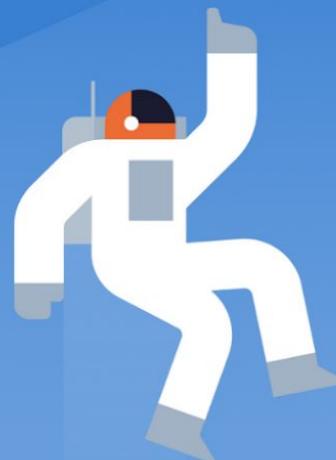


DataStax

An Easy Backend for your application

Cloud-native Cassandra series



LEVEL
UP
with the

DataStax

Developers

Buckle up and Register for each week

YOU
ARE
HERE

LIVE hands-on workshop </>

Cloud-Native Series Part 1:

An Easy Backend
for your Application

Multiple Dates | NoSQL | Beginner

LEVEL UP
with the
Developers

WED, JAN 6 5:00 PM

Cloud-Native Series - An Easy Backend for
your application



HOMEWORK



FREE

LIVE hands-on workshop </>

Cloud-Native Series Part 2:

Connecting Kubernetes
and Apache Cassandra

Multiple Dates | NoSQL | Beginner

LEVEL UP
with the
Developers

WED, JAN 13 5:00 PM

Cloud-Native Series - Connecting
Cassandra and Kubernetes



FREE

LIVE hands-on workshop </>

Cloud-Native Series Part 3:

Taking your Application
to the Cloud

Multiple Dates | NoSQL | Beginner

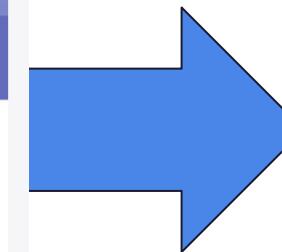
LEVEL UP
with the
Developers

WED, JAN 20 5:00 PM

Cloud-Native Series - Deploying your k8s
apps to the Cloud



FREE



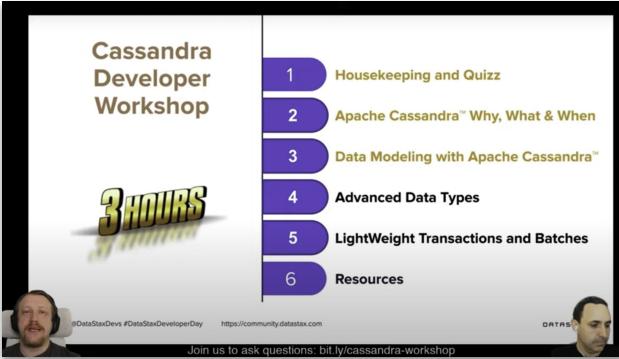
HOMEWORK



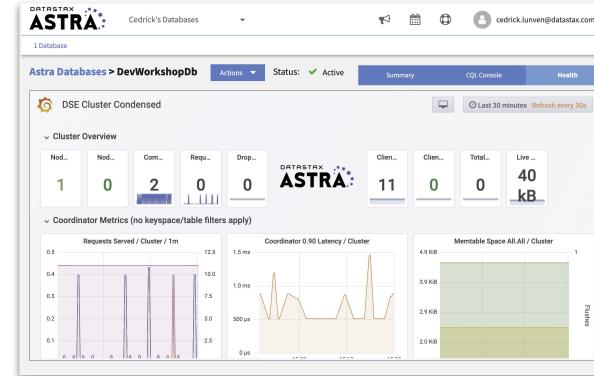
HOMEWORK

Housekeeping

Courses: youtube.com/DataStaxDevs



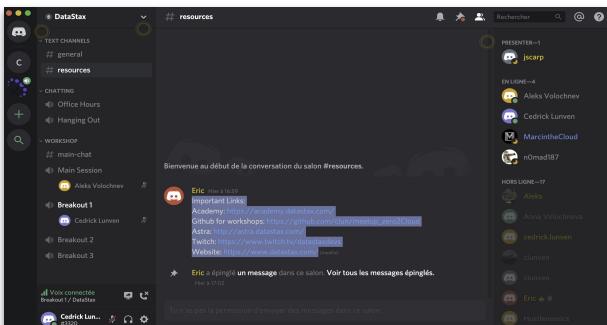
Runtime: dtsx.io/workshop



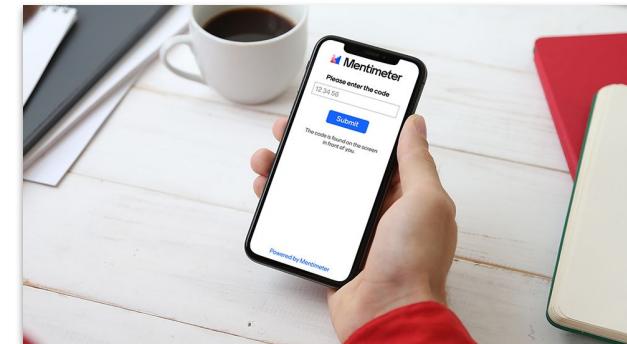
DataStax
Astra DB



Questions: bit.ly/cassandra-workshop



Quizz: menti.com



 **Mentimeter**



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 a preconfigured cloud-based IDE today during the workshop.

No need to install anything.



GitPod.io

A screenshot of the GitPod.io interface. On the left, the Explorer sidebar shows a project structure with 'crud-nodejs' and 'crud-python' folders containing various script files like db_connection.py, Ex02_Connect_to_Cassandra.py, etc. The main workspace shows a preview of README.md with the title 'CRUD operations with NodeJS and Python'. Below it, a message from DataStax welcomes users to a workshop on connecting, updating, and reading records from Apache Cassandra using Astra. A 'Video Recordings' section lists recordings for Wednesday, October 7th (NAM Time) and Thursday, October 8th (IST Time). The bottom terminal window shows the command 'gitpod /workspace/workspace-crud-with-python-and-node \$'.

Hands-on exercise material



Get your instance here:

- <http://dtsx.io/workshop>

Repository:

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



menti.com

67 47 24 6

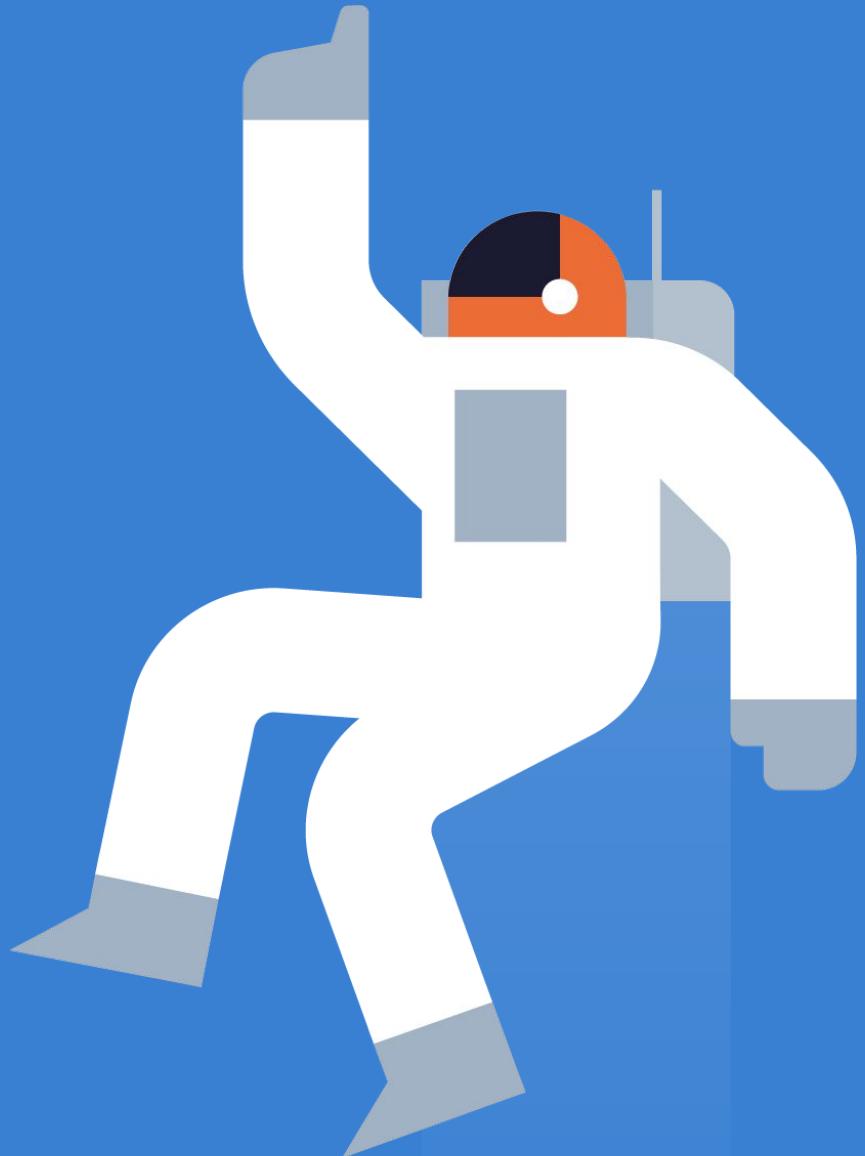


Available on the iPhone
App Store

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



What does it do ?

The screenshot shows the DataStax Astra interface titled "Getting Started with Astra". At the top right is a "LAUNCH NEW JOURNEY" button. On the right side of the screen is a large, stylized illustration of a white rocket ship with orange fins and a blue base, set against a dark background with stars.

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

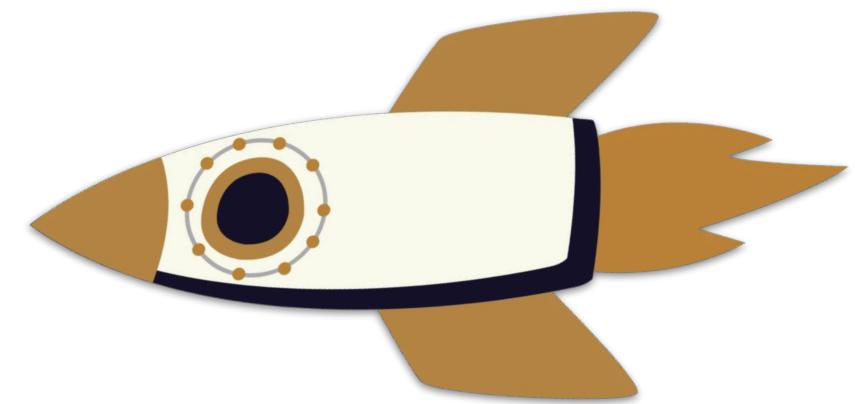
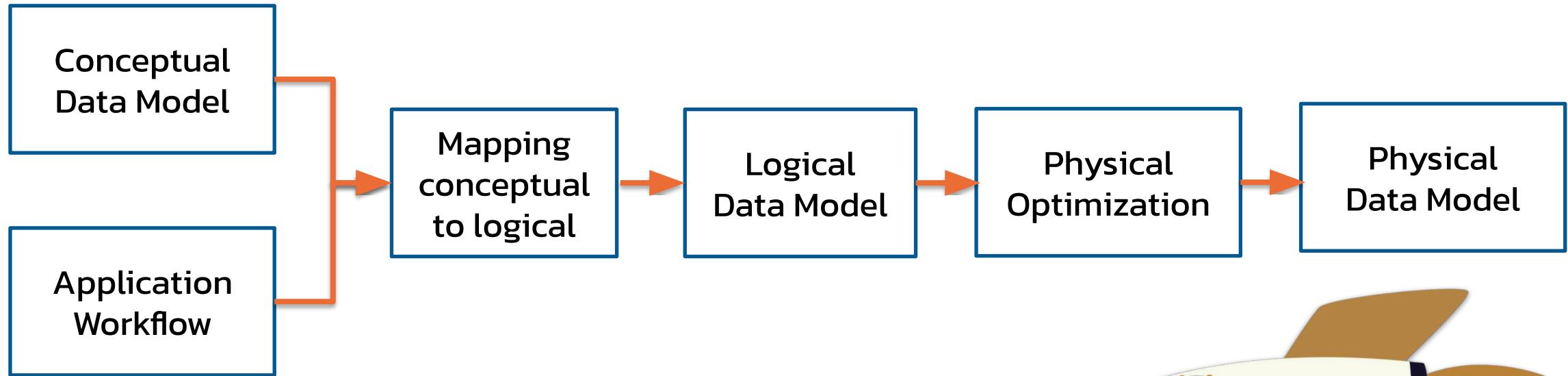
Pressure: 100.3 kPa
Astra Table: spacecraft_pressure_over_time

Current Spacecraft Location

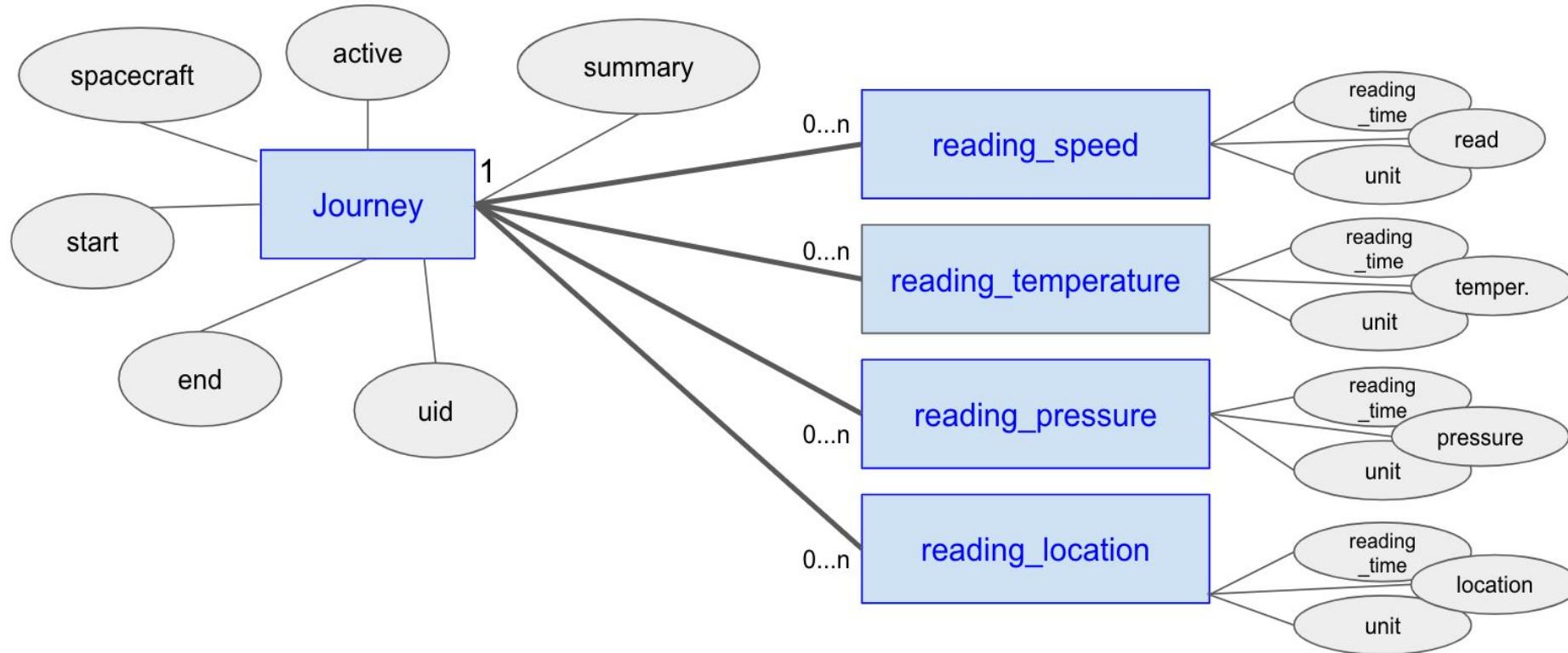
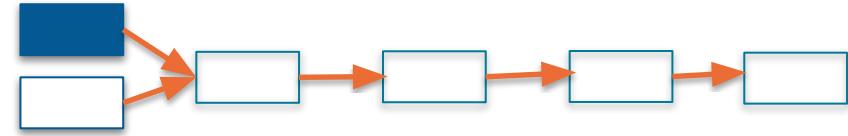
X:	14433	km
Y:	14433	km

Astra Table: spacecraft_location_over_time

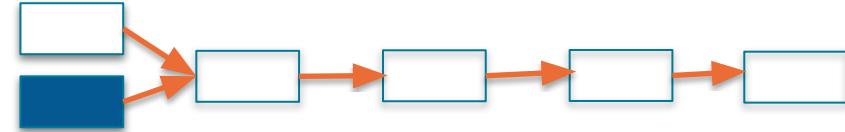
Designing your data model



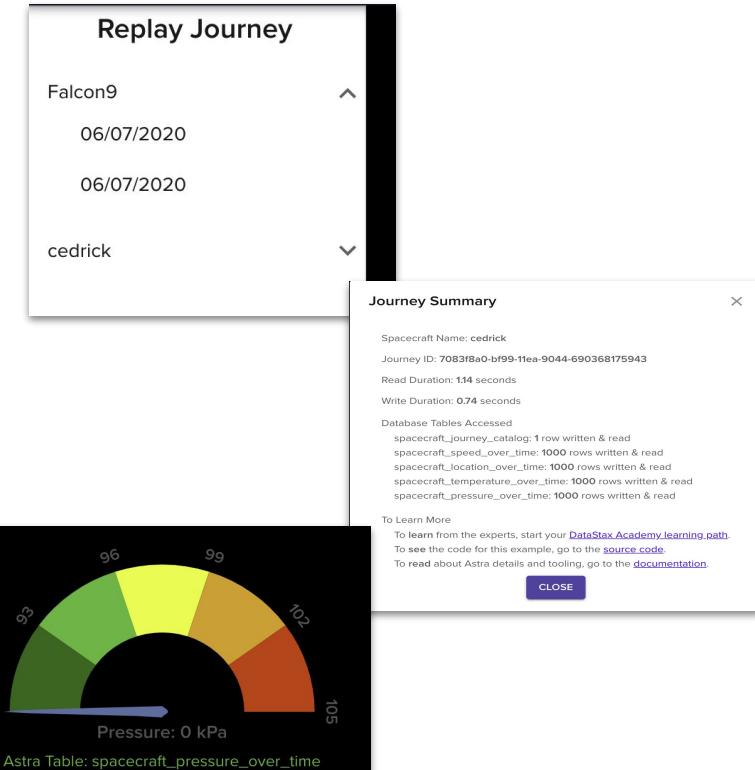
#1 Conceptual Data Model



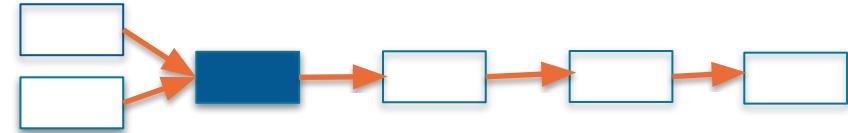
#1 Application Workflow



- **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_journey_catalog

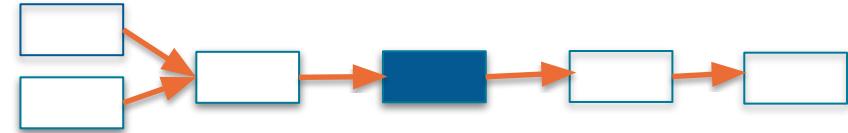
spacecraft_temperature_over_time

spacecraft_speed_over_time

spacecraft_location_over_time

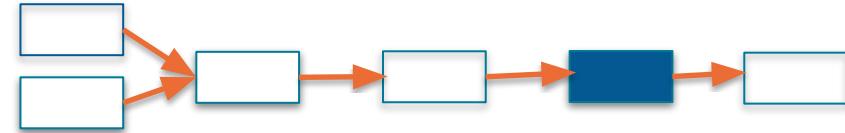
spacecraft_pressure_over_time

#3 Logical Data Model



spacecraft_journey_catalog		spacecraft_temperature_over_time	spacecraft_location_over_time
spacecraft_name	K		
journey_id	C↓		
start			
end			
active			
summary			
		spacecraft_speed_over_time	spacecraft_pressure_over_time
		spacecraft_name K	spacecraft_name K
		journey_id K	journey_id K
		reading_time C↓	reading_time C↓
		speed	pressure
		speed_unit	pressure_unit

#4 Physical Data Model

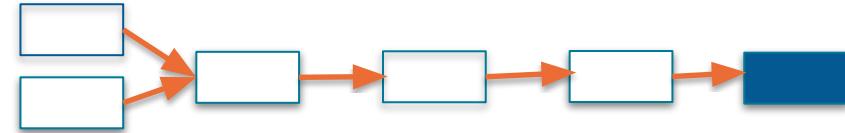


spacecraft_journey_catalog	
spacecraft_name	text
journey_id	timeuuid
start	timestamp
end	timestamp
active	boolean
summary	text

spacecraft_temperature_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

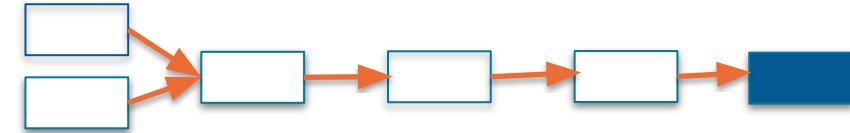
#5 CQL DDL



```
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 (jour
```

```
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

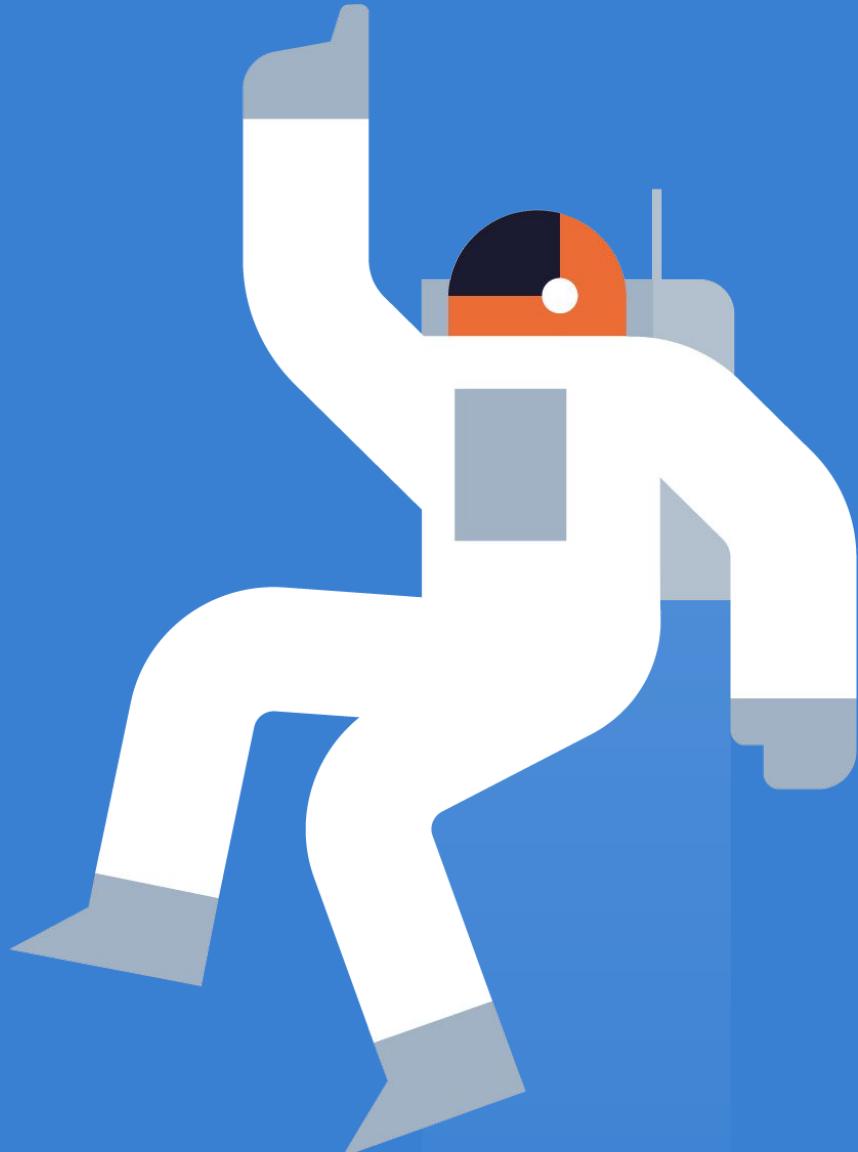


```
CREATE TYPE IF NOT EXISTS location_udt (
    x_coordinate double,
    y_coordinate double,
    z_coordinate double
);
```

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



Introduction to Astra

DataStax

Astra DB

=



Global Scale

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



No Operations

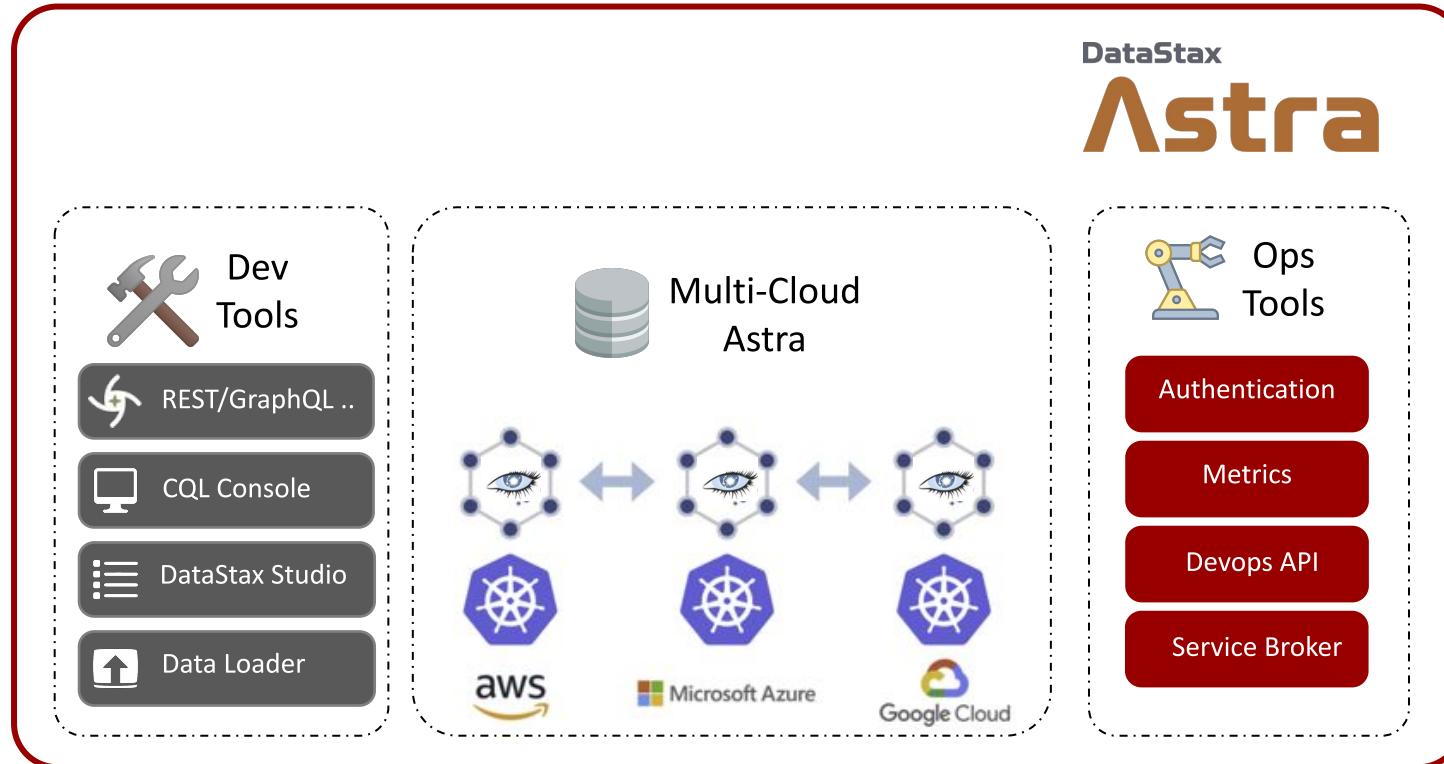
Eliminate the overhead to install, operate, and scale Cassandra.



5 Gig Free Tier

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

Astra Platform



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**

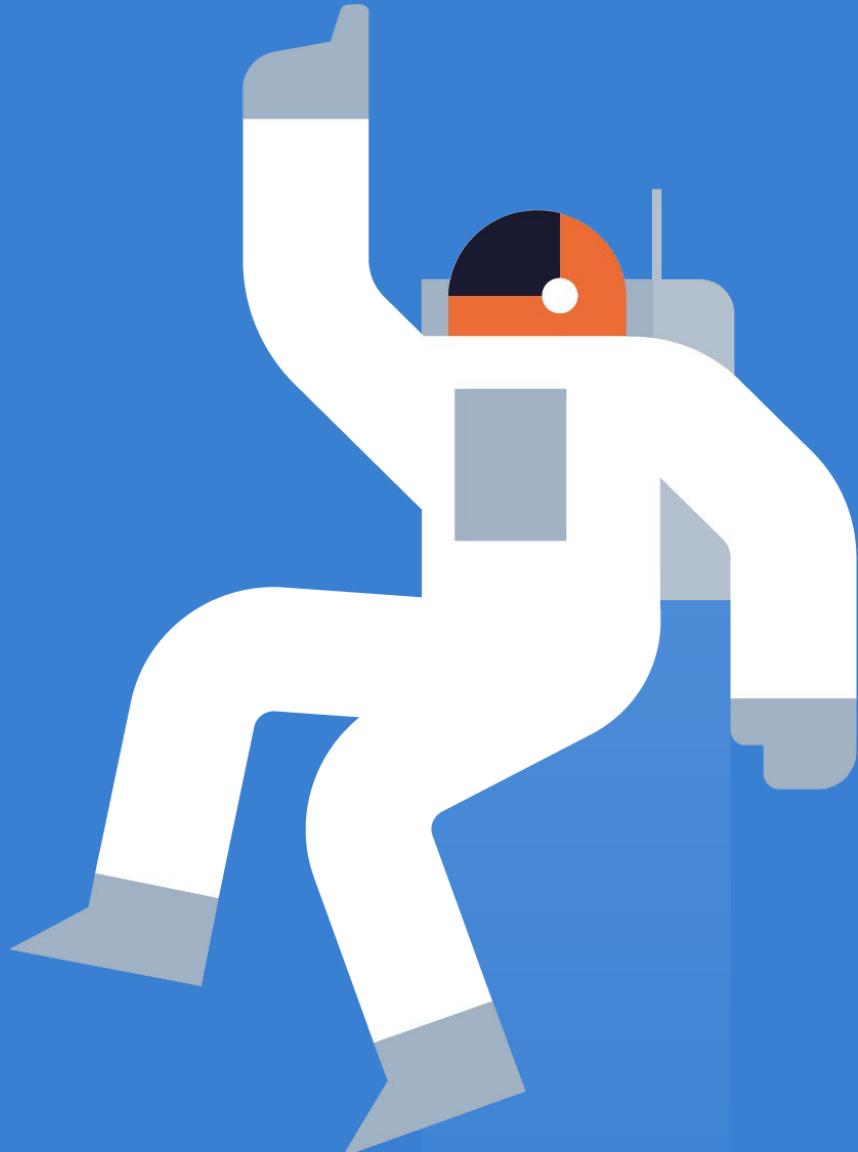
Hands-on Exercise 1:

- 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 data model
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();

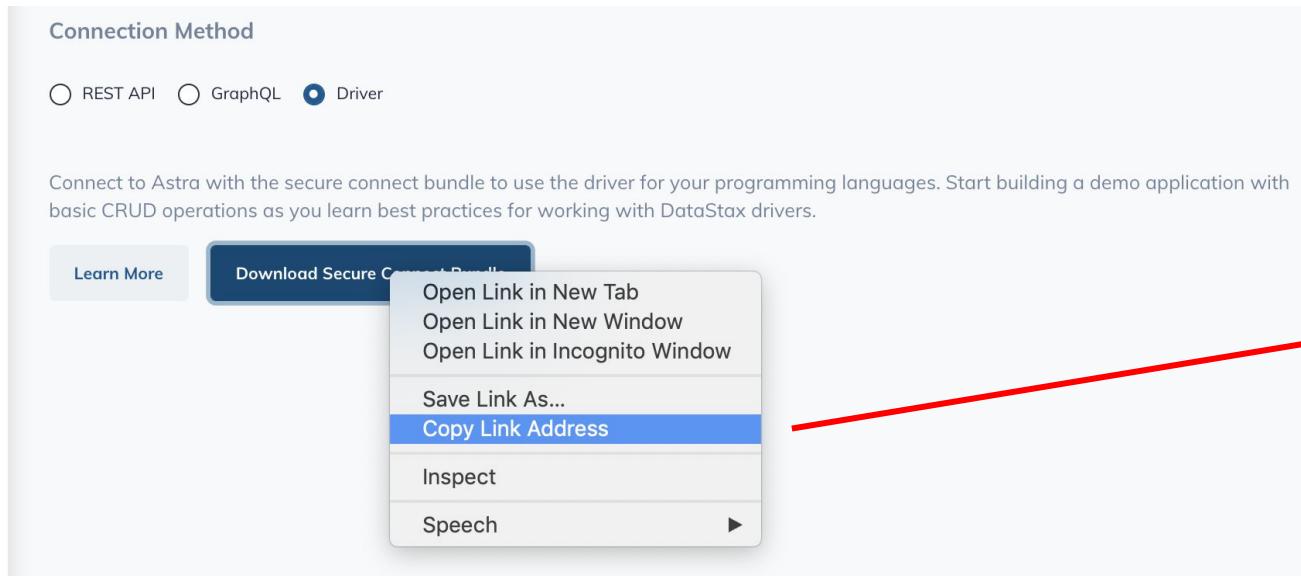
Python: session.shutdown();

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

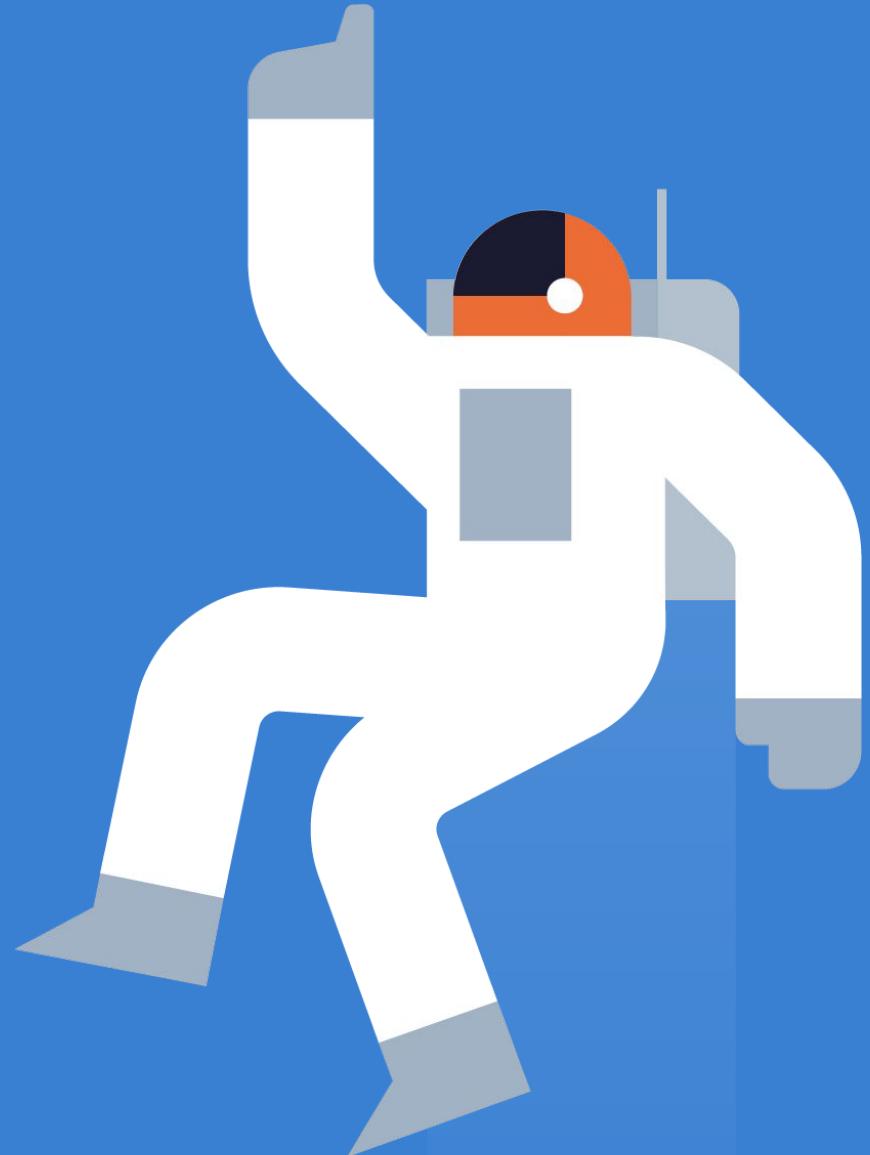


click to copy the link to
clipboard
(refresh page before copy,
link is only 'fresh' for short
time)

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

Application Development CRUD

1. Use Case and data model
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)

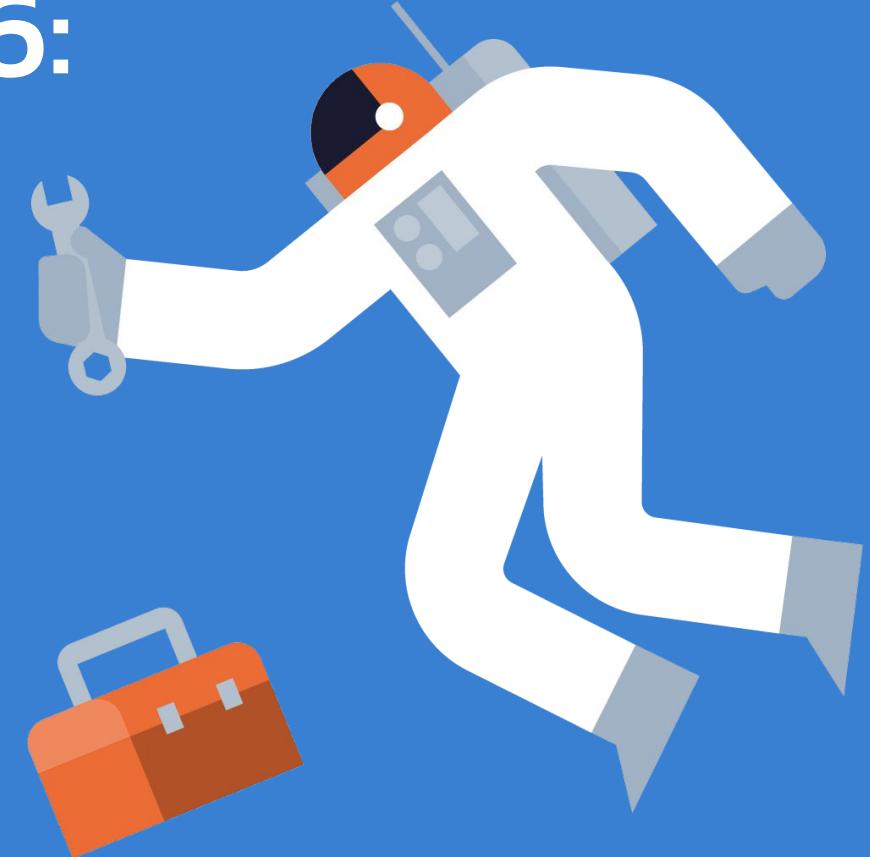
- Recommended to register UDTs with Cluster instance
- When using prepared statements, not necessary to register

```
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")])
```



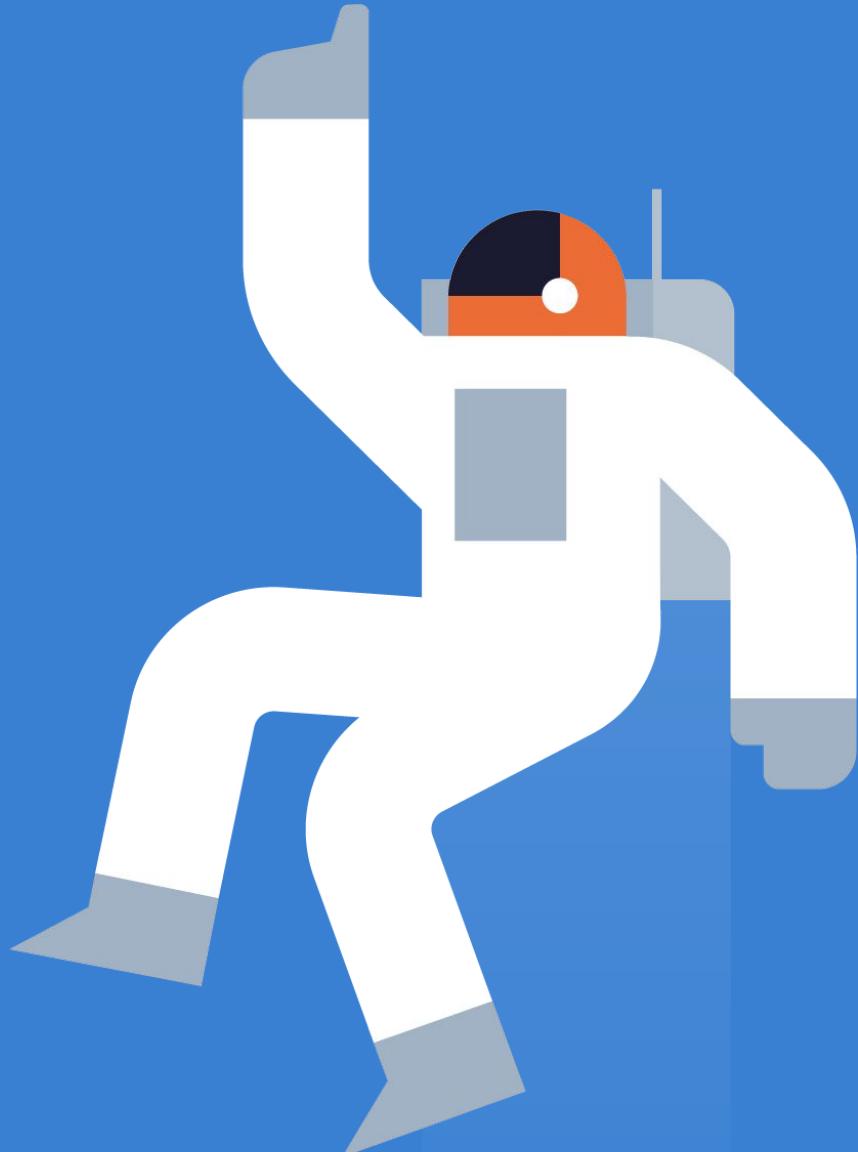
Hands-on Exercise 5 and 6: Inserts with UDTs and batches

- Adapt and run Ex05
- Mark journey complete:
 - Run Ex06



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



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

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



menti.com

67 47 24 6



Available on the iPhone
App Store

GET IT ON
Google play

Developer Resources



Subscribe

LEARN

New hands-on learning at www.datastax.com/dev
Classic courses available at DataStax Academy

ASK/SHARE

Join community.datastax.com
Ask/answer community user questions – share your expertise

CONNECT

Follow us [@DataStaxDevs](#)
We are on Youtube – Twitter – Twitch!

MATERIALS

Slides and exercises for this workshop are available at
<https://github.com/DataStax-Academy/workshop-crud-with-python-and-node>

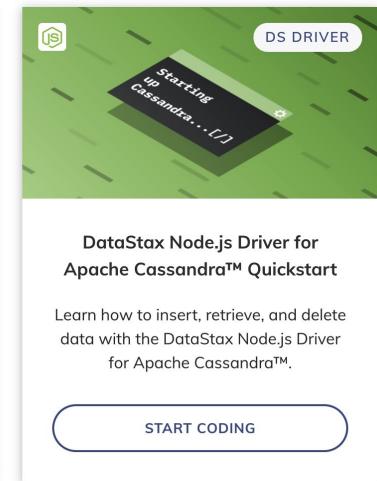
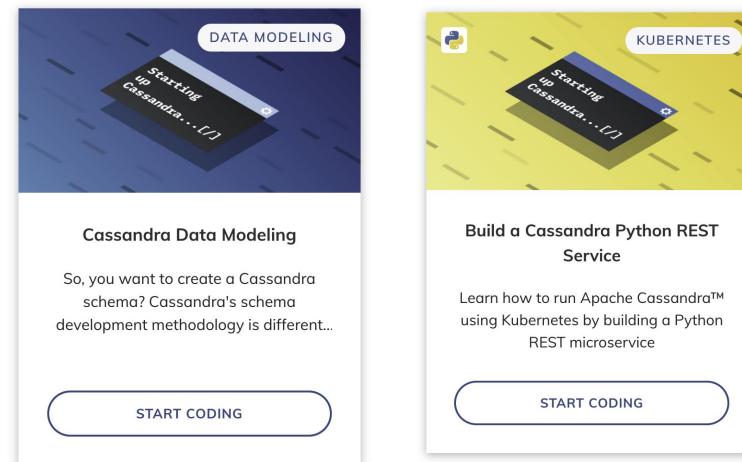
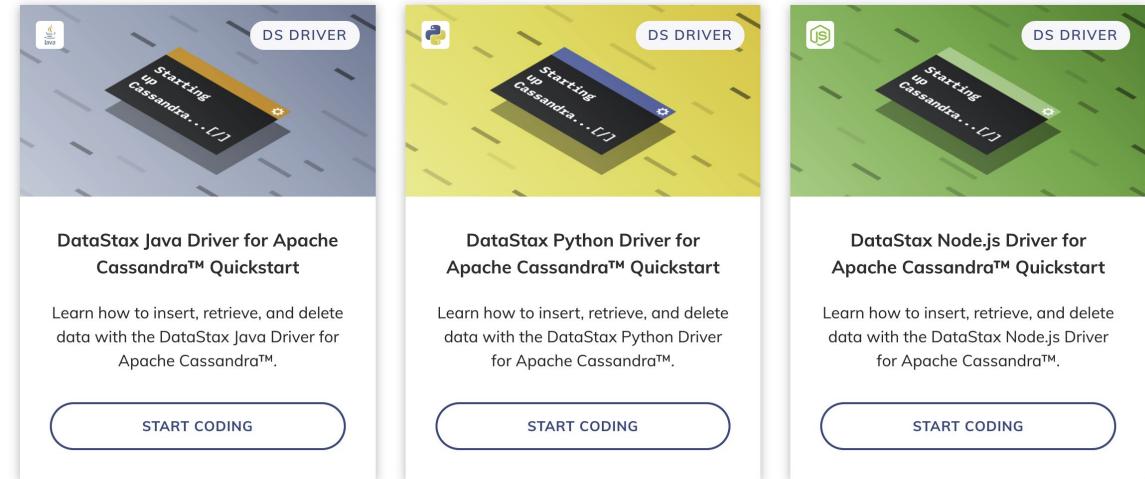
Homework (datastax.com/dev)

[Review week 1] – <https://www.datastax.com/try-it-out>

- Ex1: Cassandra Query Language (CQL)
- Ex2: Pick one of the driver quickstarts
- Bonus: Build a REST Service with Python
- Bonus: Cassandra Data Modeling

[Get ready for week 2]

- Please read <https://www.datastax.com/cloud-native>
- Please read <https://www.datastax.com/dev/kubernetes>



Certifications

<https://www.datastax.com/dev/certifications>



A promotional card with a dark blue header and footer and a white central section. The header features the DataStax eye logo and the word 'cassandra'. The footer features the Kubernetes logo and the word 'kubernetes'. In the center, the text 'COMING SOON! Apache Cassandra Operations in Kubernetes Certification' is displayed in yellow and white. Below this, a paragraph explains the program's purpose: 'As teams work to containerize and deploy applications using Kubernetes, there's increasing interest in running Cassandra in Kubernetes alongside applications as well. We're developing a new certification program to help teams level up their skills to run Cassandra successfully in cloud-native deployments.' It also mentions that the certification will cover topics like running Cassandra in Docker containers, understanding how Cassandra maps to Kubernetes, and deploying Cassandra on Kubernetes using Kubernetes operators and monitoring tools. At the bottom, a button says 'SIGN UP NOW'.

Vouchers (145\$ each), valid 3 months, with 2 attempts will be given to people who apply and register to the 3 episodes.

Thank you





Quiz Time!