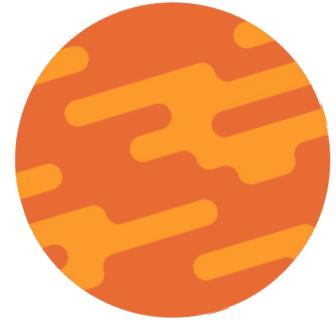


Developers

Intro to Quarkus and Cassandra





S



Cedrick
Lunven



David
Dieruf



Rags
Srinivas



Artem
Chebotko



Stefano
Lottini



Aleksandr
Volochnev



Aaron
Ploetz

S



K

S

S



Jack
Fryer



Kirste
n Hunte
r



Gary
Harvey



Sonia
Siganporia



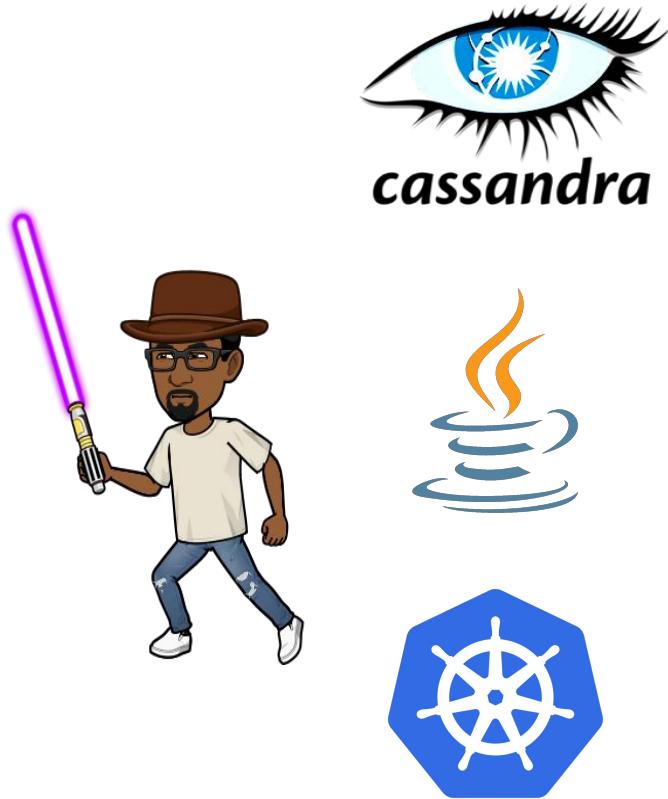
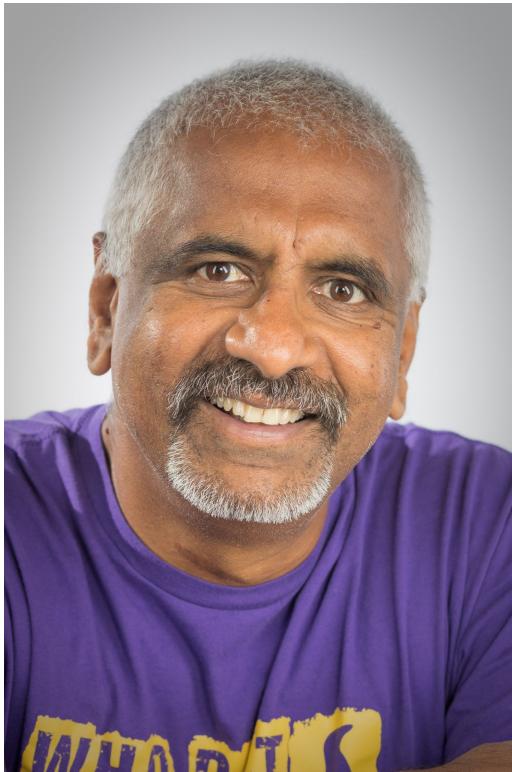
Ryan
Welford



David
Gilardi



Developer Advocate



- Developer/Architect
- Mechanical Engineer (so many moons ago)
- Distributed systems
- Love to teach and communicate
- Inner loop == developer productivity



Raghavan "Rags" Srinivas



@rags

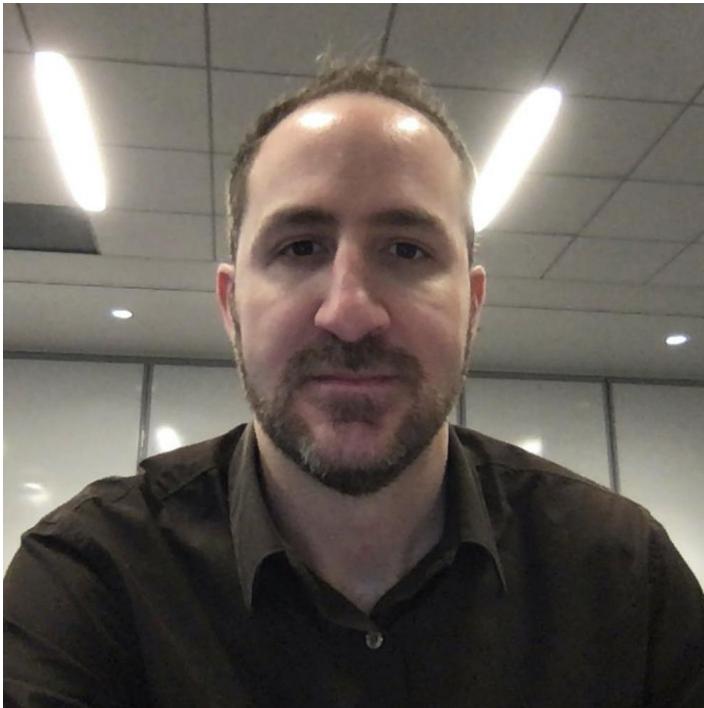


@ragsns



@ragss

Developer Advocate



Red Hat



QUARKUS



- 22+ years Developer/Architect
- 10 years as DevOps Architect
- Contributor to OSS projects
- Love to write, teach, and communicate
- Published author
- Black belt in martial arts



Eric Deandrea



@edeandrea



@edeandrea



@edeandrea

01

Intro.



02

NoSQL

03

03

Cassandra/Astra/
k8ssandra

Quarkus

04

Microservices =
Quarkus+Cassandra

05

Live Coding

06

Resources



Agenda

todos

What needs to be done?

- ✓ Week1, Learn about Apache Cassandra
- ✓ Week2, Apache Cassandra Data Model
- ✓ How to Connect to Cassandra

1 item left All Active Completed Clear completed (2)

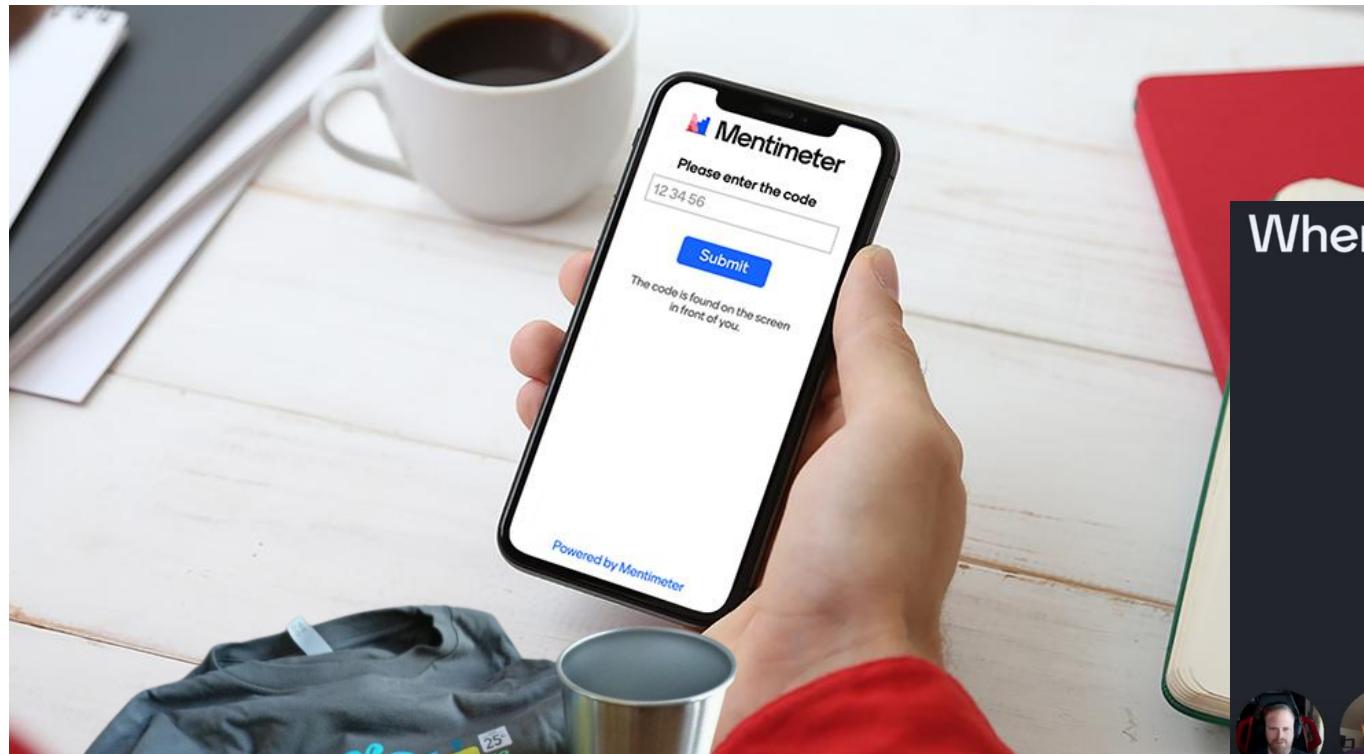
Double-click to edit a todo

Written by [Addy Osmani](#), modified by [Pete Hodgson](#) to be integrated with a [TodoBackend API](#).

Part of [TodoMVC](#) & [TodoBackend](#)



Today's challenge: Todo App



Where are you from?

Mentimeter



menti.com ⇒ enter code
Don't answer in YT chat
Look at phone (not at YT)
Keep it open for later

Menti.com for surveys and quizzes

01

Intro

02

NoSQL



04

Quarkus

05

Microservices = Quarkus
+ Cassandra

03

Cassandra/Astra/
k8ssandra

06

Live Coding

07

Resources

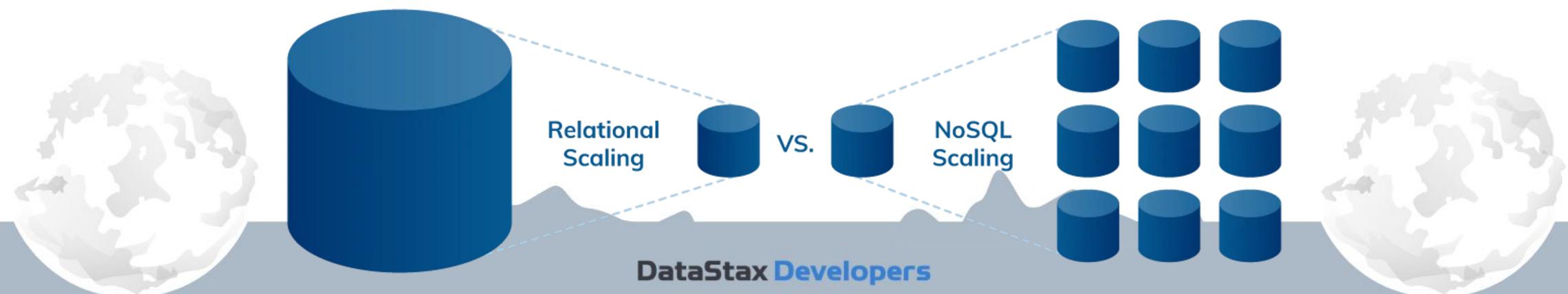
Origin of the term “NoSQL”

- Meetup name on June 11, 2009 in San Francisco
 - Catchy hashtag intended to refer to databases like BigTable and DynamoDB
 - Meetup presentations: Cassandra, MongoDB, CouchDB, HBase, Voldemort, Dynomite, and Hypertable
- Sometimes referred to “Not only SQL”



Relational vs. NoSQL

- Relational
 - Standard relational data model and language SQL
 - ACID transactions
 - Integration database
 - Designed for a single machine
 - Hard to scale
 - Impedance mismatch
- NoSQL
 - Variety of data models and languages
 - Lower-guarantee transactions
 - Application database
 - Designed for a cluster
 - Easy to scale
 - Better database-app compatibility



The CAP Theorem

Always responds,
may not always return
the most recent write

Availability

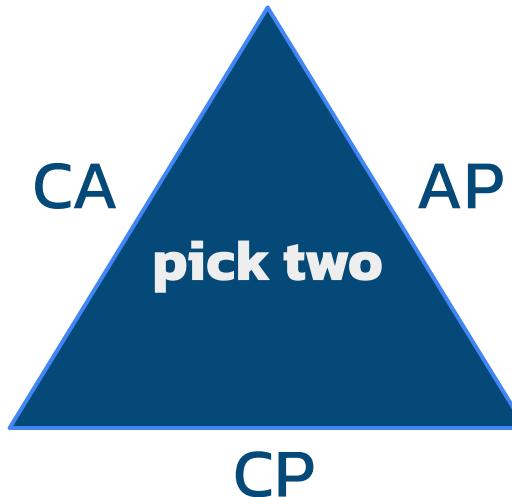
Consistency

Every read receives
the most recent write
or an error

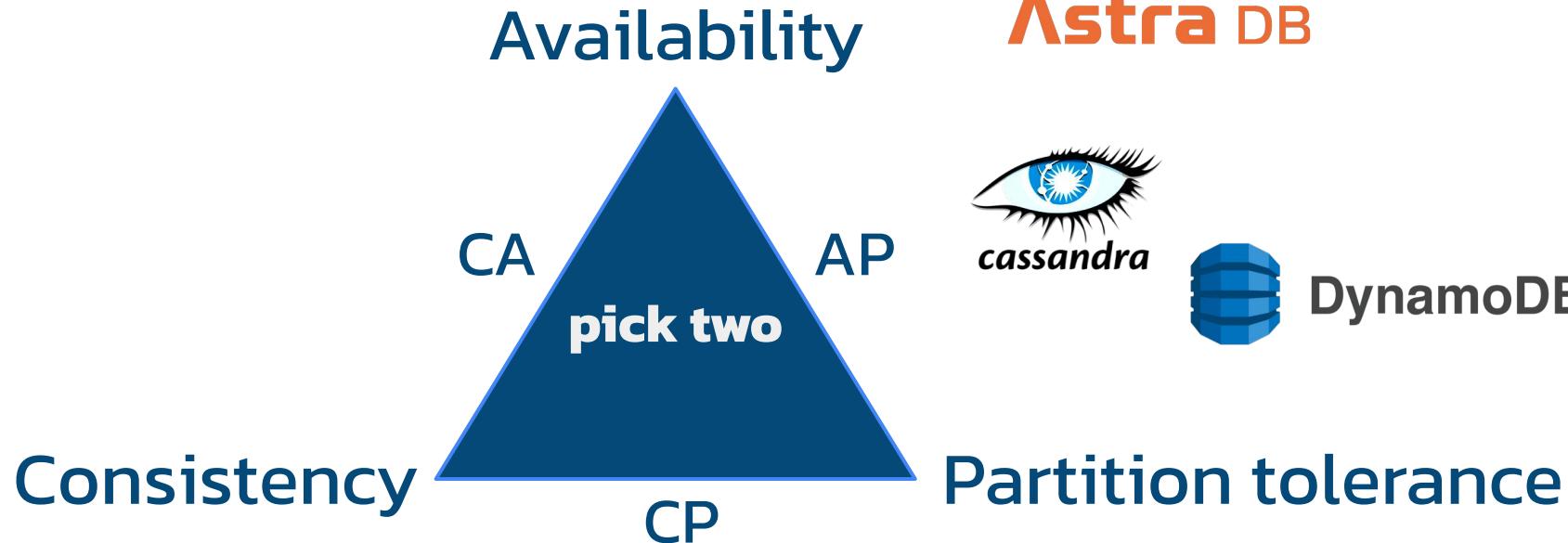


Partition tolerance

Operates in the
presence of network
partition failures



The CAP Theorem



Astra DB



cassandra



DynamoDB



Consistency

Partition tolerance



redis



BigTable



DataStax Developers

01

Intro

02

NoSQL

03

Cassandra/Astra/
k8ssandra



04

Quarkus

05

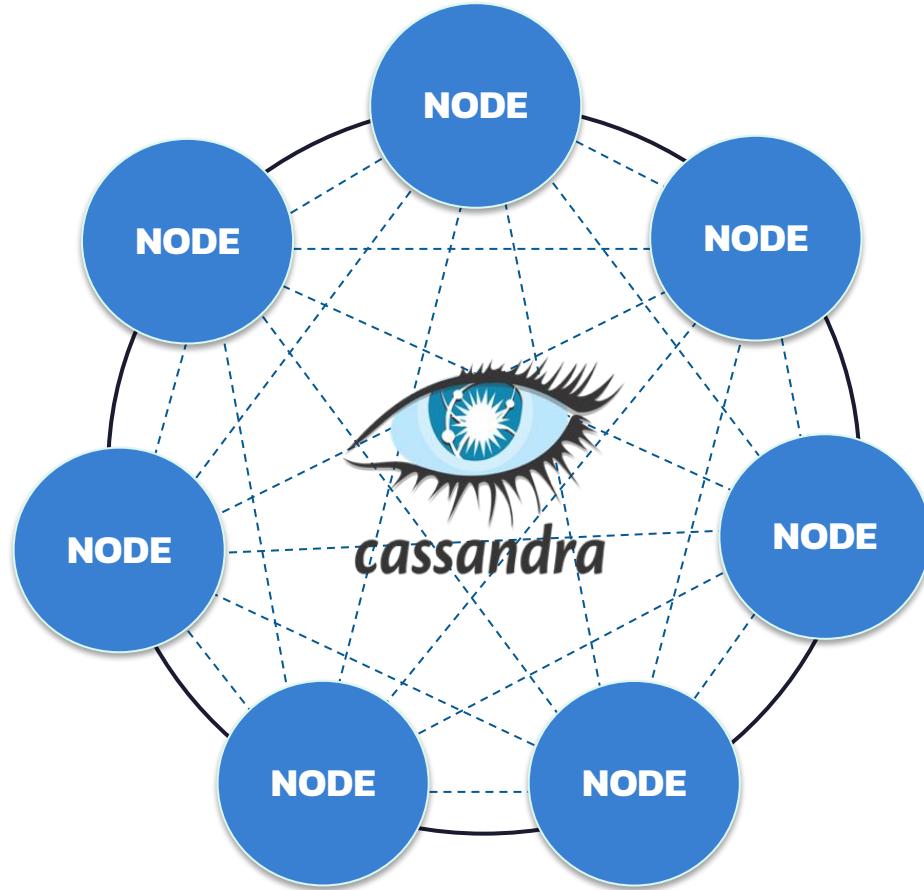
Microservices (Quarkus +
Cassandra)

06

Live Coding

07

Resources



1. **NO** Single Point of Failure
2. Scales for writes and reads
3. Application can contact any node
(in case of failure - just contact next one)

Master-less (Peer-to-Peer) Architecture

Why partitioning? Because scaling doesn't have to be [s]hard!

Big Data doesn't fit to a single server, splitting it into chunks we can easily spread them over dozens, hundreds or even thousands of servers, adding more if needed.



Cassandra is configurably consistent. In any moment of the time, for any particular query you can set the Consistency Level you require to have. It defines how many **CONFIRMATIONS** you'll wait before the response is dispatched;

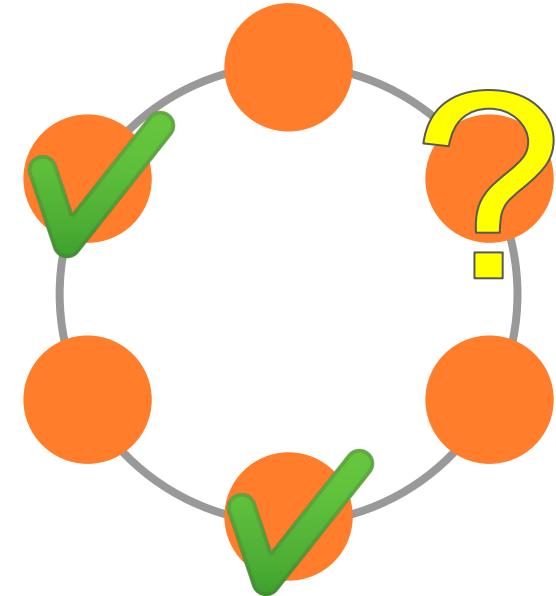
```
PreparedStatement pstmt = session.prepare(  
    "INSERT INTO product (sku, description) VALUES (?,  
?)")  
);  
pstmt.setConsistencyLevel(ConsistencyLevel.ONE);
```

```
cqlsh> CONSISTENCY
```

```
Current consistency level is QUORUM.
```

```
cqlsh> CONSISTENCY ALL
```

```
Consistency level set to ALL.
```



Is Cassandra AP or CP?

Apache Cassandra @ Netflix

- . 98% of streaming data is stored in Apache Cassandra
- . Data ranges from customer details to viewing history to billing and payments
- . Foundational datastore for serving millions of operations per second

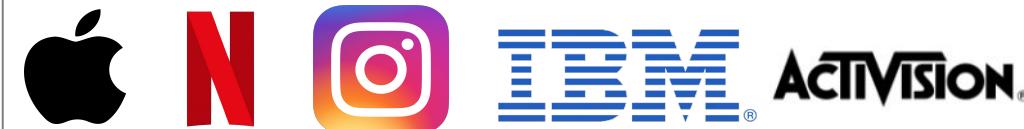
- 30 million ops/sec on most active single cluster
- 500 TB most dense single cluster
- 9216 CPUs in biggest cluster

O(100) Clusters
O(10000) Instances
O(10,000,000) Replications per second
O(100,000,000) Operations per second
O(1,000,000,000,000,000) Petabytes of data

dtsx.io/cassandra-at-netflix

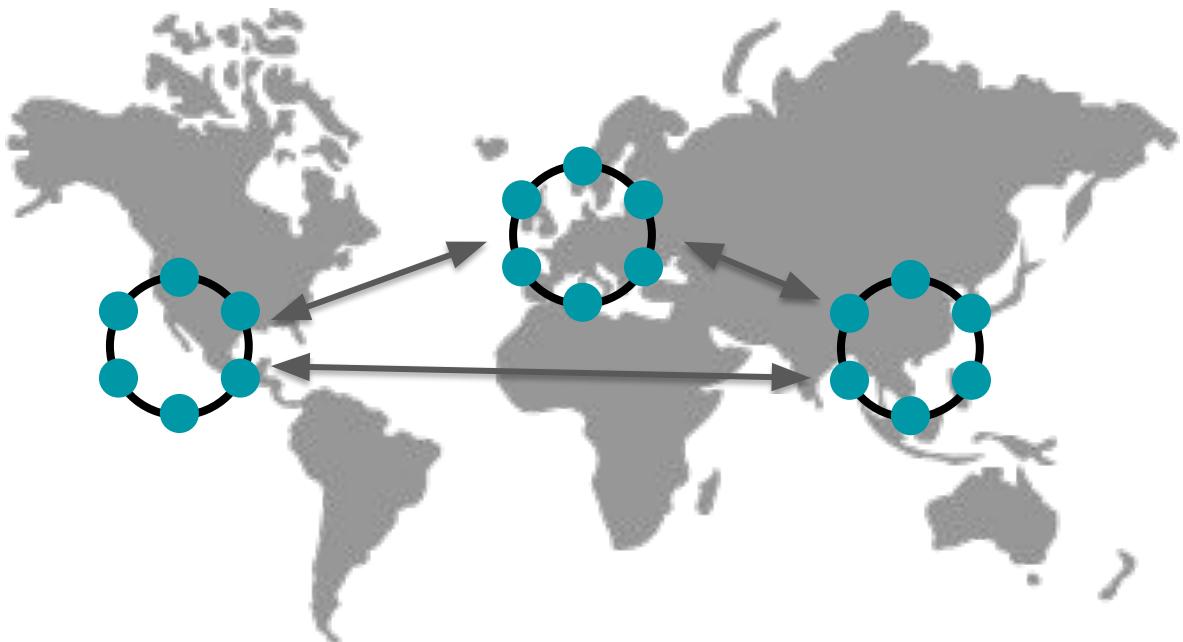
Apple Scale

- 160K+ Apache Cassandra instances
- 100+ PB stored
- Several million ops / sec
- 1000s of clusters

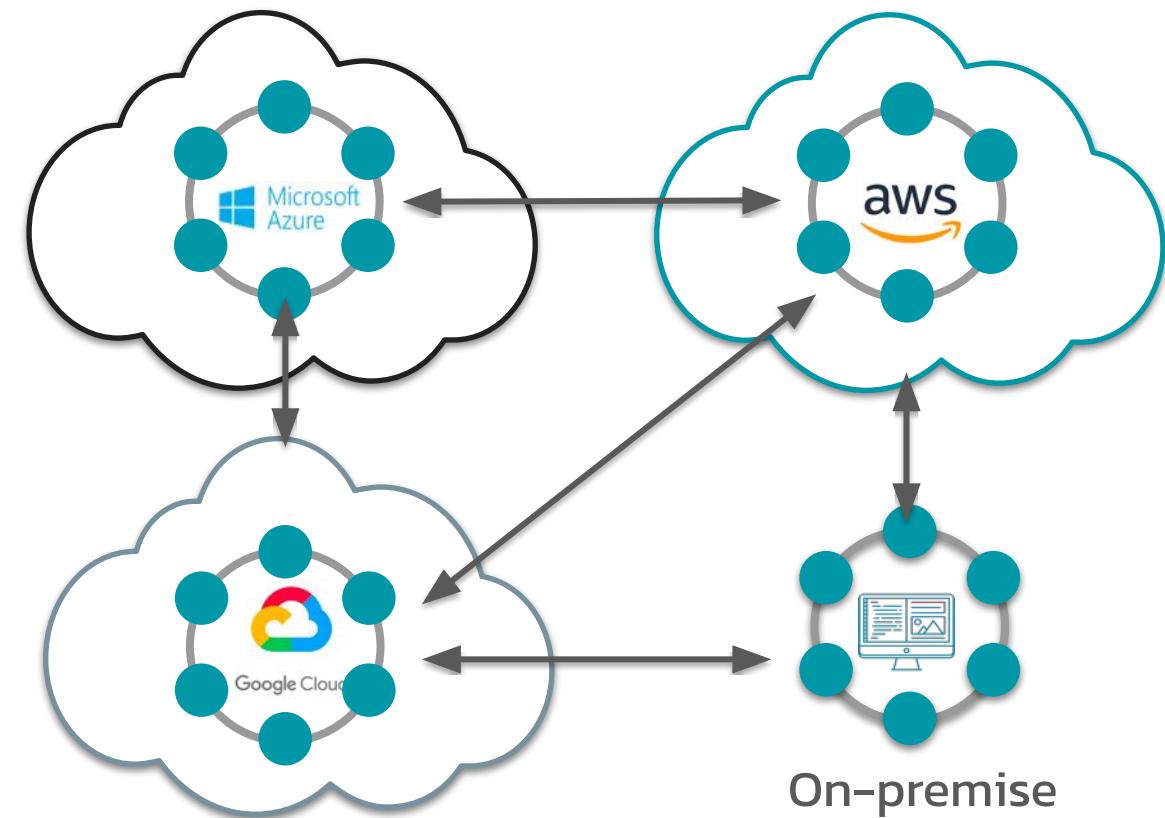


Cassandra Biggest Users (and Developers)

Geographical Distribution



Hybrid-Cloud and Multi-Cloud



Data is globally distributed

State-of-the-art NoSQL Database

Astra DB

- DBaaS, serverless, auto-scalable
- Multi-cloud, distributed, multi-node cluster
- NoSQL, multi-model
- Tabular, document, key-value
- Based on open-source *Apache cassandra*

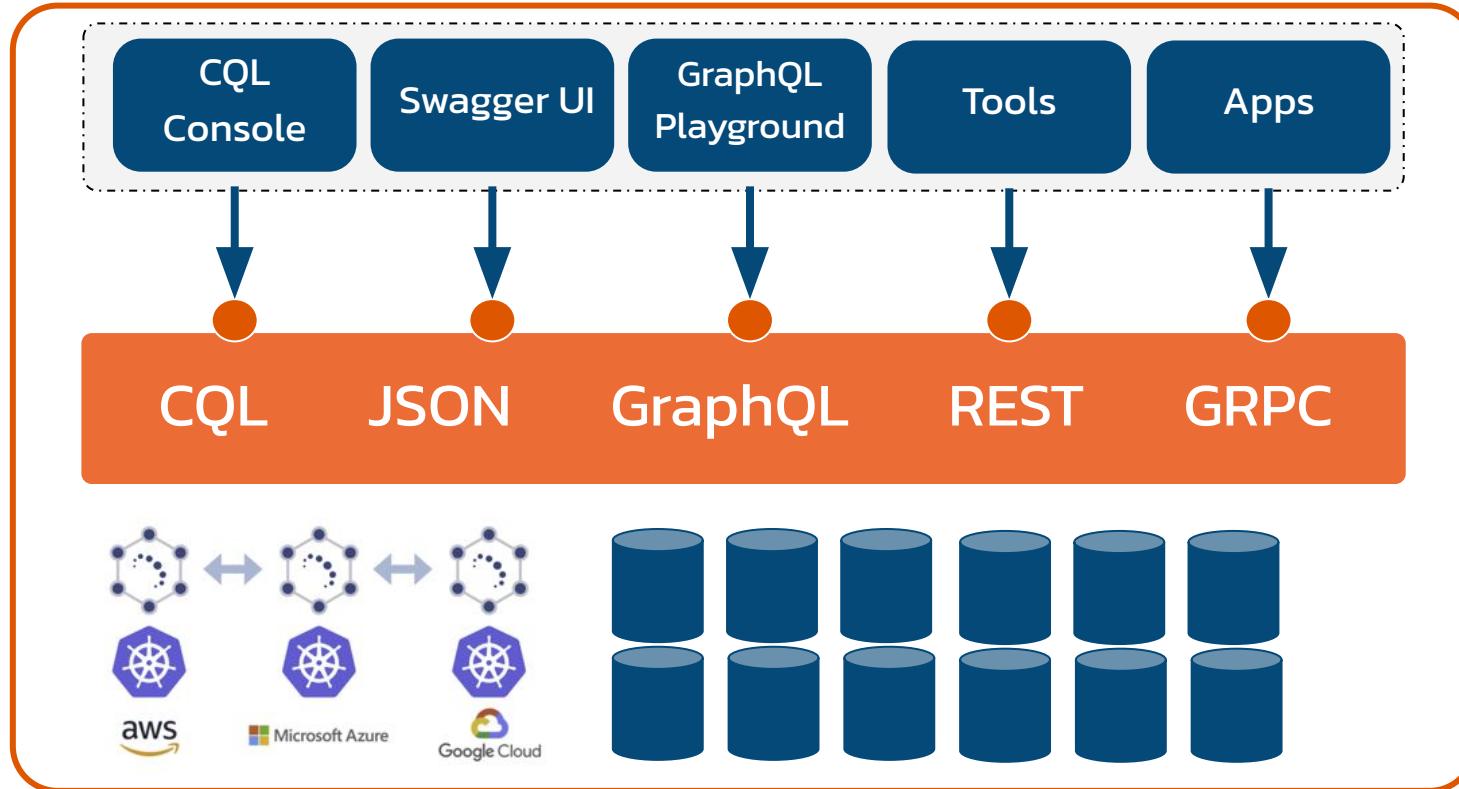


DataStax

Astra DB

\$25/month credit

Launch a database in the cloud with a few clicks, no credit card required.



User Interface
Web-based developer
tools and apps



OSS Stargate.io
A data gateway to allow
multiple usages



OSS Apache Cassandra
A tabular NoSQL database



DataStax Developers





+



=

**K8SSANDRA**

01

Intro.

02

NoSQL

03

Cassandra/Astra/
k8ssandra

04



Quarkus

05

Microservices = Quarkus
+ Cassandra

06

Live Coding

07

Resources

Inner loop == Developer Productivity

“Our developers used to wait **2 to 3 mins** to see their changes. **Live coding** does away with this!”

A cohesive platform for Microservices joy

- Based on standards
- Unified configuration
- Developer productivity
 - Zero-config Live coding
 - Developer services
 - Continuous testing
 - Dev UI
 - CLI
- Live coding
- Streamlined code for the 80% common usages
- No hassle native executable generation

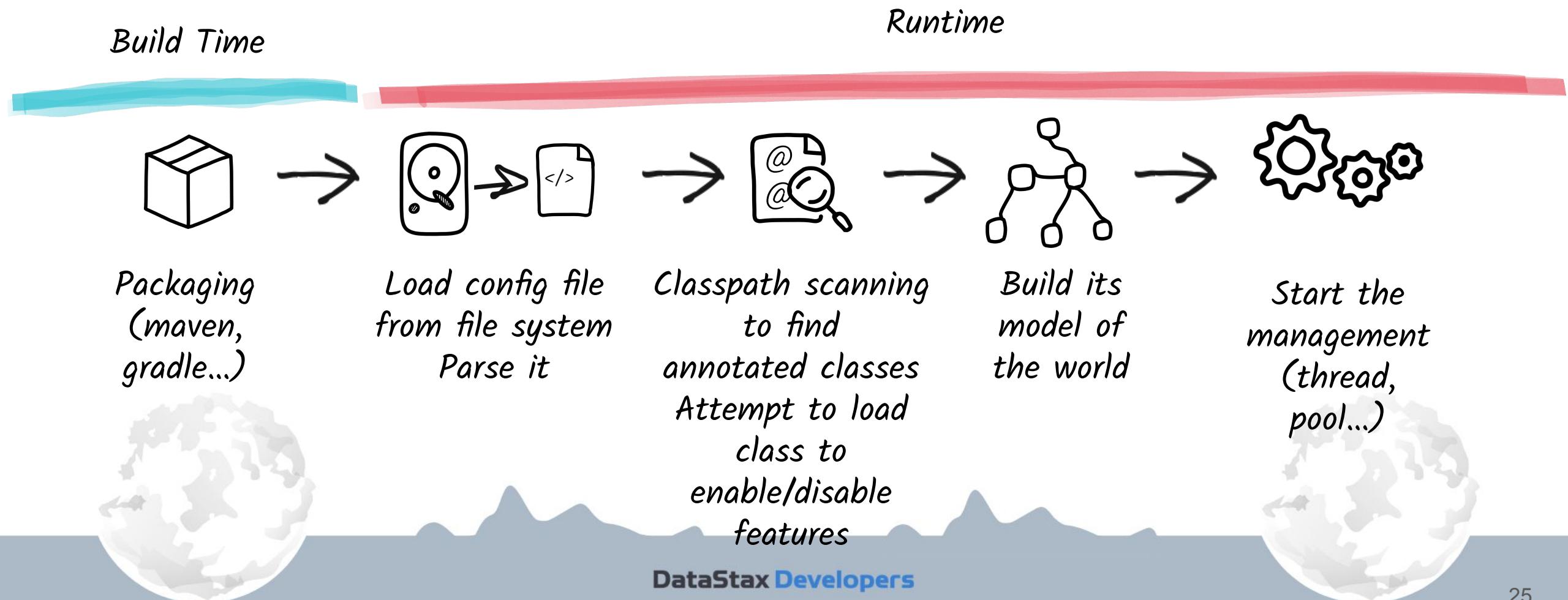




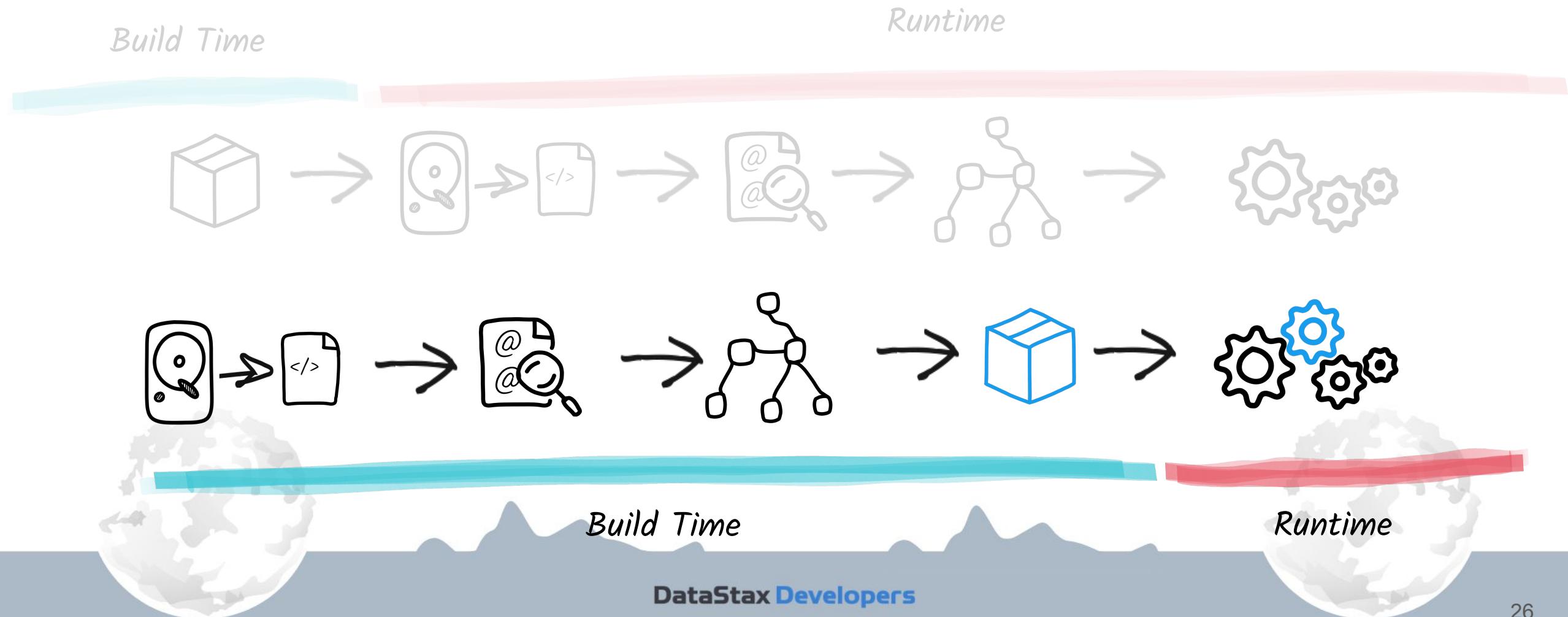
IT'S STILL JAVA!



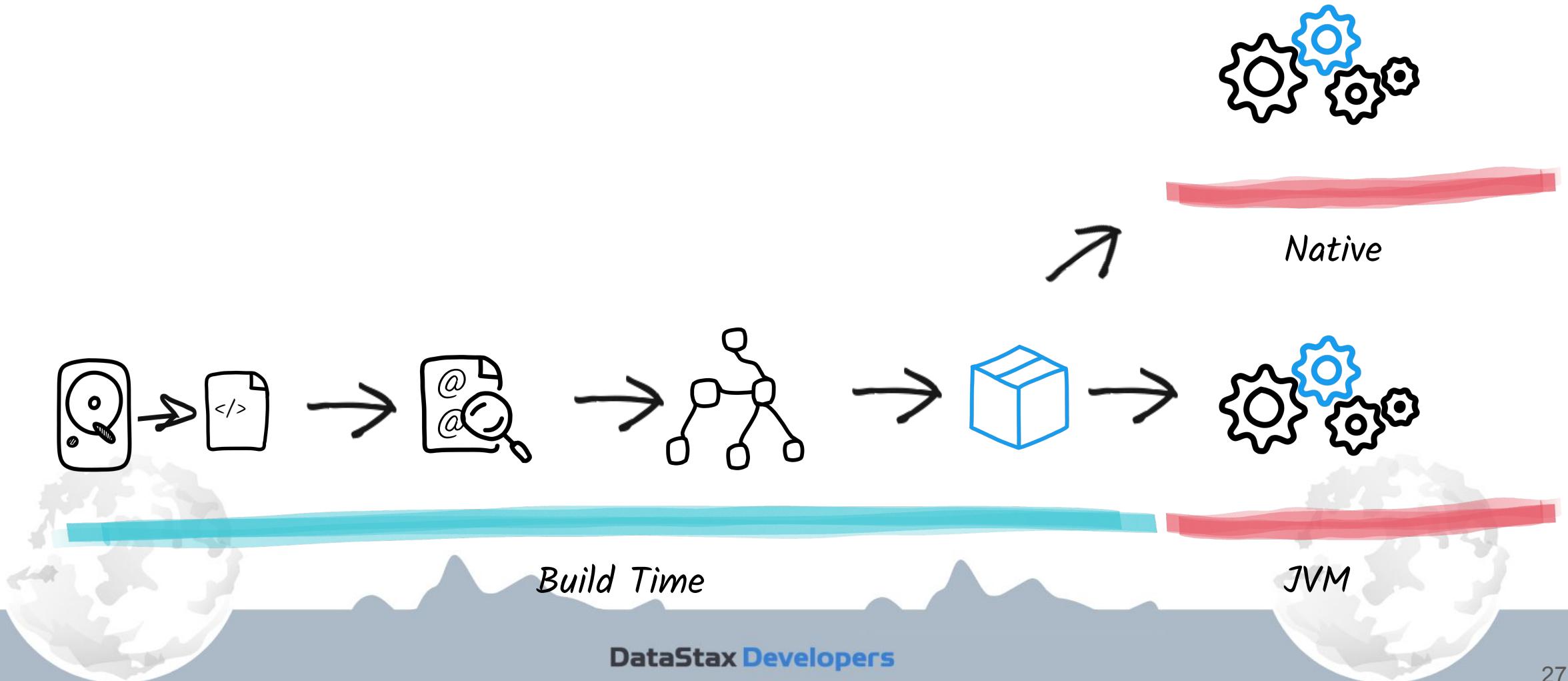
How does a Typical Java Framework Work?



The Quarkus Way



The Quarkus Way Enables Native Compilation



01

Intro

02

NoSQL

03

Cassandra/Astra
k8ssandra

04

Quarkus

05



Microservices = Quarkus
+ Cassandra

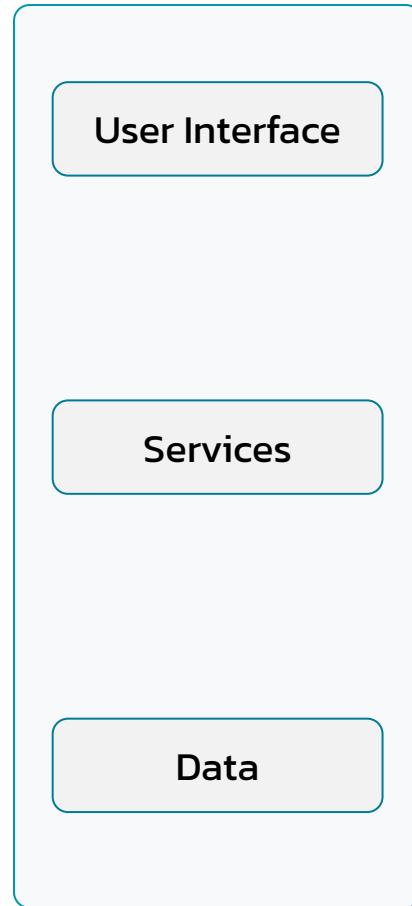
06

Live Coding

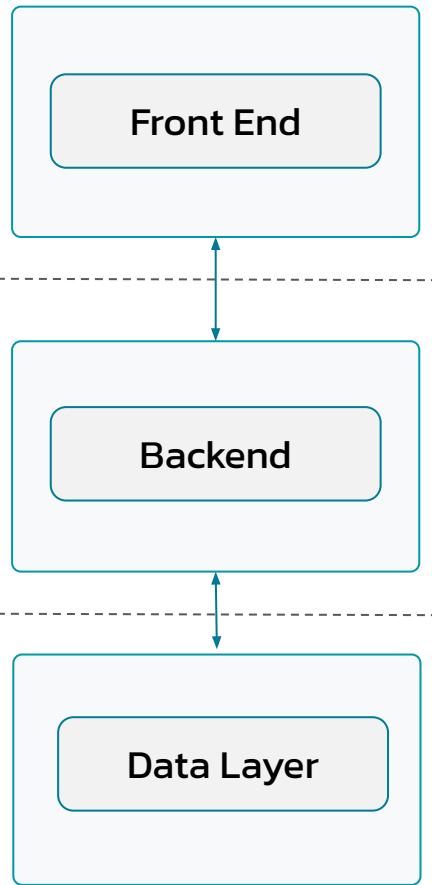
07

Resources

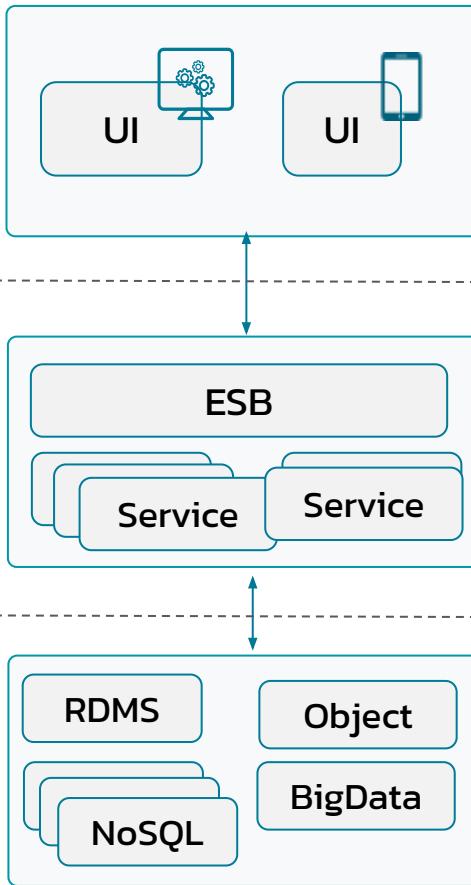
Monolith 90s



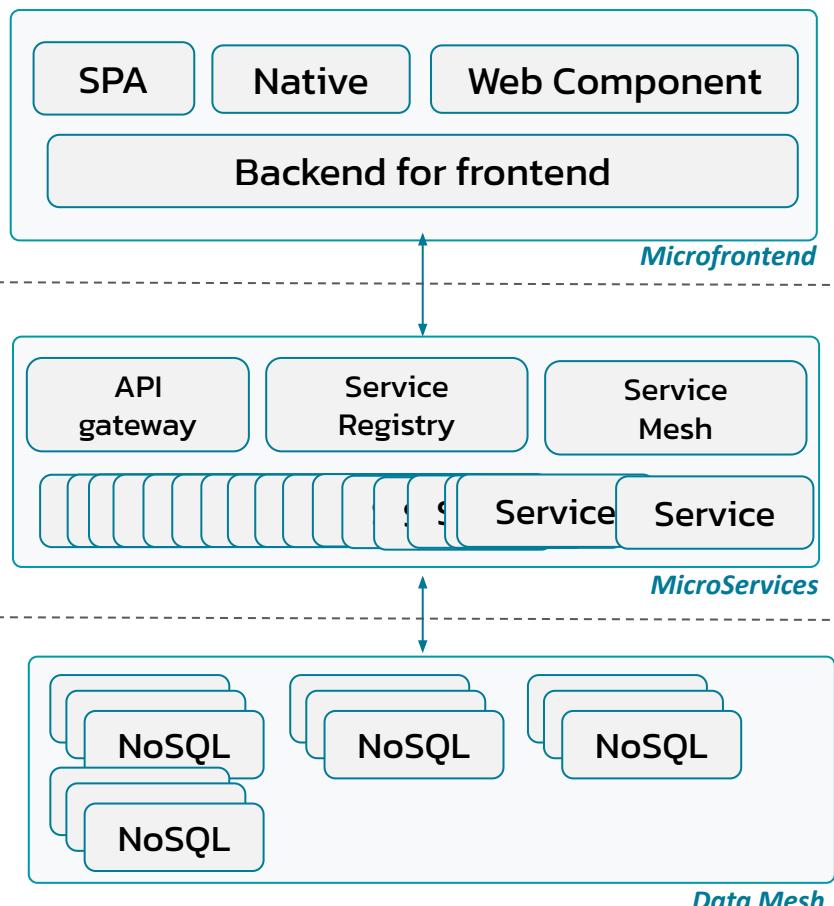
Multi Tiers 2000



SOA (2005)



Microservices (2015)



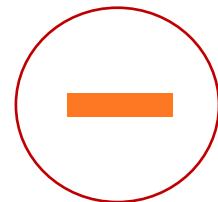
Microservices Architecture evolution

ADVANTAGES



- Reduce Cost (Scaling, Design)
- Reduce Risk (resilience)
- Increase Release Speed
- Enable Visibility (security, monitoring)

DISADVANTAGES



- Complexity (Security, Transaction, Orchestration)
- Cultural Changes
- Bigger RUN footprint



Microservices

Quarkus Cassandra Extension

- Native Quarkus Config
- Cassandra Driver Session Support
- Cassandra Driver Object Mapper Support
- Support for Mutiny Types (Reactive Types)
- Native Image Support
- Support for DataStax Astra (Cassandra DBaaS)

Native Quarkus Config

```
quarkus.cassandra.cloud.secure-connect-bundle=/path/to/astra/bundle.zip  
quarkus.cassandra.keyspace=ks1
```

```
quarkus.cassandra.auth.username=alice  
quarkus.cassandra.auth.password=s3cr3t
```

```
quarkus.cassandra.request.timeout=5 seconds  
quarkus.cassandra.request.consistency-level=LOCAL_ONE  
quarkus.cassandra.request.page-size=1000
```

```
quarkus.cassandra.metrics.enabled=true  
quarkus.cassandra.health.enabled=true
```

Cassandra Driver Session Support

```
@Inject QuarkusCqlSession session;
```



DataStax Developers



Cassandra Driver Object Mapper Support

```
@Dao interface ProductDao {  
    @Insert Uni<Void> create(Product product);  
    @Select Uni<Product> findById(String id);  
    @Select Multi<Product> findAll();  
}  
  
class ProductDaoProducer {  
    @Produces @ApplicationScoped  
    public ProductDao produceProductDao() { ... }  
}  
  
@ApplicationScoped class ProductService {  
    @Inject ProductDao dao;  
}
```

Support for Mutiny Types

```
@GET  
@Produces(MediaType.APPLICATION_JSON)  
@Path("/product/{id}")  
public Uni<Response> findProduct(@PathParam("id") String id) {  
    return dao.findById(id)  
        .map(todo -> Response.ok(todo).build())  
        .ifNoItem().after(Duration.ofSeconds(5))  
        .recoverWithItem(Response.status(Status.NOT_FOUND).build());  
}
```

01

Intro

02

NoSQL

03

Cassandra/Astra/
k8ssandra

04

Quarkus

05



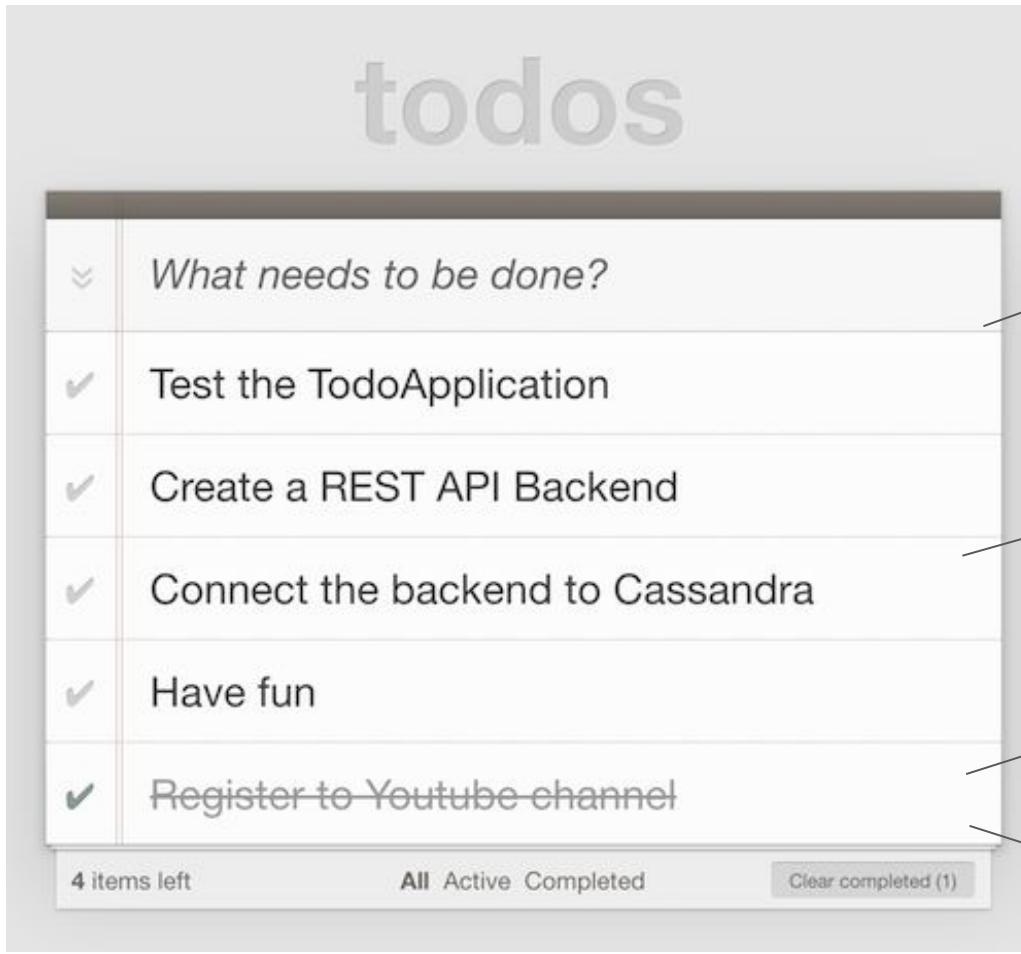
Microservices = Quarkus
+ Cassandra

06

Live Coding

07

Resources



List all tasks

Create a new Task

Mark a task as completed

Delete a task

Specification of Service layer

todoitems

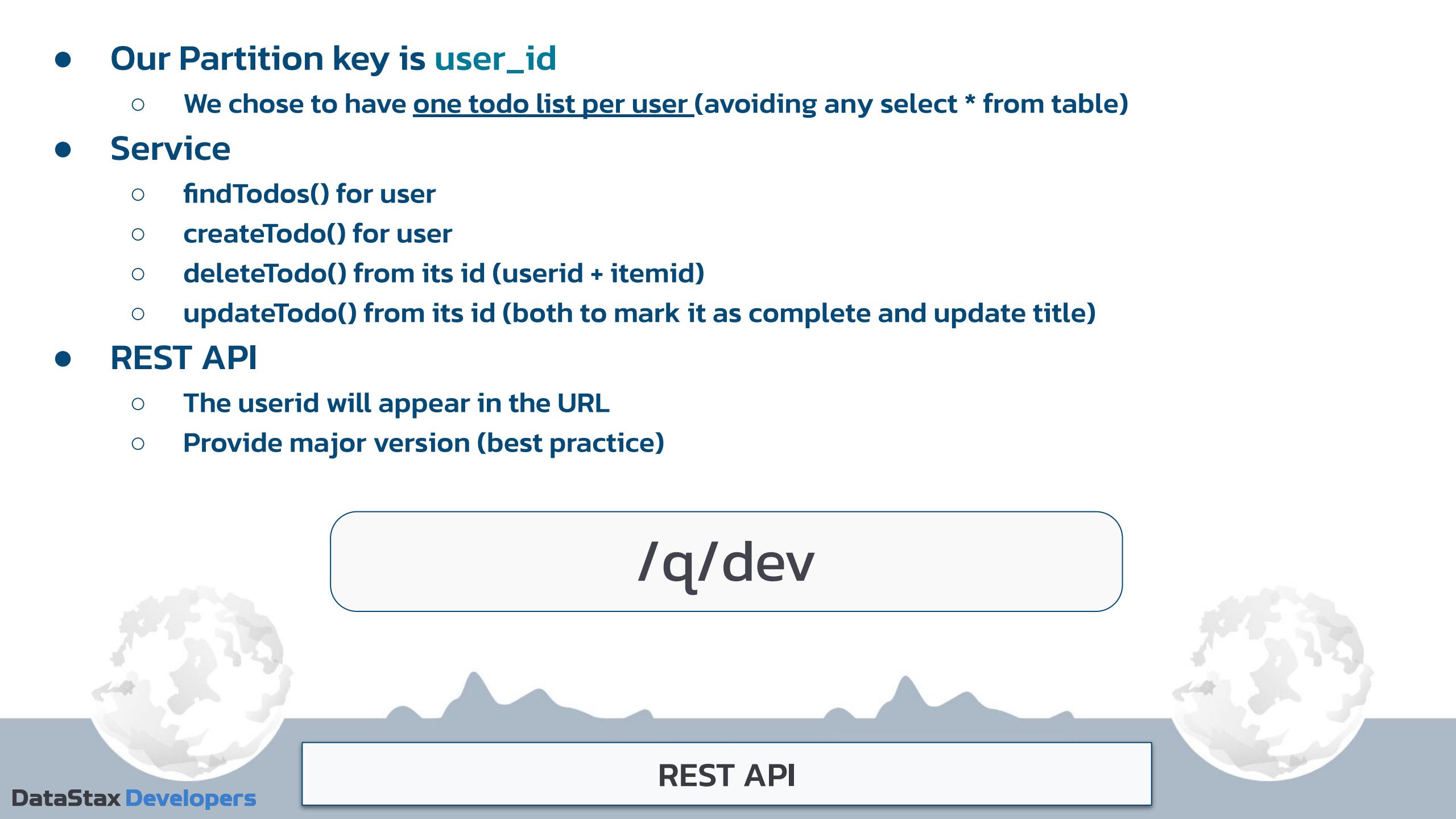
user_id	TEXT	K
item_id	TIMEUUID	C↑
completed	BOOLEAN	
title	TEXT	
offset	INT	

```
CREATE TABLE todos.todoitems (
    user_id      text,
    item_id      timeuuid,
    completed    boolean,
    title        text,
    PRIMARY KEY ((user_id),item_id)
);
```



Data Model

- Our Partition key is `user_id`
 - We chose to have one todo list per user (avoiding any select * from table)
- Service
 - `findTodos()` for user
 - `createTodo()` for user
 - `deleteTodo()` from its id (`userid + itemid`)
 - `updateTodo()` from its id (both to mark it as complete and update title)
- REST API
 - The userid will appear in the URL
 - Provide major version (best practice)



/q/dev

REST API



Helping you **select** an MV* framework

Download

[View on GitHub](#)

[Blog](#)



<http://todomvc.com/examples/angularjs/>

The screenshot shows a todo list with the following items:

- Explain the use case
- Create the Data model
- Define the queries to perform
- Create the DDL
- Connect to Cassandra
- Create the CRUD repository
- Run the API

At the bottom, there are filters: "All" (selected), "Active", and "Completed".



[TodoMVC.com](http://todomvc.com)



Todo-Backend

a shared example to showcase backend tech stacks

The Todo-Backend project defines a simple web API spec - for managing a todo list. Contributors [implement](#) that spec using various tech stacks. Those implementations are cataloged below. A [spec runner](#) verifies that each contribution implements the exact same API, by running an automated test suite which [defines](#) the API.

The Todo-Backend project was inspired by the [TodoMVC](#) project, and some code (specifically the todo client app) was borrowed directly from TodoMVC.

Created and curated by [Pete Hodgson](#).

featuring HTTP APIs built with:



aiohttp



Akka



API Platform



Axon Framework



Azure Functions



CakePHP



Catalyst



Ceylon



Clojure



CoffeeScript



Compojure



CouchDB



Crystal



C#



django



.NET



Dropwizard



Elixir



ES6



express



express.js



Finatra

Finch

Swagger UI /q/openapi Explore quarkus-astra-intro-demo 0.01 (powered by Quarkus 2.3.1.Final)

quarkus-astra-intro-demo API 0.01 OAS3

/q/openapi

Astra TODO

- GET** /api/todo/{list_id}
- POST** /api/todo/{list_id}
- POST** /api/todo/{list_id}/{id}
- DELETE** /api/todo/{list_id}/{id}

Astra Demo CQL

- GET** /hello

Schemas

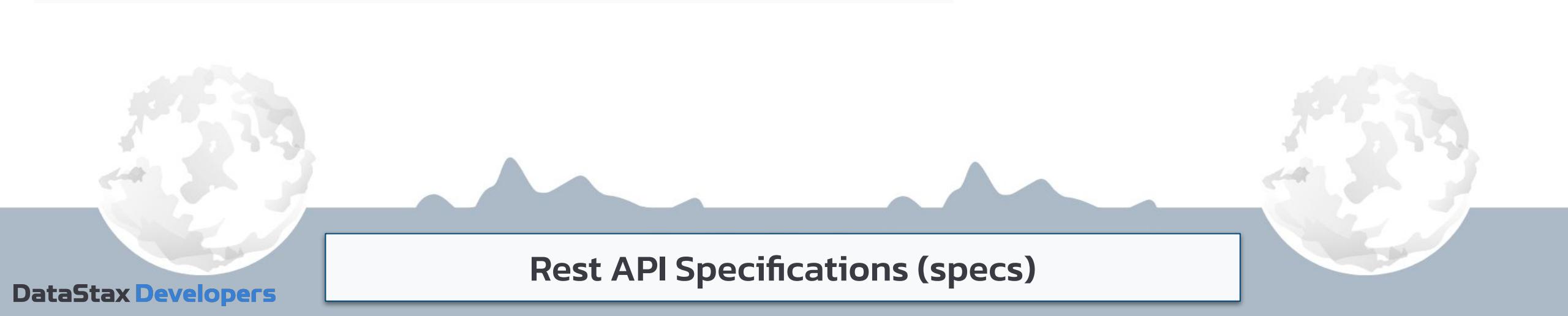
```
Todo {  
    id      string  
    title   string  
    completed boolean  
}
```

todos

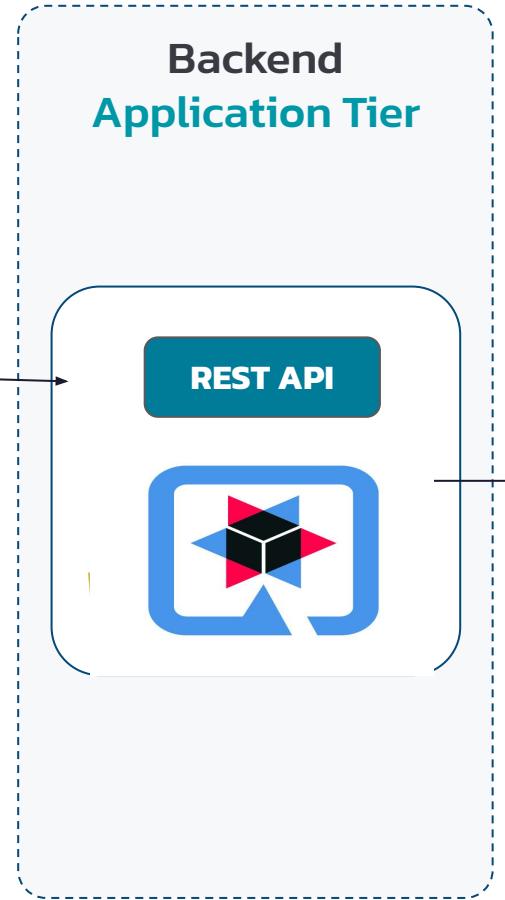
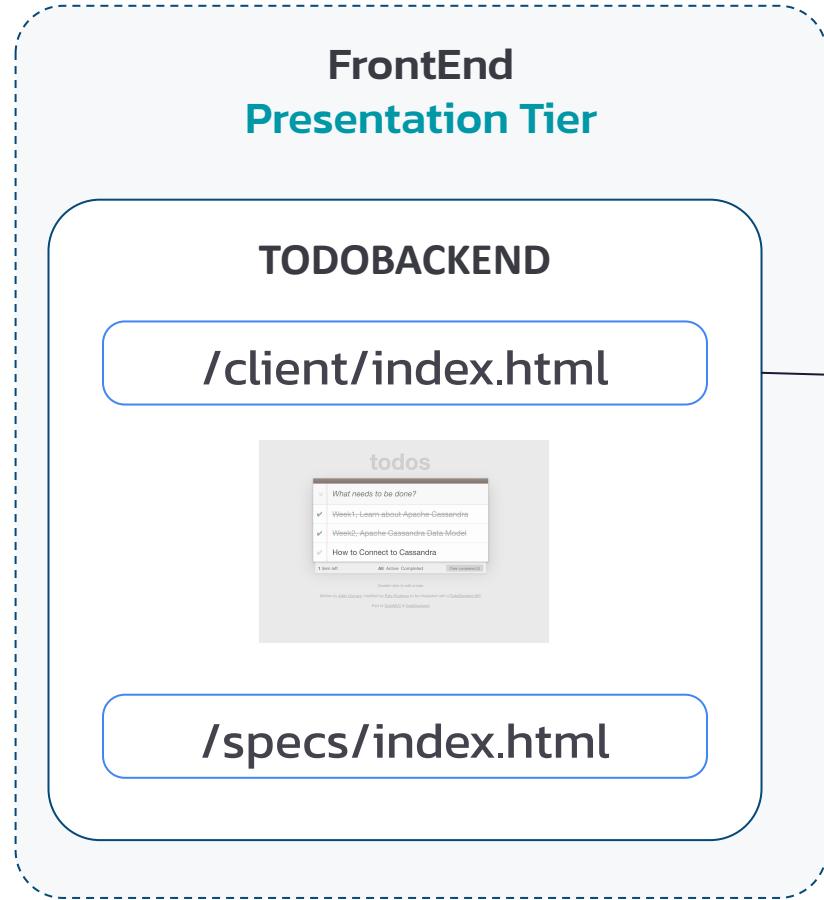
What needs to be done?

- ✓ Test the TodoApplication
- ✓ Create a REST API Backend
- ✓ Connect the backend to Cassandra
- ✓ Have fun
- ✓ Register to Youtube channel

4 items left All Active Completed Clear completed (1)



Rest API Specifications (specs)



Architecture

01

Intro

02

NoSQL

03

Cassandra/Astra/
k8ssandra

04

Quarkus

05



Microservices = Quarkus
+ Cassandra

06

Live Coding

07

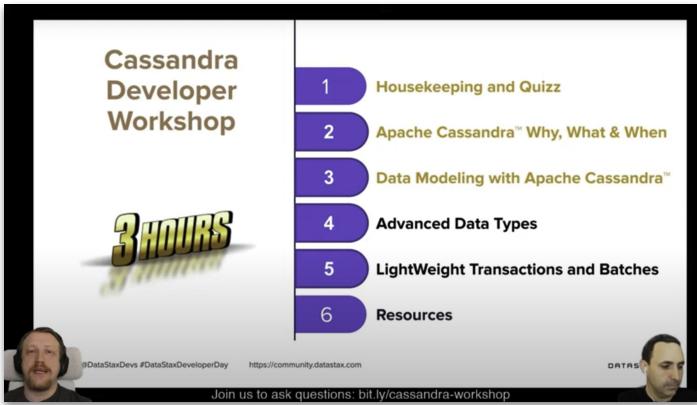
Resources



Agenda

Live and interactive

Livestream: youtube.com/DataStaxDevs



YouTube



Twitch

Questions: <https://dtsx.io/discord>



Discord

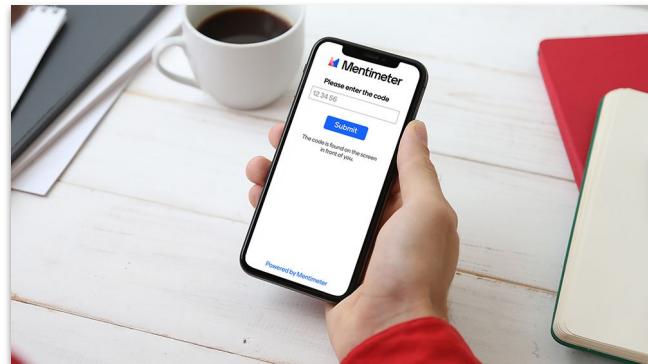


YouTube



Available on the iPhone App Store

Games menti.com



Mentimeter



GET IT ON
Google play

1

Attend the sessions

Database + GraphQL + PlayGround



DataStax
Astra DB

The screenshot shows the Gitpod IDE interface. On the left is an Explorer sidebar with project files like `WORKSHOP-SPRING-STARGATE`, `github`, `vscode`, `dataset`, `images`, `stargate-demo`, `settings`, `main`, `resources`, `test`, `.project`, `Dockerfile`, `pom.xml`, `gitpod.Dockerfile`, `gitpod.yml`, `LICENSE`, `README.md`, and `slides.pdf`. The main area displays Java code for `StargateDemoApplication.java`:

```
StargateDemoApplication.java
stargate-demo > src > main > java > com > datasetx > demo > stargate > StargateDemoApplication.java > (1) cor
package com.datasetx.demo.stargate;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
public class StargateDemoApplication {
    public static void main(String[] args) {
        SpringApplication.run(StargateDemoApplication.class, args);
    }
}
```

Below the code editor are tabs for PROBLEMS, OUTPUT, TERMINAL, and DEBUG CONSOLE. The TERMINAL tab shows the command `gitpod /workspace/workshop-spring-stargate $`. To the right of the code editor are logos for NPM, Node.js, Maven, and Tomcat.

Gitpod

O'REILLY
Katacoda
KATACODA OVERVIEW & SOLUTIONS

Connect to Astra (Cassandra as a Service) with CQL Shell

Step 1 of 5

Create your Astra DB Database

If you don't have an Astra account, set one up - it's easy.

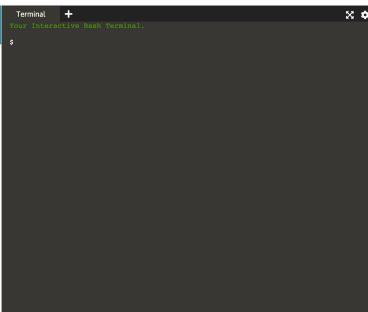
Go to the Astra DB page in your browser astra.datastax.com.

Let's create the database. Follow the steps outlined here. To make life easy, we have recommended the values you should use for this scenario.

NOTE: If you already have an Astra DB database with values that differ from what we suggested, you may have to adapt some of the operations in this scenario accordingly, or create an additional keyspace with the designated values.

If you don't already have an Astra DB database, when you log in the first time you'll see a screen that looks like the image below.

Choose Plan & Provider



The screenshot shows a GitHub repository page for `DataStax-Examples/todo-astra-jamstack-netlify`. The repository has 2 stars, 17 forks, and 20 open issues. The code tab is selected, showing a commit history with the following commits:

- kidrecursive readme updates (bfff2d4, 16 days ago)
- functions rfa todos (10 months ago)
- public rfa todos (3 months ago)
- src rfa todos (3 months ago)
- env-template Update for IAM changes (4 months ago)
- ignore update for CRA (3 months ago)
- gitpod.yml update for CRA (3 months ago)

The repository also includes sections for `astra.datastax.com/register`, `rest`, `cassandra`, `rest-api`, `dev`, `datastax`, `astra`, `building-sample-apps`, and `Readme`.

GitHub

2

Complete Workshops Labs

datastax.com/workshops



ASTRA DB'S BUILD-A-THON

MAKING SIDE-HUSTLES A REALITY

21 February - 28 May 2022



JOIN OUR ASTRA DB BUILD-A-THON HACK!

📍 3 months, 3 rounds of challenges. 📍
Join 1 month, 2 months or all 3

Each month, we'll reveal a fresh new set of challenges you can partake in.

All you have to do is have Astra DB as your backend.

USD\$41,000 worth of prizes



REGISTER -

buildathonhack.com

- Showcase & explain Quarkus, how it enables modern Java development & the Kubernetes-native experience
- Introduce familiar Spring concepts, constructs, & conventions and how they map to Quarkus
- Equivalent code examples between Quarkus and Spring as well as emphasis on testing patterns & practices
- Chapters devoted to
 - Why the need for Quarkus in the first place?
 - Getting started
 - RESTful applications
 - Persistence
 - Event-driven services
 - Cloud environments, containers, and Kubernetes



<https://red.ht/quarkus-spring-devs>

- **Create a Quarkus Serverless Project**
 - Run Serverless Functions Locally
 - Test Serverless Functions Continuously
 - Deploy Functions to a Knative Service on OpenShift
- **Make Serverless Functions Run Faster With GraalVM**
- **Make Portable Functions Across Serverless Platforms**
 - Deploy a Quarkus Funqy Application to AWS Lambda
- **Bind CloudEvents on Knative With Quarkus Serverless Functions**



<https://dzone.com/refcardz/getting-started-with-quarkus-serverless-functions>

Try Cassandra+Quarkus

- Create your quarkus + cassandra app (code.quarkus.io or running the following):

```
$ mvn io.quarkus:quarkus-maven-plugin:2.7.5.Final:create \  
-DprojectGroupId=io.quarkus.astra \  
-DprojectArtifactId=quarkus-astra-demo \  
-DprojectVersion=1.0.0 \  
-DclassName="io.quarkus.astra" \  
-Dextensions="resteasy-reactive, resteasy-reactive-jackson, micrometer-registry-prometheus, smallrye-openapi, smallrye-health, cassandra-quarkus-client"  
  
$ cd quarkus-astra-demo  
$ ./mvnw clean quarkus:dev
```

- Stand up your Astra free database (astradb.datastax.com)
- Check out <https://k8ssandra.io>

```
quarkus.cassandra.cloud.secure-connect-bundle=<path>/secure-connect-bundle.zip  
quarkus.cassandra.auth.username=<user>  
quarkus.cassandra.auth.password=<pw>
```

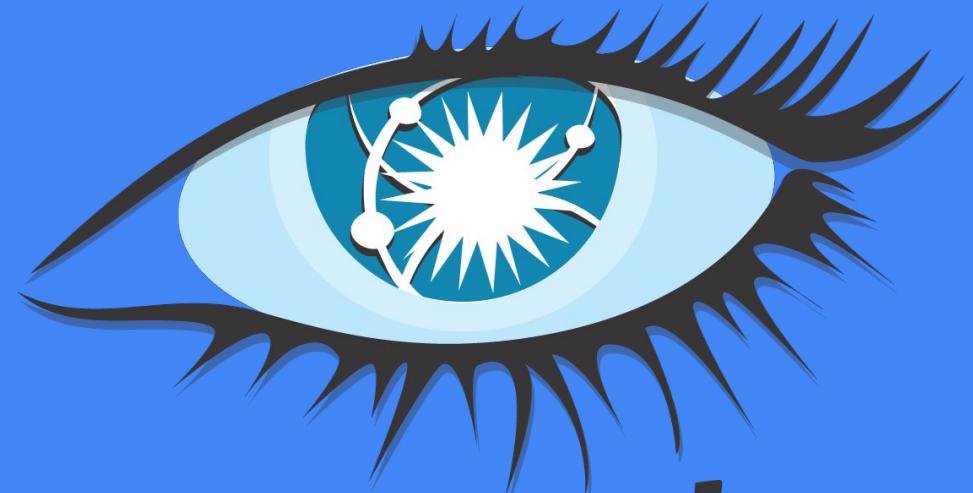
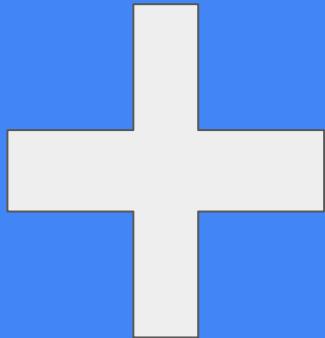
- Get coding + see docs for more info and try the Quarkus + Cassandra workshop

<https://quarkus.io/guides/cassandra>
<https://github.com/datastaxdevs/workshop-intro-quarkus-cassandra/>

DataStax



QUARKUS



Thank You!

Try Astra + Quarkus!