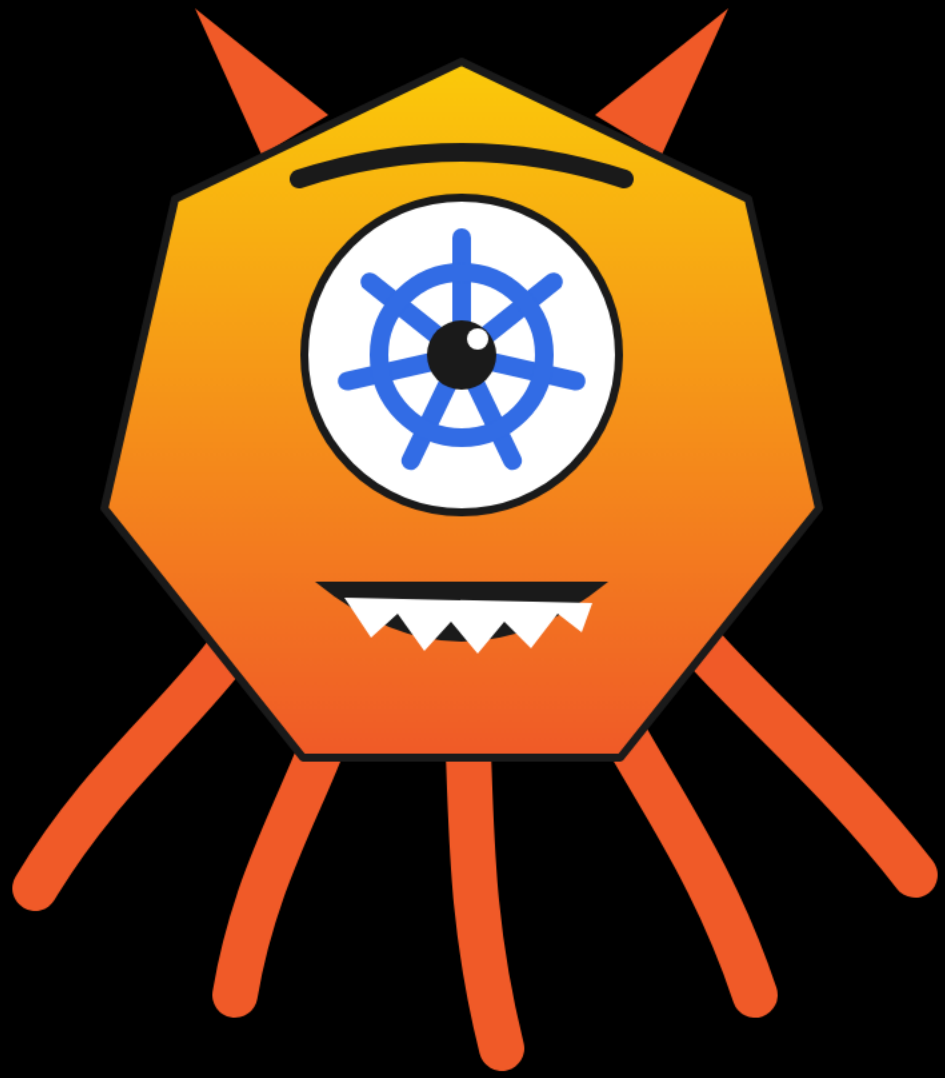
"Aggregation... felt pretty overused and directly tied to an internal implementation detail."

App platform engineer



Did we build a monster?





Aggregation looked right

one Grafana pod per stack



Cloud App Platform

Kubernetes API server

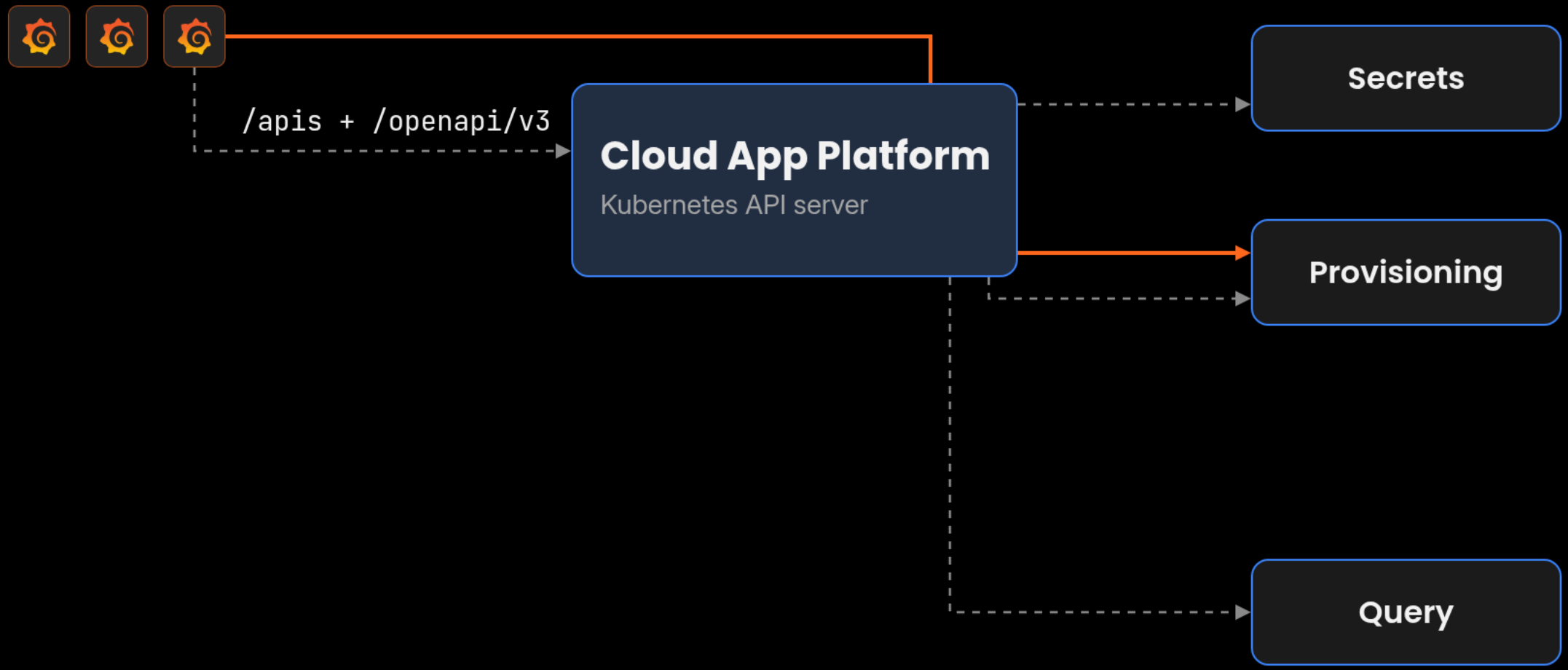
Secrets

Provisioning

Query

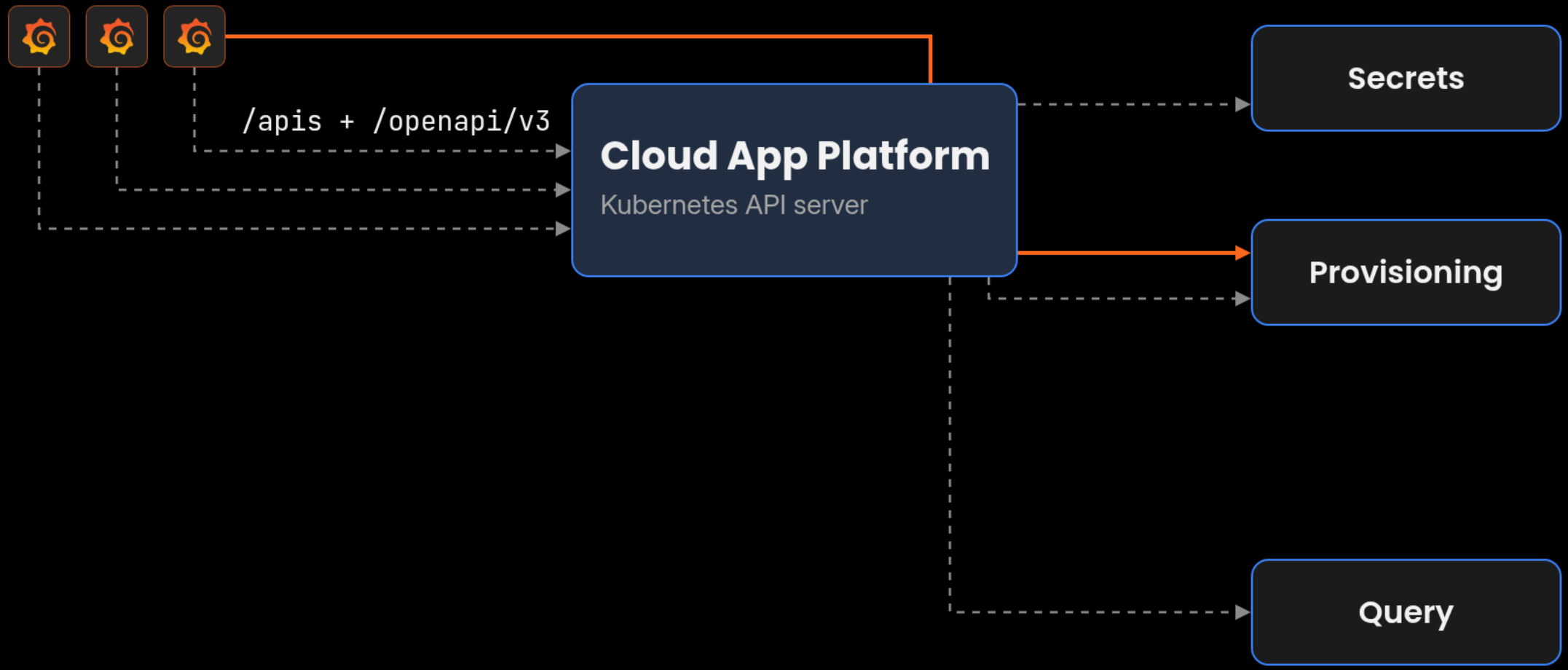
Aggregation looked right

one Grafana pod per stack



Aggregation looked right

one Grafana pod per stack



Every Grafana polled every service

one Grafana pod per stack



Every Grafana polled every service

one Grafana pod per stack

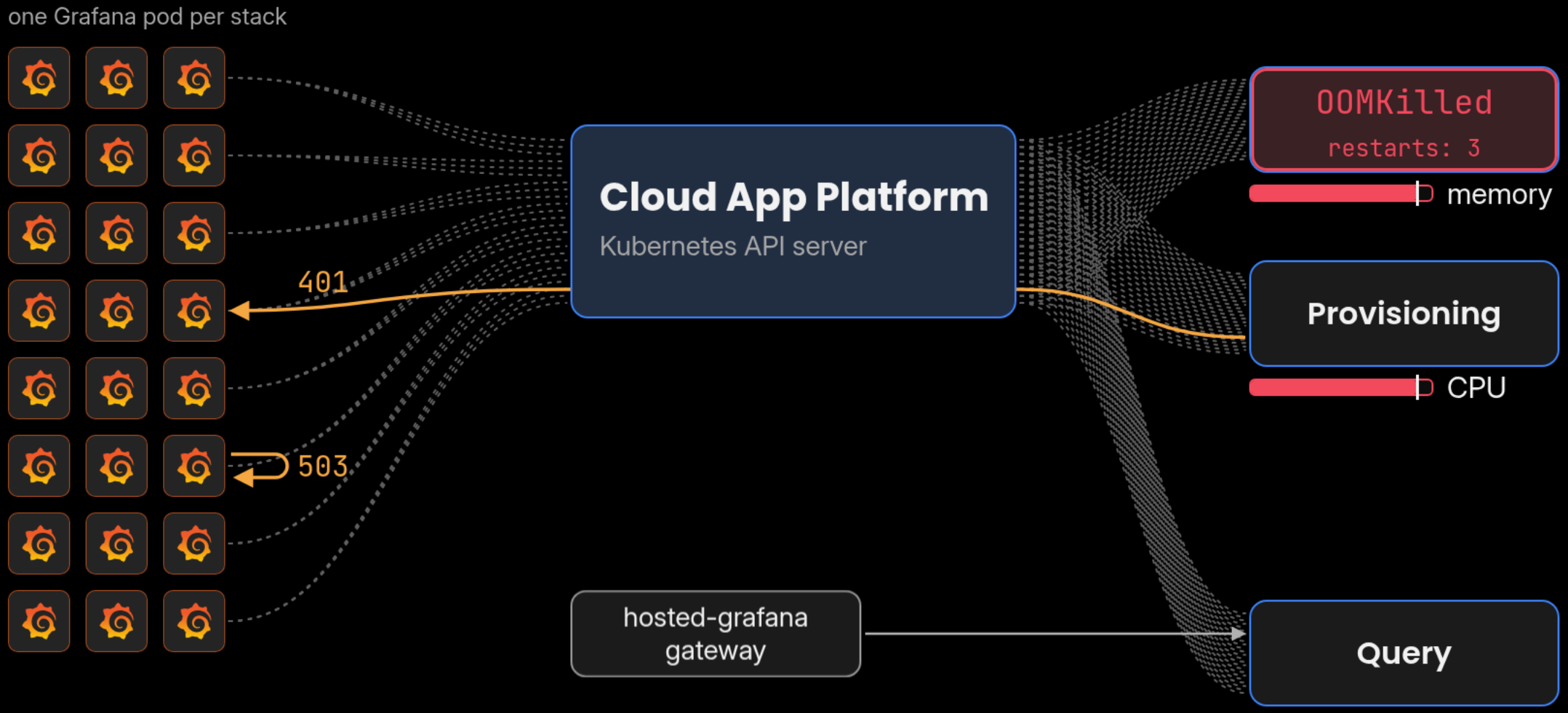


Every Grafana polled every service

one Grafana pod per stack



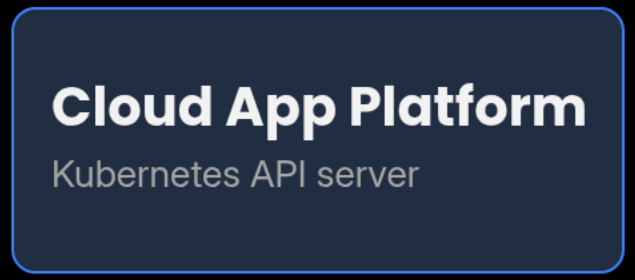
Every Grafana polled every service

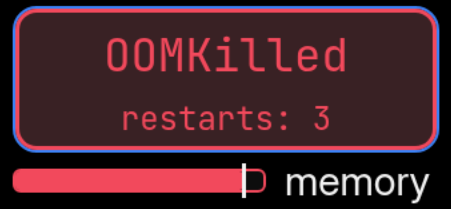


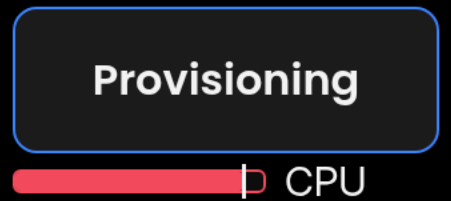
Rolling out: one router for the whole cluster

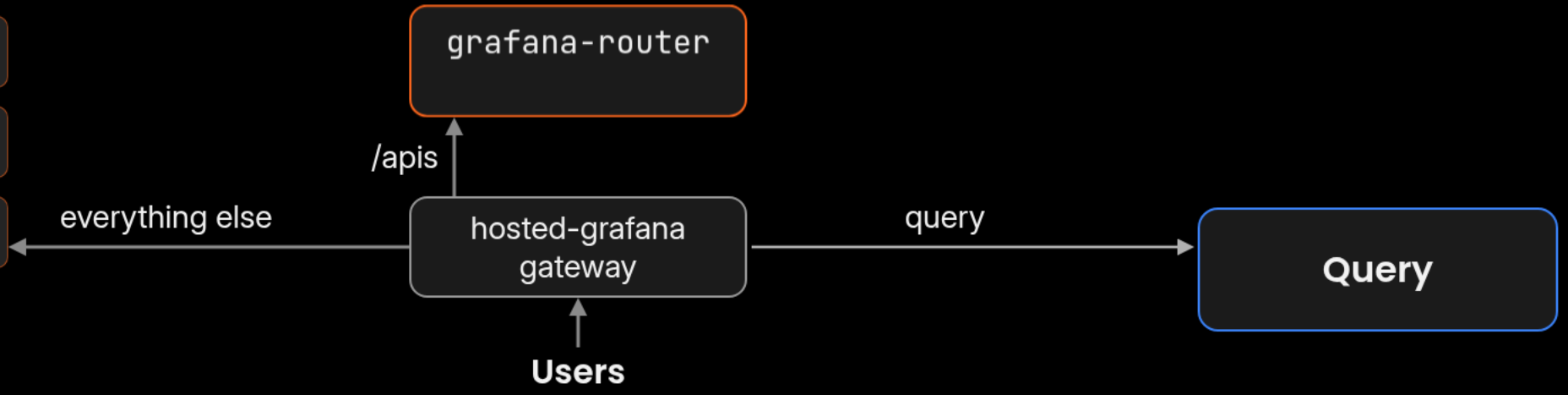
one Grafana pod per stack



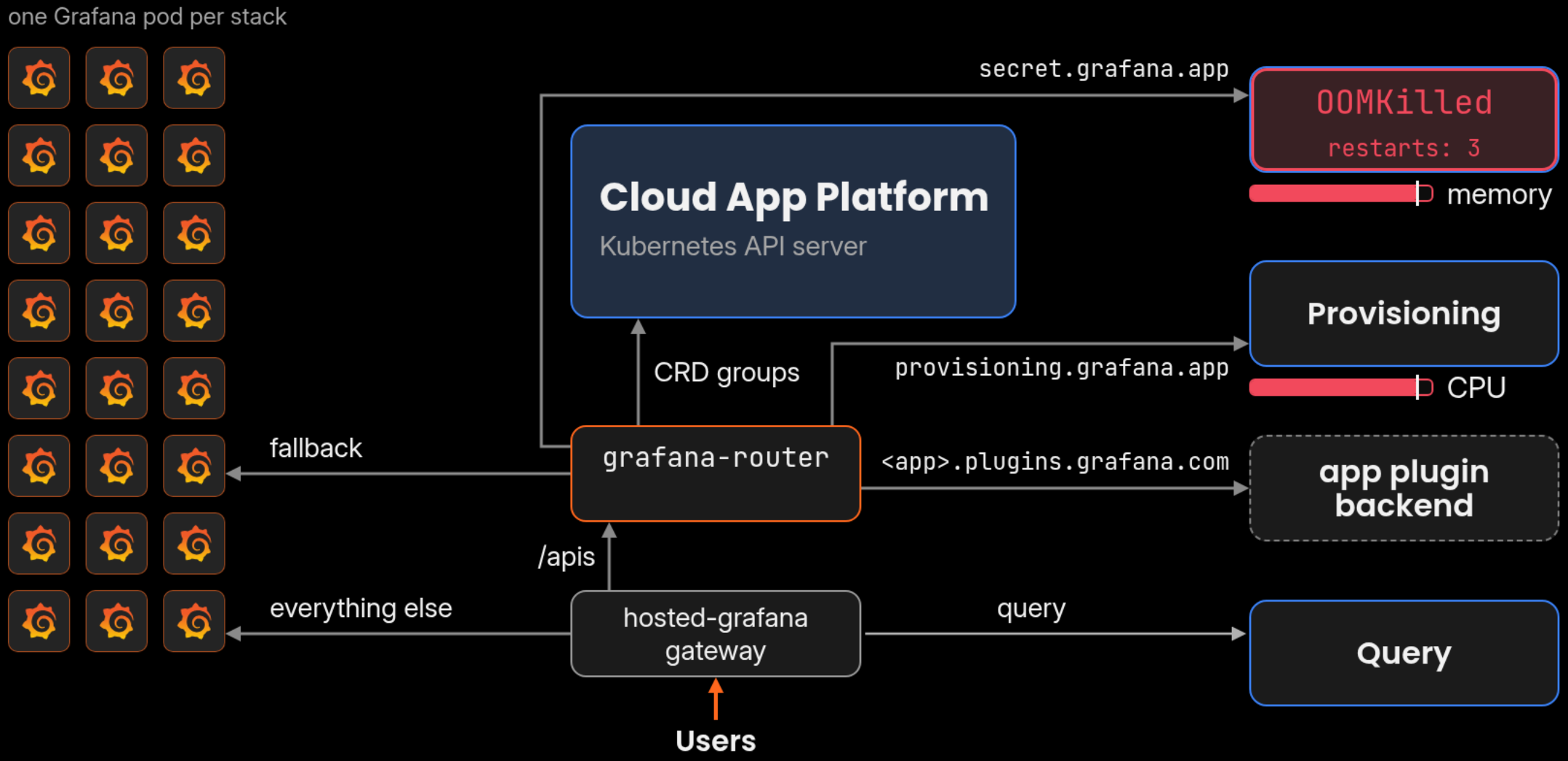




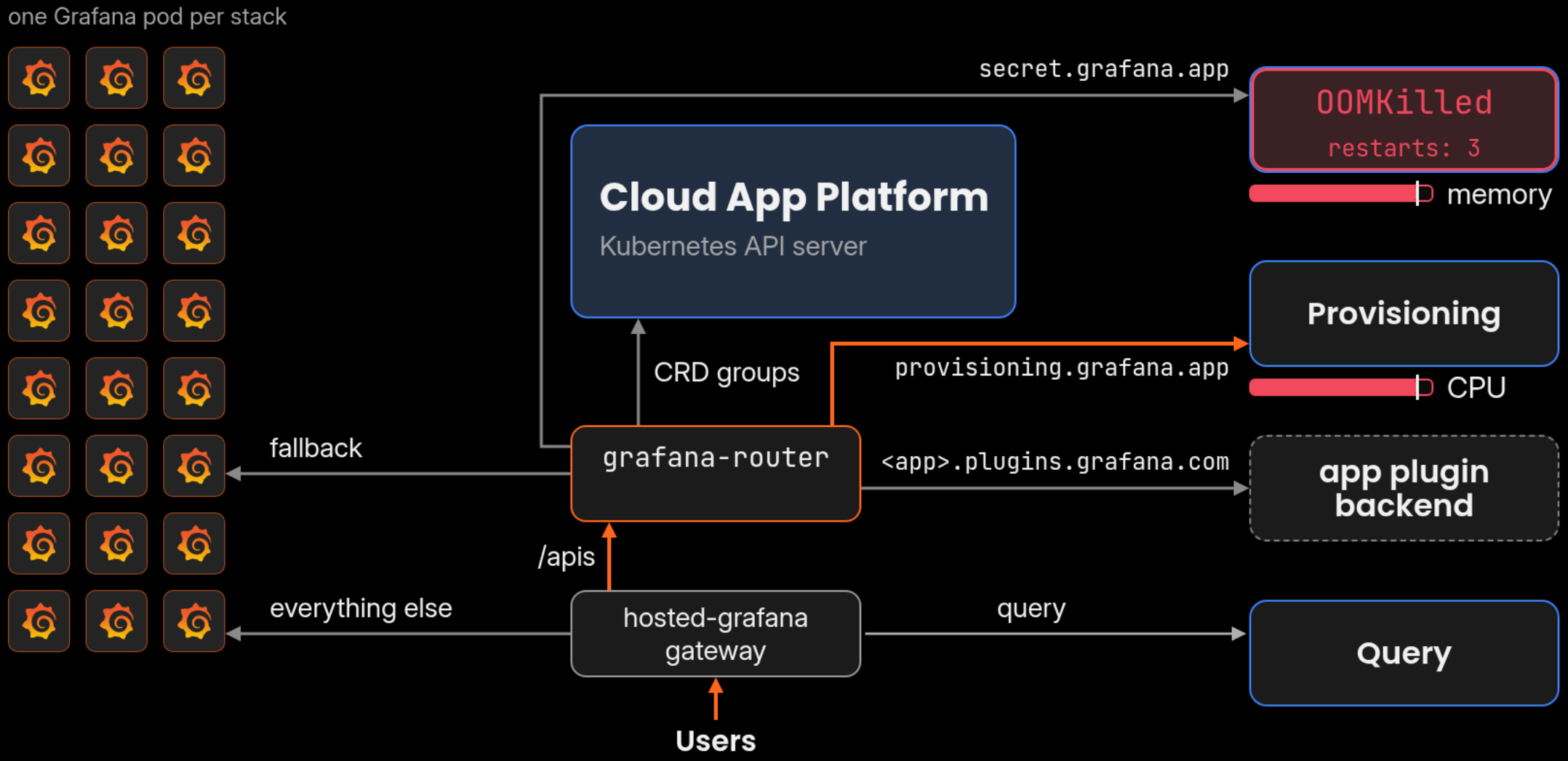




Rolling out: one router for the whole cluster



Rolling out: one router for the whole cluster



Everything we built



Dual writer	Storage migration	Dashboards v1	Folders	
Cloud App Platform	Kine fork	Playlists	Unified storage	Search



Part 3

The payoff

What one schema gets us

New features can start up quicker

"I kinda really like app platform now that I don't have to PR grafana to use it."

Hackathon feedback, from a skeptic

widget.cue

```
Widget: {
  kind: "Widget"
  schema: { spec: { title: string } }
}
```

/apis/widget.grafana.app/v1/…

```
GET     /widgets
GET     /widgets/{name}
POST    /widgets
PUT     /widgets/{name}
PATCH   /widgets/{name}
DELETE  /widgets/{name}
```



CODE grafana/grafana-app-sdk ↗

One model, tooling for free

Dashboard

```
kind: Dashboard
metadata:
  name: service-health
spec: { … }
```

terminal

```
$ gcx resources get dashboards/service-health
$ gcx resources push -p ./dashboards
```

Pull request · Git Sync

```
dashboards/service-health.json
-   "title": "Service health",
+   "title": "Service health (prod)",
```

main.tf

```
resource "grafana_dashboard" "service_health" {
  config_json = file("service-health.json")
}
```

CODE grafana/gcx ↗ grafana/grafana › apps/provisioning ↗

One model, tooling for free

Dashboard

```
kind: Dashboard
metadata:
  name: service-health
spec: { … }
```

terminal

```
$ gcx resources get dashboards/service-health
$ gcx resources push -p ./dashboards
```

Pull request · Git Sync

```
dashboards/service-health.json
-   "title": "Service health",
+   "title": "Service health (prod)",
```

main.tf

```
resource "grafana_dashboard" "service_health" {
  config_json = file("service-health.json")
}
```

CODE grafana/gcx ↗ grafana/grafana › apps/provisioning ↗

One model, tooling for free

Dashboard

```
kind: Dashboard
metadata:
  name: service-health
spec: { … }
```

terminal

```
$ gcx resources get dashboards/service-health
$ gcx resources push -p ./dashboards
```

Pull request · Git Sync

```
dashboards/service-health.json
-   "title": "Service health",
+   "title": "Service health (prod)",
```

main.tf

```
resource "grafana_dashboard" "service_health" {
  config_json = file("service-health.json")
}
```

CODE grafana/gcx ↗ grafana/grafana › apps/provisioning ↗

One model, tooling for free

Dashboard

```
kind: Dashboard
metadata:
  name: service-health
spec: { … }
```

terminal

```
$ gcx resources get dashboards/service-health
$ gcx resources push -p ./dashboards
```

Pull request · Git Sync

```
dashboards/service-health.json
-   "title": "Service health",
+   "title": "Service health (prod)",
```

main.tf

```
resource "grafana_dashboard" "service_health" {
  config_json = file("service-health.json")
}
```

CODE grafana/gcx ↗ grafana/grafana › apps/provisioning ↗

AI speeds up building today

AI speeds up building today

Without the platform

With the platform

AI-built · 1

Next feature

APIs

Storage

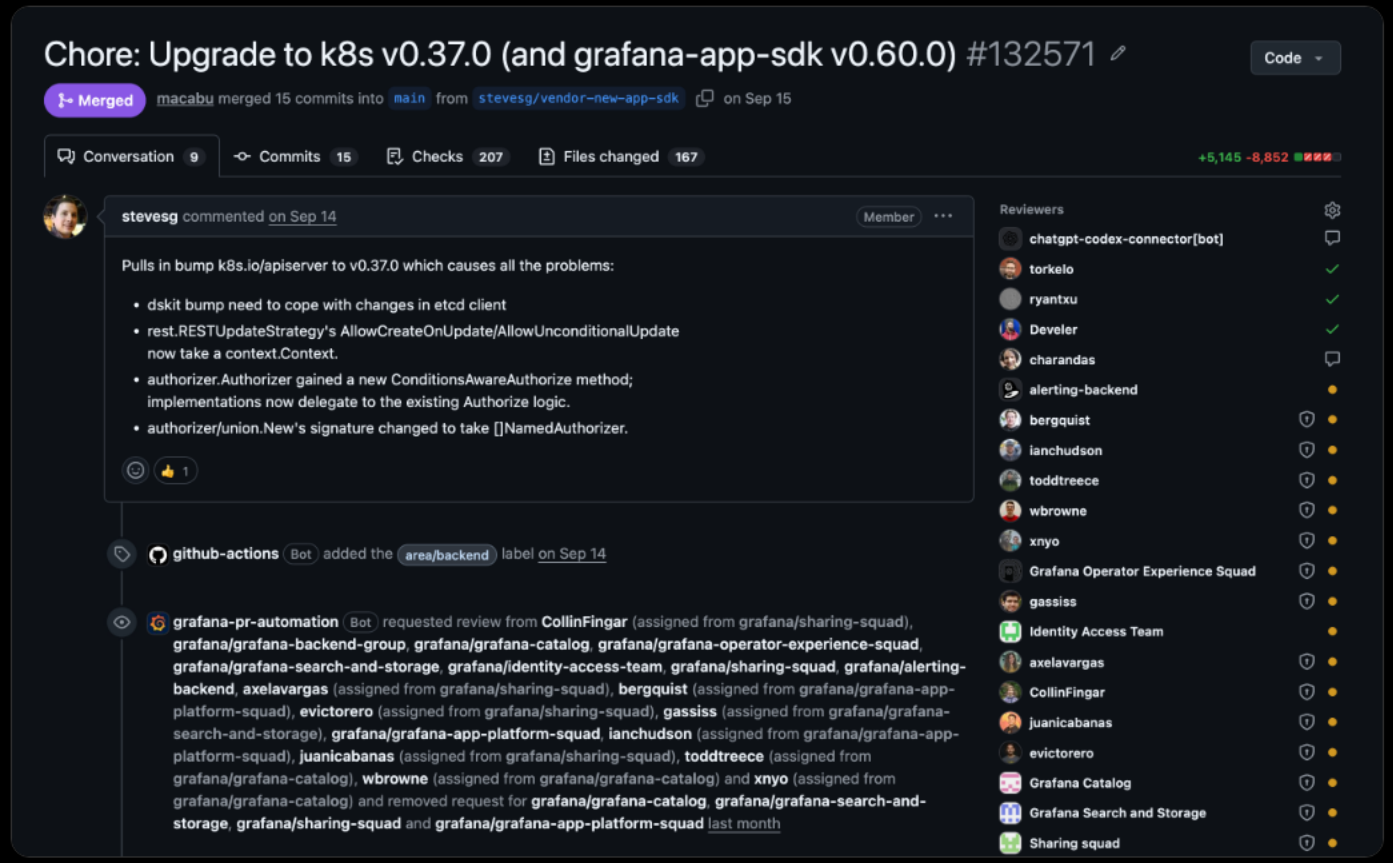
Authz

Grafana App Platform

Each feature invents its own stack.

Three features. One shared foundation.

We depend on Kubernetes libraries, so we keep up with their releases



167

files changed

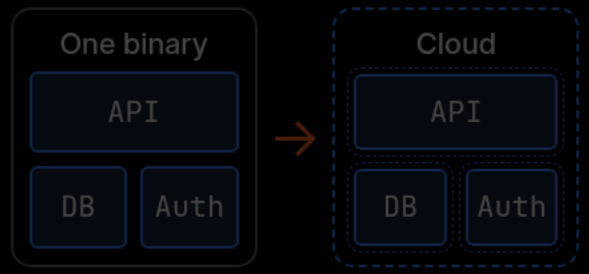
4

breaking changes upstream

8

teams asked to review

Not just for infrastructure



Versioned APIs made Grafana easier to change

State	Resource + controller
Backend	Standard API
Query	Custom route

Only add what you need

what-developers-see.yaml

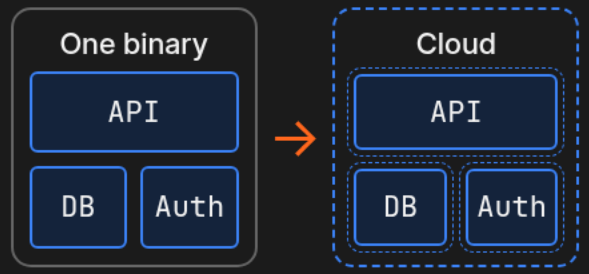
```
apiVersion: app/v1
kind: YourFeature
metadata:
  name: next-idea
spec: { … }
```

Hide the internals



Kubernetes

Not just for infrastructure



Versioned APIs made Grafana easier to change

State	Resource + controller
Backend	Standard API
Query	Custom route

Only add what you need

what-developers-see.yaml

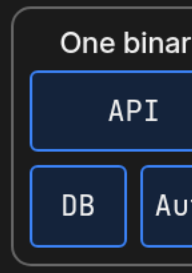
```
apiVersion: app/v1
kind: YourFeature
metadata:
  name: next-idea
spec: { … }
```

Hide the internals



Kubernetes

Not just for infrastructure



Versioned APIs made Grafana easier to change

State	Resource + controller
Backend	Standard API
Query	Custom route

Only add what you need

what-developers-see.yaml

```
apiVersion: app/v1
kind: YourFeature
metadata:
  name: next-idea
spec: { … }
```

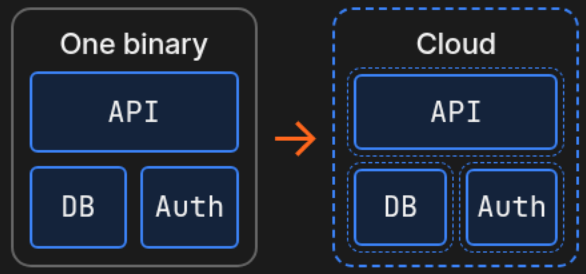
Hide the internals





Kubernetes

Not just for infrastructure



Versioned APIs made Grafana easier to change

State	Resource + controller
Backend	Standard API
Query	Custom route

Only add what you need

what-developers-see.yaml

```
apiVersion: app/v1
kind: YourFeature
metadata:
  name: next-idea
spec: { … }
```

Hide the internals





Kubernetes

Thank you

Questions?

Read the code

`grafana/grafana`	`grafana-app-sdk`	`gcx`
apps/	Build an app	The CLI



We'd love your feedback