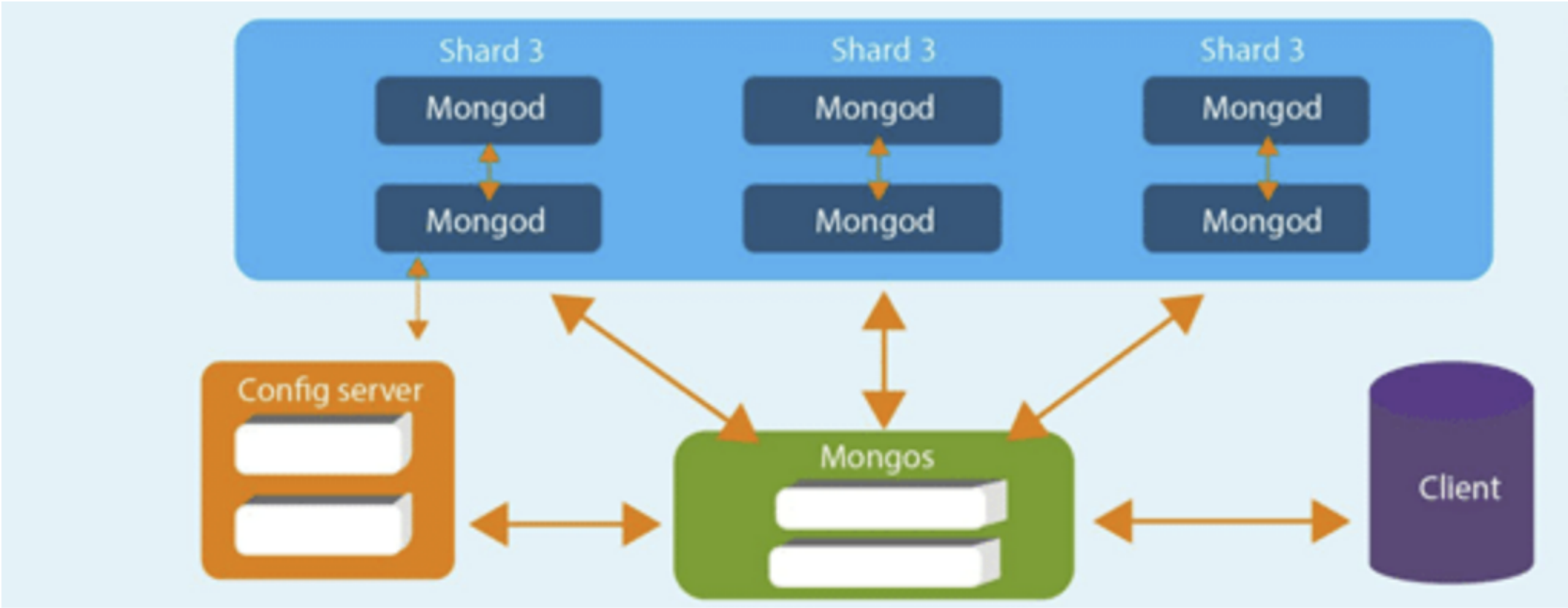
Springboot With Database

Classified as a NoSQL database program, MongoDB uses JSON-like documents with optional schemas. MongoDB is developed by MongoDB Inc.

Architectural style



Code Deep Dive


C
M

```
</dependency>
<!-- mongodb -->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-mongodb</artifactId>
</dependency>
<!-- mongodb -->
```

```
import org.springframework.data.mongodb.repository.MongoRepository;
import org.springframework.stereotype.Repository;
@Repository
public interface CustomerRepository extends MongoRepository<Customer, String>{

  public Customer findByName(String name);
  public Customer findBycustId(String custId);
  public void deleteByname(String name);
  public Customer save(Customer customer);

}
```

```yaml
version: '3'
services:
  mongo:
    image: mongo:3.4.7
    ports:
      - "27017:27017"
  app:
    image: falcon007/mongo-spring-data:latest
    links:
      - mongo
    environment:
      spring.data.mongodb.uri: mongodb://mongo:27017/data
    ports:
      - "8083:8083"
volumes:
  installation:
    external: false
```

```
#Build

mvn clean install

# Run
docker-compose -f docker-compose-mongo.yml up -d
mvn spring-boot:run

# training
Post:http://localhost:8080/customer/create?name=shanker&custId=12&address=bangalore
Get Read all:http://localhost:8080/customer/read
Delete by name:http://localhost:8080/customer/delete?name=shanker

{
  "name": "Rama",
  "custId": "1re206",
  "address": "Bangalore"
}

#Show data::

docker exec -it day3_mongo_1 bash
mongo
show dbs
show tables
db.customer.find()
```

Lab4:Create Your first mongo project

Create a springboot project which store the message data in to mongo db

Input:
Auto message communication :

Message: to,from,content,corelationid(uuid)

Operation:
Get
Post
Put
Delete

Run and test through curl and postman

Spring boot with mysql

Code Deep Dive


Web
SQL

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
    </dependency>
```

Tag name: **artifactId**

```java
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;

@Entity
public class Users {

    @Id
    @GeneratedValue
    @Column(name = "id")
    private Integer id;
    @Column(name = "name")
    private String name;
    @Column(name = "team_name")
    private String teamName;
    @Column(name = "salary")
    private Integer salary;

    public Users() {
    }
```

```java
import com.rama.db.model.Users;
import org.springframework.data.repository.CrudRepository;

public interface UsersRepository extends CrudRepository<Users, Integer> {
}
```

```yaml
spring:
  datasource:
    url: jdbc:mysql://${MYSQL_HOST:localhost}:3306/db
    username: user
    password: password
  jpa:
    hibernate.ddl-auto: update
```

```yaml
version: '3.3'
services:
  mysql-db:
    image: mysql:5.7
    restart: always
    environment:
      MYSQL_DATABASE: 'db'
      # So you don't have to use root, but you can if you like
      MYSQL_USER: 'user'
      # You can use whatever password you like
      MYSQL_PASSWORD: 'password'
      # Password for root access
      MYSQL_ROOT_PASSWORD: 'password'
    ports:
      # <Port exposed> : < MySQL Port running inside container>
      - '3306:3306'
    expose:
      # Opens port 3306 on the container
      - '3306'
      # Where our data will be persisted
    volumes:
      - my-db:/var/lib/mysql
# Names our volume
volumes:
  my-db:
```

POST http://localhost:8080/user/create Params Send

Authorization Headers (1) Body Pre-request Script Tests

form-data x-www-form-urlencoded raw binary JSON (application/json)

```
{
  "name": "Rama",
  "team_name": "alpha",
  "salary": 23000
}
```

Body Cookies Headers (5) Test Results Status: 200 OK

Pretty Raw Preview Text

```
saved
```

GET http://localhost:8080/user/read Params Send

Authorization Headers Body Pre-request Script Tests

Type No Auth

Body Cookies Headers (5) Test Results Status: 200 OK

Pretty Raw Preview JSON

```
[
    {
        "id": 1,
        "name": "Rama",
        "teamName": null,
        "salary": 23000
    }
]
```

Create Springboot microservice with mysql

Create a springboot project which store the message data in to mysql db

Input:
Auto message communication :

Message: to,from,content,corelationid(uuid)

Operation:
Get
Post
Put
Delete

Run and test through curl and postman

Create the docker image for bothe application and test with database.

THANKS !