DataStax

Developers

Spring, Quarkus, Micronaut with Apache Cassandra

Director of Developer Relations

clunven

clunven

clun

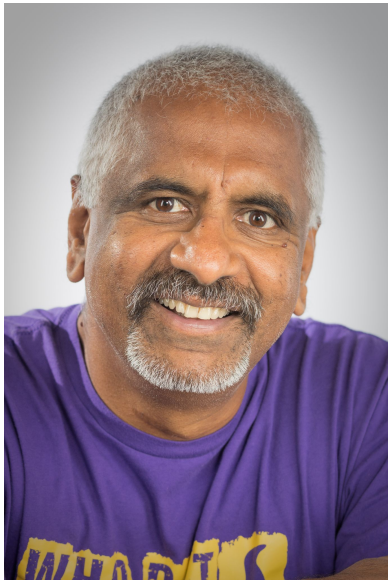
- Trainer
- Public Speaker
- Developers Support
- Developer Applications
- Developer Tooling

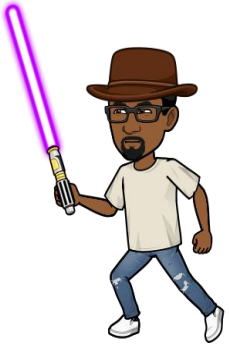
- Creator of ff4j (ff4j.org)
- Maintainer for 8 years+

- Happy developer for 14 years
- Spring Petclinic Reactive & Starters
- Implementing APIs for 8 years

Cédrick Lunven

Developer Advocate





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

@rags

@ragsns

@ragss

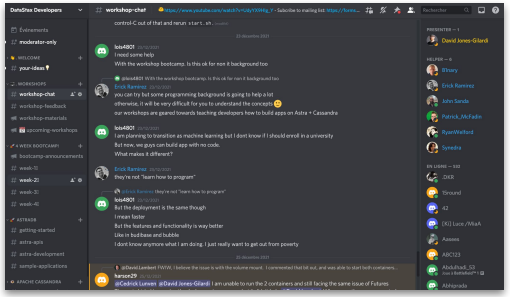
DataStax Developers Crew

Livestream: youtube.com/DataStaxDevs

YouTube

Twitch

Questions: https://dtsx.io/discord

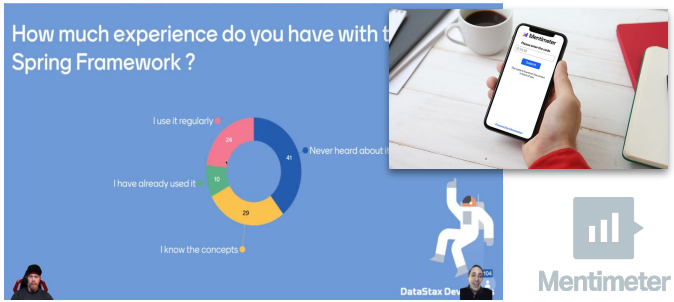


YouTube
(with nighbot)

Discord
(#workshop-chat)

!discord

Games and quizzes: menti.com

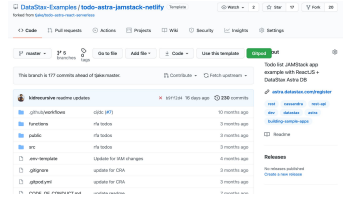


Mentimeter

!menti

Live Sessions

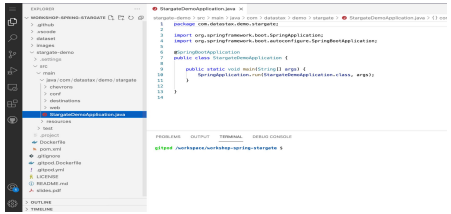
Source code + exercises + slides



GitHub

!github

IDE



Gitpod

!gitpod

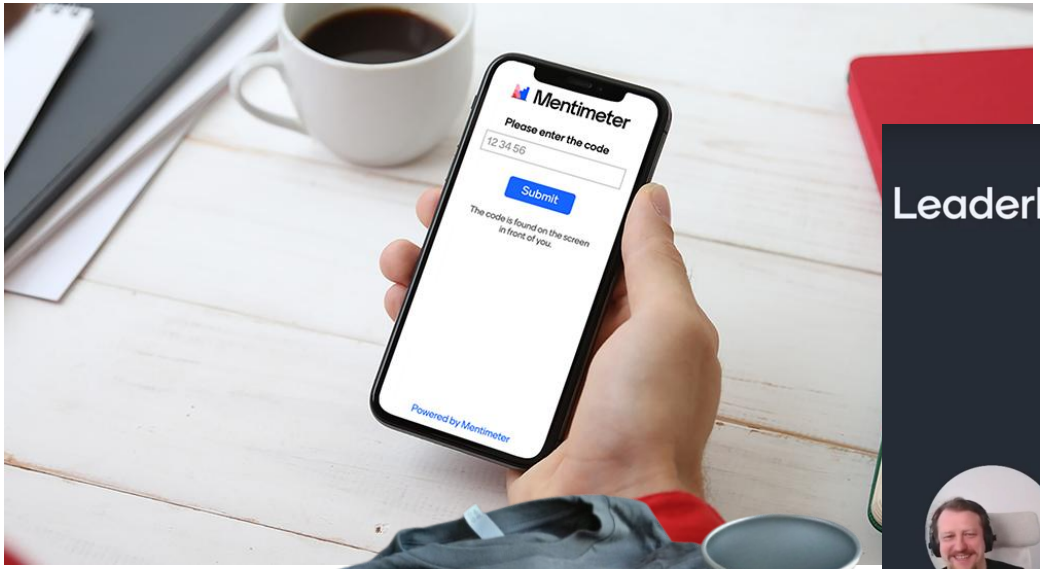
Nothing to install !

Database + Api + Streaming

DataStax Astra DB

!astra

Hands-On Housekeeping



`menti.com` ⇒ enter code
Don't answer in YT chat
Look at phone (not at YT)

Quiz on "Menti" !

01

Intro.

02

Cassandra

03

Java Drivers

04

Java Framework(s) & evolution

05

Spring, Quarkus, Micronaut

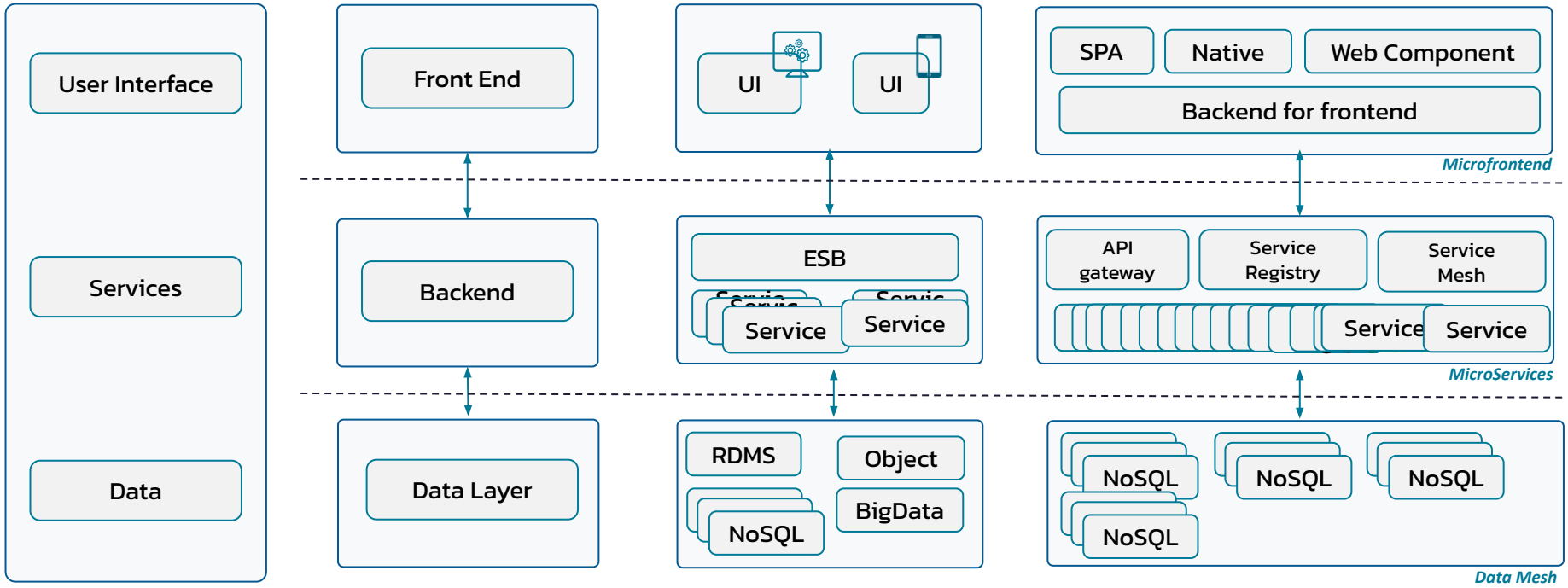
06

What's next?

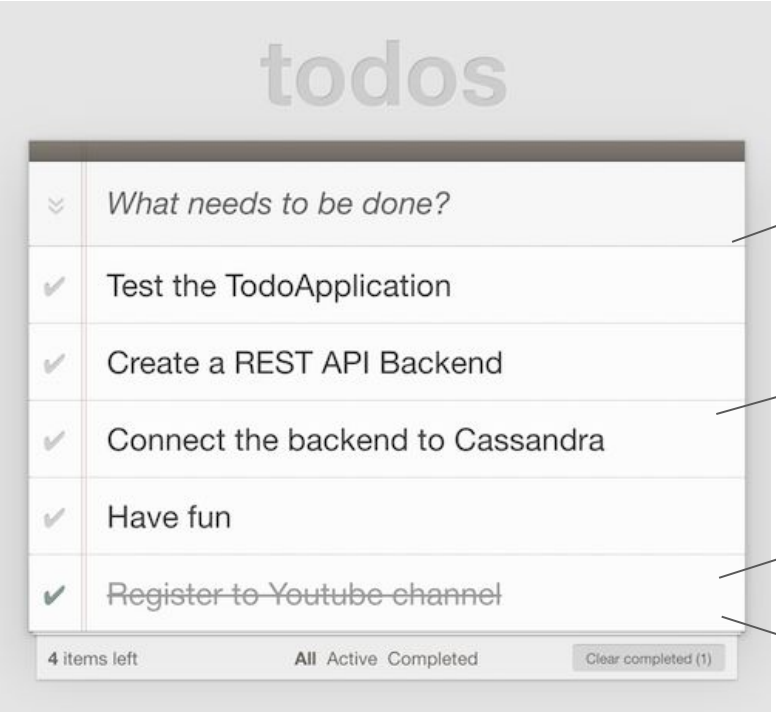
Quiz, Homework, Next week

Agenda

- Creating a ToDo app on multiple Java frameworks (Spring Boot, Quarkus, Micronaut)
- Data drivers for the app to connect to Astra
- Connect the App to Astra with minimal configuration changes for each framework
- Java Native and the respective frameworks
- [Stretch] Building a native app on each of the frameworks



Microservices Architecture evolution

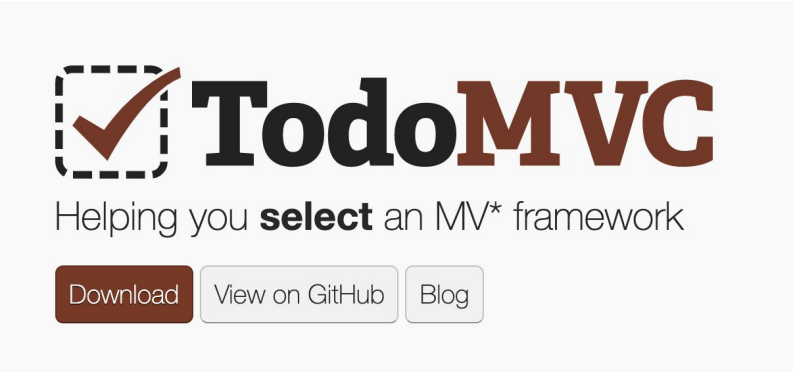


List all tasks

Create a new Task

Mark a task as completed/uncomplete

Delete a task



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

Todo-Backend

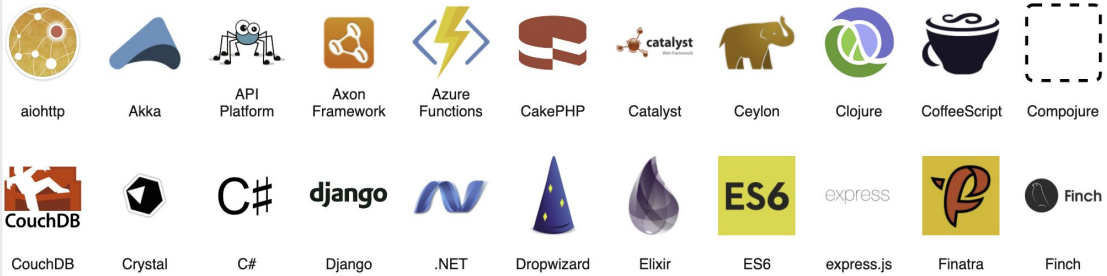
a shared example to showcase backend tech stacks

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.

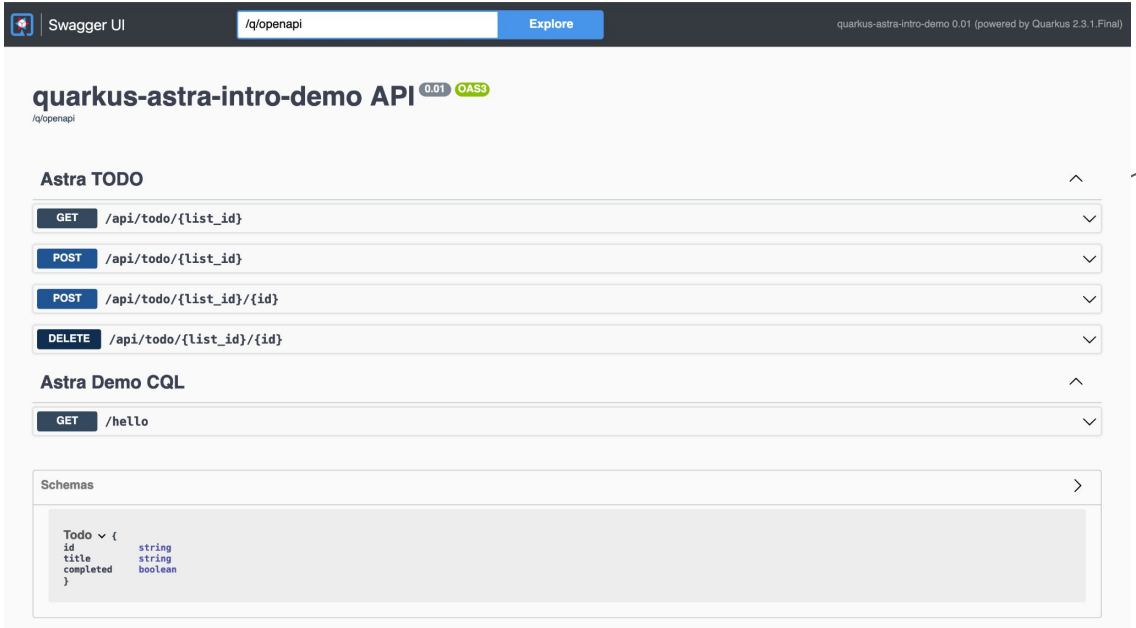
The Todo-Backend project was inspired by the TodoMVC project, and some code (specifically the todo client app) was borrowed directly from TodoMVC.

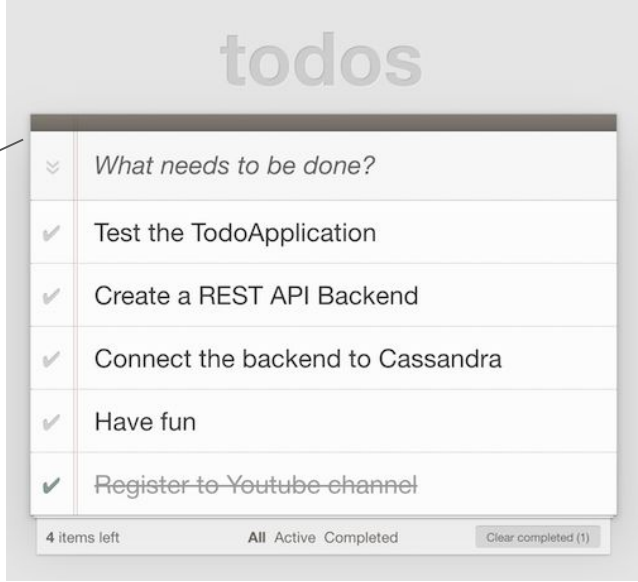
Created and curated by Pete Hodgson.

featuring HTTP APIs built with:

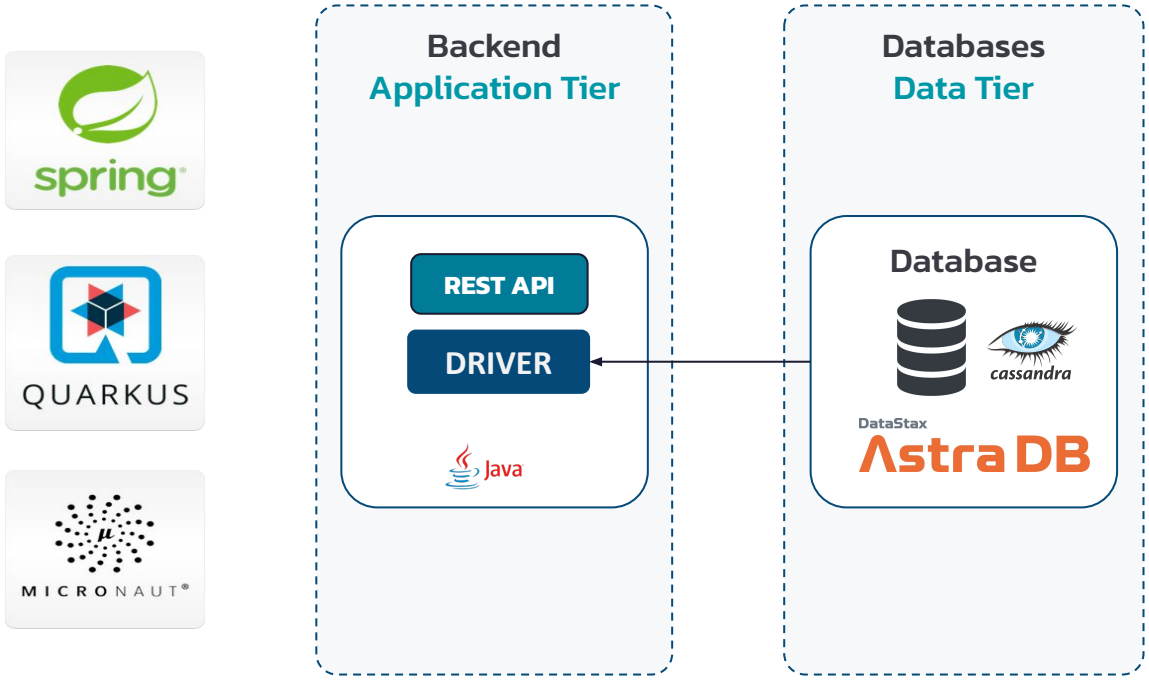


Todo Backend.com





Rest API Specifications (specs)



App Architecture

And today's badge for your pleasure (TBD)

01

Intro.

02

Cassandra

03

Java Drivers

04

Java Framework(s) & evolution

05

Spring, Quarkus, Micronaut

06

What's next?

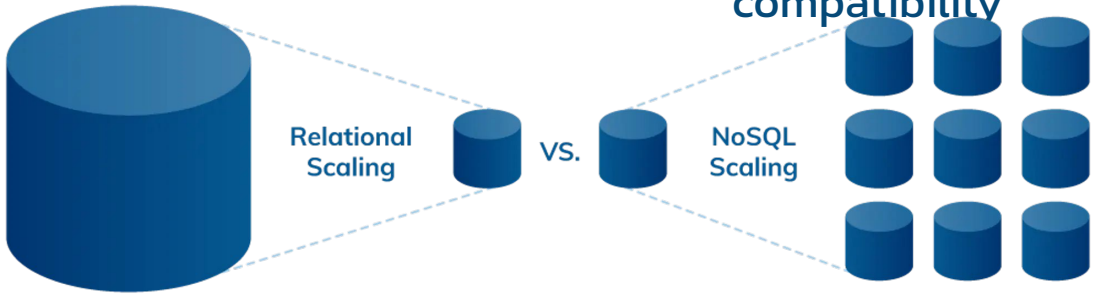
Quiz, Homework, Next week

Agenda

- 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”

Origin of the term “NoSQL”

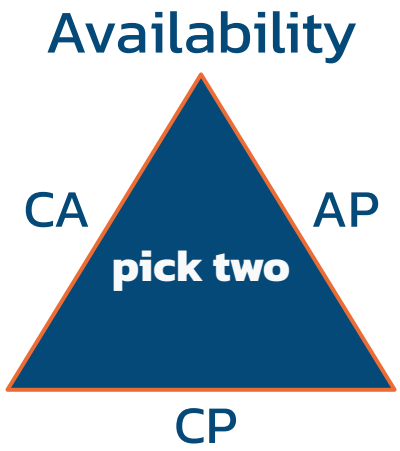
- Relational
  - Standard relational data model and language SQL
  - ACID transactions
  - Integration database
  - Designed for a single machine
  - Hard to scale
  - Impedance mismatch
- NoSQL
  - Variety of data models and languages
  - Lower-guarantee transactions
  - Application database
  - Designed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



Relational vs. NoSQL

The CAP Theorem

Always responds,
may not always return
the most recent write

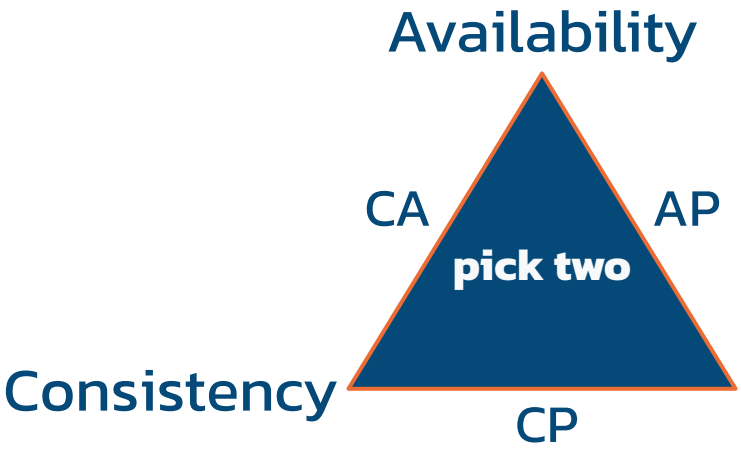


Every read receives
the most recent write
or an error

Operates in the
presence of network
partition failures

DataStax Graph

Astra DB



cassandra

DynamoDB

riak

redis

BigTable

APACHE HBASE

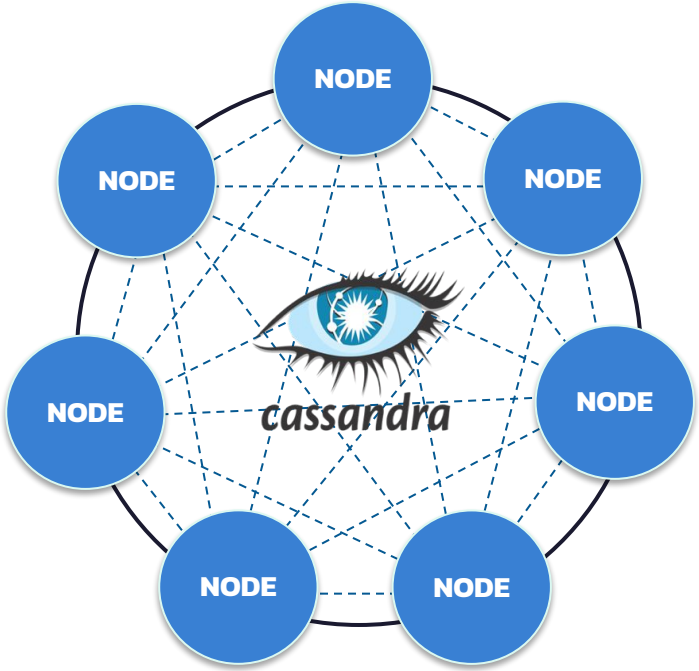
mongoDB.

Couchbase

The CAP Theorem

Cassandra doesn't belong to any of commercial vendors but controlled by a non-profit Open Source **Apache Software Foundation**, already familiar to you by *Hadoop*, *Spark*, *Kafka*, *Zookeeper, Maven* and many other projects.

Apache Cassandra= NoSQL Distributed Database



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

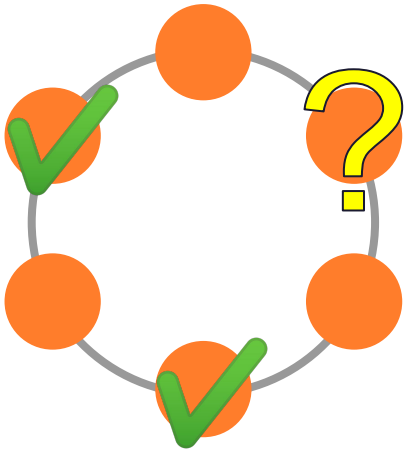
Why partitioning?
Because scaling doesn't have to be [s]hard!

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;

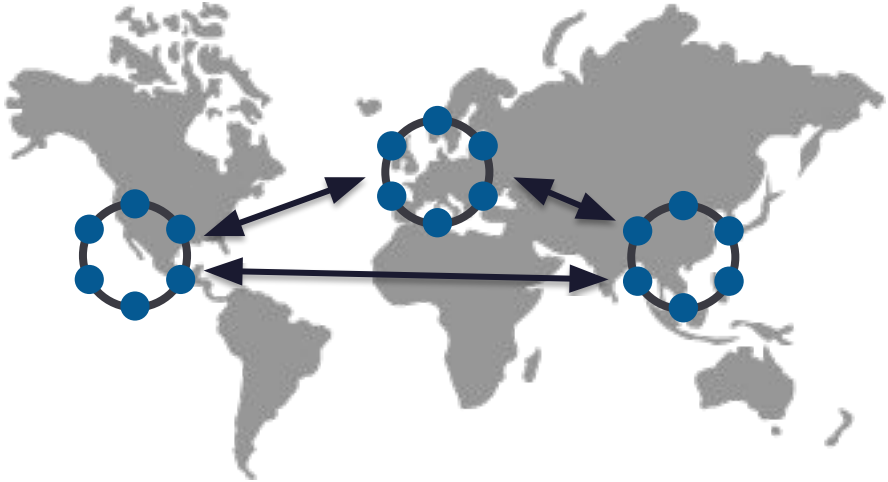
```
PreparedStatement pstmt = session.prepare(
    "INSERT INTO product (sku, description) VALUES (?, ?)"
);
pstmt.setConsistencyLevel(ConsistencyLevel.ONE);
```

```
cqlsh> CONSISTENCY

Current consistency level is QUORUM.

cqlsh> CONSISTENCY ALL

Consistency level set to ALL.
```

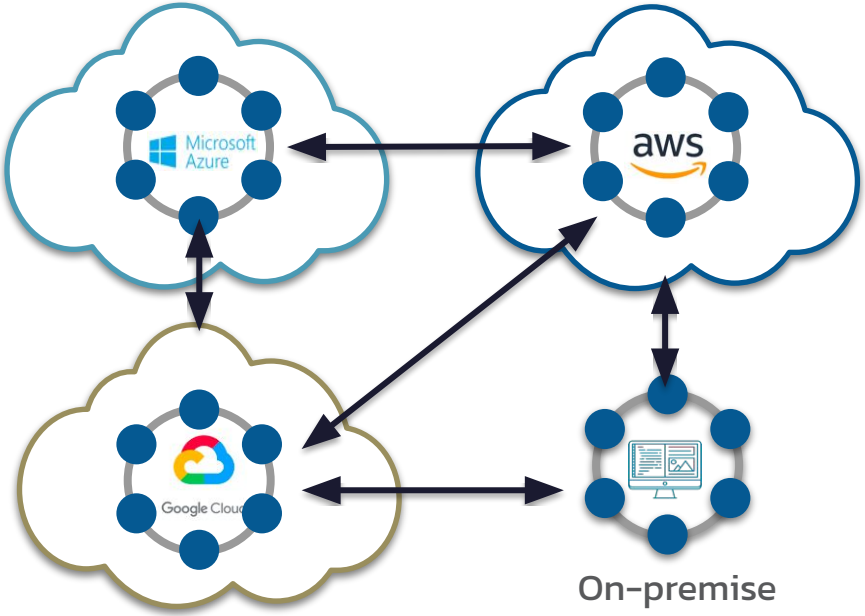


Is Cassandra AP or CP?

Geographical Distribution

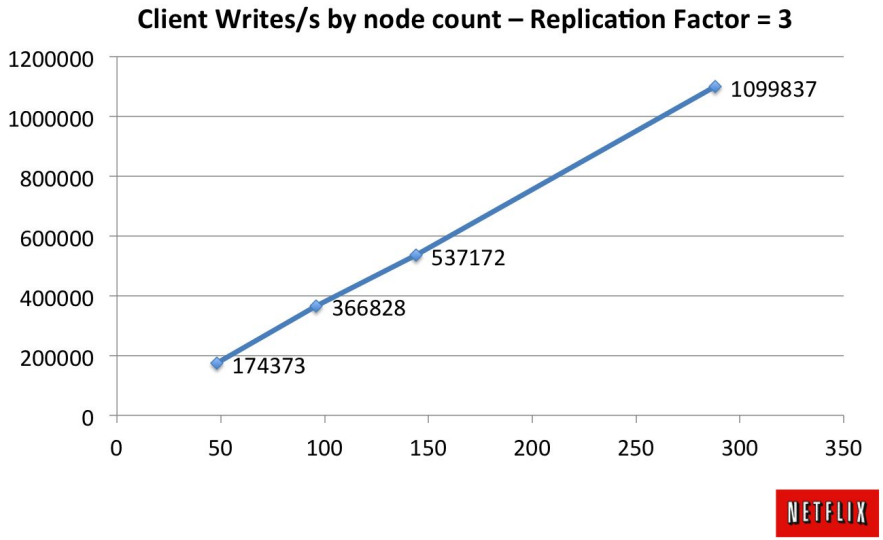


Hybrid-Cloud and Multi-Cloud



Data is globally distributed

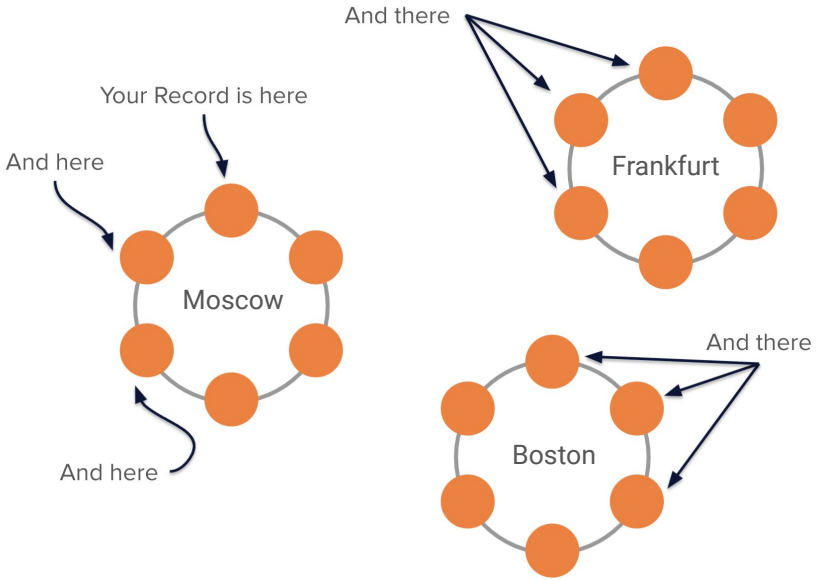
- For volume or velocity, there are no limitations
- **Linear** - No overhead on new nodes, scales with your needs*



Linear Scalability

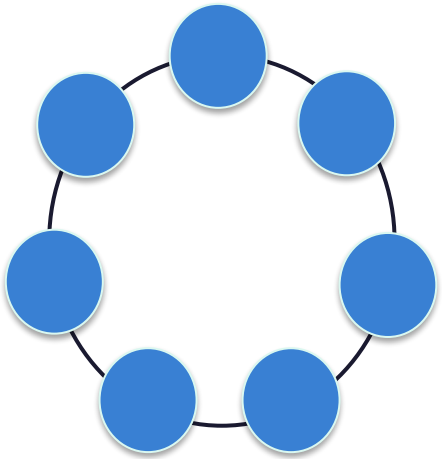
Replication, Decentralisation, and Topology-Aware Placement Strategy take care of possible downtimes:

- Multiple Live Replicas
- No Single Point of Failure
- Network topology-aware data placement
- Client-side Smart Reconnection and Strong Retry Mechanism



Highest Availability

Partitioning over distributed architecture makes the database capable to handle data of any size: we mean petabytes scale. Need more volume? Add more nodes.

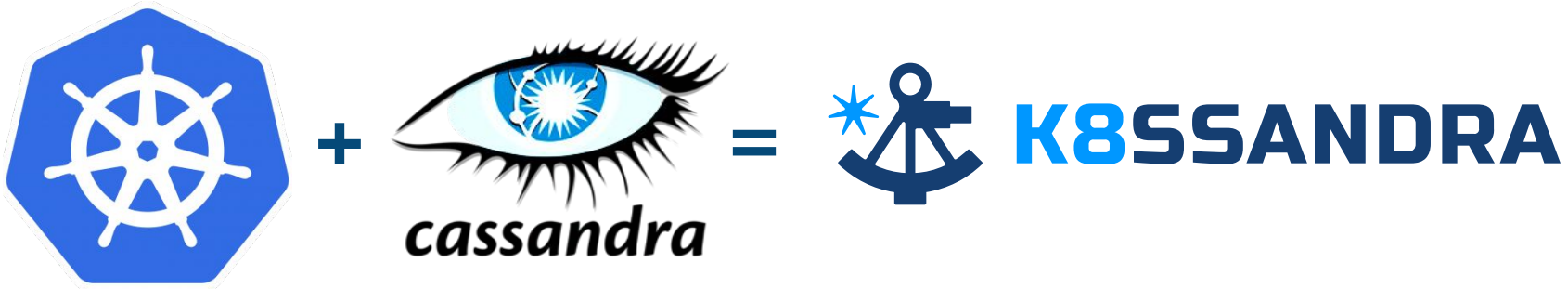


Big Data Ready

Astra DB

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

State-of-the-art NoSQL Database



K8ssandra – on Kubernetes

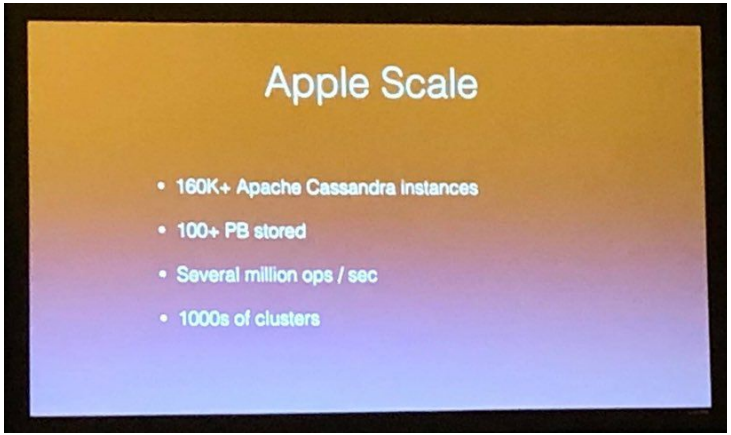
Apache Cassandra @ Netflix

- 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

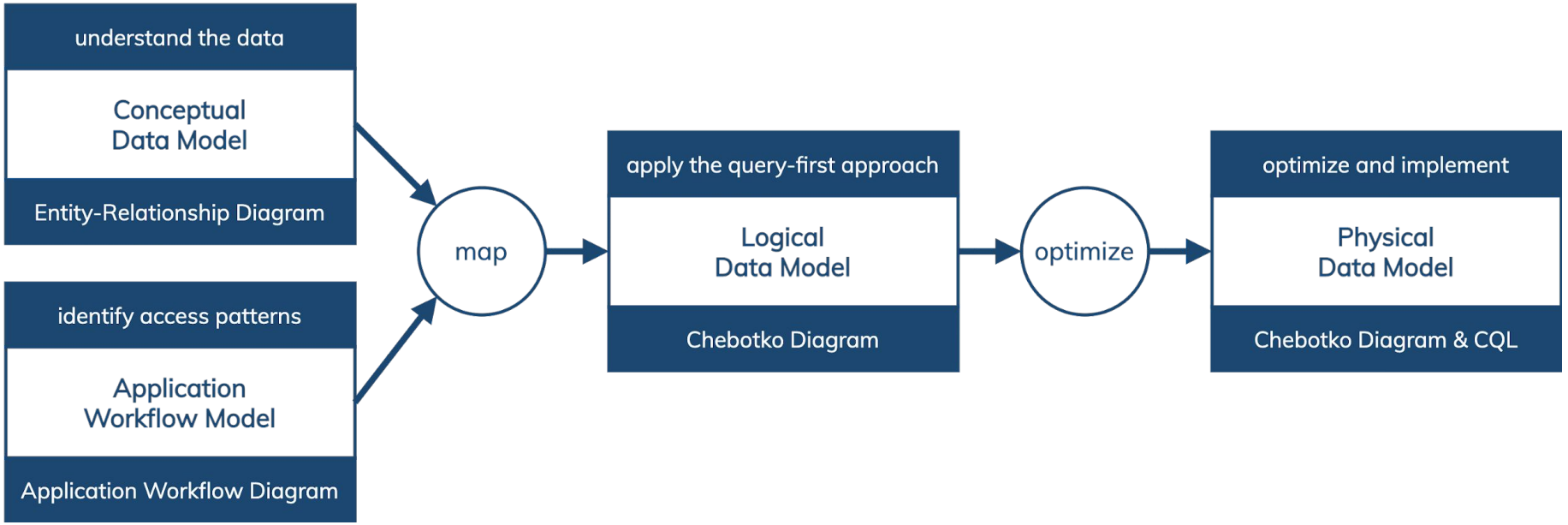
O(100)	Clusters
O(10000)	Instances
O(10,000,000)	Replications per second
O(100,000,000)	Operations per second
O(1,000,000,000,000,000)	Petabytes of data

dtsx.io/cassandra-at-netflix



And many others...

Cassandra Biggest Users (and Developers)


understand the data
Conceptual Data Model
Entity-Relationship Diagram
identify access patterns
Application Workflow Model
Application Workflow Diagram
map
apply the query-first approach
Logical Data Model
Chebotko Diagram
optimize
optimize and implement
Physical Data Model
Chebotko Diagram & CQL

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,
    offset      int,
    PRIMARY KEY ((user_id),item_id)
);
```

```
token@cqlsh:native_java> select * from todoitems where user_id='john';

 user_id | item_id                              | completed | offset | title
---------+--------------------------------------+-----------+--------+--------------------
    john | 11111111-5cff-11ec-be16-1fedb0dfd057 |      True |   null |        Walk the dog
    john | 22222222-5cff-11ec-be16-1fedb0dfd057 |     False |   null | Have lunch tomorrow
    john | 708520d0-76af-11ec-9bee-6f21c69b6a43 |     False |      0 |        Sample task1

(3 rows)
token@cqlsh:native_java>
```

TodoApplication DataModel

Lab 0

Setup

https://github.com/datastaxdevs/workshop-spring-quarkus-micronaut-cassandra#setup

01

Intro.

02

Cassandra

03

Drivers

04

Java Framework(s) & evolution

05

Spring, Quarkus, Micronaut

06

What's next?

Quiz, Homework, Next week

Agenda





Connectivity

- ★ Token & Datacenter Aware
- ★ Load Balancing Policies
- ★ Retry Policies
- ★ Reconnection Policies
- ★ Connection Pooling
- ★ Health Checks
- ★ Authentication | Authorization
- ★ SSL

Query

- ★ CQL Support
- ★ Schema Management
- ★ Sync/Async/Reactive API
- ★ Query Builder
- ★ Compression
- ★ Paging

Parsing Results

- ★ Lazy Load
- ★ Object Mapper
- ★ Spring Support
- ★ Paging

Maven

```
<dependency>
    <groupId>com.datastax.oss</groupId>
    <artifactId>java-driver-core</artifactId>
    <version>4.13.0</version>
</dependency>
```

Java

4.6.3 npm

```
npm install cassandra-driver
```

JavaScript

3.24.0 python Package Index

```
pip install cassandra-driver
```

python

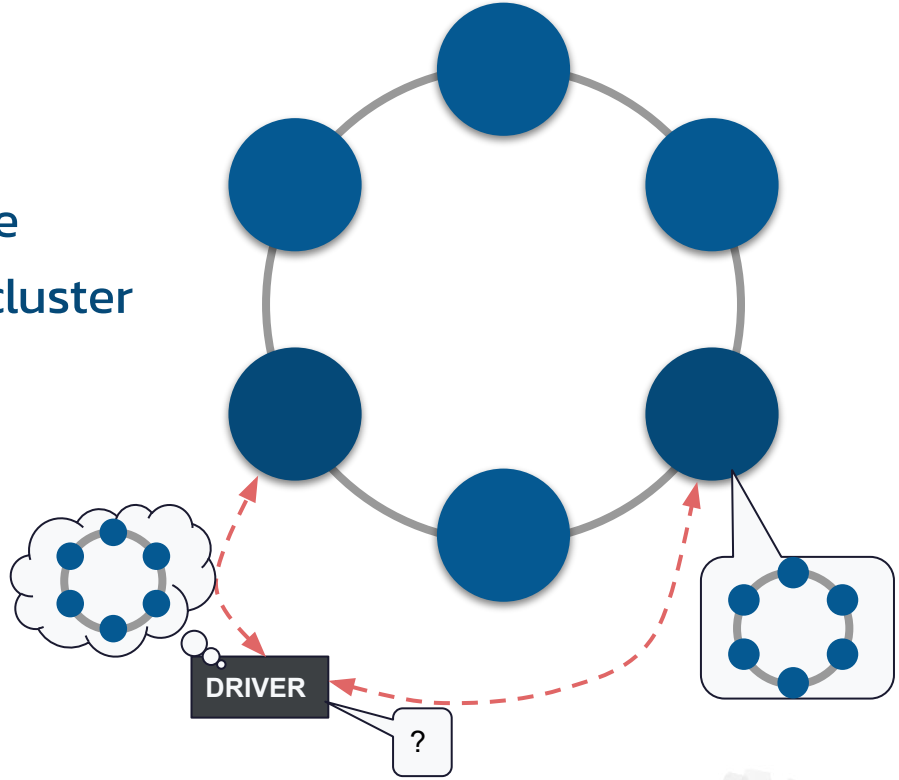
nuget v3.15.0

```
Install-Package CassandraCSharpDriver -Version 3.15.0
```

C#

Installing the drivers

- Only one necessary
- Unless that node is down
- Better ~3 nodes per DC for resilience
- From there, drivers discover whole cluster
- Local Datacenter



- **CqlSession** is a stateful object handling communications with each node
- **CqlSession** should be unique in the Application (*Singleton*)
- **CqlSession** should be closed at application shutdown (*shutdown hook*) in order to free opened TCP sockets (*stateful*)

```
@PreDestroy
public void cleanup() {
 if (null != cqlSession) {
   cqlSession.close();
 }
}
```

```java
CqlSession cqlSession = CqlSession.builder()
 .addContactPoint(new InetSocketAddress("127.0.0.1", 9042))
 .withKeyspace("killrvideo")
 .withLocalDatacenter("dc1")
 .withAuthCredentials("U","P")
 .build();
```

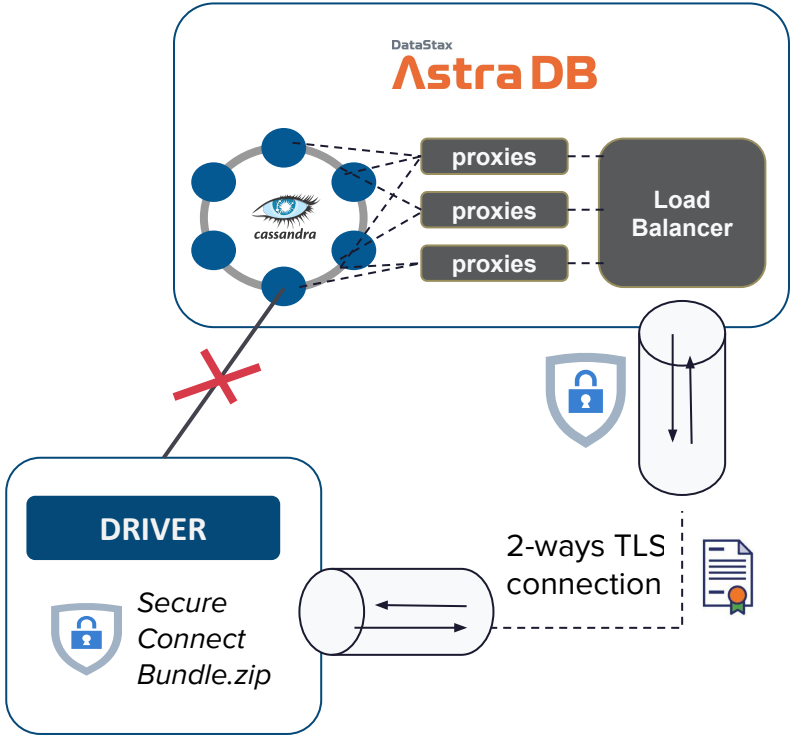
```node
const client = new cassandra.Client({
  contactPoints: ['127.0.0.1'],
  localDataCenter: 'dc1',
  keyspace: 'killrvideo',
  credentials: { username: 'U', password: 'P' }
});
```

```python
auth_provider = PlainTextAuthProvider(
   username='U', password='P')
cluster = Cluster(['127.0.0.1'],
  auth_provider=auth_provider, protocol_version=2)
session = cluster.connect('todos')
```

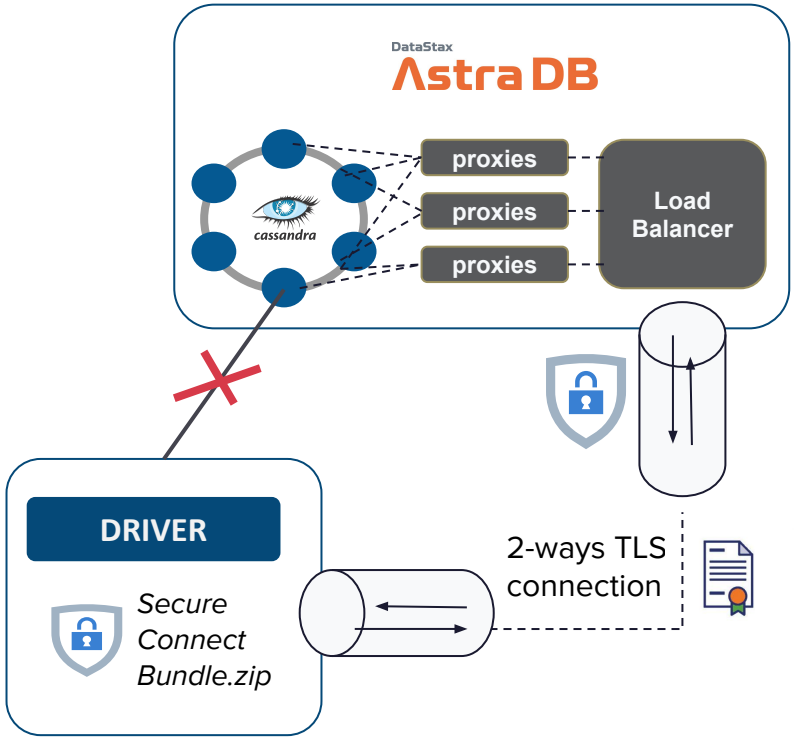
```dotnet
Cluster cluster = Cluster.Builder()
.AddContactPoint("127.0.0.1")
.WithCredentials("U", "P")
.Build();
session = cluster.Connect("todos");
```

Create Session/Client with Contact Points

- Secured connection over HTTP required.
- A *secureConnectBundle* ZIP is expected
- Your username is clientId
- Your Password is clientSecret
- No Single Point of failure (SPOF)



Contact Points with ASTRA


DataStax
Astra DB
proxies
proxies
proxies
Load Balancer
cassandra
DRIVER
Secure Connect Bundle.zip
2-ways TLS connection

```java
CqlSession cqlSession = CqlSession.builder()
 .withCloudSecureConnectBundle(Paths.get("secure.zip"))
 .withAuthCredentials("U","P")
 .withKeyspace("todos")
 .build();
```


```javascript
const client = new cassandra.Client({
  cloud: { secureConnectBundle: 'secure.zip' },
  credentials: { username: 'u', password: 'p' }
});
```


```python
auth_provider = PlainTextAuthProvider(
   username='U', password='P')
cluster = Cluster(
   Cloud ={ Secure_connect_bundle: 'secure.zip'},
   auth_provider=auth_provider, protocol_version=2)
session= cluster.connect('todos')
```


```
var cluster = Cluster.Builder()
 .WithCloudSecureConnectionBundle("secure.zip")
 .WithCredentials("u", "p")
 .Build();
var session = cluster.Connect("todos");
```

Create Session/Client with ASTRA

```
datastax-java-driver {
   session-keyspace = ks_java
   cloud {
     secure-connect-bundle = /workspace/workshop-spring-quarkus-micronaut-cassandra/secure-bundle-workshops.zip
   }
 }
  advanced {
   auth-provider {
     class = PlainTextAuthProvider
     username = "token"
     password = ${ASTRA_DB_APP_TOKEN}
   }
 }
}
CqlSession cqlSession = CqlSession.builder().build()
```

Parameters in Statements

- **CqlSession** execute method.

```
cqlSession.execute("SELECT * FROM ks.todos");
```

```
Statement statement = …

// (1) Explicit SimpleStatement Definition
SimpleStatement.newInstance("select * from t1 where c1 = 5");

// (2) Externalize Parameters (no name)
SimpleStatement.builder("select * from t1 where c1 = ?")
               .addPositionalValue(5);

// (3) Externalize Parameters (name)
SimpleStatement.builder("select * from t1 where c1 = :myVal")
               .addNamedValue("myVal”, 5);

// (4) Paging rows at a time
SimpleStatement.builder("select * from t1")
               .setPageSize(10);
cqlSession.execute(statement);
```

Parameters in Statements

- Compiled once on each node automatically as needed
- Prepare each statement only once per application
- Use one of the many bind variations to create a BoundStatement

```
PreparedStatement ps = cqlSession.prepare("SELECT * from t1 where c1 = ?");

BoundStatement bound = ps.bind(5);

cqlSession.execute(bound);
```

- ResultSet is the object returned for executing query. It contains ROWS (data) and EXECUTION INFO.
- ResultSet is iterable and as such you can navigate from row to row.
- Results are always paged for you (avoiding memory and response time issues)

```
ResultSet rs = cqlSession.execute(myStatement);

ExecutionInfo info = rs.getExecutionInfo();
int executionTime = info.getQueryTrace().getDurationMicros();

Iterator<Row> iterRow = rs.iterator();
int itemsFirstCall = rs.getAvailableWithoutFetching();
```

```
// Sample row
Row row = resultSet.one();

// Check null before read
Boolean isUserNameNull = row.isNull("userName");

// Reading Values from row
String userName1 = row.get("username", String.class);
String userName2 = row.getString("username");
String userName3 = row.getString(CqlIdentifier.fromCql("username"));

// Tons of types available
row.getUuid("userid");
row.getBoolean(”register");
row.getCqlDuration(”elapsed");
…
```


Parsing Rows

- WHAT ?
  - Abstracts details of mapping Java attributes to/from CQL types and UDTs
  - Packaged separately from the driver – pom.xml update required

- HOW ?
  - Some annotation processors will GENERATE Mapper, DAO, and Entity implementations for you
  - At each update in the files, the IDE (eclipse, intelliJ) will use annotation processor
  - Compiler plugin must be updated to define the annotation processor

```
<dependency>
 <groupId>com.datastax.oss</groupId>
<artifactId>java-driver-mapper-runtime</artifactId>
</dependency>


<plugin>
 <groupId>org.apache.maven.plugins</groupId>
 <artifactId>maven-compiler-plugin</artifactId>
 <configuration>
   <release>${java.version}</release>
   <source>${java.version}</source>
   <target>${java.version}</target>
   <annotationProcessorPaths>
    <path>
      <groupId>com.datastax.oss</groupId>
      <artifactId>java-driver-mapper-processor</artifactId>
      <version>${cassandra-java-driver.version}</version>
    </path>
  </annotationProcessorPaths>
</configuration>
</plugin>
```

Object Mapper

```
@Entity
@CqlName("user_v")
public class UserVideo {

 @PartitionKey
 @CqlName("userid")
 private UUID userid;

 @ClusteringColumn(1)
 @CqlName("added")
 private UUID videoid;

}
```

TABLE NAME, KEYSPACE

PARTITION KEY COLUMNS

CLUSTERING COLUMNS

Annotate Entities

```
@Dao
public interface VideoDao {

  @Select
  Optional<UserVideo> findUserById(UUID userid);

  @Query("SELECT * FROM ${tableId}")
  PagingIterable<UserVideo> findAll();

  @Select(customWhereClause = "videoid = : videoid")
  PagingIterable<UserVideo>
   findUserByVideoId(@CqlName("videoid") UUID vid);
```

Annotated Dao Interfaces (1 ON 2)

```
// Save a bean
@Insert
void save(UserVideo userVideo);

// Userid id is PK
@Delete
void delete(UUID userid);

// Custom implementations
@QueryProvider(
  providerClass = MySampleQueryProvider.class,
  entityHelpers = { UserVideo.class })
String doSomething(String abc);
```

Annotated Dao Interfaces (2 ON 2)

Lab 1

Java Drivers

https://github.com/datastaxdevs/workshop-spring-quarkus-micronaut-cassandra#lab-1---understanding-java-drivers

Lab 2

Test Connection

https://github.com/datastaxdevs/workshop-spring-quarkus-micronaut-cassandra#lab-2---spring-data-cassandra

01

Intro.

02

Cassandra

03

Drivers

04

Java Framework(s) & evolution

05

Spring, Quarkus, Micronaut

06

What's next?

Quiz, Homework, Next week

Agenda

Traditional

- **Throughput** at the expense of **footprint**
- **Long running** at expense of **startup speed**
- Rich, dynamic behavior for mutable systems

Cloud Native

- **Throughput** solved by scaling
- Ephemeral, immutable systems
- Footprint and performance matter

Java: Designed for the modern era

Build Time

Runtime

Packaging (maven, gradle...)

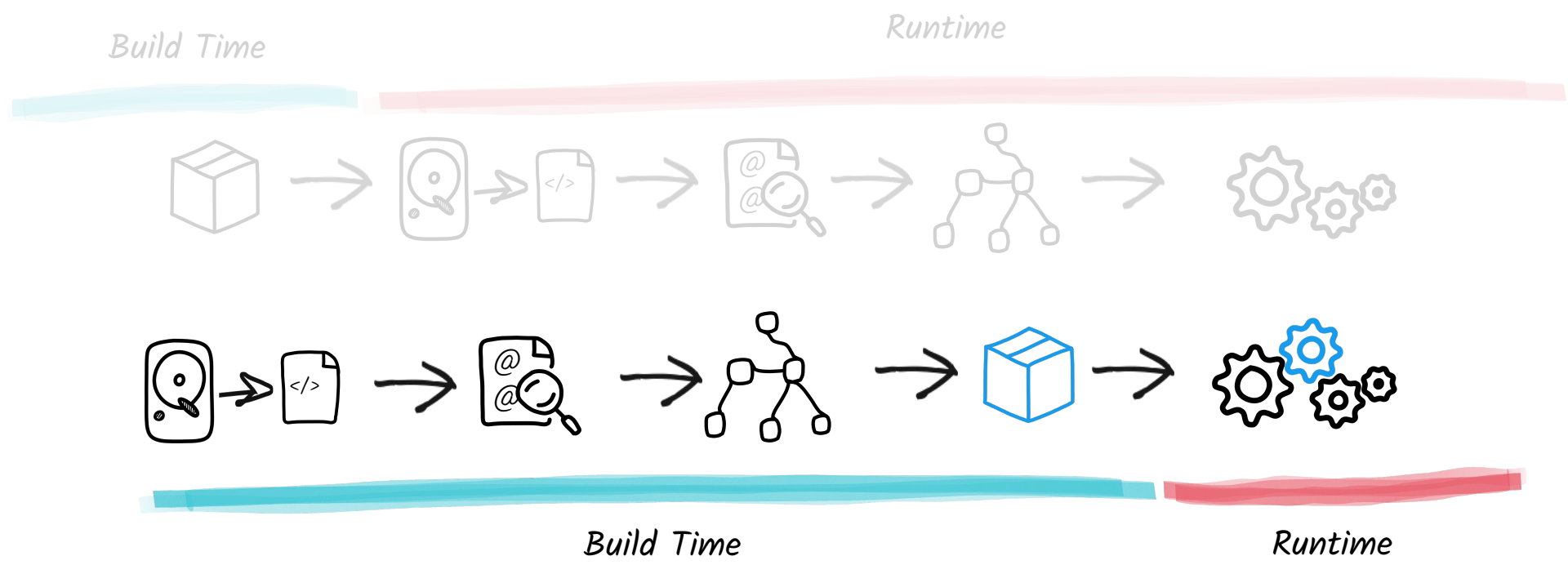
Load config file from file system
Parse it

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...)

How does a Typical Java Framework Work?

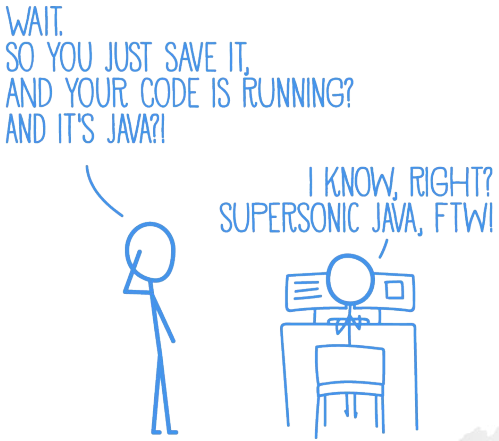


The “cloud native” Way

A cohesive framework for Microservices

- Developer productivity
  - Zero-config Live coding
  - Developer services
  - Continuous testing
  - Dev UI and CLI
- Streamlined code for the 80% common usages
- Native executable generation

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

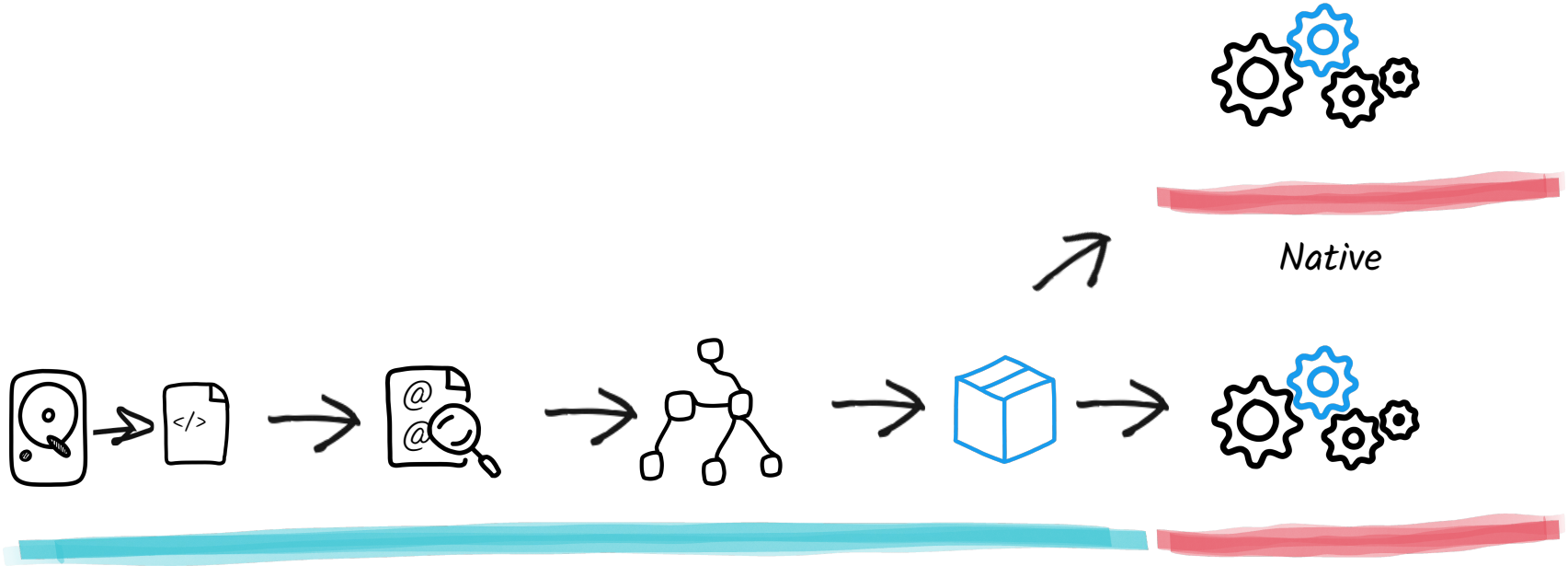


Inner loop == Developer Productivity

- Just-in Time (JIT) compilation (optimized at run time)
  - GraalVM
- Ahead of Time (AoT) compilation (optimized at build time)
  - Spring Native
  - Quarkus
  - Micronaut
- Native executable generation
  - GraalVM
    - Early Adopter Technology

- Milliseconds start
- No JVM resource overhead
- Cold starts (no warmup).
- Suitable for containerization
- Reduced attacks.

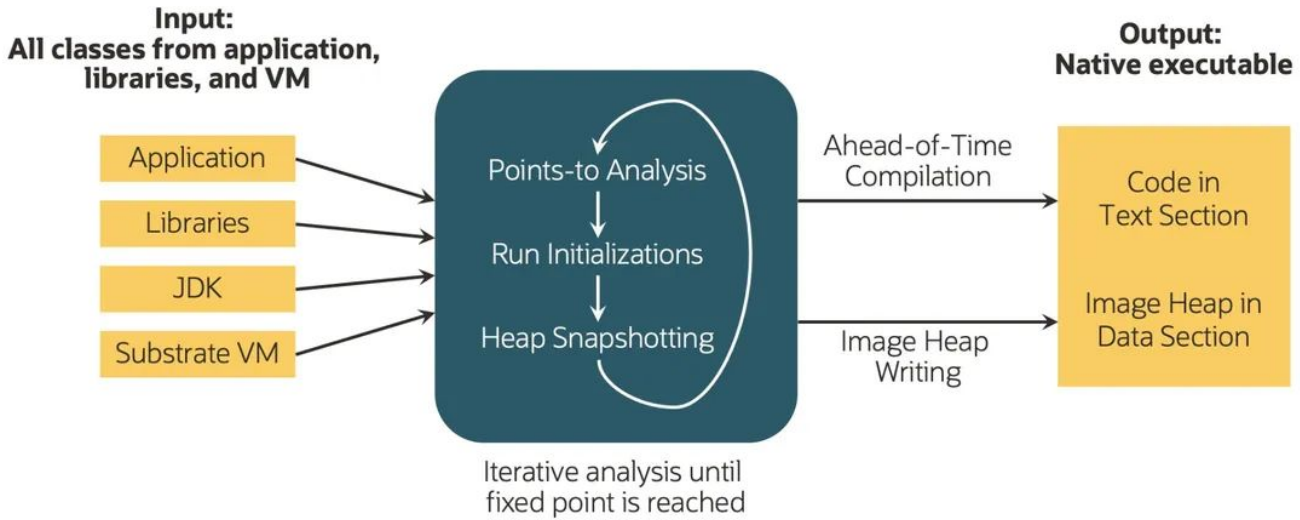

Ahead of Time (AoT) Compilation



Build Time

JVM

Native Compilation



Native Images (pic. Courtesy InfoQ)

STILL JAVA!

01

Intro.

02

Cassandra

03

Drivers

04

Java Framework(s) evolution

05

Spring ...

06

What's next?

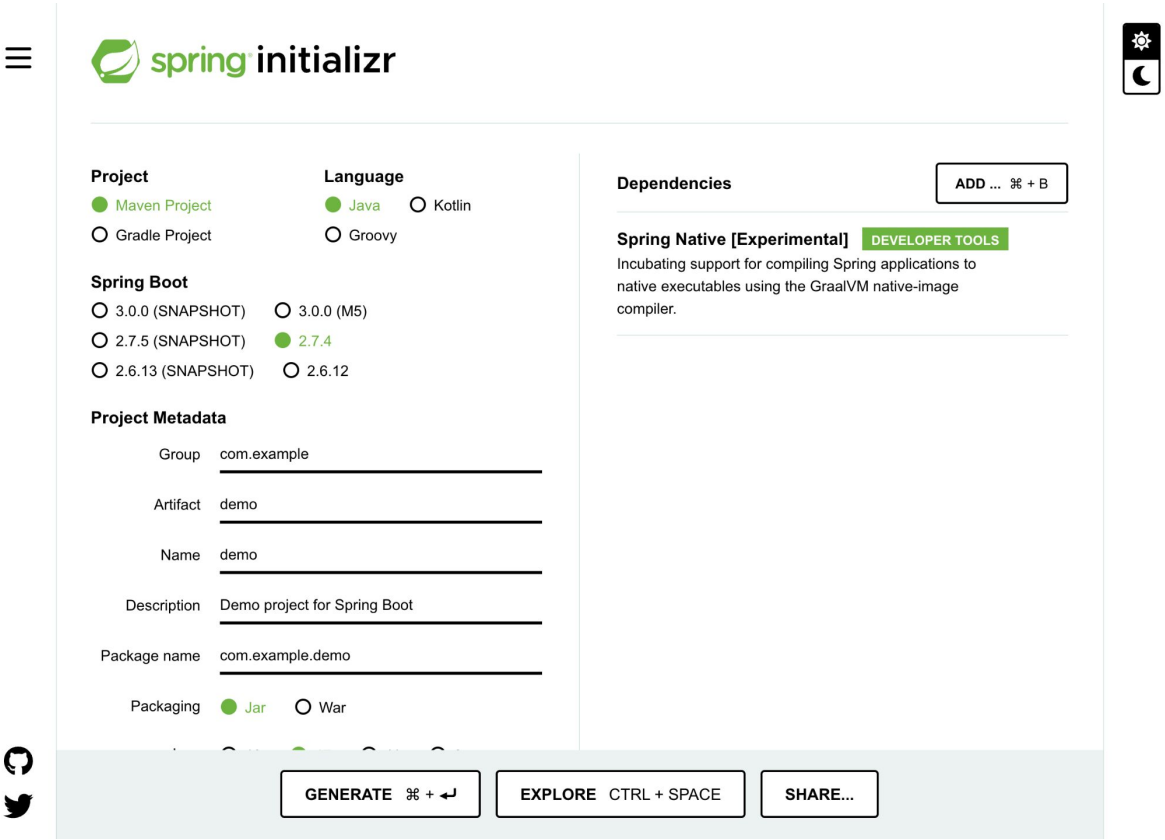
Quiz, Homework, Next week

Agenda

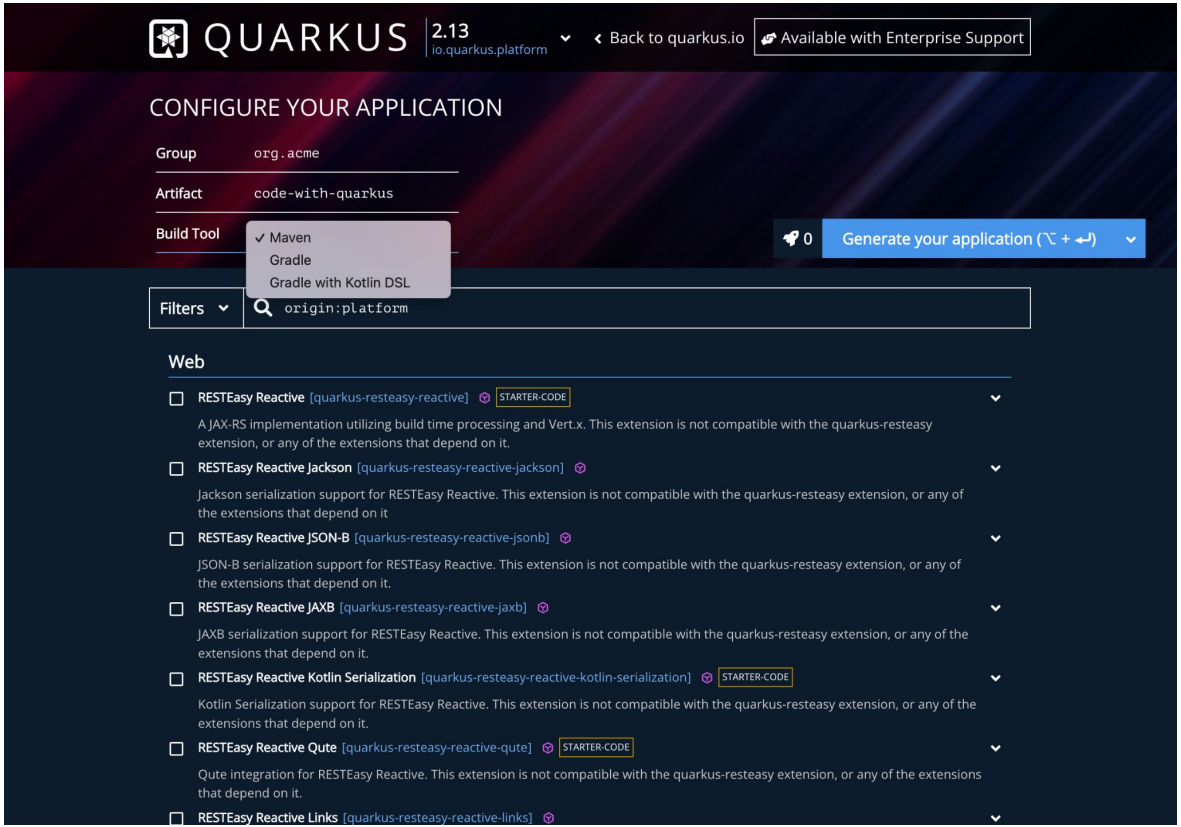
Lab 3

Spring Boot/ Data

https://github.com/datastaxdevs/workshop-spring-quarkus-micronaut-cassandra#lab-2---spring-data-cassandra



Spring Initializr



Quarkus Initializtion

01

Intro.

02

Cassandra

03

Drivers

04

Java Framework(s) evolution

05

..., Quarkus, ...

06

What's next?

Quiz, Homework, Next week

Agenda

- 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)

```
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
```

Native Quarkus Config

```
@Inject QuarkusCqlSession session;


@GET
@Produces(MediaType.APPLICATION_JSON)
@Path(“/product/{id}”)
public Uni<Response> findProduct(@PathParam(“id”) String id) {
  return session.executeReactive("SELECT * FROM table_name WHERE key = " + id)
    .toUni()
    .map(row -> new Product(row.getString("id"), row.getString("name")))
    .map(product -> Response.ok(product).build())
    .ifNoItem().after(Duration.ofSeconds(5))
    .recoverWithItem(Response.status(Status.NOT_FOUND).build());
}
```

Cassandra Driver Session Support

```
@Dao interface ProductDao {
  @Insert Uni<Void> create(Product product);
  @Select Uni<Product> findById(String id);
  @Select Multi<Product> findAll();
}

interface ProductMapper {
  @DaoFactory
  ProductDao productDao();
}

@ApplicationScoped class ProductService {
  @Inject ProductDao dao;
}
```

Cassandra Driver Object Mapper Support

```
@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());
}
```

Support for Mutiny Types

Lab 4

Quarkus

https://github.com/datastaxdevs/workshop-spring-quarkus-micronaut-cassandra#lab-2---spring-data-cassandra

01

Intro.

02

Cassandra

03

Drivers

04

Java Framework(s) evolution

05

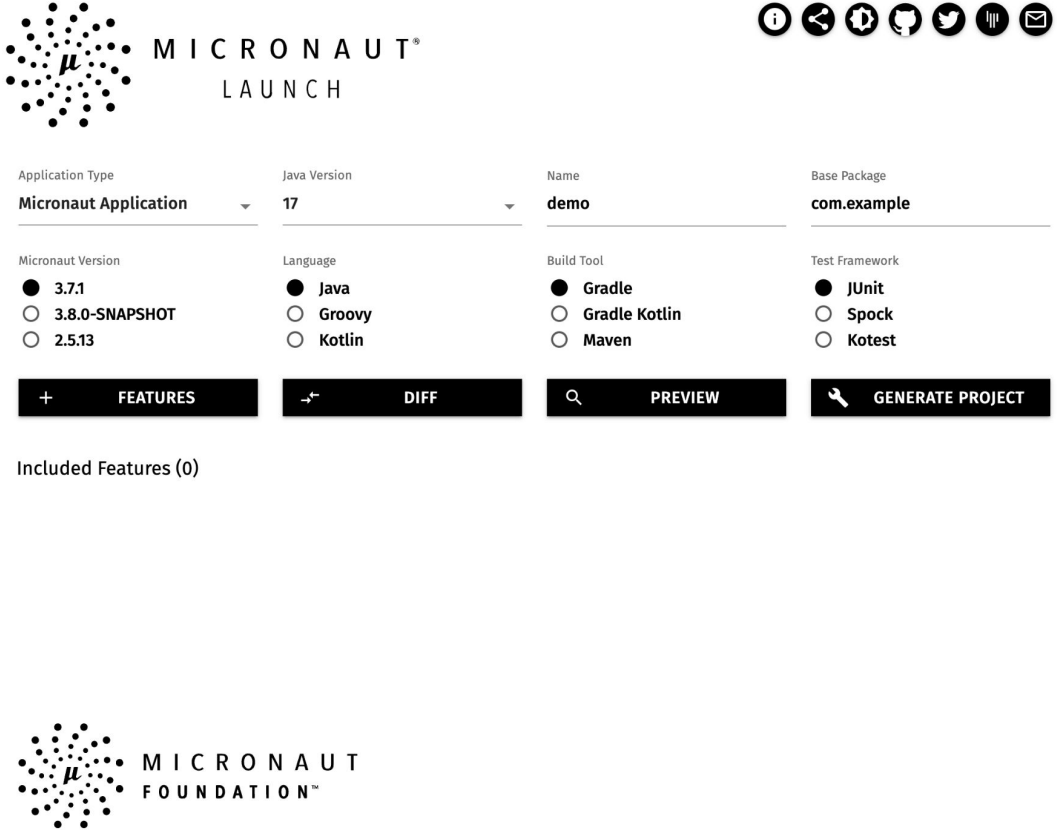
..., Micronaut

06

What's next?

Quiz, Homework, Next week

Agenda



Micronaut Initialization

Lab 5

Micronaut

https://github.com/datastaxdevs/workshop-spring-quarkus-micronaut-cassandra#lab-2---spring-data-cassandra

01

Intro.

02

Cassandra

03

Drivers

04

Java Framework(s) & evolution

05

Spring, Quarkus, Micronaut

06

What's next?

Quiz, Homework, Next week

Agenda

Workshop(s)

https://github.com/datastaxdevs/workshop-spring-quarkus-micronaut-cassandra

https://github.com/datastaxdevs/workshop-intro-quarkus-cassandra

https://www.infoq.com/articles/native-compilations-boosts-java/

Series Contents

1 [Revolutionizing Java with GraalVM Native Image](#)

GraalVM Native Image is an ahead-of-time compiler that generates native Java executables. These executables start very fast and use less CPU and memory. This makes Java in the cloud cheaper. GraalVM can even achieve peak throughput on par with the JVM. Many Java frameworks already support GraalVM, such as Spring Boot, Micronaut, Quarkus, Gluon, etc.

ARTICLE BY: Alina Yurenko

2 [Kubernetes Native Java with Quarkus](#)

Quarkus is an industry leader in startup time and memory utilization for native and JVM-based Java applications. This reduces cloud costs. Kubernetes is a first-class deployment platform in Quarkus with support for its primitives and features. Developers can use their Java knowledge of APIs like Jakarta EE, MicroProfile, Spring, etc. Applications can be imperative or reactive - or both!

ARTICLE BY: Jason Greene, John Clingan, Eric Deandrea

3 [Native Java in The Real World](#)

Java Native

- 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

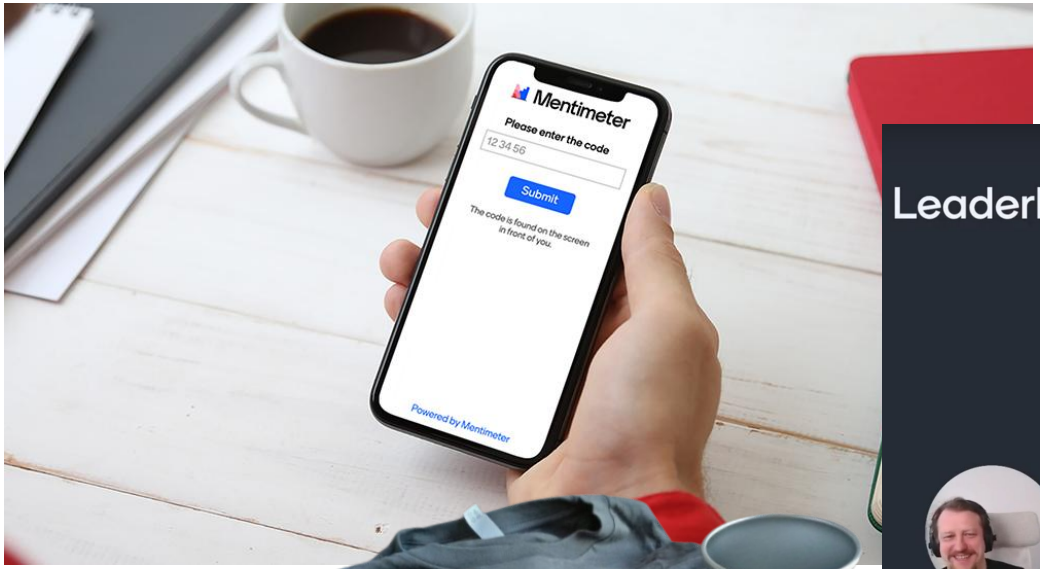
https://dzone.com/refcardz/getting-started-with-quarkus-serverless-functions

Try Cassandra+Java Frameworks

- Initializers
  - https://start.spring.io/
  - https://code.quarkus.io/
  - https://micronaut.io/launch/

- Check out https://k8ssandra.io
- Get coding + see docs for more info and try the Java Native + Cassandra workshop

https://github.com/datastaxdevs/workshop-spring-quarkus-micronaut-cassandra

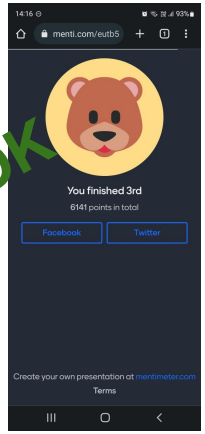


`menti.com` ⇒ enter code
Don't answer in YT chat
Look at phone (not at YT)

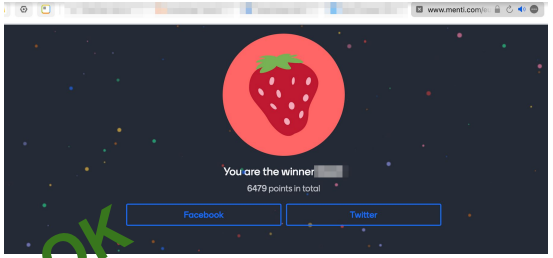
Quiz on "Menti" !

SWAG WINNERS 🎉

Congratulations to 1st, 2nd and 3rd place on the Menti quiz!



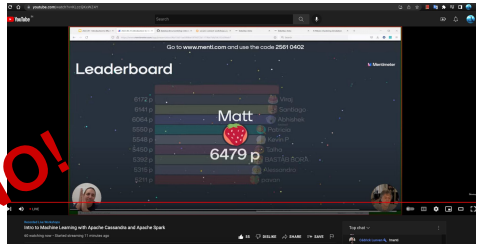
OK



OK

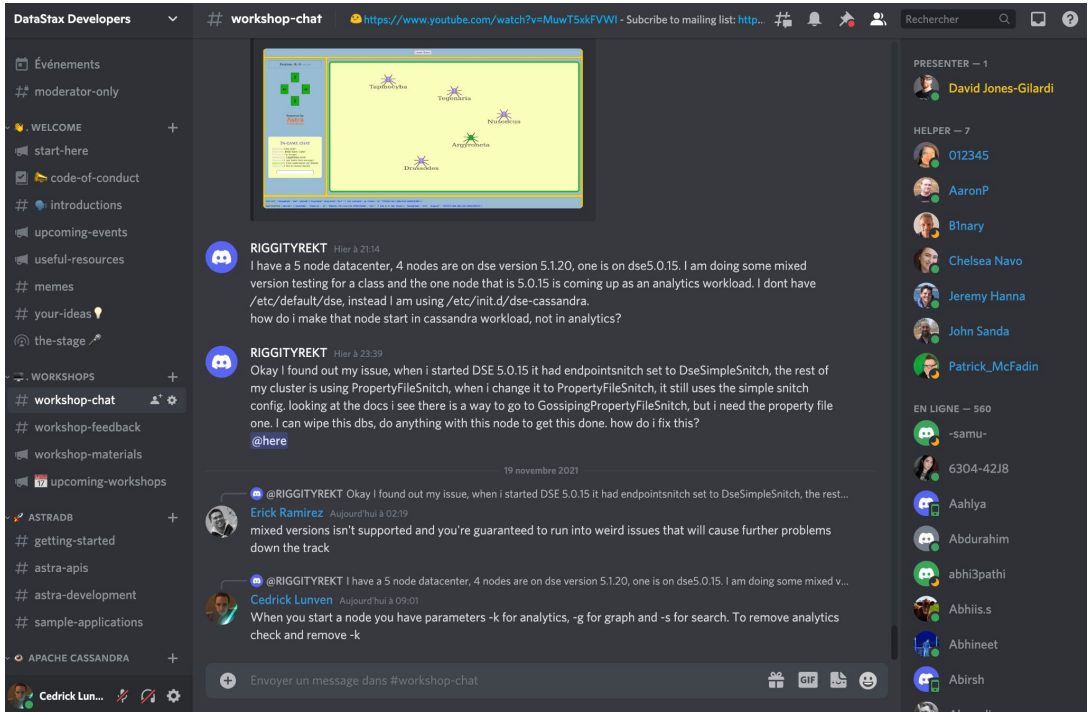
To claim your prize:

Take a screenshot of your Menti screen



NO!

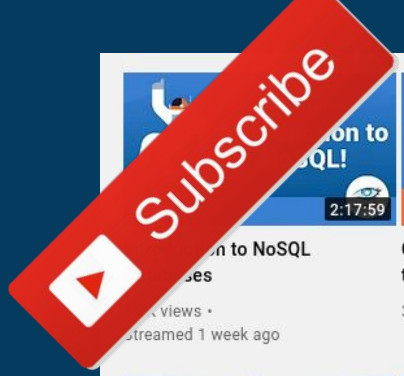
Fill the form at dtsx.io/workshop-swag



!discord

dtsx.io/discord

Datastax Developers Discord (18k+)



...n to NoSQL ...ses

...K views •
...Streamed 1 week ago

Crash Course | Introduction to Cassandra for Developers

331 views • 1 week ago

Introduction to NoSQL Databases

3.5K views •
Streamed 1 week ago

Introduction to NoSQL Databases

10K views •
Streamed 1 week ago

Build your own NETFLIX clone!

4K views •
Streamed 2 weeks ago

Build your own NETFL... clone!

7.4K views •
Streamed 2 weeks ago

Astra Streaming Demo

177 views • 2 weeks ago

Kubernetes Ingress Management with Traefik...

496 views •
Streamed 2 weeks ago

Build your own TikTok clone!

1.9K views •
Streamed 3 weeks ago

Build your own TikTok Clone!

4K views •
Streamed 3 weeks ago

How to use the Connect Driver in Astra DB

113 views • 4 weeks ago

How to use the CQL Console in Astra DB

39 views • 4 weeks ago

How to create an Authentication Token in...

37 views • 4 weeks ago

How to use the Data Loader in Astra DB

62 views • 4 weeks ago

Astra DB Sample App Gallery

36 views • 4 weeks ago

How to use Secure Connect in Astra DB

42 views • 4 weeks ago

Cassandra Day India: CL Room (Workshops)

2.4K views •
Streamed 4 weeks ago

Cassandra Day India: RF Room (Talks)

1.3K views •
Streamed 1 month ago

datastax.com/workshops

Become a Jedi Master of Astra

Cassandra Days … in-person events are back!

~~Berlin~~	~~Sept 20~~
London	Oct 11
Amsterdam	Oct 13
Santa Clara	Nov 10
Seattle	Nov 10
Houston	Nov 10
Hanoi	Nov 8
Jakarta	Nov 10
Singapore	Nov 15

Find out more and register at https://www.datastax.com/events

- **Training day March 12**
- **In-person**
- **Virtual**
- **CFP and Reg coming soon**

Thank You!