

Code Velocity： 环境 & 持续测试 & 代码门禁

林紫嫣（蚂蚁国际高级测试开发专家）

目录

CONTENTS

1 什么是Code Velocity

2 测试左移：代码门禁

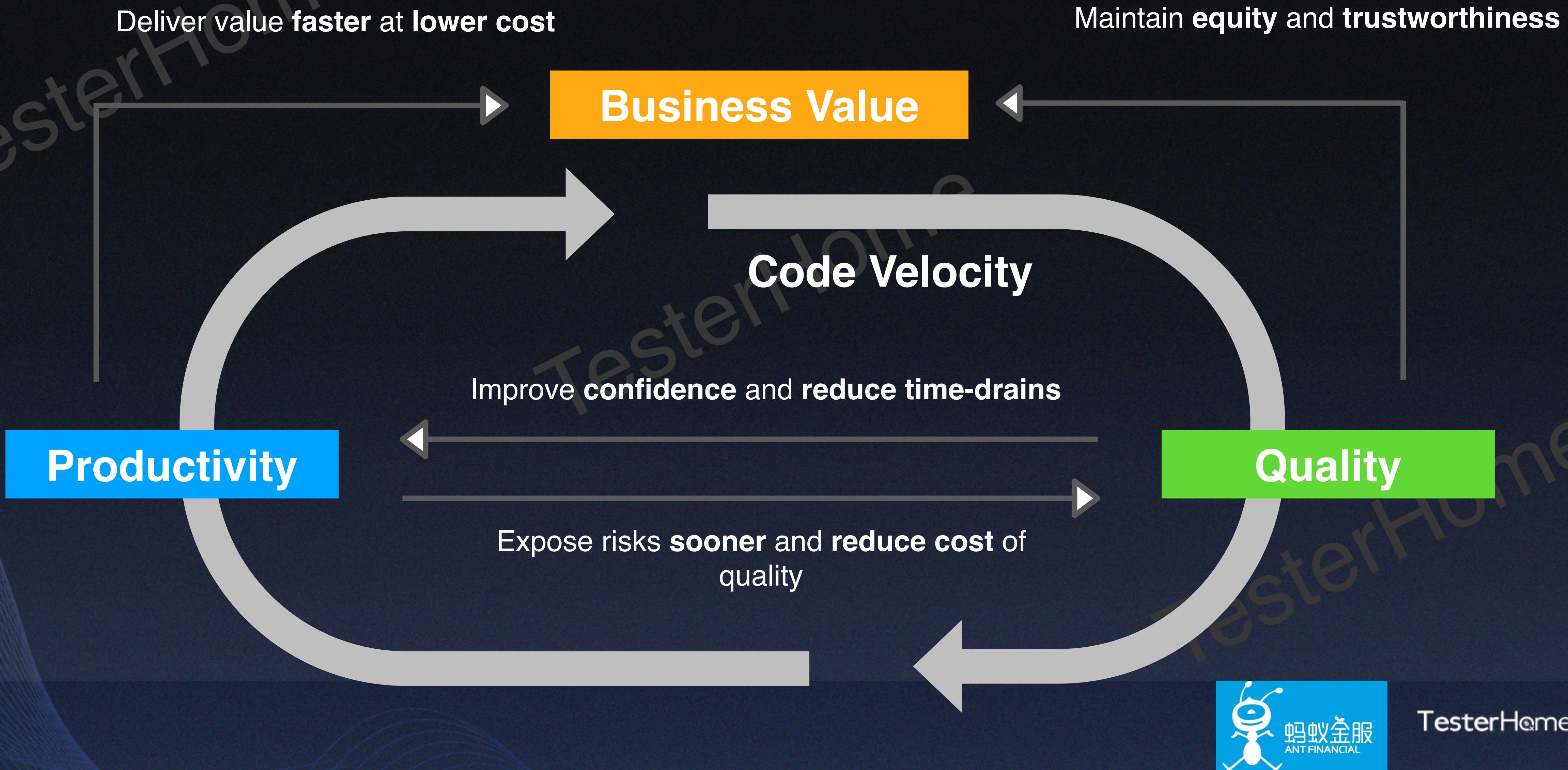
3 持续测试怎么做

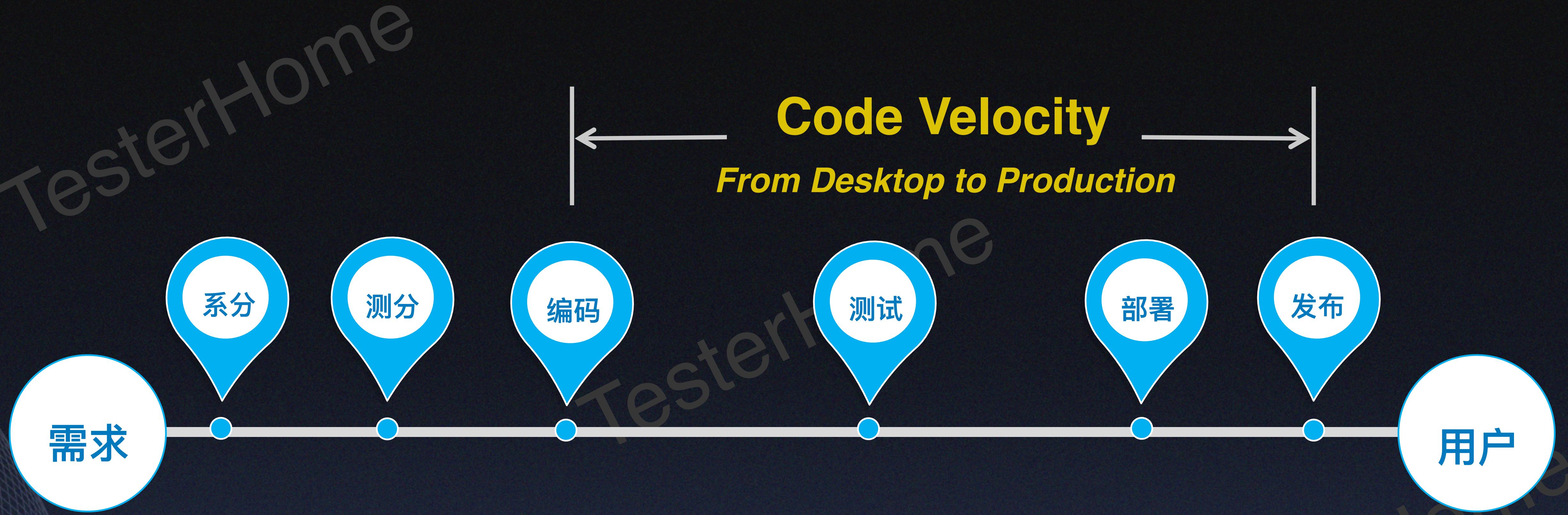
4 一键拉起全量隔离环境

5 蚂蚁国际测试技术核心理念

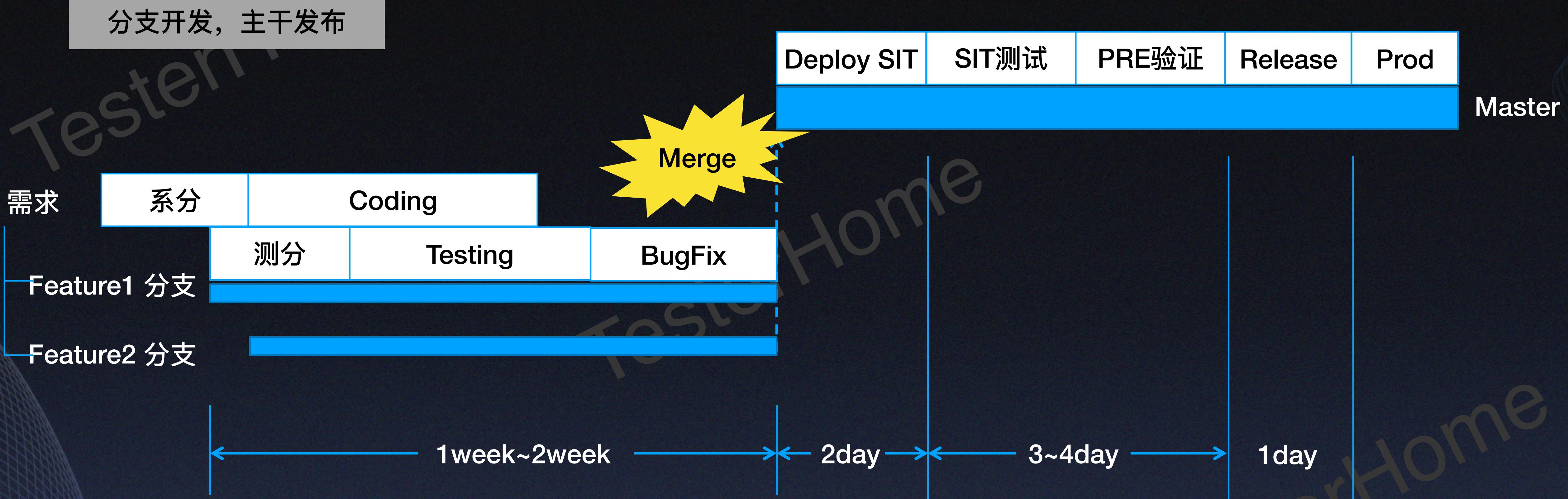


TesterHome



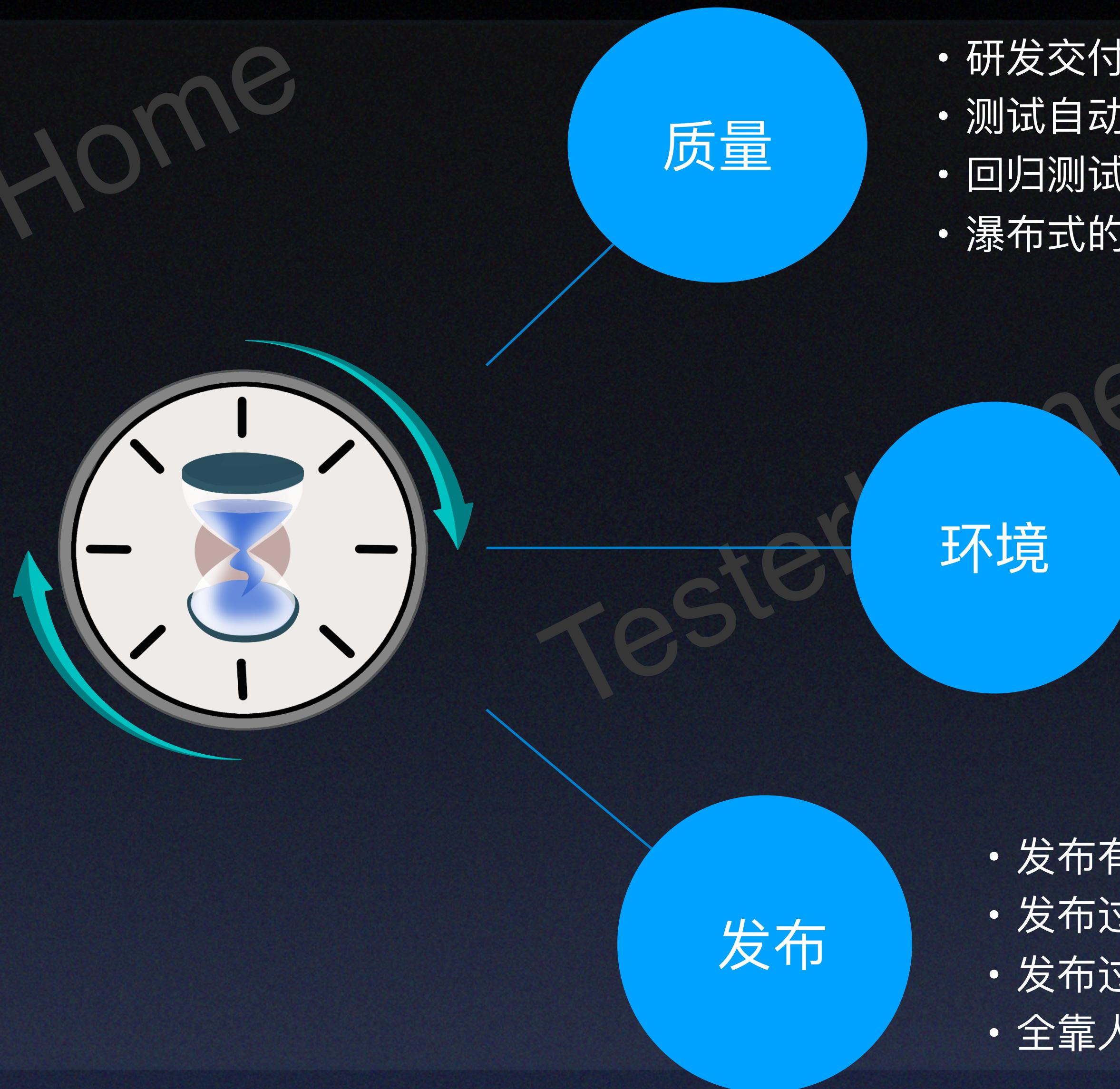


如何衡量Code Velocity



Code Velocity = 从代码提交到在线上运行，通常需要2~4week

为什么快不起来?



如何提高Code Velocity?



1 什么是Code Velocity

2 测试左移：代码门禁

3 持续测试怎么做

4 一键拉起全量隔离环境

5 蚂蚁国际测试技术核心理念

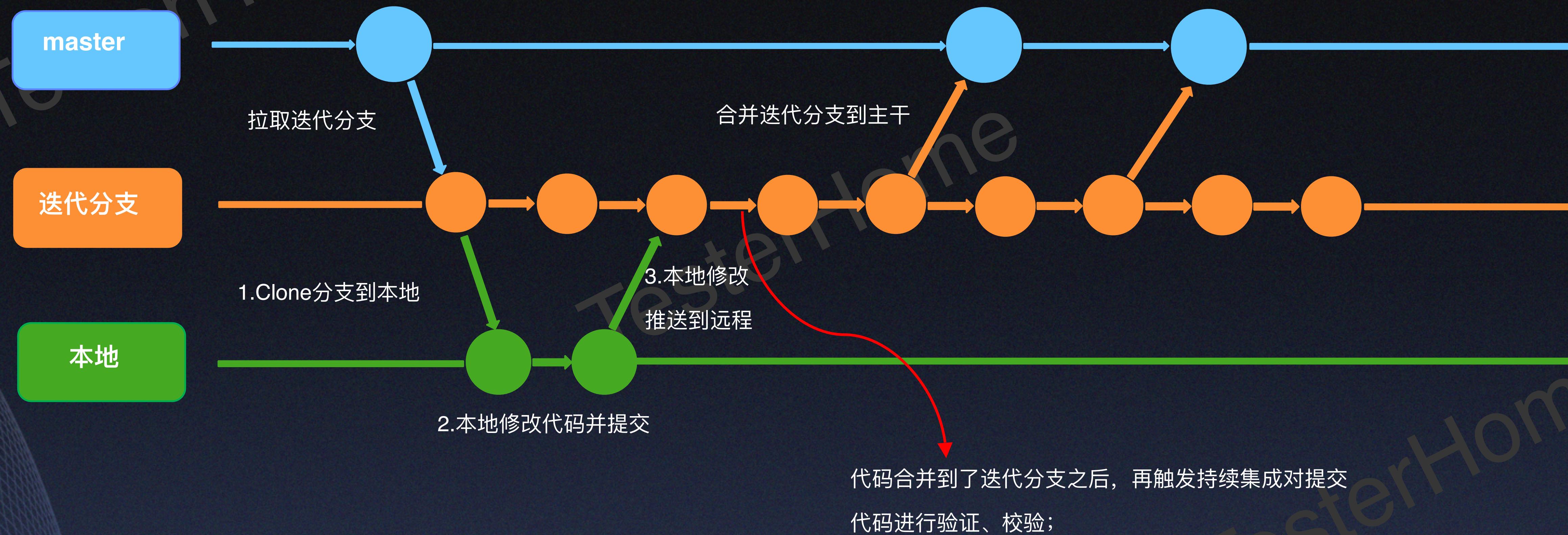
目录

CONTENTS

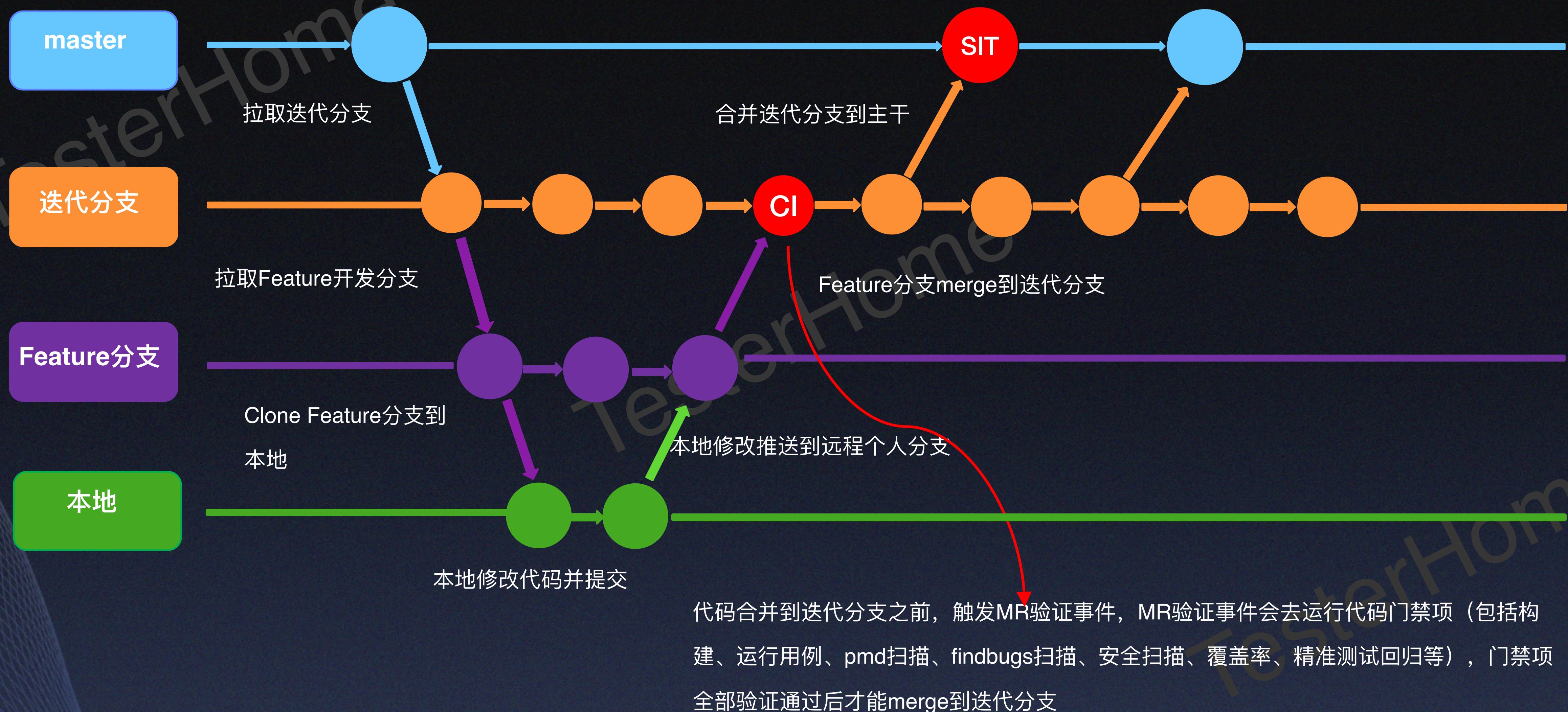


TesterHome

传统模式——代码先合并，后验证



代码门禁——代码先验证，后合并



代码门禁——质量卡点

门禁项目	说明	备注
编译	必须编译通过	阈值
Pmd 扫描	校验P0, P1 级别问题不能超过阈值	根据项目来定
findBug 扫描	Findbug 未修复问题不能超过阈值	根据项目来定
安全问题扫描	黑盒, 白盒安全扫描, 扫描出未修复问题数不能超过阈值	100%通过
覆盖率	行覆盖率、变更覆盖率、分支覆盖率不能低于阈值	90%覆盖
精准测试	精准测试的通过率不能低于阈值;	100%通过
全量用例测试	全量用例测试的通过率不低于阈值;	95%通过
CodeReview	需要code review 通过才能merge	必须signoff
其他门禁项	门禁检查项在不断丰富中.....	

CI991

- 90%通过率
- 90%覆盖率
- 10min执行完

高频

- 每次代码提交<300行
- 每日至少提交一次

稳定

- 执行时间<15min
- 门禁错误率<5%

目录

CONTENTS

1 什么是Code Velocity

2 测试左移：代码门禁

3 持续测试怎么做

4 一键拉起全量隔离环境

5 蚂蚁国际测试技术核心理念



TesterHome

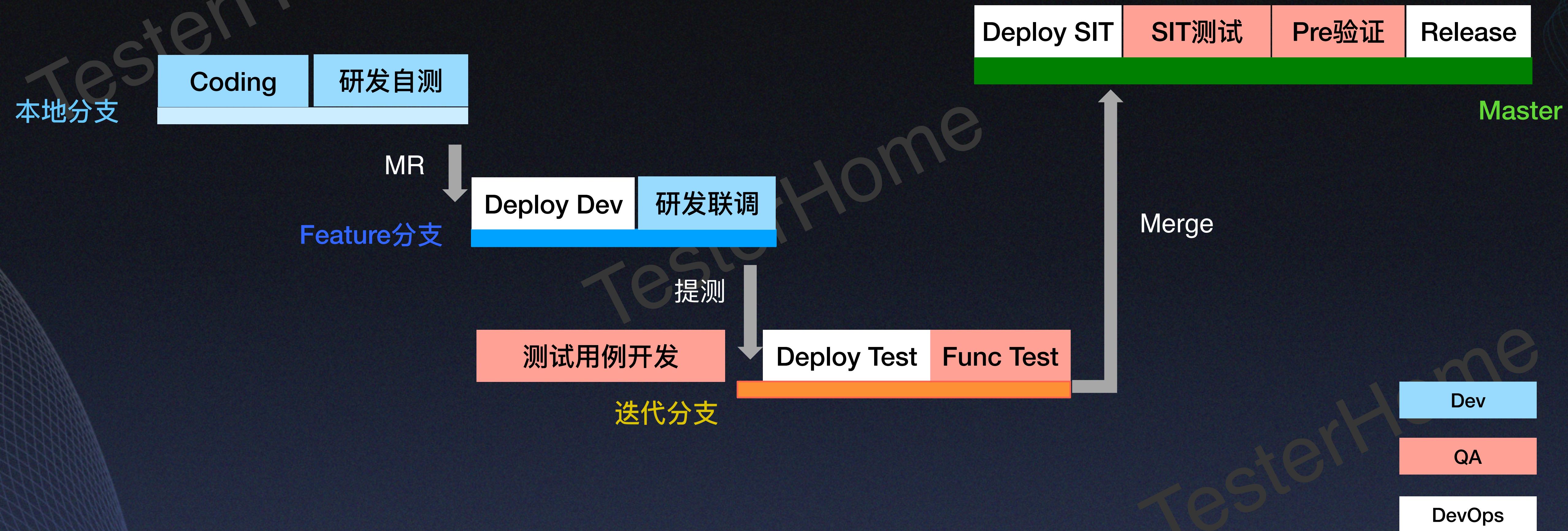
持续测试 = 持续联调 + 持续回归



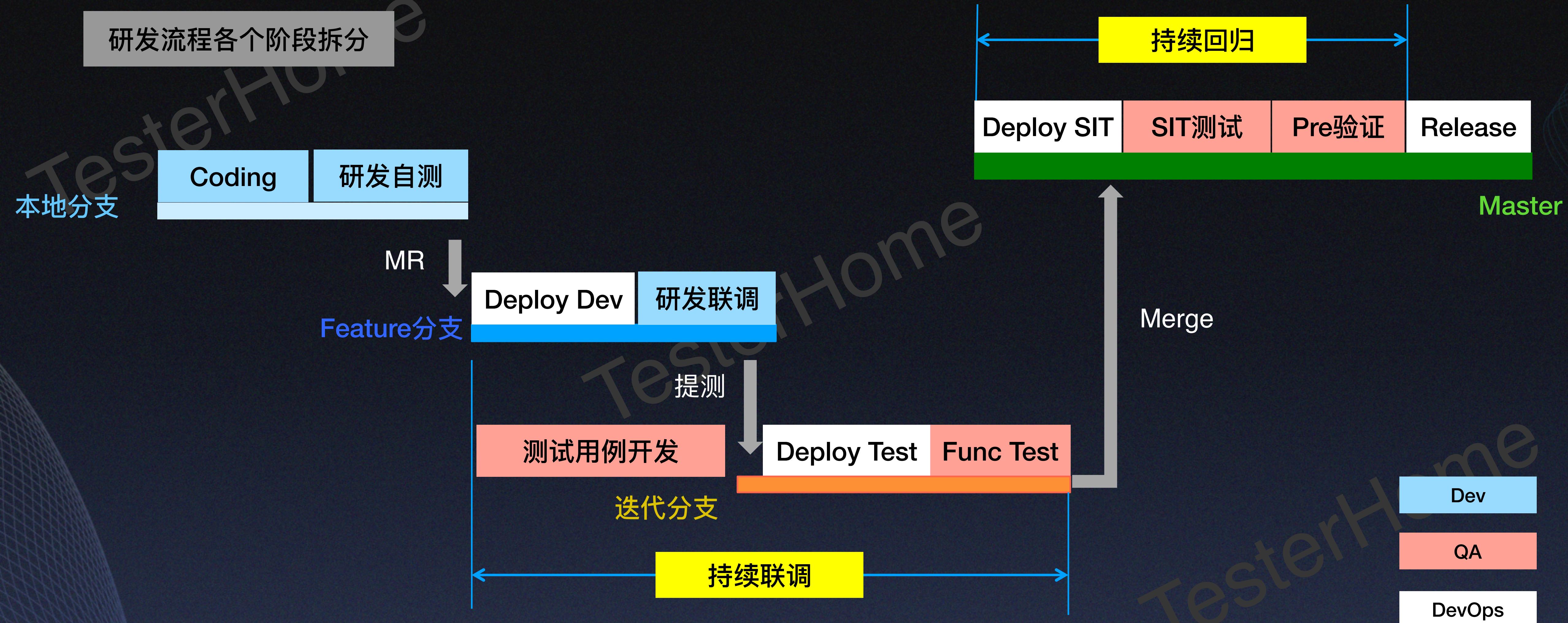
TesterHome

持续测试 = 持续联调 + 持续回归

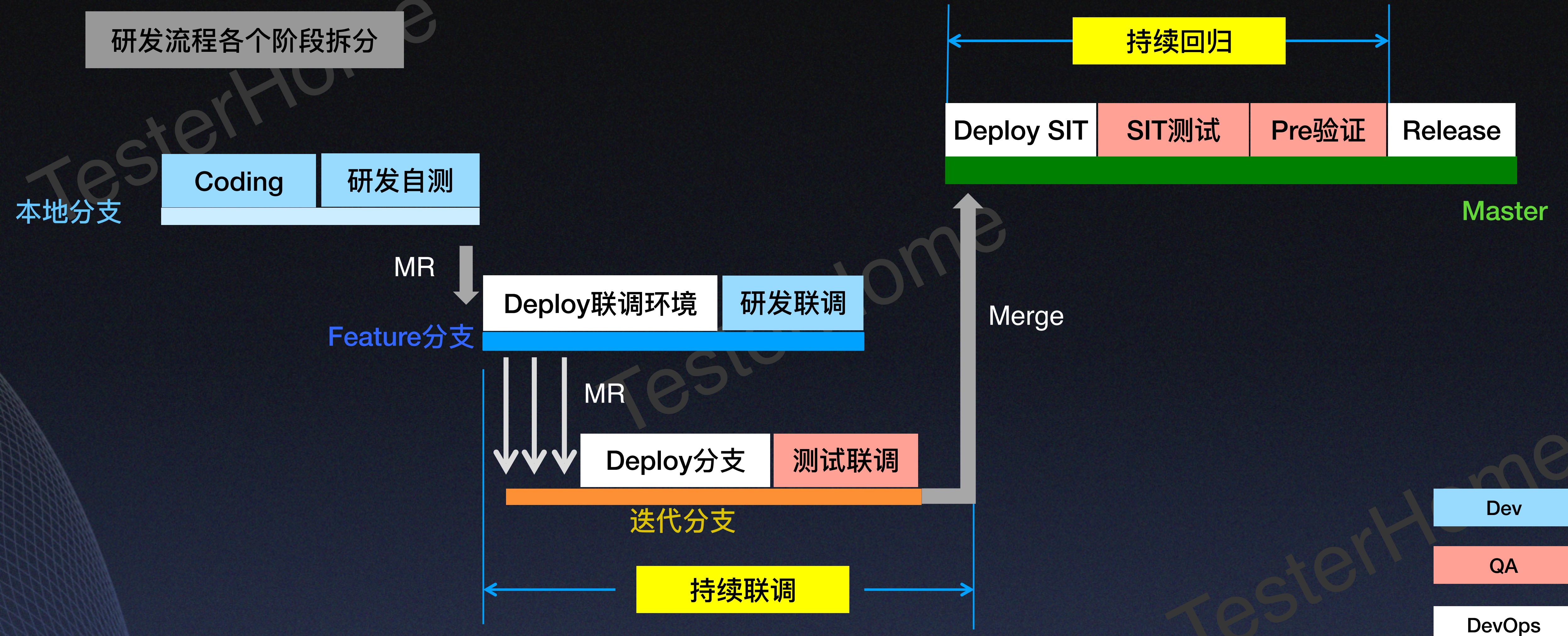
研发流程各个阶段拆分



持续测试 = 持续联调 + 持续回归



持续测试 = 持续联调 + 持续回归



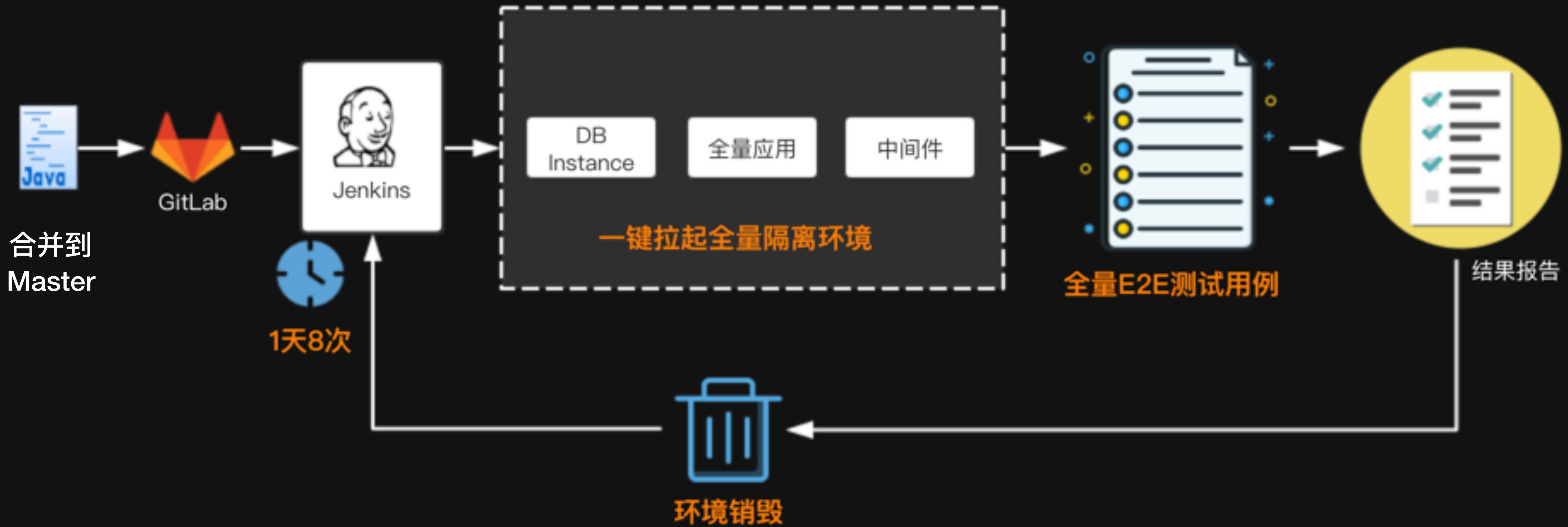
Dev

QA

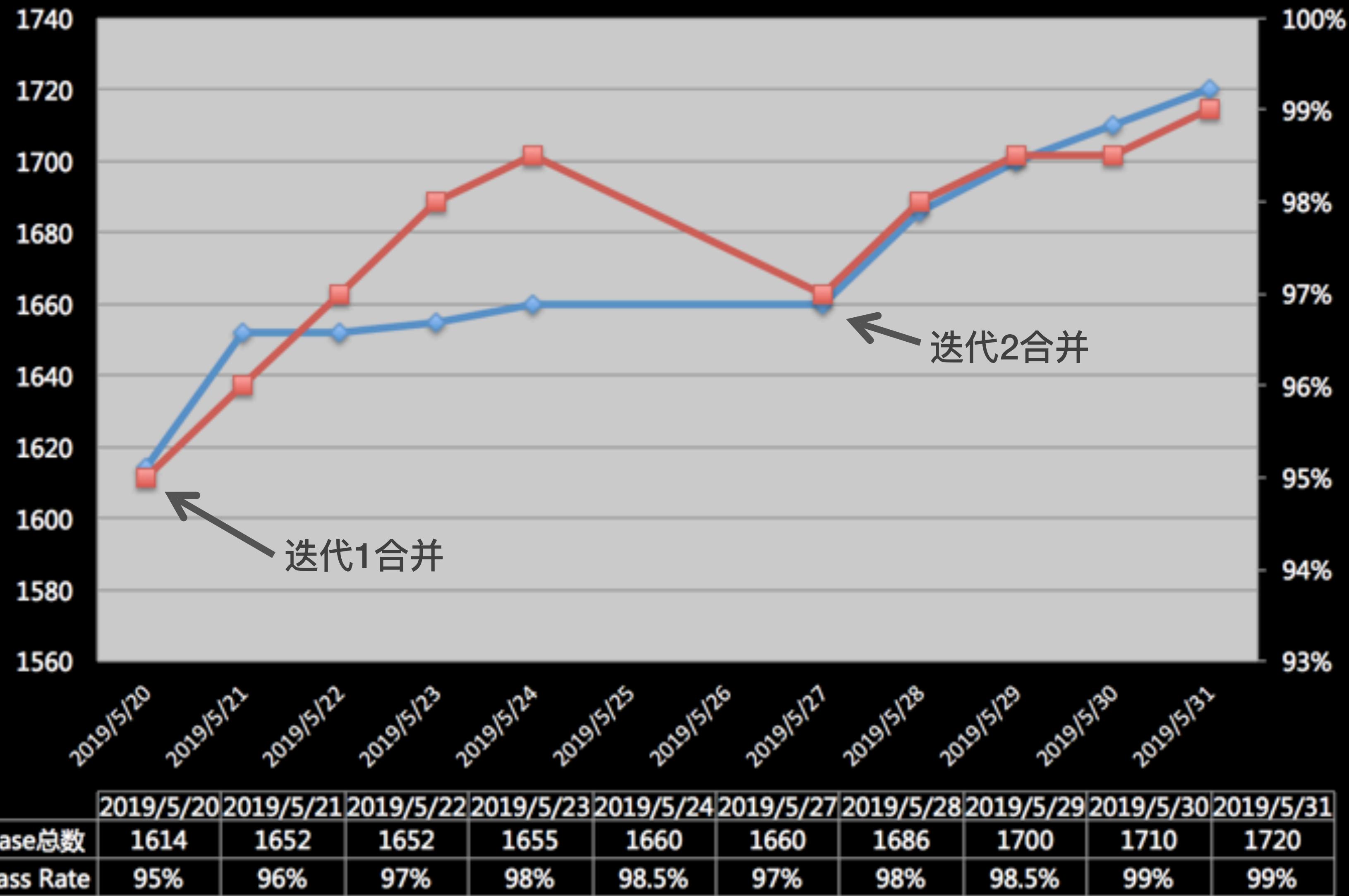
DevOps



持续回归



持续回归用例通过率趋势图



- 拉起主干全量环境
- 高频回归，用例始终保持较高通过率，主干代码稳定
- 在合并迭代时，当关联用例合入回归用例集，用例通过率略微有所下降

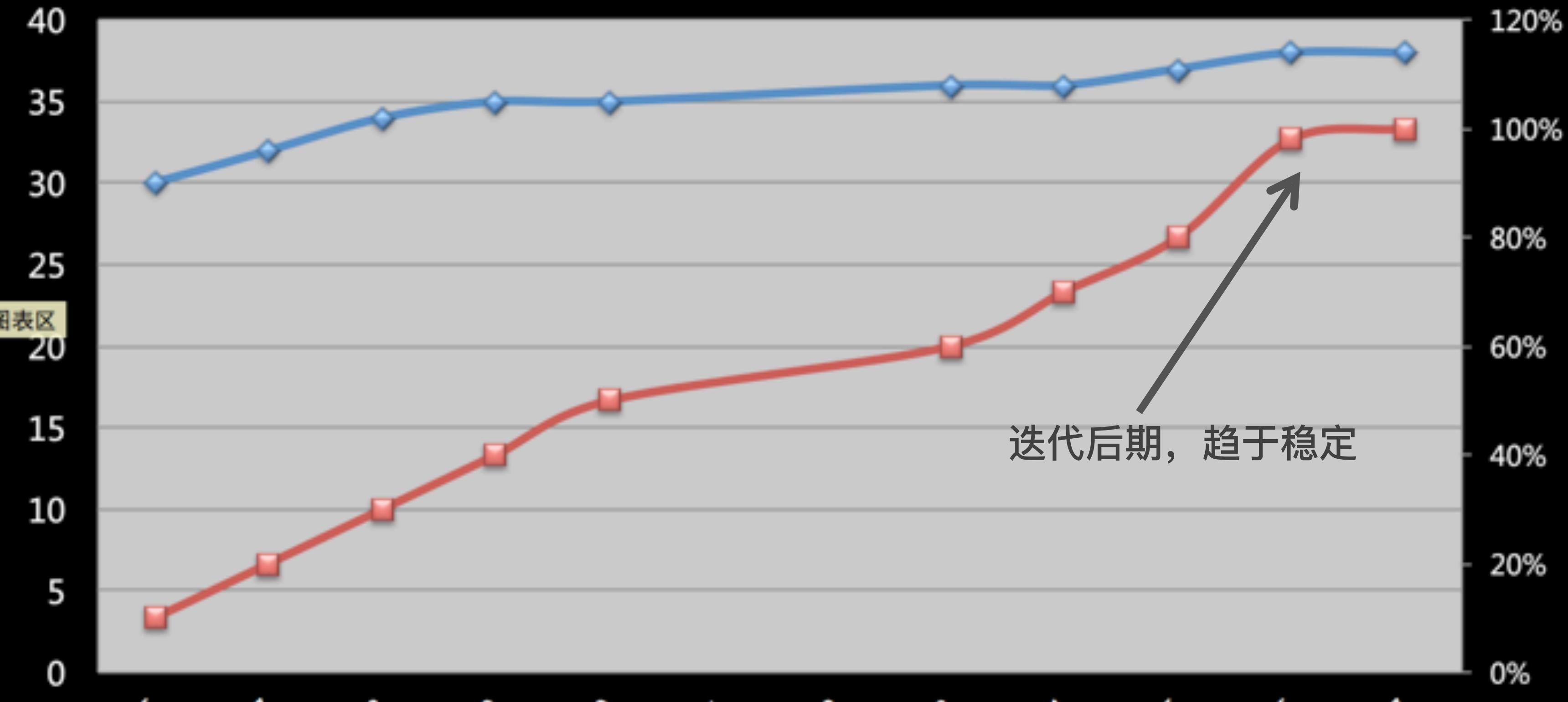
持续联调

环境隔离

用例通过率持续收敛



持续联调用例通过率趋势图



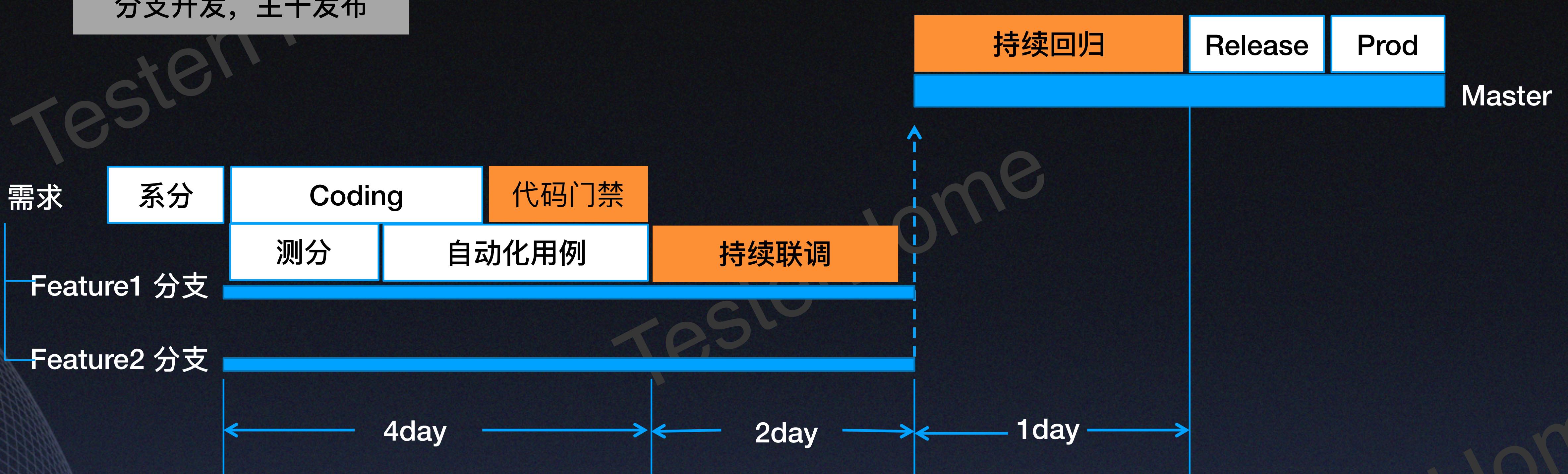
- 拉起分支联调环境
- 一次迭代中，随着测试用例的逐渐上升，通过率稳定上升

	2019/5/6	2019/5/7	2019/5/8	2019/5/9	2019/5/10	2019/5/13	2019/5/14	2019/5/15	2019/5/16	2019/5/17
Case总数	30	32	34	35	35	36	36	37	38	38
Pass Rate	10%	20%	30%	40%	50%	60%	70%	80%	98%	100%

持续回归 vs 持续联调

	持续回归	持续联调
拉起的应用系统	全量隔离环境	项目迭代变更的部分应用
代码&配置版本	Master Head	Branch Head
环境隔离情况	硬隔离	软隔离
环境拉起频率	一天多次	两天一次
执行用例	全量回归老用例	新增的联调用例

持续测试：提升研发质量和效率



Code Velocity = 从代码提交到在线上运行, 通常只需要 **7day**

目录

CONTENTS

1 什么是Code Velocity

2 测试左移：代码门禁

3 持续测试怎么做

4 一键拉起全量隔离环境

5 蚂蚁国际测试技术核心理念



TesterHome

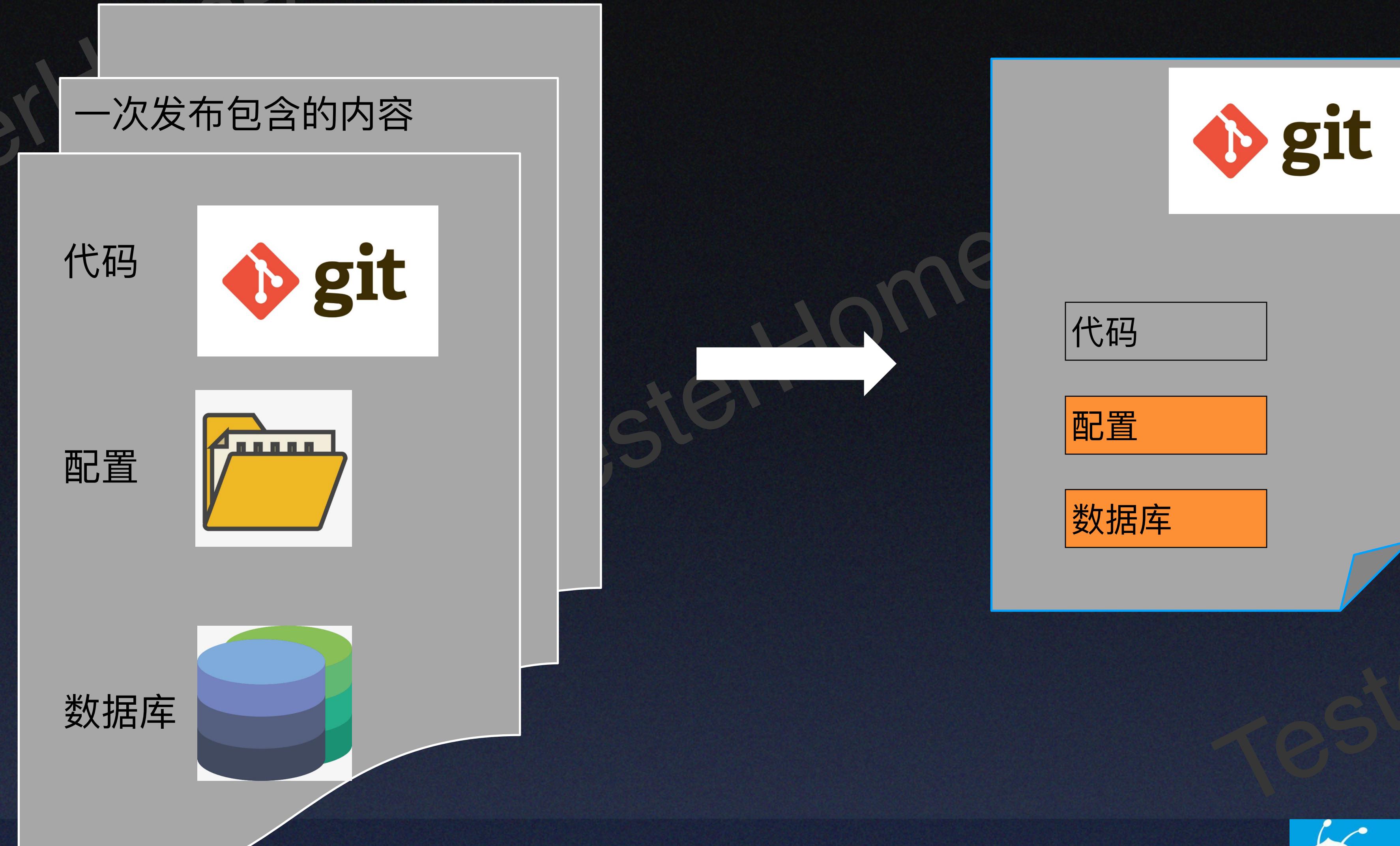
一键拉环境 = 配置代码化 + DB代码化



TesterHome

问题分析

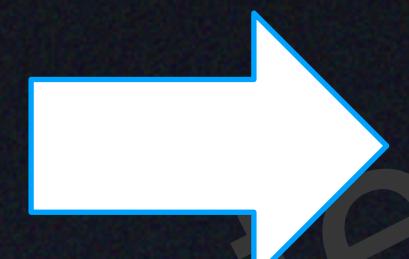




声明式设计

Declarative design (声明式设计) 是相对于 Imperative design (命令式设计) 而言，指的是这么一种软件设计理念和做法：我们向一个工具描述我们想要让一个事物达到的目标状态，由这个工具自己内部去 figure out 如何令这个事物达到目标状态。

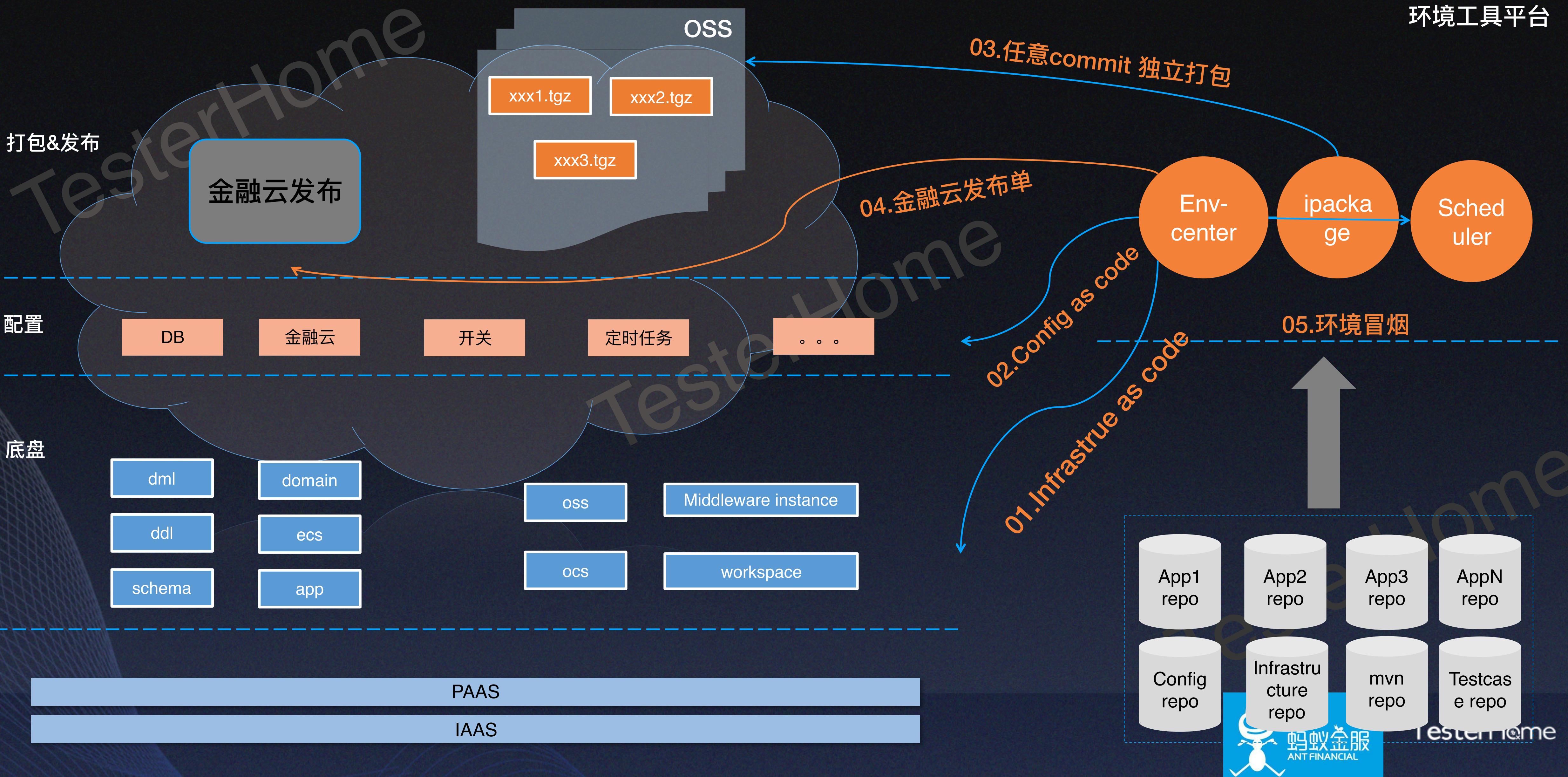
```
resource "aws_instance" "example" {  
    count = 10  
    ami   = "ami-v1"  
    instance_type = "t2.micro"  
}
```



```
resource "aws_instance" "example" {  
    count = 15  
    ami   = "ami-v1"  
    instance_type = "t2.micro"  
}
```

```
> terraform plan  
+ aws_instance.example.11  
  ami:                      "ami-v1"  
  instance_type:             "t2.micro"  
+ aws_instance.example.12  
  ami:                      "ami-v1"  
  instance_type:             "t2.micro"  
+ aws_instance.example.13  
  ami:                      "ami-v1"  
  instance_type:             "t2.micro"  
+ aws_instance.example.14  
  ami:                      "ami-v1"  
  instance_type:             "t2.micro"  
+ aws_instance.example.15  
  ami:                      "ami-v1"  
  instance_type:             "t2.micro"  
Plan: 5 to add, 0 to change, 0 to destroy.
```

一键拉环境解决方案

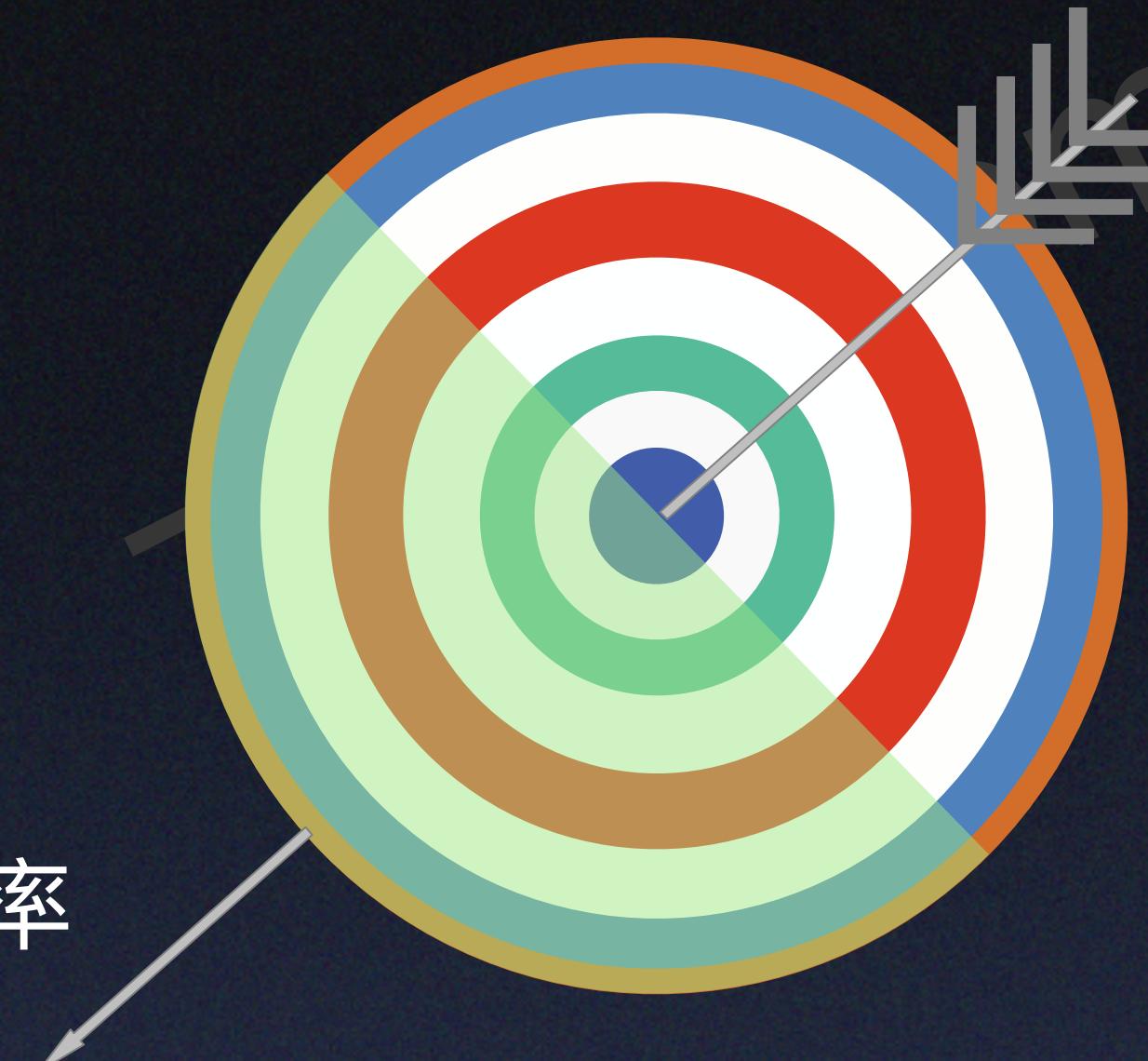


一键拉环境：“每日高频部署”目标达成

环境中心、打包平台

✓ 环境“所想即所得”
随时拉起所需环境

✓ 显著的提升测试效率



✓ 代码本身问题

✓ 金融云功能和稳定性问题

✓ 自动化测试用例问题

目录

CONTENTS

1 什么是Code Velocity

2 测试左移：代码门禁

3 持续测试怎么做

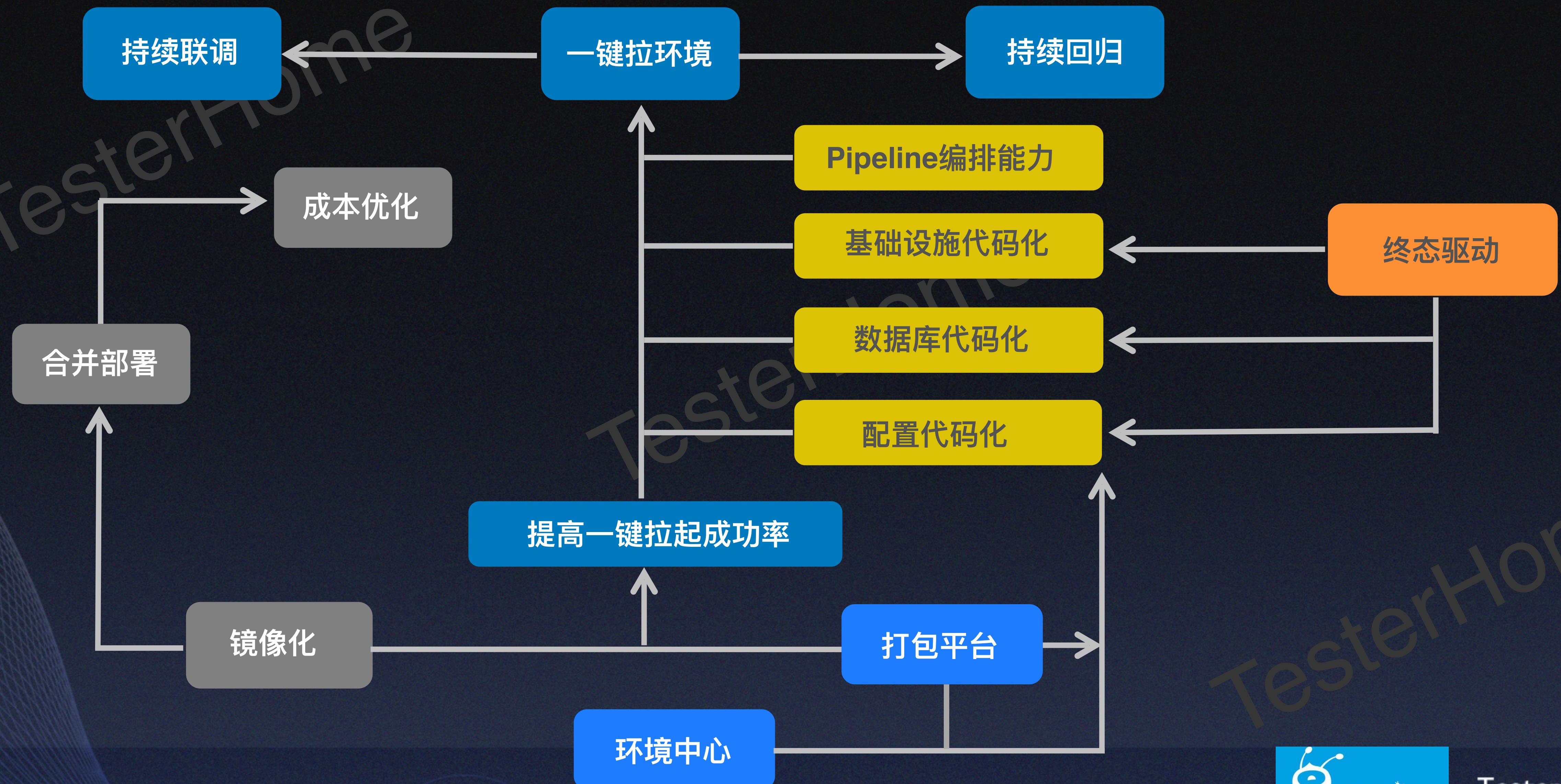
4 一键拉起全量隔离环境

5 蚂蚁国际测试技术核心理念



TesterHome

持续测试的核心思路



- ▶ 环境只在需要时拉起，用完即拆，释放资源
- ▶ 不浪费时间在环境问题排查上，只关注用例执行结果
- ▶ 拉起的环境稳定、无脏数据、全隔离无干扰
- ▶ 高频执行研发CI、主干回归用例，同时保证环境的稳定性

关键词：高频、持续



If It Hurts, Do It More Often

謝謝
THANKS

