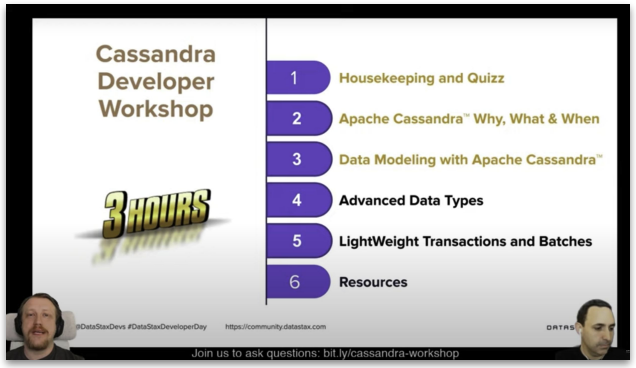

spring
Welcome
Build a Spring Java Microservice
with Apache Cassandra™
cassandra
#CassandraWorkshopSeries
DATASTAX

The Crew

DataStax Developer Advocacy Special Unit

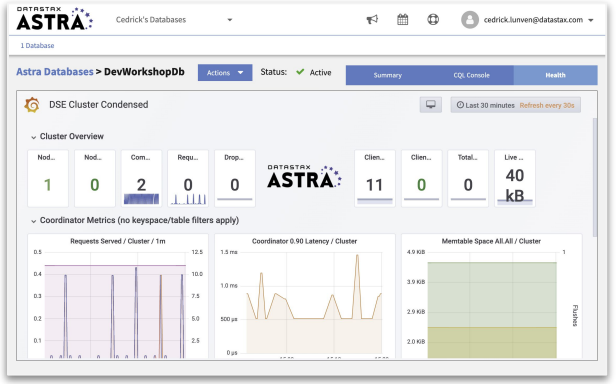
Courses: youtube.com/DataStaxDevs



YouTube

Twitch

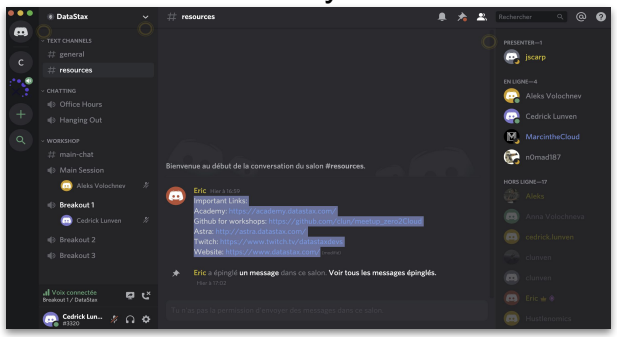
Runtime: dtsx.io/workshop



cassandra

DATASTAX ASTRA

Questions: bit.ly/cassandra-workshop

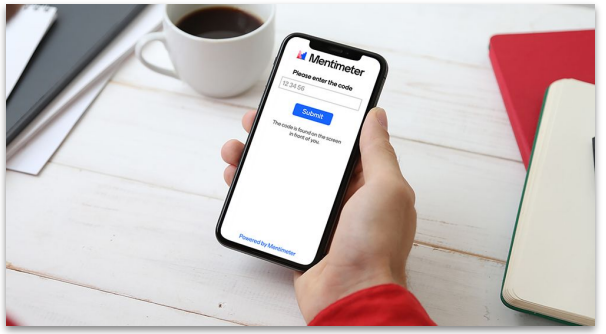


Discord

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 Java language and have a github account.

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

Hands-on exercise material

- **Get your Astra instance here:**
  - http://dtsx.io/workshop

- **Github repo:**
  - https://github.com/DataStax-Academy/Spring-boot-todo-app

- **Driver documentation:**
  - https://docs.datastax.com/en/developer/java-driver/4.9/


menti.com
72 49 38 7
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

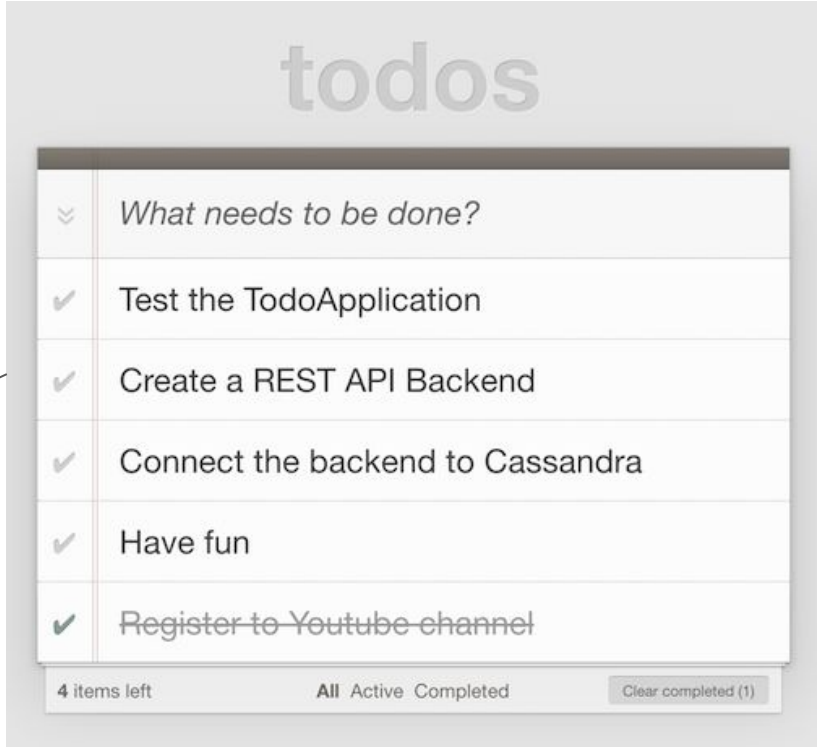
Intro to Cassandra for Developers

- ➔ **Demo !**
- ➔ **Introduction to Astra**
- ➔ **Introduction to Spring and Spring Boot**
- ➔ **Plan the Spring Application**
- ➔ **Working with Datastax drivers**
- ➔ **Build and test the basic CRUD app**

Demo

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



Intro to Cassandra for Developers

- ➔ Demo
- ➔ **Introduction to Astra**
- ➔ Introduction to Spring and Spring Boot
- ➔ Plan the Spring Application
- ➔ Working with Datastax drivers
- ➔ Build and test the basic CRUD app

Introduction to Astra

DATASTAX ASTRA ==

Fully managed Cassandra
Without the ops!

DataStax Astra

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.

10 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: todoapp
  - user name: KVUser
  - password: KVPassword
- Download the secure-connect-bundle, **this is unique to you and your database instance**

Intro to Cassandra for Developers

- ➔ **Demo !**
- ➔ **Introduction to Astra**
- ➔ **Introduction to Spring and Spring Boot**
- ➔ **Plan the Spring Application**
- ➔ **Working with Datastax drivers**
- ➔ **Build and test the basic CRUD app**

Inversion of control : Know only the interface

```java
public interface GreetingServices {
    String sayHello();
}
```

```java
@Component("greeting.fr")
public class GreetingServicesFR implements GreetingServices {
  public String sayHello() { return "Bonjour !"; }
}
```

```java
@Component("greeting.en")
public class GreetingServicesEN implements GreetingServices {
 public String sayHello() { return "Hello !"; }
}
```

```java
@RestController
public class MyAPI {

    @Autowired
    @Qualifier("greeting.en") // <--
    private GreetingServices greetingService;

    @GetMapping
    public String sayHello() {
        return greetingService.sayHello();
    }
}
```

What Spring can do

Microservices

Quickly deliver production-grade features with independently evolvable microservices.

Reactive

Spring's asynchronous, nonblocking architecture means you can get more from your computing resources.

Cloud

Your code, any cloud—we've got you covered. Connect and scale your services, whatever your platform.

Web apps

Frameworks for fast, secure, and responsive web applications connected to any data store.

Serverless

The ultimate flexibility. Scale up on demand and scale to zero when there's no demand.

Event Driven

Integrate with your enterprise. React to business events. Act on your streaming data in realtime.

Batch

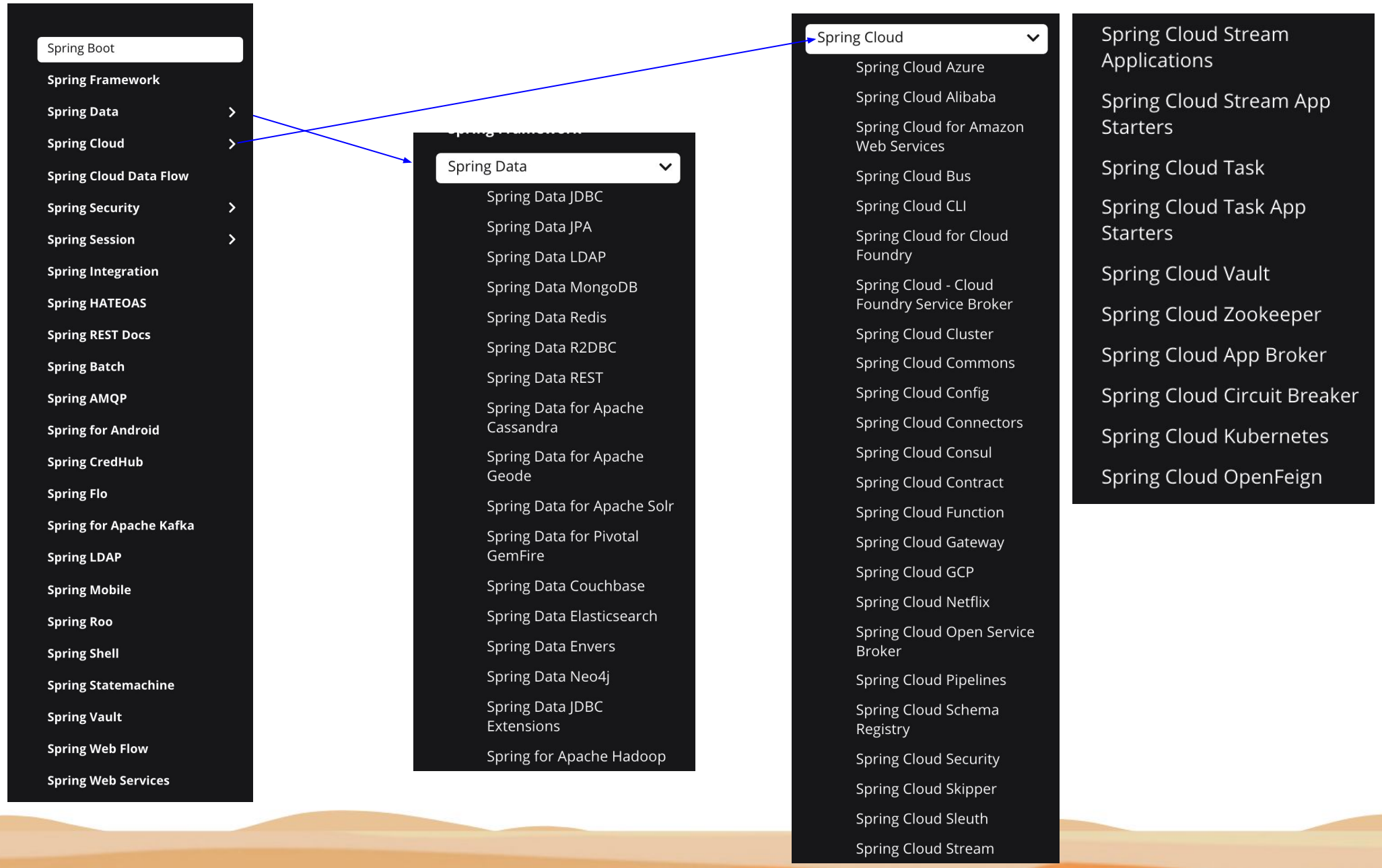
Automated tasks. Offline processing of data at a time to suit you.

spring

spring.io

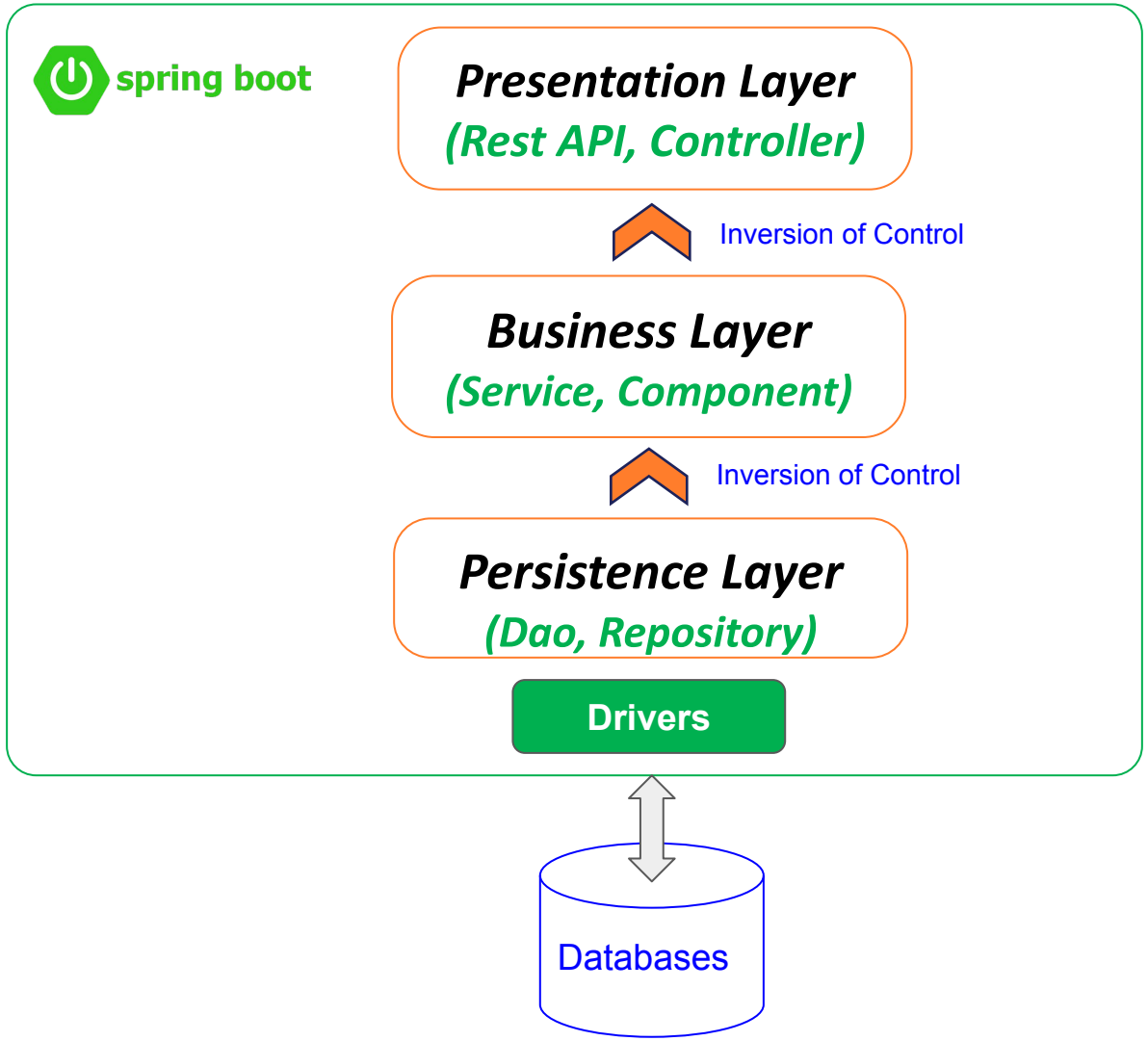
*The Spring Framework provides a comprehensive programming and configuration model for **modern Java-based** enterprise applications - on any kind of deployment platform.*
spring.io

Core technologies	Dependency injection, events, resources, i18n, validation, data binding, AOP.
Web applications	Model View Controller (MVC), Rest, Reactive,
Data Access	Transactions, DAO support, JDBC, Object Mapping
Testing	Mock objects, TestContext
Integrations	Remoting, JMS, JCA, JMX, email, tasks, scheduling, cache.
Languages	Kotlin, Groovy, dynamic languages.


Spring Boot
Spring Framework
Spring Data
Spring Cloud
Spring Cloud Data Flow
Spring Security
Spring Session
Spring Integration
Spring HATEOAS
Spring REST Docs
Spring Batch
Spring AMQP
Spring for Android
Spring CredHub
Spring Flo
Spring for Apache Kafka
Spring LDAP
Spring Mobile
Spring Roo
Spring Shell
Spring Statemachine
Spring Vault
Spring Web Flow
Spring Web Services
Spring Data
Spring Data JDBC
Spring Data JPA
Spring Data LDAP
Spring Data MongoDB
Spring Data Redis
Spring Data R2DBC
Spring Data REST
Spring Data for Apache Cassandra
Spring Data for Apache Geode
Spring Data for Apache Solr
Spring Data for Pivotal GemFire
Spring Data Couchbase
Spring Data Elasticsearch
Spring Data Envers
Spring Data Neo4j
Spring Data JDBC Extensions
Spring for Apache Hadoop
Spring Cloud
Spring Cloud Azure
Spring Cloud Alibaba
Spring Cloud for Amazon Web Services
Spring Cloud Bus
Spring Cloud CLI
Spring Cloud for Cloud Foundry
Spring Cloud - Cloud Foundry Service Broker
Spring Cloud Cluster
Spring Cloud Commons
Spring Cloud Config
Spring Cloud Connectors
Spring Cloud Consul
Spring Cloud Contract
Spring Cloud Function
Spring Cloud Gateway
Spring Cloud GCP
Spring Cloud Netflix
Spring Cloud Open Service Broker
Spring Cloud Pipelines
Spring Cloud Schema Registry
Spring Cloud Security
Spring Cloud Skipper
Spring Cloud Sleuth
Spring Cloud Stream
Spring Cloud Stream Applications
Spring Cloud Stream App Starters
Spring Cloud Task
Spring Cloud Task App Starters
Spring Cloud Vault
Spring Cloud Zookeeper
Spring Cloud App Broker
Spring Cloud Circuit Breaker
Spring Cloud Kubernetes
Spring Cloud OpenFeign

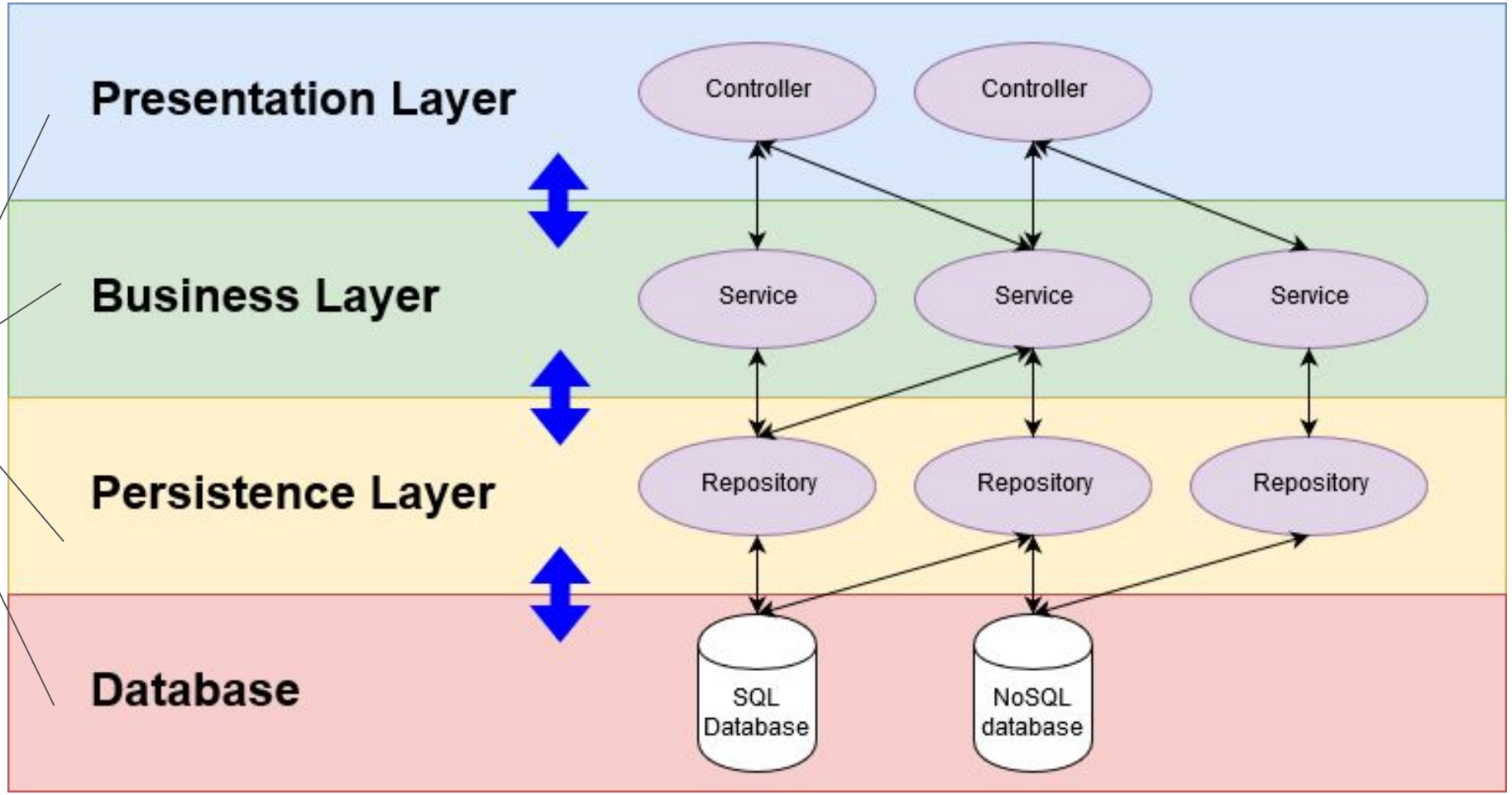
Spring Boot

- Create **stand-alone** Spring applications (no need to deploy WAR files)
- **Simplify** your build configuration
- **Automatically** configure Spring and 3rd party libraries
- Production-ready features such as
  - metrics
  - health checks
  - externalized configuration



Vocabulary

spring boot



Intro to Cassandra for Developers

- ➔ **Demo**
- ➔ **Introduction to Astra**
- ➔ **Introduction to Spring and Spring Boot**
- ➔ **Plan the Spring Application**
- ➔ **Working with Datastax drivers**
- ➔ **Build and test the basic CRUD app**

What is a microservice?

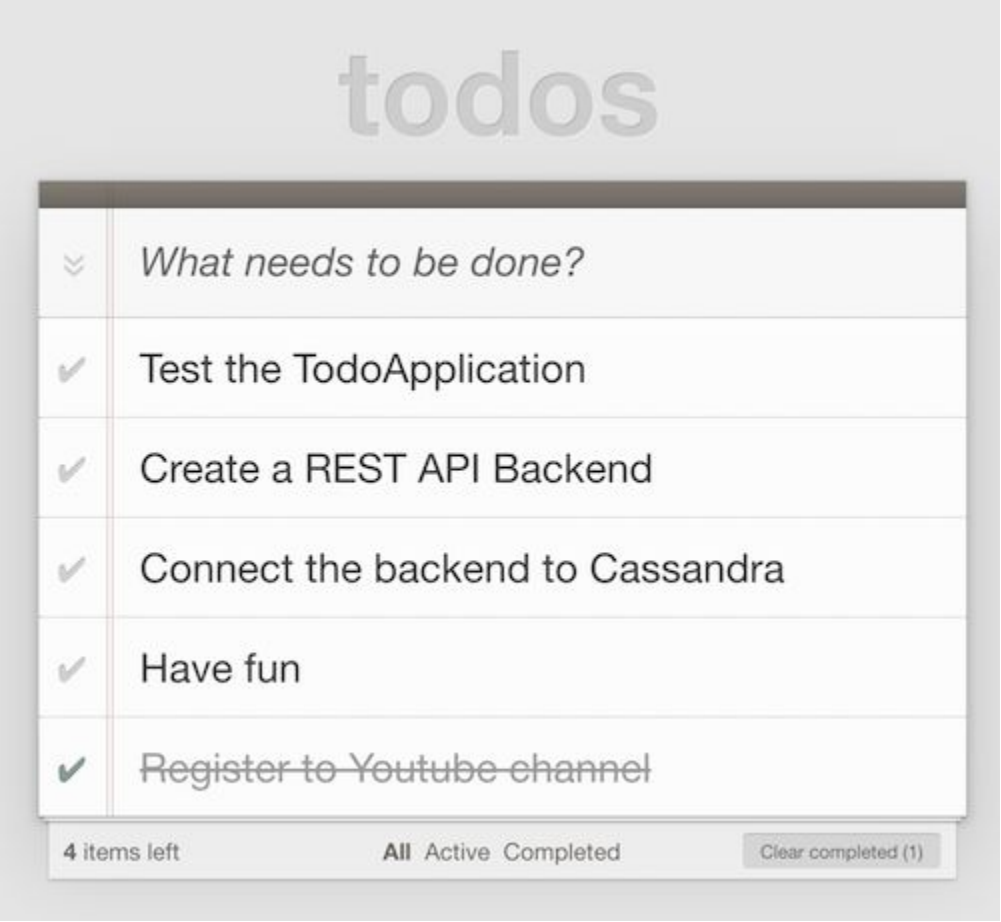
Microservices - also known as the microservice architecture - is an architectural style that structures an application as a collection of services that are

- Highly maintainable and testable
- Loosely coupled
- Independently deployable
- Organized around business capabilities
- Owned by a small team

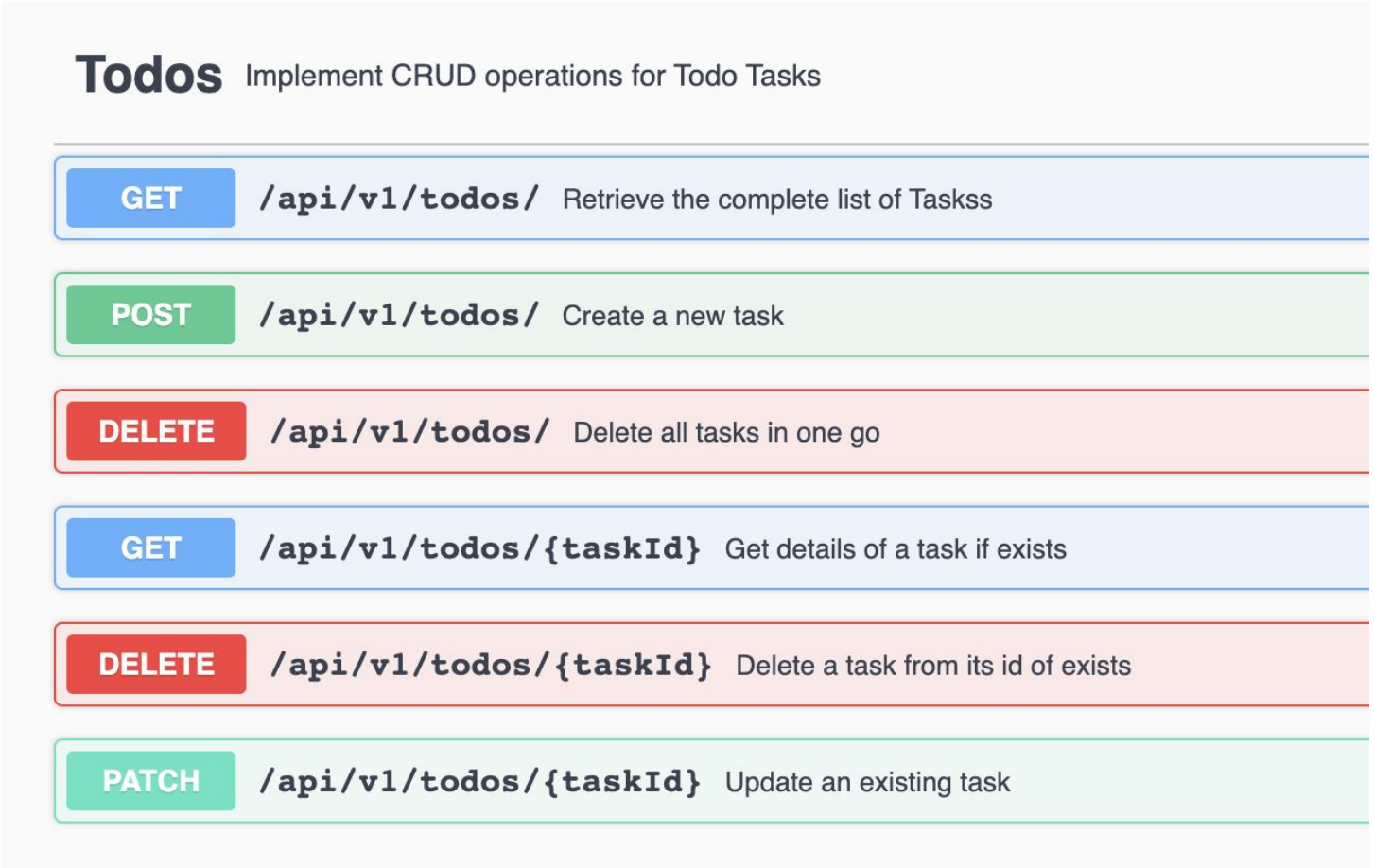
The microservice architecture enables the rapid, frequent and reliable delivery of large, complex applications. It also enables an organization to evolve its technology stack.

source: https://microservices.io/

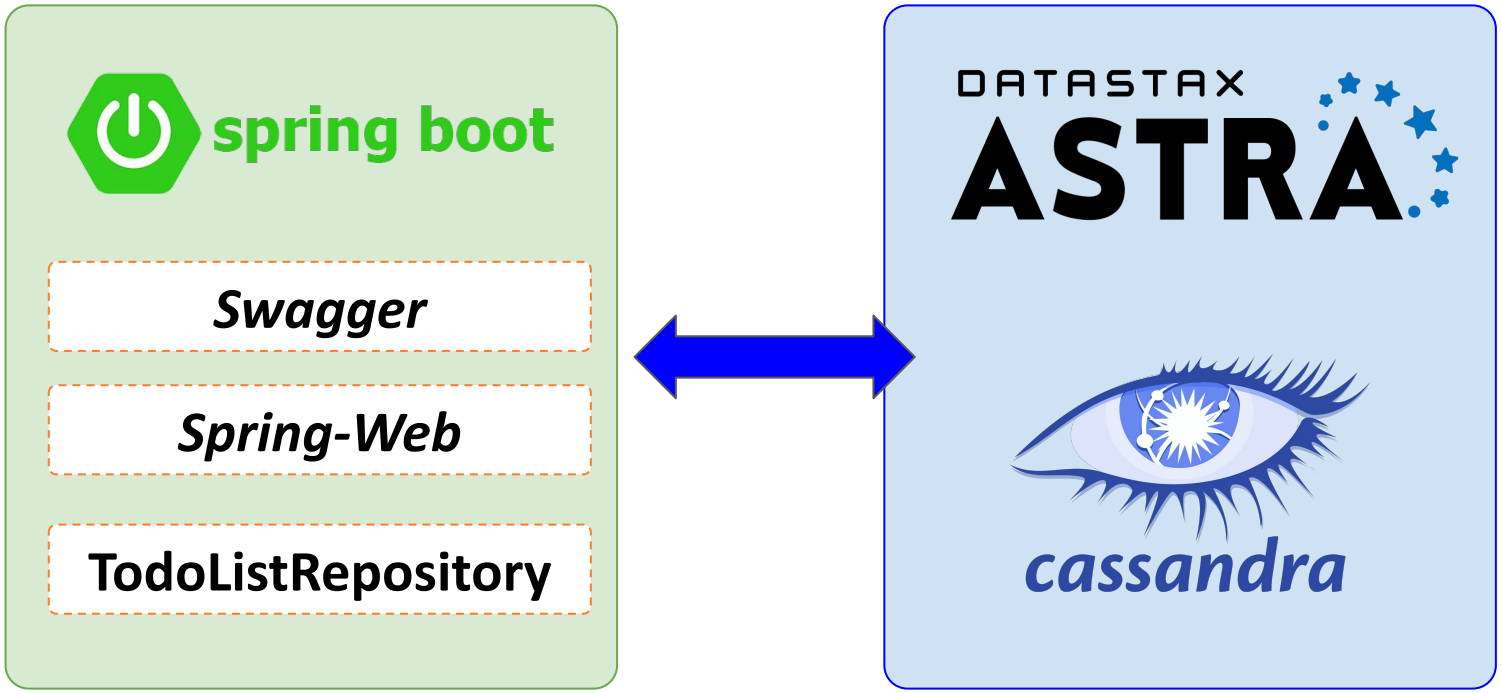
What service are we building? A ToDo Service



The REST Api

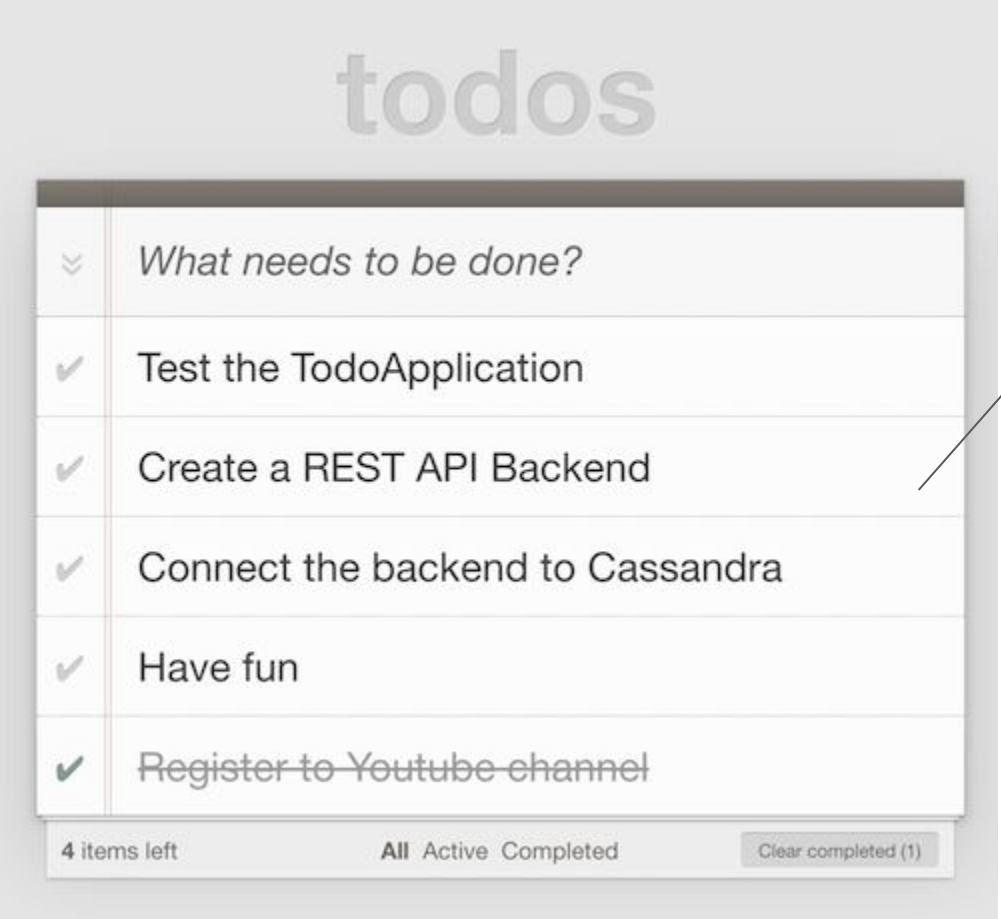

Todos
Implement CRUD operations for Todo Tasks
GET
/api/v1/todos/
Retrieve the complete list of Taskss
POST
/api/v1/todos/
Create a new task
DELETE
/api/v1/todos/
Delete all tasks in one go
GET
/api/v1/todos/{taskId}
Get details of a task if exists
DELETE
/api/v1/todos/{taskId}
Delete a task from its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

The full ToDo app architecture



todobackend-cassandra

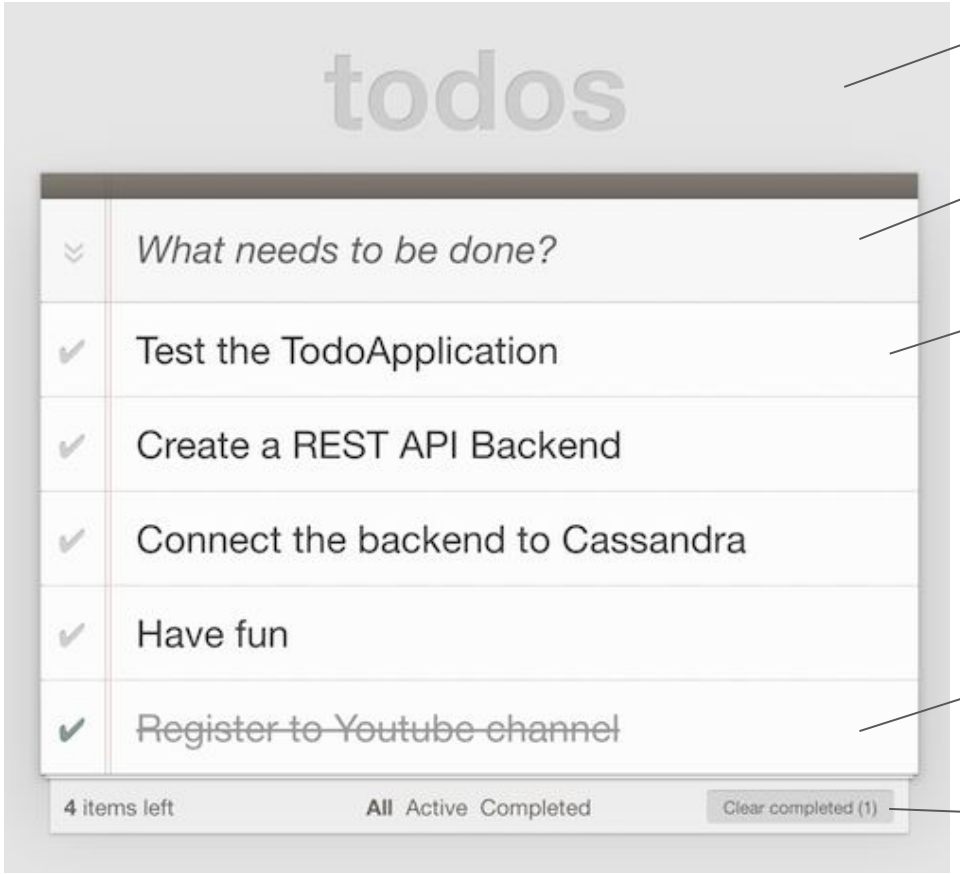
The main data entity? Tasks



Tasks

- Task UUID
- title
- completed flag
- order

What are the queries? CRUD for ToDos



List all tasks

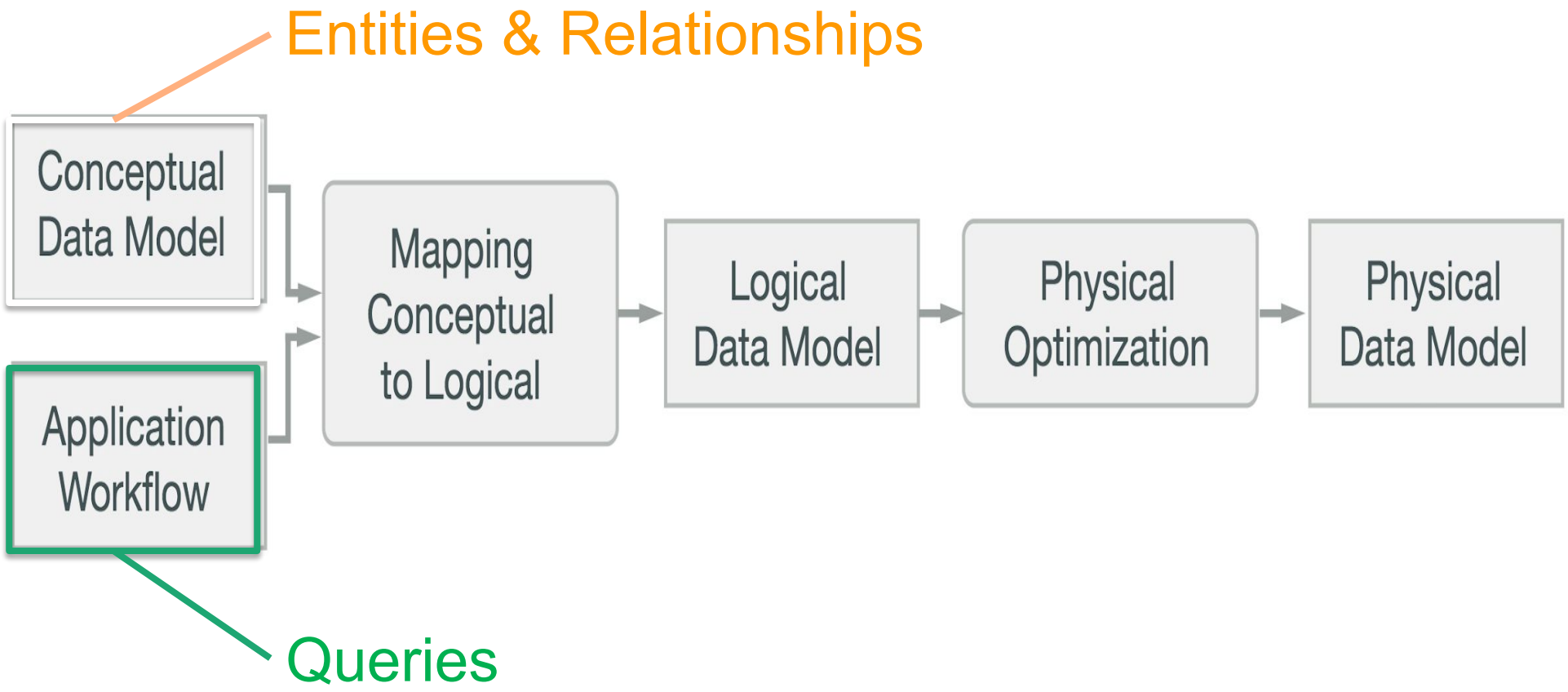
Create a new Task

Mark a task as completed/uncomplete

Delete a task

Delete all

Designing the Cassandra datamodel



The final task model

todo_tasks		
uid	UUID	K
title	TEXT	
completed	BOOLEAN	
offset	INT	

```
CREATE TABLE todoapp.todo_tasks (
    uid         uuid,
    title       text,
    completed   boolean,
    offset      int,
    PRIMARY KEY ((uid))
 );
```

Intro to Cassandra for Developers

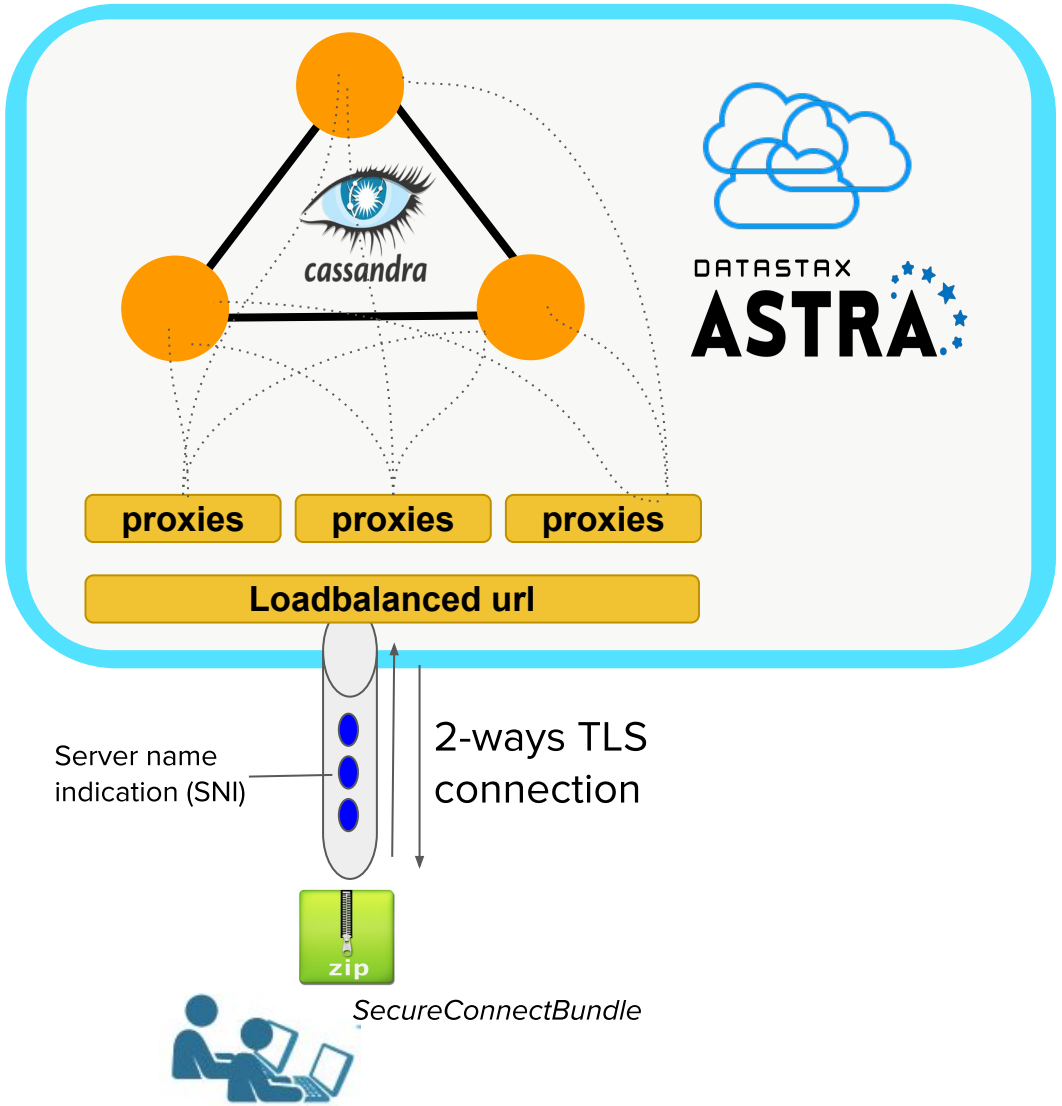
- ➔ **Demo**
- ➔ **Introduction to Astra**
- ➔ **Introduction to Spring and Spring Boot**
- ➔ **Plan the Spring Application**
- ➔ **Working with Datastax drivers**
- ➔ **Build and test the basic CRUD app**

Connecting the REST Api to Astra

- Use the Datastax drivers
- Download and use the secure connect bundle
- Pass user name, password, keyspace name and secure connect bundle to build the cql session
- Astra handles the load balancing to the multinode Cassandra cluster

Contact points with Astra

- Download and use the secure connect bundle
- Pass user name, password, keyspace name and secure connect bundle to build the cql session



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

Working with the driver: Open a session

```
// Explicit Settings

CqlSession cqlSession = CqlSession.builder()

 .withCloudSecureConnectBundle(Paths.get("/tmp/secure-connect-bundle.zip"))

 .withKeyspace("todoapp")

 .withAuthCredentials("TDUser", "TDPassword")

 .build();
```


// Delegate all configuration to file or default

File based driver configuration

application.conf

Based on Typesafe Config

Attributes are grouped into **basic** and **advanced** categories

A reference file **`reference.conf`** provide default values embedded in the jar file. Can be override with key in application.conf.

Driver searches **`application.conf`** in the classpath

```
datastax-java-driver {
  basic {...
    session-keyspace = killrvideo
    cloud {
      secure-connect-bundle = ...
    }
 }
  advanced {
    auth-provider {
      class = PlainTextAuthProvider
      username = KVUser
      password = KVPassword
    } ...
```

Open a session with app.conf

```
// Delegate all configuration to file or default

CqlSession cqlSession = CqlSession.builder().build();
```

Important about the CqlSession

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

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

Hands-on Exercise 1:
Test the connection to Astra

```
mvn test
-Dtest=com.datastax.examples.ConnectivityToAstraExplicitTest#should_connect_to_Astra_static
```

```
mvn test
-Dtest=com.datastax.examples.ConnectivityToAstraWithConfTest#should_connect_to_Astra_withConfig
```

Working with the driver: Execute queries

- First job of CqlSession is to execute queries using the **`execute`** method.

```
cqlSession.execute("SELECT * FROM todoapp.tasks");
```

Working with the driver: Simple Statements

```
Statement statement = …

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

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

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

cqlSession.execute(statement);
```

Working with the driver: Prepare and bind a statement

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

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

BoundStatement bound = ps.bind(5);

cqlSession.execute(bound);
```

Working with the driver: ResultSets

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

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

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

// Data: NOT ALL DATA RETRIEVED IMMEDIATELY (only when needed .next())
Iterator<Row> iterRow = rs.iterator();
int itemsFirstCall = rs.getAvailableWithoutFetching();
```

Parsing ResultSets

```
// We know there is a single row (eg: count)
Row singleRow = resultSet.one();

// We know there are not so many results we can get all (fetch all pages)
List<Row> allRows = resultSet.all();

// Browse iterable
for(Row myRow : resultSet.iterator()) {
   // .. Parsing rows
}

// Use Lambda
rs.forEach(row -> { row.getColumnDefinitions(); });
```

Parsing Rows

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

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

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

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

Hands-on Exercise 2:
Set up schema and test inserts

```
mvn test -Dtest=com.datastax.examples.CreateSchemaInAstraTest#should_create_expected_table
```

```
mvn test -Dtest=com.datastax.examples.CRUDWithAstraTest#test_Insert
```

Intro to Cassandra for Developers

- ➔ **Demo**
- ➔ **Introduction to Astra**
- ➔ **Introduction to Spring and Spring Boot**
- ➔ **Plan the Spring Application**
- ➔ **Working with Datastax drivers**
- ➔ **Build and test the basic CRUD app**

Building the REST API with Spring Boot

```
public interface TodoRepository {
     //schema definition
     //query definitions
}
```

```
@Repository("todobackend.repo.cassandra-driver")
public class TodoListRepositoryCassandraDriverImpl
implements TodoListRepository {

    @Autowired
    public CqlSession cqlSession;
```

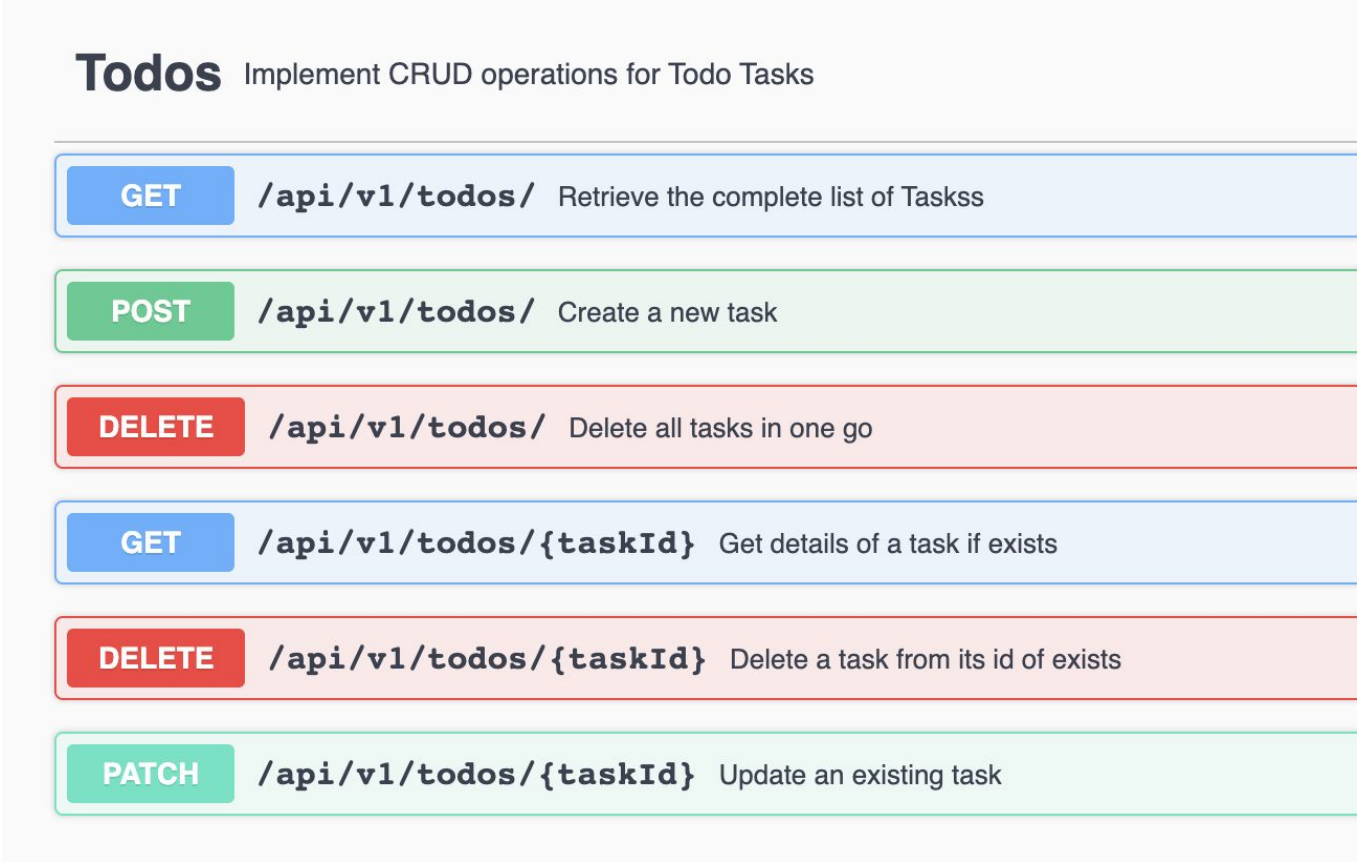
```
@Configuration
public class CassandraDriverConfig {
    @Bean
    public CqlSession cqlSession() {
        return CqlSession.builder().build();
    }
}
```

```
@RestController
@RequestMapping("/api/v1/todos")
public class TodoListRestController {
     @Autowired
     @Qualifier("todobackend.repo.cassandra-dri
     ver")
     private TodoListRepository todoRepository;
...
}
```

Hands-on Exercise 3: Configure and run the application

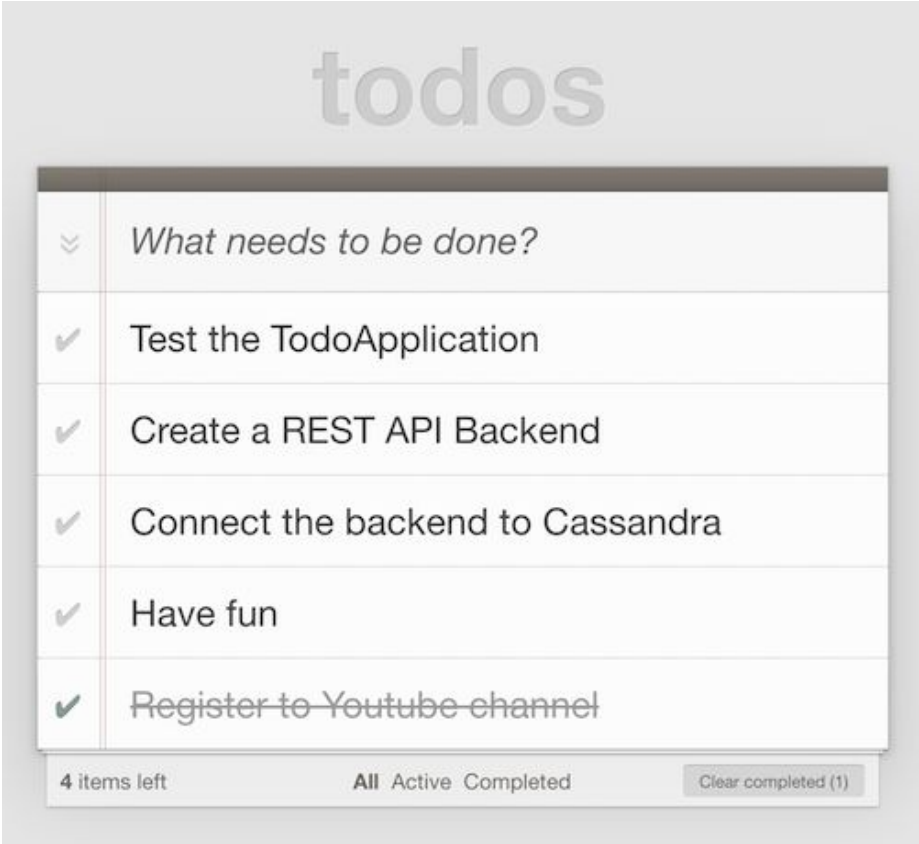
```
mvn spring-boot:run
```

Hands-on Exercise 4:
Test CRUD with Swagger




menti.com
72 49 38 7
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

Bonus Exercise:
Test with Todo MVC, client GUI and API specs



Learn more?

Micro-service **REST**

https://bit.ly/31RL62I

Micro-service **GraphQL**

https://bit.ly/2MVicup

Micro-service **GRPC**

https://bit.ly/2pofk0b

Micro-service **Kafka**

https://bit.ly/2JwsFdM

Reactive with Webflux

https://bit.ly/34ePzhL

Micro-service **Serverless**

https://bit.ly/31VQz8G