apigee

Usergrid

Open Source Mobile Backend on Cassandra

Hi I’m
Ed Anuff

apigee

apigee

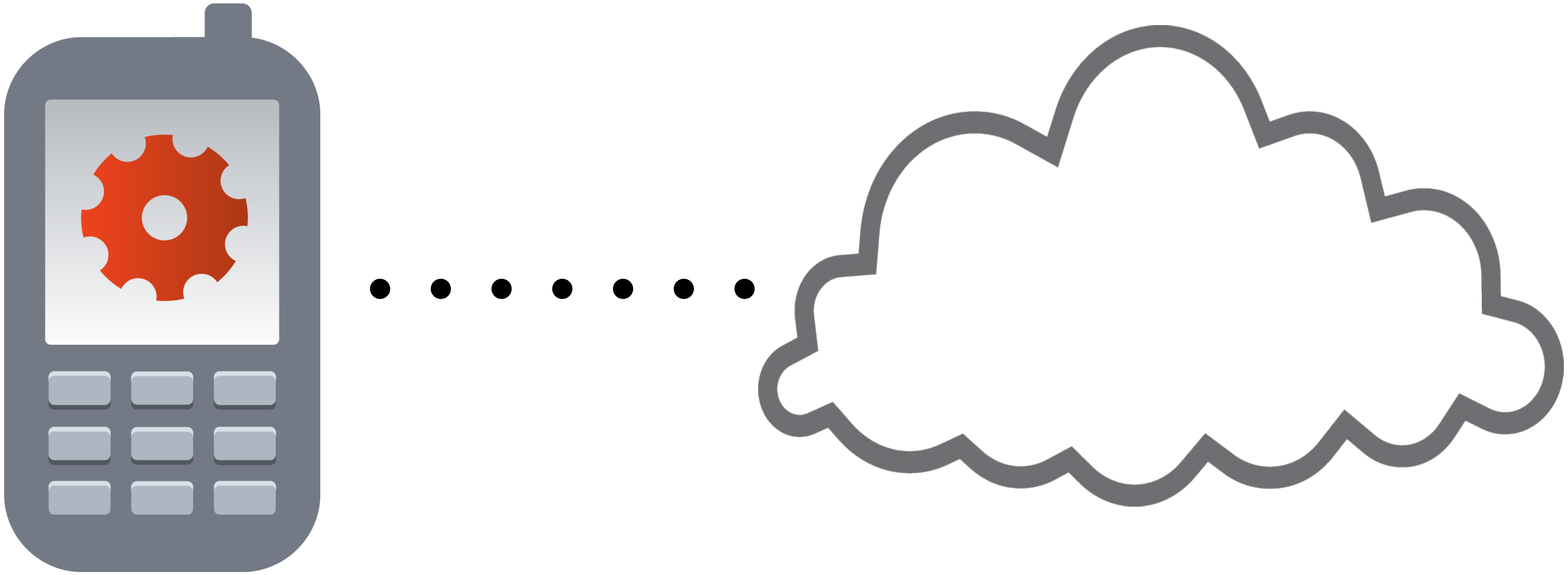
API Management
API Analytics
API Tools

Usergrid

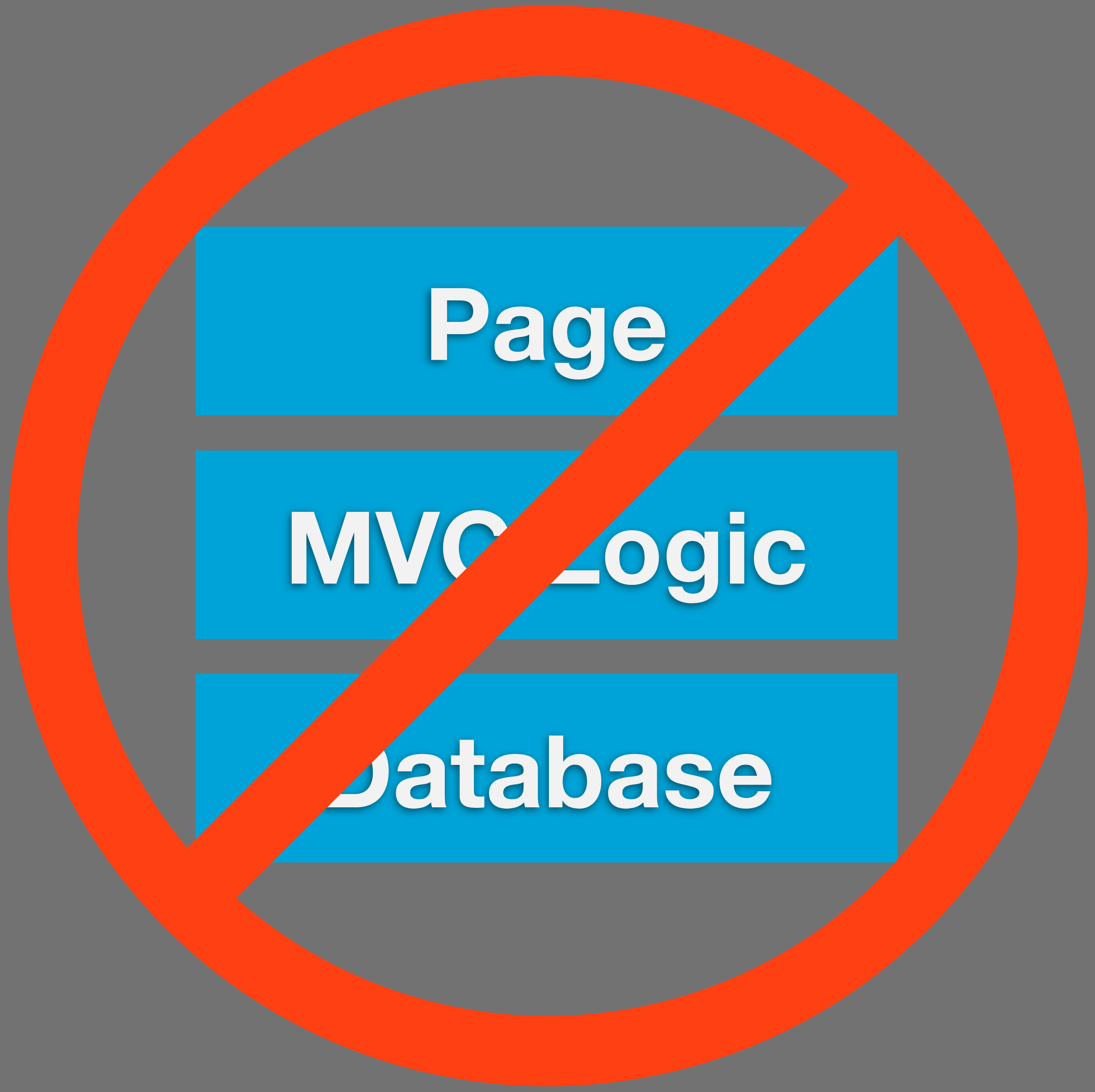
Backend-as-a-Service
& Open Source

Mobile and
Rich client apps

App, Cloud, & HTTP



No Web Stack

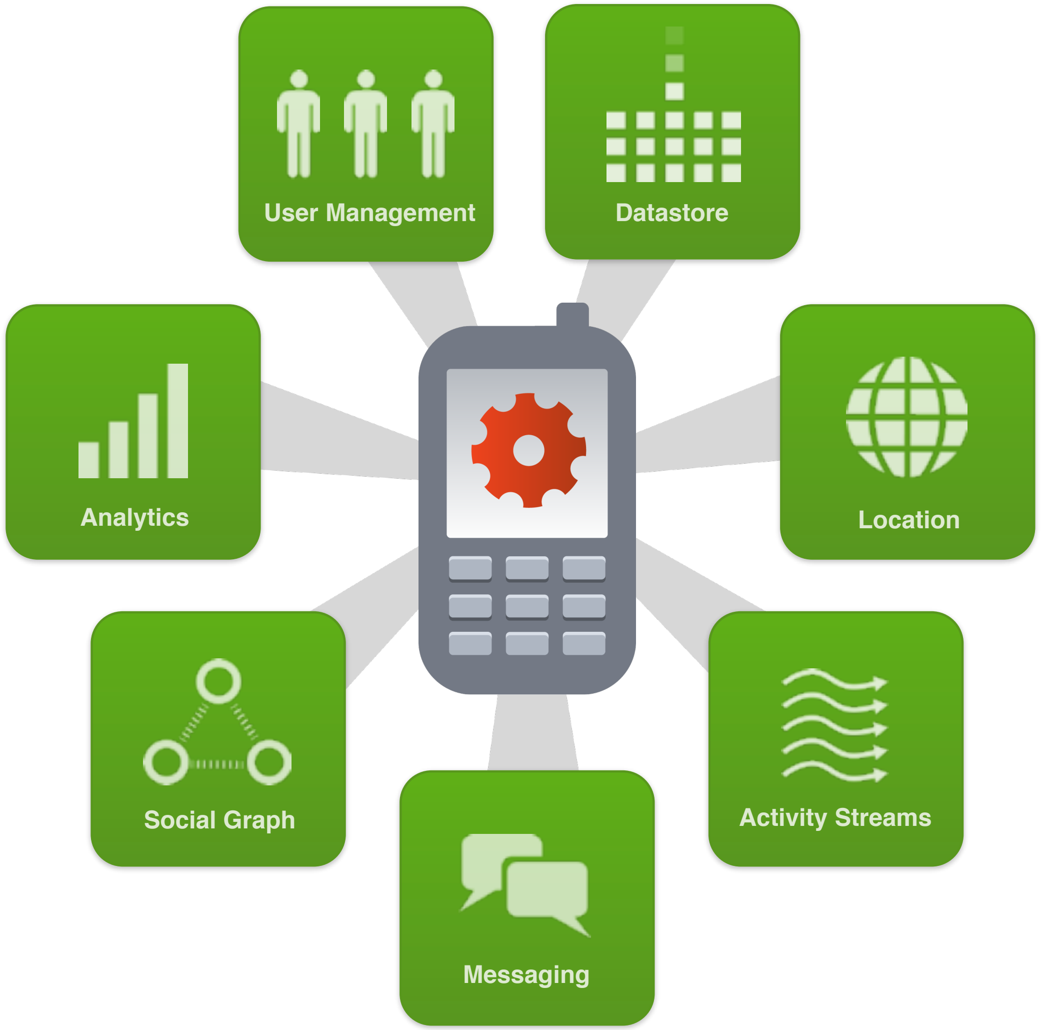


Page

MVC Logic

Database

Just Apps and APIs



Key Capabilities

A platform not a single app

Flexible data model

Easy to use from any language

Why should Cassandra developers care about Usergrid?

Open Source (APLv2)

Built on Hector

Multi-tenant (Virtual Keyspaces)

JSON and full-text indexing

Rich permissions model

REST API

Multi-tenant data for Apps

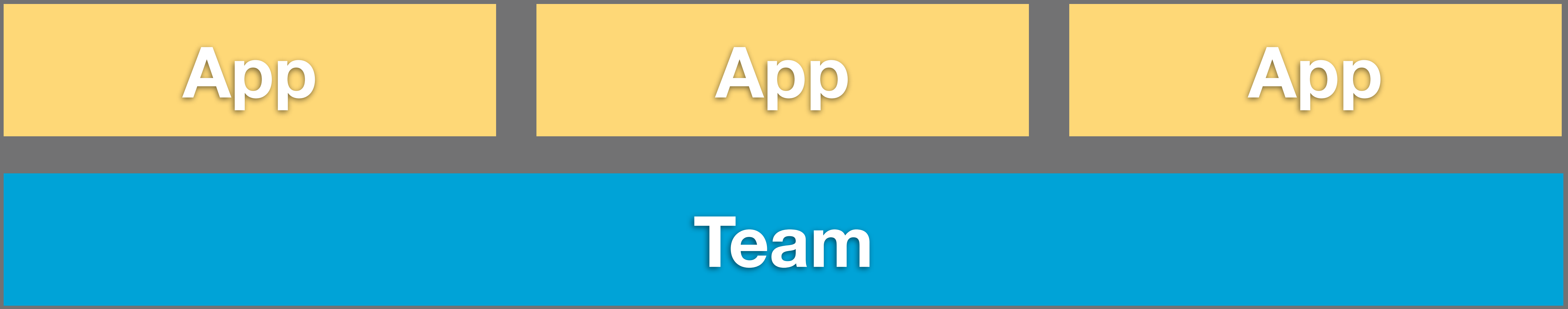
Lots of Teams...

Team

Team

Team

Lots of Apps...



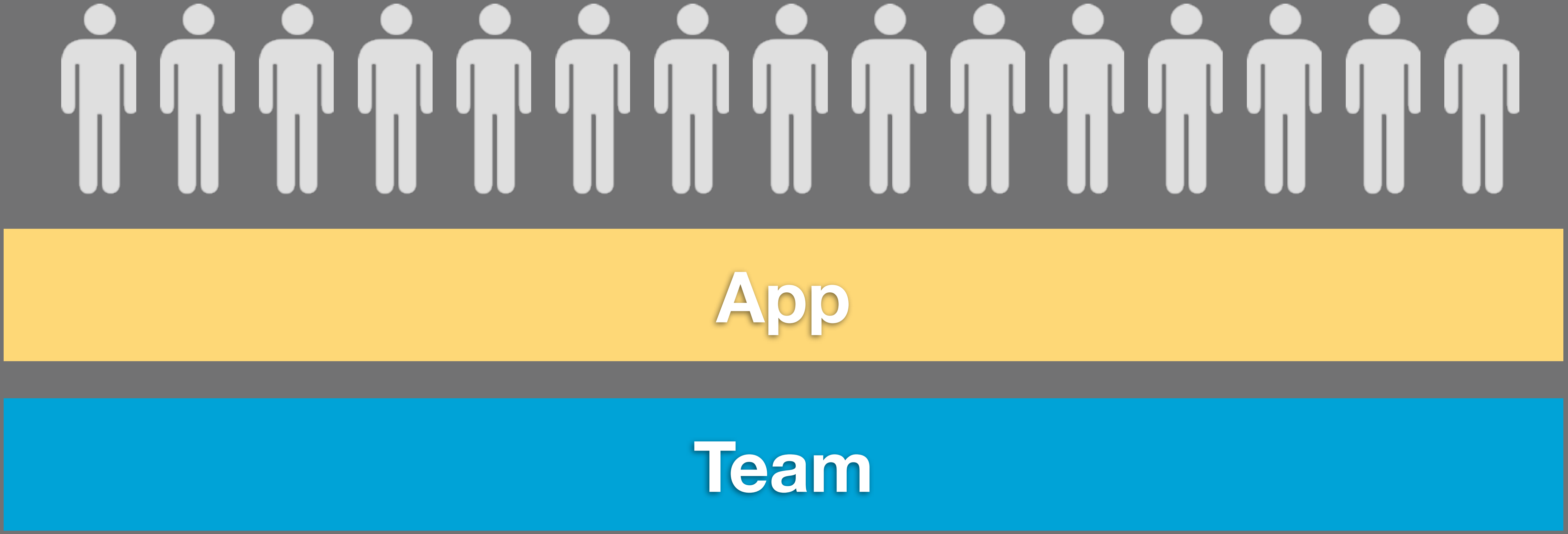
App

App

App

Team

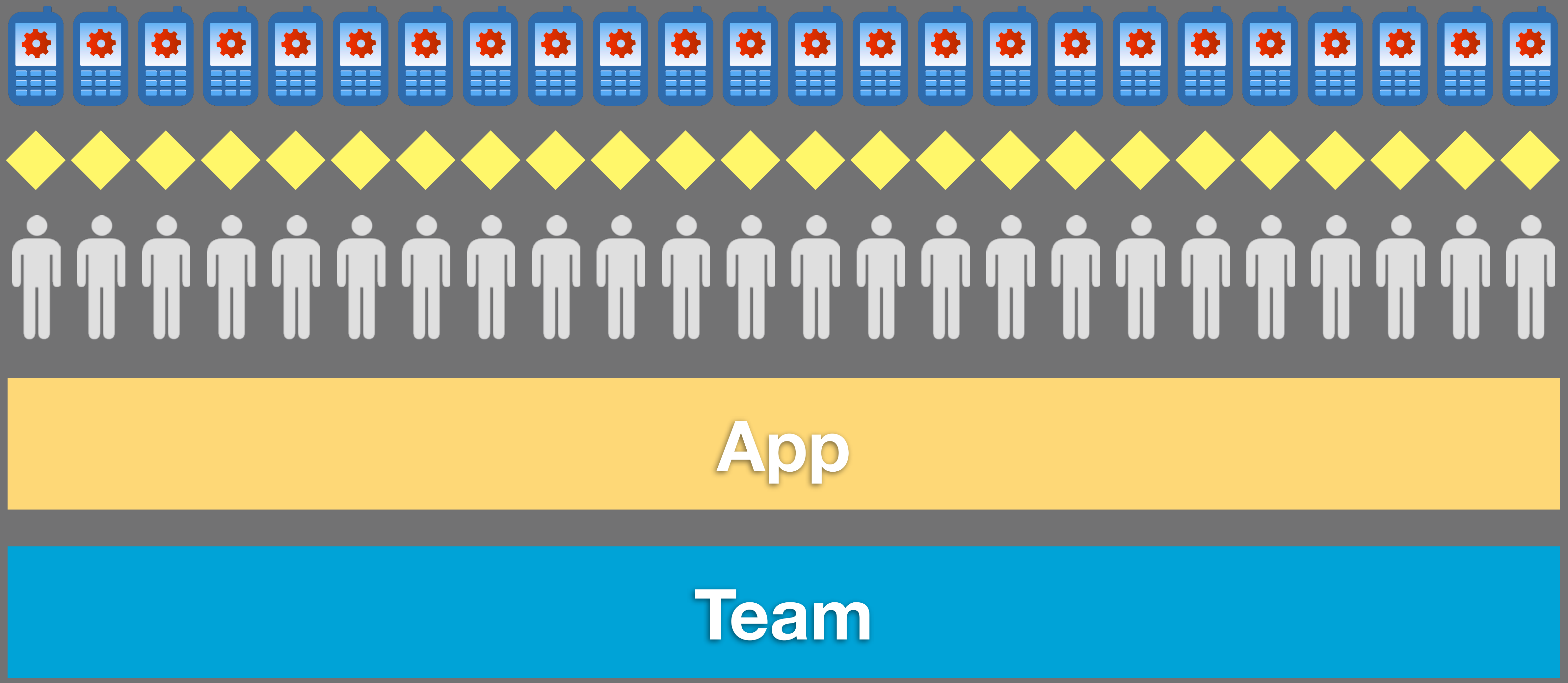
And App Users...



App

Team

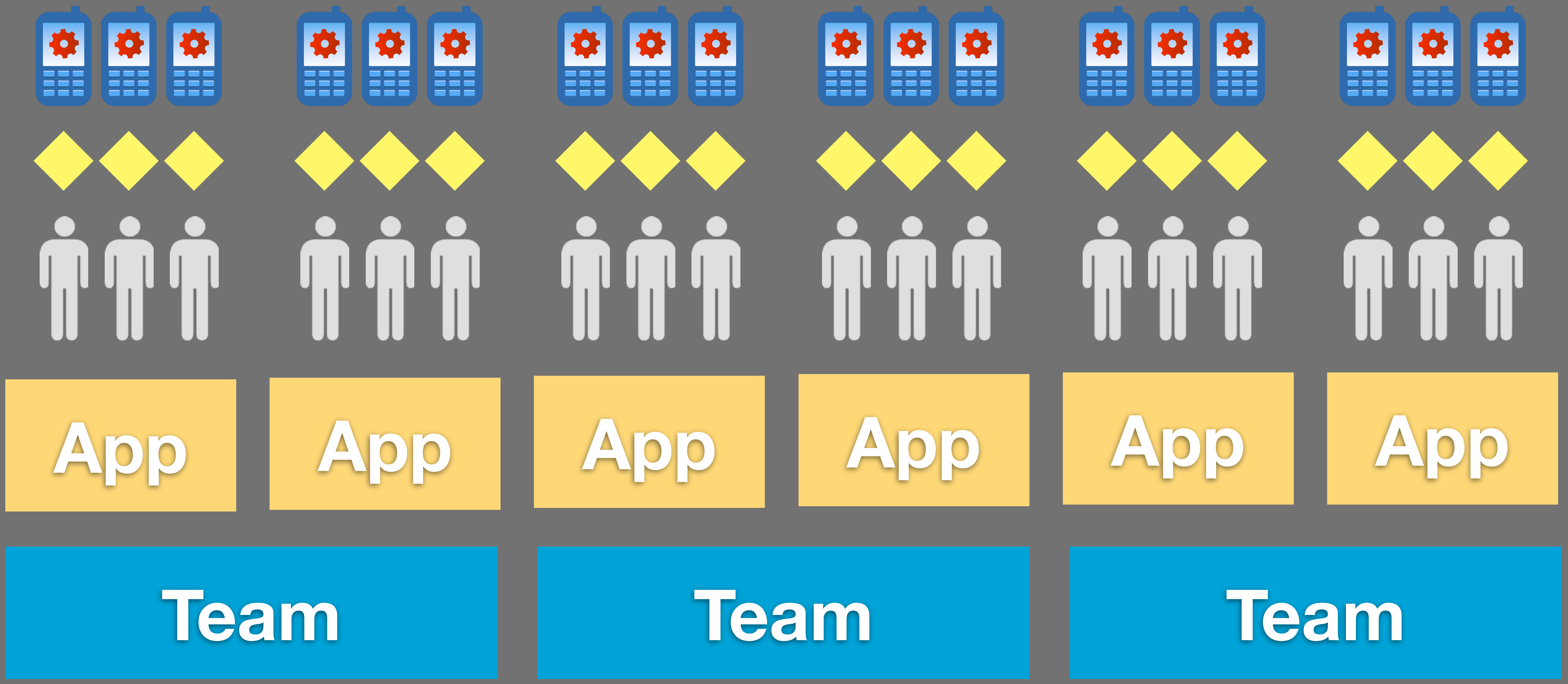
And data and devices



App

Team

Multi-tenancy at scale



App App App App App App

Team Team Team

	Separate Database	Shared Database
Shared Schema	- Scalability + Isolation + Simple	+ Scalability - No Isolation - Complicated
Separate Schema	- Scalability + Isolation + Not Complicated	- Scalability - No Isolation - Complicated

CREATE DATABASE

ALTER TABLE

Sharding SQL is too much work.

Oracle is too much money.

What about NoSQL?

Most NoSQL works at the wrong abstraction level.

Does **Cassandra** provide any more options?

Sort of

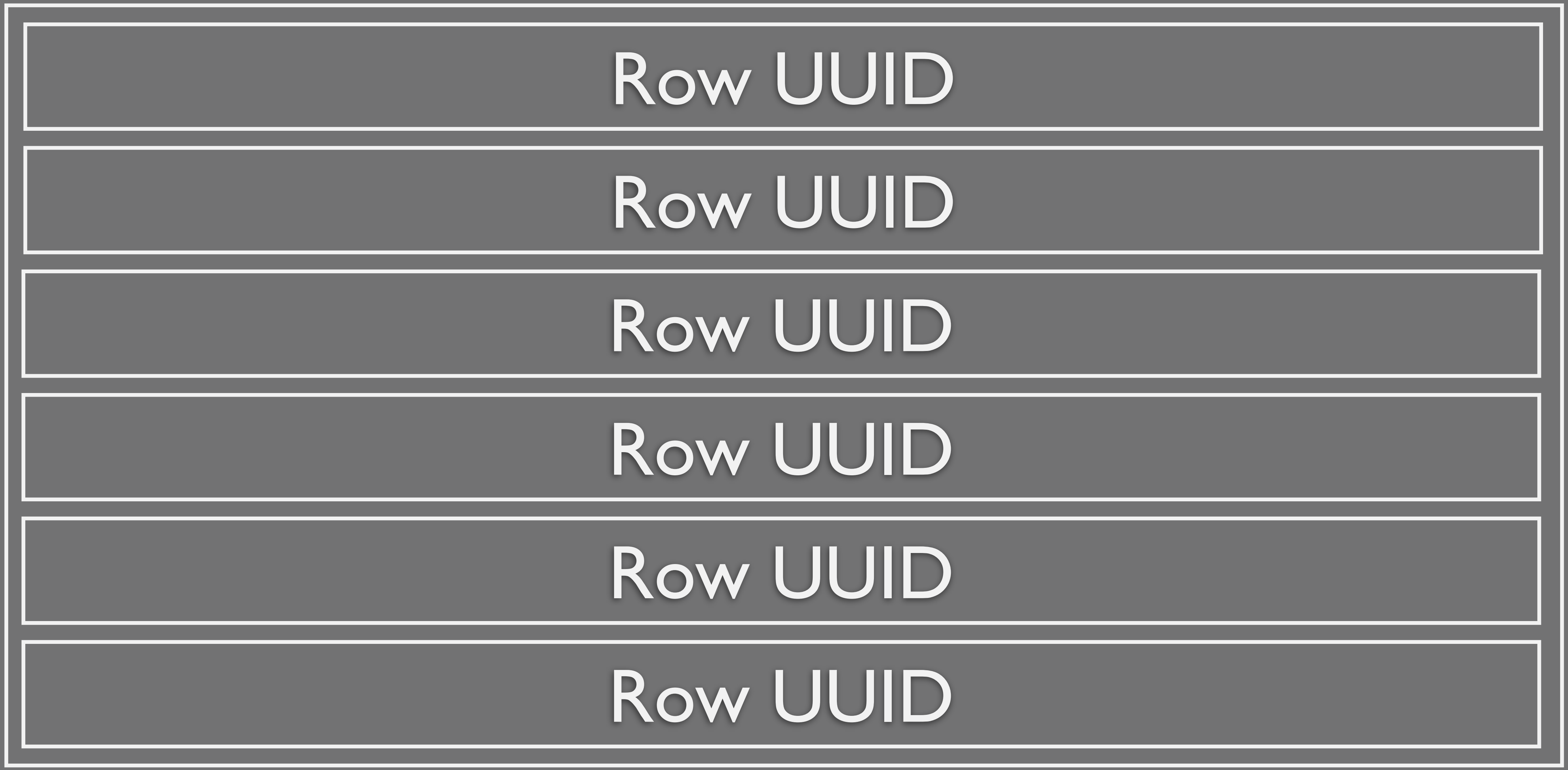
	Separate Keyspace	Shared Keyspace
Shared Schema	Expensive	Interesting
Separate Schema	Very Expensive	Unwieldy

How would Shared Schema/Shared Keyspace work?

```
Users = {
  "4e3c0423-…" : {
    username: "edanuff",
    email: "ed@anuff.com",
    … : …
  },
  "e5d61f2b-…" : {
    username: "jdoe",
    email: "john.doe@gmail.com",
    … : …
  }
}
```

```
Users = {
  "4e3c0423-…" : {
    username: "edanuff",
    email: "ed@anuff.com",
    … : …
  },
  "e5d61f2b-…" : {
    username: "jdoe",
    email: "john.doe@gmail.com",
    … : …
  }
}
```

Conventional Row Keys In Single Keyspace



Row UUID

Row UUID

Row UUID

Row UUID

Row UUID

Row UUID

Multi-tenant Row Keys In Shared Keyspace

Tenant ID	Row UUID
Tenant ID	Row UUID
Tenant ID	Row UUID
Tenant ID	Row UUID
Tenant ID	Row UUID
Tenant ID	Row UUID

“Wonderful, more things for me to build myself in the app tier...”

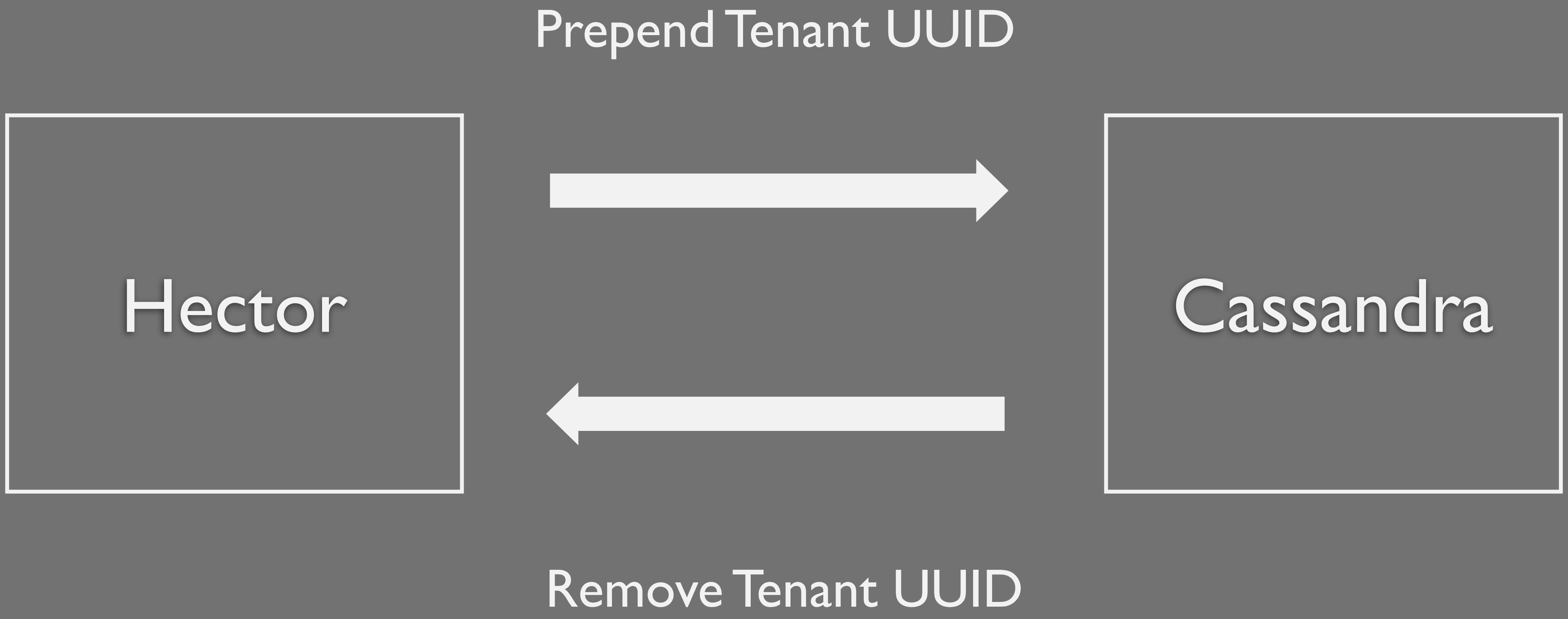
Hold that thought...

Shared Schema/
Shared Keyspace is
“Virtual Keyspaces”

Part of Hector Client

```
public static <E> Keyspace
 createVirtualKeyspace(
  String keyspace,
  E keyPrefix,
  Serializer<E> kps,
  Cluster cluster);
```

Hector hides the Tenant Prefixes



What about indexes?

Hector doesn’t filter results

Don’t add a tenant column

App-tier CF indexes preferable

What about CQL?

See previous slide for index limitations

Hector doesn’t currently support Virtual Keyspaces via CQL

I don’t actually ever use CQL

So what else is in there?

JSON

Full JSON values, fully indexed, using Jackson and stuff from Lucene

Permissions

Fine grained, per user, built on Shiro

REST

Full REST API, built on Jersey

Anything else?

Mongo Compatible API

Experimental **Mongo** API emulation via **Netty**

Mongo binary wire protocol compatible

That’s what
Usergrid is.

Use the ideas

Use the code

Use the cloud

usergrid.com
git.io/usergrid

Thanks!

apigee