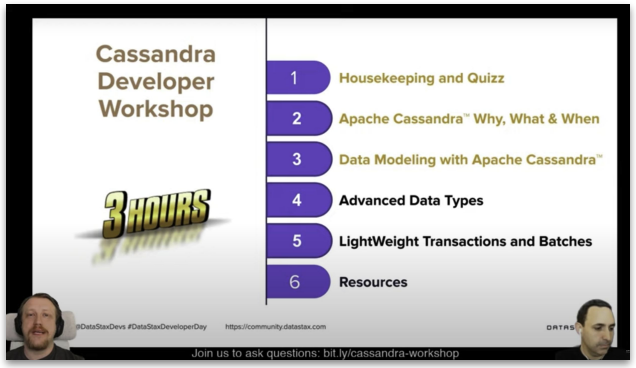
Welcome

CRUD with NodeJS and Python and Datastax Astra

The Crew

DataStax Developer Advocacy Special Unit

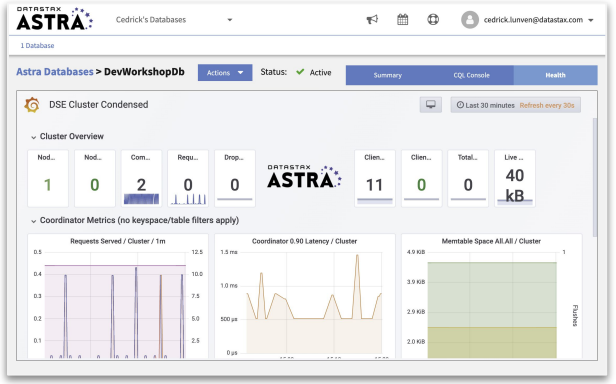
Courses: youtube.com/DataStaxDevs



YouTube

Twitch

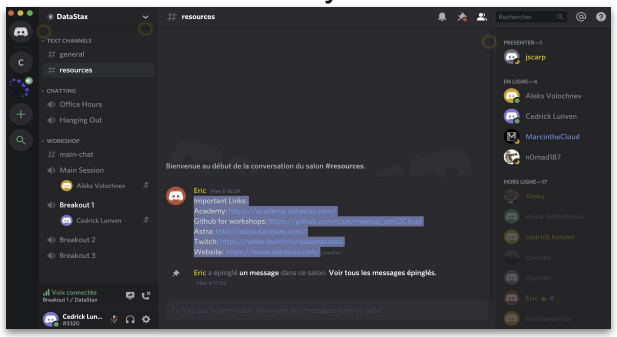
Runtime: dtsx.io/workshop



cassandra

DATASTAX ASTRA

Questions: bit.ly/cassandra-workshop

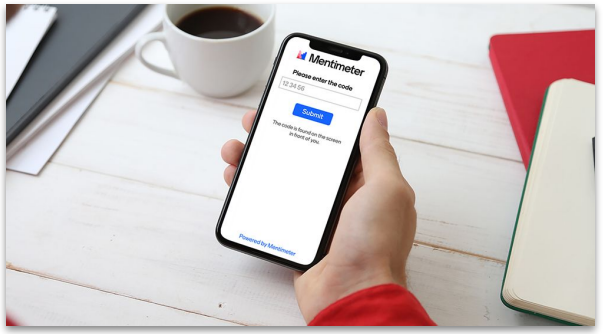


Discord

YouTube

Available on the iPhone App Store

Quizz: menti.com



Mentimeter

GET IT ON Google play

Disclaimer

This is a coding session. You will need some experience with the NodeJS or Python and have a github account.

You can do these exercises on a local install, but we will only use Gitpod today during the workshop.

No need to install anything.

Hands-on exercise material

DATASTAX ASTRA

Get your instance here:

- http://dtsx.io/workshop

GitHub

Repository:

- https://github.com/DataStax-Academy/workshop-crud-with-python-and-node

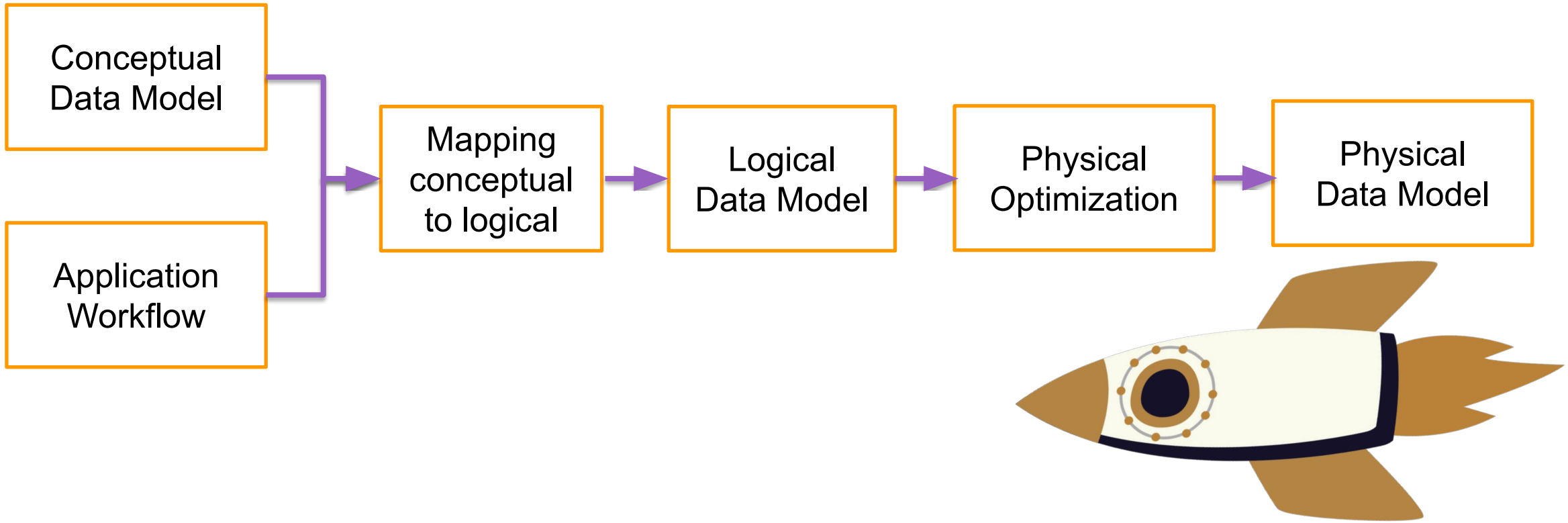

menti.com
Telenor SE
16:04
83 %
menti.com
Mentimeter
What is the most desirable feature of your ideal workplace?
Flexible time
Social factor
Big paycheck
Available on the iPhone
App Store
GET IT ON
Google play

Application Development **CRUD**

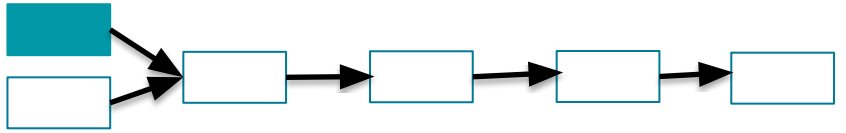
1. Use Case and data model
2. Set up Astra database and schema
3. Connect to Astra
4. Create and update records
5. Read results

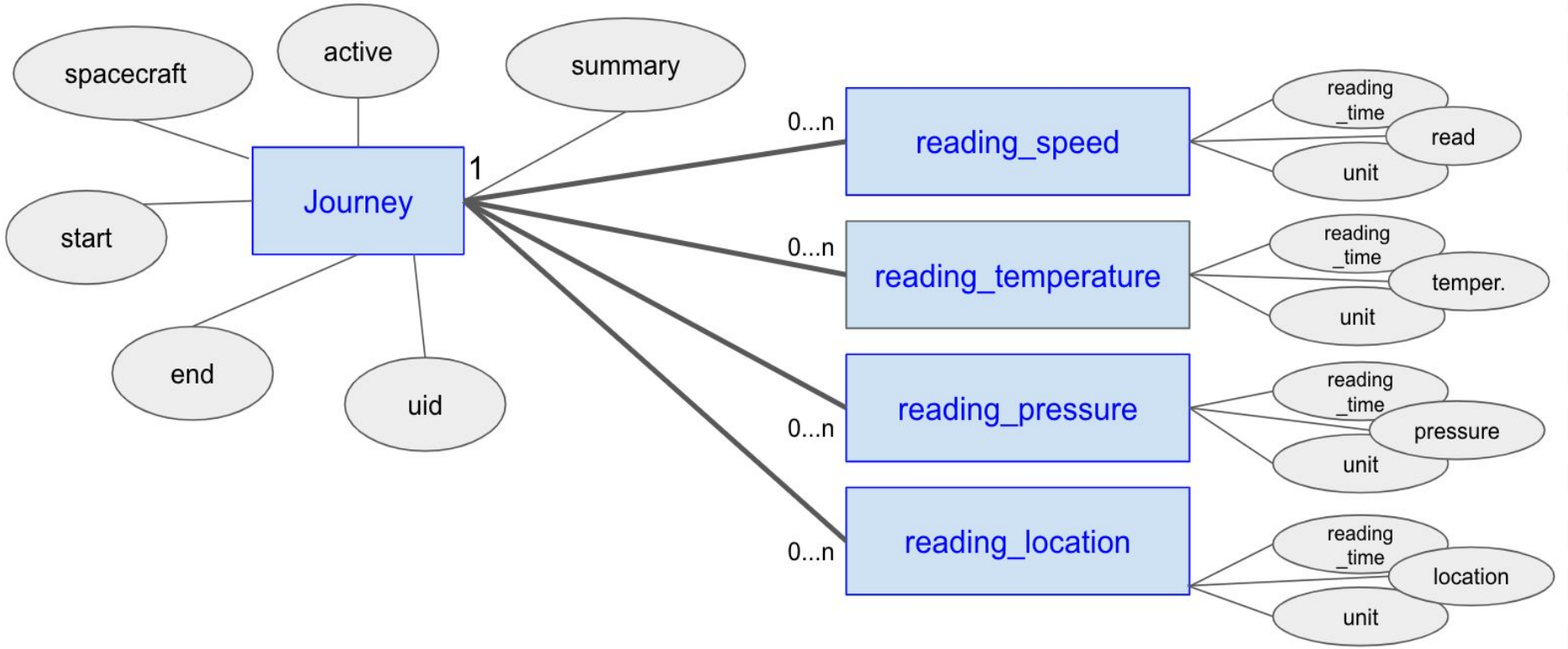

DATASTAX Getting Started with Astra
LAUNCH NEW JOURNEY
Showtime !!
Journey Checklist
Writing 1000 Rows to Astra
Reading 1000 Rows from Astra
Replaying Rows
Astra Table | Rows Written | Rows Read | Current Row Displayed
spacecraft_temperature_over_time | 1000 | 1000 | 189
spacecraft_speed_over_time | 1000 | 1000 | 189
spacecraft_pressure_over_time | 1000 | 1000 | 189
spacecraft_location_over_time | 1000 | 1000 | 189
Temperature: 69.85 Fahrenheit
Astra Table: spacecraft_temperature_over_time
Speed: 31536.95 km/h
Astra Table: spacecraft_speed_over_time
Pressure: 100.3 kPa
Astra Table: spacecraft_pressure_over_time
X: 14433 km
Y: 14433 km
Astra Table: spacecraft_location_over_time

Designing your data model

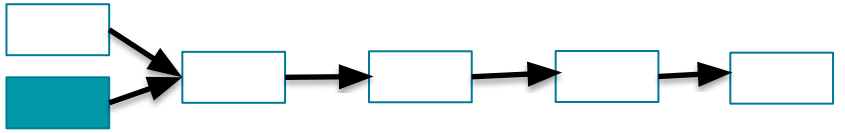


#1 Conceptual Data Model



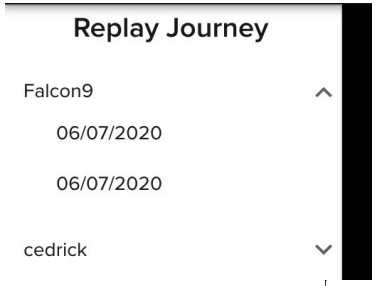


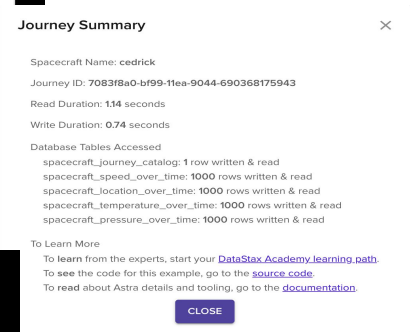
#1 Application Workflow

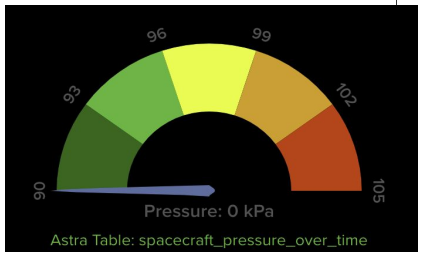


- **Space crafts catalog queries**
  - Look up all of the journeys for a spacecraft
  - Look up the state of a journey
  - Create a new journey

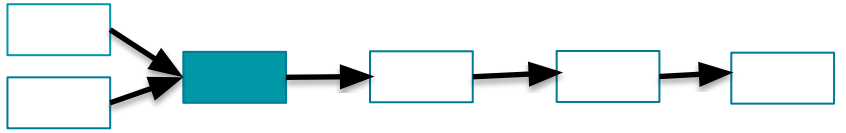
- **Sensor readings queries :** Speed, Pressure, Temperature, Location
  - Save readings over time
  - Analyze each dimension independently
  - Analyze data per journey
  - Less than 100.000 records per journey per dimension







#2 Map to Logical Data Model



spacecraft_temperature_over_time

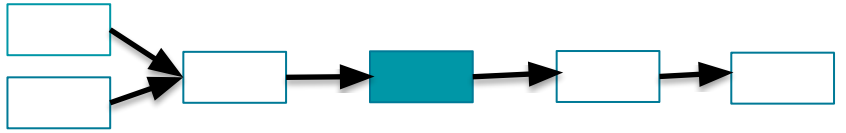
spacecraft_location_over_time

spacecraft_journey_catalog

spacecraft_speed_over_time

spacecraft_pressure_over_time

#3 Logical Data Model

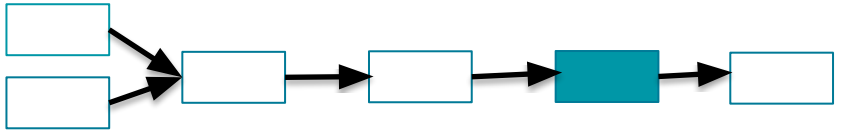


spacecraft_journey_catalog	
spacecraft_name	K
journey_id	C ↓
start	
end	
active	
summary	

spacecraft_temperature_over_time	
spacecraft_name	K
journey_id	K
reading_time	C ↓
speed	
speed_unit	

spacecraft_location_over_time	
spacecraft_name	K
journey_id	K
reading_time	C ↓
pressure	
pressure_unit	

#4 Physical Data Model



spacecraft_temperature_over_time

spacecraft_location_over_time

spacecraft_journey_catalog

spacecraft_name	text
journey_id	timeuuid
start	timestamp
end	timestamp
active	boolean
summary	text

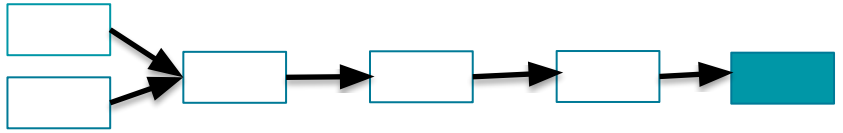
spacecraft_speed_over_time

spacecraft_name	text
journey_id	timeuuid
reading_time	timestamp
speed	double
speed_unit	text

spacecraft_pressure_over_time

spacecraft_name	text
journey_id	timeuuid
reading_time	timestamp
pressure	double
pressure_unit	text

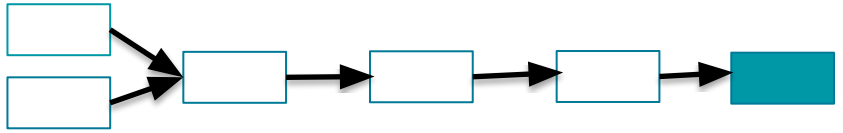
#5 CQL DDL



```
CREATE TABLE IF NOT EXISTS spacecraft_journey_catalog (
  spacecraft_name text,
  journey_id      timeuuid,
  start           timestamp,
  end             timestamp,
  active          boolean,
  summary         text,
  PRIMARY KEY ((spacecraft_name),
  WITH CLUSTERING ORDER BY (journ
```

```
CREATE TABLE IF NOT EXISTS spacecraft_speed_over_time (
  spacecraft_name text,
  journey_id      timeuuid,
  speed           double,
  reading_time    timestamp,
  speed_unit      text,
  PRIMARY KEY ((spacecraft_name, journey_id), reading_time))
  WITH CLUSTERING ORDER BY (reading_time DESC);
```

#5 CQL DDL with UDT



```
CREATE TYPE IF NOT EXISTS location_udt (
    x_coordinate double,
    y_coordinate double,
    z_coordinate double
);
```

```
CREATE TABLE IF NOT EXISTS spacecraft_location_over_time (
    spacecraft_name text,
    journey_id timeuuid,
    reading_time timestamp,
    location frozen<location_udt>,
    location_unit text,
    PRIMARY KEY ((spacecraft_name, journey_id), reading_time)
);
```

Application Development **CRUD**

1. Use Case and Datamodel
2. Set up Astra database and schema
3. Connect to Astra
4. Create and update records
5. Read results

Introduction to Astra

DATASTAX ASTRA == cassandra

Global Scale
Put your data where you need it without compromising performance, availability, or accessibility.

No Operations
Eliminate the overhead to install, operate, and scale Cassandra.

5 Gig Free Tier
Launch a database in the cloud with a few clicks, no credit card required.

Hands-on prep work: Create your Astra database

- If you are new to Astra, sign up here: **dtsx.io/workshop**
- add a new keyspace (if you already have one)
- Take a note of the username, password, keyspace of your cluster
- We will be using these defaults:
  - keyspace: spacecraft
  - user name: SUser
  - password: SPassword1
- Download the secure-connect-bundle, **this is unique to you and your database instance**

Hands-on Exercise 1:

- Create the Astra database
- Create the schema in the database
  - Log into Astra CQL console
  - Copy schema from github and paste into CQL console

Application Development **CRUD**

1. Use Case and Datamodel
2. Set up Astra database and schema
3. Connect to Astra
4. Create and update records
5. Read results

Datastax Drivers

One of set drivers to connect them all - **January 2020**

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

Install Drivers

```
npm install cassandra-driver
```

```
pip install cassandra-driver
```

Connection to Cassandra ...with ASTRA

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

```
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('killrvideo')
```

Important about the session

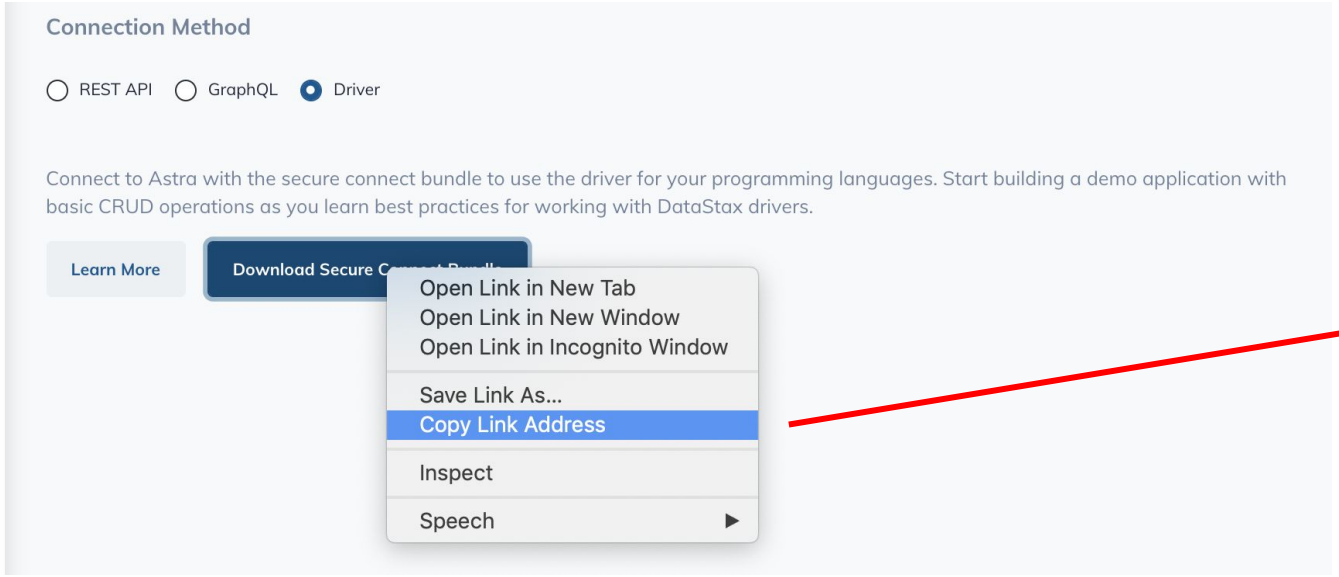
- It is a stateful object handling communications with each node
- session should be unique per application (Singleton)
- session should be closed at application shutdown (shutdown hook) in order to free opened TCP sockets (stateful)

```
Node:    client.shutdown();
Python:  session.shutdown();
```

Hands-on Exercise 2:
Setup connection to Astra

- **Get gitpod ready**
- **Install the drivers**
- **Save the secure connection bundle in gitpod**
- **Update the connection settings in node and python**
- **Test the connection:**
  - **Run Ex02 code**

Get the secure bundle with curl



click to copy the link to clipboard
(refresh page before copy, link is only 'fresh' for short time)

```
curl -L "<insert link here>" > creds.zip
```

Application Development **CRUD**

1. Use Case and Datamodel
2. Set up Astra database and schema
3. Connect to Astra
4. Create and update records
5. Read results

Execute simple statements

```
client.execute('select * from t1 where c1 = ?', [5]);
```

```
session.execute("select * from t1 where c1 = %s", 5);
```

Prepared and Bound Statements

- Compiled once on each node automatically as needed
- Prepare each statement only once per application

```
const query = 'select * from t1 where c1 =?;';
client.execute(query, [5], { prepare: true })
```

```
prepared = session.prepare("select * from t1 where c1=?")
result = session.execute(prepared, [5])
```

Hands-on Exercise 3 and 4:
Inserts with simple and prepared statements

- **Simple insertion into spacecraft_journey_catalog:**
  - **Run Ex03 code**
- **Insertion with prepared statement:**
  - **Adapt Ex04 code and run**

Working with batch statements

```
var myBatch = [
       { query: insertSpeed, params: [spacecraft_name, journey_id, speed,readingTime,'km/hour' ] },
       { query: insertTemperature, params: [spacecraft_name, journey_id, pressure, readingTime,'Pa' ] },
       { query: insertPressure, params: [spacecraft_name, journey_id, temperature, readingTime,'K' ] },
       { query: insertLocation, params: [spacecraft_name, journey_id, location,readingTime,'AU' ] }
]
const result = await connection.client.batch(myBatch, { prepare: true })
```

```
batch = BatchStatement()
batch.add(prepared_insertLocation, [spacecraft_name, journey_id, Location(x,y,z),readingTime,'AU' ])
batch.add(prepared_insertSpeed, [spacecraft_name, journey_id, speed,readingTime,'km/hour' ])
batch.add(prepared_insertTemperature, [spacecraft_name, journey_id, pressure, readingTime,'Pa' ])
batch.add(prepared_insertPressure, [spacecraft_name, journey_id, temperature, readingTime,'K' ])
connection.session.execute(batch)
```

Working with UDTs (Python special)

- Recommended to register UDTs with Cluster instance
- When using prepared statements, not necessary to register

```
class Foo(object):
    def __init__(self, street, zipcode, otherstuff):
        self.street = street
        self.zipcode = zipcode
        self.otherstuff = otherstuff

insert_statement = session.prepare("INSERT INTO users (id, location) VALUES (?, ?)")

session.execute(insert_statement, [0, Foo("123 Main St.", 78723, "some other stuff")])
```

Hands-on Exercise 5:
Inserts with UDTs and batches

- **Adapt and run Ex05**

Application Development **CRUD**

1. Use Case and Datamodel
2. Set up Astra database and schema
3. Connect to Astra
4. Create and update records
5. Read results

Hands-on Exercise 7:
Simple selects, parsing results and paging

- **Simple selects: Ex07**
- **Parsing: Ex08 and Ex09**
- **Paging: Ex10**

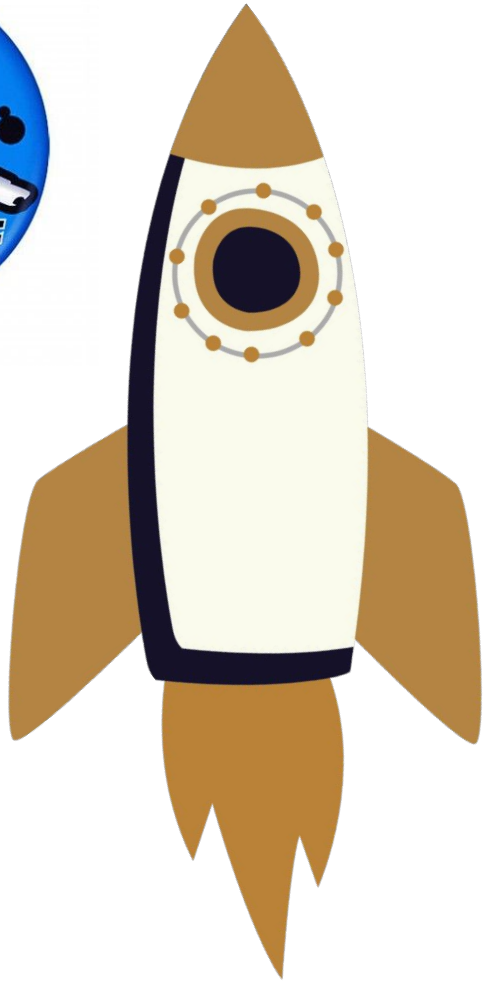

menti.com
Telenor SE
16:04
83 %
menti.com
Mentimeter
What is the most desirable feature of your ideal workplace?
Flexible time
Social factor
Big paycheck
Available on the iPhone
App Store
GET IT ON
Google play

Engage !

Share with us you Cassandra use cases !

Share with your vision and future of Cloud Native

Share what you need to succeed with Cassandra.



Developer Resources

LEARN

- Join academy.datastax.com
- Browse www.datastax.com/dev

ASK/SHARE

Join community.datastax.com

Ask/answer community user questions - share your expertise

CONNECT

Follow us

We are on Youtube - Twitter - Twitch!

MATERIALS

Slides and materials from all workshops are available at https://github.com/DataStax-Academy

DATASTAX®

Thank You