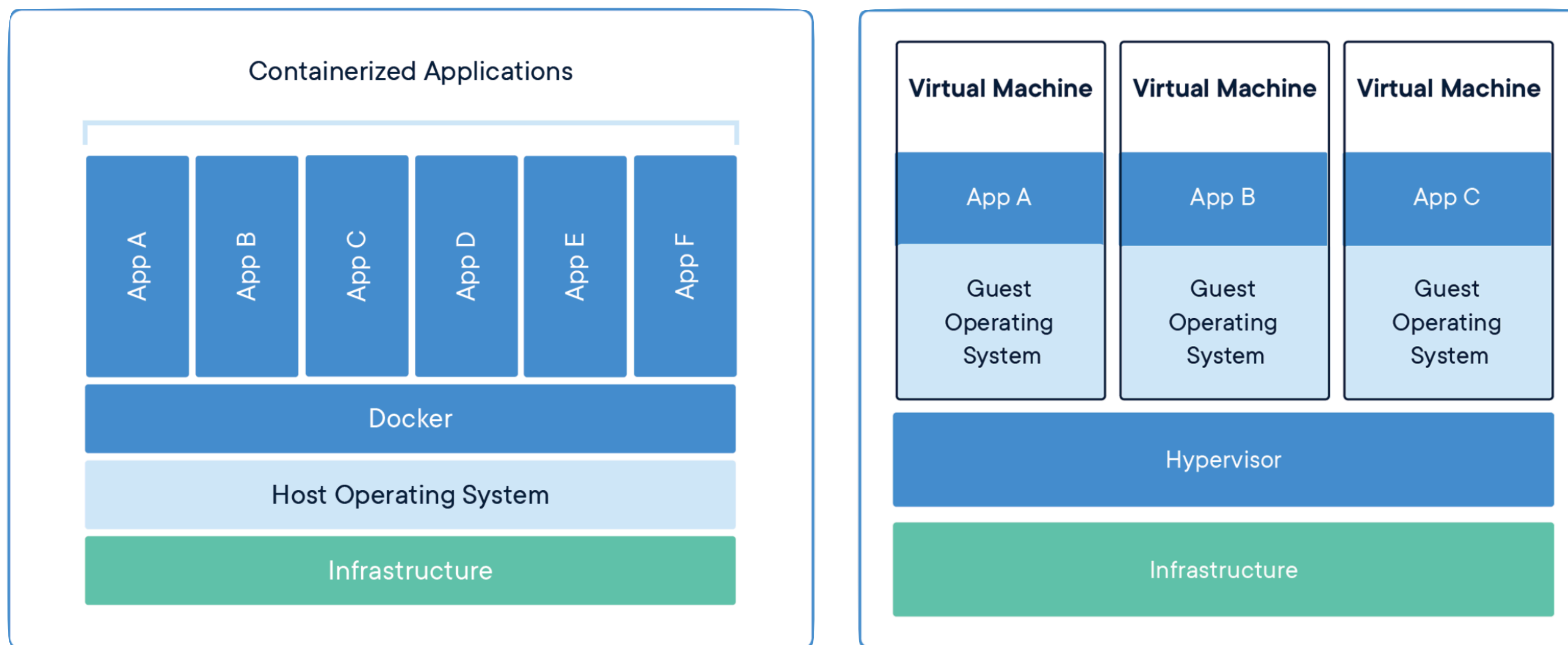


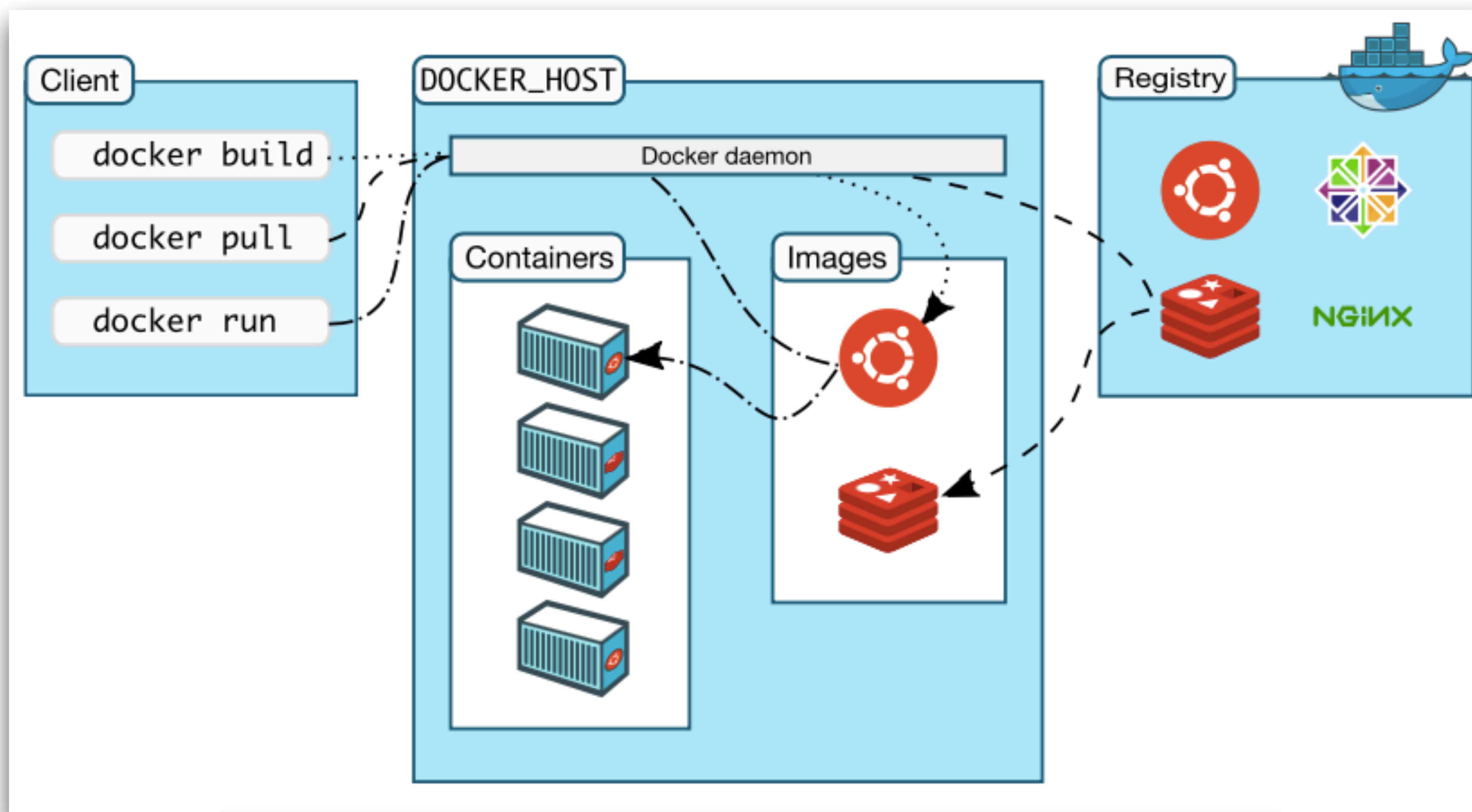
# NoSQL 实践

# 通过 Docker 辅助开发

# 什么是容器



# 认识 Docker



# 不同人眼中的 Docker

## 开发眼中的 Docker

- 简化了重复搭建开发环境的工作

## 运维眼中的 Docker

- 交付系统更为流畅
- 伸缩性更好

# Docker 常用命令

## 镜像相关

- `docker pull <image>`
- `docker search <image>`

## 容器相关

- `docker run`
- `docker start/stop <容器名>`
- `docker ps <容器名>`
- `docker logs <容器名>`

# docker run 的常用选项

**docker run [OPTIONS] IMAGE [COMMAND] [ARG...]**

## 选项说明

- -d, 后台运行容器
- -e, 设置环境变量
- --expose / -p 宿主端口:容器端口
- --name, 指定容器名称
- --link, 链接不同容器
- -v 宿主目录:容器目录, 挂载磁盘卷

# 国内 Docker 镜像配置

## 官方 Docker Hub

- <https://hub.docker.com>

## 官方镜像

- 镜像 <https://www.docker-cn.com/registry-mirror>
- 下载 <https://www.docker-cn.com/get-docker>

## 阿里云镜像

- <https://dev.aliyun.com>



# 通过 Docker 启动 MongoDB

## 官方指引

- [https://hub.docker.com/\\_/mongo](https://hub.docker.com/_/mongo)

## 获取镜像

- `docker pull mongo`

## 运行 MongoDB 镜像

- `docker run --name mongo -p 27017:27017 -v ~/docker-data/mongo:/data/db -e MONGO_INITDB_ROOT_USERNAME=admin -e MONGO_INITDB_ROOT_PASSWORD=admin -d mongo`

# 通过 Docker 启动 MongoDB

## 登录到 MongoDB 容器中

- `docker exec -it mongo bash`

## 通过 Shell 连接 MongoDB

- `mongo -u admin -p admin`

# 在 Spring 中访问 MongoDB

# Spring 对 MongoDB 的支持

**MongoDB 是一款开源的文档型数据库**

- <https://www.mongodb.com>

**Spring 对 MongoDB 的支持**

- Spring Data MongoDB
  - `MongoTemplate`
  - Repository 支持

# Spring Data MongoDB 的基本用法

## 注解

- @Document
- @Id

## MongoTemplate

- save / remove
- Criteria / Query / Update

# 初始化 MongoDB 的库及权限

## 创建库

```
use springbucks;
```

## 创建用户

```
db.createUser(  
  {  
    user: "springbucks",  
    pwd: "springbucks",  
    roles: [  
      { role: "readWrite", db: "springbucks" }  
    ]  
  }  
)
```

**“Talk is cheap, show me the code.”**

*Chapter 4 / mongo-demo*

# Spring Data MongoDB 的 Repository

**@EnableMongoRepositories**

对应接口

- `MongoRepository<T, ID>`
- `PagingAndSortingRepository<T, ID>`
- `CrudRepository<T, ID>`



**“Talk is cheap, show me the code.”**

*Chapter 4 / mongo-repository-demo*

# 在 Spring 中访问 Redis

# Spring 对 Redis 的支持

**Redis 是一款开源的内存 KV 存储，支持多种数据结构**

- <https://redis.io>

## **Spring 对 Redis 的支持**

- Spring Data Redis
  - 支持的客户端 Jedis / Lettuce
  - RedisTemplate
  - Repository 支持

# Jedis 客户端的简单使用

- Jedis 不是线程安全的
- 通过 JedisPool 获得 Jedis 实例
- 直接使用 Jedis 中的方法

# Jedis 客户端的简单使用

```
@Bean
@ConfigurationProperties("redis")
public JedisPoolConfig jedisPoolConfig() {
    return new JedisPoolConfig();
}

@Bean(destroyMethod = "close")
public JedisPool jedisPool(@Value("${redis.host}") String host) {
    return new JedisPool(jedisPoolConfig(), host);
}
```

# 通过 Docker 启动 Redis

## 官方指引

- [https://hub.docker.com/\\_/redis](https://hub.docker.com/_/redis)

## 获取镜像

- `docker pull redis`

## 启动 Redis

- `docker run --name redis -d -p 6379:6379 redis`

**“Talk is cheap, show me the code.”**

*Chapter 4 / jedis-demo*

# Redis 的哨兵与集群模式

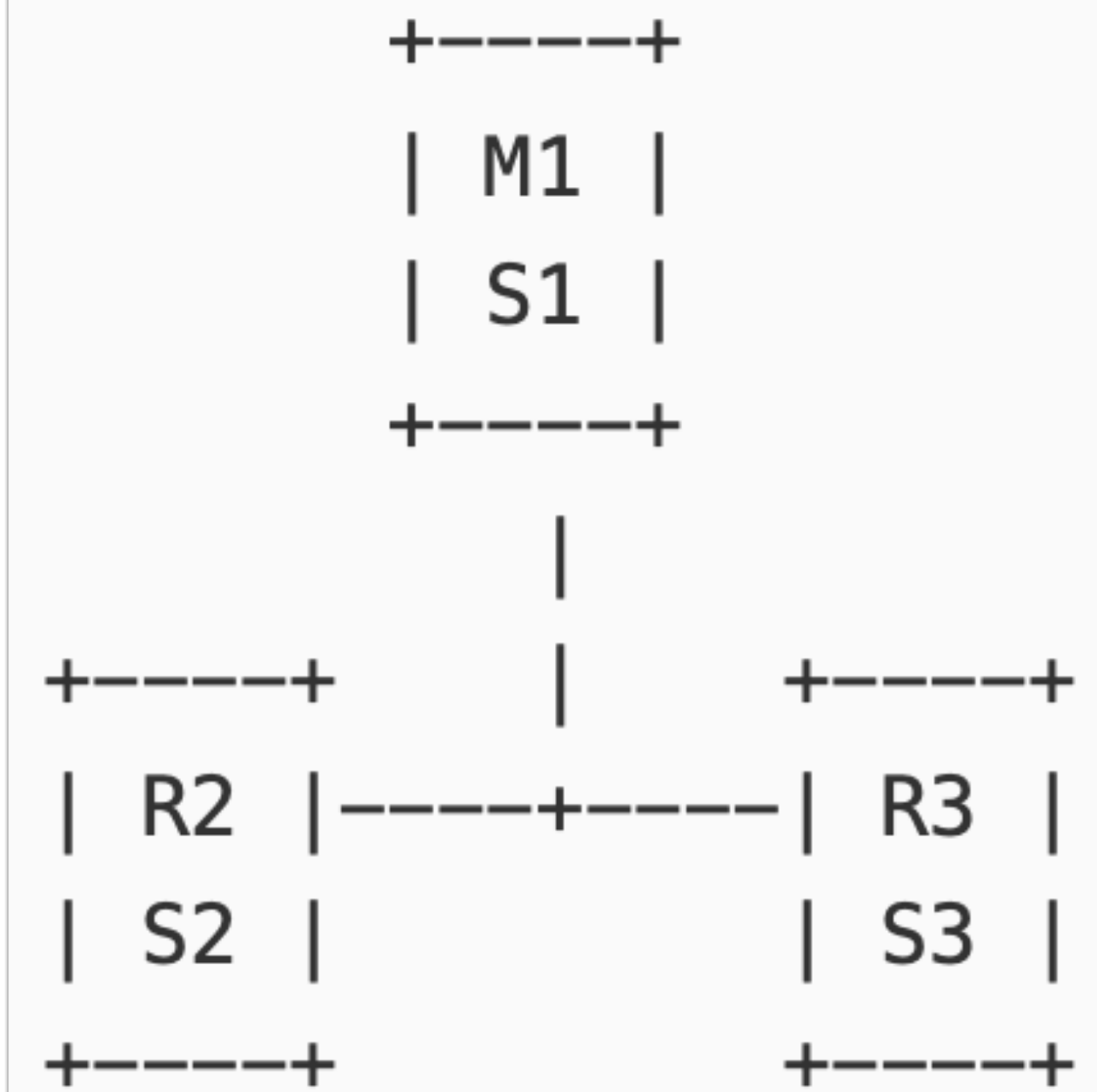


# Redis 的哨兵模式

**Redis Sentinel 是 Redis 的一种高可用方案**

- 监控、通知、自动故障转移、服务发现

**JedisSentinelPool**



Configuration: quorum = 2

# Redis 的集群模式

## Redis Cluster

- 数据自动分片（分成16384个 Hash Slot ）
- 在部分节点失效时有一定可用性

## JedisCluster

- Jedis 只从 Master 读数据，如果想要自动读写分离，可以定制

# 了解 Spring 的缓存抽象

# Spring 的缓存抽象

为不同的缓存提供一层抽象

- 为 Java 方法增加缓存，缓存执行结果
- 支持ConcurrentMap、EhCache、Caffeine、JCache (JSR-107)
- 接口
  - `org.springframework.cache.Cache`
  - `org.springframework.cache.CacheManager`

# 基于注解的缓存

## **@EnableCaching**

- @Cacheable
- @CacheEvict
- @CachePut
- @Caching
- @CacheConfig

**“Talk is cheap, show me the code.”**

*Chapter 4 / cache-demo*

# 通过 Spring Boot 配置 Redis 缓存

```
<dependency>  
    <groupId>org.springframework.boot</groupId>  
    <artifactId>spring-boot-starter-cache</artifactId>  
</dependency>  
<dependency>  
    <groupId>org.springframework.boot</groupId>  
    <artifactId>spring-boot-starter-data-redis</artifactId>  
</dependency>
```

```
spring.cache.type=redis  
spring.cache.cache-names=coffee  
spring.cache.redis.time-to-live=5000  
spring.cache.redis.cache-null-values=false  
  
spring.redis.host=localhost
```

# 通过 Spring Boot 配置 Redis 缓存

```
@Slf4j
@Service
@CacheConfig(cacheNames = "coffee")
public class CoffeeService {
    @Autowired
    private CoffeeRepository coffeeRepository;

    @Cacheable
    public List<Coffee> findAllCoffee() {
        return coffeeRepository.findAll();
    }

    @CacheEvict
    public void reloadCoffee() {
    }
}
```



**“Talk is cheap, show me the code.”**

*Chapter 4 / cache-with-redis-demo*

# Redis 在 Spring 中的其他用法

# 与 Redis 建立连接

## 配置连接工厂

- LettuceConnectionFactory 与 JedisConnectionFactory
  - RedisStandaloneConfiguration
  - RedisSentinelConfiguration
  - RedisClusterConfiguration

# 读写分离

Lettuce 内置支持读写分离

- 只读主、只读从
- 优先读主、优先读从

LettuceClientConfiguration

LettucePoolingClientConfiguration

LettuceClientConfigurationBuilderCustomizer

# RedisTemplate

RedisTemplate<K, V>

- opsForXxx()

StringRedisTemplate

一定要注意设置过期时间!!!

**“Talk is cheap, show me the code.”**

*Chapter 4 / redis-demo*

# Redis Repository

## 实体注解

- @RedisHash
- @Id
- @Indexed

# 处理不同类型数据源的 Repository

## 如何区分这些 Repository

- 根据实体的注解
- 根据继承的接口类型
- 扫描不同的包



**“Talk is cheap, show me the code.”**

*Chapter 4 / redis-repository-demo*

# SpringBucks 进度小结

# 本章小结

- 了解了 Docker 在本地的基本用法
- 了解了 Spring Data MongoDB 的基本用法
- 了解了 Spring Data Redis 的基本用法
- 了解了 Redis 的几种运行模式
- 了解了 Spring 的缓存抽象

# SpringBucks 进度小结

- 使用不同类型的数据库存储咖啡信息
- 结合 JPA 与 Redis 来优化咖啡信息的存储