CLOUDFLARE

Monitoring the world: Scaling Thanos in Dynamic Prometheus Environments

Colin Douch, Observability Tech Lead @ Cloudflare

CLOUDFLARE

- Globally Distributed
- Highly Available
- ~50TB of metrics daily
- > 1000 sidecars
- 600 transient locations

CLOUDFLARE

- Globally Distributed
- Highly Available
- ~50TB of metrics daily
- > 1000 sidecars
- 600 transient locations

A Quick Introduction

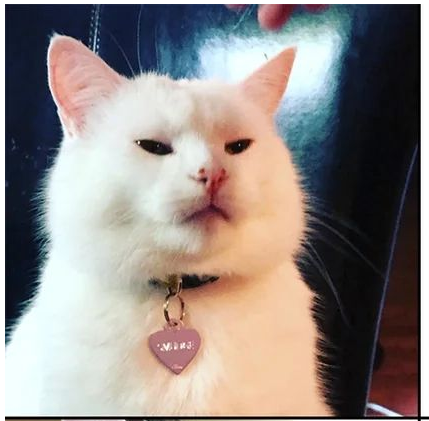
Hi! I'm Colin

Observability Lead @ Cloudflare

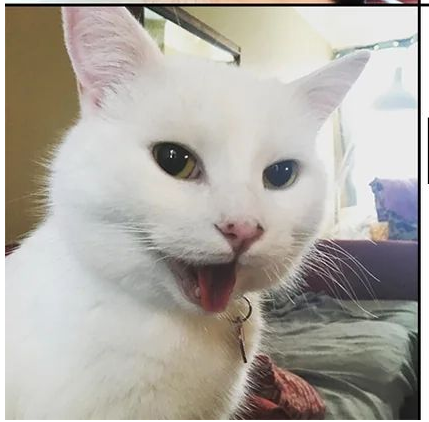
Has bad opinions

Not British

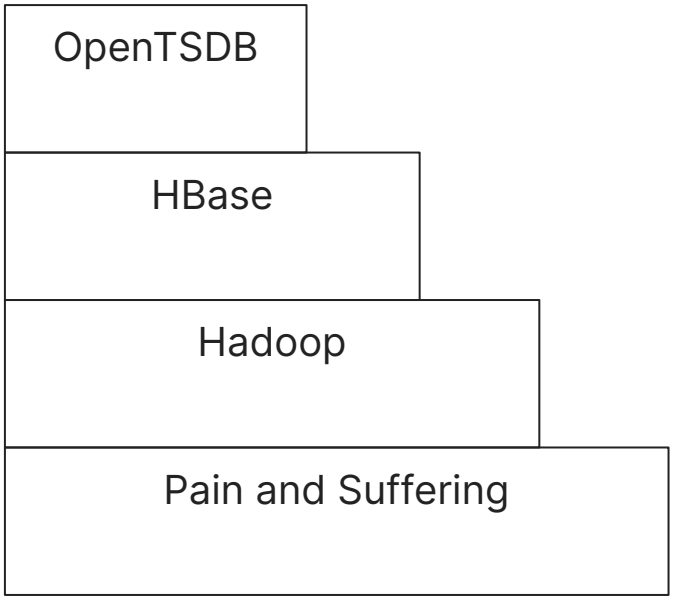
CLOUDFLARE



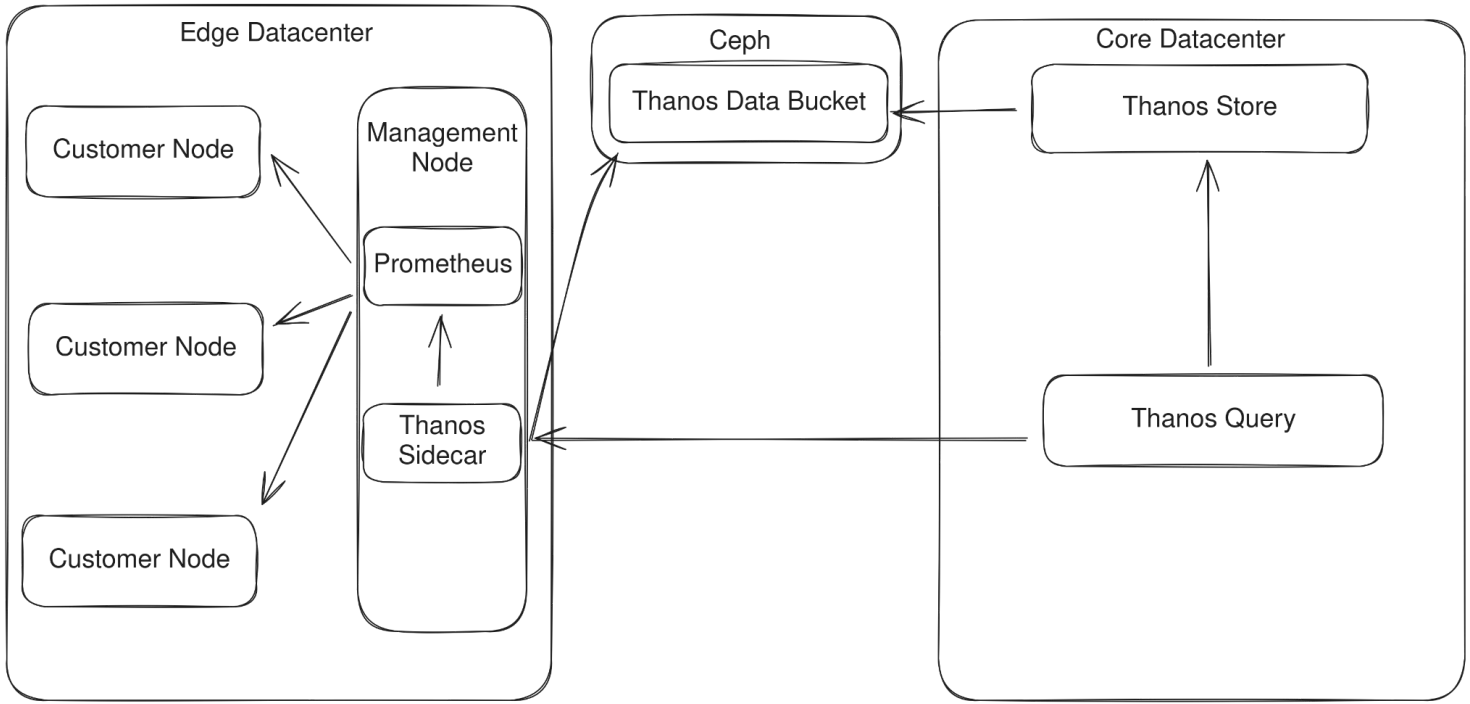
MAINTAINING OPENTSDB



ROLLING THE DICE ON A NEW UNTESTED TECHNOLOGY

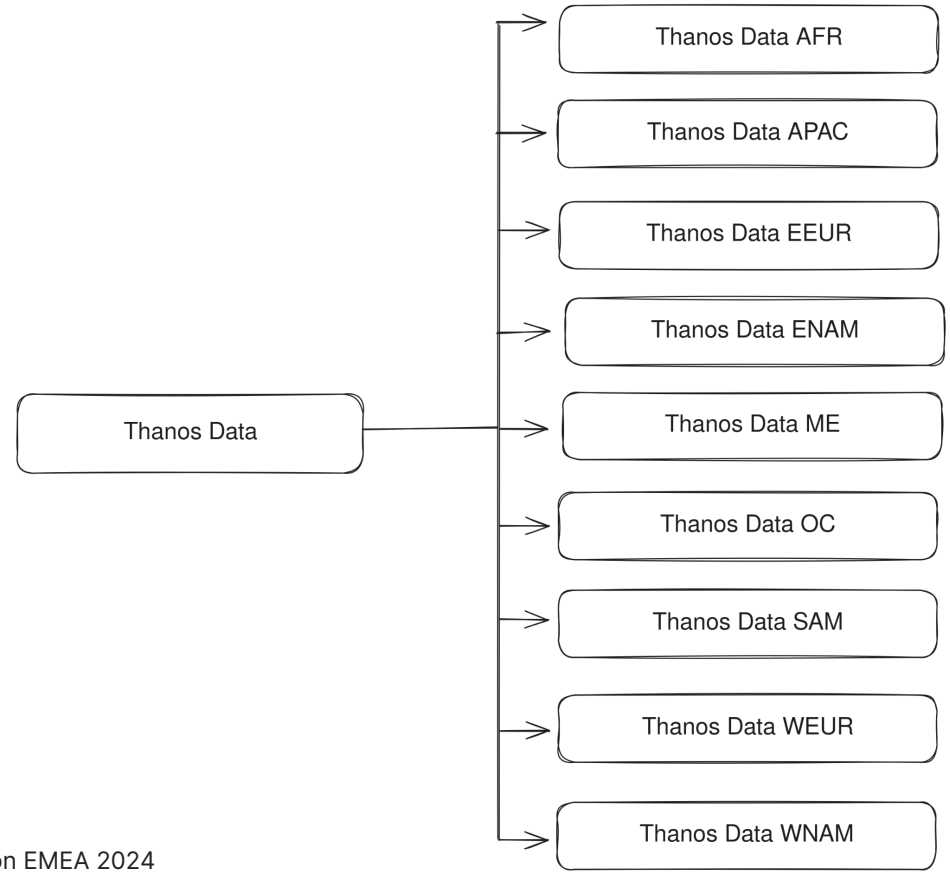

OpenTSDB
HBase
Hadoop
Pain and Suffering

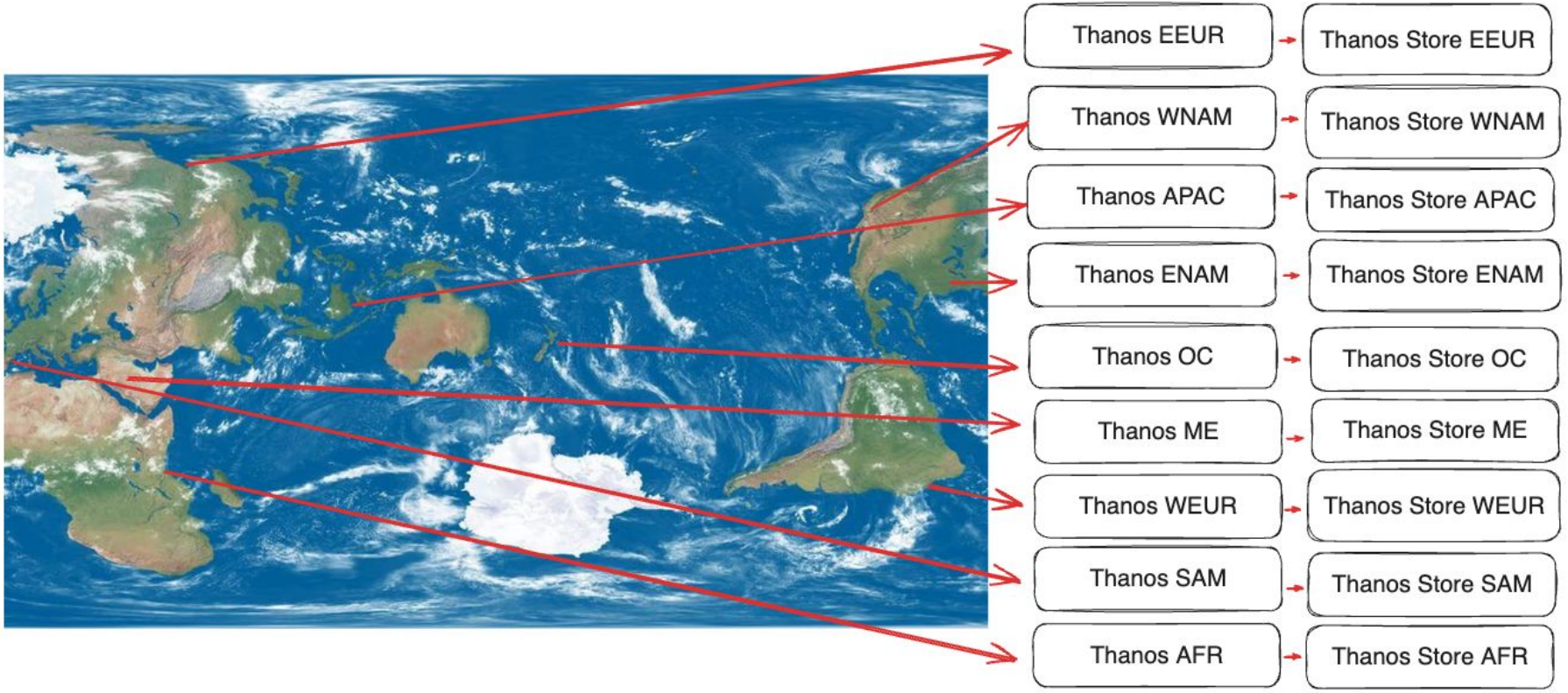

9?!

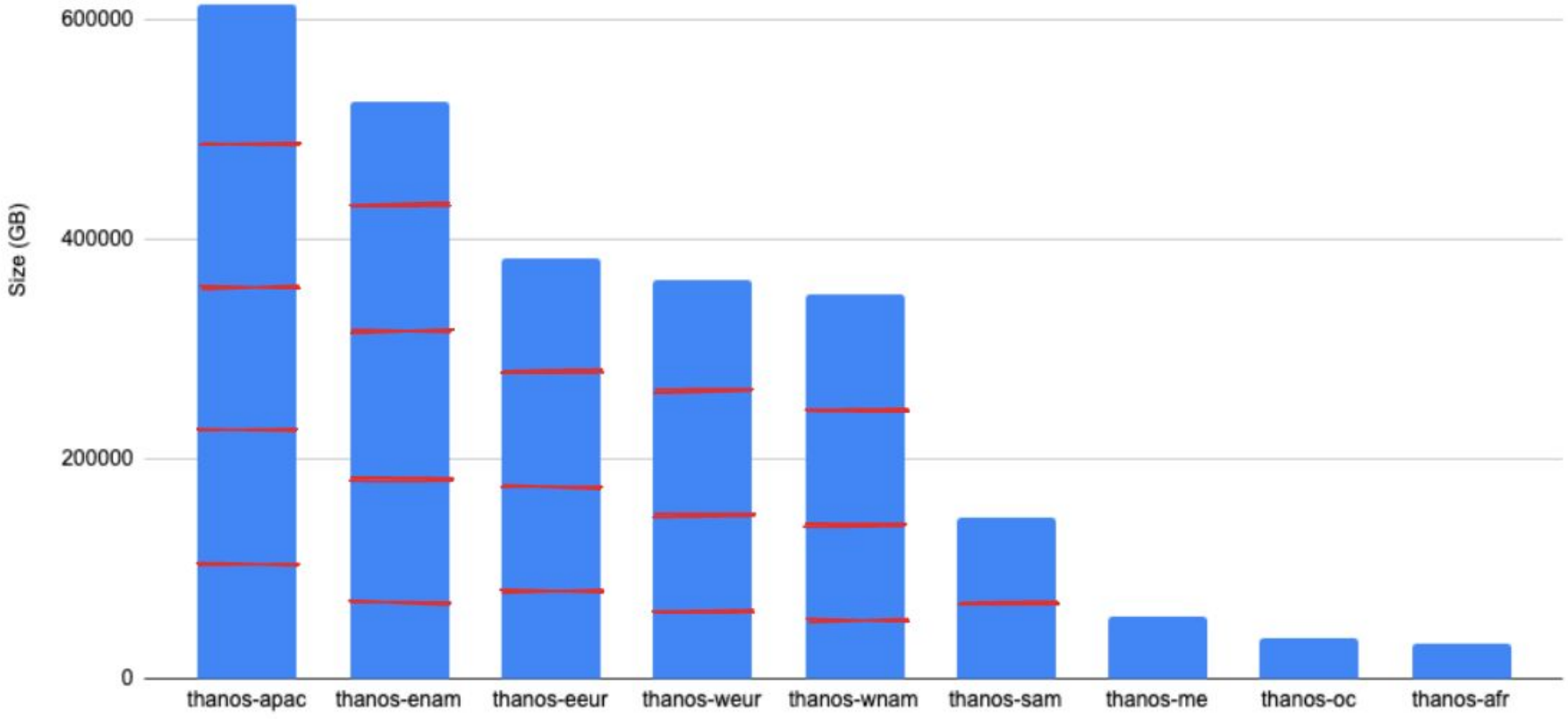

Edge Datacenter
Customer Node
Customer Node
Customer Node
Management Node
Prometheus
Thanos Sidecar
Ceph
Thanos Data Bucket
Core Datacenter
Thanos Store
Thanos Query

It’s 2019 and we’ve got issues

- Stuck with Vertical Scaling, with a lack of time partitioning
- Compactors were running out of memory
- Everything was falling apart
- No one was having a good time


Thanos Data
Thanos Data AFR
Thanos Data APAC
Thanos Data EEUR
Thanos Data ENAM
Thanos Data ME
Thanos Data OC
Thanos Data SAM
Thanos Data WEUR
Thanos Data WNAM


Thanos EEUR
Thanos Store EEUR
Thanos WNAM
Thanos Store WNAM
Thanos APAC
Thanos Store APAC
Thanos ENAM
Thanos Store ENAM
Thanos OC
Thanos Store OC
Thanos ME
Thanos Store ME
Thanos WEUR
Thanos Store WEUR
Thanos SAM
Thanos Store SAM
Thanos AFR
Thanos Store AFR

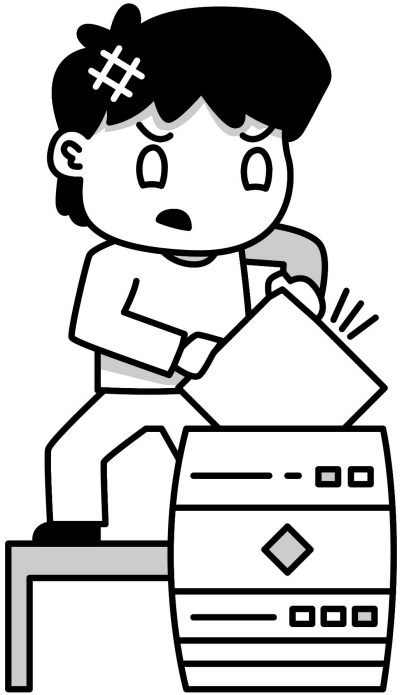

600000
400000
200000
0
Size (GB)
thanos-apac
thanos-enam
thanos-eeur
thanos-weur
thanos-wnam
thanos-sam
thanos-me
thanos-oc
thanos-afr

We were mostly doing ***vibe based scaling***:

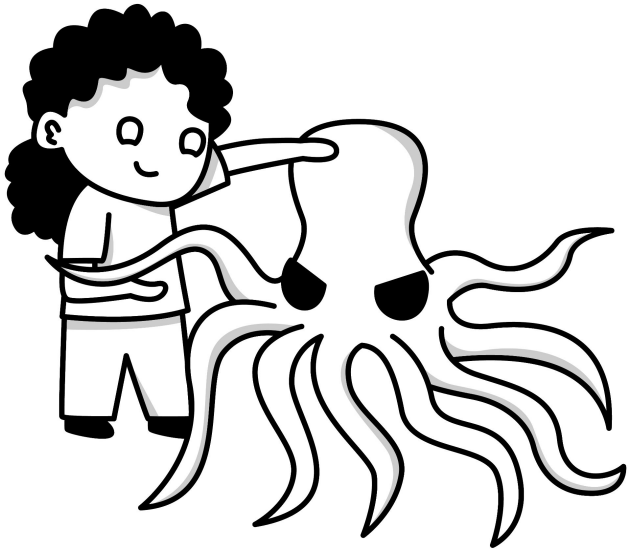
Create a new store when things *felt* slow

CLOUDFLARE

Welcome to Scaling Thanos in Dynamic Prometheus Environments



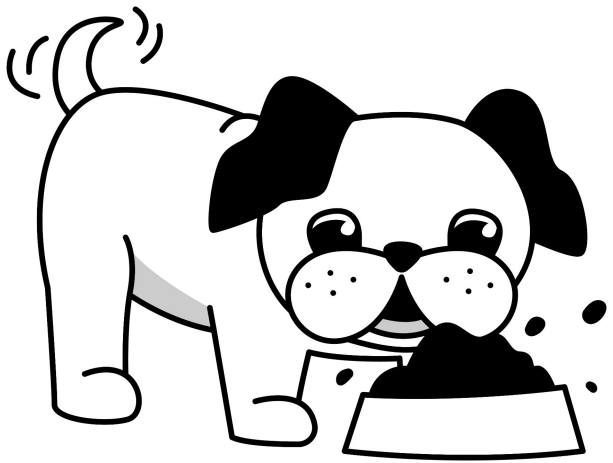
Ceph



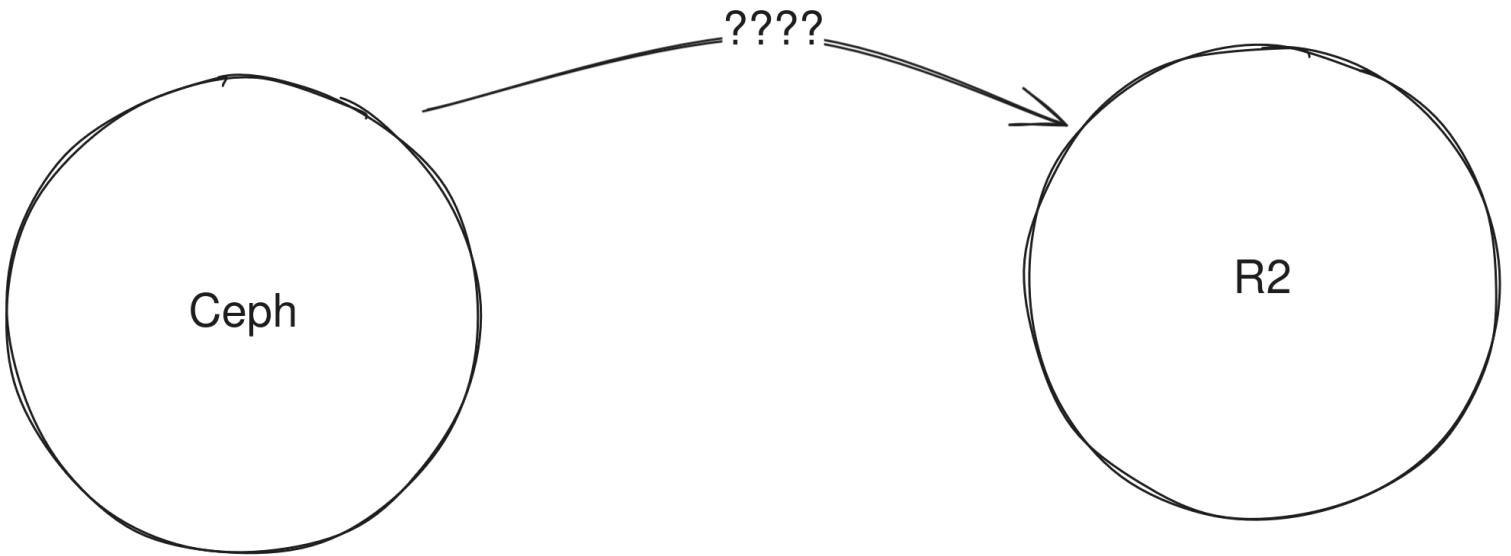
The worst storage system, except for all the others

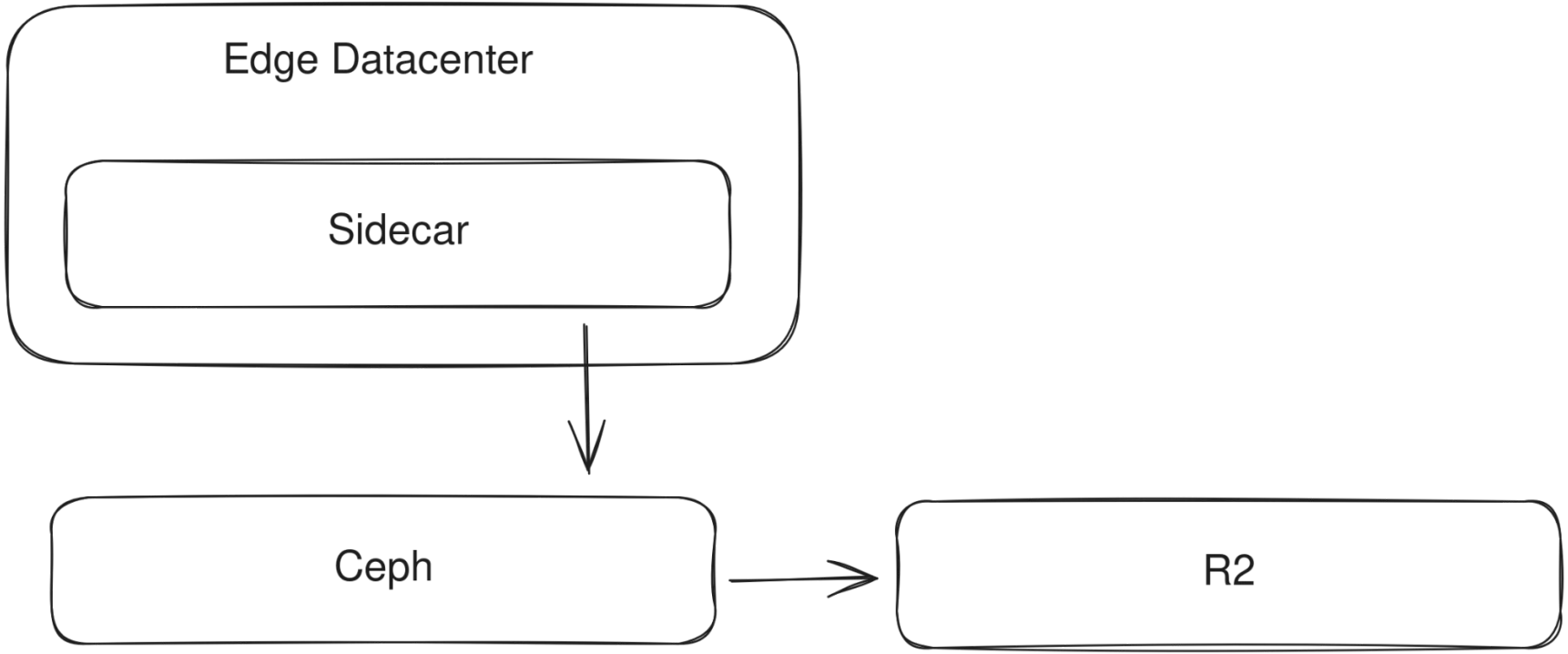
Cloudflare R2

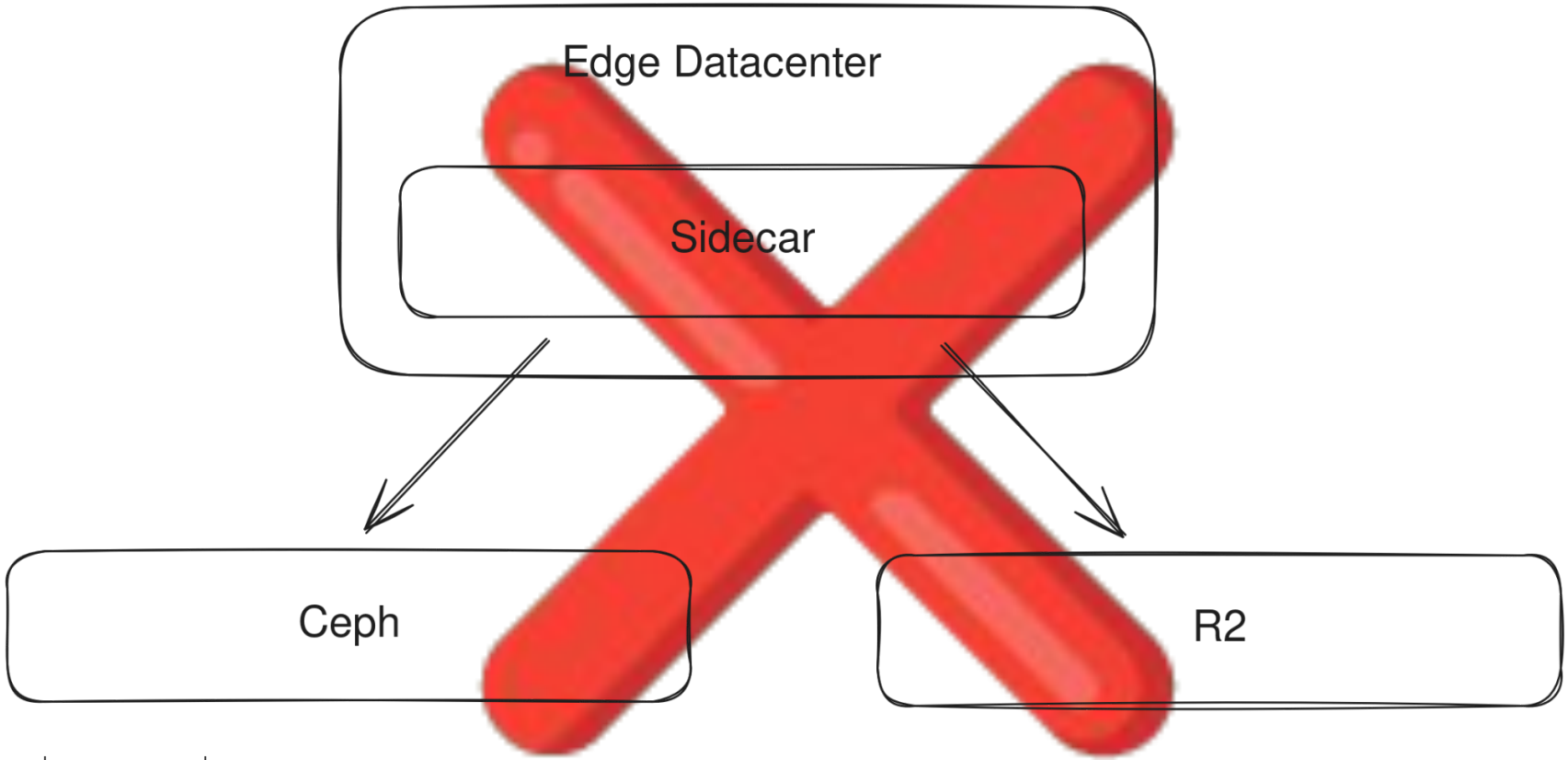
- We didn’t have to manage it
- Dogfooding!
- Decentralising our storage



How do we migrate?




Edge Datacenter
Sidecar
Ceph
R2


Edge Datacenter
Sidecar
Ceph
R2

CLOUDFLARE

```
const (
        // MetaFilename is the known JSON filename for meta information.
        MetaFilename = "thanos.shipper.json"
```

Infinite Sadness

CLOUDFLARE

✕ Allow customizing the shipper metadata file name

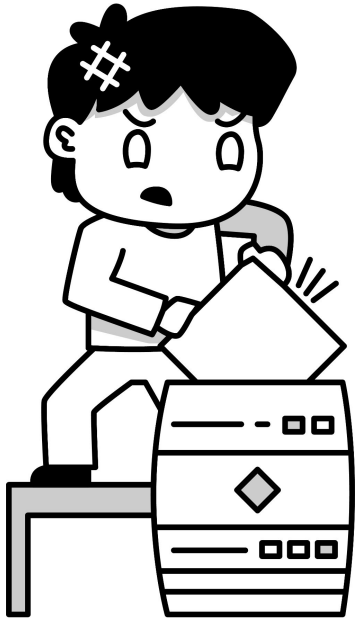
```
Currently, the shipper metadata file is always called `thanos.shipper.json`
in the Prometheus data directory. This precludes running multiple sidecars
that upload to different object stores, as they will overwrite each other's
metadata file.

This commit allows the metadata file name to be customized via a flag. The
default is unchanged, but it can be overridden with the `--shipper.meta-file-name`
flag.

As part of this, we update the signatures of `WriteMetaFile` and `ReadMetaFile`
to take the full path of the metadata file, rather than just the directory,
and updates the tests that go along with this.

Signed-off-by: sinkingpoint <colin@quirl.co.nz>
```

main (#6886)

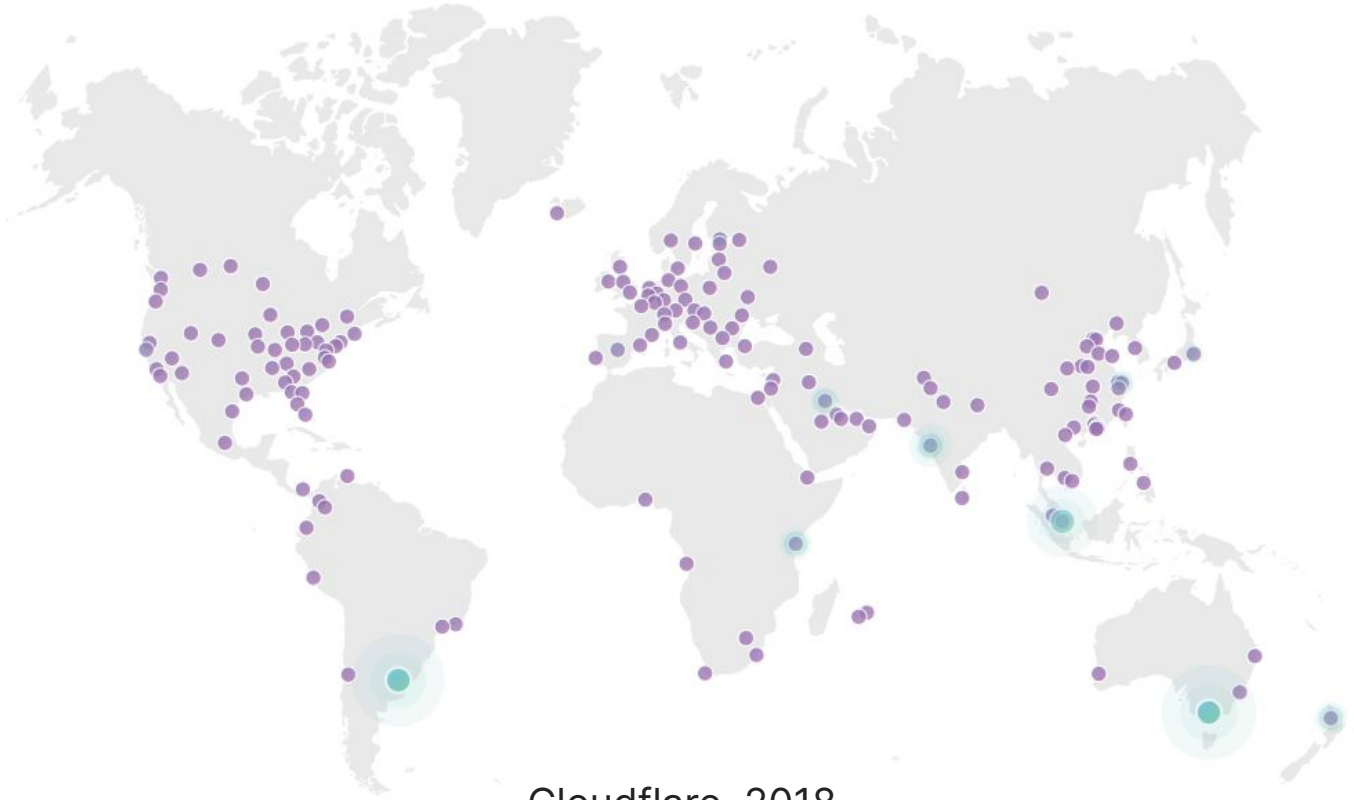
v0.34.0 ... v0.34.0-rc.0

Buckets, buckets everywhere

And far too much data to drink



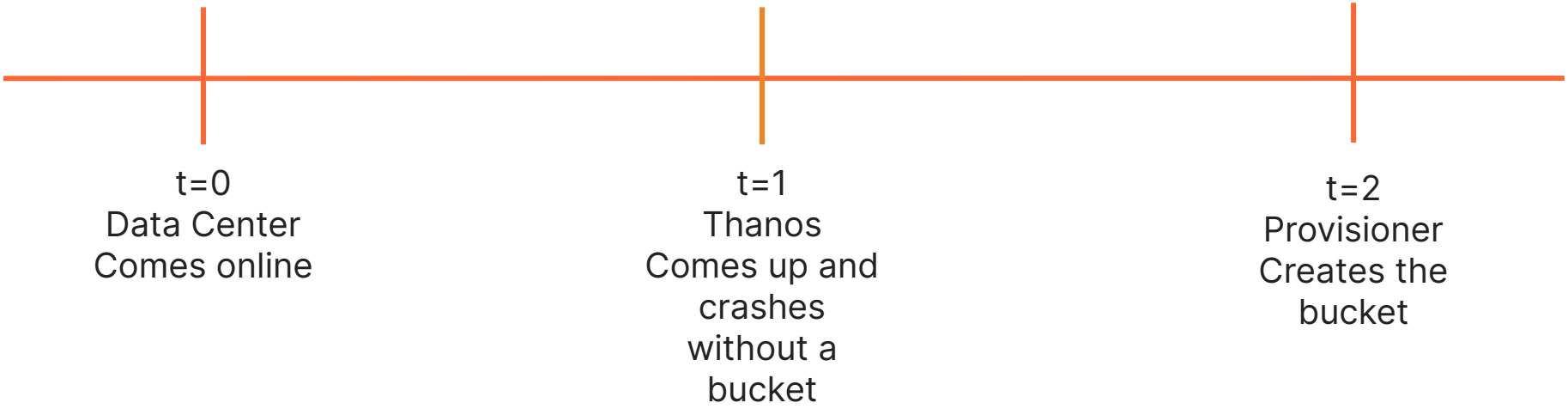
Bucket Granularity **puts a limit** on how much you can scale out because a store has to be able to handle a given time range



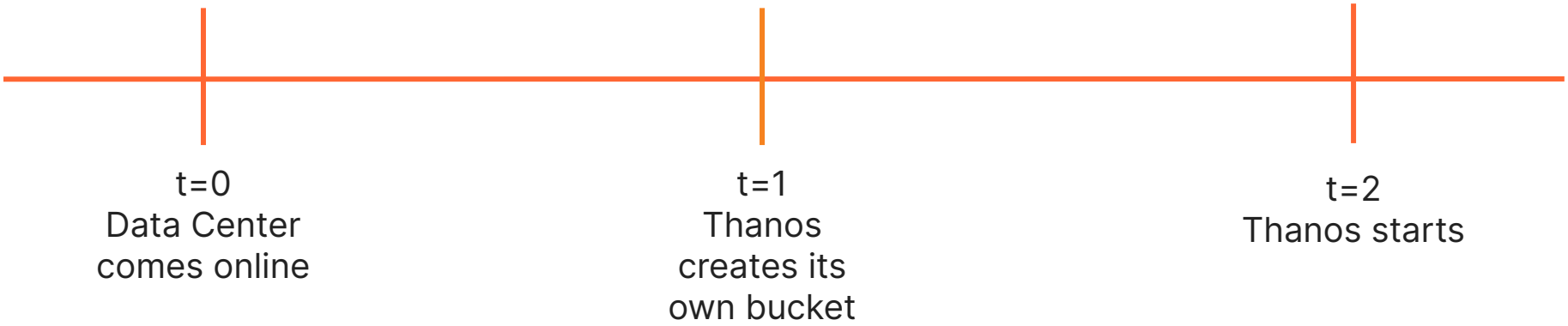
Cloudflare, 2018

CLOUDFLARE

Centralised provisioning leads to race conditions

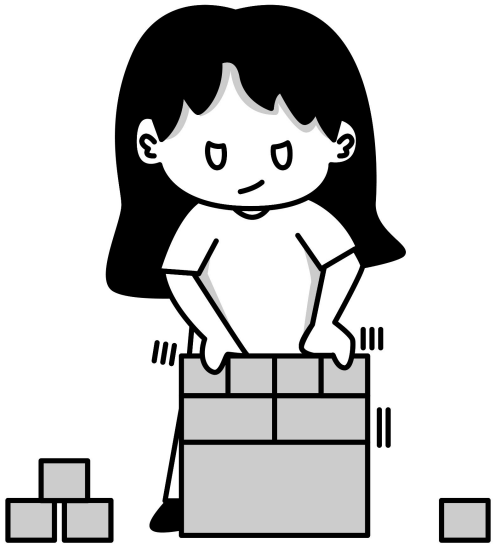


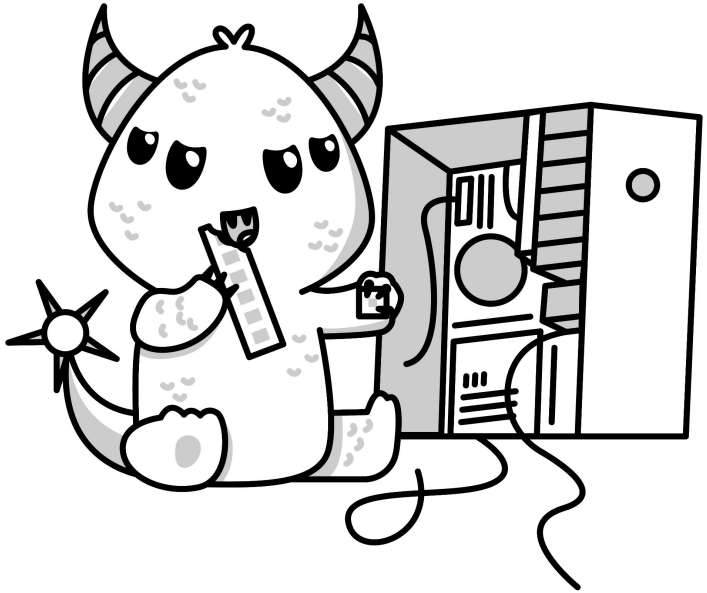
Distributed Provisioning works



Compactors compacting

Oh no where did all my memory go





Compactors are:

- ***Bursty***
- ***Heavy***
- ***Kind of awkward***

CLOUDFLARE

But where to put them?

Centralised in core?

Distributed on the edge?

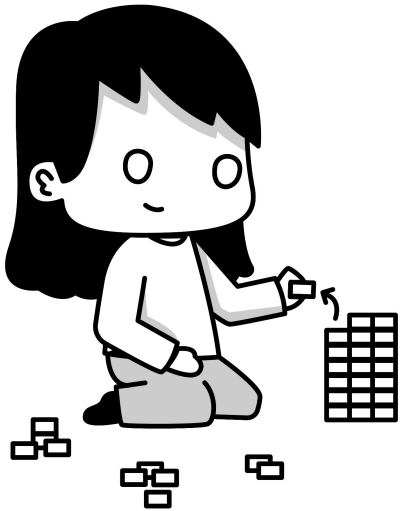

02/07 12:00
02/08 00:00
02/08 12:00
02/09 00:00
02/09 12:00
02/10 00:00
02/10 12:00
02/11 00:00
02/11 12:00
02/12 00:00
02/12 12:00
02/13 00:00
02/13 12:00
02/14 00:00

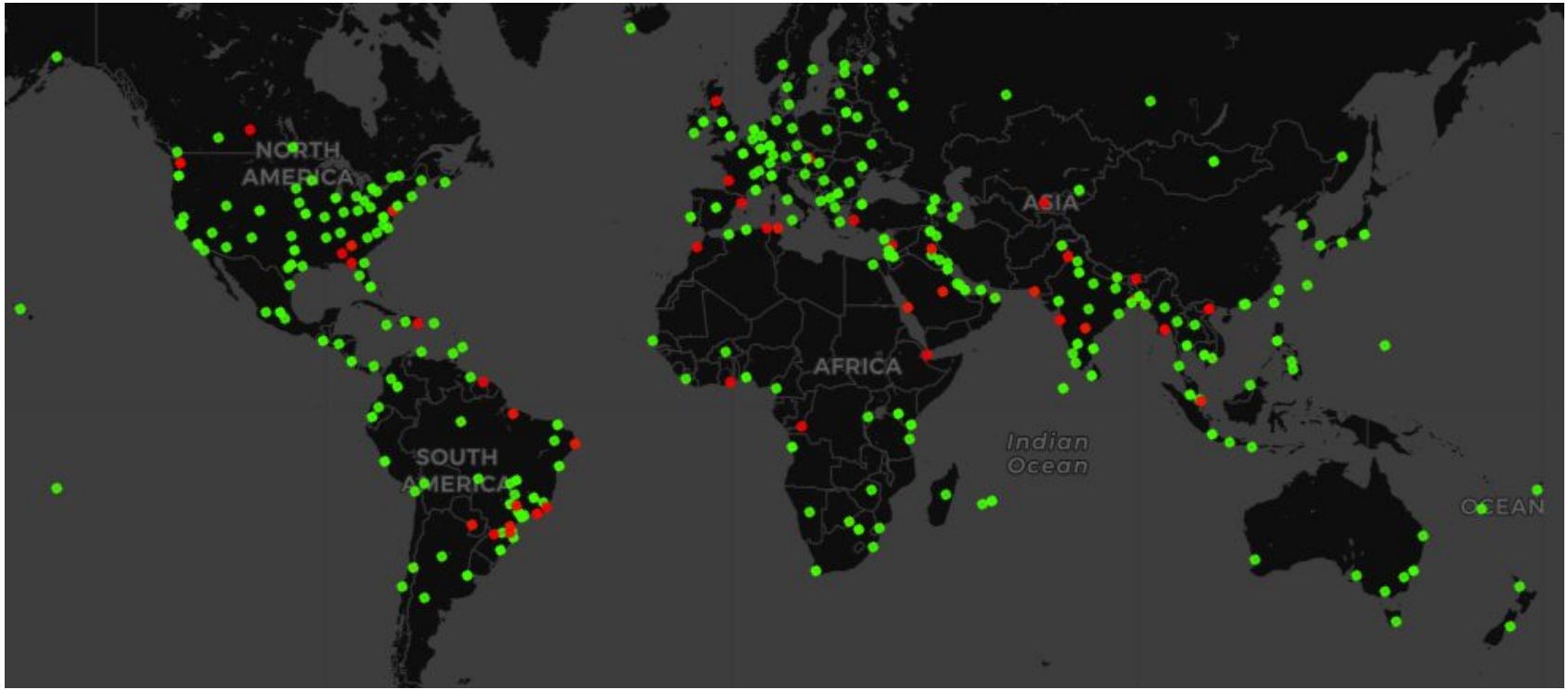
```
job "thanos-compactor" {
 periodic {
   cron = "{{ low_time_start_minute }} {{ low_time_start_hour }} * * * *"
 }
 group "compactor" {
   driver = "docker"
   config {
     image = "docker-registry/thanos:latest"
     command = "timeout"
     args = [
       "{{ low_time_length }}"
       "thanos"
       "compact"
     ]}}}
```

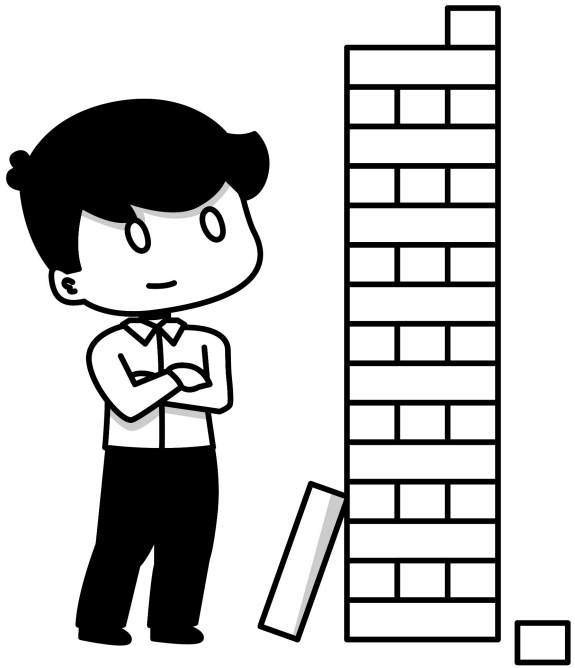
CLOUDFLARE

Stores Storing

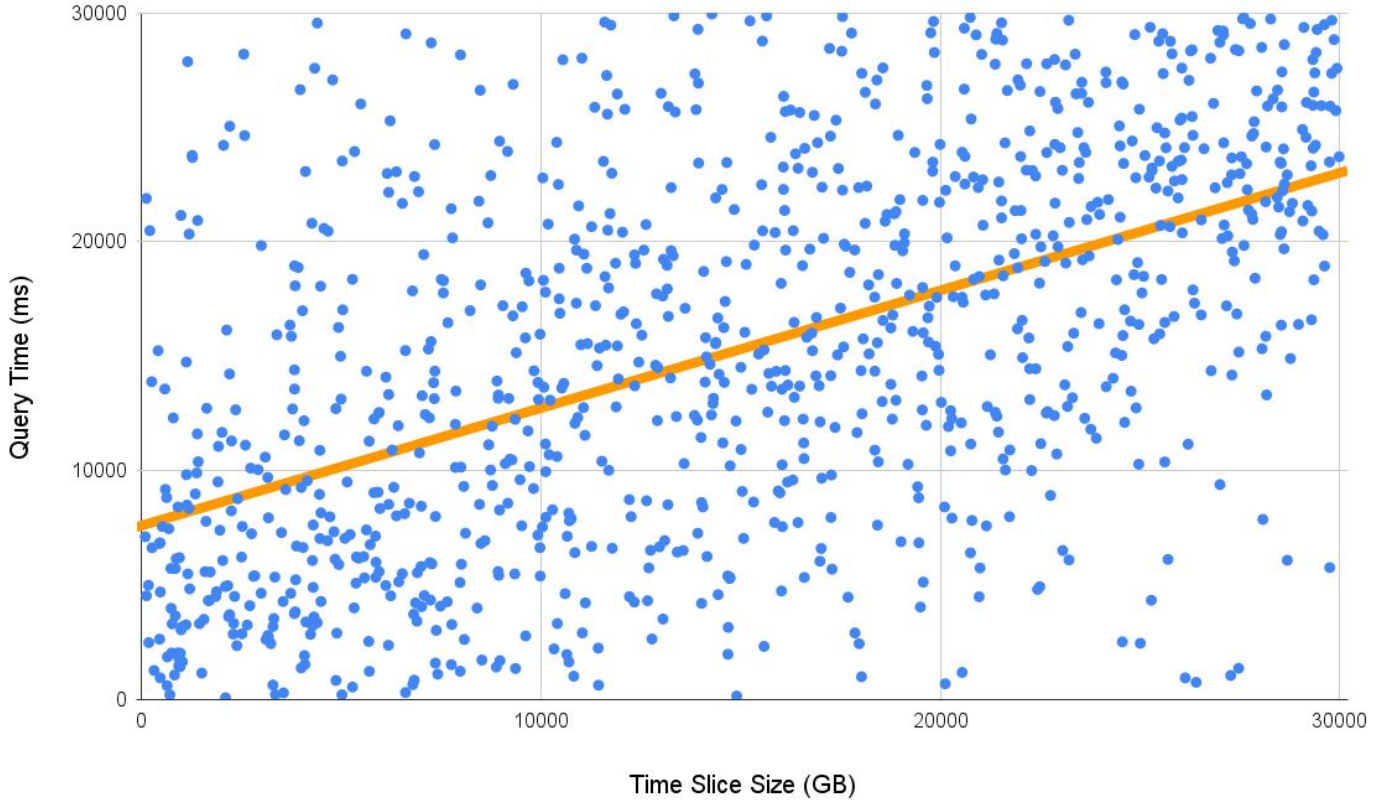
It turns out that querying halfway around the world is slow

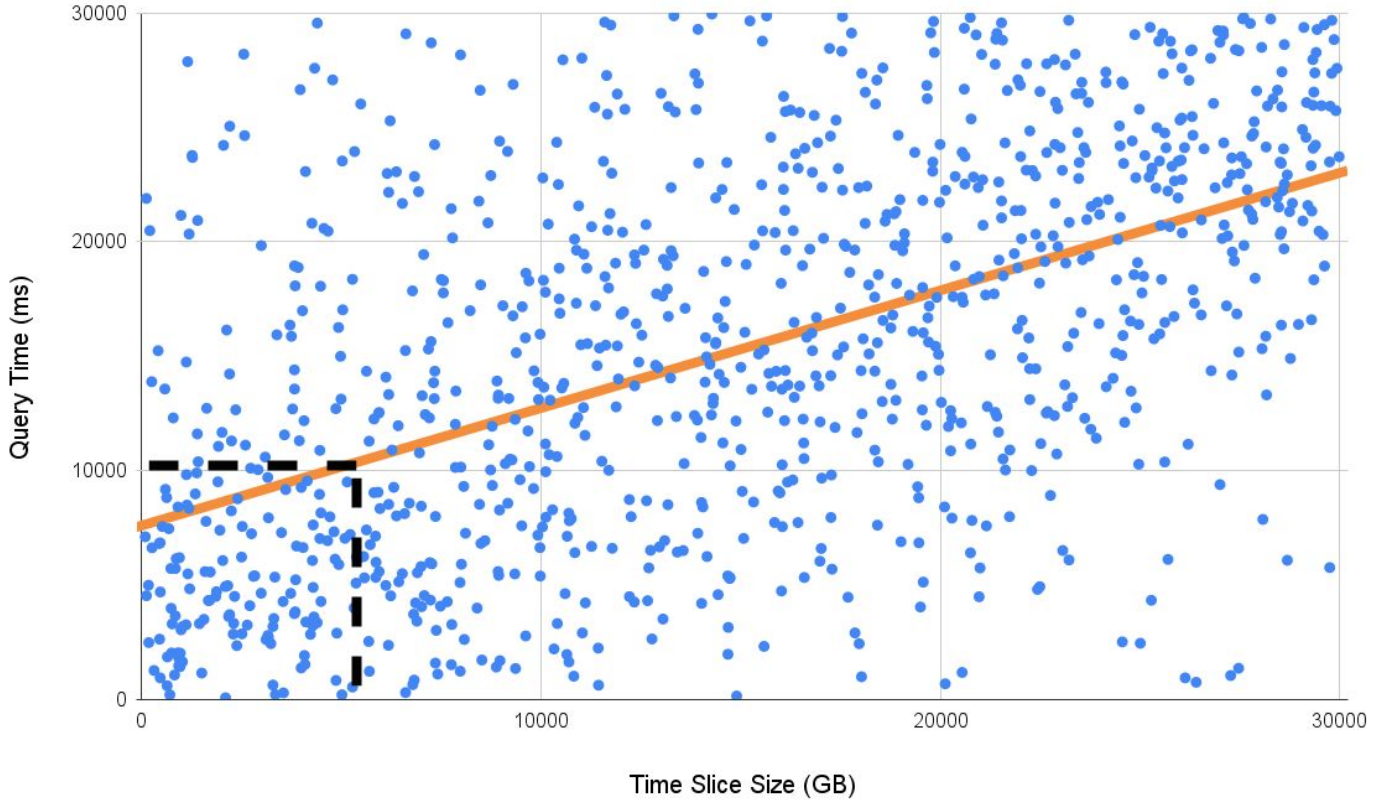



NORTH
AMERICA
SOUTH
AMERICA
AFRICA
ASIA
Indian
Ocean
OCEAN



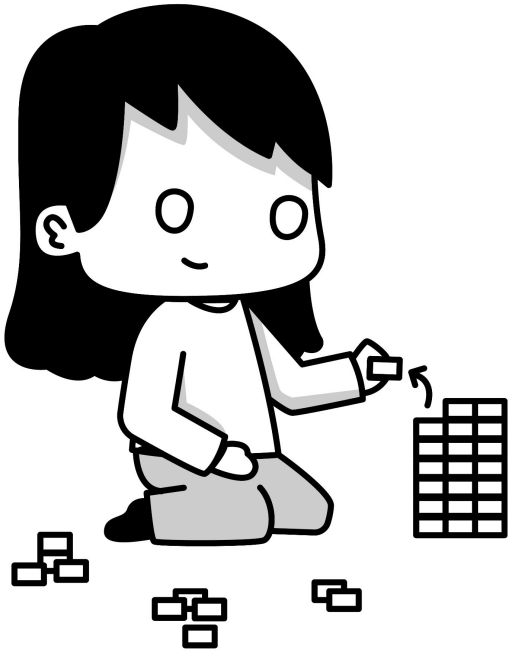
Stores have a **really frustrating** resource profile to scale

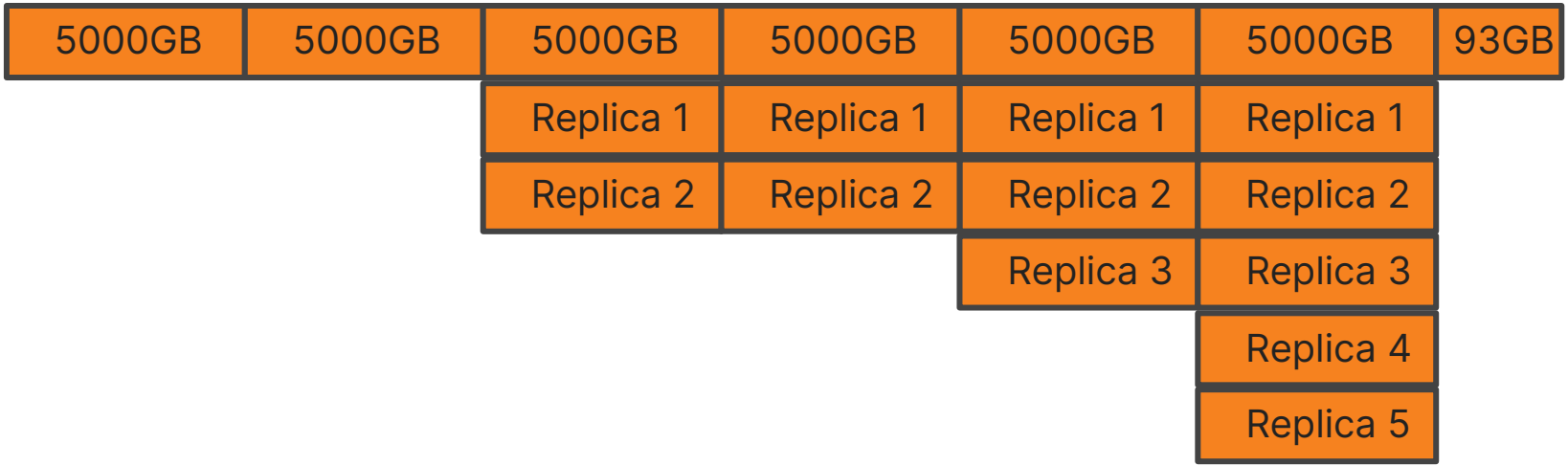

30000
20000
10000
0
Query Time (ms)
0
10000
20000
30000
Time Slice Size (GB)


30000
20000
10000
0
Query Time (ms)
0
10000
20000
30000
Time Slice Size (GB)

5000GB	5000GB	5000GB

5000GB	5000GB	5000GB	93GB

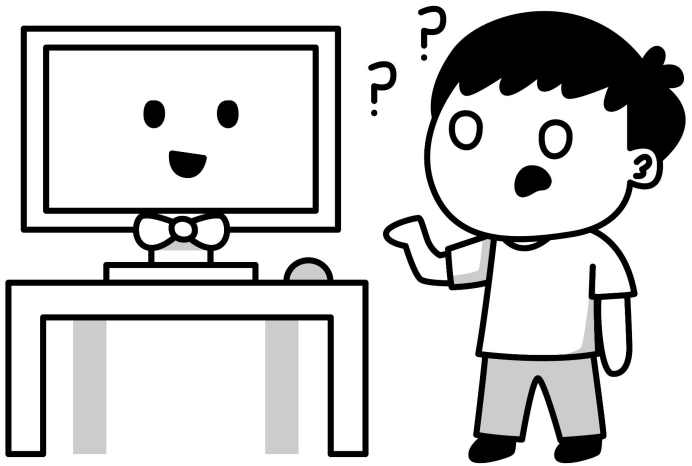


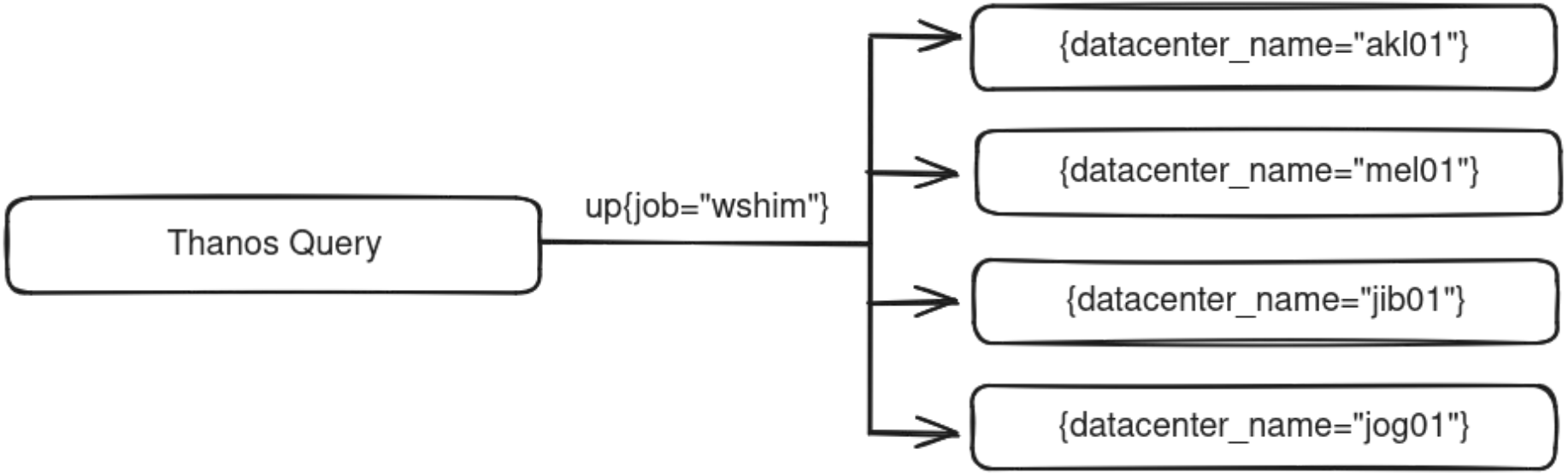

5000GB
5000GB
5000GB
5000GB
5000GB
5000GB
93GB
Replica 1
Replica 1
Replica 1
Replica 1
Replica 2
Replica 2
Replica 2
Replica 2
Replica 3
Replica 3
Replica 4
Replica 5

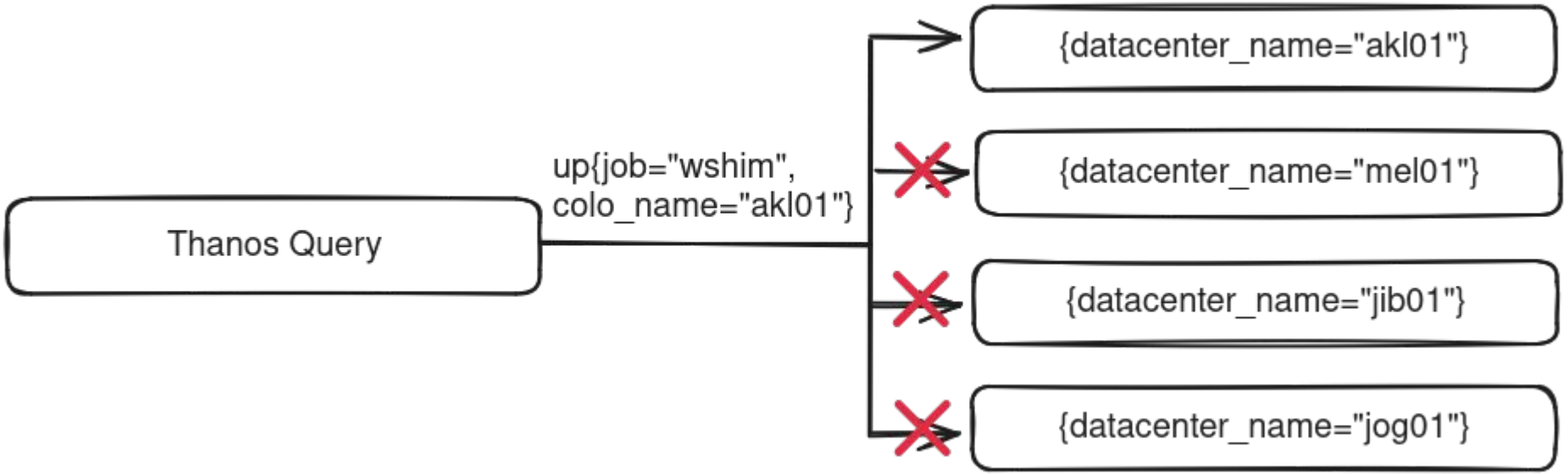
CLOUDFLARE

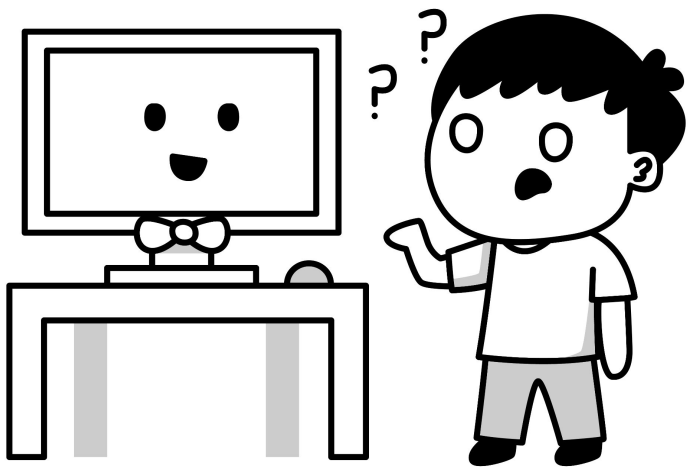
Querys Querying

Or: “Why is my Thanos query always timing out?!?!?”




Thanos Query
up{job="wshim"}
{datacenter_name="akl01"}
{datacenter_name="mel01"}
{datacenter_name="jib01"}
{datacenter_name="jog01"}


Thanos Query
up{job="wshim",
colo_name="akl01"}
{datacenter_name="akl01"}
{datacenter_name="mel01"}
{datacenter_name="jib01"}
{datacenter_name="jog01"}



The **Label Enforcer** forces teams to use external label filters, and guides them towards fast queries

Bad: `my_metric`
Better: `my_metric{datacenter_name="akl01"}`

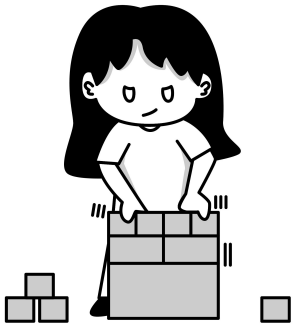
Bad: `my_metric{datacenter_name="akl01"} + my_second_metric`
Better: `my_metric{datacenter_name="akl01"} + my_second_metric{datacenter_name="akl01"}`

Also OK: `my_metric{datacenter_name=~"akl01|mel01"}`
Also OK: `my_metric{datacenter_name=~".+"}`

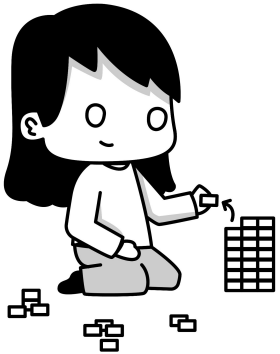
That’s about it

Automated bucket provisioning

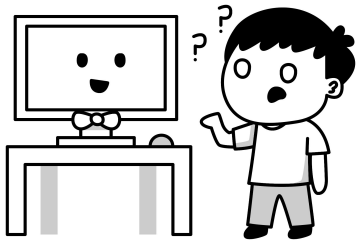
Compactors running with spare capacity



Stores that get broken up to meet our SLOs



Guiding engineers to fast Queries by default



CLOUDFLARE

Thanks!

Art by Alicia Edwards ^

Slides ^