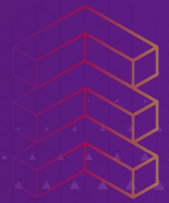
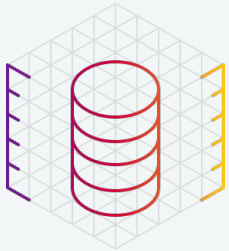
DATASTAX
DEVELOPERS

Application Development Bootcamp Week III: Building the App!

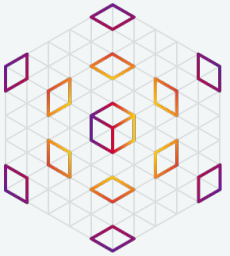
Implement Java, Python and Javascript (Nodejs) APIs on Apache Cassandra







Agenda



01

Introduction

02

Microservices

Why with Apache Cassandra?

03

REST APIs

04

Sensor Application

Drivers; via command-line

05

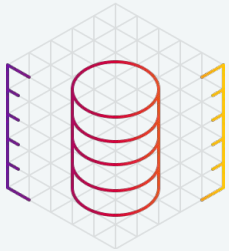
Sensor REST API

FastAPI, Spring Boot, Express.js

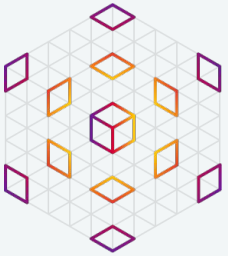
06

What's next?

Homework, next workshops



Agenda



01

Introduction

02

Microservices

Why with Apache Cassandra?

03

REST APIs

04

Sensor Application

Drivers; via command-line

05

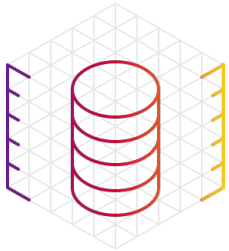
Sensor REST API

FastAPI, Spring Boot, Express.js

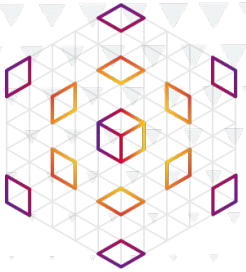
06

What's next?

Homework, next workshops



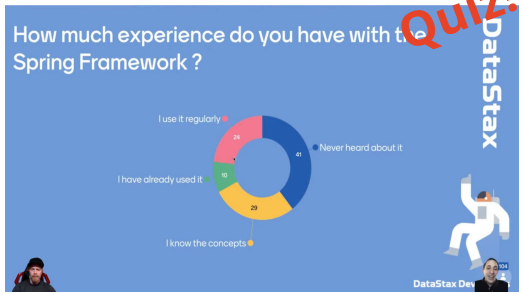
❯ Live & Interactive



youtube.com/DataStaxDevs

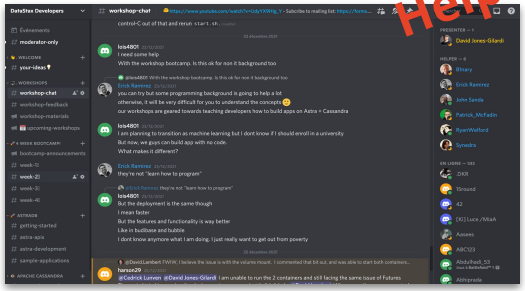
YouTube

⌨ !menti



Mentimeter

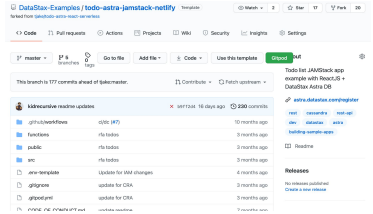
⌨ !discord



Discord (#workshop-chat)

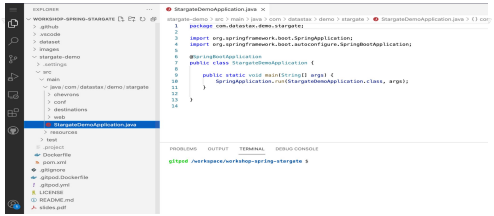
Hands-on housekeeping

Source code + exercises + slides



GitHub !github

IDE

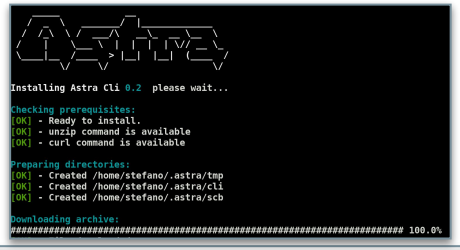


Gitpod !gitpod

Database + Api + Streaming

DATASTAX ASTRA DB !astra

Command-line database management

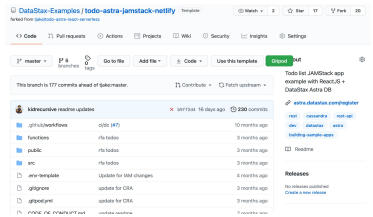


astra-cli

Nothing to install !

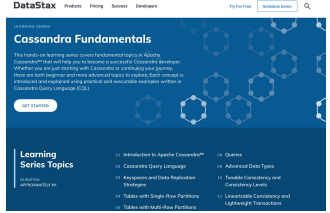
❯ Practice, Labs, Assignments & more

Instructions & practice



GitHub !github

Learning resources



datastax.com/dev

DATASTAX ACADEMY

academy.datastax.com

We sponsor your learning!

❯ Assignment ⇒ Badge

API and Microservices with Cassandra Homework

Welcome and thank you!
Here you can submit your homework for the DataStax Developers "Api a
with Cassandra" workshop.
In case of any questions please contact the organizers at https://dtsx
https://dtsx.io/cedrick or just send an email to aleksandr.volochnev

- Workshop materials: https://github.com/datastaxdevs/worksho
development#readme
- Discord chat: https://dtsx.io/discord

Email *

Your email

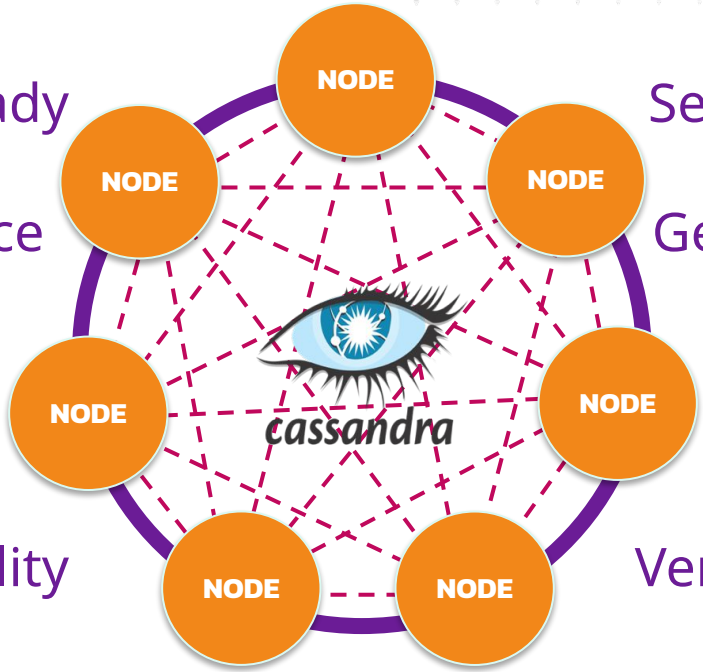
Intro to Apache Cassandra

Big Data Ready

Read/Write Performance

Linear Scalability

Highest Availability

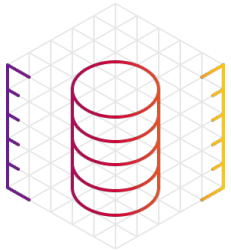


Self-Healing, Automation

Geographical Distribution

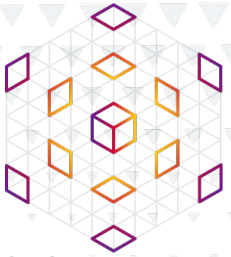
Platform-agnostic

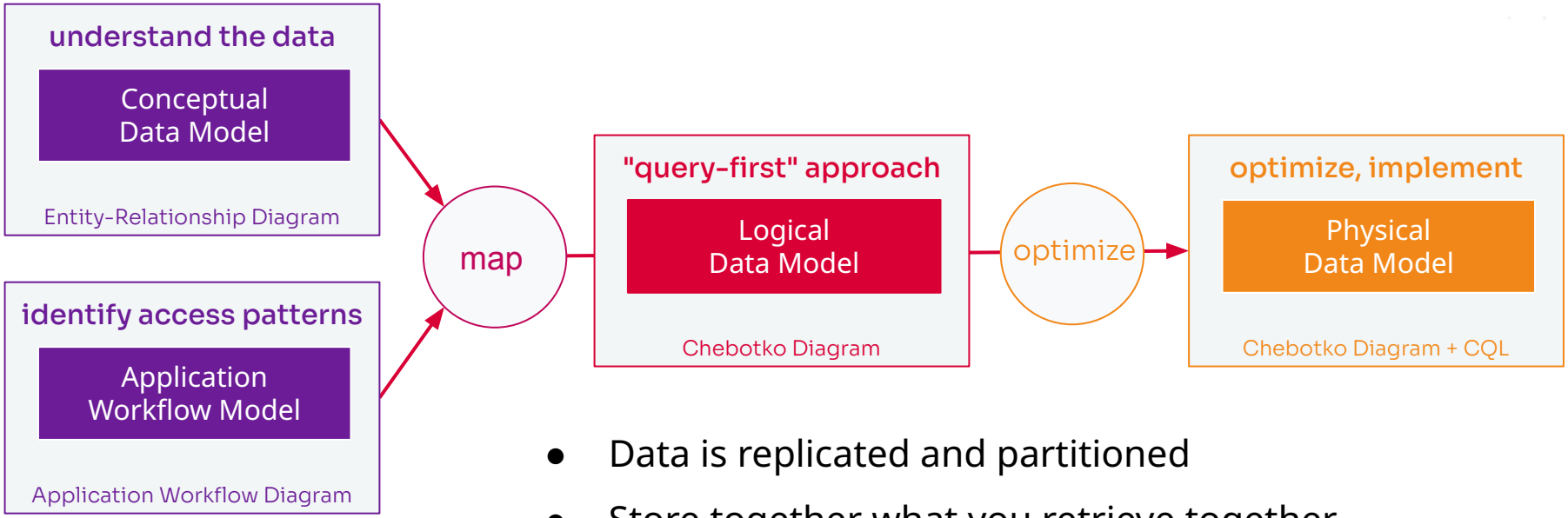
Vendor-independent



Data Modeling with Cassandra

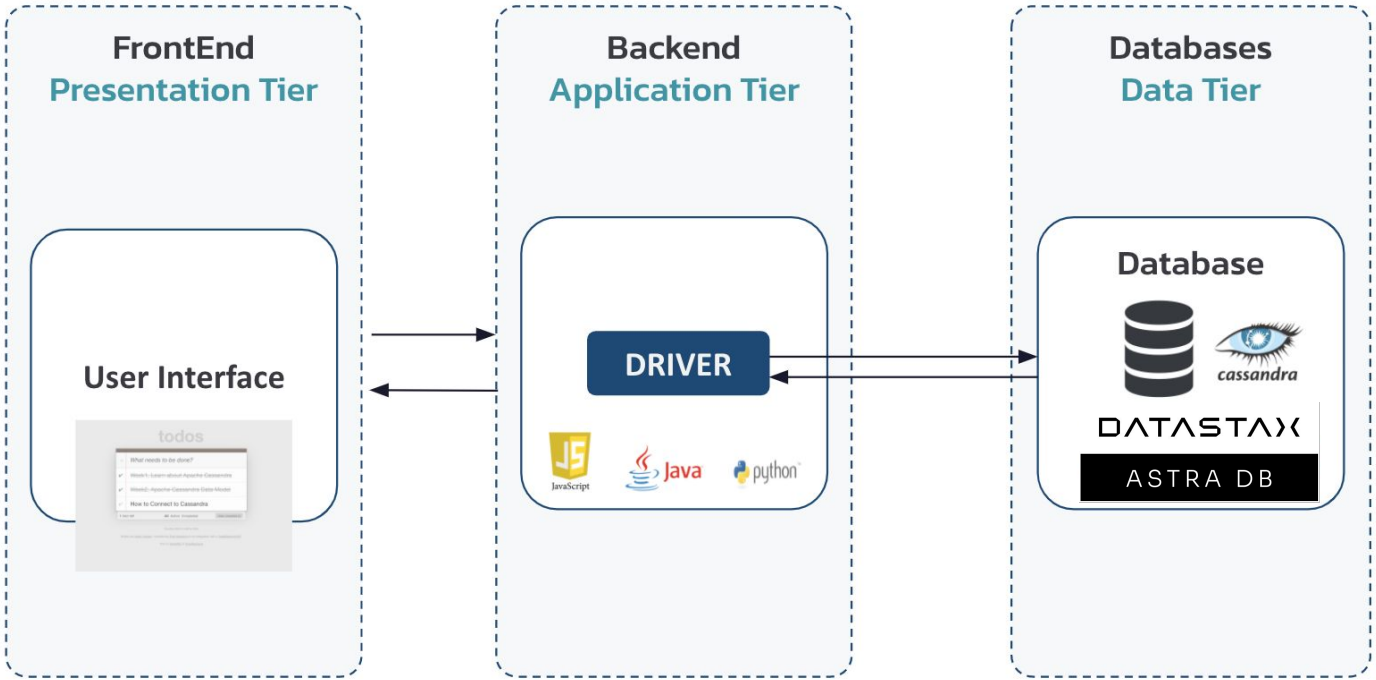
"From queries to tables"

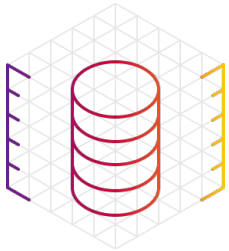




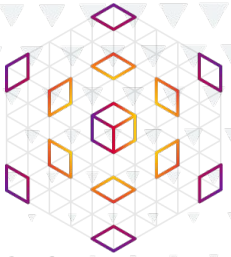
- Data is replicated and partitioned
- Store together what you retrieve together
- Avoid big/hot partitions

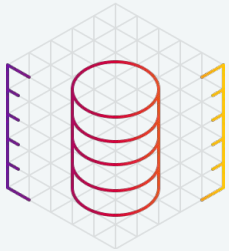
App development with Cassandra



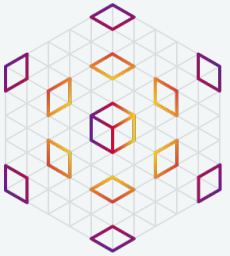


Pick your language





Agenda



01

Introduction

02

Microservices

Why with Apache Cassandra?

03

REST APIs

04

Sensor Application

Drivers; via command-line

05

Sensor REST API

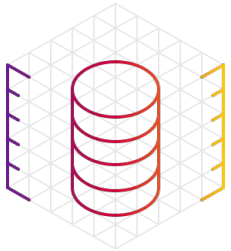
FastAPI, Spring Boot, Express.js

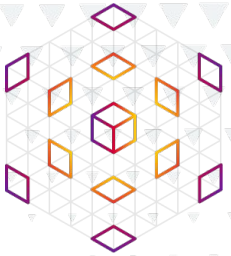
06

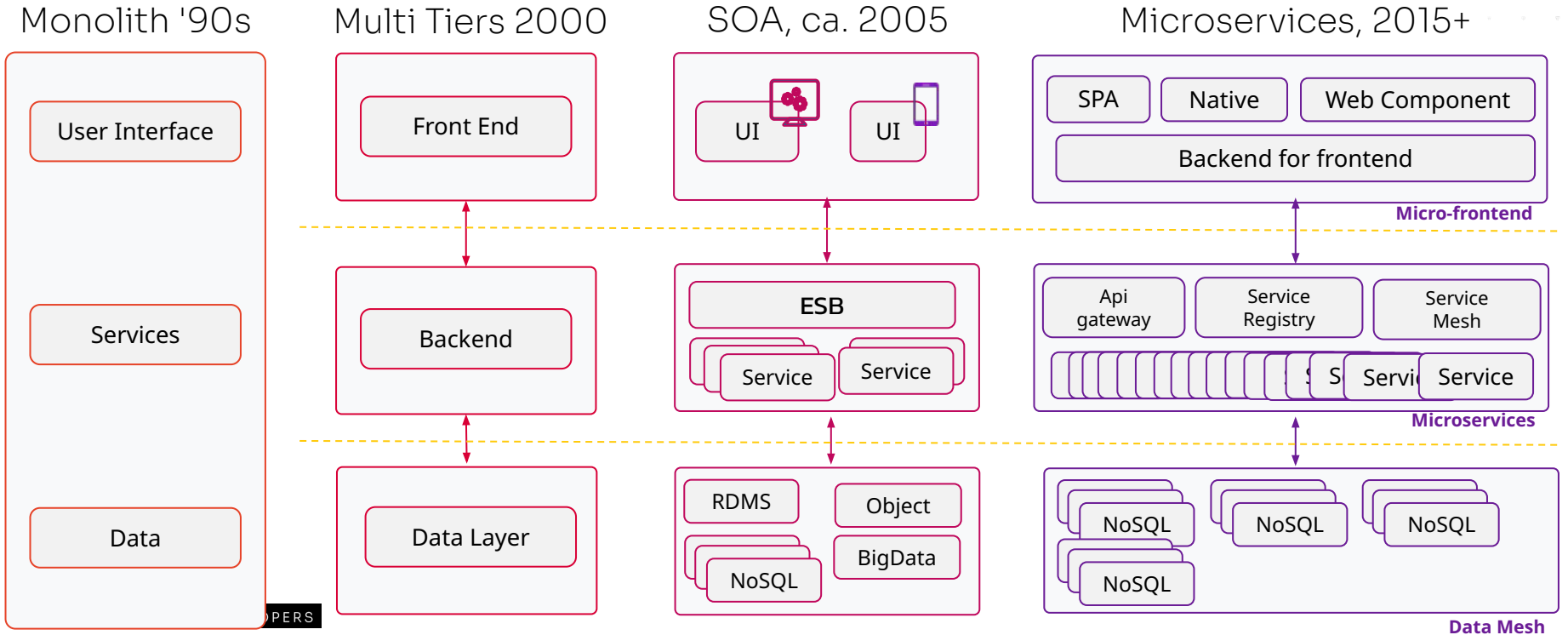
What's next?

Homework, next workshops

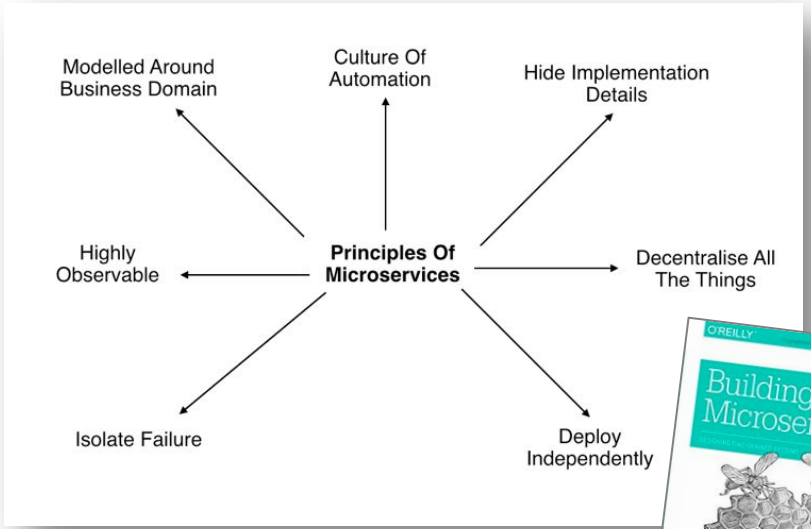
The road to Microservices

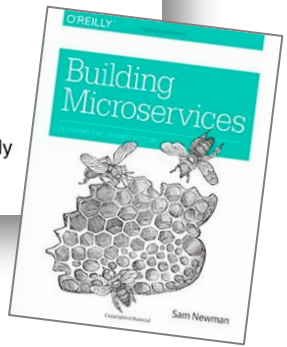




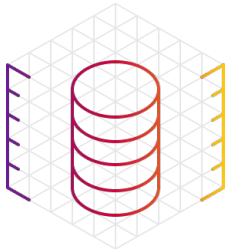


Microservices, principles

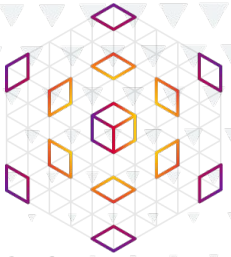




Organized around Business Capabilities
Products not Projects
Smart endpoints and dumb pipes
Decentralized Governance
Decentralized Data Management
Infrastructure Automation
Design for failure
Evolutionary Design

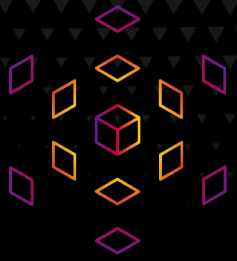


Microservices, pros/cons



Cost reduction (scaling, design)
Risk reduction (resilience)
Release speed increase
Better visibility (security, monitoring)

Complexity (security, transaction, orchestration)
Cultural changes
Bigger run footprint

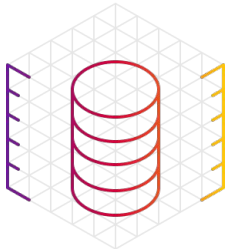


"Share nothing" IRL

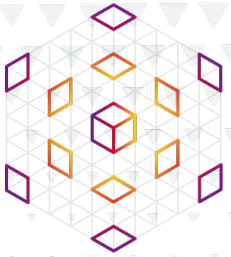
in other words:

"Do you really install
a separate DB for each service?"

Neither do I.



❯ ACID ⇒ BASE

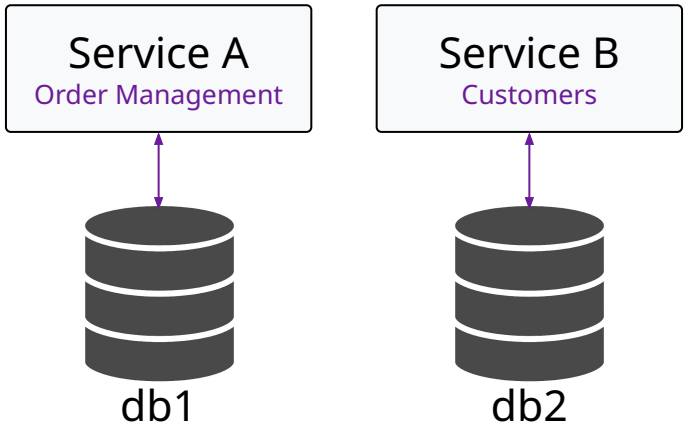


Atomicity

Consistency

Isolation

Durability



Basic

Availability

Soft state

Eventual consistency

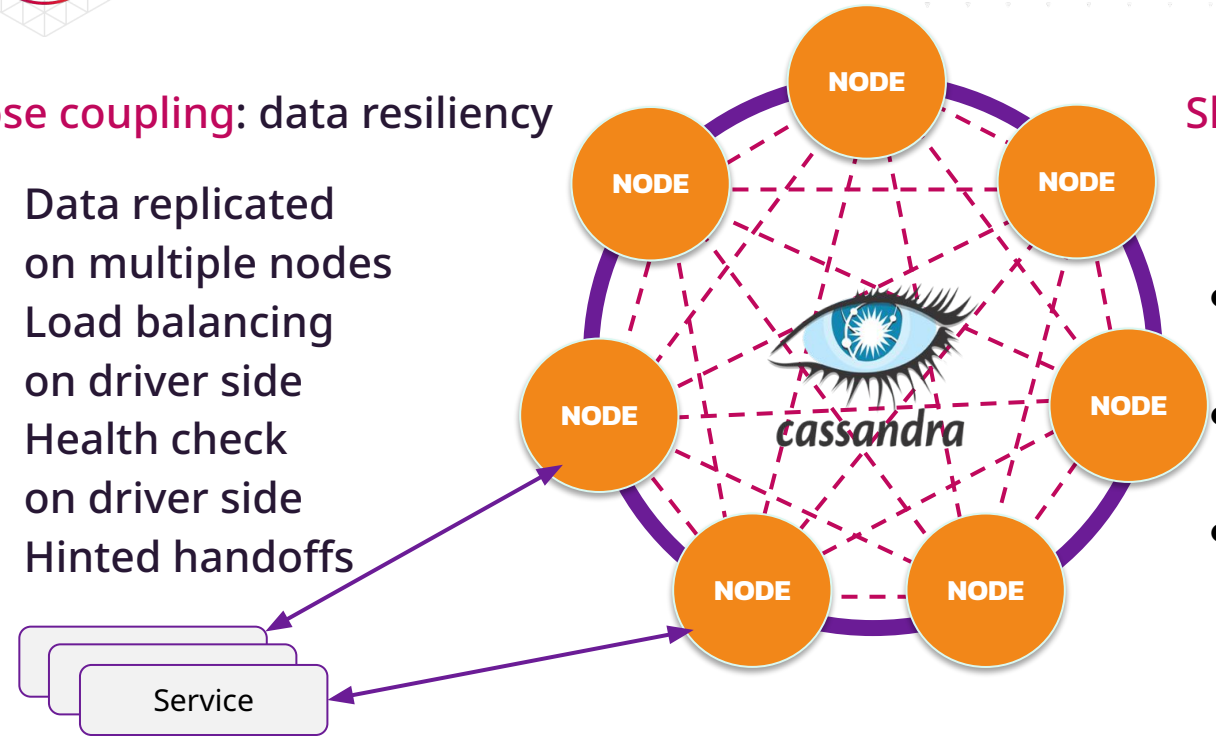
ACID:
❌ no distributed transactions across services

BASE:
✅ "AP" systems with idempotence
✅ event sourcing, CQRS

❯ Cassandra 💘 microservices

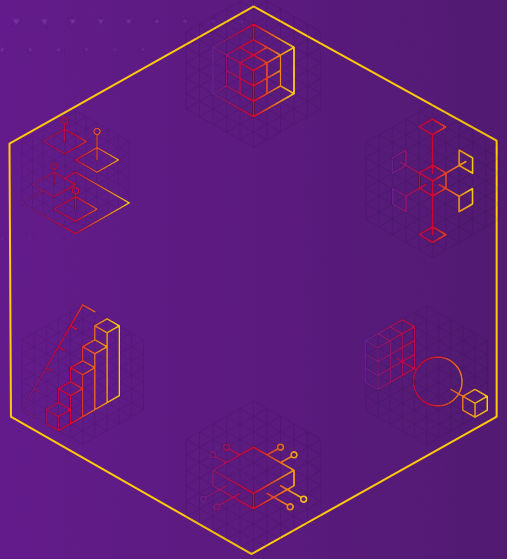
Loose coupling: data resiliency

- Data replicated on multiple nodes
- Load balancing on driver side
- Health check on driver side
- Hinted handoffs



Shared nothing: data isolation

- Per keyspace (with replication)
- Per table (1 query = 1 table)
- Per profile (RBAC)

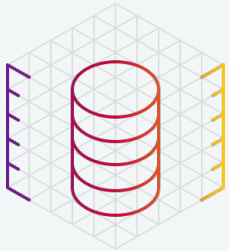


Lab 1

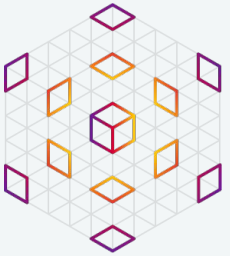
Create your Astra DB instance

github.com/datastaxdevs/workshop-cassandra-application-development

- ❑ Create DB / resume if "hibernated"
- ❑ Create schema & populate tables



Agenda



01

Introduction

02

Microservices

Why with Apache Cassandra?

03

REST APIs

04

Sensor Application

Drivers; via command-line

05

Sensor REST API

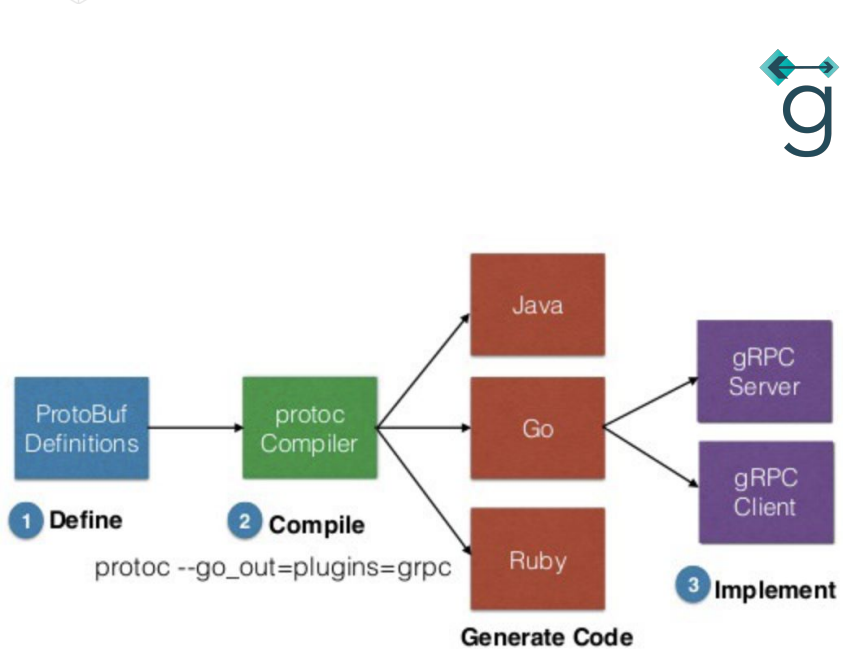
FastAPI, Spring Boot, Express.js

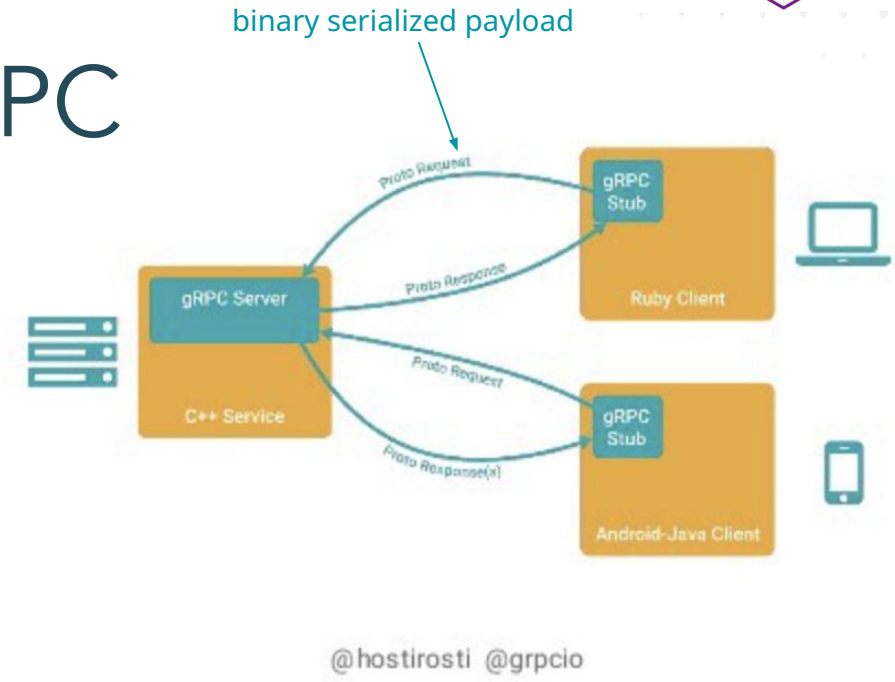
06

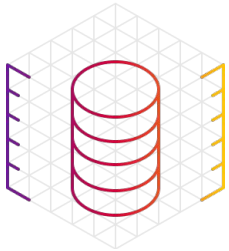
What's next?

Homework, next workshops

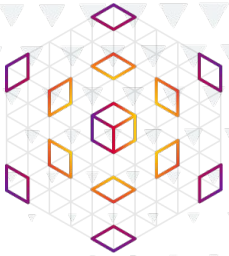
❯ Google Remote Procedure Call







› Graph Query Language



GraphQL

Describe your data

schema snippet

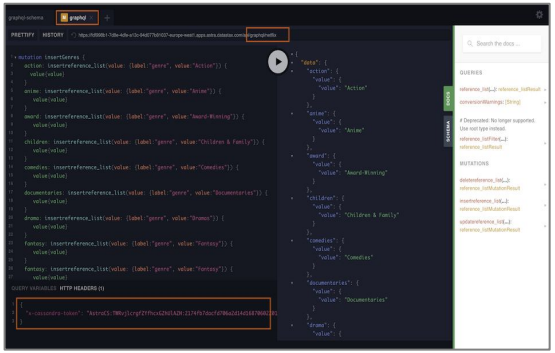
```
type Query {
      shows: [Show]
}

type Show {
    title: String!
    blurb: String
    actors: [Actor]
    releaseYear: Int
}
```

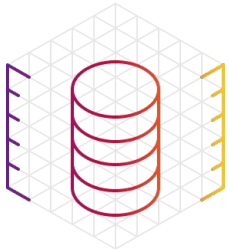
Ask for what you need

query sample

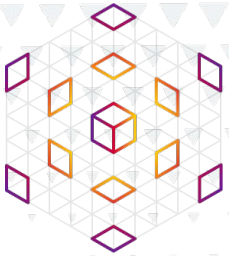
```
query getShows {
     shows {
          title
          releaseYear
     }
}
```

Discoverability





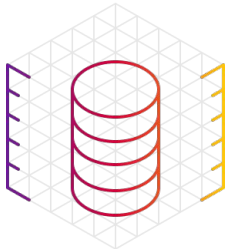
Representational state transfer



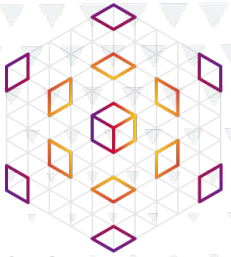
{ REST }

Todos Implement CRUD operations for Todo Tasks

GET	/api/v1/todos/	Retrieve the complete list of Taskss
POST	/api/v1/todos/	Create a new task
DELETE	/api/v1/todos/	Delete all tasks in one go
GET	/api/v1/todos/{taskId}	Get details of a task if exists
DELETE	/api/v1/todos/{taskId}	Delete a task from its id of exists
PATCH	/api/v1/todos/{taskId}	Update an existing task



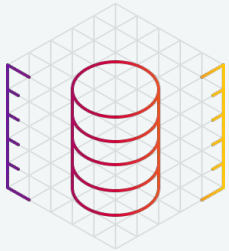
REST, pros/cons



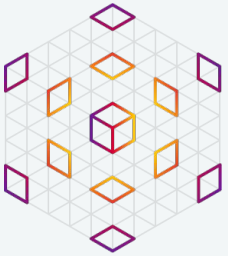
- Client/server decoupling (*schema-on-read*)
- Api lifecycle (*versioning*)
- Tooling (*API management, serverless*)

- Verbose payloads (*json, xml*)
- No discoverability
- Not suitable for command-like (functions) API

- CRUD superstar
- Relevant for mutations (OLTP)
- Public and web APIs
- Limited business scope



Agenda



01

Introduction

02

Microservices

Why with Apache Cassandra?

03

REST APIs

04

Sensor Application

Drivers; via command-line

05

Sensor REST API

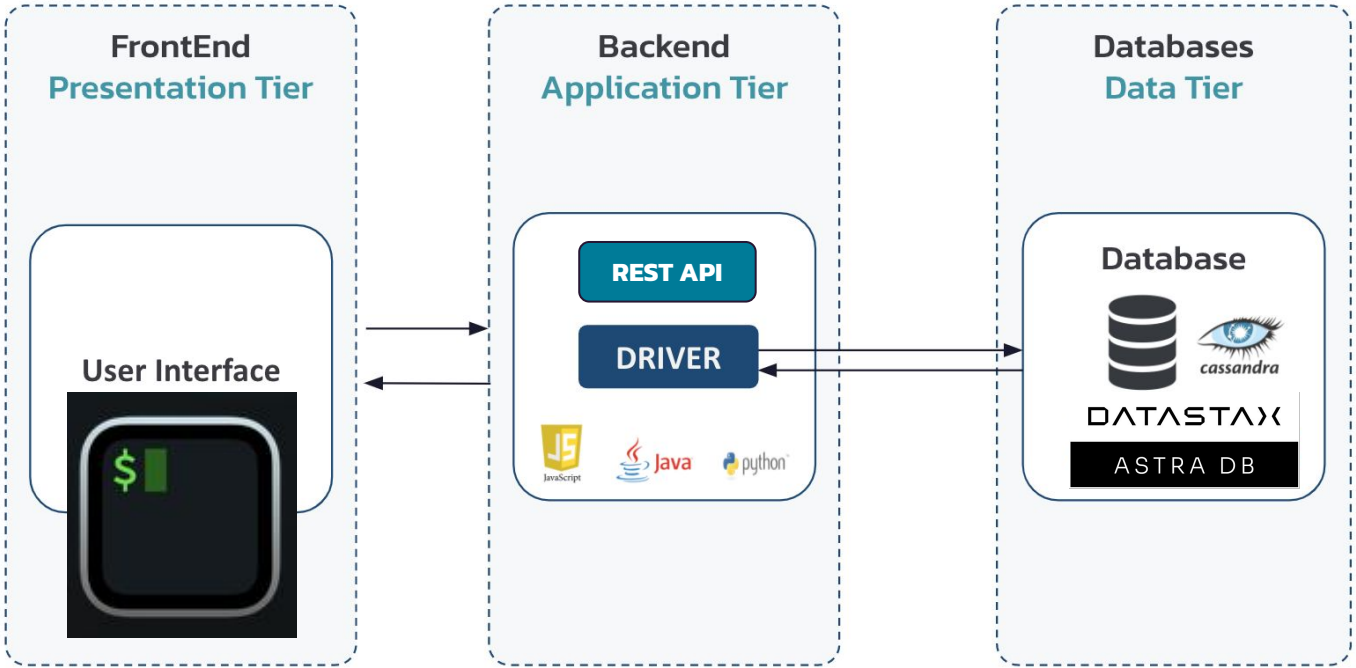
FastAPI, Spring Boot, Express.js

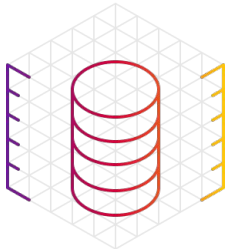
06

What's next?

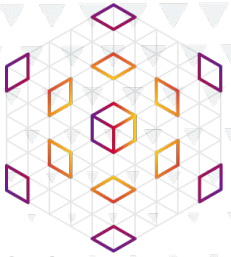
Homework, next workshops

Today's microservice





❯ Did someone say "Drivers"?








Connectivity

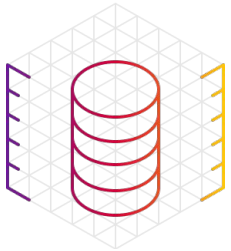
- ★ Token & Datacenter Aware
- ★ Load Balancing Policies
- ★ Retry Policies
- ★ Reconnection Policies
- ★ Connection Pooling
- ★ Health Checks
- ★ Authentication | Authorization
- ★ SSL

Query

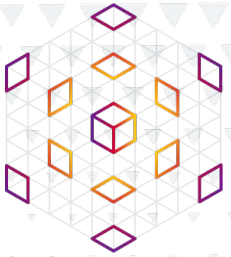
- ★ CQL Support
- ★ Schema Management
- ★ Sync/Async/Reactive API
- ★ Query Builder
- ★ Compression
- ★ Paging

Parsing Results

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



Installation



Maven

```
<dependency>
    <groupId>com.datastax.oss</groupId>
    <artifactId>java-driver-core</artifactId>
    <version>4.13.1</version>
</dependency>
```

Java

4.6.3 npm

```
npm install cassandra-driver
```

```
{
  "dependencies": {
      "cassandra-driver": "^4.6.3"
  }
}
```

JavaScript

python Package Index

```
pip install cassandra-driver==3.25.0
```

python

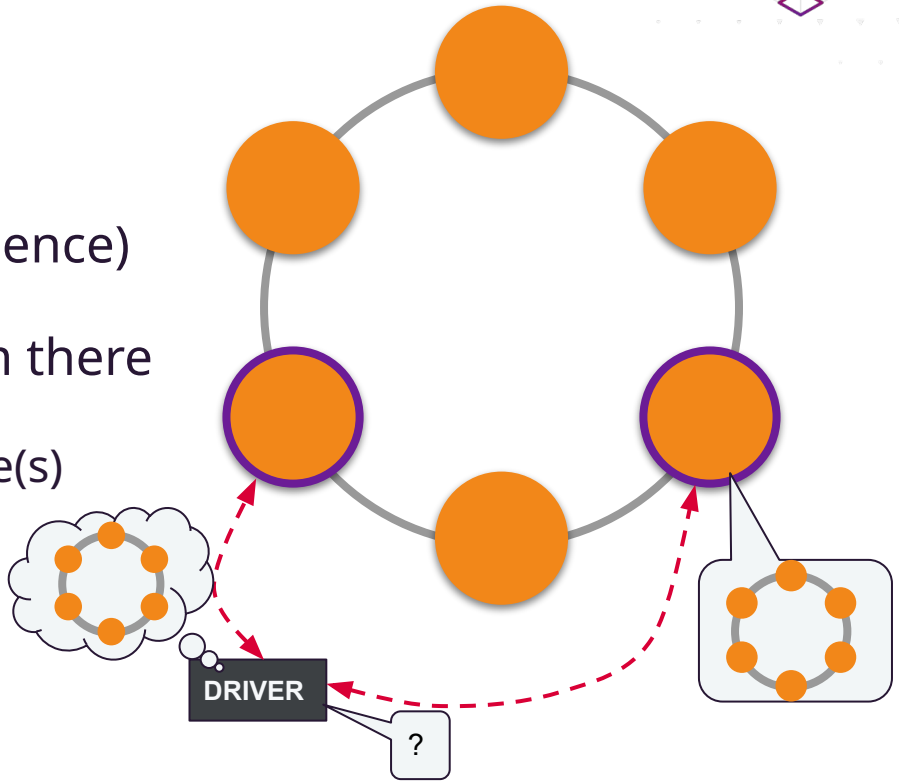
nuget v3.15.0

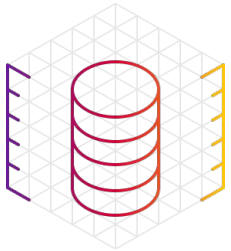
```
Install-Package CassandraCSharpDriver -Version 3.15.0
```

C#

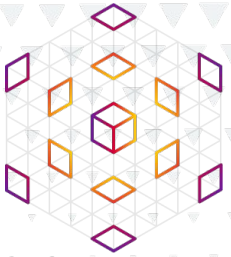
❯ Contact point (a Cassandra node)

- Specify *at least* one **contact point**
- Suggested: **3 per datacenter** (resilience)
- **Cluster** discovery is automatic from there
- Queries "magically" routed to best node(s)





❯ Create Session (contact points)



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

Java

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

node.js

```python
auth_provider = PlainTextAuthProvider(
   username='U', password='P')
cluster = Cluster(['127.0.0.1'],
  auth_provider=auth_provider, protocol_version=5)
session = cluster.connect('sensor_data')
```

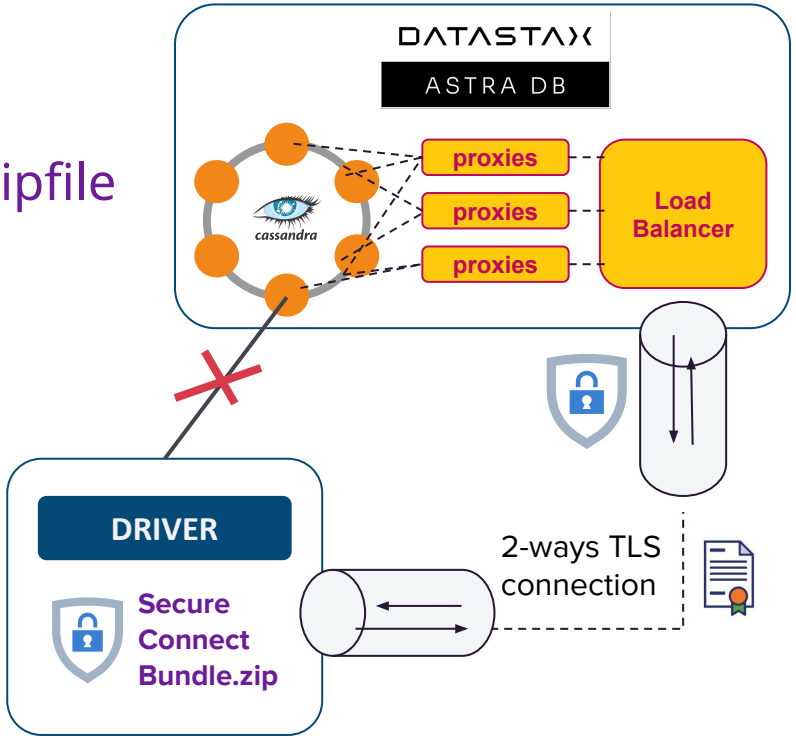
python

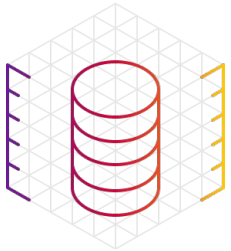
```dotnet
Cluster cluster = Cluster.Builder()
.AddContactPoint("127.0.0.1")
.WithCredentials("U", "P")
.Build();
session = cluster.Connect("sensor_data");
```

C#

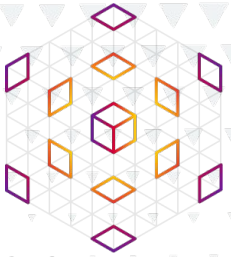
Connecting to Astra DB - architecture

- Secure connection over HTTP required
- Must provide secure-connect-bundle zipfile
- Credentials in your "DB Token":
  - `clientId/clientSecret`
  - or: `"token"/"AstraCS:..."`
- No SPOF (single point of failure)





❯ Create Session (Astra DB)



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

Java

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

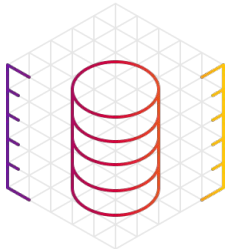
node

```
auth_provider = PlainTextAuthProvider(
   username='U', password='P')
cluster = Cluster(
   cloud ={'secure_connect_bundle': 'secure.zip'},
   auth_provider=auth_provider, protocol_version=4)
session= cluster.connect('sensor_data')
```

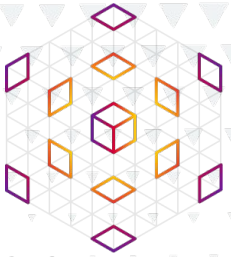
python

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

C#

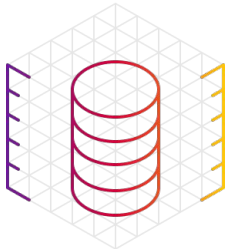


❯ the Session object

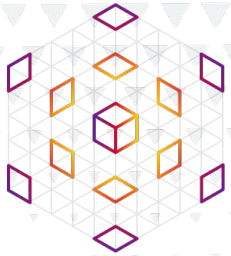


- **Stateful** object handling communications with each node
- Should be **unique** in the Application (*Singleton*)
- Should be **closed** at application shutdown (*shutdown hook*) in order to free TCP sockets

```
Java:       cqlSession.close();
Python:     session.shutdown();
Node:       client.shutdown();
CSharp:     IDisposable
```



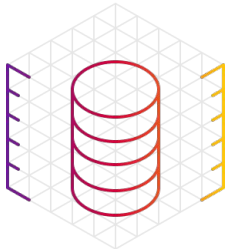
❯ Run CQL queries



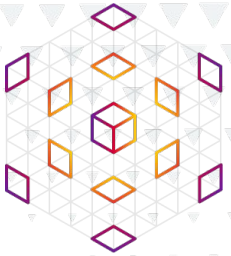
```
session.execute(
     "SELECT * FROM sensors_by_network WHERE network = %s;",
     (network,)
)
```

```
cqlSession.execute(
     "SELECT * FROM sensors_by_network WHERE network = '" + network + "'"
);
```

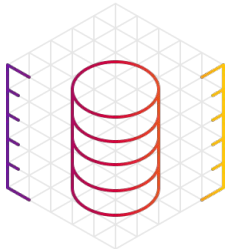
```
session.execute(
     "SELECT * FROM sensors_by_network WHERE network = ?;", [network]
).then( ......
```



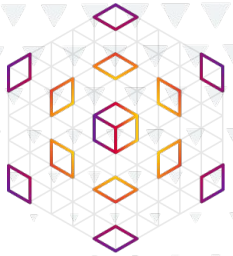
A Javascript note: type hints!



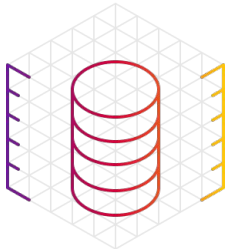
```
await client.execute(
  "INSERT INTO metals (kind, name, density) VALUES (?, ?, ?);",
  ['regular', 'palladium', 12.02],
  {
    hints: [
      'text',
      'text',
      'float'
    ]
  }
);
```



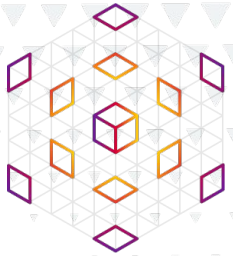
❯ Prepare your statements (1)



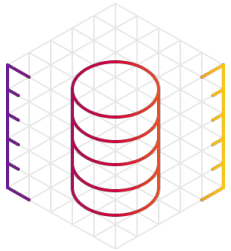
```
q3_statement = session.prepare(
    "SELECT * FROM sensors_by_network WHERE network = ?;"
)

rows = session.execute(q3_statement, (network,) )
```



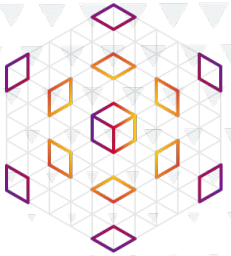
❯ Prepare your statements (2)



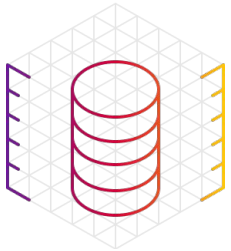
```
PreparedStatement q3Prepared = session.prepare(
    "SELECT * FROM sensors_by_network WHERE network = ?");

BoundStatement q3Bound = q3Prepared.bind(network);
ResultSet rs = session.execute(q3Bound);
```



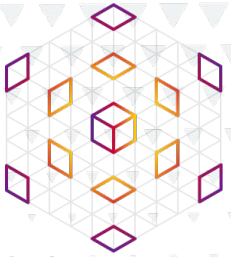
❯ Prepare your statements (3)



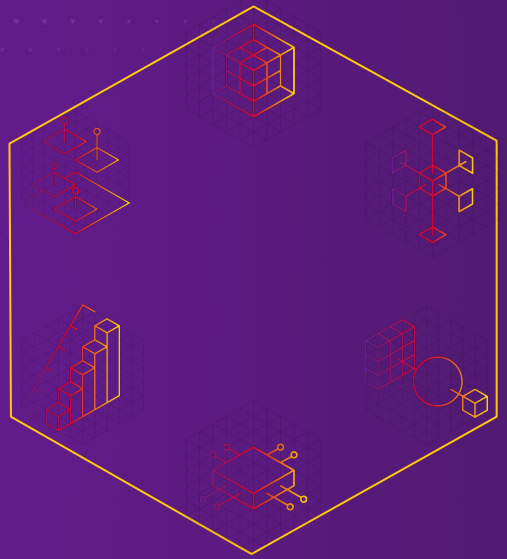
```
session.execute(
  "SELECT * FROM sensors_by_network WHERE network = ?;",
  [network],
  {prepare: true}
).then( ......
```



❯ Prepared statements, advantages



- Parse once, run many times
- Saves network trips for result set metadata
- Client-side type validation
- Statements binding on partition keys compute their own cluster routing

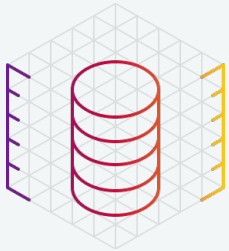


Lab 2

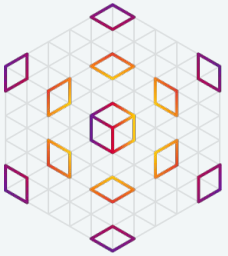
Read data from DB programmatically

`github.com/datastaxdevs/workshop-cassandra-application-development`

- ❑ Gitpod & setup DB connection parameters
- ❑ Run connectivity test
- ❑ Run queries programmatically
- ❑ Prepared statements



Agenda



01

Introduction

02

Microservices

Why with Apache Cassandra?

03

REST APIs

04

Sensor Application

Drivers; via command-line

05

Sensor REST API

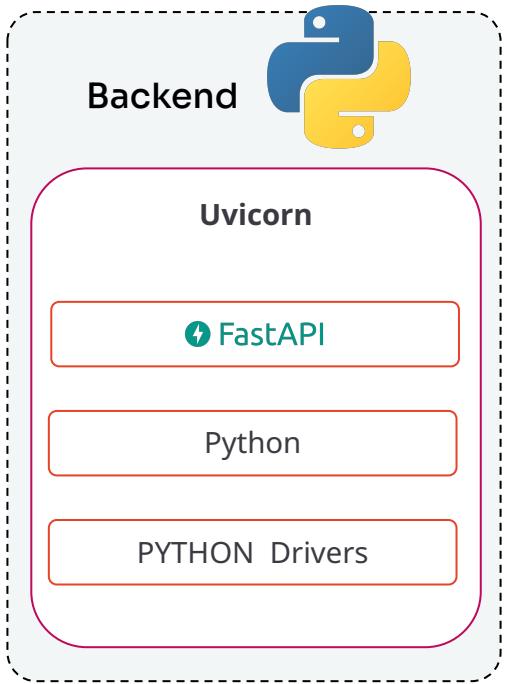
FastAPI, Spring Boot, Express.js

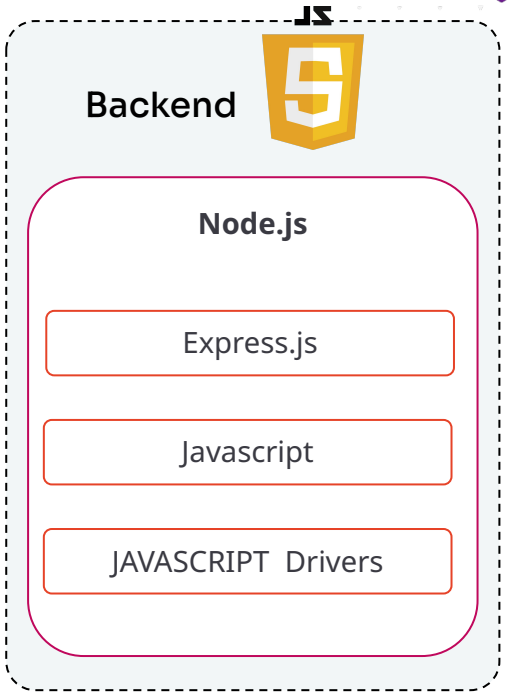
06

What's next?

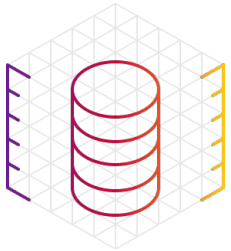
Homework, next workshops

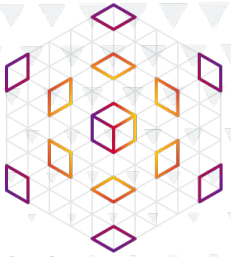
Technical stacks

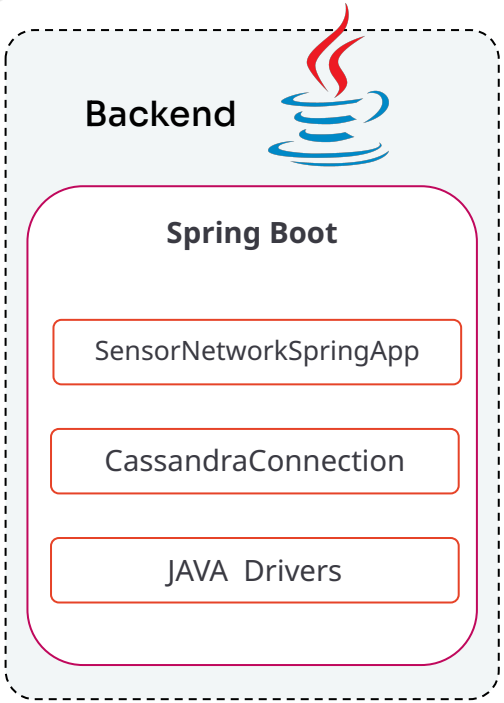


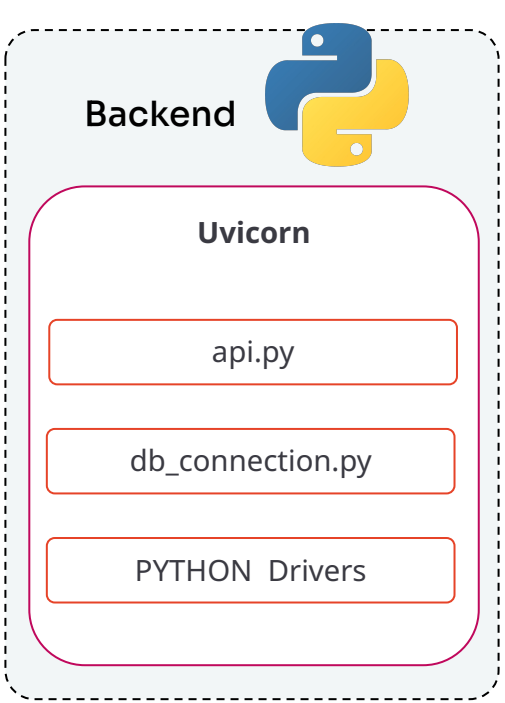


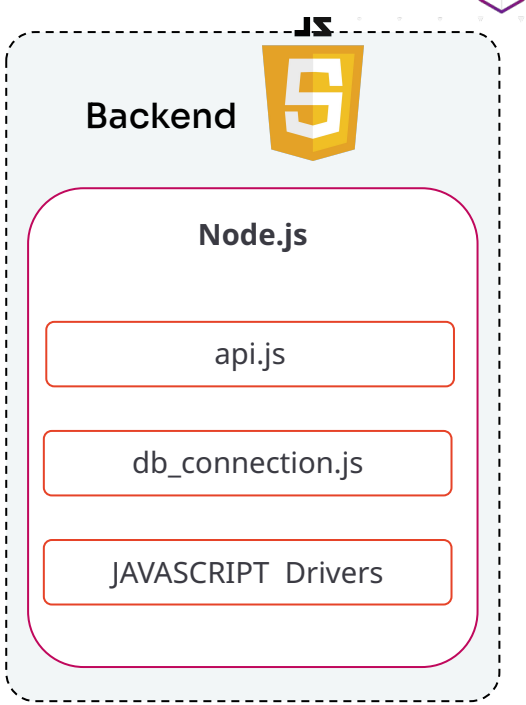
API structure/modules

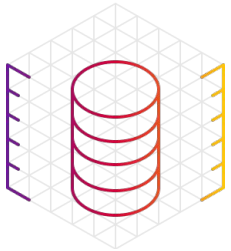




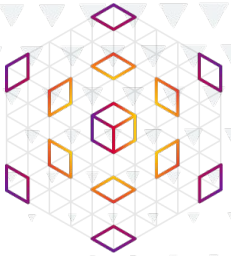








❯ REST API specs



Sensor API

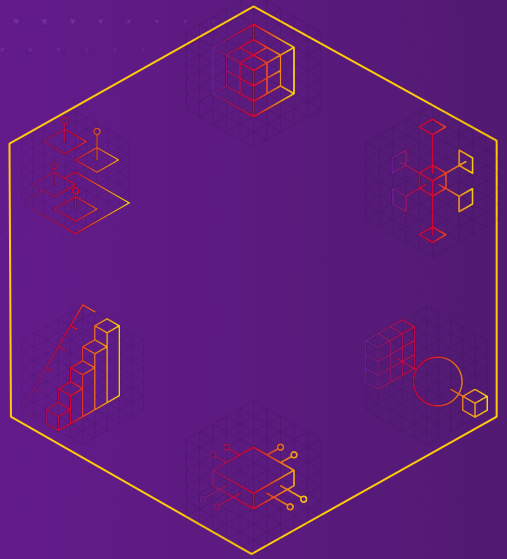
Two endpoints:

- Retrieve sensor data by network
- Retrieve measurements by date

GET	/sensors_by_network/{network}
POST	/measurements_by_sensor_date/

```
{"sensor":"s1001", "date":"2020-07-04"}
```

Wait a minute...
Why a POST?

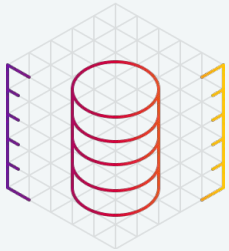


Lab 3

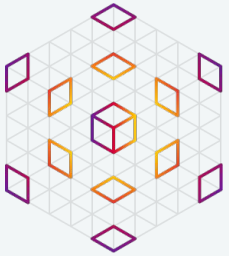
Sensor API

`github.com/datastaxdevs/workshop-cassandra-application-development`

- ❑ Run RESTful API
- ❑ Issue HTTP requests
- ❑ Inspect endpoints code



Agenda



01

Introduction

02

Microservices

Why with Apache Cassandra?

03

REST APIs

04

Sensor Application

Drivers; via command-line

05

Sensor REST API

FastAPI, Spring Boot, Express.js

06

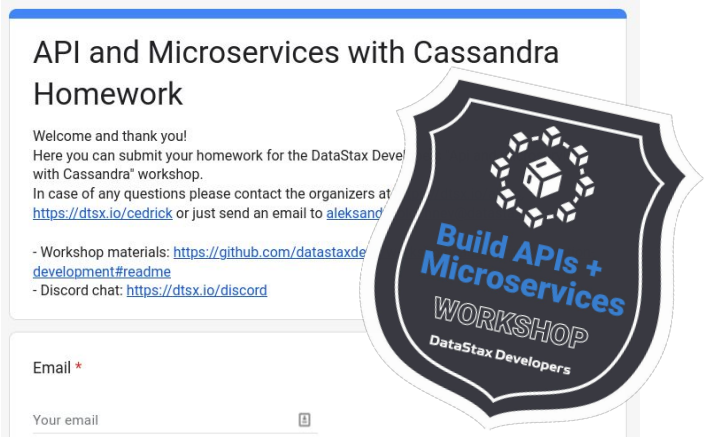
What's next?

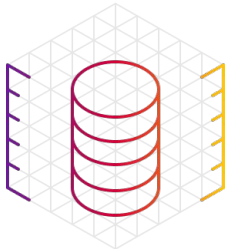
Homework, next workshops

❯ Assignment ⇒ Badge

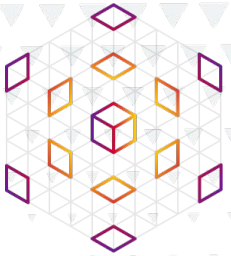
Coding exercise: enrich the API with a **new GET endpoint** for Q1 ("get all networks")

Full instructions in the Github repo





❯ You are not alone in your journey



Discord: **dtsx.io/discord**

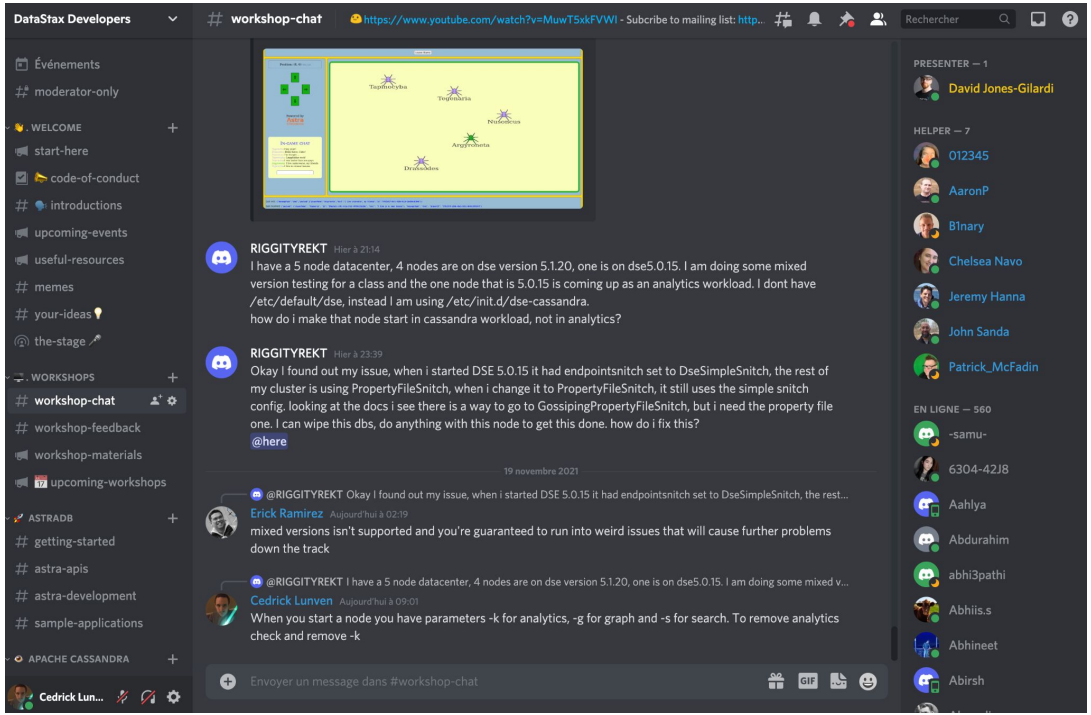
StackOverflow(*): **stackoverflow.com/questions/tagged/cassandra**

DBA Stack Exchange(*): **dba.stackexchange.com/questions/tagged/cassandra**

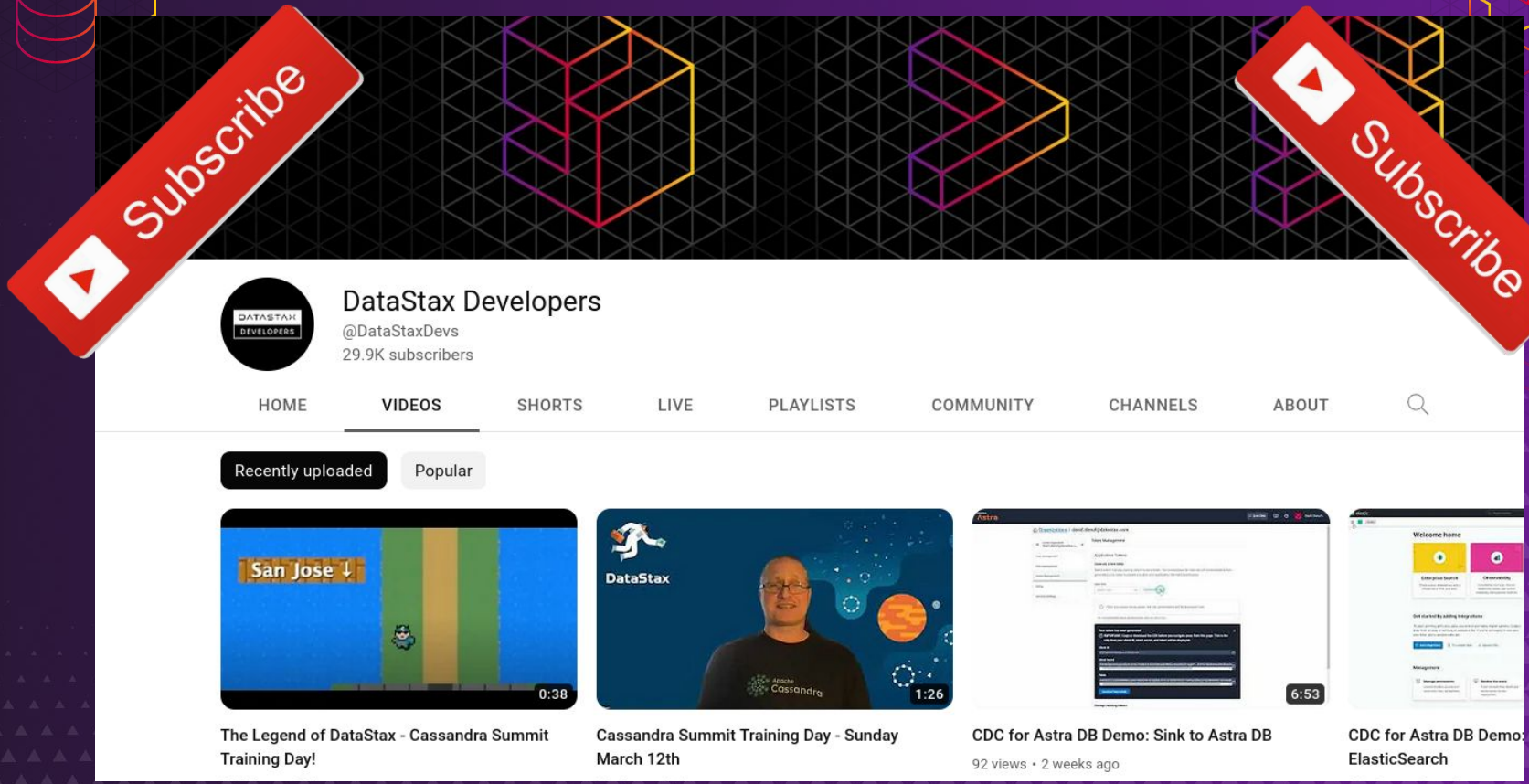
(*) For best results, follow the cassandra tag

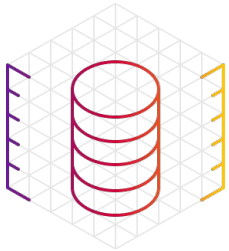
❯ Our Discord community (18k+)

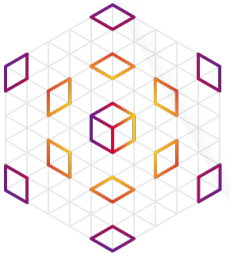
⌨ !discord



dtsx.io/discord


Subscribe
Subscribe
DataStax Developers
@DataStaxDevs
29.9K subscribers
HOME
VIDEOS
SHORTS
LIVE
PLAYLISTS
COMMUNITY
CHANNELS
ABOUT
Recently uploaded
Popular
San Jose
0:38
The Legend of DataStax - Cassandra Summit Training Day!
DataStax
Apache Cassandra
1:26
Cassandra Summit Training Day - Sunday March 12th
6:53
CDC for Astra DB Demo: Sink to Astra DB
92 views • 2 weeks ago
Welcome home
CDC for Astra DB Demo:
ElasticSearch





Thank You