Spring Cloud

Hexavarsity

Session Plan

Spring Cloud

Spring Cloud Features

Distributed Configuration

Distributed Messaging

Circuit Breakers

Spring Data

Spring Cloud

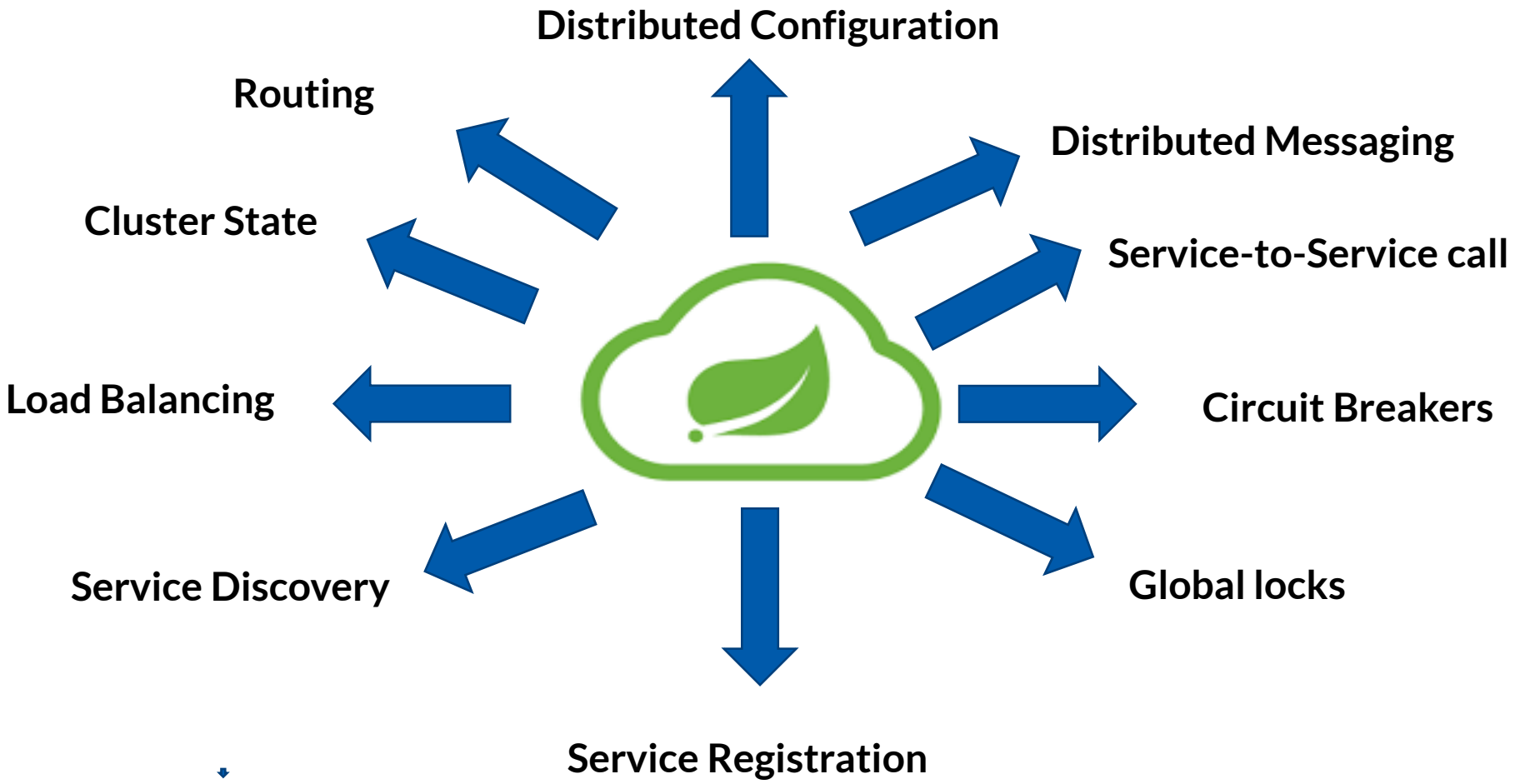
- Spring boot is a java framework to work on auto-configuration in Web Application. Spring cloud is part of Spring boot, where Spring boot is Stand Alone, App – Centric Application framework.

- Spring Cloud is a Spring module that provides the RAD (Rapid Application Development) feature to the Spring framework

- Spring Cloud is Configuration server technology and communicate with many services and collect in one Application.

- Spring Cloud provides tools for developers to build the common patterns in distributed systems quickly.

Features of Spring Cloud



Distributed Configuration

- Spring Data Commons is part of the umbrella Spring Data project it provides shared infrastructure across the Spring Data projects.

- It contains technology neutral repository interfaces as well as a metadata model for persisting Java classes, Powerful Repository and custom object-mapping abstractions.

Distributed Messaging

- In a microservice environment or any other distributed system you may come upon the requirement to exchange events between services.

- Spring provides a JMS integration framework and the Advanced Message Queuing Protocol (AMQP).
- Spring also support Message driven pojo's of EJB

Circuit Breakers

- In the microservices world, to fulfill a client request, one microservice may need to talk to other microservices.
- We should minimize this kind of direct dependencies on other microservices, but in some cases, it is unavoidable.
- If a microservice is down or not functioning properly then the issue may cascade up to the upstream services.
- Netflix created Hystrix library implementing the Circuit Breaker pattern to address these kinds of issues.
- We can use Spring Cloud Netflix Hystrix Circuit Breaker to protect microservices from cascading failures.

Global Clocks

- The current date and time depend on the time zone and, for globalized applications, a time provider is necessary to ensure that the date and time are created with the correct time zone.

- Spring Global clock feature help to test that our code changes work with different time zones or – when using a fixed clock – that time doesn't affect our code.

Service Registration

- Client-side service discovery allows services to find and communicate with each other without hard-coding hostname and port. The only ‘fixed point' in such an architecture consists of a service registry with which each service has to register.

- Netflix Eureka each client can simultaneously act as a server, to replicate its status to a connected peer. In other words, a client retrieves a list of all connected peers of a service registry and makes all further requests to any other services through a load-balancing algorithm.

Service Discovery

- Service Discovery is one of the key tenets of a microservice-based architecture.
- Trying to hand-configure each client or some form of convention can be difficult to do and can be brittle. Eureka is the Netflix Service Discovery Server and Client.
- The server can be configured and deployed to be highly available, with each server replicating state about the registered services to the others.

Load Balancing

- Microservice application use the Spring Cloud LoadBalancer to provide client-side load-balancing in calls to another microservice.

Spring Data - JDBC

Spring Data JDBC is based on the following design decisions:

- No lazy loading or caching is done.
- If you save an entity, it gets saved permanently. There is no dirty tracking.
- Simple model to map entities to tables.

Spring Data JPA

- Sophisticated support to build repositories based on Spring and JPA
- Transparent auditing of domain class
- Pagination support, dynamic query execution, ability to integrate custom data access code
- Validation of @Query annotated queries at bootstrap time
- Support for XML based entity mapping

Spring Data JPA

```
<dependencies>
<dependency>
<groupId>org.springframework.data</groupId>
<artifactId>spring-data-jpa</artifactId>
<version>version-number</version>
</dependency>
</dependencies>
```

```
<dependencies>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-data-jpa</artifactId>
<version>version-number</version>
</dependency>
</dependencies>
```

Repository - JPA-managed entity

The Spring Data allows application programmers to work with data stores using consistent interfaces.

These three core interfaces of Spring Data are:

1) Repository,

2) CrudRepository

3) PagingAndSortingRepository

Repositories

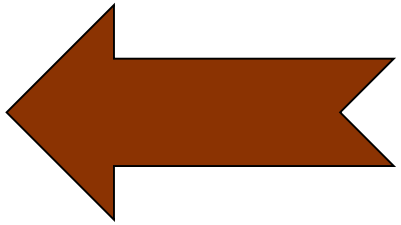
- Application programmers can access data in a consistent way.
- It is easy to switch the underlying storage for a domain entity.
- Specific implementations can provide features specific to data stores.

Repositories

```
@Entity
@Table(name = "Employee")
public class Emp {
   @Id
   @Column(name = "EmpId")
    private Long id;

   @Column(name = "EmpName")
  private String name;
//setter
//getter
//constuctor
//toString

    }
```



Creating repository for JPA managed class using entity class

Repositories

Creating repository for JPA managed class using entity class

```
public interface EmpRepository extends
CrudRepository<Emp, Long>
 {
   public User findByName(String name); }
```

Name must be an attribute on the Emp entity class.
The method name must begin with find, get or read.

Summary

Explore more on spring data

- Gives an overview on Spring Data, its modules and repositories.
- Gives birds view on Spring Data JDBC, Spring Data JPA

Thank you

Inn

Delighted Customers