Cassandra & Redis

Diego Pacheco

About me...

- ❑ Cat's Father
- ❑ Principal Software Architect
- ❑ Agile Coach
- ❑ SOA/Microservices Expert
- ❑ DevOps Practitioner
- ❑ Speaker
- ❑ Author

diegopacheco

@diego_pacheco

http://diego-pacheco.blogspot.com.br/

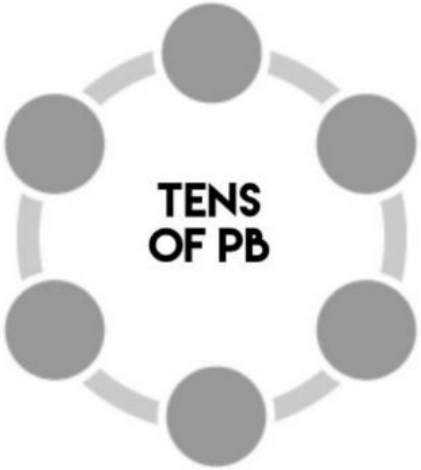
https://diegopacheco.github.io/

Cassandra

Cassandra

- FOSS
- NoSQL Database
- Low Latency
- High Throughput
- Battle tested by Netflix, Apple, Facebook.
- AP on CAP - Favors availability over consistency.
- Multi-region support.
- Great software but hard to operate.

Cassandra Battle Tested



NETFLIX

Spotify

Cassandra ~ Netflix 2013

NETFLIX

Cassandra ~ Apple 2016

Massive

- Cassandra Nodes: 75,000+
- Data Size: 10s of PetaBytes
- Ops/sec: In the millions
- Largest Cluster: 1000+
- Versions: 1.2.x and 2.0.x

Cassandra ~ Spotify 2019

Cassandra at Spotify

100+
production clusters

3000+
production machines

150+ TB
of unique data

4 million
queries per second
at peak

Google Cloud

Spotify

Cassandra ~ Data Model

Cassandra	RDBMS Equivalent
KEYSPACE	DATABASE/SCHEMA
COLUMN FAMILY	TABLE
ROW	ROW
FLEXIBLE COLUMNS	DEFINED COLUMNS

How to have Success with Cassandra?

- ❑ Understand system / service / app requirements
- ❑ Identify your data access patterns
- ❑ Model around the access patterns
- ❑ Denormalization is your friend (by* query tables)
- ❑ Benchmark
- ❑ Tuneup
- ❑ Repeat

Netflix Benchmark: 1M writes/sec

Benchmark: 1 million writes per second

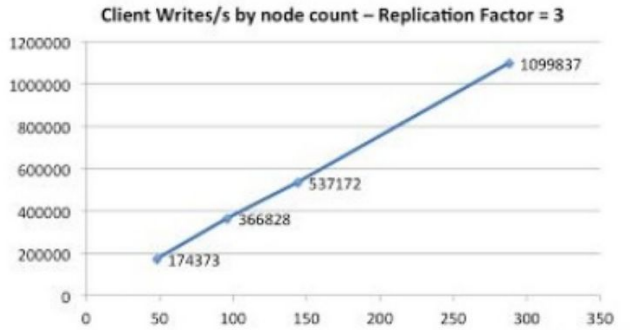
Per Node Activity

Per Node	48 Nodes	96 Nodes	144 Nodes	288 Nodes
Per Server Writes/s	10,900 w/s	11,460 w/s	11,900 w/s	11,456 w/s
Mean Server Latency	0.0117 ms	0.0134 ms	0.0148 ms	0.0139 ms
Mean CPU %Busy	74.4 %	75.4 %	72.5 %	81.5 %
Disk Read	5,600 KB/s	4,590 KB/s	4,060 KB/s	4,280 KB/s
Disk Write	12,800 KB/s	11,590 KB/s	10,380 KB/s	10,080 KB/s
Network Read	22,460 KB/s	23,610 KB/s	21,390 KB/s	23,640 KB/s
Network Write	18,600 KB/s	19,600 KB/s	17,810 KB/s	19,770 KB/s

Node specification – Xen Virtual Images, AWS US East, three zones
- Cassandra 0.8.6, CentOS, SunJDK6
- AWS EC2 m1 Extra Large – Standard price $ 0.68/Hour
- 15 GB RAM, 4 Cores, 1Gbit network
- 4 internal disks (total 1.6TB, striped together, md, XFS)

NETFLIX

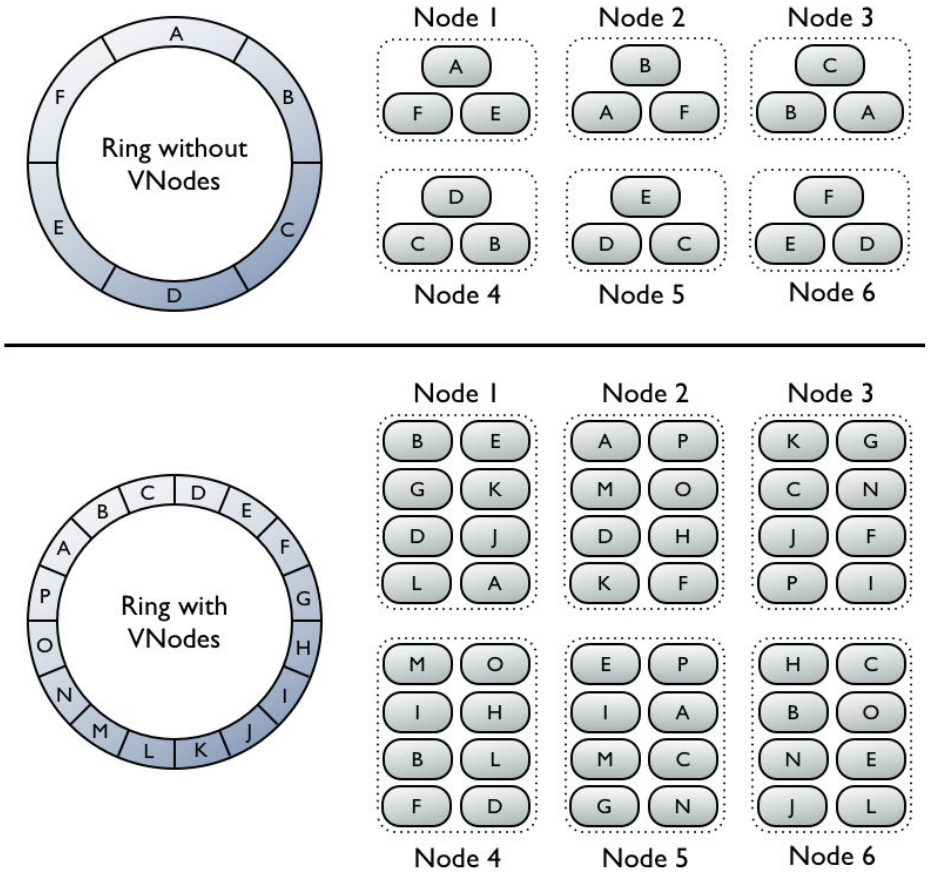
Scale-Up Linearity



NETFLIX

http://techblog.netflix.com/2011/11/benchmarking-cassandra-scalability-on.html

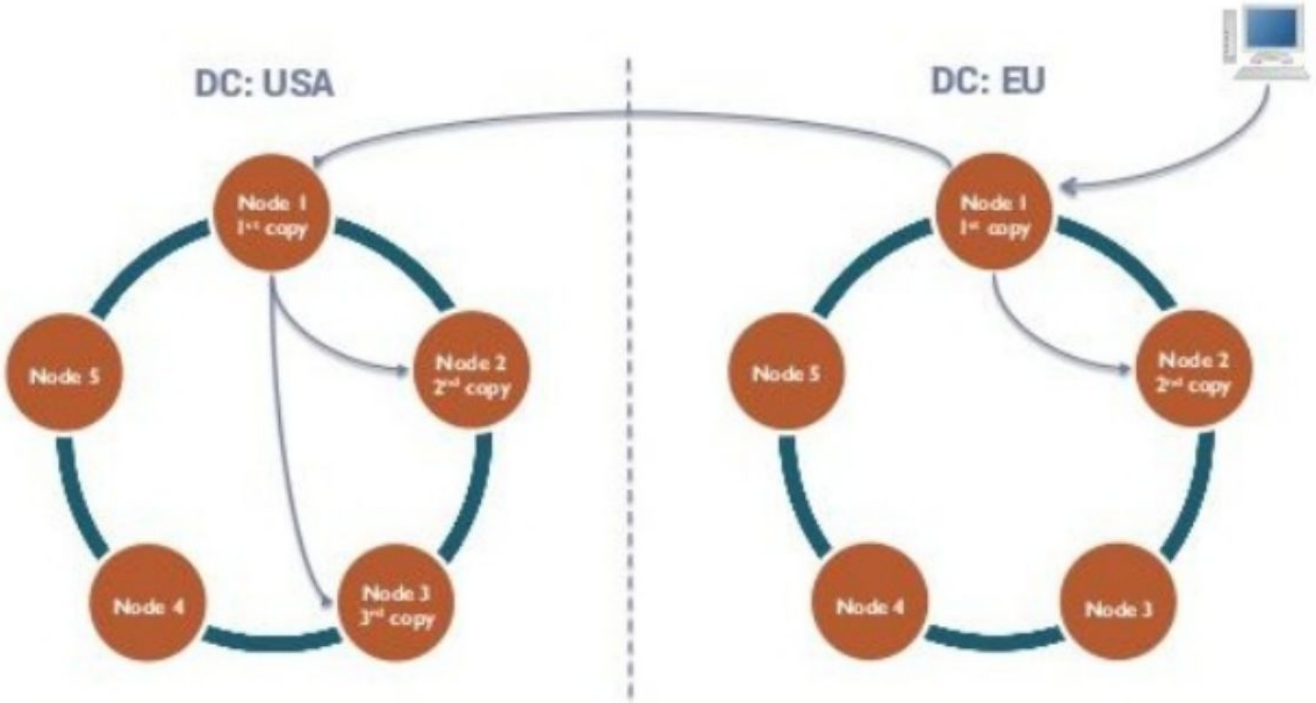
Cassandra Cluster



- Master less
- Anti-entropy HH
- Consistent Hashing
  - Murmur3
- Virtual Nodes
- Dynamic Scale out

Cassandra Data Replication

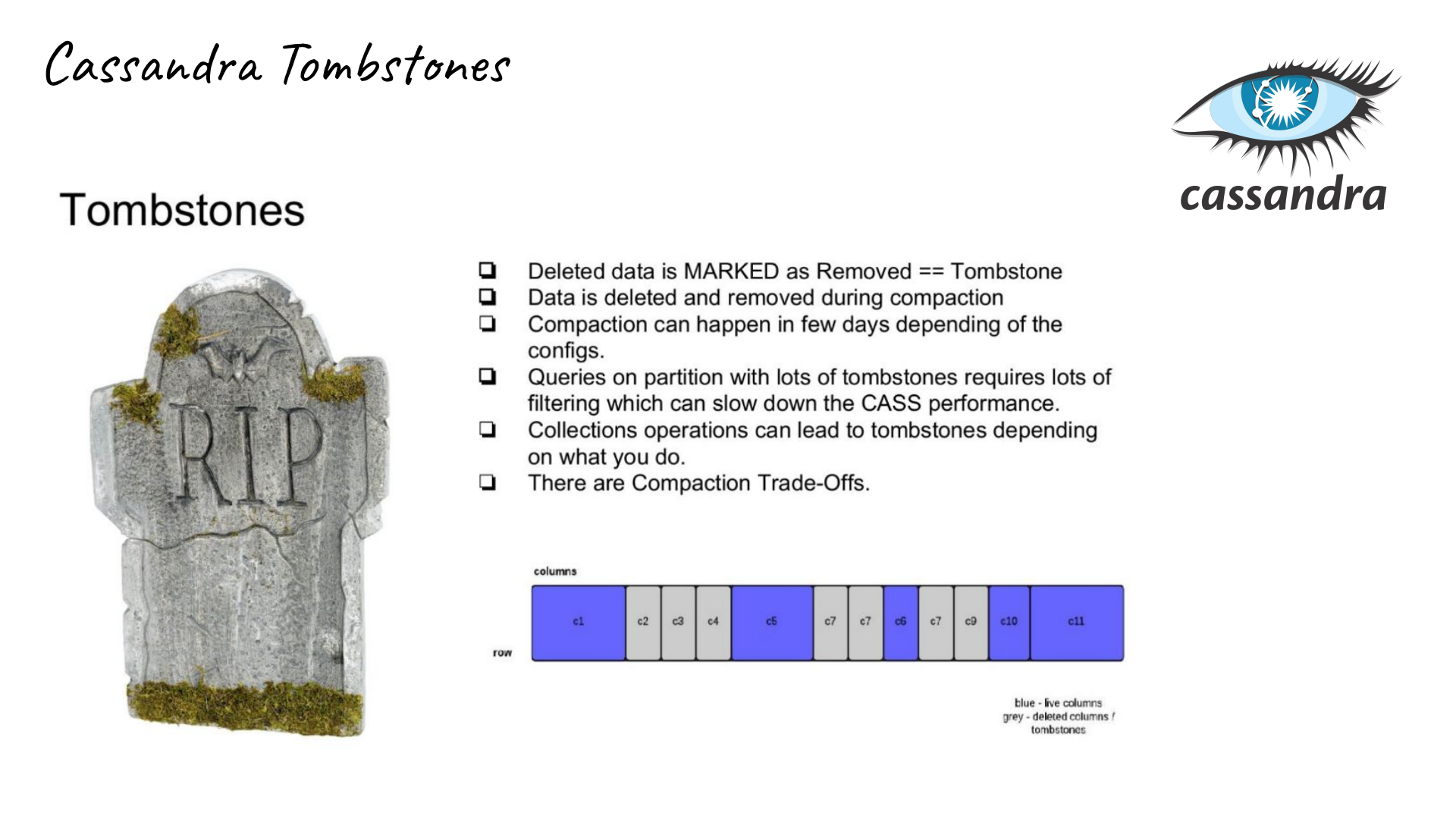
```
CREATE KEYSPACE johnny WITH REPLICATION =
{'class':'NetworkTopologyStrategy', 'USA':3, 'EU': 2};
```



Cassandra Tunable Consistency

- Tunable for Read / Writes
- Consistency VS Availability
  - One, Two, Three,
  - QUORUM(majority n/2+1), Local_Quorun
  - Each_Quorun
  - Local_one
  - All, Any

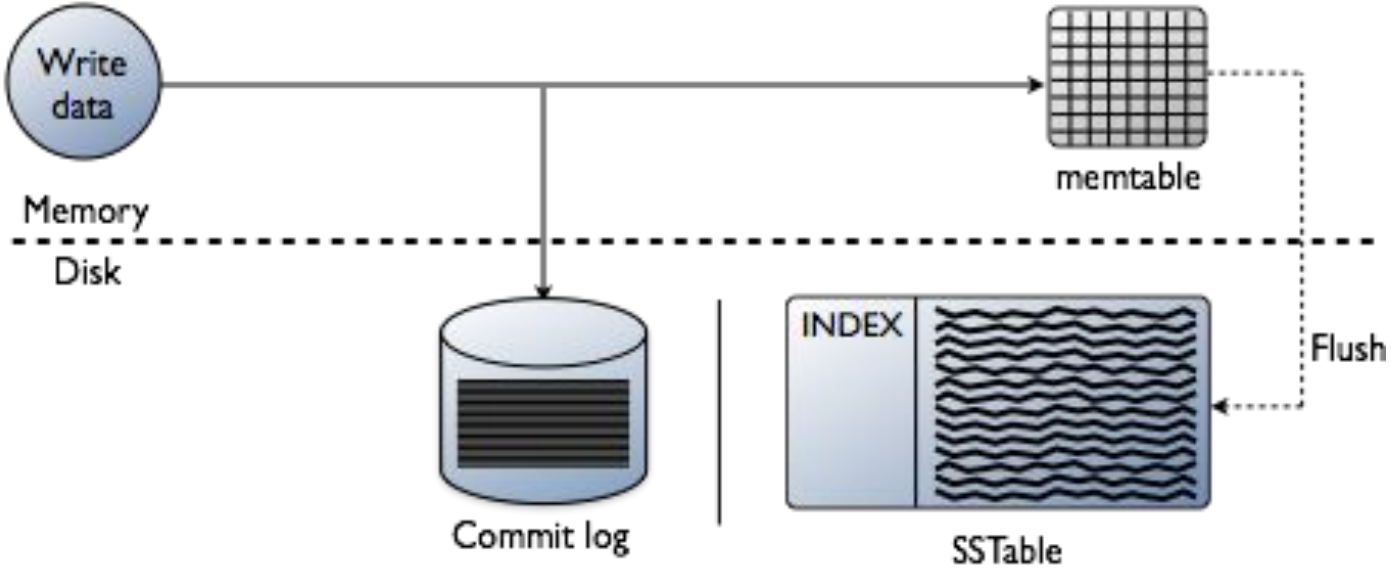
Cassandra Tombstones



Tombstones

- Deleted data is MARKED as Removed == Tombstone
- Data is deleted and removed during compaction
- Compaction can happen in few days depending of the configs.
- Queries on partition with lots of tombstones requires lots of filtering which can slow down the CASS performance.
- Collections operations can lead to tombstones depending on what you do.
- There are Compaction Trade-Offs.

Cassandra: Write Path



Cassandra: Read Path

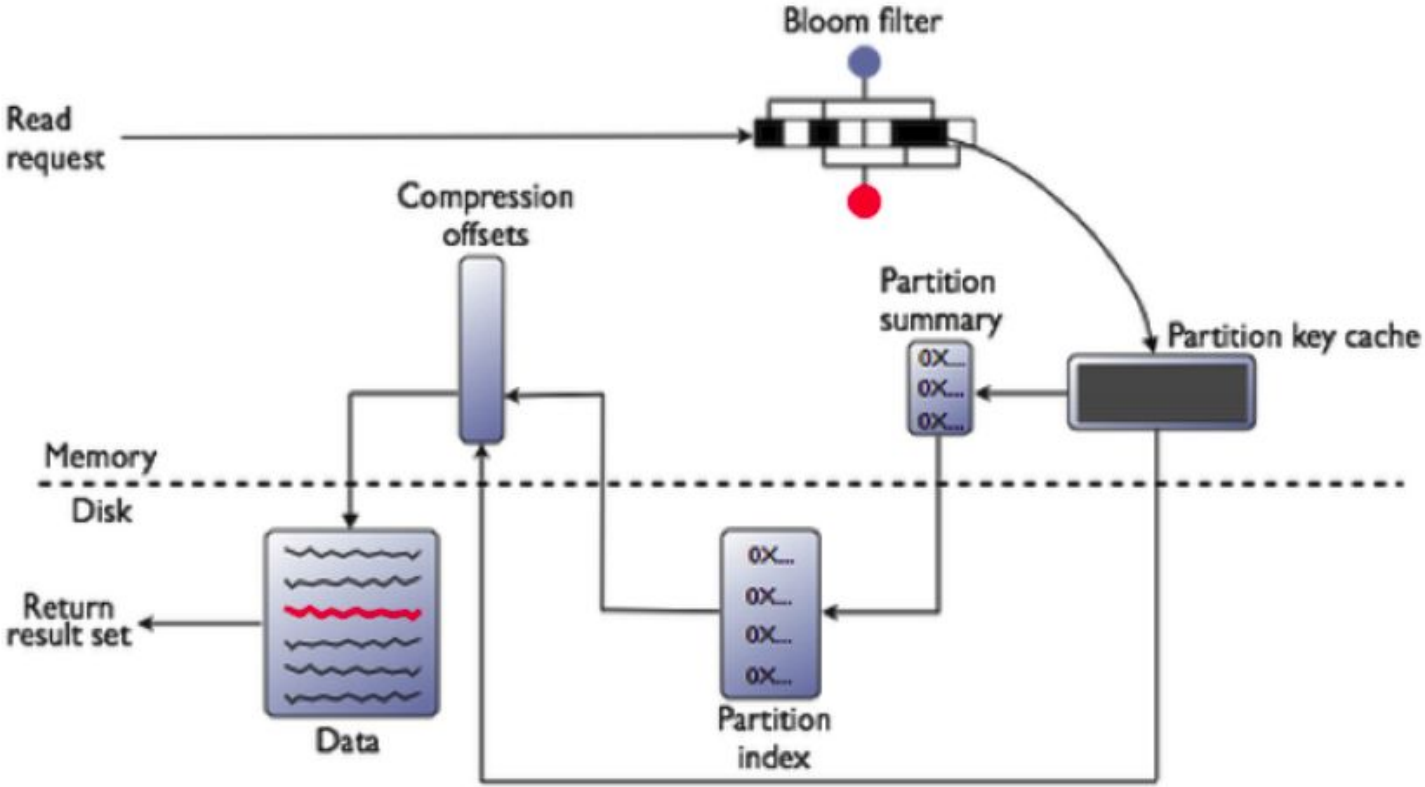


Figure 3. The Cassandra Read Path

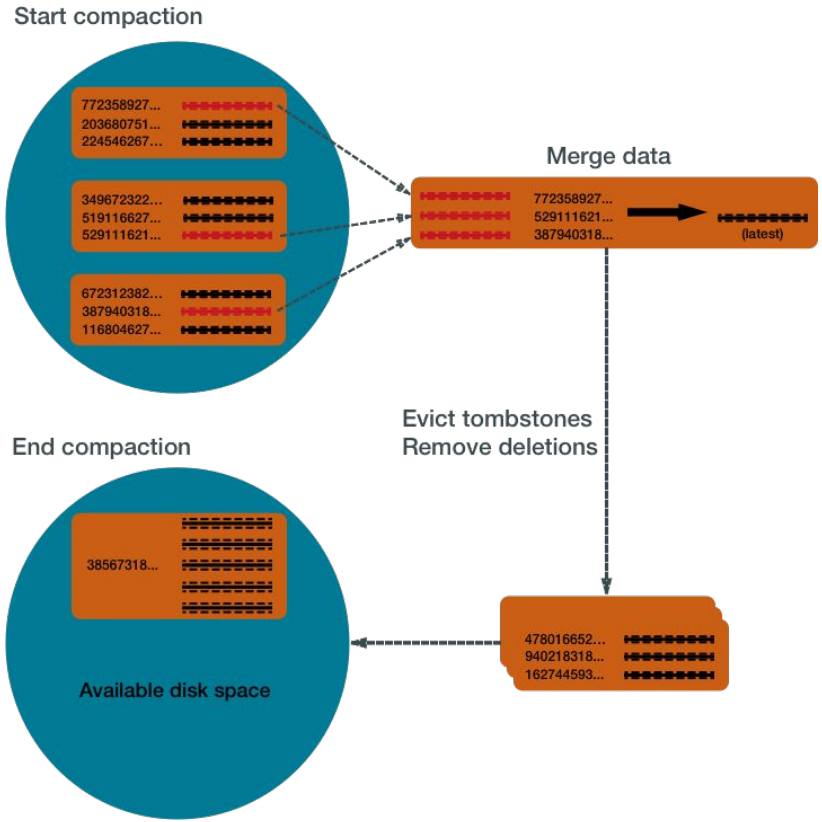
Cassandra: Reads & Index

- Partition Key Cache / Row Cache
  - Off Heap
  - Configurable
- Secondary Index
  - Filters data on table by non-primary key
  - Allow Filtering (it can be problematic)

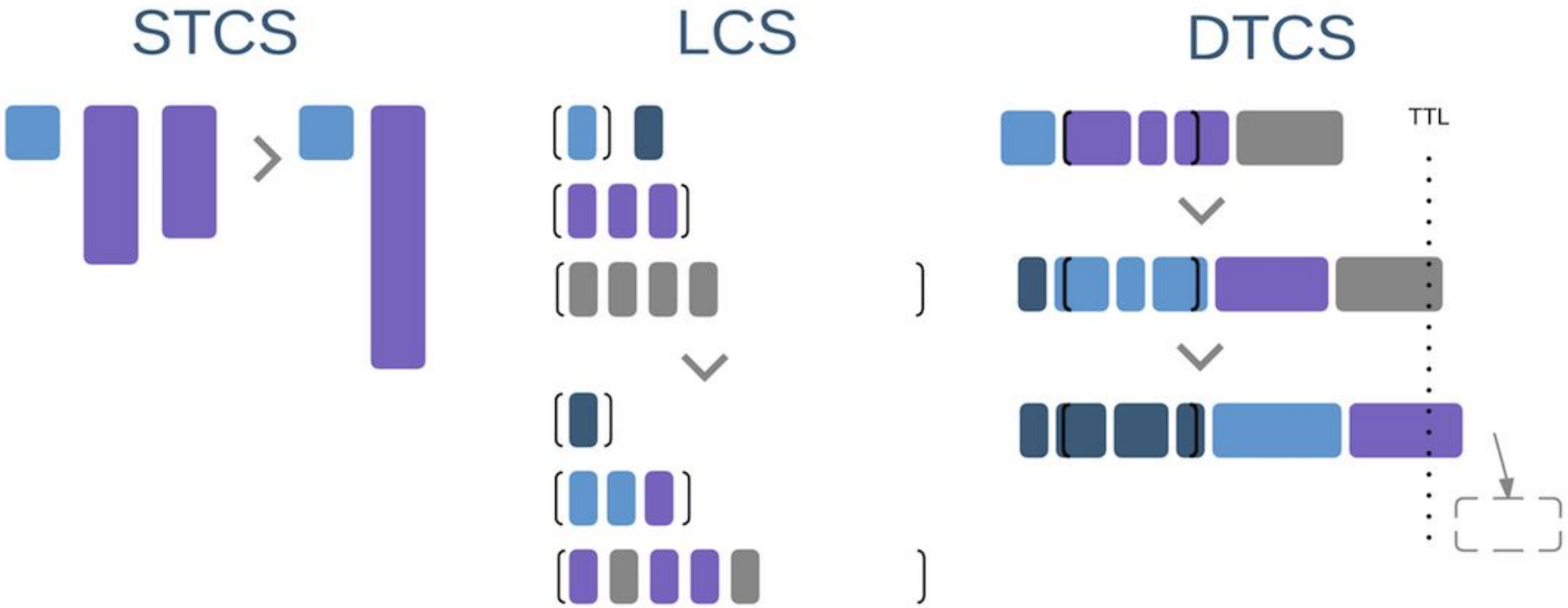
Cassandra: Compaction

- Merge SSTables
- Can be trigger by node_tool
- Impact Cassandra Performance
- Different Strategies for different workloads

Cassandra: Compaction



Cassandra: Compaction Strategies



Cassandra: Repair

- Anti-entropy mechanism (expensive, slow)
- 100% needed in Multi-region clusters
- Other scenarios that you need repairs:
  - Low consistent reads and writes
  - Frequent nodes outage
  - Low reads repair chance
  - Flaky networks

Cassandra: Read Repair

- Easy and cheap (when reading from multiples replicas)
- Hot Data will be repaired a lot

Cassandra: When to not Repair

- Short TTLs
- High Consistent read and writes
- High Read repair chance

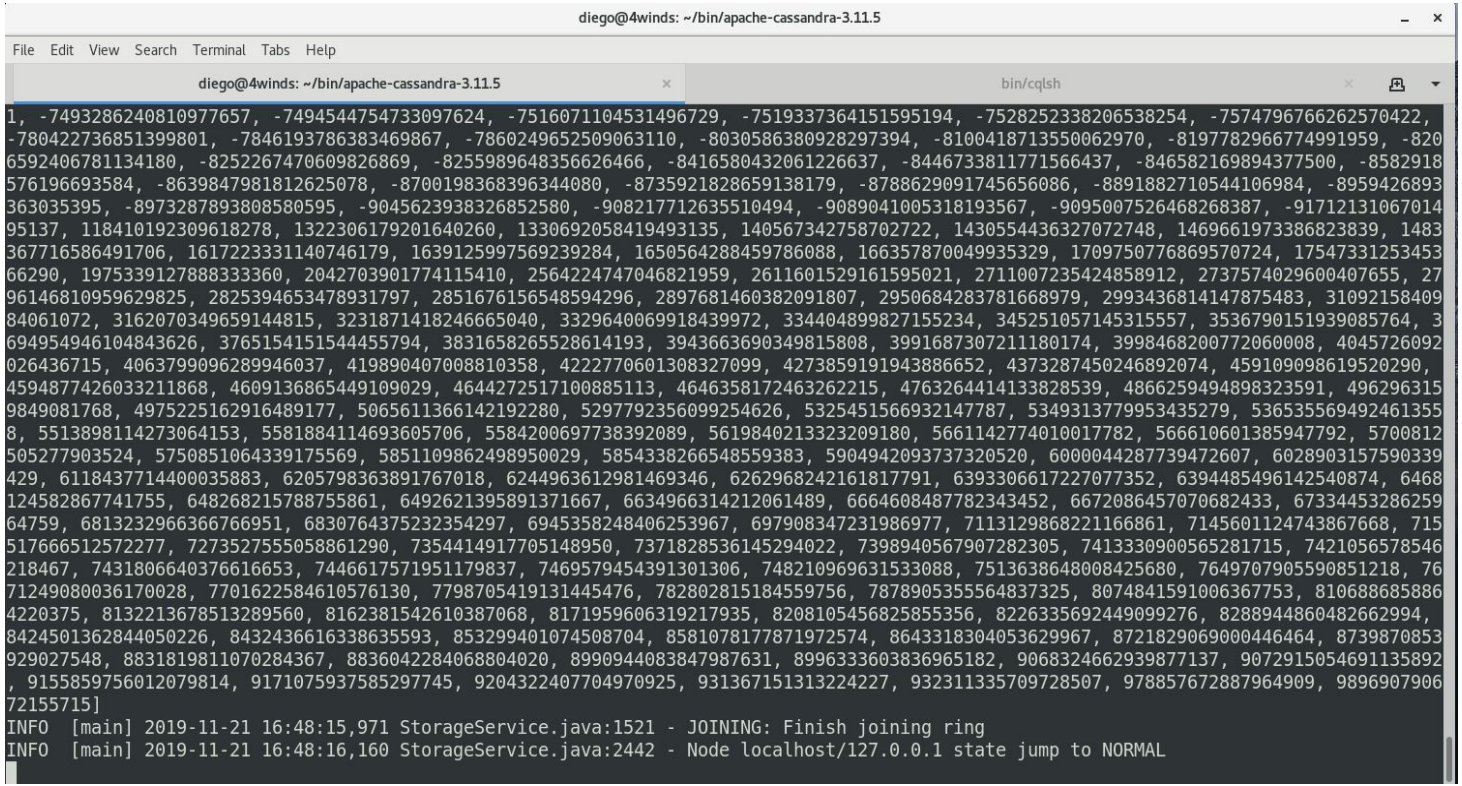
Cassandra: Repair Pain points - most on scale

- ❑ Stream timeouts
- ❑ Tracking / Resume Repairs
- ❑ CPU Usage
- ❑ Disk I/O
- ❑ Latencies / SLA
- ❑ Wide Partitions

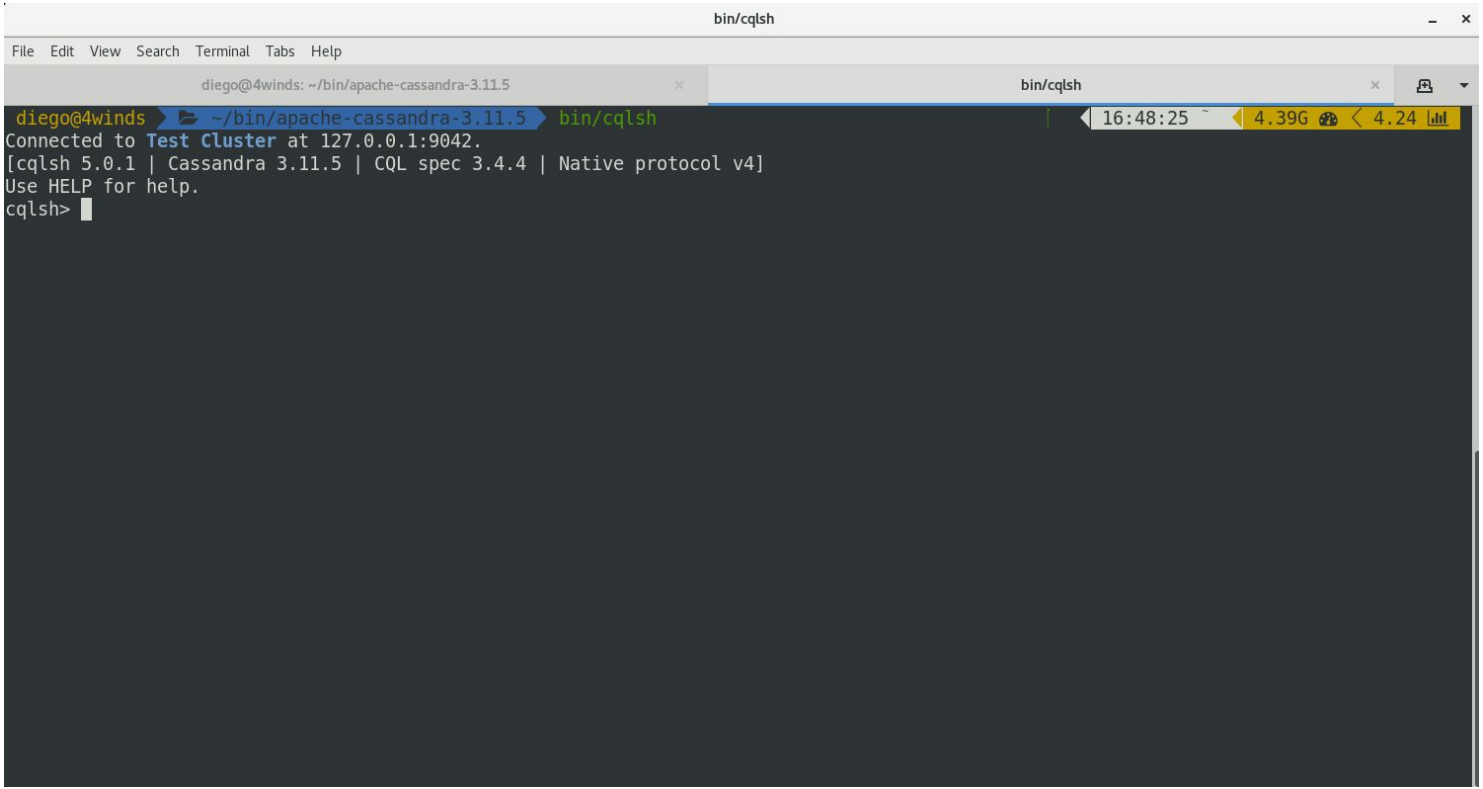
Cassandra: Poor Usage - Anti-Patterns

- Batch usage: Select * from whole table / re-insert periodically
- Lack of denormalized tables to support queries (allow filtering)
- Queue
- Extreme large row sizes
- Abuse of Collections Data types(tombstones)
- Read Before Writing (or write and read immediately.)

Running Cassandra: $ bin/cassandra



Running Cassandra: $ bin/cqlsh



Running Cassandra: Java Code


```java
import com.datastax.driver.core.Cluster;
import com.datastax.driver.core.ResultSet;
import com.datastax.driver.core.Row;
import com.datastax.driver.core.Session;

public class Basic {
    Run | Debug
    public static void main(String[] args) {

        Cluster cluster = null;
        try {
            cluster = Cluster.builder().addContactPoint("127.0.0.1").build();
            Session session = cluster.connect();
            ResultSet rs = session.execute("select release_version from system.local");
            Row row = rs.one();
            System.out.println(row.getString("release_version"));
        } finally {
            if (cluster != null)
                cluster.close();
        }

    }
}
```

Running Cassandra: Java Code


```
nov 21, 2019 5:02:36 PM com.datastax.driver.core.Cluster logDriverVersion
INFO: DataStax Java driver 3.8.0 for Apache Cassandra
nov 21, 2019 5:02:36 PM com.datastax.driver.core.GuavaCompatibility selectImplementation
INFO: Detected Guava >= 19 in the classpath, using modern compatibility layer
nov 21, 2019 5:02:37 PM com.datastax.driver.core.ClockFactory newInstance
INFO: Using native clock to generate timestamps.
nov 21, 2019 5:02:37 PM com.datastax.driver.core.NettyUtil <clinit>
INFO: Did not find Netty's native epoll transport in the classpath, defaulting to NIO.
nov 21, 2019 5:02:37 PM com.datastax.driver.core.policies.DCAwareRoundRobinPolicy init
INFO: Using data-center name 'datacenter1' for DCAwareRoundRobinPolicy (if this is incorrect, please provide the correct
datacenter name with DCAwareRoundRobinPolicy constructor)
nov 21, 2019 5:02:37 PM com.datastax.driver.core.Cluster$Manager init
INFO: New Cassandra host /127.0.0.1:9042 added
3.11.5
```

Running Cassandra: Java Code

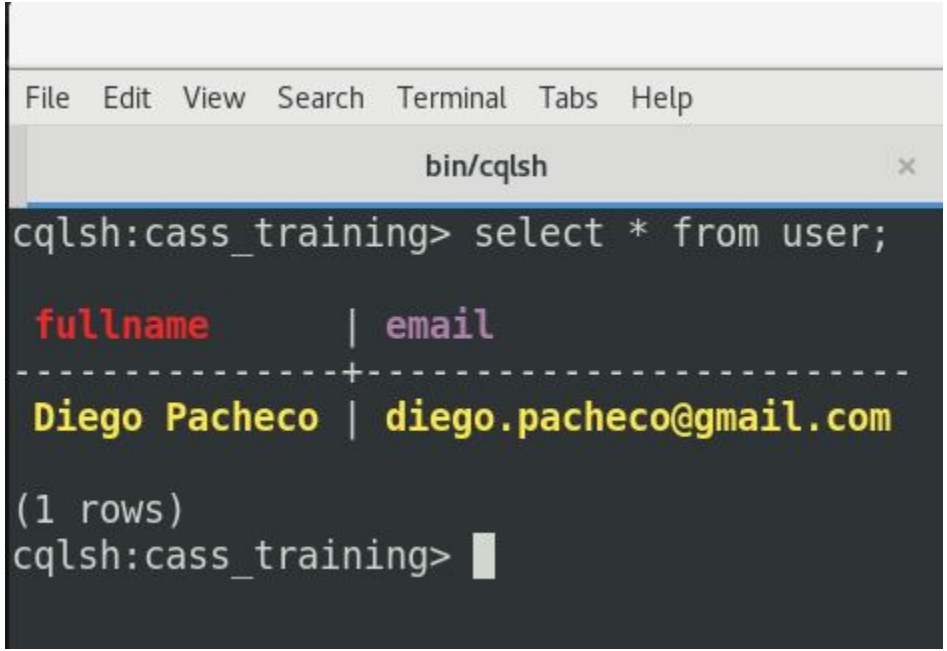
src > main > java > SchemaQuery.java > SchemaQuery > main(String[])

```java
import com.datastax.driver.core.Cluster;
import com.datastax.driver.core.ResultSet;
import com.datastax.driver.core.Row;
import com.datastax.driver.core.Session;

public class SchemaQuery {
    Run | Debug
    public static void main(String[] args) {
        Cluster cluster = Cluster.builder().addContactPoint("127.0.0.1").build();
        Session session = cluster.connect();
        try {
            session.execute("CREATE KEYSPACE cass_training " +
                            "WITH replication = { 'class' : 'SimpleStrategy', 'replication_factor' : 1 };");
            System.out.println("Schema Created");
        }catch(Exception e){
        }

        try {
            session.execute("USE cass_training;");
            session.execute("CREATE TABLE user "
                                +" (fullName text PRIMARY KEY, email text);");
            System.out.println("Table Created");
        }catch(Exception e){
            System.out.println(e);
        }

```

Running Cassandra: Java Code


```
26          try {
27              session.execute("INSERT INTO user " +
28              "(fullName,email) VALUES ('Diego Pacheco','diego.pacheco@gmail.com');");
29              System.out.println("row inserted.");
30          }catch(Exception e){
31              System.out.println(e);
32          }
33
34          try {
35              System.out.println("List data");
36              ResultSet rs = session.execute("SELECT * from user;");
37              for (Row row : rs) {
38                  System.out.println("Name: " + row.getString("fullName"));
39                  System.out.println("Mail: " + row.getString("email"));
40              }
41              System.out.println("row inserted.");
42          }catch(Exception e){
43              System.out.println(e);
44          }
45
46          if (cluster != null)
47              cluster.close();
48      }
49  }
```

Running Cassandra: Java Code



Running Cassandra: Java Code Async


```java
import com.datastax.driver.core.Cluster;
import com.datastax.driver.core.ResultSet;
import com.datastax.driver.core.Session;
import com.google.common.base.Function;
import com.google.common.util.concurrent.AsyncFunction;
import com.google.common.util.concurrent.FutureCallback;
import com.google.common.util.concurrent.Futures;
import com.google.common.util.concurrent.ListenableFuture;

public class Async{
    Run | Debug
    public static void main(String[] args) throws Exception {

        Cluster cluster = Cluster.builder().addContactPoint("127.0.0.1").build();
        ListenableFuture<Session> session = cluster.connectAsync();

        ListenableFuture<ResultSet> resultSet = Futures.transformAsync(session,
        new AsyncFunction<Session, ResultSet>() {
            public ListenableFuture<ResultSet> apply(Session session) throws Exception {
                return session.executeAsync("select release_version from system.local");
            }
        });
```

Running Cassandra: Java Code Async


```
23          ListenableFuture<String> version = Futures.transform(resultSet,
24          new Function<ResultSet, String>() {
25              public String apply(ResultSet rs) {
26                  return rs.one().getString("release_version");
27              }
28          });
29
30          Futures.addCallback(version, new FutureCallback<String>() {
31              public void onSuccess(String version) {
32                  System.out.printf("Cassandra version: %s%n", version);
33              }
34              public void onFailure(Throwable t) {
35                  System.out.printf("Failed to retrieve the version: %s%n",
36                      t.getMessage());
37              }
38          });
39
40          Thread.sleep(3000L);
41          if (cluster != null)
42              cluster.close();
43      }
44  }
```

Running Cassandra: Java Code Async


```
nov 22, 2019 1:34:35 PM com.datastax.driver.core.Cluster logDriverVersion
INFO: DataStax Java driver 3.8.0 for Apache Cassandra
nov 22, 2019 1:34:35 PM com.datastax.driver.core.GuavaCompatibility selectImplementation
INFO: Detected Guava >= 19 in the classpath, using modern compatibility layer
nov 22, 2019 1:34:35 PM com.datastax.driver.core.ClockFactory newInstance
INFO: Using native clock to generate timestamps.
nov 22, 2019 1:34:35 PM com.datastax.driver.core.NettyUtil <clinit>
INFO: Did not find Netty's native epoll transport in the classpath, defaulting to NIO.
nov 22, 2019 1:34:36 PM com.datastax.driver.core.policies.DCAwareRoundRobinPolicy init
INFO: Using data-center name 'datacenter1' for DCAwareRoundRobinPolicy (if this is incorrect, please provide the correct
datacenter name with DCAwareRoundRobinPolicy constructor)
nov 22, 2019 1:34:36 PM com.datastax.driver.core.Cluster$Manager init
INFO: New Cassandra host /127.0.0.1:9042 added
Cassandra version: 3.11.5
```

Running Cassandra: Java Code Mapper


```java
import com.datastax.driver.mapping.annotations.Column;
import com.datastax.driver.mapping.annotations.PartitionKey;
import com.datastax.driver.mapping.annotations.Table;

@Table(keyspace = "cass_training", name = "user",
        readConsistency = "QUORUM",
        writeConsistency = "QUORUM",
        caseSensitiveKeyspace = false,
        caseSensitiveTable = false)
public class User{

    @PartitionKey
    @Column(name = "fullName")
    private String fullName;
    private String email;

    public User(){}

    public User(String name,String email){
        this.fullName = name;
        this.email = email;
    }

    public void setEmail(String email) {
        this.email = email;
    }
    public String getEmail() {
        return email;
    }
    public String getEmail() {
        return email;
    }
    public void setFullName(String fullName) {
        this.fullName = fullName;
    }
    public String getFullName() {
        return fullName;
    }

    @Override
    public String toString() {
        return "Name: " + this.fullName + " Email: " + this.email;
    }
}
```

Running Cassandra: Java Code Mapper

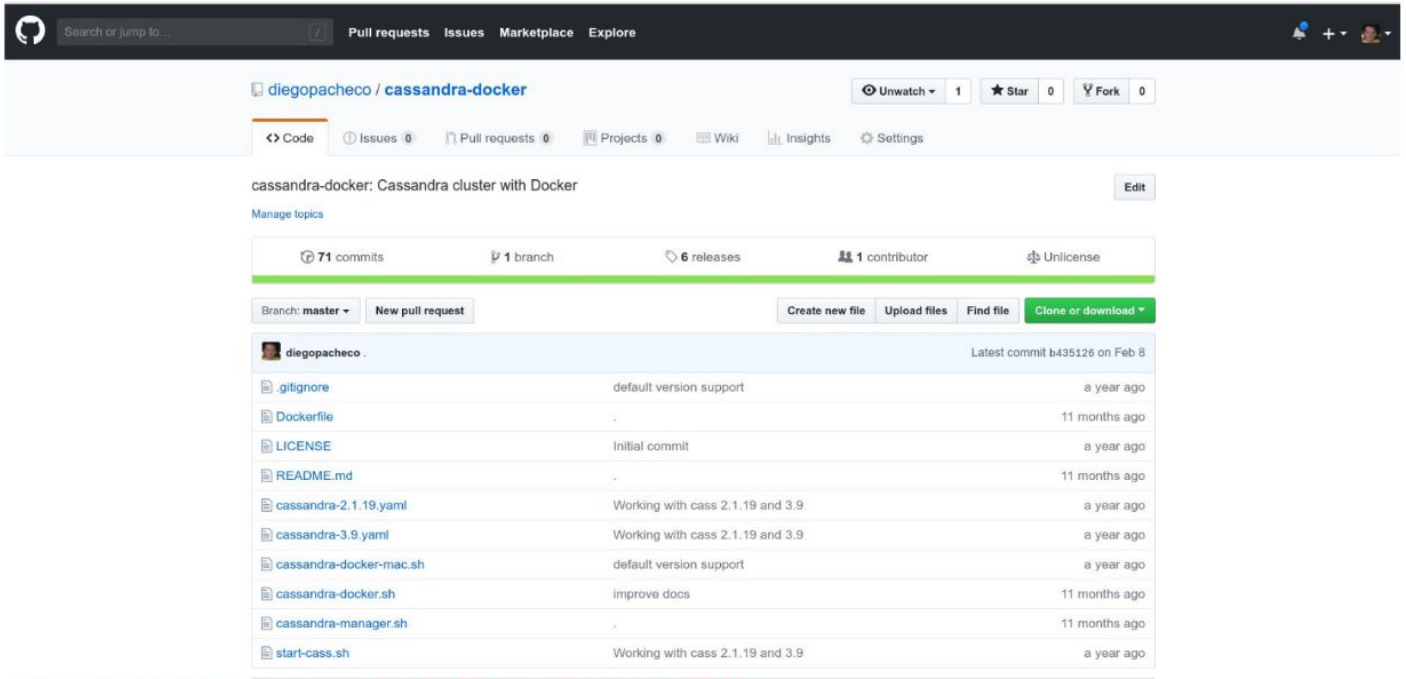
```java
import com.datastax.driver.core.Cluster;
import com.datastax.driver.core.ResultSet;
import com.datastax.driver.core.Session;
import com.datastax.driver.mapping.Mapper;
import com.datastax.driver.mapping.MappingManager;
import com.datastax.driver.mapping.Result;

public class MapperMain {
    Run | Debug
    public static void main(String[] args) {
        Cluster cluster = Cluster.builder().addContactPoint("127.0.0.1").build();
        Session session = cluster.connect();

        MappingManager manager = new MappingManager(session);
        Mapper<User> mapper = manager.mapper(User.class);

        String name = "" + System.currentTimeMillis();
        User u = new User("John Cat N" + name, "cat" + name + "@cats.com");
        mapper.save(u);

        u = mapper.get("Diego Pacheco");
        System.out.println("user: " + u);

        System.out.println("Getting all users... ");
        ResultSet results = session.execute("SELECT * FROM cass_training.user");
        Result<User> users = mapper.map(results);
        for (User user : users) {
            System.out.println("User : " + user);
        }
    }
}
```

Running Cassandra: Java Code Mapper


```
ncoding=UTF-8 -cp /tmp/cp_6s9o6nzqsl4yhtqbb6n7o05fq.jar mapper.MapperMain
nov 22, 2019 1:48:54 PM com.datastax.driver.core.Cluster logDriverVersion
INFO: DataStax Java driver 3.8.0 for Apache Cassandra
nov 22, 2019 1:48:54 PM com.datastax.driver.core.GuavaCompatibility selectImplementation
INFO: Detected Guava >= 19 in the classpath, using modern compatibility layer
nov 22, 2019 1:48:54 PM com.datastax.driver.core.ClockFactory newInstance
INFO: Using native clock to generate timestamps.
nov 22, 2019 1:48:54 PM com.datastax.driver.core.NettyUtil <clinit>
INFO: Did not find Netty's native epoll transport in the classpath, defaulting to NIO.
nov 22, 2019 1:48:55 PM com.datastax.driver.core.policies.DCAwareRoundRobinPolicy init
INFO: Using data-center name 'datacenter1' for DCAwareRoundRobinPolicy (if this is incorrect, please provide the correct datacenter name with DCAware
RoundRobinPolicy constructor)
nov 22, 2019 1:48:55 PM com.datastax.driver.core.Cluster$Manager init
INFO: New Cassandra host /127.0.0.1:9042 added
user: Name: Diego Pacheco Email: diego.pacheco@gmail.com
Getting all users...
User : Name: John Cat N1574459233432 Email: cat1574459233432@cats.com
User : Name: Diego Pacheco Email: diego.pacheco@gmail.com
User : Name: John Cat N1574459335562 Email: cat1574459335562@cats.com
User : Name: John Cat N1574459308465 Email: cat1574459308465@cats.com
```

Want to join the DevOps side of the Force?

Learn & Play Local - diegopacheco/cass-docker



https://github.com/diegopacheco/cassandra-docker

Redis

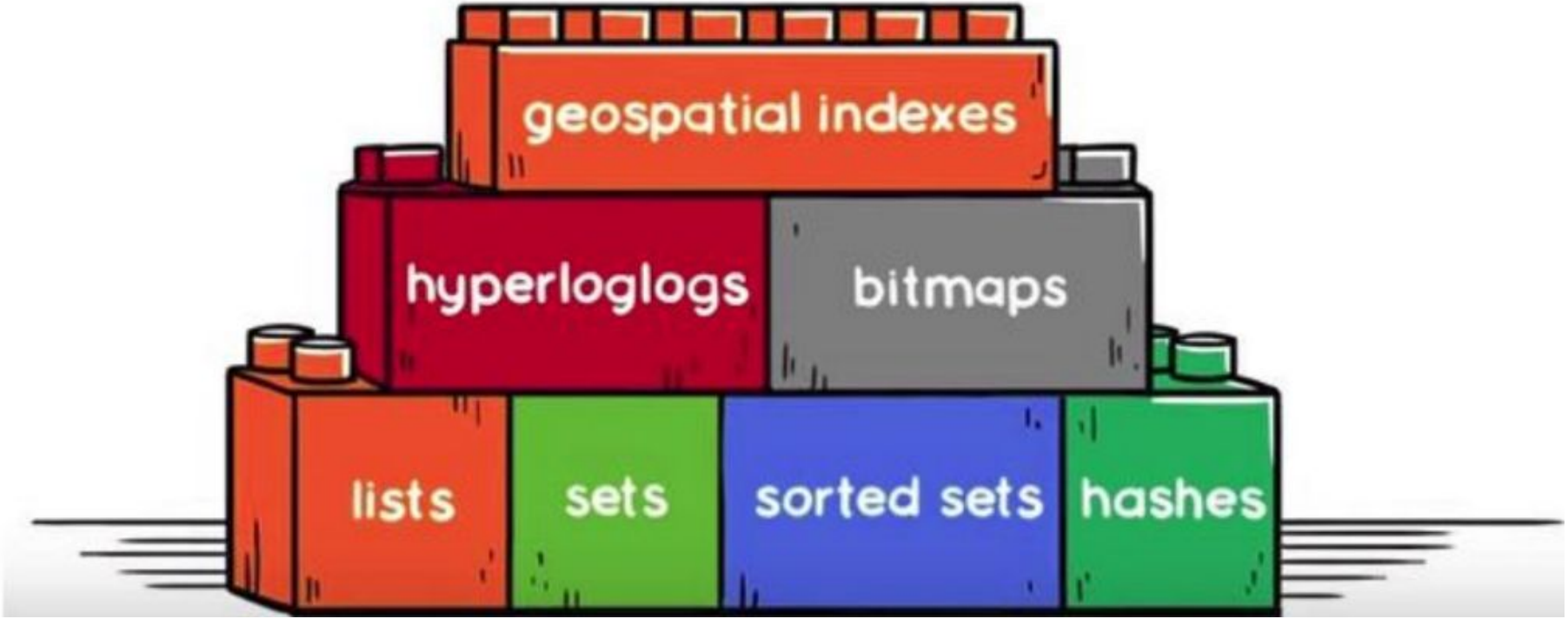
Redis

- FOSS
- In Memory K/V Store written in C
- Create for: Caching, Session Store, Queue, Analytics
- Specific Commands per Data Structures: Strings, Hash, sets
- Keys TTL
- Bring your own Data Structure
- Fast, Low Latency, Battle tested everybody. :D
- Single Thread :(

Redis data strata

v1.0	Strings	v2.2	Bit arrays
	Lists	v2.8.9	HyperLogLog
	Sets	v3.2	Geo Sets
v1.2	Sorted Sets		Bit fields
v2.0	Hashes	v4	Streams (?)
		V5	**MODULES!**

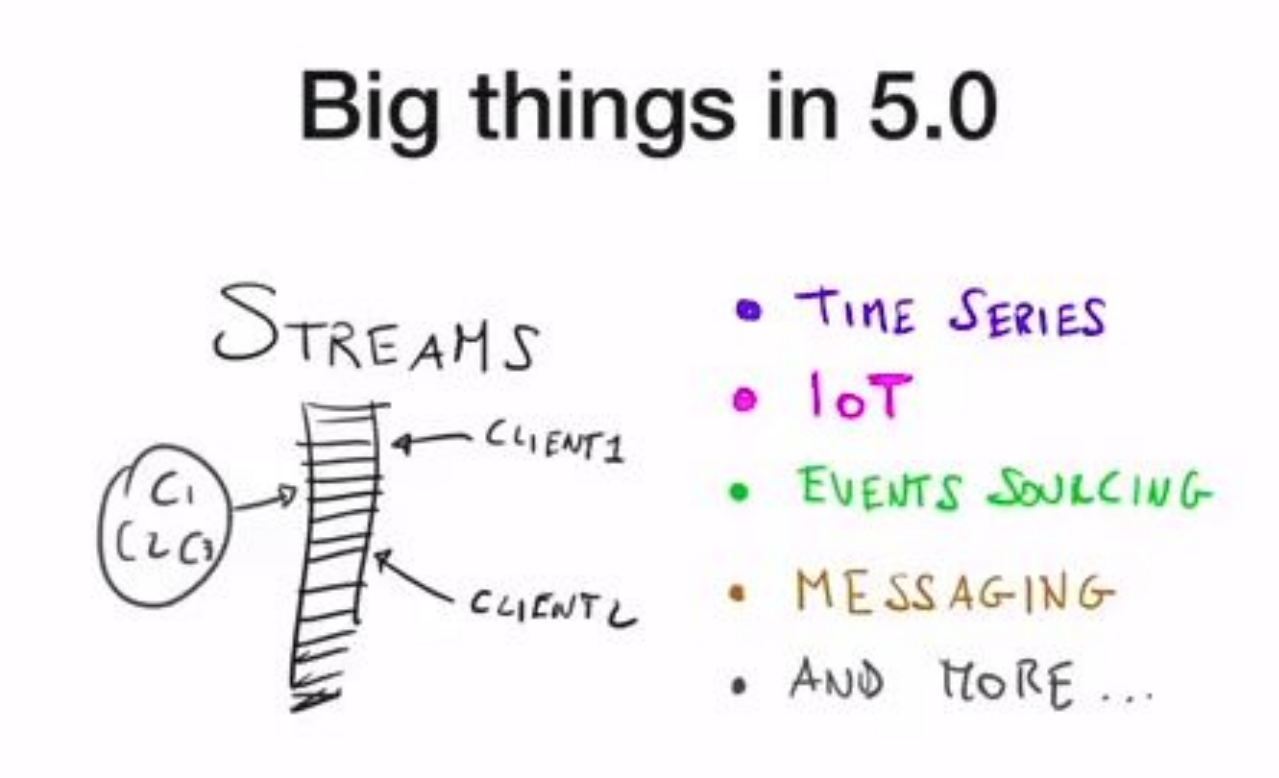
Redis - Data Structures



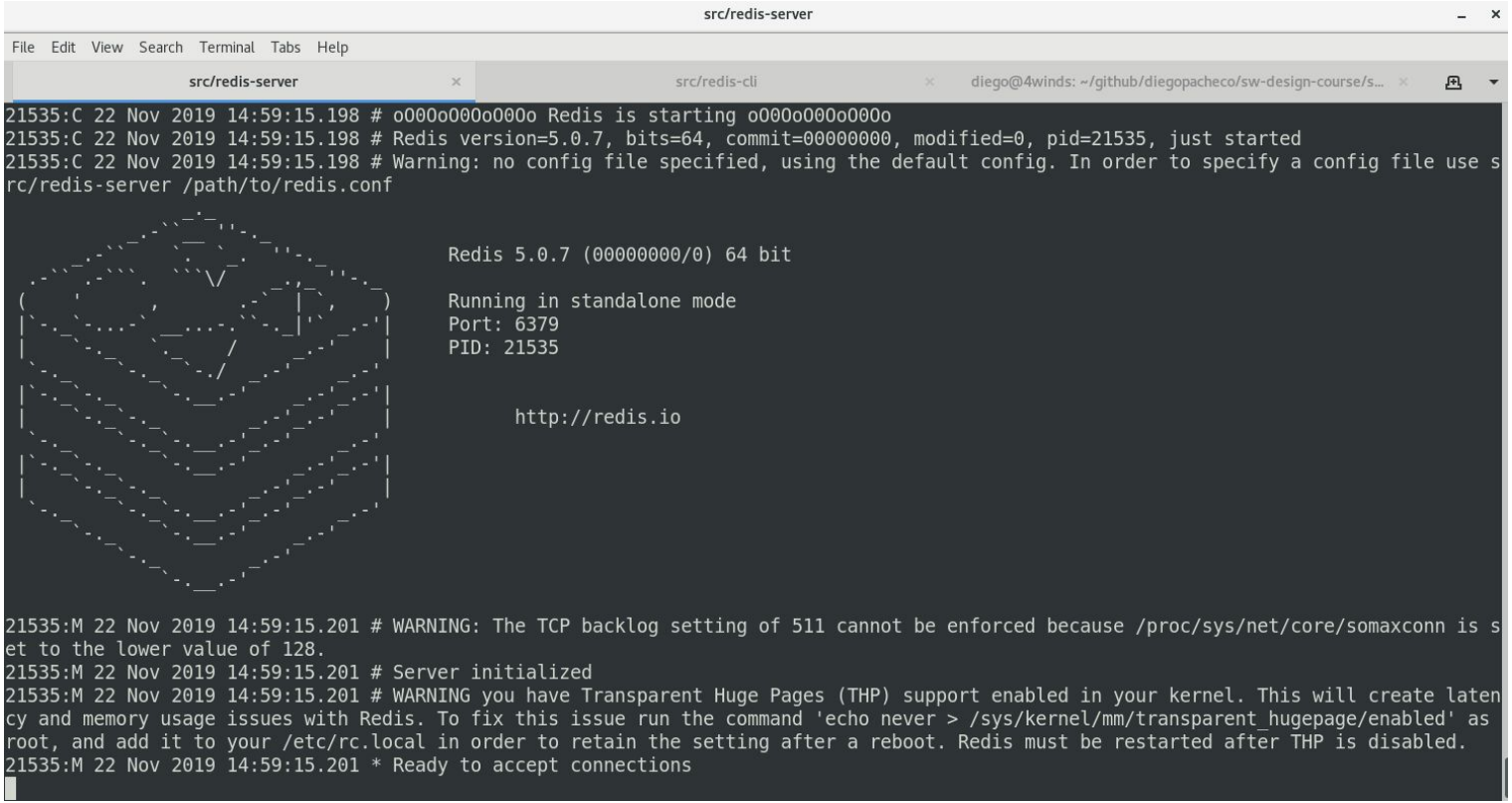
Redis >= 4.x

- Redis-Modules
  - It's not Lua Scripting
  - Built in C
  - Low Latency / Embedded in Redis
  - Use cases: Extends Redis / New Capabilities DSs

Redis >= 5.x

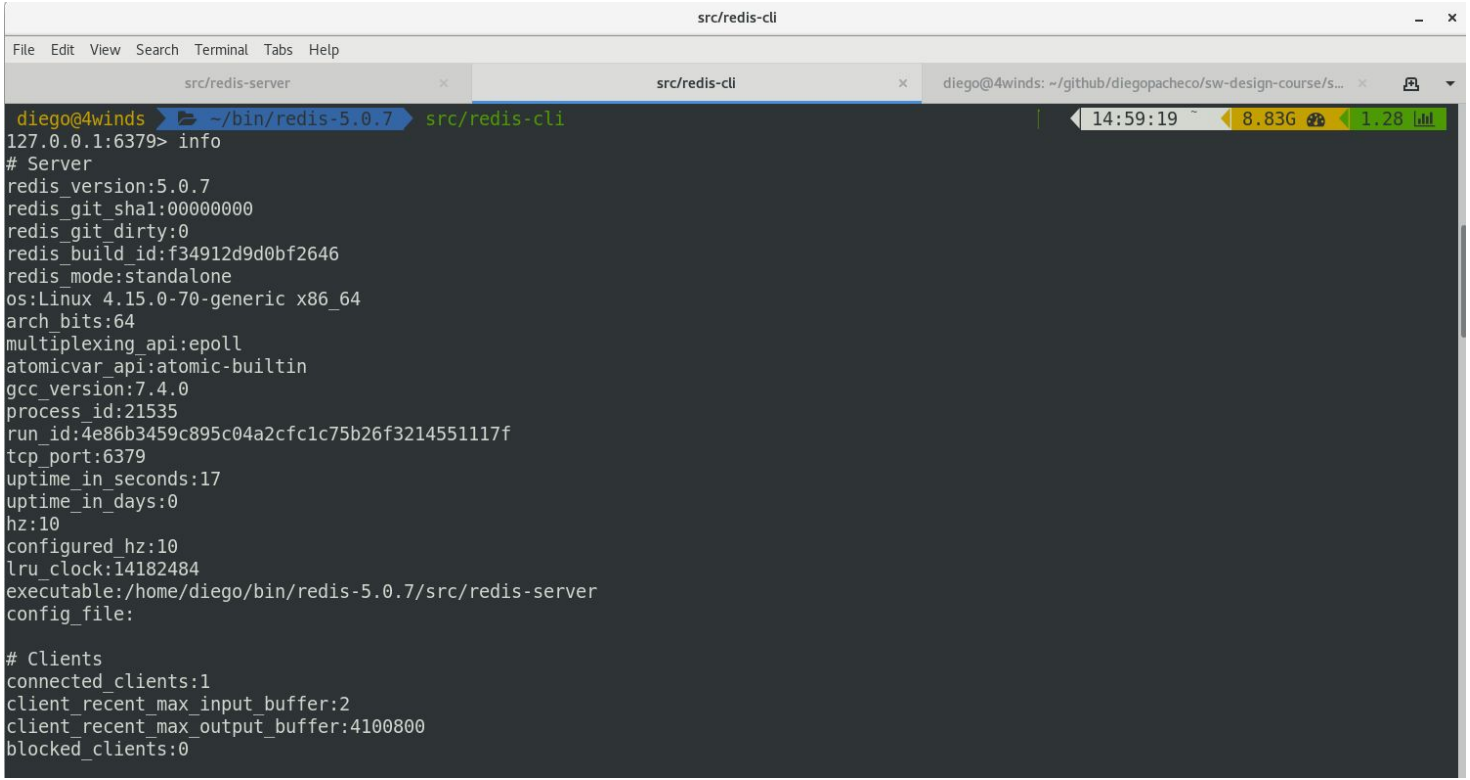

Big things in 5.0
STREAMS
C1
C2
C3
CLIENT1
CLIENTL
• TIME SERIES
• IoT
• EVENTS SOURCING
• MESSAGING
• AND MORE ...

Redis: make ; src/redis-server



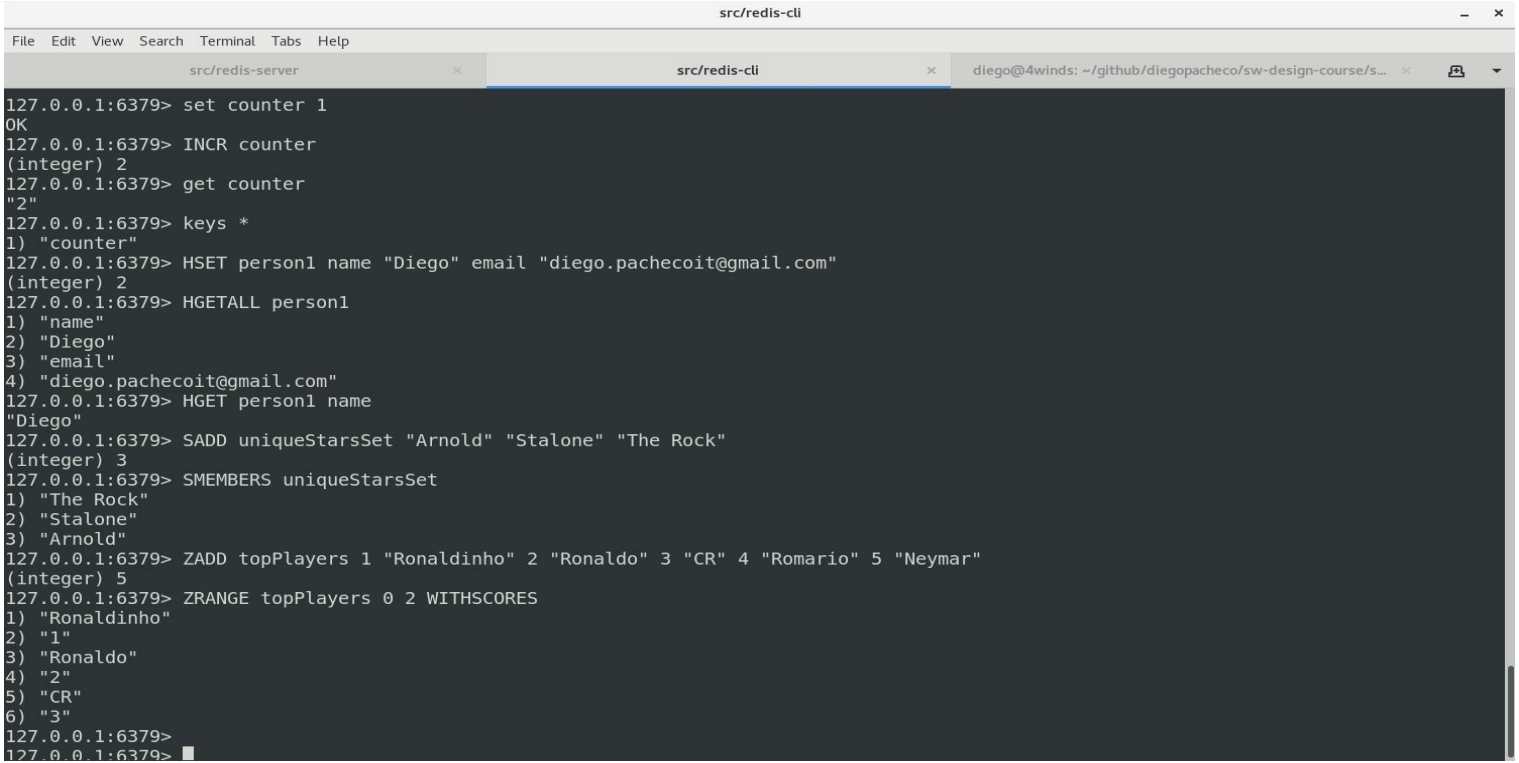
https://redis.io/download

Redis: src/redis-cli



https://redis.io/download

Redis: src/redis-cli



https://redis.io/commands/

Redis & Java: Lettuce


```java
import io.lettuce.core.RedisClient;
import io.lettuce.core.api.StatefulRedisConnection;

public class SimpleLettuceMain {
    Run | Debug
    public static void main(String[] args) {
        RedisClient redisClient = RedisClient.create("redis://localhost/0");
        StatefulRedisConnection<String, String> connection = redisClient.connect();

        System.out.println("Connected to Redis");
        connection.sync().set("k1", "Hello World");
        System.out.println("Key from redis k1: " + connection.sync().get("k1"));

        connection.close();
        redisClient.shutdown();
    }
}
```

```
nov 22, 2019 3:24:23 PM io.lettuce.core.EpollProvider <clinit>
INFO: Starting without optional epoll library
nov 22, 2019 3:24:23 PM io.lettuce.core.KqueueProvider <clinit>
INFO: Starting without optional kqueue library
Connected to Redis
Key from redis k1: Hello World
```

Redis & Java: Spring Data + Lettuce

src/redis-cli

File Edit View Search Terminal Tabs Help

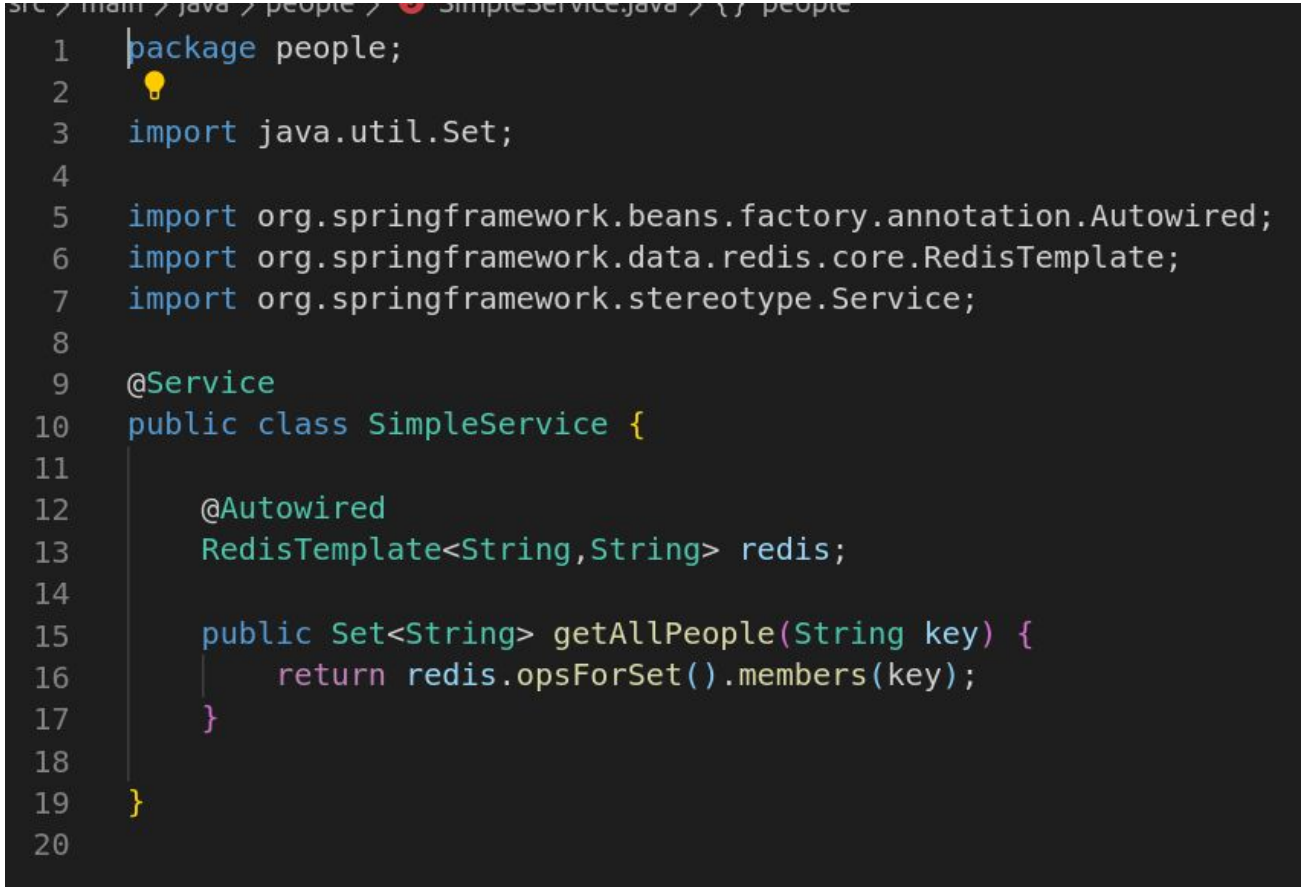
src/redis-cli × src/redis-server ×

```
127.0.0.1:6379> sadd people "diego" "melina" "gandalfy"
(integer) 0
127.0.0.1:6379> SMEMBERS people
1) "melina"
2) "gandalfy"
3) "diego"
127.0.0.1:6379>
```

Redis & Java: Spring Data + Lettuce


```java
package people;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.data.redis.connection.RedisStandaloneConfiguration;
import org.springframework.data.redis.connection.lettuce.LettuceConnectionFactory;
import org.springframework.data.redis.core.RedisTemplate;

@Configuration
class AppConfig {

  @Bean
  public LettuceConnectionFactory redisConnectionFactory() {
    return new LettuceConnectionFactory(new RedisStandaloneConfiguration("127.0.0.1", 6379));
  }

  @Bean
  public RedisTemplate<?, ?> redisTemplate() {
      RedisTemplate<byte[], byte[]> template = new RedisTemplate<>();
      template.setConnectionFactory(redisConnectionFactory());
      return template;
  }

}

```

Redis & Java: Spring Data + Lettuce



```java
package people;

import java.util.Set;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.redis.core.RedisTemplate;
import org.springframework.stereotype.Service;

@Service
public class SimpleService {

    @Autowired
    RedisTemplate<String,String> redis;

    public Set<String> getAllPeople(String key) {
        return redis.opsForSet().members(key);
    }

}
```

Redis & Java: Spring Data + Lettuce


```java
package people;

import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.Bean;

@SpringBootApplication
public class SpringDataRedisMain {

    Run | Debug
    public static void main(String[] args) {
        SpringApplication.run(SpringDataRedisMain.class, args);
    }

    @Bean
    public CommandLineRunner commandLineRunner(ApplicationContext ctx) {
        return args -> {

            System.out.println("Running Spring Boot application in Console... ");

            SimpleService service = ctx.getBean(SimpleService.class);
            System.out.println("All People :" + service.getAllPeople("people"));

        };
    }
}

```

Redis & Java: Spring Data + Lettuce

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: Java Process Console

```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v2.2.1.RELEASE)

2019-11-22 16:10:14.823  INFO 18965 --- [           main] people.SpringDataRedisMain               : Starting Spring
DataRedisMain on 4winds with PID 18965 (/home/diego/github/diegopacheco/sw-design-course/src/java/java-spring-data-r
edis-fun/bin/main started by diego in /home/diego/github/diegopacheco/sw-design-course/src/java/java-spring-data-red
is-fun)
2019-11-22 16:10:14.827  INFO 18965 --- [           main] people.SpringDataRedisMain               : No active profi
le set, falling back to default profiles: default
2019-11-22 16:10:15.469  INFO 18965 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Multiple Spring
 Data modules found, entering strict repository configuration mode!
2019-11-22 16:10:15.473  INFO 18965 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping S
pring Data repositories in DEFAULT mode.
2019-11-22 16:10:15.520  INFO 18965 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring
 Data repository scanning in 24ms. Found 0 repository interfaces.
2019-11-22 16:10:16.573  INFO 18965 --- [           main] people.SpringDataRedisMain               : Started SpringD
ataRedisMain in 2.161 seconds (JVM running for 2.568)
Running Spring Boot application in Console...
2019-11-22 16:10:16.791  INFO 18965 --- [           main] io.lettuce.core.EpollProvider            : Starting withou
t optional epoll library
2019-11-22 16:10:16.792  INFO 18965 --- [           main] io.lettuce.core.KqueueProvider           : Starting withou
t optional kqueue library
All People :[melina, gandalfy, diego]
```

Redis & Java: Spring Data + Lettuce : Hash Mapping


```java
package people;

import org.springframework.data.annotation.Id;
import org.springframework.data.redis.core.RedisHash;

@RedisHash("players")
public class Person{
    @Id String id;
    String name;

    public Person(){}

    public Person(String id,String name){
        this.id=id;
        this.name=name;
    }

    public String getId() {
        return id;
    }
    public void setId(String id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    @Override
    public String toString() {
        return "id: " + id + " name: " + name;
```

Redis & Java: Spring Data + Lettuce : Hash Mapping


```
1   package people;
2
3   import org.springframework.data.repository.CrudRepository;
4
5   public interface PersonRepository extends CrudRepository<Person, String> {}
6
```

Redis & Java: Spring Data + Lettuce : Hash Mapping


```java
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.Bean;

@SpringBootApplication
public class SpringDataRedisRepositoryMain{

    Run | Debug
    public static void main(String[] args) {
        SpringApplication.run(SpringDataRedisRepositoryMain.class, args);
    }

    @Bean
    public CommandLineRunner commandLineRunnerMapper(ApplicationContext ctx) {
        return args -> {

            System.out.println("Running Spring Boot application in Console... ");

            PersonRepository repo = ctx.getBean(PersonRepository.class);
            Person p = new Person("diegopacheco", "diego.pacheco.it@gmail.com");
            repo.save(p);
            System.out.println("Persons: " + repo.findAll());
            System.out.println("Count: " + repo.count());
            repo.delete(p);
        };
    }

}
```

Redis & Java: Spring Data + Lettuce : Hash Mapping

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: Java Process Console

```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v2.2.1.RELEASE)

2019-11-22 16:45:46.449  INFO 16552 --- [           main] people.SpringDataRedisRepositoryMain     : Starting Spring
DataRedisRepositoryMain on 4winds with PID 16552 (/home/diego/github/diegopacheco/sw-design-course/src/java/java-spr
ing-data-redis-fun/bin/main started by diego in /home/diego/github/diegopacheco/sw-design-course/src/java/java-sprin
g-data-redis-fun)
2019-11-22 16:45:46.453  INFO 16552 --- [           main] people.SpringDataRedisRepositoryMain     : No active profi
le set, falling back to default profiles: default
2019-11-22 16:45:46.962  INFO 16552 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Multiple Spring
 Data modules found, entering strict repository configuration mode!
2019-11-22 16:45:46.965  INFO 16552 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping S
pring Data repositories in DEFAULT mode.
2019-11-22 16:45:47.185  INFO 16552 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring
 Data repository scanning in 199ms. Found 1 repository interfaces.
2019-11-22 16:45:48.274  INFO 16552 --- [           main] people.SpringDataRedisRepositoryMain     : Started SpringD
ataRedisRepositoryMain in 2.226 seconds (JVM running for 3.039)
Running Spring Boot application in Console...
2019-11-22 16:45:48.491  INFO 16552 --- [           main] io.lettuce.core.EpollProvider            : Starting withou
t optional epoll library
2019-11-22 16:45:48.493  INFO 16552 --- [           main] io.lettuce.core.KqueueProvider           : Starting withou
t optional kqueue library
All People :[melina, gandalfy, diego]
Running Spring Boot application in Console...
Persons: [id: diegopacheco name: diego.pacheco.it@gmail.com]
Count: 1
```

Exercises

Constraints

You can code this exercises with Java, Scala or Clojure.
You can use any framework you like. UI is not required.
You need to run Cassandra 3.x and Redis 5.x (nativily, docker, does not matter).

1. **Use cassandra and build Movie Catalog Service** Where movies should have ID(UUID), name, category, length, linkToYouTubeTrailer. You should be able to query by category, name and ID.
2. **Use Redis and Build a Twitter application** where you should have tweet(message) and you should be able to follow people and people could follow you.
3. **Build a profile Service** using Cassandra, you should have a REST interface with the following data: ID(UUID), name, email, dateOfBirth, gender, twitterURL, faceURL. You need store the data in Cassandra and use Redis as a Cache. You need support the following operations:
   a. List all users by Id
   b. List all users by Email
   c. Add / Remove / Update users

Cassandra & Redis

Diego Pacheco