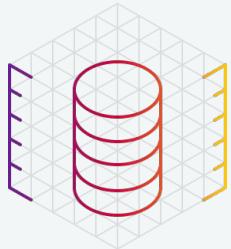




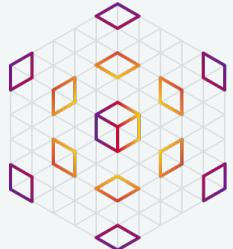
# Application Development Bootcamp Week III: Building the App!

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

---

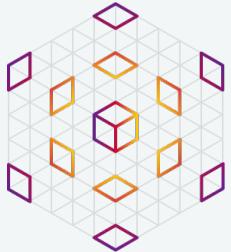
**06**

**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**

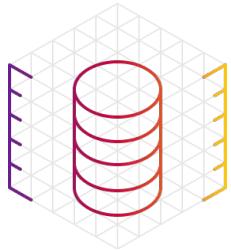
FastAPI, Spring Boot, Express.js

**06**

**What's next?**

Homework, next workshops

# » Live & Interactive



[youtube.com/DataStaxDevs](https://youtube.com/DataStaxDevs)

Live!

Agenda

<b>01</b> PetClinic Architecture & Use Case	<b>02</b> DataStax Astra Cassandra Database The Art of Data Modeling	<b>03</b> Reactive Drivers Reactive vs Async
<b>04</b> Spring Reactive Boot and WebFlux	<b>05</b> User Interface Angular	<b>06</b> Game & Resources

DataStax Developers



YouTube

!menti

How much experience do you have with the **Spring Framework**?

Quiz!

Experience Level	Percentage
Never heard about it.	41%
I know the concepts.	24%
I have already used it.	10%
I use it regularly.	25%

DataStax Developers



Mentimeter

!discord

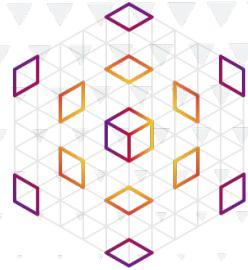
Help!

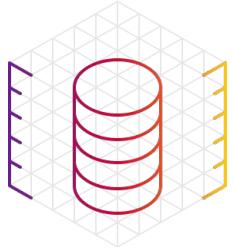
Discord Developers

#workshop-chat



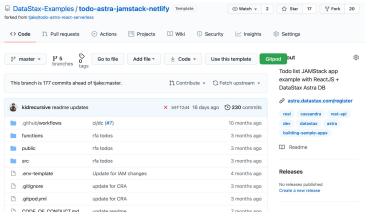
Discord (#workshop-chat)





# ► Hands-on housekeeping

## Source code + exercises + slides



 GitHub  !github

# Database + Api + Streaming



DATASTAX  
ASTRA DB

 lastra

IDE



The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure for "stargate-spring-startergate".
- Run Configuration:** A "StargateDemoApplication" configuration is selected, showing the main class as "StargateDemoApplication" and the arguments as "-Dspring.profiles.active=demo -Dstargate".
- Java Persistence API (JPA) Configuration:** A "stargate-spring-startergate" configuration is also visible.
- Terminal:** The terminal output shows the command "stargate /workspace/workshop-spring-startergate 5".

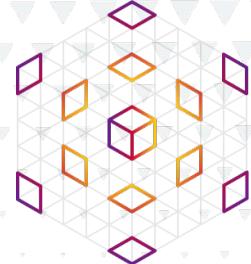
 Gitpod  !gitpod

# Command-line database management



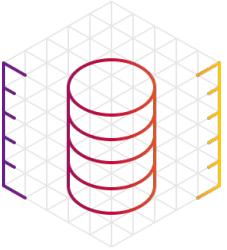
## astra-cli

DATASTAX DEVELOPERS



A white smartphone is shown from a top-down perspective, displaying a vibrant, diagonal banner across its screen. The banner has a gradient background transitioning from orange at the top to purple at the bottom. It features the text "Nothing to install !" in a white, sans-serif font, with the exclamation mark being significantly larger than the rest of the text. The phone is positioned on the left side of the frame, with its screen facing towards the right.

# › Practice, Labs, Assignments & more



## Instructions & practice

A screenshot of a GitHub repository page for 'DataStax-Examples/todo-astra-jemstack-netlify'. The page shows a list of files and commits. Key commits include 'Add reactive reader updates' (16 days ago), 'Update for VM changes' (4 months ago), and 'update for CRA' (2 months ago). The repository has 2300 commits and 47 pull requests.

 GitHub  !github

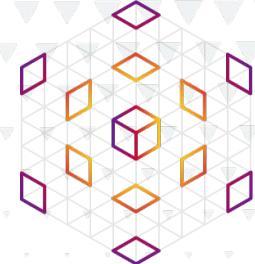
## Learning resources

A screenshot of the 'Cassandra Fundamentals' learning series page on datastax.com/dev. It features a dark blue header with the title 'Cassandra Fundamentals' and a sub-section 'Learning Series Topics'. Below the header, there are several small cards with topics like 'Introduction to Apache Cassandra™', 'Cassandra Query Language', 'Replicating Data Replication Strategies', 'Tables with Single-Row Partitions', and 'Linearizable Consistency and Partition Tolerance'.

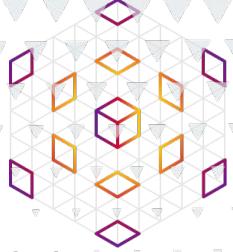
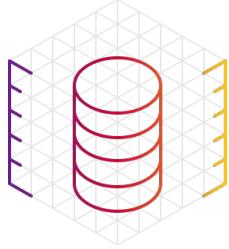
[datastax.com/dev](https://datastax.com/dev)



[academy.datastax.com](https://academy.datastax.com)



# 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.io/cedrick> or just send an email to [aleksandr.volochnev@datastax.com](mailto:aleksandr.volochnev@datastax.com)

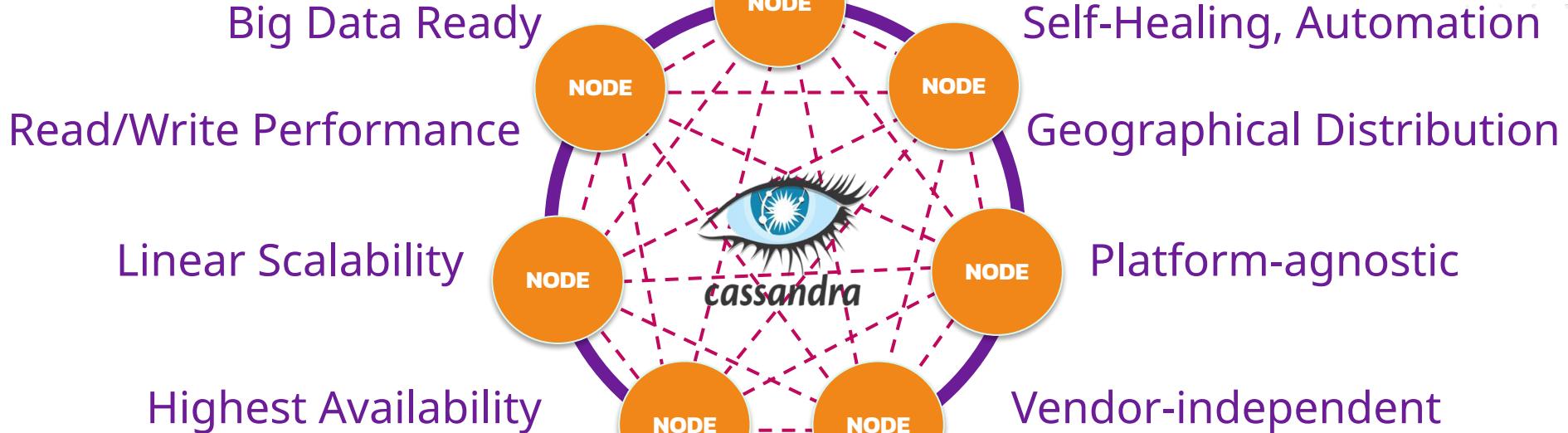
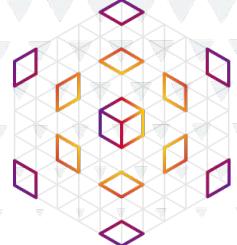
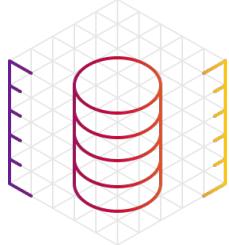
- Workshop materials: <https://github.com/datastaxdevs/workshop-development#readme>
- Discord chat: <https://dtsx.io/discord>

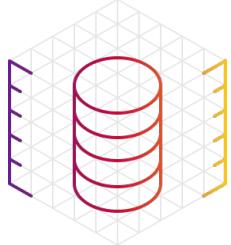
Email \*

Your email



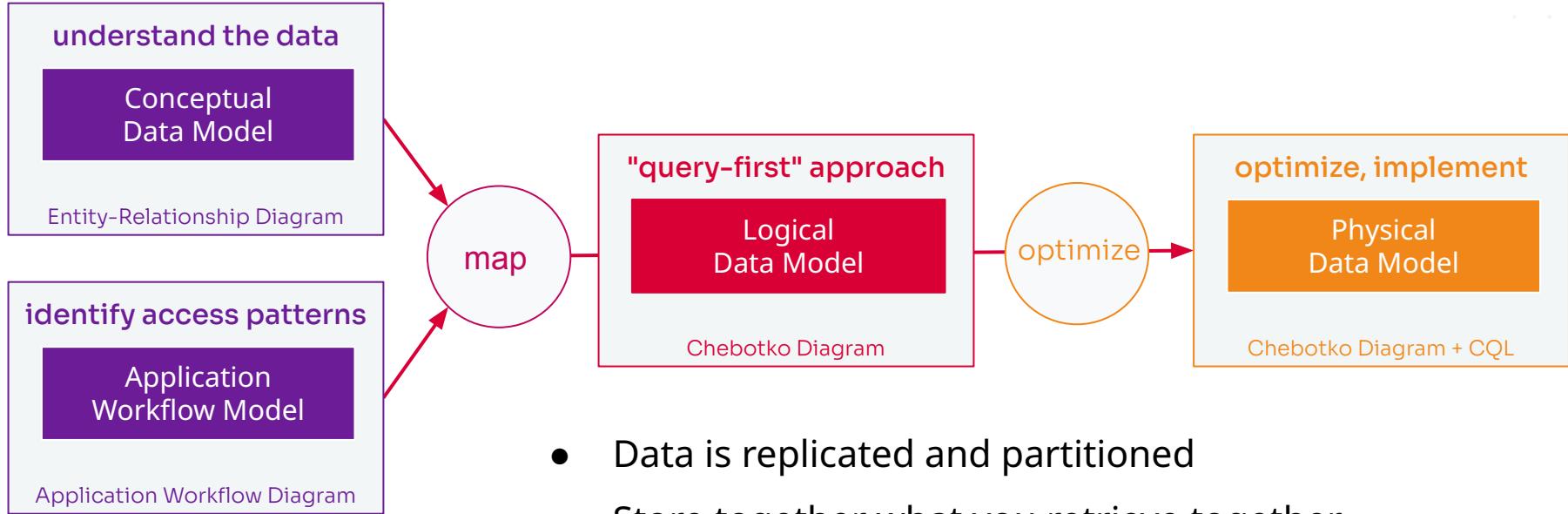
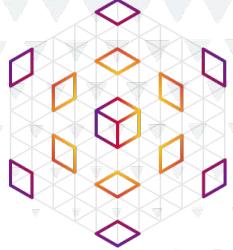
# › Intro to Apache Cassandra



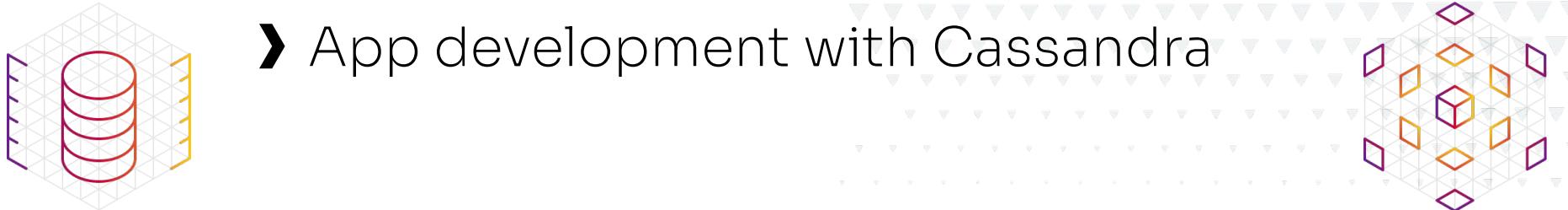


# » Data Modeling with Cassandra

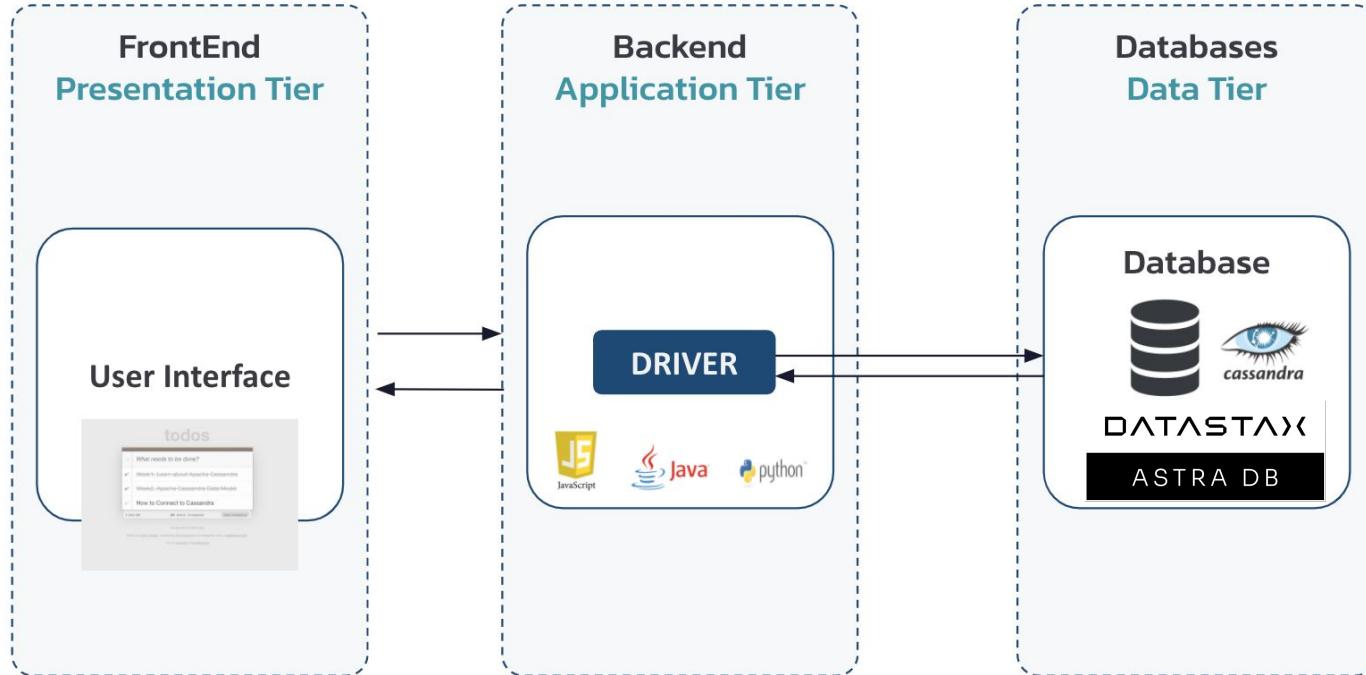
"From queries to tables"

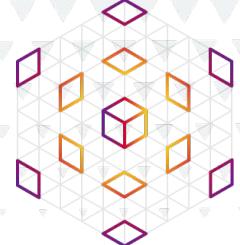
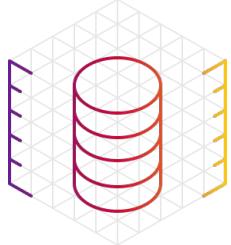


- Data is replicated and partitioned
- Store together what you retrieve together
- Avoid big/hot partitions

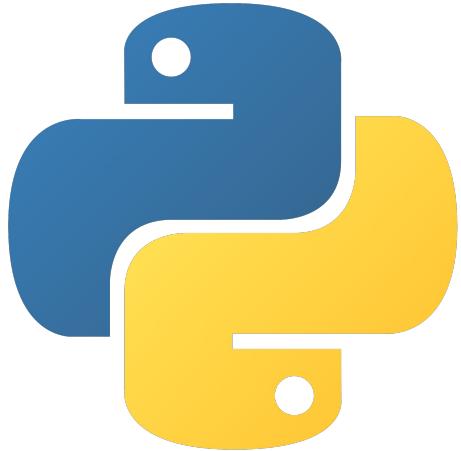


# App development with Cassandra



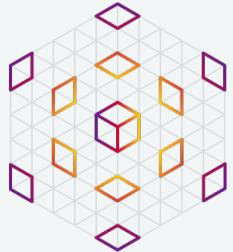


› Pick your language





# » Agenda



**01**

**Introduction**

**02**

**Microservices**

Why with Apache Cassandra?

**03**

**REST APIs**

**04**

**Sensor Application**

Drivers; via command-line

**05**

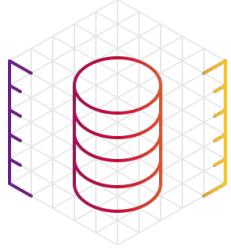
**Sensor REST API**

FastAPI, Spring Boot, Express.js

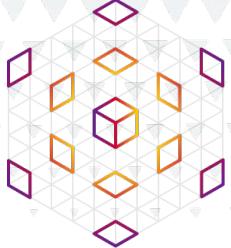
**06**

**What's next?**

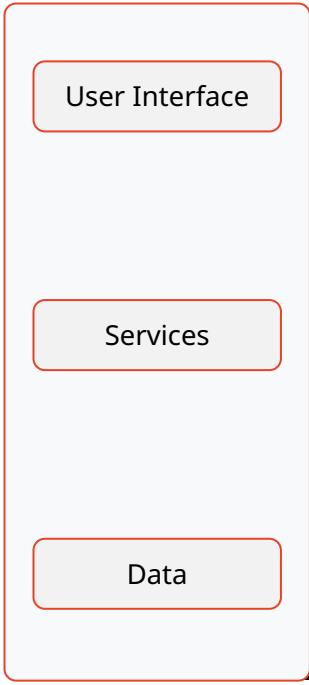
Homework, next workshops



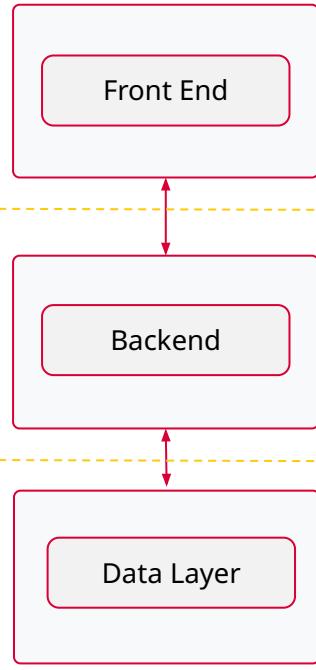
# The road to Microservices



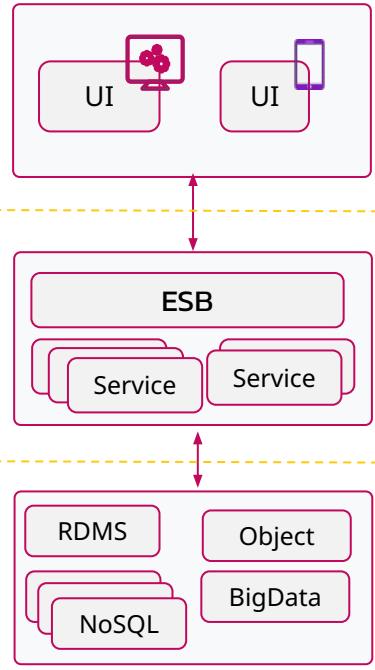
Monolith '90s



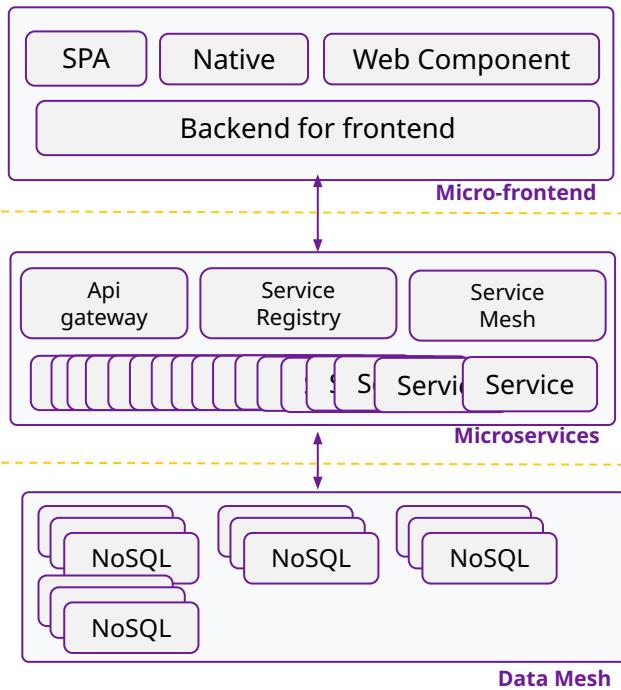
Multi Tiers 2000



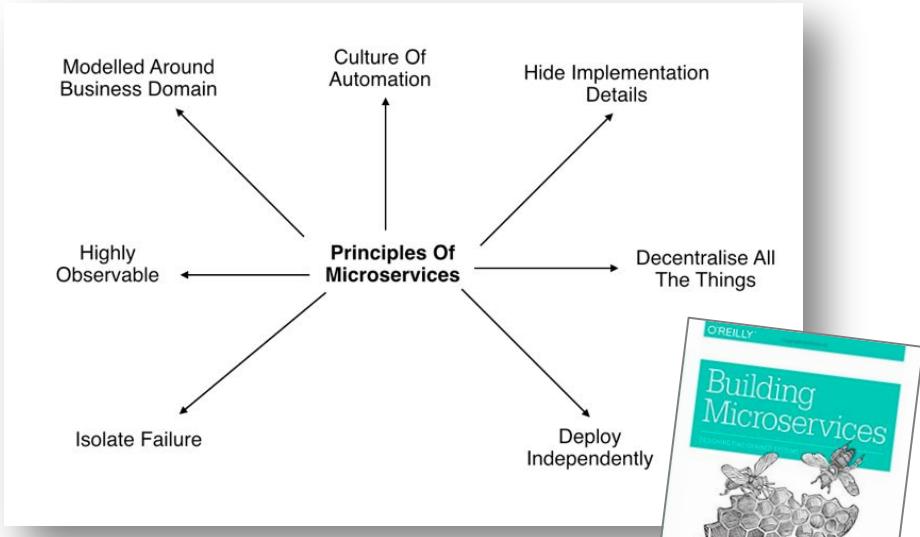
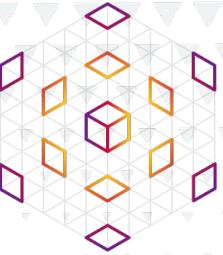
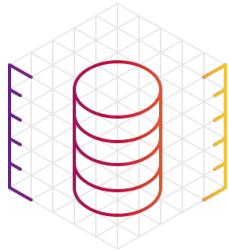
SOA, ca. 2005



Microservices, 2015+

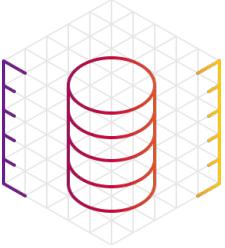


# ➤ Microservices, principles

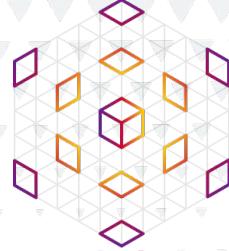


Organized around Business Capabilities  
Products not Projects  
Smart endpoints and dumb pipes  
Decentralized Governance  
Decentralized Data Management  
Infrastructure Automation  
Design for failure  
Evolutionary Design





## ➤ 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



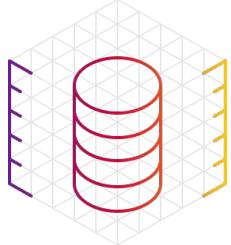
# "Share nothing" IRL



*in other words:*

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

Neither do I.



# » ACID ⇒ BASE

**A**tomicity

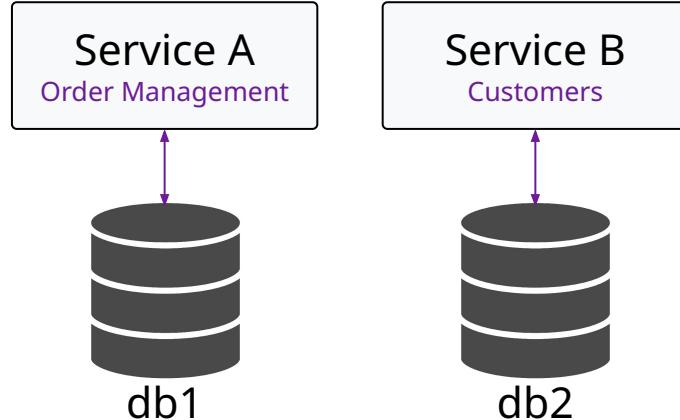
**C**onsistency

**I**solation

**D**urability

ACID:

✗ no distributed transactions  
across services



**B**asic

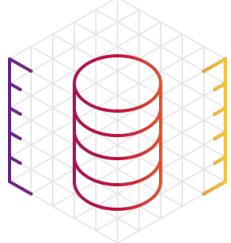
**A**vailability

**S**oft state

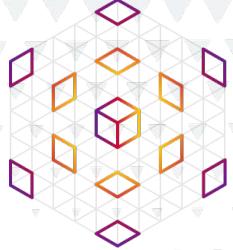
**E**ventual consistency

BASE:

- ✓ "AP" systems with idempotence
- ✓ event sourcing, CQRS

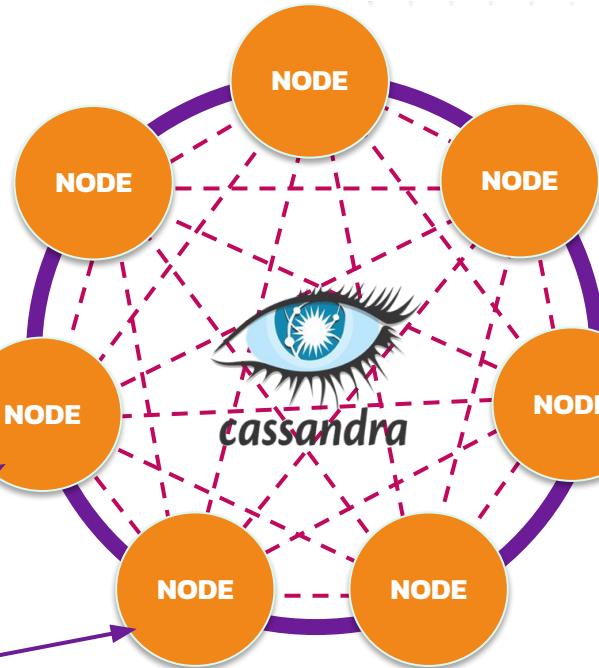
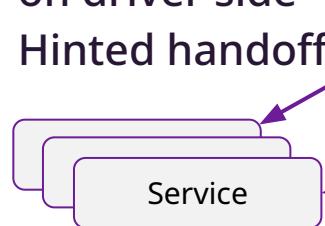


# ➤ Cassandra ❤️ microservices



## 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

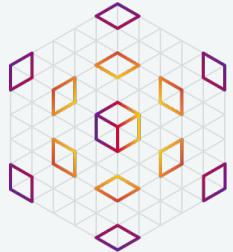
Create your Astra DB instance

[github.com/datastaxdevs/workshop-cassandra-application-development](https://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**

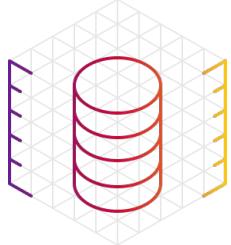
**Sensor REST API**

FastAPI, Spring Boot, Express.js

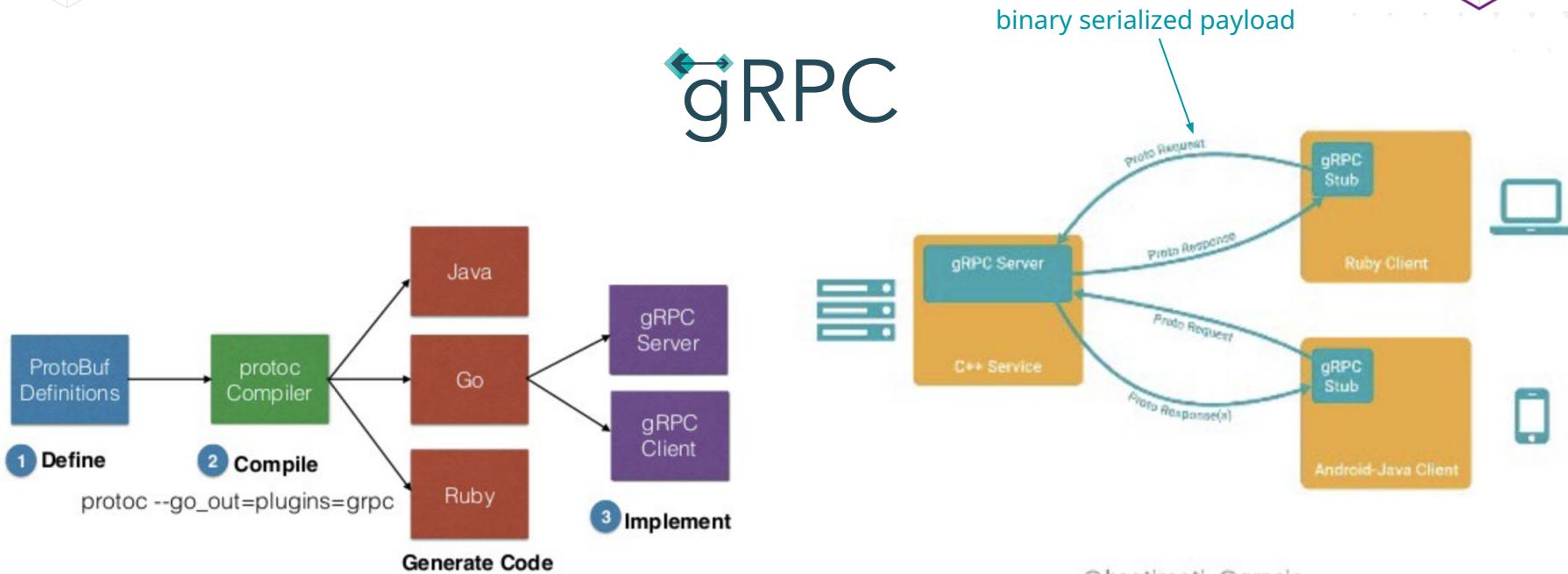
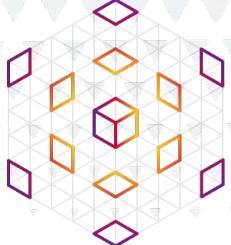
**06**

**What's next?**

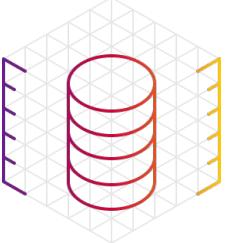
Homework, next workshops



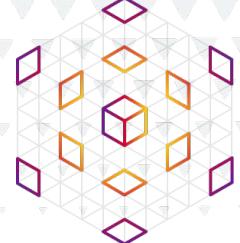
# ➤ Google Remote Procedure Call



@hostirosti @grpcio



# › Graph Query Language



Describe your data

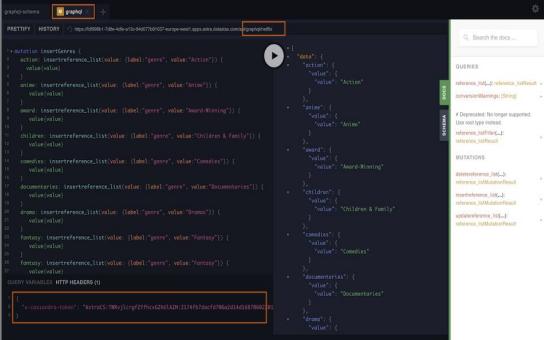
```
schema snippet
type Query {
    shows: [Show]
}

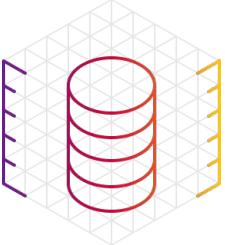
type Show {
    title: String!
    blurb: String
    actors: [Actor]
    releaseYear: Int
}
```

Ask for what you need

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

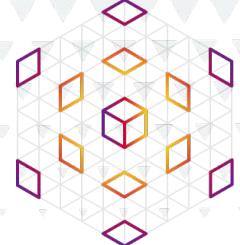
Discoverability





# › Representational state transfer

{ REST }



## Todos Implement CRUD operations for Todo Tasks

**GET** /api/v1/todos/ Retrieve the complete list of Taskss

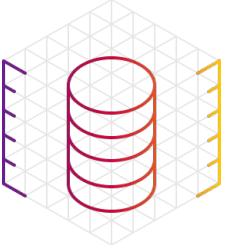
**POST** /api/v1/todos/ Create a new task

**DELETE** /api/v1/todos/ Delete all tasks in one go

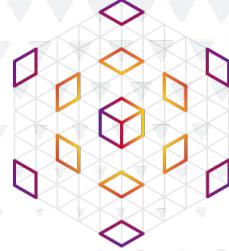
**GET** /api/v1/todos/{taskId} Get details of a task if exists

**DELETE** /api/v1/todos/{taskId} Delete a task from its id if 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*)



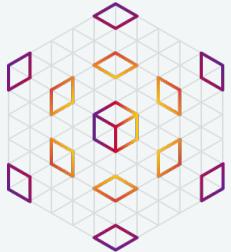
- Verbose payloads (*json, xml*)
- No discoverability
- Not suitable for command-like (*functions*) API



- 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**

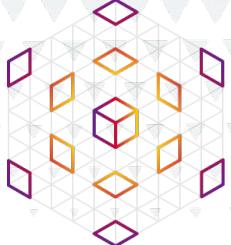
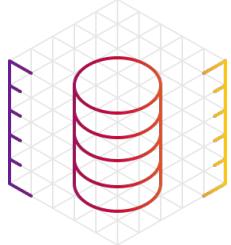
**Sensor REST API**

FastAPI, Spring Boot, Express.js

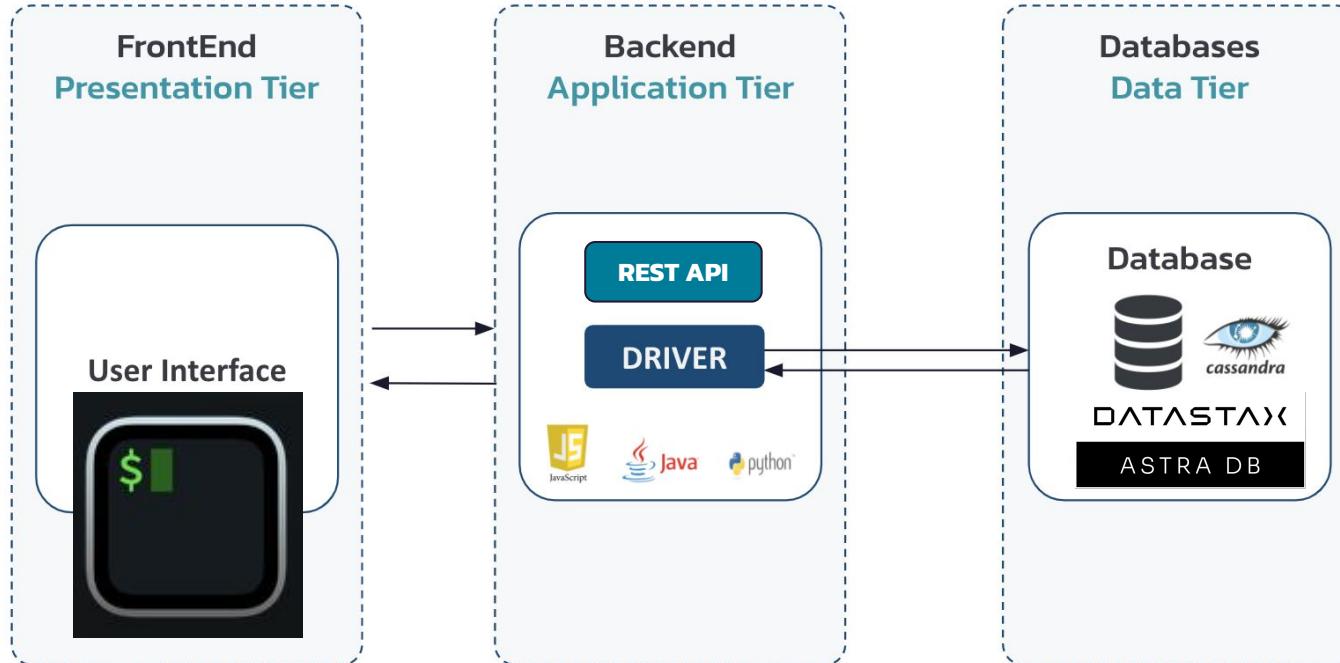
**06**

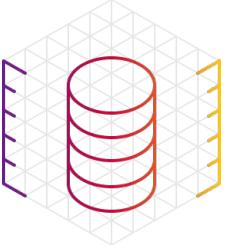
**What's next?**

Homework, next workshops

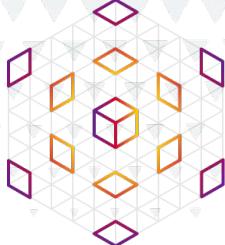


# › Today's microservice





# ➤ Did someone say "Drivers"?



## Connectivity

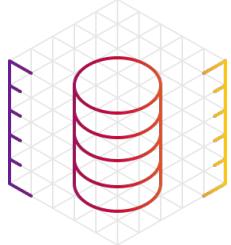
- ★ Token & Datacenter Aware
- ★ Load Balancing Policies
- ★ Retry Policies
- ★ Reconnection Policies
- ★ Connection Pooling
- ★ Health Checks
- ★ Authentication | Authorization
- ★ SSL

## Query

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

## Parsing Results

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



# Installation

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



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



```
npm install cassandra-driver
```

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

4.6.3

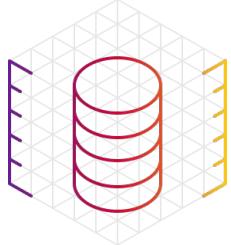


JavaScript

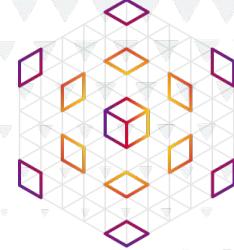
nuget v3.15.0

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

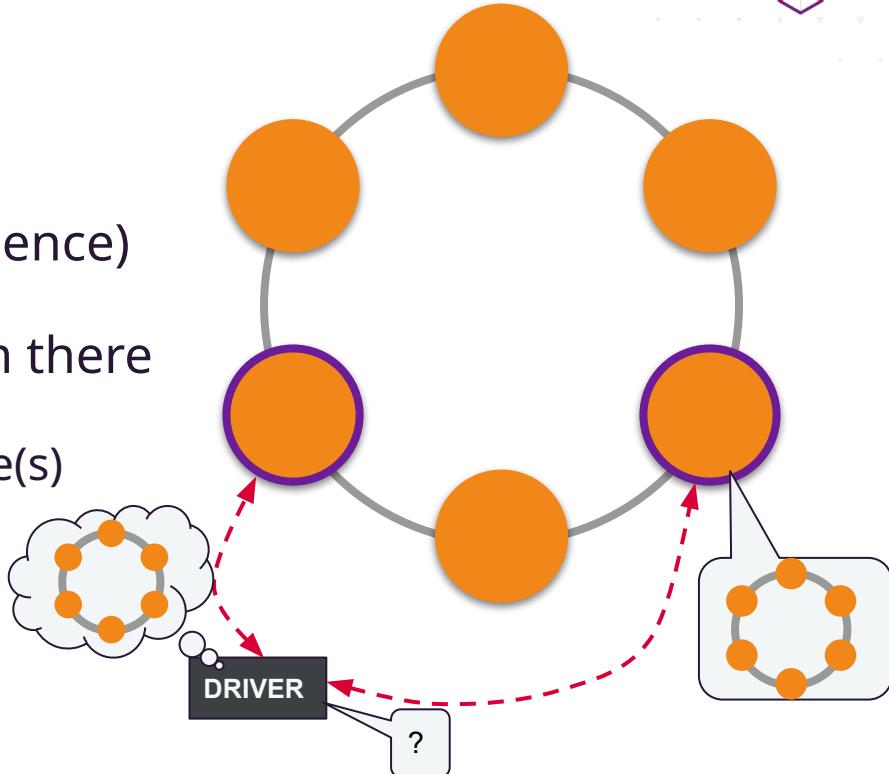


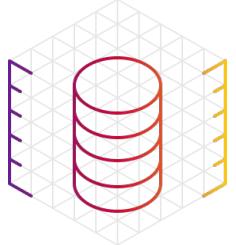


## ➤ Contact point (a Cassandra node)



- 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)



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



```
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')
```



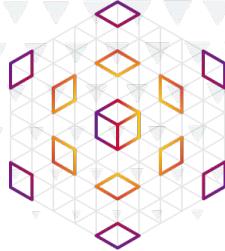
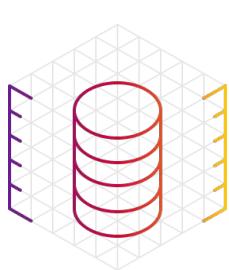
```
const client = new cassandra.Client({  
  contactPoints: ['127.0.0.1'],  
  localDataCenter: 'dc1',  
  keyspace: 'sensor_data',  
  credentials: { username: 'U', password: 'P' }  
});  
await client.connect();
```



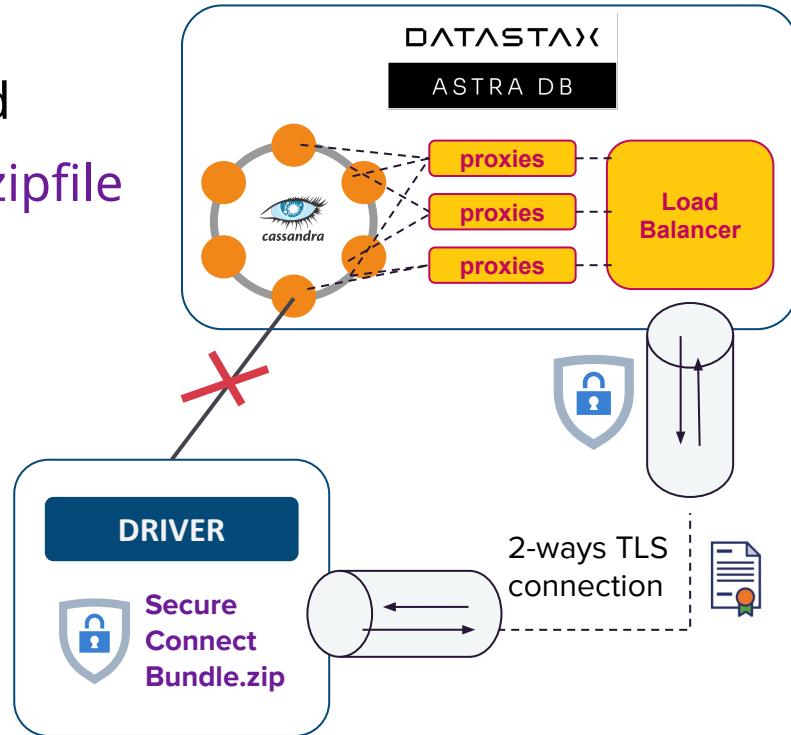
```
Cluster cluster = Cluster.Builder()  
    .AddContactPoint("127.0.0.1")  
    .WithCredentials("U", "P")  
    .Build();  
  
session = cluster.Connect("sensor_data");
```

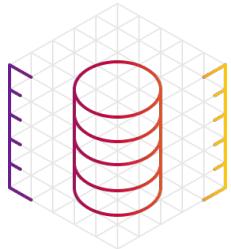


# ➤ Connecting to Astra DB - architecture



- 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();
```



```
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')
```

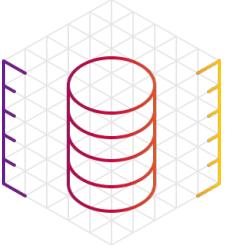


```
const client = new cassandra.Client({  
    cloud: { secureConnectBundle: 'secure.zip' },  
    credentials: { username: 'u', password: 'p' },  
    keyspace: 'sensor_data'  
});  
  
await client.connect();
```

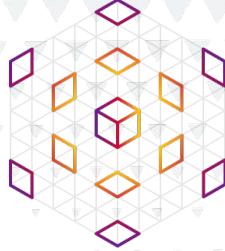


```
var cluster = Cluster.Builder()  
    .WithCloudSecureConnectionBundle("secure.zip")  
    .WithCredentials("u", "p")  
    .Build();  
  
var session = cluster.Connect("sensor_data");
```



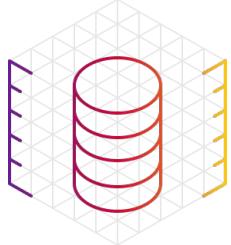


## ➤ the Session object

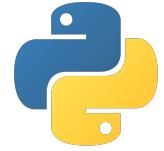
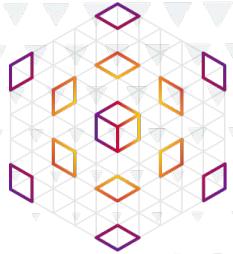


- **Stateful** object handling communications with each node
- Should be **unique** in the Application (*Singleton*)
- Should be **closed** at application shutdown (*shutdown hook*) in order to free TCP sockets

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



## Run CQL queries



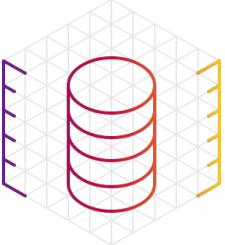
```
session.execute(  
    "SELECT * FROM sensors_by_network WHERE network = %s;",  
    (network,),  
)
```



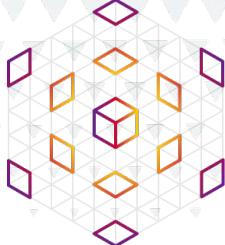
```
cqlSession.execute(  
    "SELECT * FROM sensors_by_network WHERE network = '" + network + "'"  
) ;
```



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

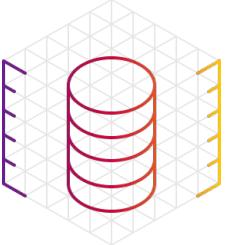


# » A Javascript note: type hints!

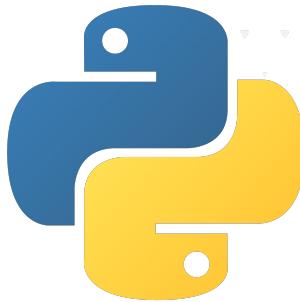


```
await client.execute(  
    "INSERT INTO metals (kind, name, density) VALUES (?, ?, ?);",  
    ['regular', 'palladium', 12.02],  
    {  
        hints: [  
            'text',  
            'text',  
            'float'  
        ]  
    }  
);
```

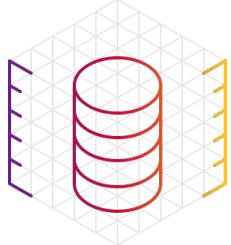




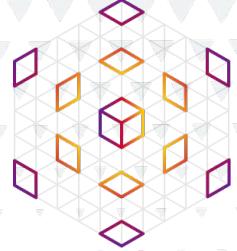
## ➤ Prepare your statements (1)



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

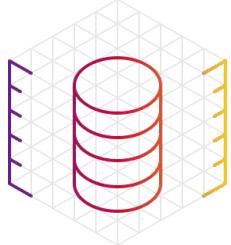


## ➤ Prepare your statements (2)

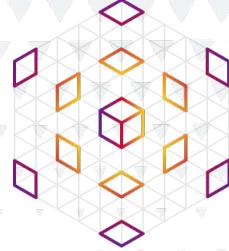


```
PreparedStatement q3Prepared = session.prepareStatement(  
    "SELECT * FROM sensors_by_network WHERE network = ?");
```

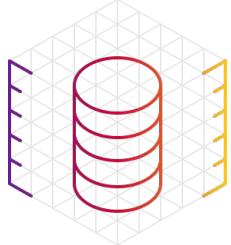
```
BoundStatement q3Bound = q3Prepared.bind(network);  
ResultSet rs = session.execute(q3Bound);
```



## ➤ Prepare your statements (3)

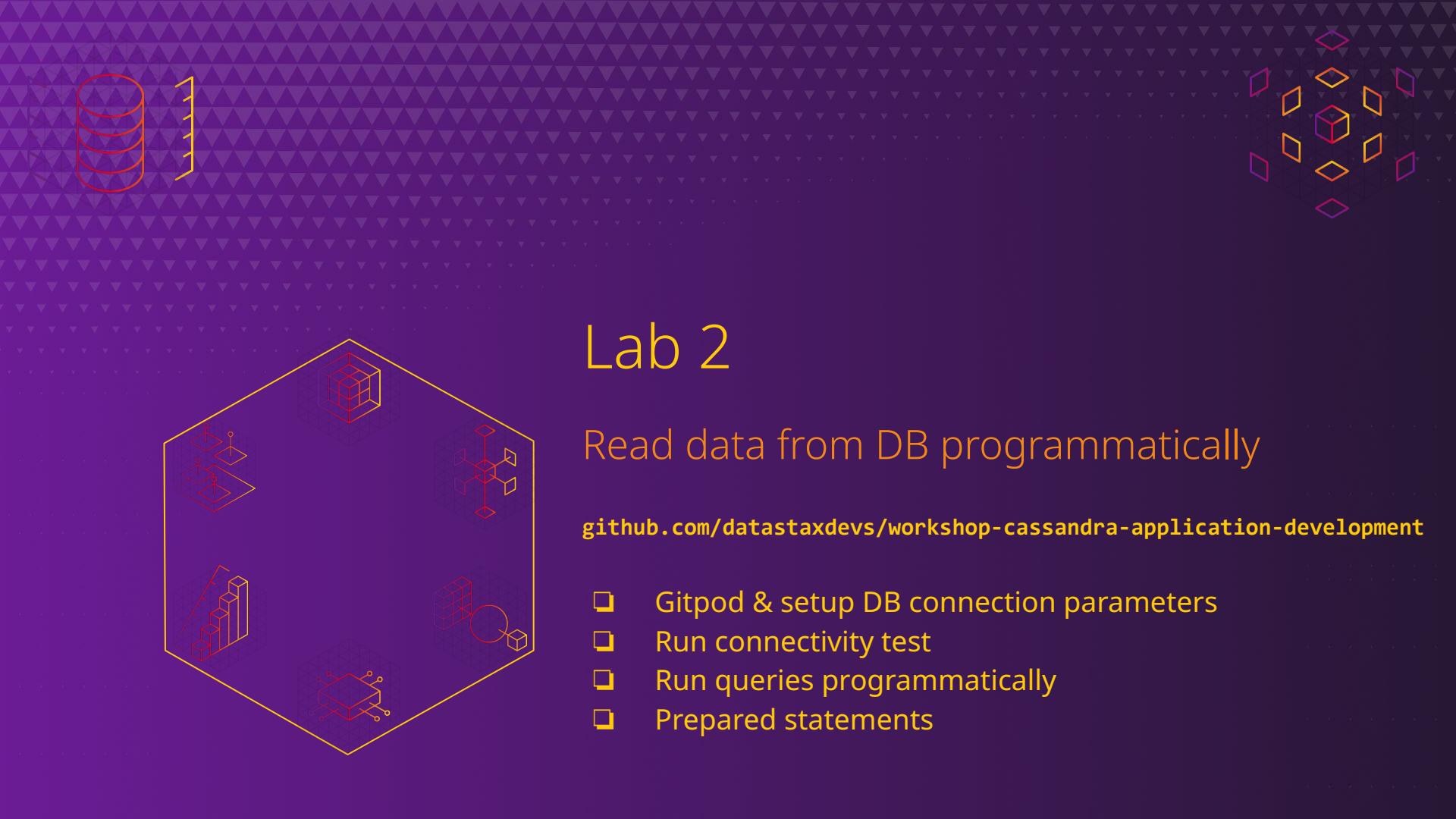


```
session.execute(  
    "SELECT * FROM sensors_by_network WHERE network = ?;",  
    [network],  
    {prepare: true}  
).then( .....
```



## ➤ 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

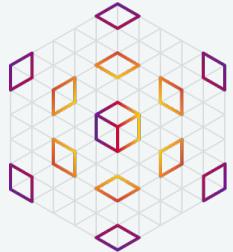
Read data from DB programmatically

[github.com/datastaxdevs/workshop-cassandra-application-development](https://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**

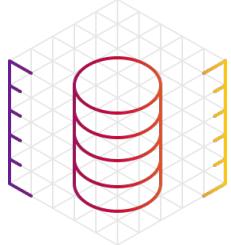
FastAPI, Spring Boot, Express.js

---

**06**

**What's next?**

Homework, next workshops



# › Technical stacks



Backend

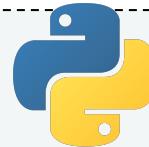


Spring Boot

Spring MVC

JAVA Drivers

Backend



Uvicorn

FastAPI

Python

PYTHON Drivers

Backend

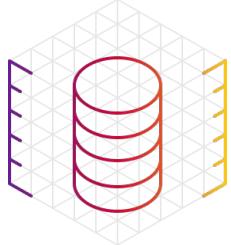


Node.js

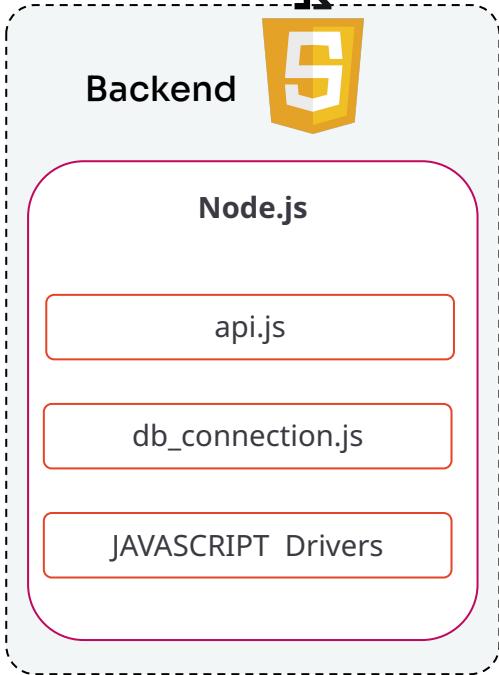
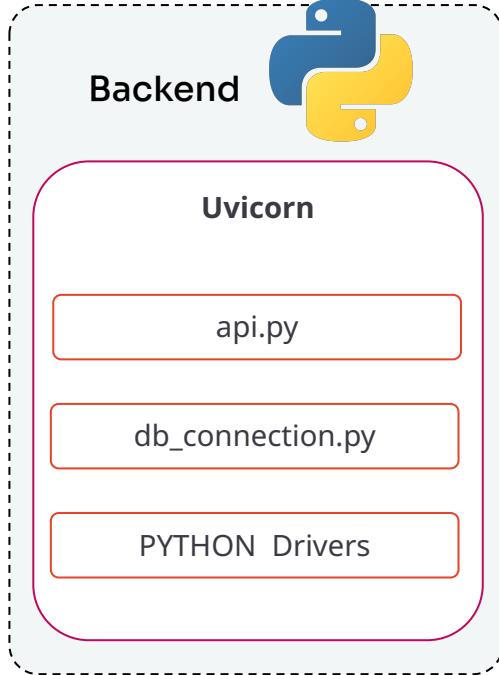
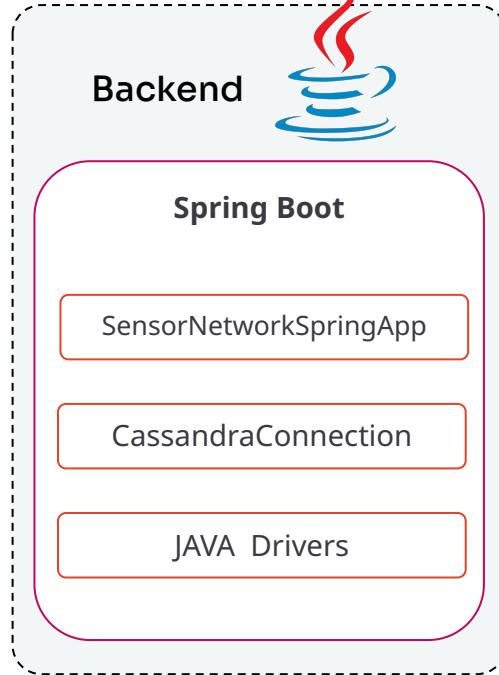
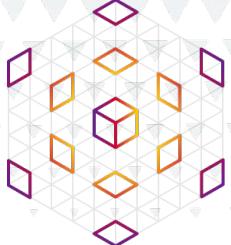
Express.js

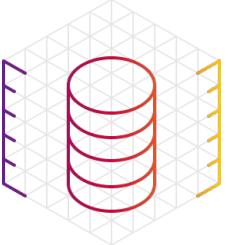
Javascript

JAVASCRIPT Drivers



# ➤ API structure/modules



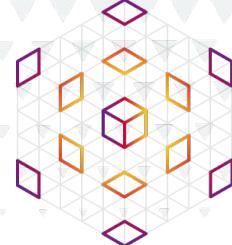


## » REST API specs

# 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"}
```

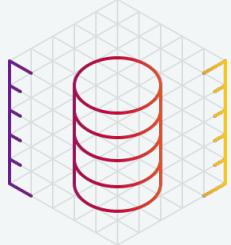
Wait a minute...  
Why a POST?

# Lab 3

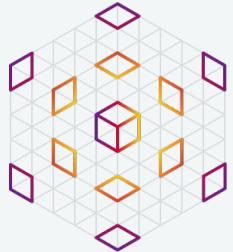
## Sensor API

[github.com/datastaxdevs/workshop-cassandra-application-development](https://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**

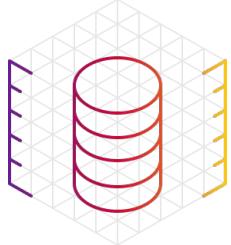
FastAPI, Spring Boot, Express.js

---

**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*



## API and Microservices with Cassandra Homework

Welcome and thank you!

Here you can submit your homework for the DataStax Developers with Cassandra® workshop.

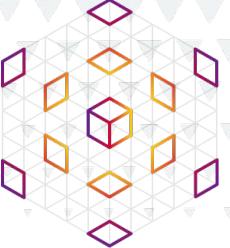
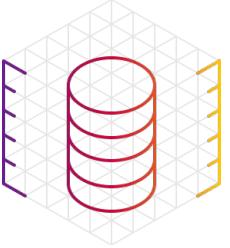
In case of any questions please contact the organizers at <https://dtsx.io/cedrick> or just send an email to [aleksandar@datastax.com](mailto:aleksandar@datastax.com)

- Workshop materials: <https://github.com/datastaxdev/development#readme>
- Discord chat: <https://dtsx.io/discord>

Email \*

Your email



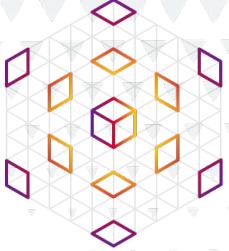
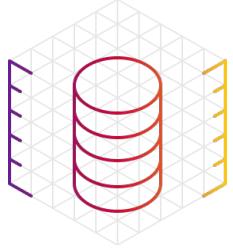


› You are not alone in your journey

- Discord:** [dtsx.io/discord](https://dtsx.io/discord)
- StackOverflow<sup>(\*)</sup>:** [stackoverflow.com/questions/tagged/cassandra](https://stackoverflow.com/questions/tagged/cassandra)
- DBA Stack Exchange<sup>(\*)</sup>:** [dba.stackexchange.com/questions/tagged/cassandra](https://dba.stackexchange.com/questions/tagged/cassandra)

*(\*) For best results, follow the cassandra tag*

# Our Discord community (18k+)



!discord

**PRESENTER — 1**

- David Jones-Gillard

**HELPER — 7**

- 012345
- AaronP
- BInary
- Chelsea Navo
- Jeremy Hanna
- John Sanda
- Patrick\_McFadin

**EN LIGNE — 560**

- samu-
- 6304-42J8
- Aahlya
- Abdurahim
- abhi3pathi
- Abhiis.s
- Abhineet
- Abirsh

19 novembre 2021

**RIGGITYREKT** Hier à 21:14  
I have a 5 node datacenter, 4 nodes are on dse version 5.1.20, one is on dse5.0.15. I am doing some mixed version testing for a class and the one node that is 5.0.15 is coming up as an analytics workload. I dont have /etc/default/dse, instead I am using /etc/init.d/dse-cassandra  
how do i make that node start in cassandra workload, not in analytics?

**RIGGITYREKT** Hier à 23:39  
Okay I found out my issue, when i started DSE 5.0.15 it had endpointsnitch set to DseSimpleSnitch, the rest of my cluster is using PropertyFileSnitch, when i change it to PropertyFileSnitch, it still uses the simple snitch config. looking at the docs i see there is a way to go to GossipingPropertyfileSnitch, but i need the property file one. I can wipe this dbs, do anything with this node to get this done. how do i fix this?  
@here

**Erick Ramirez** Aujourd'hui à 02:19  
mixed versions isn't supported and you're guaranteed to run into weird issues that will cause further problems down the track

**RIGGITYREKT** I have a 5 node datacenter, 4 nodes are on dse version 5.1.20, one is on dse5.0.15. I am doing some mixed v...  
**Cedrick Lunven** Aujourd'hui à 09:01  
When you start a node you have parameters -k for analytics, -g for graph and -s for search. To remove analytics check and remove -k

Envoyer un message dans #workshop-chat

dtsx.io/discord

**DataStax Developers**  
@DataStaxDevs  
29.9K subscribers

HOME VIDEOS SHORTS LIVE PLAYLISTS COMMUNITY CHANNELS ABOUT

Recently uploaded Popular

0:38 1:26 6:53

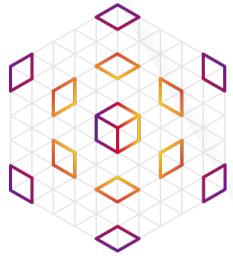
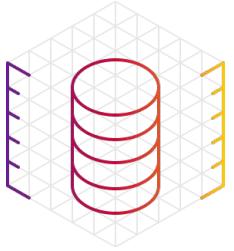
The Legend of DataStax - Cassandra Summit Training Day!

Cassandra Summit Training Day - Sunday March 12th

CDC for Astra DB Demo: Sink to Astra DB

92 views • 2 weeks ago

CDC for Astra DB Demo: ElasticSearch



# Thank You