



# Discovery 【探索】

蓝绿灰度发布  
最佳实践



<http://www.nepxion.com>

<https://github.com/Nepxion>

©2017-2050 Nepxion Studio. All Rights Reserved.

Let us start Discovery



**Discovery** 【探索】



# 最佳实践

流量染色：组染色

## 参数方式

- 染色方式：启动参数 `java -jar -Dmetadata.group=nepxion abc.jar`

## 配置方式

- 染色方式：配置文件 `spring.cloud.discovery.metadata.group=nepxion`



Discovery Wiki <http://nepxion.com/discovery>  
Polaris Wiki <http://nepxion.com/polaris>



# 最佳实践

## 流量染色：版本染色

### 静态方式

版本不可排序模式

- 版本格式：服务实例旧版本赋值为green，新版本赋值为blue，每次上线交替使用
- 染色方式：启动参数 `java -jar -Dmetadata.version=blue abc.jar`

### 动态方式

版本可排序模式

- ① 基础架构平台以编译插件git-commit-id-plugin进行染色
  - 版本格式：例如，**20210601-567**，日期 + Git提交次数
  - 染色方式：权力下放给基础架构平台，自动染色，不需要加启动参数（参考《附录参考：流量染色》）
- ② 基础架构平台以POM版本号进行染色
  - 版本格式：例如，**1.0.0**，**1.0.1**，**1.1.0**
  - 染色方式：读取业务服务POM中的版本号作为服务实例的版本号（参考《》）
- ③ DevOps运维平台以时间戳方式进行染色
  - 版本格式：例如，**20210601-0003**，时期 + 指定服务当天发布次数
  - 染色方式：权力下放给DevOps运维平台，需要加启动参数 `-Dmetadata.version=20210601-0003`
- ④ DevOps运维平台以数字递增方式进行染色
  - 版本格式：例如，**1.0.0**，**1.0.1**，**1.1.0**
  - 染色方式：权力下放给DevOps运维平台，需要加启动参数 `-Dmetadata.version=1.0.0`



Discovery Wiki <http://nepxion.com/discovery>  
Polaris Wiki <http://nepxion.com/polaris>



# 最佳实践

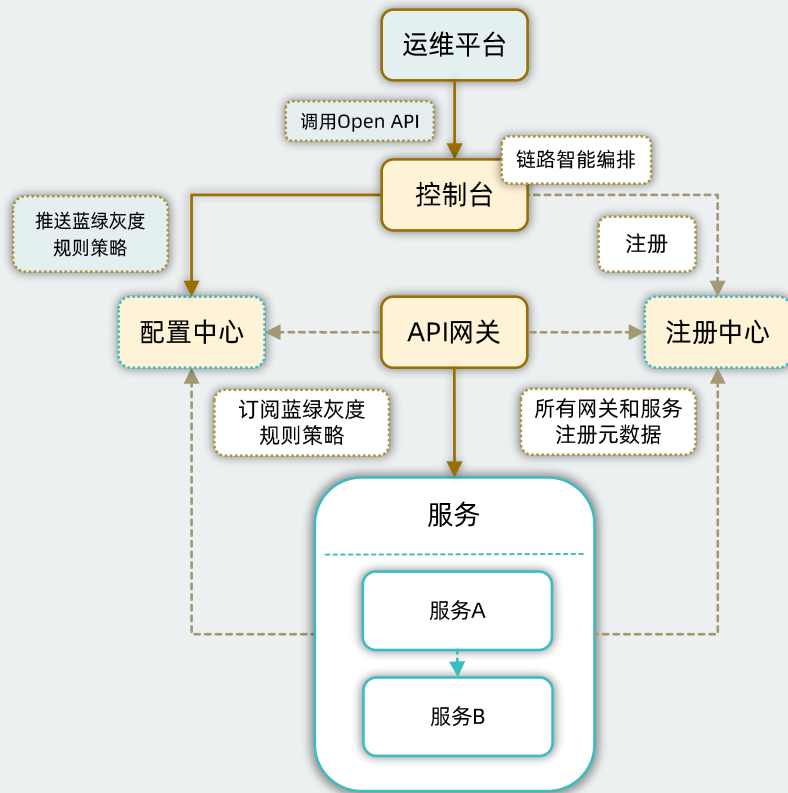
对接DevOps运维平台实施蓝绿灰度发布

## 架构

- ① 控制台需要连接注册中心和配置中心
- ② 控制台建议实现高可用架构，控制台前面部署API网关和运维平台对接

## 方案

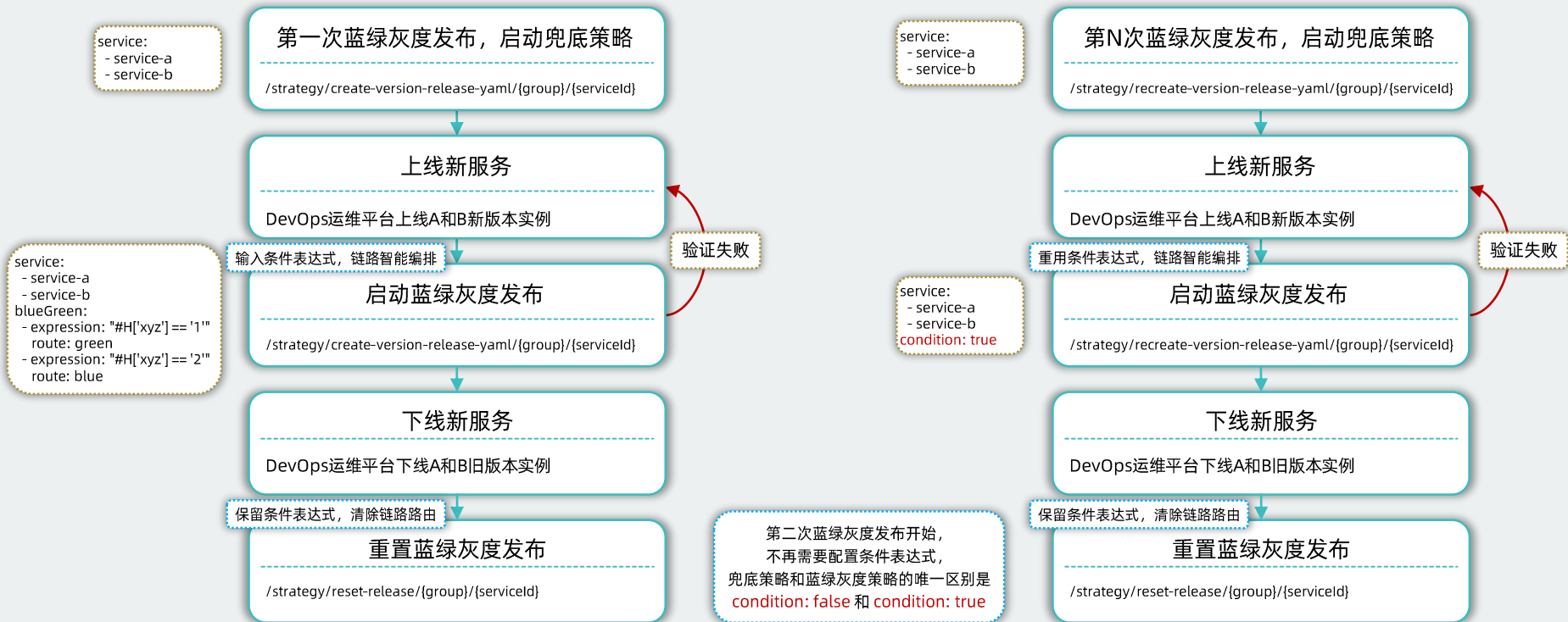
- ① 运维平台调用控制台的Open API，控制台进行链路智能编排
- ② 控制台把最终蓝绿灰度规则策略推送到配置中心





# 最佳实践

## 对接DevOps运维平台实施蓝绿灰度发布链路智能编排流程





# 最佳实践

## 全链路智能编排实施蓝绿灰度发布

### 概念

- ① 路由链路在后台会智能化编排
- ② 无需关心服务版本的信息
- ③ 只需配置条件表达式

### 方案

- ① 向控制台发送请求
- ② 控制台根据新旧版本的判断，智能编排出两条新旧路由链路，并给它们赋予不同的条件表达式
- ③ 解析简单规则策略为最终规则策略，保存至配置中心

#### 兜底规则策略

```
service:
- a
- b
sort: version
```

#### 蓝绿规则策略

```
service:
- a
- b
blueGreen:
- expression: "#H['xyz'] == '1'"
  route: green
- expression: "#H['xyz'] == '2'"
  route: blue
sort: version
```

#### 灰度规则策略

```
service:
- a
- b
gray:
- expression: "#H['xyz'] == '3'"
  weight:
- 90
- 10
- expression: "#H['xyz'] == '4'"
  weight:
- 70
- 30
- weight:
- 100
- 0
sort: version
```

#### 二次蓝绿灰度规则策略

```
service:
- a
- b
condition: true
sort: version
```

#### 规则策略解释

- ① 指定要实施的服务列表（例如，**a**和**b**）
- ② 版本号排序类型（**sort**），可选值为**version**和**time**，缺省为**version**（不需要配置**sort: version**）  
当排序类型为**version**时，适用于版本号采用时间戳或者数字递增的方式。处理逻辑为将排序后版本号列表的第一个值作为旧的稳定版本  
当排序类型为**time**时，不限于版本号的格式。处理逻辑为将根据服务实例全局唯一ID的时间戳前缀进行排序，把上线时间最早的服务实例的版本号作为旧的稳定版本
- ③ 兜底规则策略无需指定条件（**expression**）
- ④ 蓝绿规则策略必须指定蓝绿两个条件（**expression**）和路由（**route**）项  
路由**green**代表绿（旧版本）路由链路，路由**blue**代表蓝（新版本）路由链路
- ⑤ 灰度规则策略条件和权重项可以无数个（不同条件下的不同权重配比），允许有一项条件（**expression**）为空（无条件驱动下的权重）  
灰度权重（**weight**）必须为两个整数（可以大于100）的数字，按次序为别代表稳定（旧版本）路由链路权重，灰度（新版本）路由链路权重
- ⑥ 上述三个规则策略组合在一起，为蓝绿灰度混合发布
- ⑦ 二次蓝绿灰度，表示第一次执行过蓝绿灰度发布后，会保留条件（**expression**）项，清除路由（**route**）项，以后蓝绿灰度不必填写条件（**expression**）项，只需用**condition: true**代替即可
- ⑧ 支持Yaml和Json两种格式



# 最佳实践

## 全链路自动化模拟流程测试

### 方案

- ① 全链路智能编排 + 流量侦测
- ② 网关或服务为侦测入口
- ③ 自动判断结果准确性
- ④ 服务引入discovery-plugin-admin-center-starter依赖

### 过程

- ① 启动控制台
- ② 修改配置和规则策略
- ③ 执行命令行

### 适用

- ① 测试环境
- ② 开发环境

### 测试结果部分示例

【模拟场景3】蓝绿策略，测试全链路侦测，Header xyz等于1...  
侦测次数：100  
侦测结果：discovery-guide-service-a@1.1命中次数=0  
侦测结果：discovery-guide-service-a@1.0命中次数=100  
侦测结果：discovery-guide-service-b@1.1命中次数=0  
侦测结果：discovery-guide-service-b@1.0命中次数=100  
测试结果：通过  
测试耗时：3 秒

【模拟场景3】灰度策略，测试全链路侦测，Header xyz等于3...  
侦测次数：500  
侦测进度：第100次...  
侦测进度：第200次...  
侦测进度：第300次...  
侦测进度：第400次...  
侦测进度：第500次...  
侦测结果：discovery-guide-service-a@1.1命中次数=52  
侦测结果：discovery-guide-service-a@1.0命中次数=448  
侦测结果：discovery-guide-service-b@1.1命中次数=52  
侦测结果：discovery-guide-service-b@1.0命中次数=448  
权重结果偏差值=5%  
期望结果：旧版本路由权重=90%，新版本路由权重=10%  
最终结果：旧版本路由权重=89.6%，新版本路由权重=10.4%  
测试结果：通过  
测试耗时：12 秒





# 最佳实践

## 全链路自动化流量侦测测试

### 方案

- ① 全链路流量侦测
- ② 网关或服务为侦测入口
- ③ 人工判断结果准确性
- ④ 服务引入discovery-plugin-admin-center-starter依赖

### 过程

- ① 修改配置
- ② 执行命令行

### 适用

- ① 生产环境

### 测试结果部分示例

【侦测场景1】测试全链路侦测...

侦测次数：10

侦测结果：[ID=discovery-guide-gateway][V=1.0] -> [ID=discovery-guide-service-a][V=1.0] -> [ID=discovery-guide-service-b][V=1.0]

侦测结果：[ID=discovery-guide-gateway][V=1.0] -> [ID=discovery-guide-service-a][V=1.1] -> [ID=discovery-guide-service-b][V=1.1]

侦测结果：[ID=discovery-guide-gateway][V=1.1] -> [ID=discovery-guide-service-a][V=1.0] -> [ID=discovery-guide-service-b][V=1.0]

侦测结果：[ID=discovery-guide-gateway][V=1.1] -> [ID=discovery-guide-service-a][V=1.1] -> [ID=discovery-guide-service-b][V=1.1]

侦测结果：[ID=discovery-guide-gateway][V=1.0] -> [ID=discovery-guide-service-a][V=1.0] -> [ID=discovery-guide-service-b][V=1.0]

侦测结果：[ID=discovery-guide-gateway][V=1.0] -> [ID=discovery-guide-service-a][V=1.1] -> [ID=discovery-guide-service-b][V=1.1]

侦测结果：[ID=discovery-guide-gateway][V=1.1] -> [ID=discovery-guide-service-a][V=1.0] -> [ID=discovery-guide-service-b][V=1.0]

侦测结果：[ID=discovery-guide-gateway][V=1.1] -> [ID=discovery-guide-service-a][V=1.1] -> [ID=discovery-guide-service-b][V=1.1]

侦测结果：[ID=discovery-guide-gateway][V=1.0] -> [ID=discovery-guide-service-a][V=1.0] -> [ID=discovery-guide-service-b][V=1.0]

侦测结果：[ID=discovery-guide-gateway][V=1.0] -> [ID=discovery-guide-service-a][V=1.1] -> [ID=discovery-guide-service-b][V=1.1]

测试耗时：0 秒



## 附录参考

### 流量染色

#### Git编译插件染色

- 添加编译插件git-commit-id-plugin

```
<plugin>
```

```
<groupId>pl.project13.maven</groupId>
```

```
<artifactId>git-commit-id-plugin</artifactId>
```

```
<executions>
```

```
<execution>
```

```
<goals>
```

```
<goal>revision</goal>
```

```
</goals>
```

```
</execution>
```

```
</executions>
```

```
<configuration>
```

```
<generateGitPropertiesFile>true</generateGitPropertiesFile>
```

```
<dateFormat>yyyyMMdd</dateFormat>
```

```
</configuration>
```

```
</plugin>
```

- 开启开关

```
spring.application.git.generator.enabled=true
```



Discovery Wiki <http://nepxion.com/discovery>  
Polaris Wiki <http://nepxion.com/polaris>



## 附录参考

### 流量染色

#### POM版本号染色

- 基于环境装载EnvironmentPostProcessor设置

```
public class MyEnvironmentPostProcessor implements EnvironmentPostProcessor, Ordered {  
    @Override  
    public void postProcessEnvironment(ConfigurableEnvironment environment, SpringApplication application) {  
        if (EnvironmentUtil.isStandardEnvironment(environment)){  
            // 获取业务服务的POM版本号pomVersion  
            PluginMetaDataPreInstallation.getMetadata().put(DiscoveryConstant.VERSION, pomVersion);  
        }  
    }  
  
    @Override  
    public int getOrder() {  
        return Ordered.HIGHEST_PRECEDENCE;  
    }  
}
```

- 在src/main/resources/META-INF/spring.factories加上  
org.springframework.boot.env.EnvironmentPostProcessor=\ncom.xxx.yyy.zzz.MyEnvironmentPostProcessor



Discovery Wiki <http://nepxion.com/discovery>  
Polaris Wiki <http://nepxion.com/polaris>



# 文档指南

文档指南和联系方式



主页地址: <http://www.nepxion.com>

源码地址: <https://github.com/Nepxion/Discovery>

镜像地址: <https://gitee.com/Nepxion/Discovery>

指南地址: <https://github.com/Nepxion/DiscoveryGuide>

框架文档: <https://nepxion.com/discovery>

平台文档: <https://nepxion.com/discovery-platform>

微信



钉钉



公众号



文档



Thanks for Watching



**Discovery** 【探索】