DataStax

DataStax Astra

QUARKUS

Building Microservices with Cassandra & Quarkus

Eric Deandrea, Red Hat
Sr. Principal Technical Evangelist

David Gilardi, DataStax
Developer Advocate

Sebastián Estévez, DataStax
Astra guy

Pieter Humphrey, DataStax
Product Manager

Agenda

01

Quarkus

Why?
How does it work?
Use Cases

02

Astra

Serverless Cassandra on the cloud
Quarkus Cassandra Extension

03

Live Coding

Deploy a Database
Get started with Quarkus
Build
Swag

QUARKUS

Supersonic.
Subatomic.
Java.

https://quarkus.io/community/

Java...
the Enterprise Workhorse

Monolith

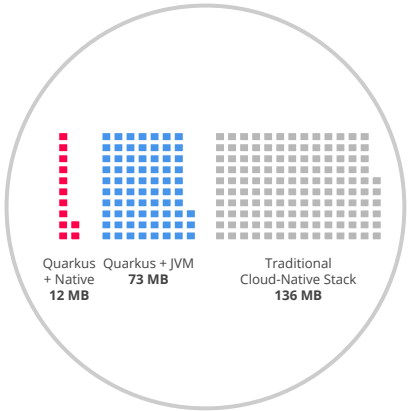
Microservices
Cloud-Native

Cloud-Native
Microservices
Serverless
Event-Driven

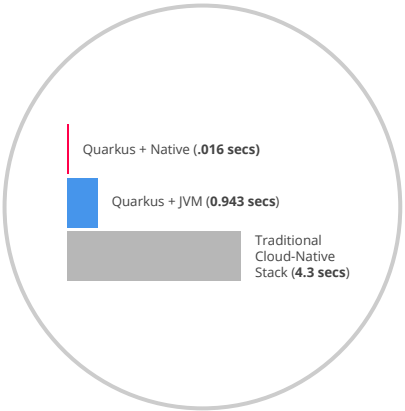
Container First

"We went from **1-min** startup times to **400 milliseconds!**"

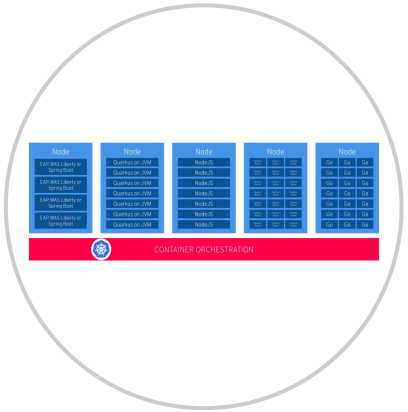
Reduced Memory Footprint



Fast Startup Time



Smaller Disk Footprint

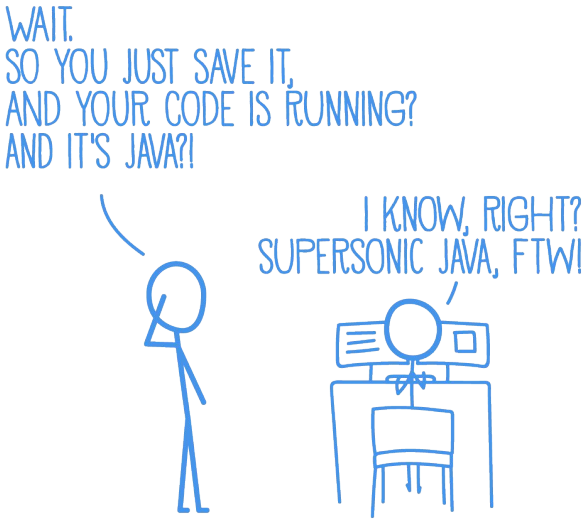


Developer Joy

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

A cohesive platform for optimized developer joy

- Based on standards
- Unified configuration
- Live coding
- Streamlined code for the 80% common usages
  - Flexible for the 20% uncommon
- No hassle native executable generation



Unifies Imperative & Reactive

```
@Inject
SayService say;

@GET
@Produces(MediaType.TEXT_PLAIN)
public String hello() {
      return say.hello();
}
```

```
@Inject @Stream(”kafka”)
Publisher<String> reactiveSay;

@GET
@Produces(MediaType.SERVER_SENT_EVENTS)
@SseElementType(MediaType.APPLICATION_JSON)
public Publisher<String> stream() {
      return reactiveSay;
}
```

- Combine both reactive & imperative development within the same application
- Inject the Vert.x **EventBus** or **Context**
- Use the technology that fits your use case
- Key for reactive systems based on event driven applications

IT'S STILL JAVA!

How does a Typical Java Framework Work?

Build Time

Runtime

Packaging (maven, gradle...)

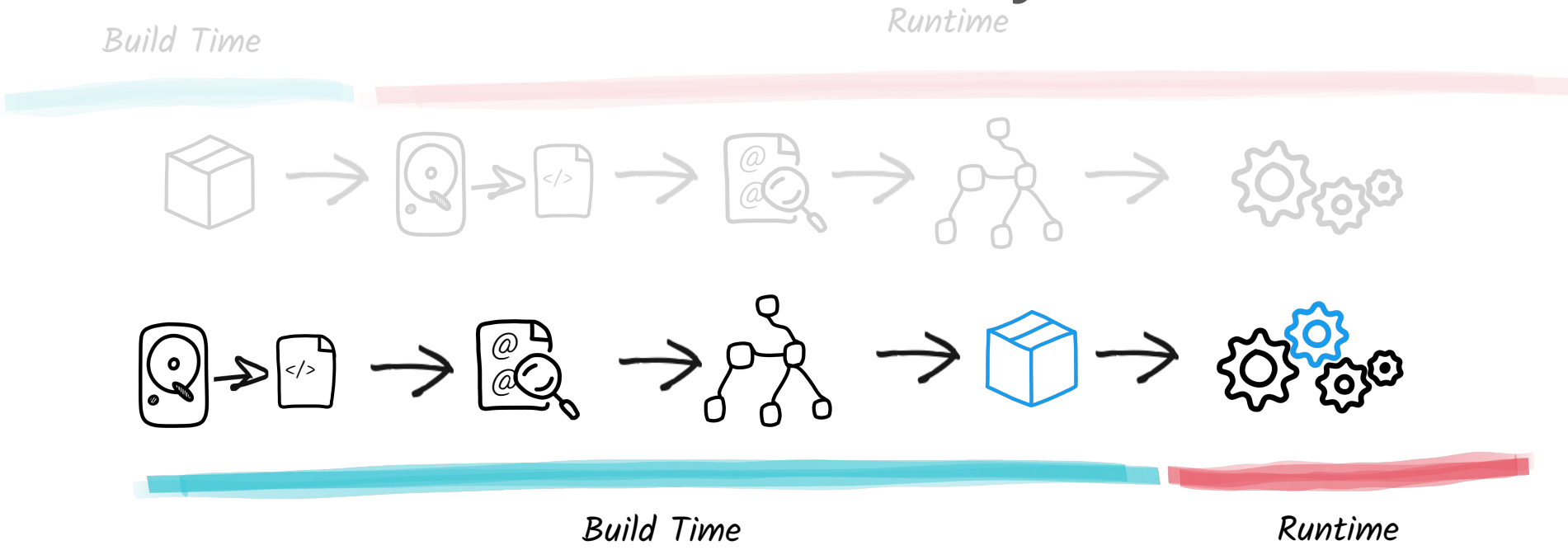
Load config file from file system
Parse it

Classpath scanning to find annotated classes
Attempt to load class to enable/disable features

Build its model of the world

Start the management (thread, pool...)

The Quarkus Way



Use Cases

Quarkus is an ideal runtime for...

Net New

Low memory footprint + lightning fast startup time + small disk footprint = an ideal runtime for Kubernetes-native applications

Mono 2 Micro

Modernizing existing monolithic applications by breaking into smaller, loosely-coupled services

Serverless

Scaling up or down (including to 0!) is extremely fast, making Quarkus an ideal runtime for serverless applications

Event-Driven / Reactive

Quarkus utilizes an asynchronous, reactive event loop core that makes it easy to create reactive applications

IoT / Edge

Take advantage of under-utilized edge resources, shifting business logic closer to the data emitter

DataStax

Serverless Cassandra on the cloud
Cassandra Quarkus Extension

Apache Cassandra™ = NoSQL Distributed Database

- World's Most Scalable Database
- Highest Availability
- Geographical Distribution
- Read/Write Performance
- Vendor Independent

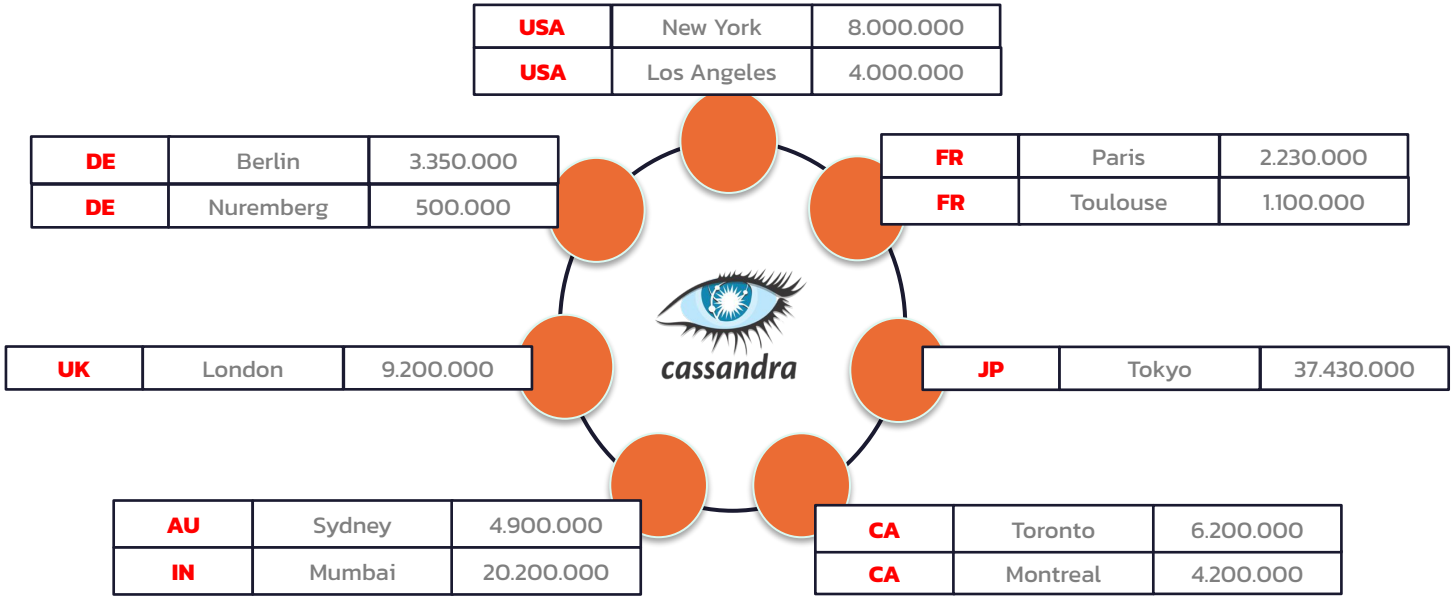
Data is Distributed

Country	City	Population
USA	New York	8.000.000
USA	Los Angeles	4.000.000
FR	Paris	2.230.000
DE	Berlin	3.350.000
UK	London	9.200.000
AU	Sydney	4.900.000
DE	Nuremberg	500.000
CA	Toronto	6.200.000
CA	Montreal	4.200.000
FR	Toulouse	1.100.000
JP	Tokyo	37.430.000
IN	Mumbai	20.200.000

Partition Key

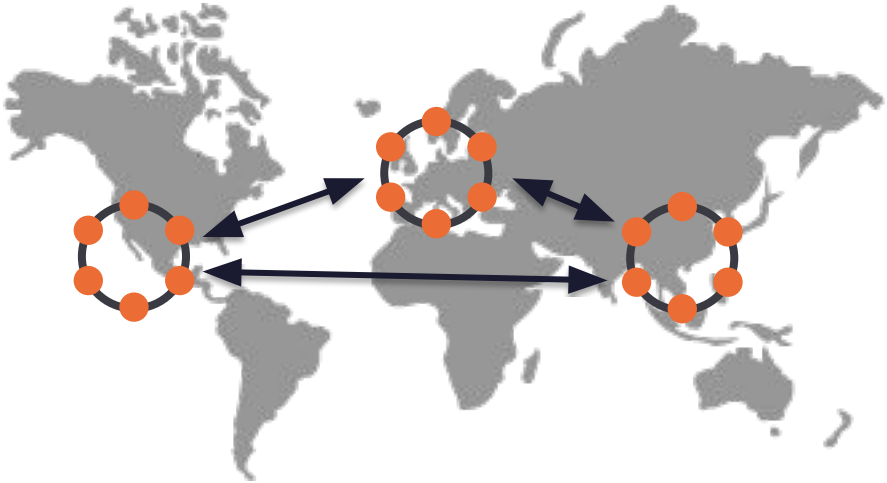
Data is Distributed

Country	City	Population

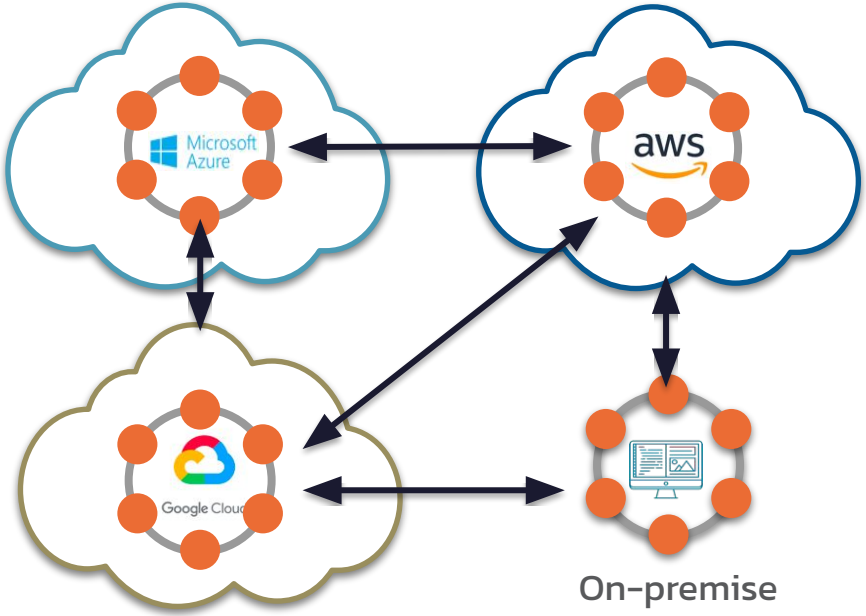


Data Everywhere

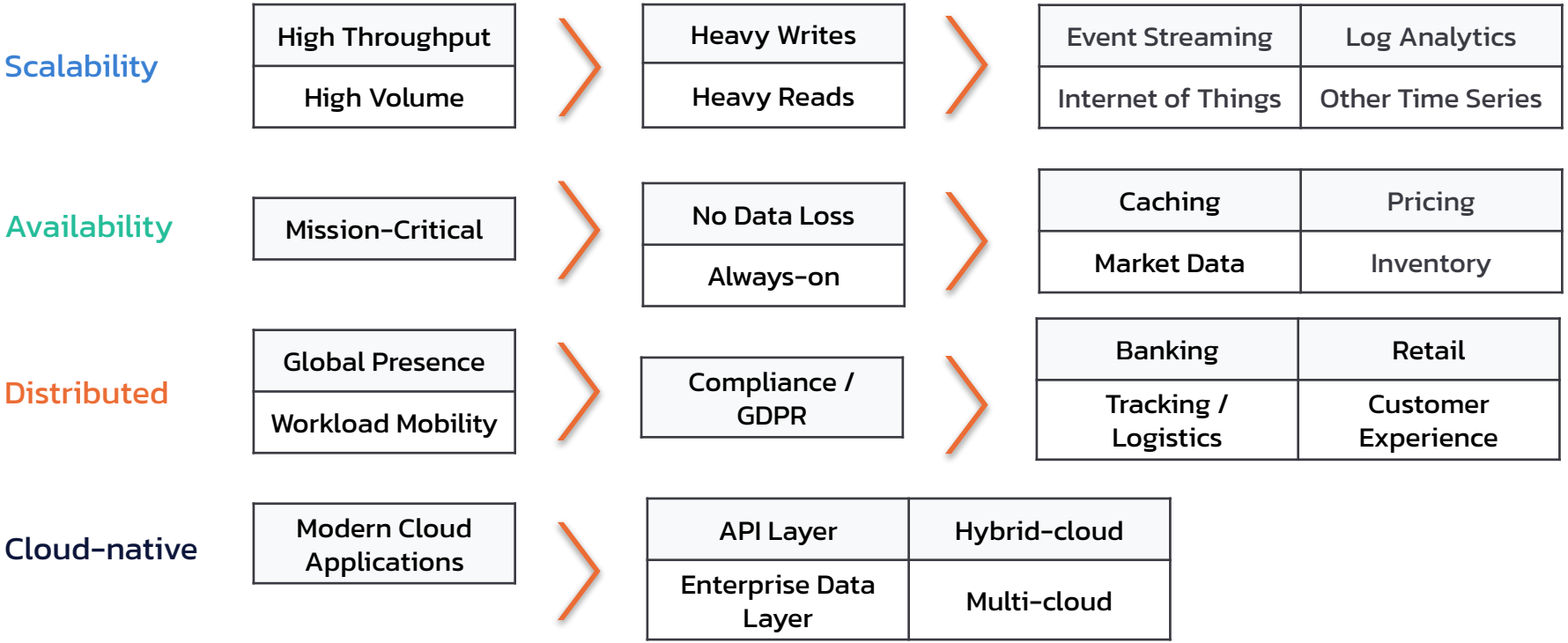
- Geographic Distribution



- Hybrid-Cloud and Multi-Cloud



When do you use Cassandra?



What is DataStax Astra: Cassandra Made Easy in the Cloud

Cassandra-as-a-Service

Cloud-native
Database-as-a-Service built
on Apache Cassandra

Serverless

Eliminate the overhead
to install, operate, and
scale Cassandra

Powerful APIs

Out-of-the-box REST, Doc
and GraphQL endpoints
and browser CQL shell

Cloud Native

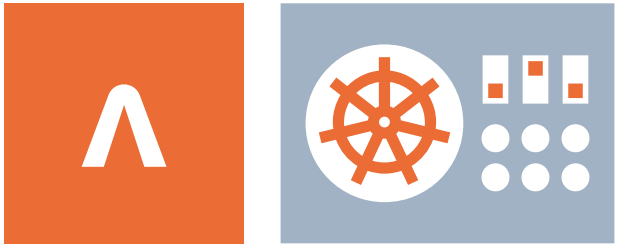
Powered by our
open-source Kubernetes
Operator for Cassandra

Zero Lock-in

Deploy on AWS,Azure or GCP
and
keep compatibility with
open-source Cassandra

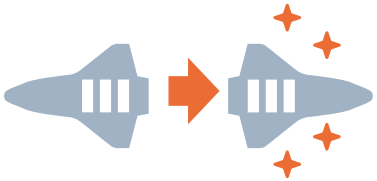
$25 free per month

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



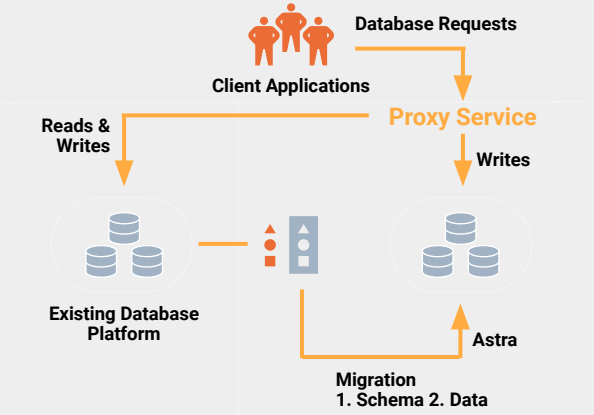
What is Astra

- Serverless, auto-scaling DBaaS
- Always the latest Cassandra OSS
- Auto-generated API endpoints
  - REST, Document API, GraphQL, CQL
- Separated storage and compute
- True cloud economics, i.e. priced based on:
  - Storage
  - Network Data Out
  - Read & Write Operations



Astra Migrations

Zero downtime

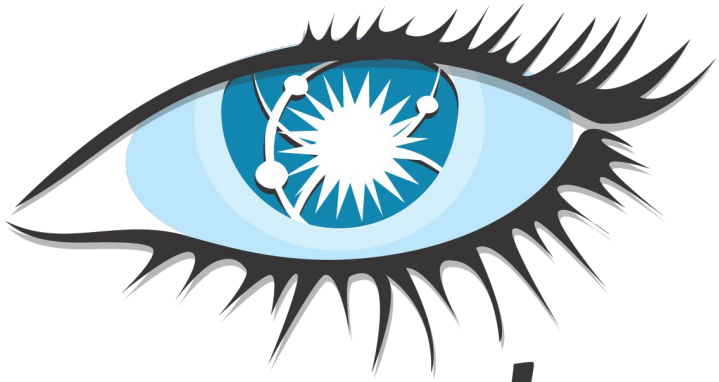


From any modern Cassandra version CloudGate proxy double writes for you. Only change your connection string, the proxy handles the hard stuff:

- Support for CQL protocol v2 and greater
- Routes reads go to origin cluster
- Routes writes to both clusters

QUARKUS

+



cassandra

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;
```

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());
}
```

Astra Developer Endpoints

REST / JSON Document API

- Built in OpenAPI (swagger) Test Harness
- https://quarkus.io/guides/rest-client

GraphQL

- Built in GraphQL playground

Javascript SDK

Java SDK - coming soon

```
curl -L -X GET
'http://localhost:8082/v2/keyspaces/blog/vehicle?
where=\{"manufacturer":\{"$eq":"Tesla"\}\}' \
-H "X-Cassandra-Token: $AUTH_TOKEN" \
  -H 'Content-Type: application/json'
```

```
curl --location \
--request POST
'localhost:8082/v2/namespaces/blog/collections/ve
hicles' \
--header "X-Cassandra-Token: $AUTH_TOKEN" \
--header 'Content-Type: application/json' \
--data '{
  "vin": "7OS5T5HQLGT073117",
  "model": "Spyder",
  "type": "Sedan",
  "color": "teal",
  "manufacturer":"Tesla"
  }'
```

DataStax

Live Coding

Deploy a Database
Get started with Quarkus
Build
Swag

Try it out

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

```
$ mvn io.quarkus:quarkus-maven-plugin:1.12.1.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
```

- Browse to http://localhost:8080/q/dev
- Stand up your Astra free database (astra.datastax.com)
- Point your app to Astra

```
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

https://quarkus.io/guides/cassandra
https://github.com/phact/quarkus-astra-demos

DataStax

Thank You!

Now go try Astra + Quarkus!