Microservices vs Monoliths

Edward Welch
Goutham Veeramachaneni

Grafana Labs

Us

Goutham Veeramachaneni

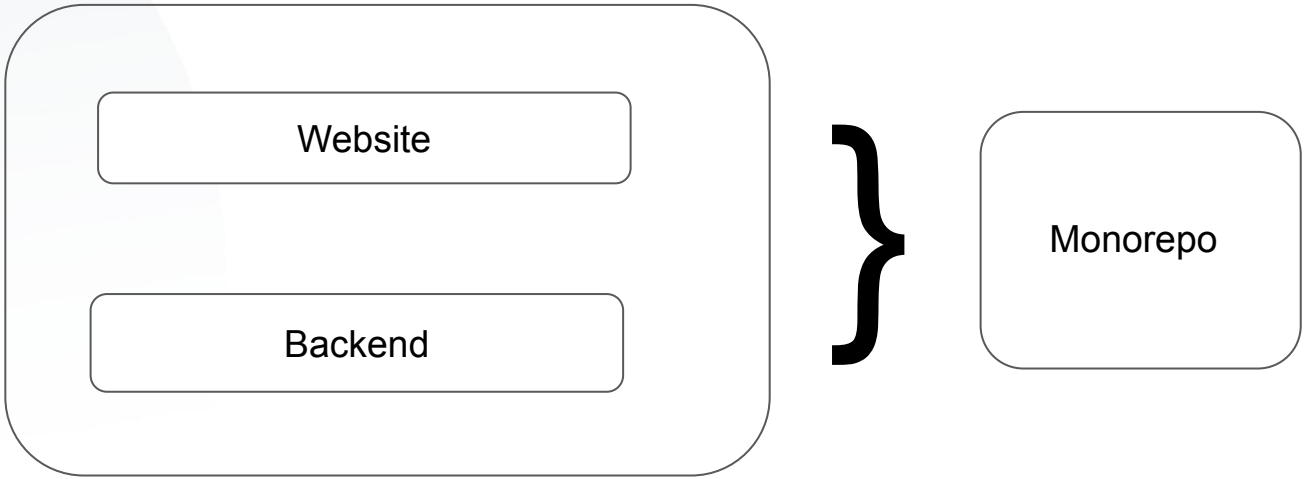
- Prometheus
- Cortex
- Loki
- Biker

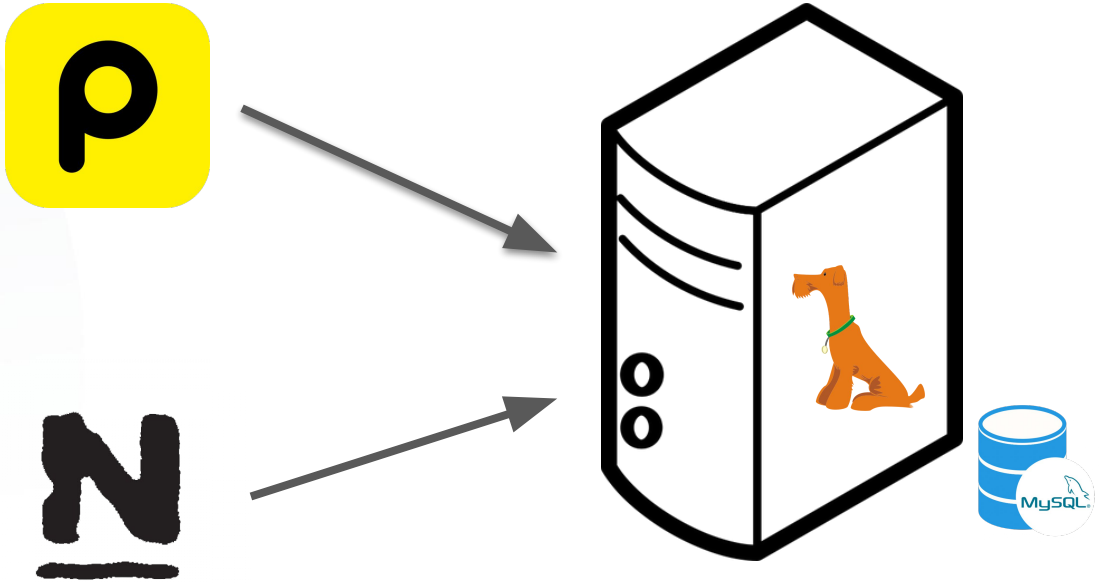
Edward Welch

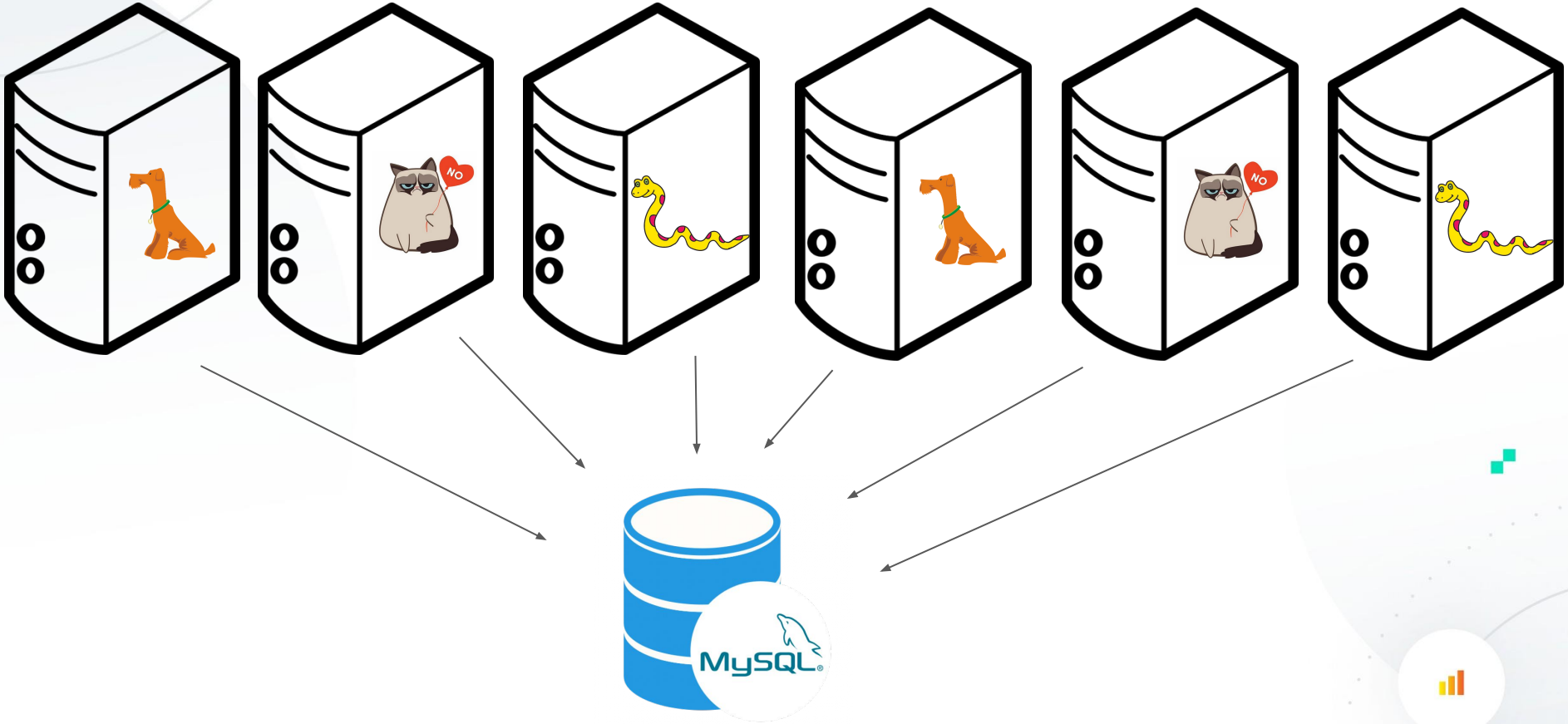
- Loki
- Cortex
- Used to build Java monoliths

Monoliths
Microservices
Monomicroliths

Monoliths


Website
Backend
Monorepo

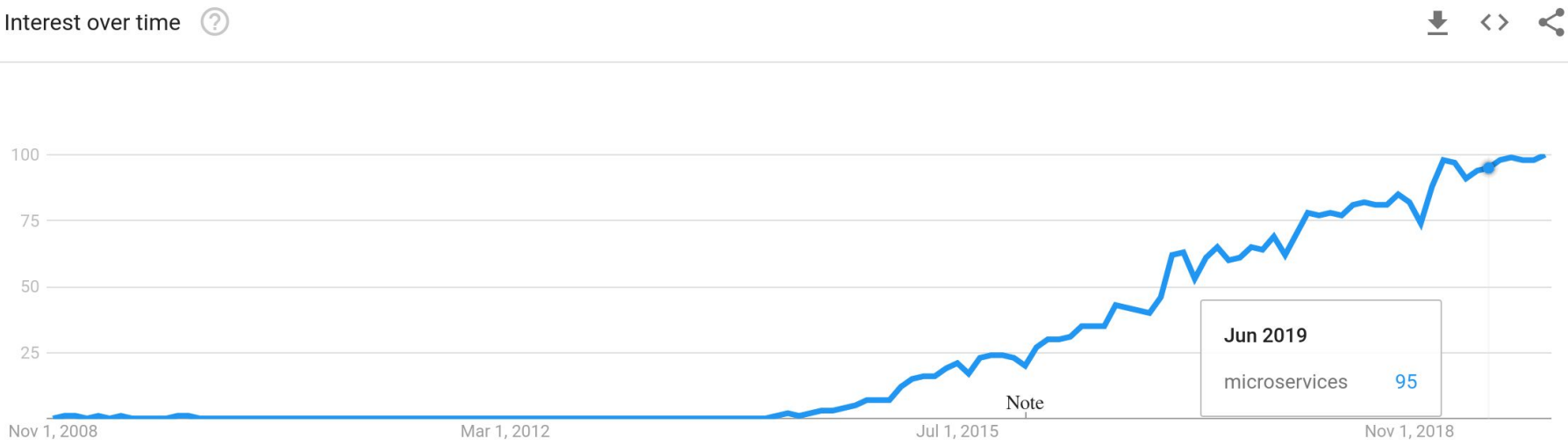

MySQL


NO
NO
MySQL

Any organization that designs a system (defined broadly) will produce a design whose structure is a copy of the organization's communication structure.

- Mel Conway

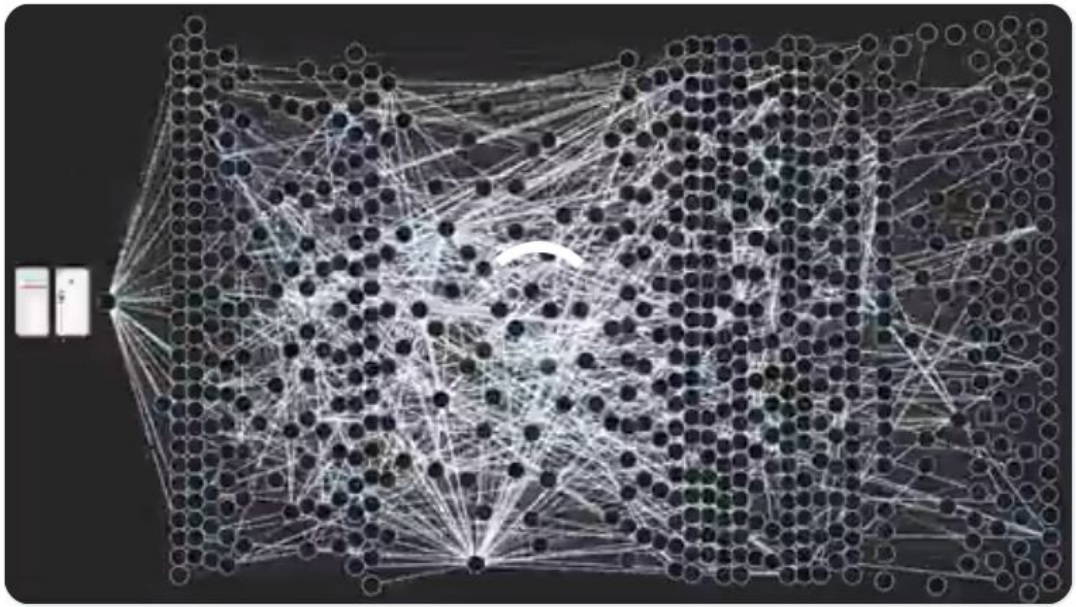
Microservices


Interest over time
100
75
50
25
Nov 1, 2008
Mar 1, 2012
Jul 1, 2015
Nov 1, 2018
Note
Jun 2019
microservices 95

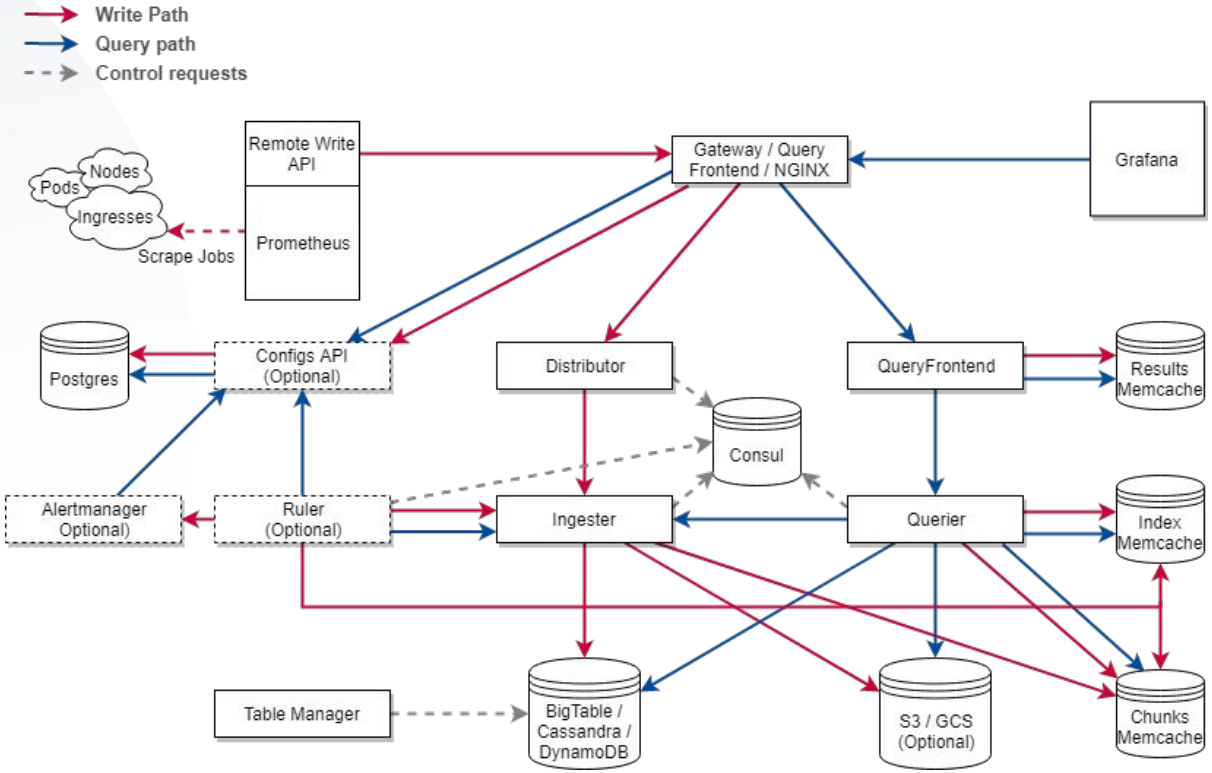
Suhail Patel @suhailpatel · Nov 17

Here's the live request flow on a large subset of the 1500+ microservices on the @MakingMonzo Platform. A lot of things are involved in running a bank!

If you used the @monzo app at around 3pm today, you might be represented 😃



49 230 944


Write Path
Query path
Control requests
Nodes
Pods
Ingresses
Scrape Jobs
Remote Write API
Prometheus
Gateway / Query Frontend / NGINX
Grafana
Postgres
Configs API (Optional)
Distributor
QueryFrontend
Results Memcache
Consul
Alertmanager Optional)
Ruler (Optional)
Ingester
Querier
Index Memcache
Table Manager
BigTable / Cassandra / DynamoDB
S3 / GCS (Optional)
Chunks Memcache

Developing

To build:

```
make
```

(By default the build runs in a Docker container, using an image built with all the tools required. The source code is mounted from where you run `make` into the build container as a Docker volume.)

To run the test suite:

```
make test
```

To checkout Cortex in minikube:

```
kubectl create -f ./k8s
```

(these manifests use `latest` tags, i.e. this will work if you have just built the images and they are available on the node(s) in your Kubernetes cluster)

Cortex will sit behind an nginx instance exposed on port 30080. A job is deployed to scrape it itself. Try it:

http://192.168.99.100:30080/api/prom/api/v1/query?query=up

```
make test
```

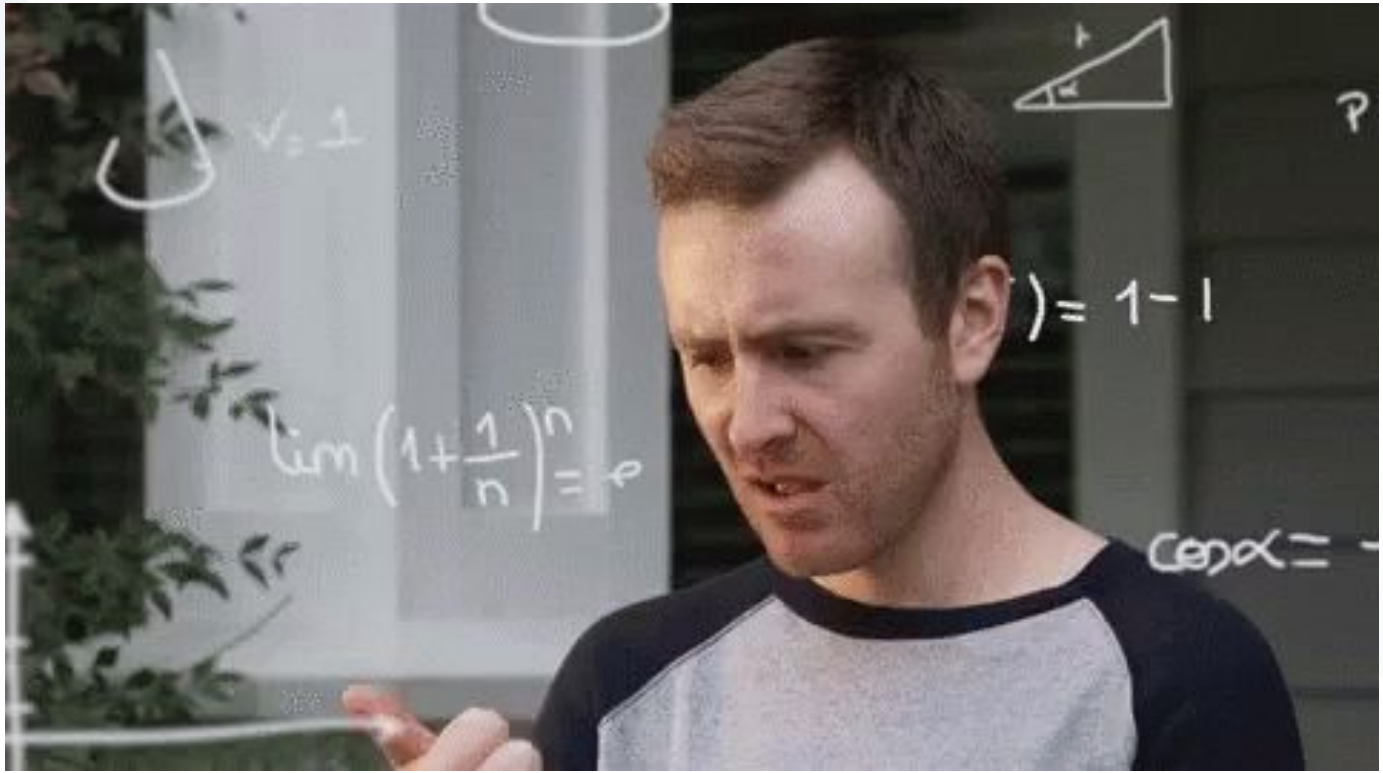
To checkout Cortex in minikube:

```
kubectl create -f ./k8s
```

(these manifests use `latest` tags, i.e. this will work if you have just built the images and they a
your Kubernetes cluster)

Cortex will sit behind an nginx instance exposed on port 30080. A job is deployed to scrape it its

http://192.168.99.100:30080/api/prom/api/v1/query?query=up


$\lim\left(1+\frac{1}{n}\right)^n = e$

Debugging

segment · Blog

All Company Integrations Product Growth Events Engineering

Get a Segment D

ENGINEERING

Goodbye Microservices: From 100s of problem children to 1 superstar

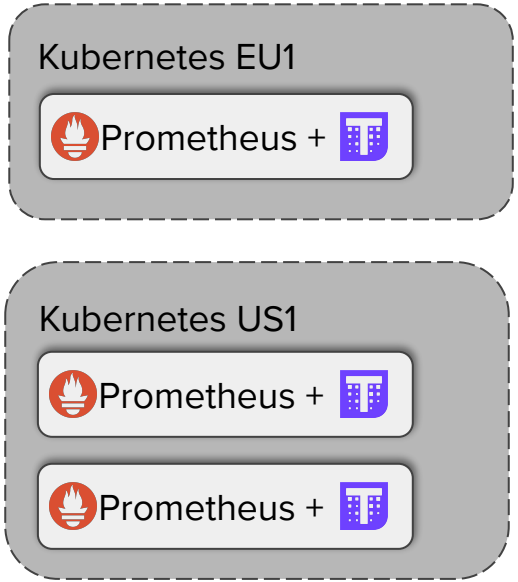
Alexandra Noonan on Jul 10th 2018

Unless you've been living under a rock, you probably already know that microservices is the architecture *du jour*. Coming of age alongside this trend, Segment adopted this as a best practice early-on, which served us well in some cases, and, as you'll soon learn, not so well in others.

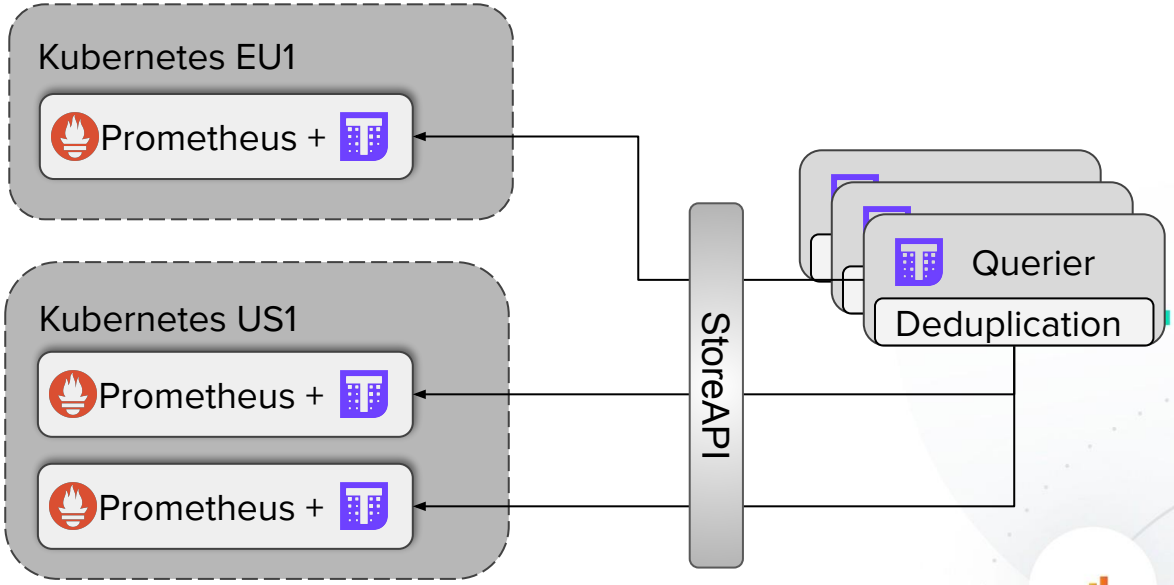
Monomicroliths

- Single Binary Monolith.
- Composed of Microservices.

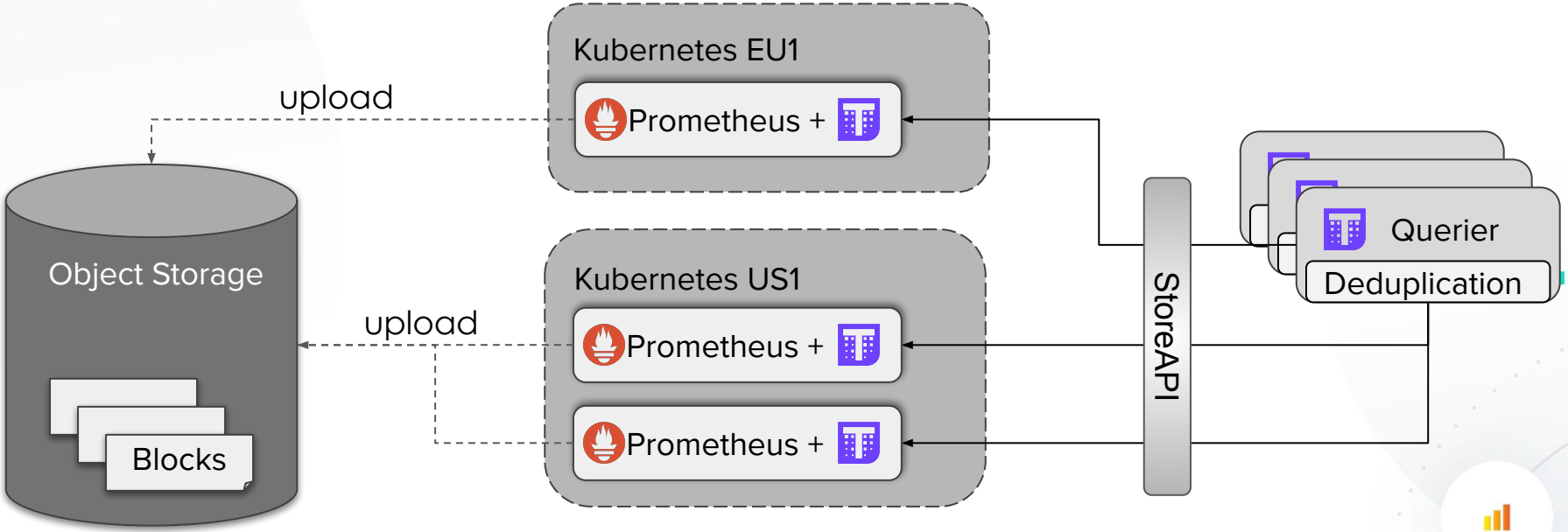
Thanos


Kubernetes EU1
Prometheus +
Kubernetes US1
Prometheus +
Prometheus +

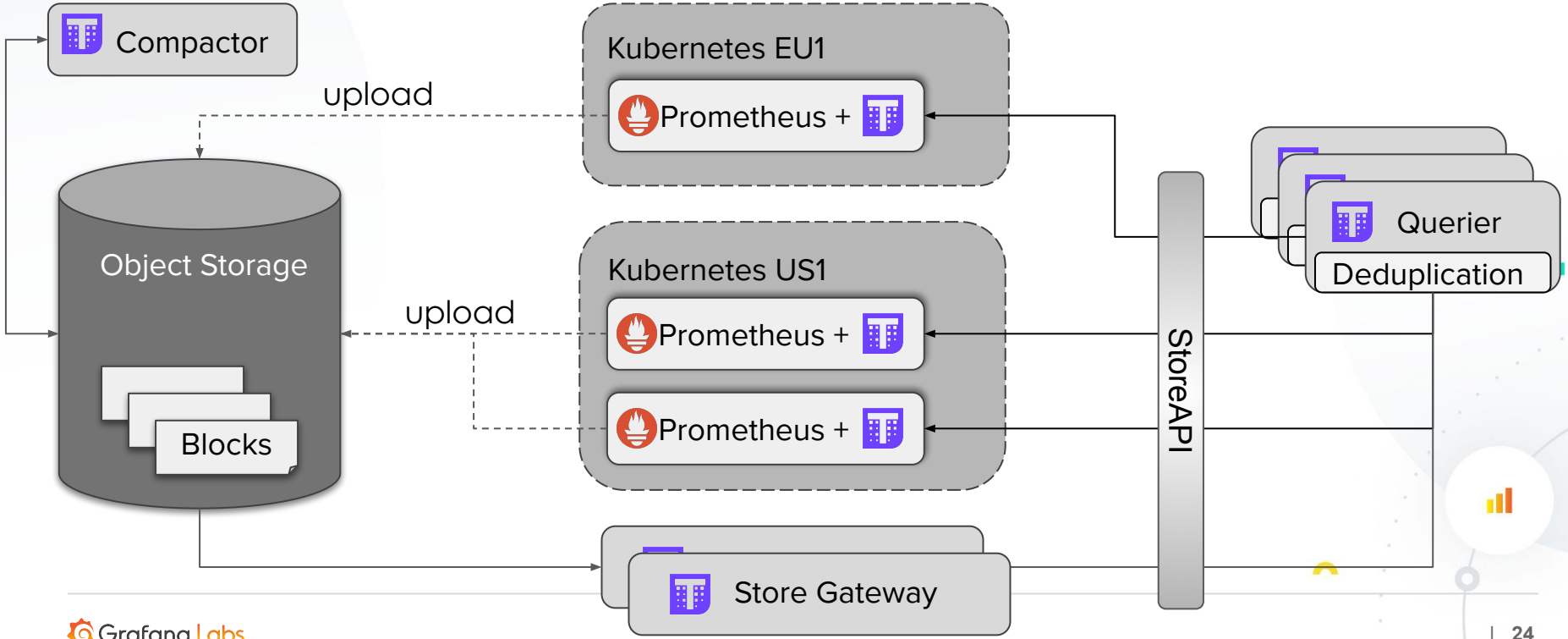
Thanos


Kubernetes EU1
Prometheus +
Kubernetes US1
Prometheus +
Prometheus +
StoreAPI
Querier
Deduplication

Thanos


Kubernetes EU1
Prometheus +
upload
Object Storage
Blocks
upload
Kubernetes US1
Prometheus +
Prometheus +
StoreAPI
Querier
Deduplication

Thanos


Compactor
upload
Kubernetes EU1
Prometheus +
Object Storage
Blocks
upload
Kubernetes US1
Prometheus +
Prometheus +
StoreAPI
Querier
Deduplication
Store Gateway

Single instance, single process.

For simplicity & to get started, we'll run it as a single process with no dependencies:

```
$ go build ./cmd/cortex
$ ./cortex -config.file=./docs/single-process-config.yaml
```

JAEGER

Modular Codebase

Read

Write

Calculate/Process

```go
// Config is the root config for Cortex.
type Config struct {
    Target      moduleName `yaml:"target,omitempty"`
    AuthEnabled bool       `yaml:"auth_enabled,omitempty"`
    PrintConfig bool       `yaml:"-"`
    HTTPPrefix  string     `yaml:"http_prefix"`

    Server         server.Config            `yaml:"server,omitempty"`
    Distributor    distributor.Config       `yaml:"distributor,omitempty"`
    Querier        querier.Config           `yaml:"querier,omitempty"`
    IngesterClient client.Config            `yaml:"ingester_client,omitempty"`
    Ingester       ingester.Config          `yaml:"ingester,omitempty"`
    Storage        storage.Config           `yaml:"storage,omitempty"`
    ChunkStore     chunk.StoreConfig        `yaml:"chunk_store,omitempty"`
    Schema         chunk.SchemaConfig       `yaml:"schema,omitempty"`
    LimitsConfig   validation.Limits        `yaml:"limits,omitempty"`
    Prealloc       client.PreallocConfig    `yaml:"prealloc,omitempty"`
    Worker         frontend.WorkerConfig    `yaml:"frontend_worker,omitempty"`
    Frontend       frontend.Config          `yaml:"frontend,omitempty"`
    TableManager   chunk.TableManagerConfig `yaml:"table_manager,omitempty"`
    Encoding       encoding.Config          `yaml:"-"` // No yaml for this, it only works with flags.

    Ruler        ruler.Config                              `yaml:"ruler,omitempty"`
    ConfigStore  config_client.Config                      `yaml:"config_store,omitempty"`
    Alertmanager alertmanager.MultitenantAlertmanagerConfig `yaml:"alertmanager,omitempty"`
}
```

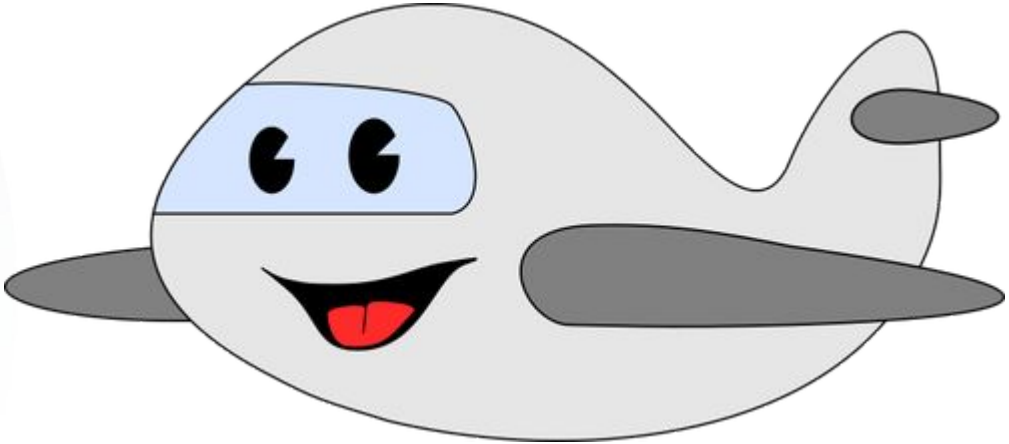
Single instance, single process.

For simplicity & to get started, we'll run it as a single process with no dependencies:

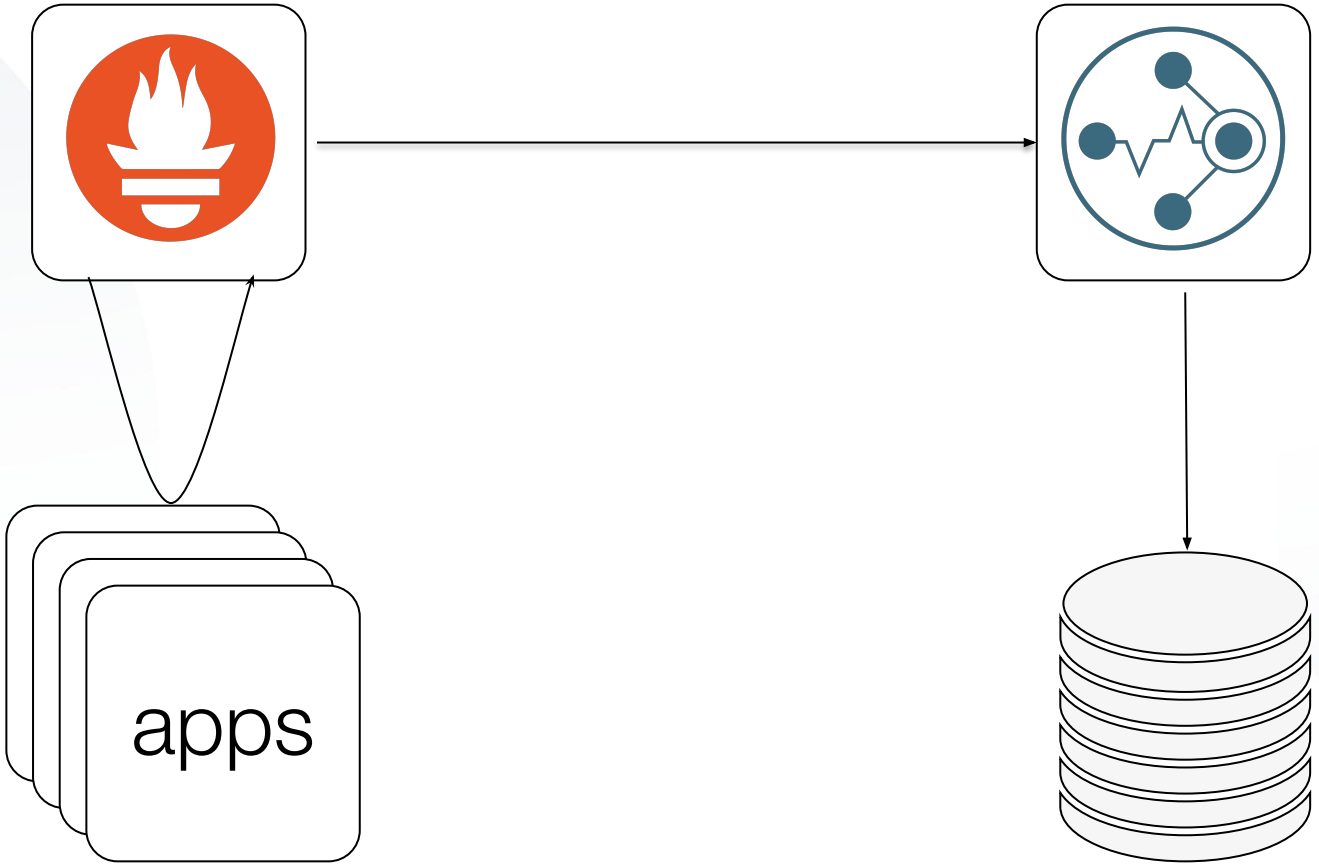
```
$ go build ./cmd/cortex
$ ./cortex -config.file=./docs/single-process-config.yaml
```

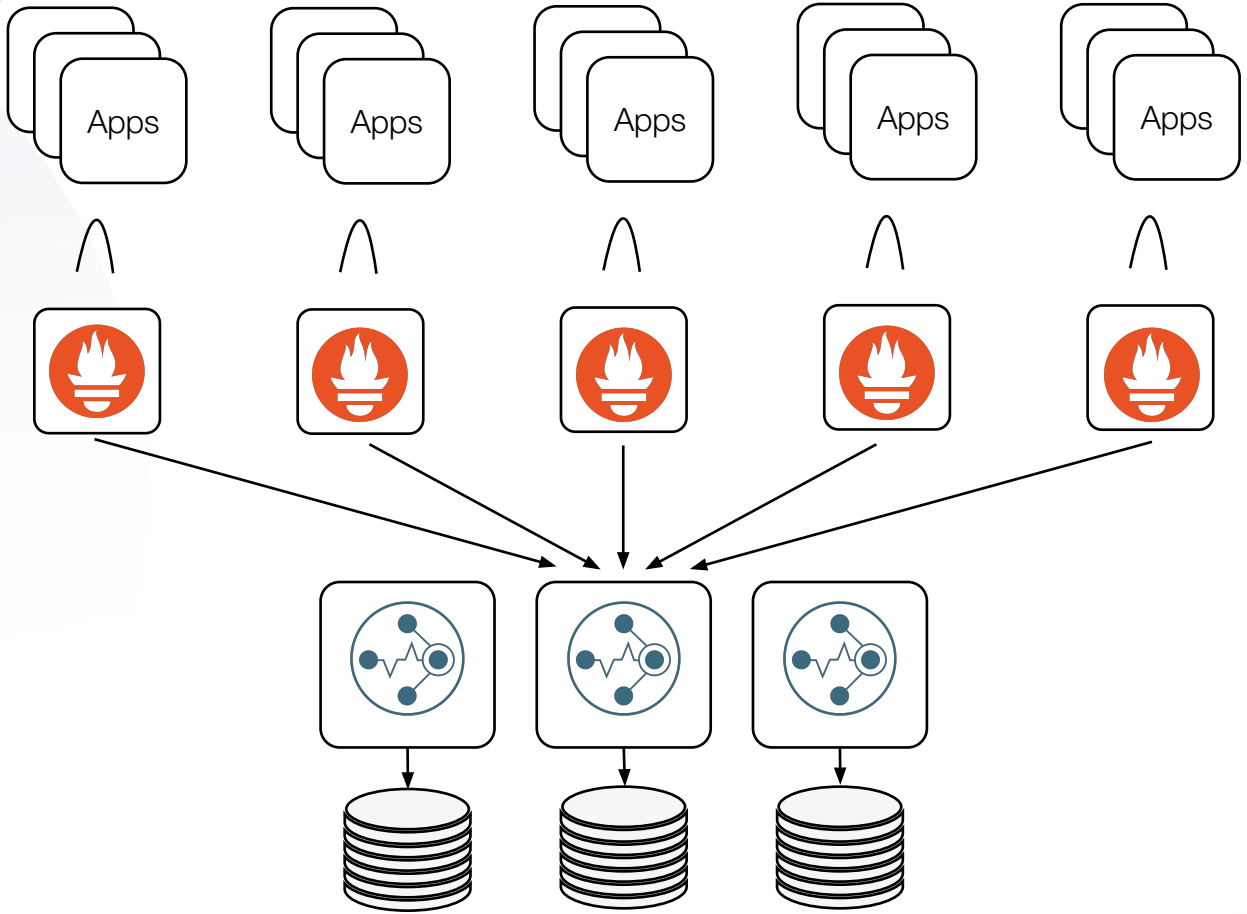
```
work-pc:~ ewelch$ kc describe deployments distributor ingester querier table-manager | egrep 'Image|Args|config.file|target'
    Image:      grafana/loki/v1.0.0
    Args:
      -config.file=/etc/loki/config.yaml
      -target=distributor
    Image:      grafana/loki/v1.0.0
    Args:
      -config.file=/etc/loki/config.yaml
      -target=ingester
    Image:      grafana/loki/v1.0.0
    Args:
      -config.file=/etc/loki/config.yaml
      -target=querier
    Image:      grafana/loki/v1.0.0
    Args:
      -config.file=/etc/loki/config.yaml
      -target=table-manager
```


gRPC
A high performance, open-source universal
RPC framework

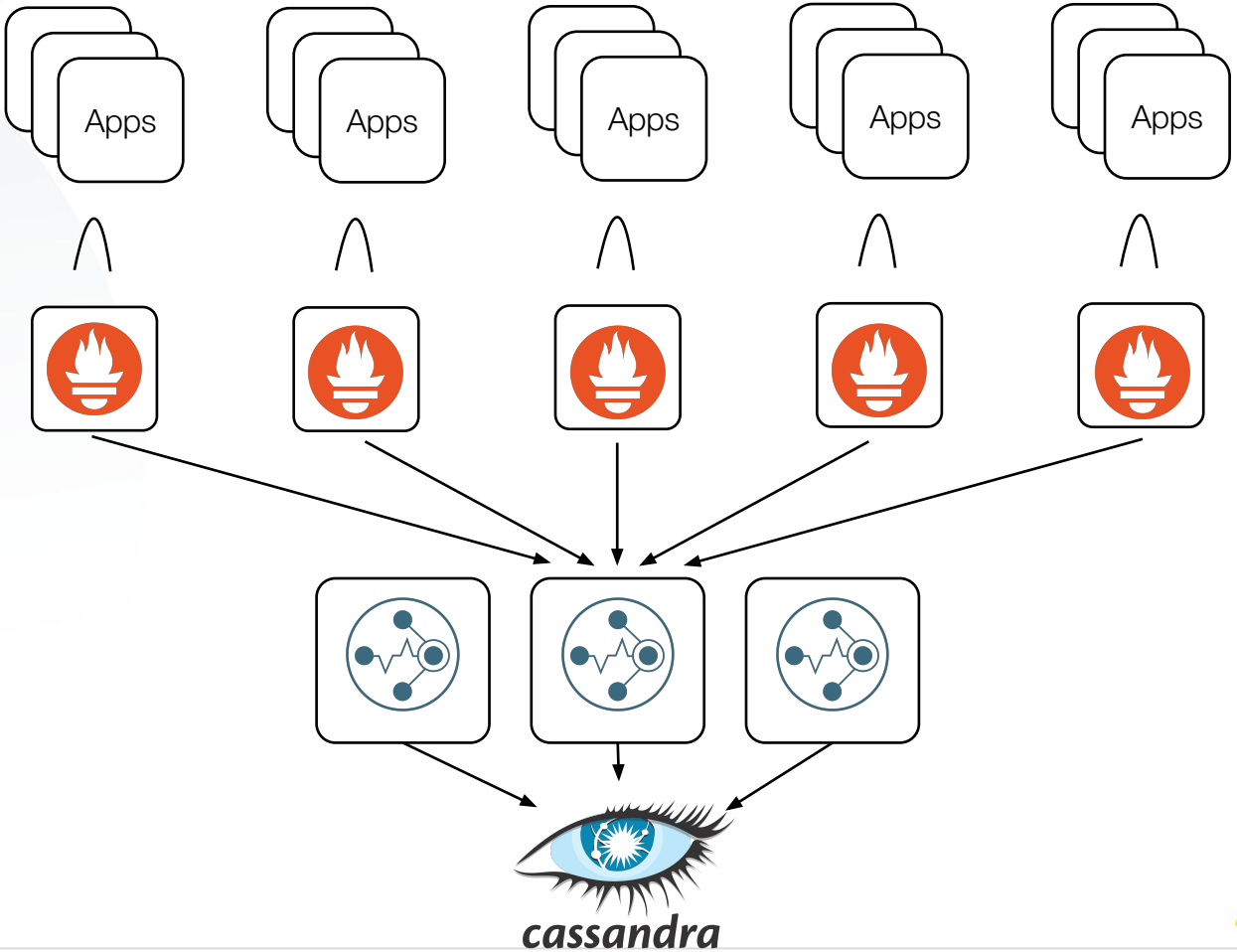


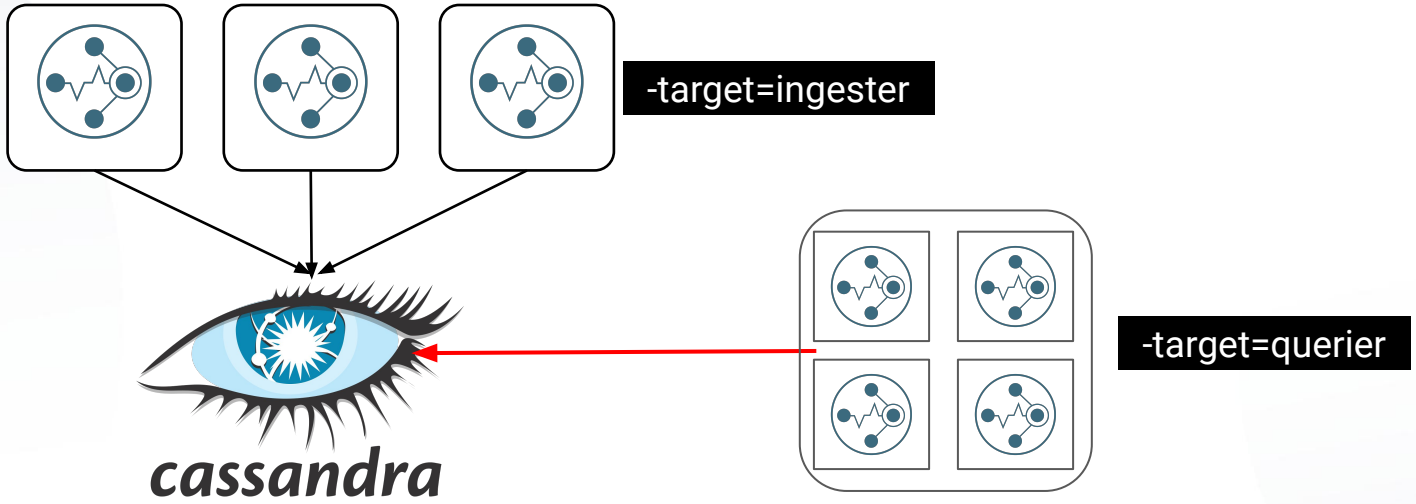

MONOREPO
ALL THE THINGS!

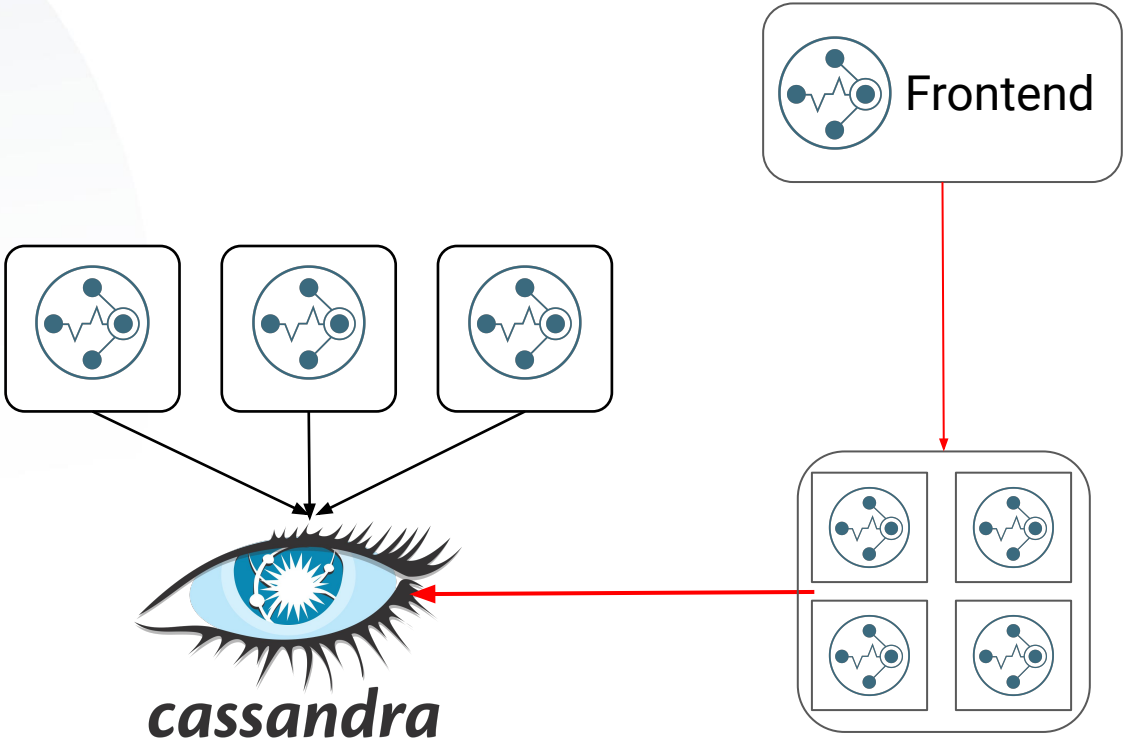

apps

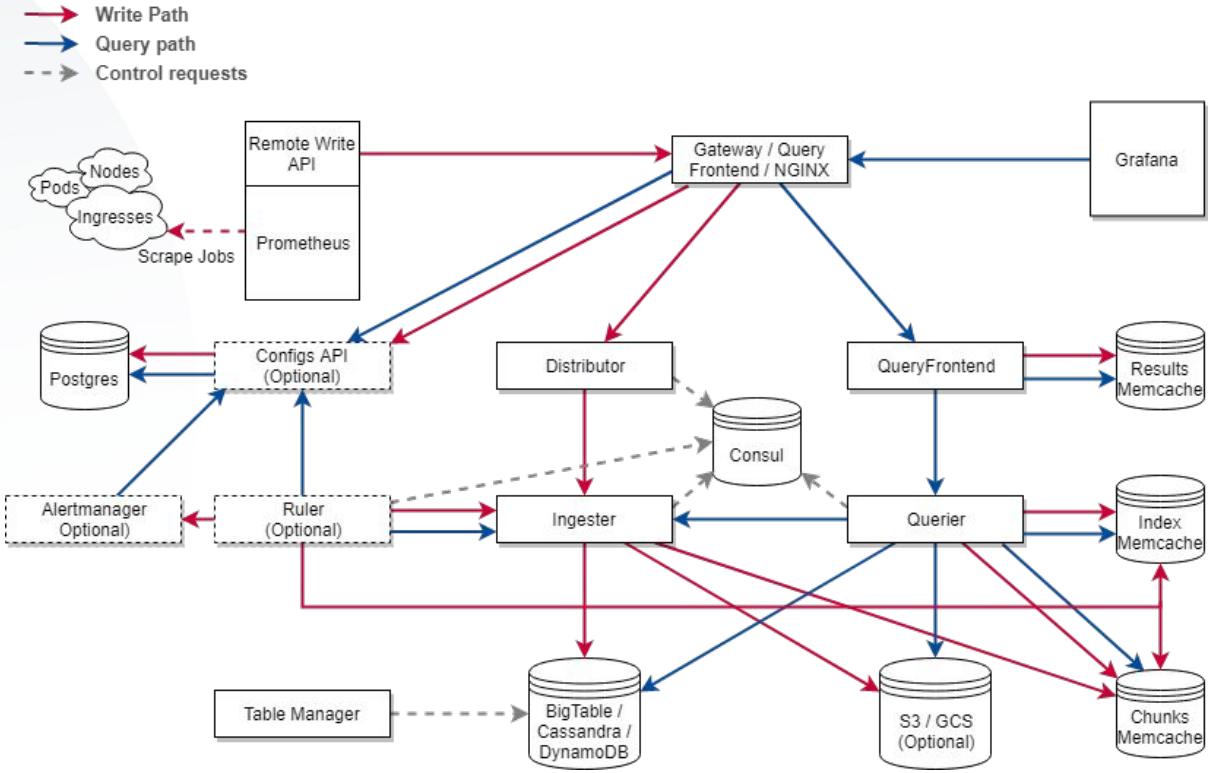


Scale out


Apps
Apps
Apps
Apps
Apps
cassandra


-target=ingester
cassandra
-target=querier


Frontend
cassandra


Write Path
Query path
Control requests
Remote Write API
Gateway / Query Frontend / NGINX
Grafana
Nodes
Pods
Ingresses
Scrape Jobs
Prometheus
Postgres
Configs API (Optional)
Distributor
QueryFrontend
Results Memcache
Consul
Alertmanager Optional)
Ruler (Optional)
Ingester
Querier
Index Memcache
Table Manager
BigTable / Cassandra / DynamoDB
S3 / GCS (Optional)
Chunks Memcache

Monomicroliths

You can have your monolith and scale it too.

Questions?

Come visit Grafana Labs at booth SE22.