GIAC | 全球互联网架构大会
GLOBAL INTERNET ARCHITECTURE CONFERENCE

持续集成技术实践

郭扬@fir.im

2017.thegiac.com

开发中的场景 – 效率问题

- 可读性问题
- 重复逻辑
- 重构？

```
/**
* Code Readability
*/
if (readable()) {
    be_happy();
} else {
    refactor();
}
```

开发中的场景 – 质量问题

开发了新功能，老功能产生新的 BUG

修好一个 BUG，又产生其他 BUG，甚至出现连环 BUG

出现的 BUG 比较多，修改代码要很谨慎，不熟悉的模块一般不敢动, 怕引起问题

开发中的场景 – 沟通问题

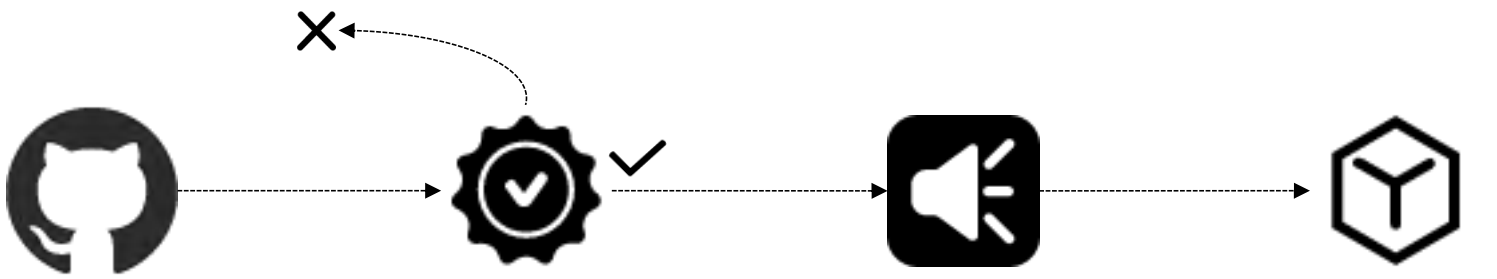
看板更新不及时

通知不及时

持续集成 (CI) 概览

- XP 的一个工程实践
- 在软件工程中，持续集成是频繁地（一天多次）将代码合并到主干的实践
- 通过自动构建的方式验证每次提交，让团队尽早的发现问题

CI 流水线



版本控制	自动化测试	消息通知	自动化部署
• Git • SVN • TFS • Clear Case	• 静态检查 • 单元测试 • 集成测试 • 系统测试 • 性能测试 • 压力测试	• 邮件通知	• 开发环境 • 测试环境 • 生产环境

持续集成 – 价值

- **尽早暴露问题，把握开发节奏**

  问题暴露的越早，修复代码的成本越低，成功部署的胜算就越大。持续集成高频率地编译、测试、审查、部署项目代码，这其中代码集成是主要的风险来源。要想规避这个风险，只有提早集成，持续而有规律的集成，以此来确保当前代码库的质量，把握开发的进程和节奏。
  当然发现问题代码，也不要一味地坠入快速的简单修复之中，要投入时间和精力保持代码的整洁、敞亮。
  很明显的一点，使用持续集成后，程序员们提交代码也会变得更加小心谨慎。想想应该没人乐意让其他同事不停地见到自己的分支上 CI 失败的通知邮件吧

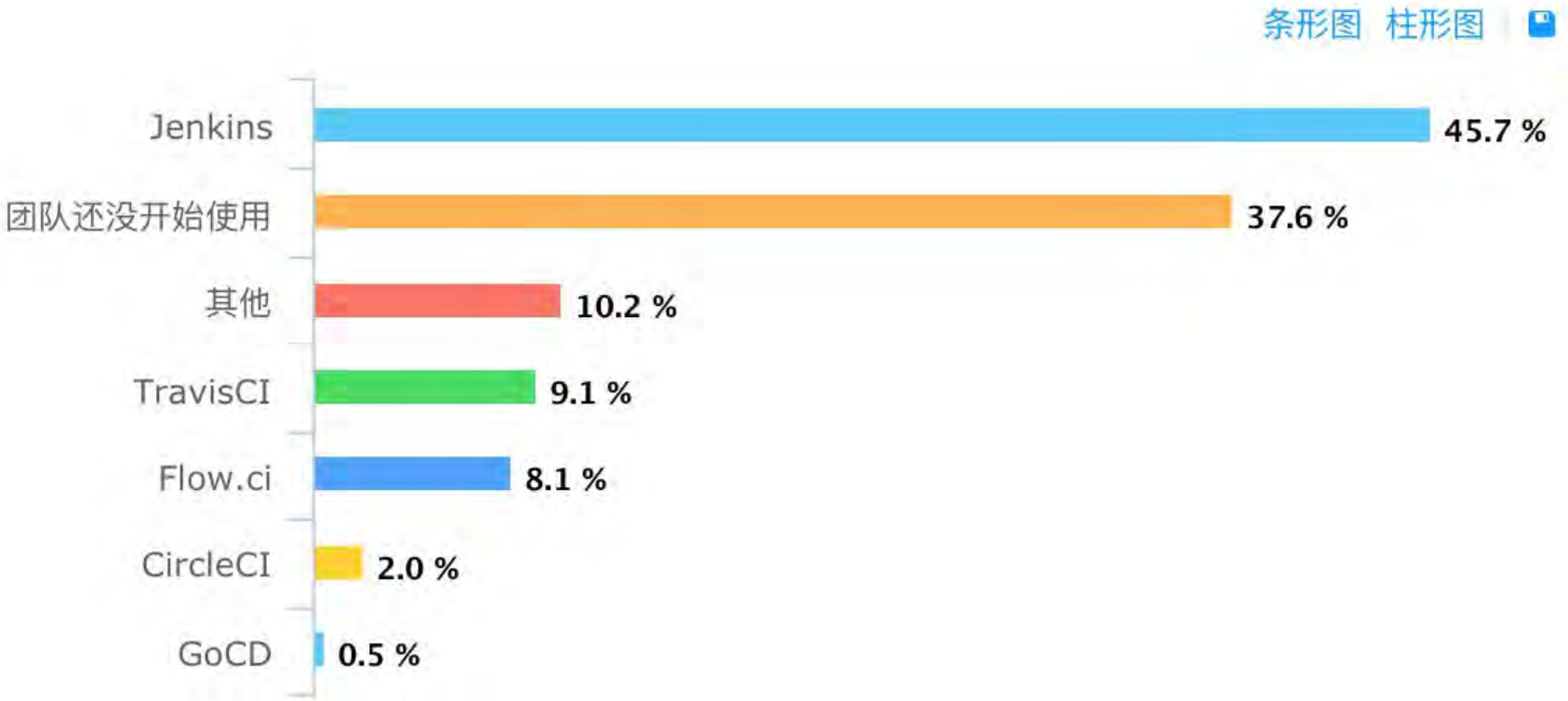
- **保持随时部署，简化发布流程**

  每日高频率的集成保证了项目随时处于可部署运行的状态，如果没有持续集成，项目发布之前将不得不手动地集成，然后花费大量精力修复集成问题，弄的团队成员疲惫不堪。

- **增强团队信心，建立工程师文化**

  无论什么样的工程师，都会对存在大量 bug 的代码产生恐惧心理，这就是心理学上的的 Broken Windows 综合症（Broken Windows Syndrome）。CI 可以有效防止破窗综合症，让开发团队一点点积累起对产品的信心，对使用技术的保持成就感。

持续集成 – 现状

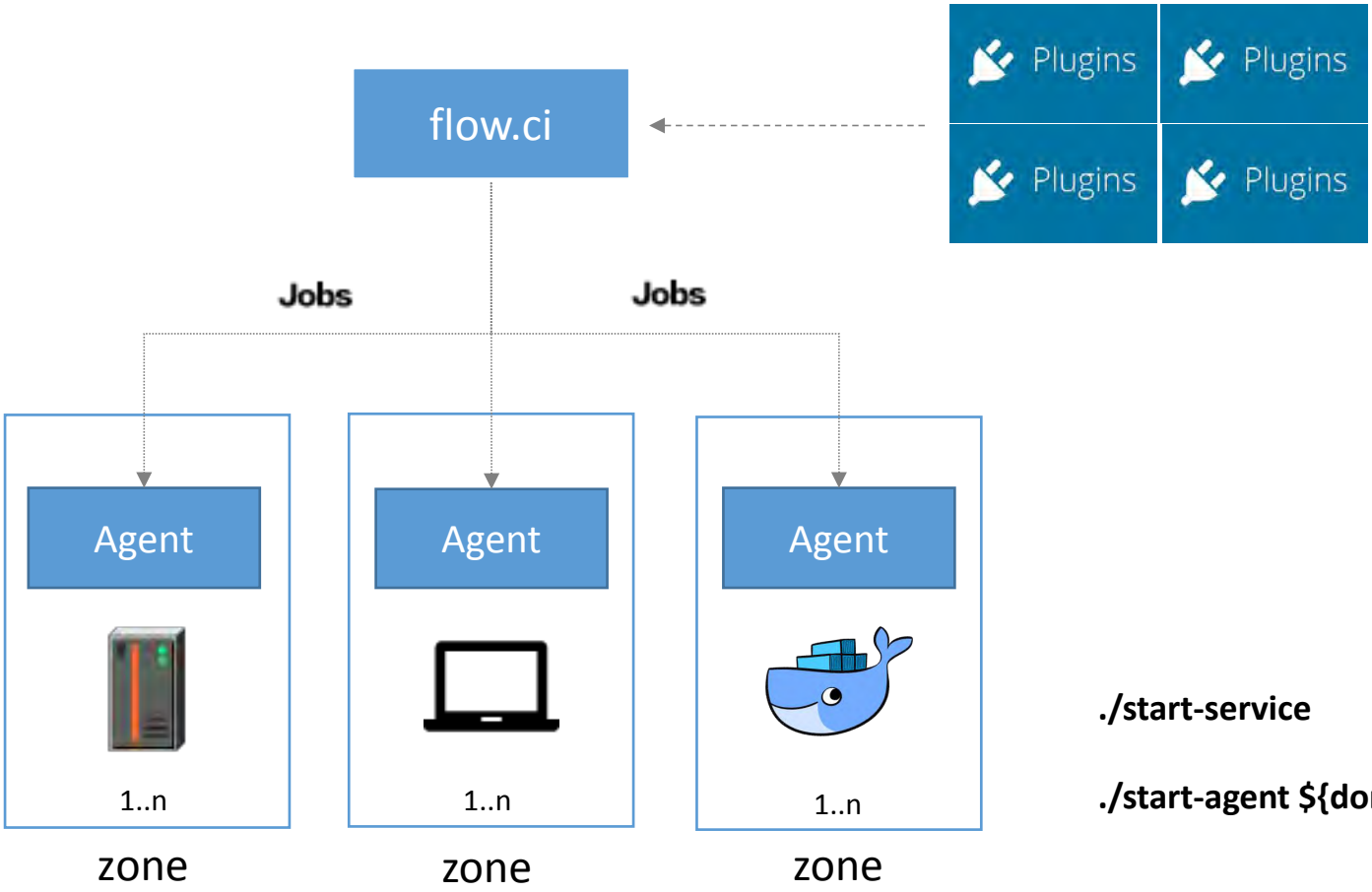

条形图
柱形图
Jenkins
45.7 %
团队还没开始使用
37.6 %
其他
10.2 %
TravisCI
9.1 %
Flow.ci
8.1 %
CircleCI
2.0 %
GoCD
0.5 %

持续集成 – 阻力

- CI 太复杂了：流程复杂，环境配置复杂
- 意识上的阻力
- 测试失败，久而久之就放弃了

flow.ci

flow.ci



./start-service

./start-agent ${domain} ${token}

flow.ci

YAML

GUI

```yaml
# flow.ci templates

flow:
  - envs:
      FLOW_WELCOME_MESSAGE: "hello.world"

  steps:
    - name: Print
      condition: |
        true
      script: |
        echo ${FLOW_WELCOME_MESSAGE}

    - name: End
      script: |
        echo "End"
```

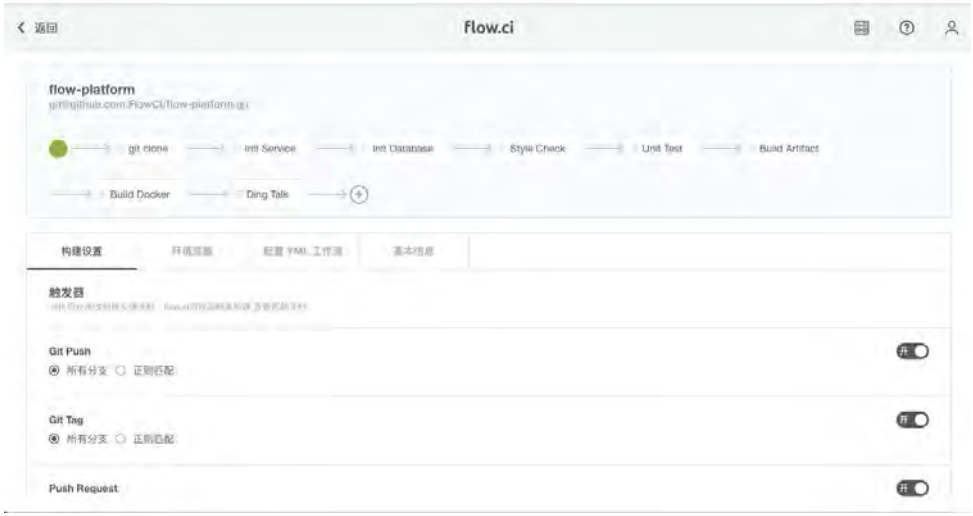
环境变量定义

步骤名称

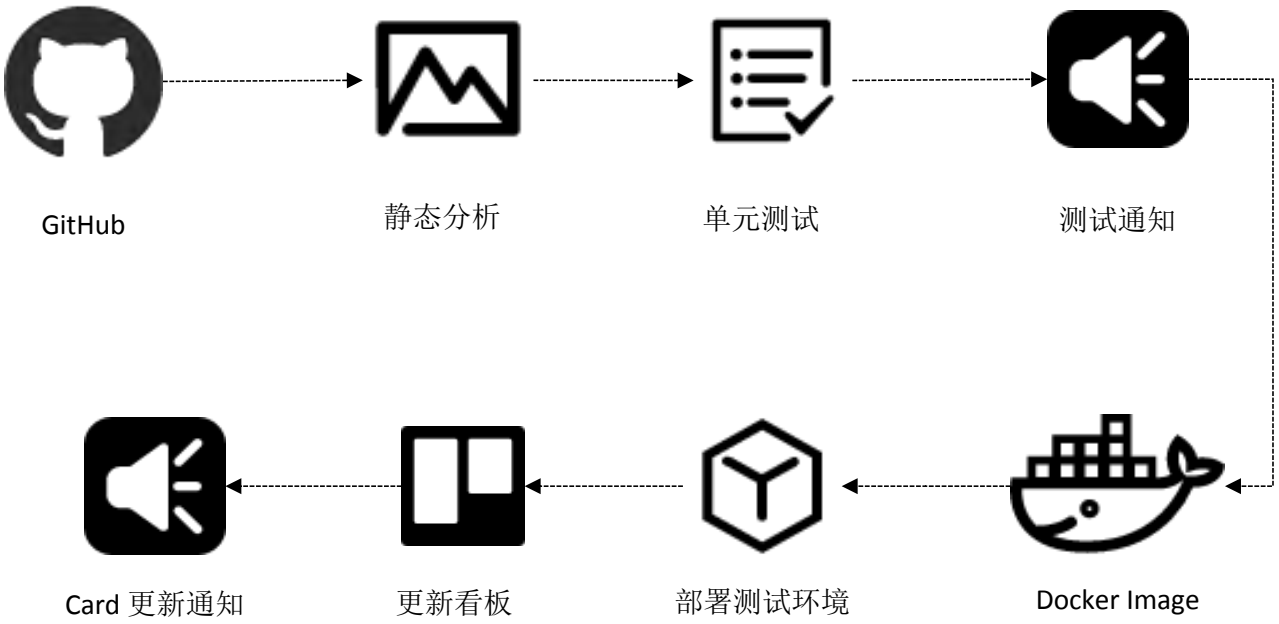
运行条件

运行的 Shell 脚本

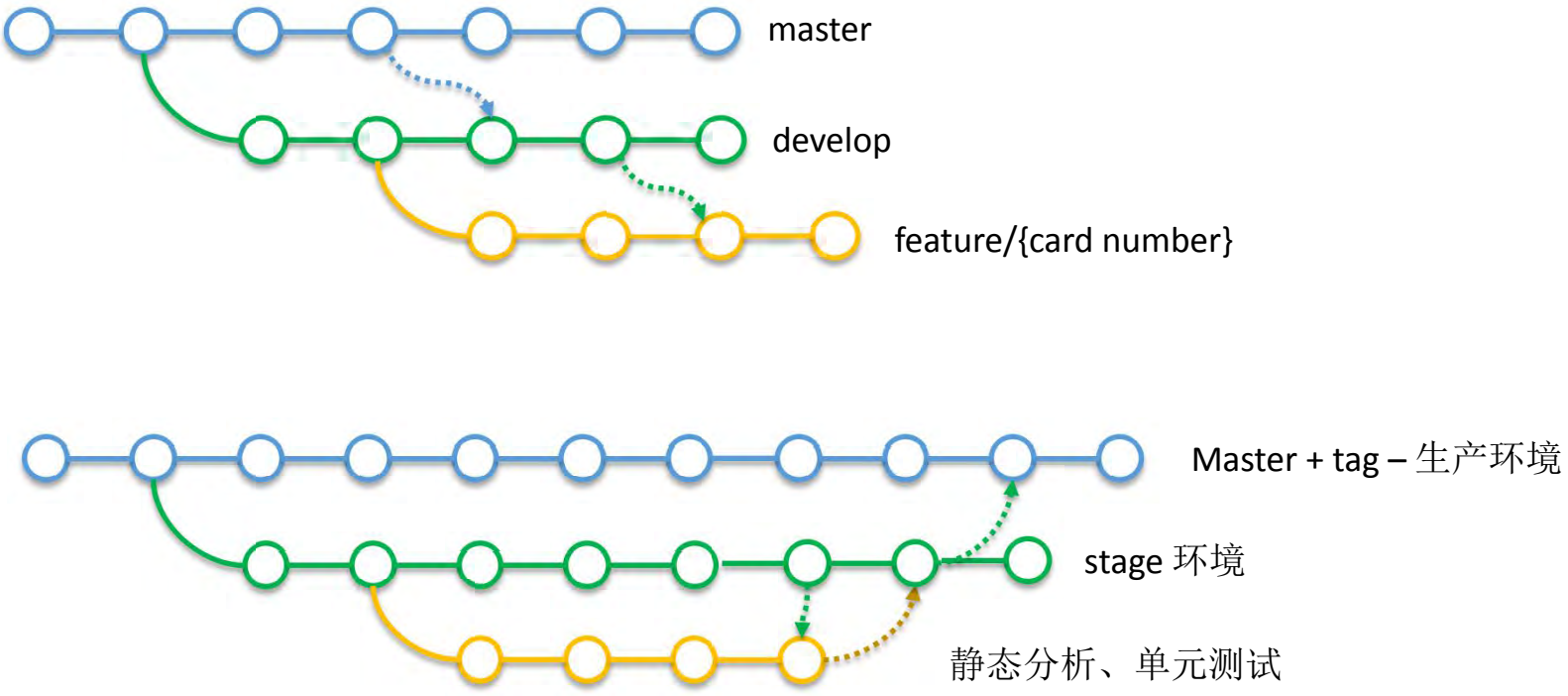
工作流定义



flow.ci 流水线



不同分支，不同构建策略


master
develop
feature/{card number}
Master + tag – 生产环境
stage 环境
静态分析、单元测试

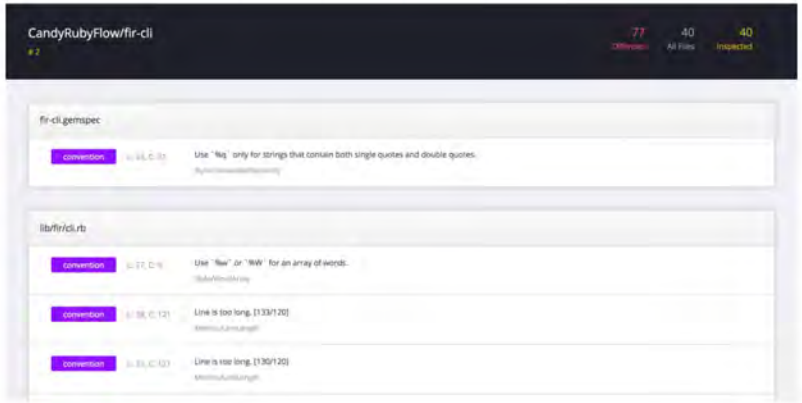
从静态分析开始

代码质量

潜在 Bug

安全检查

Report →



- Check Style
- Error Prone – Bug Pattern
  com/google/errorprone/bugpatterns

单元测试

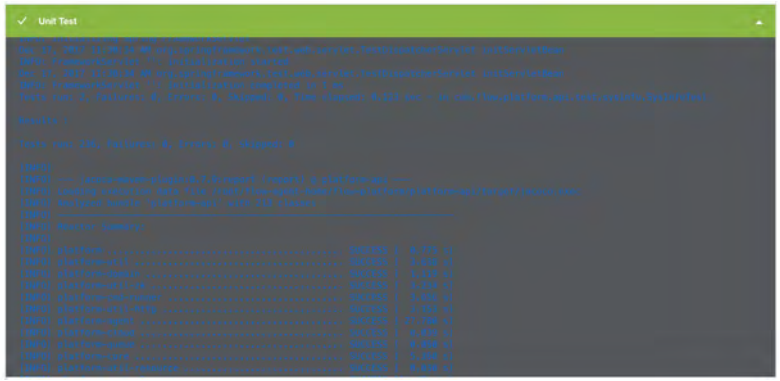
测试不只是测试人员的事，所有人都要对质量负责

测试金字塔的“核心测试”

持续回归测试，第一时间发现代码变更问题

单元测试

- 测试结果



- 代码覆 盖率信息

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
com.flow.platform.api.service		64%		48%	101	211	234	640	22	105	1	16
com.flow.platform.api.controller		51%		36%	113	193	232	455	57	123	0	19
com.flow.platform.api.domain.user		51%		11%	82	174	80	259	29	120	0	11
com.flow.platform.api.service.job		80%		76%	51	194	113	644	4	83	1	12
com.flow.platform.api.consumer		18%		17%	25	36	77	100	17	27	4	8
com.flow.platform.api.util		80%		67%	62	161	64	384	18	71	1	14
com.flow.platform.api.domain		50%		11%	49	109	54	167	24	83	0	9
com.flow.platform.api.service.user		73%		66%	43	110	62	262	8	54	0	5
com.flow.platform.api.service.node		78%		70%	27	105	65	329	4	51	0	4
com.flow.platform.api.domain.job		84%		47%	35	160	29	267	12	126	0	10

单元测试

Thread.sleep(xxxx)

ApplicationEvent + CountDownLatch

```
final CountDownLatch countDown = new CountDownLatch(num);
springContext.registerApplicationListener(new JobStatusEventConsumer() {
    @Override
    public void onApplicationEvent(JobStatusChangeEvent event) {
        if (event.getJob().equals(job) && event.getTo() == expect) {
            countDown.countDown();
        }
    }
});
return countDown;
```

CountDownLatch latch = new CountDownLatch(1)
latch.await(30, TimeUnit.*SECONDS*)

单元测试

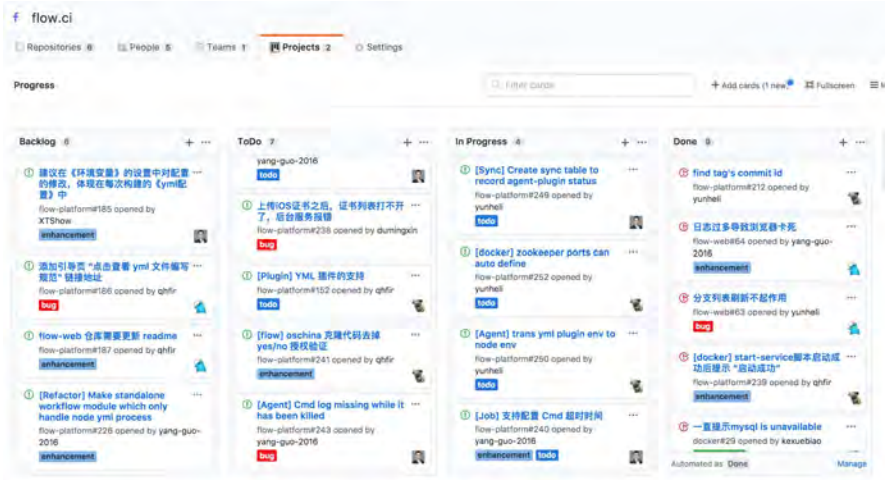
数据清理

- @BeforeClass
- @Before
- @After
- @AfterClass

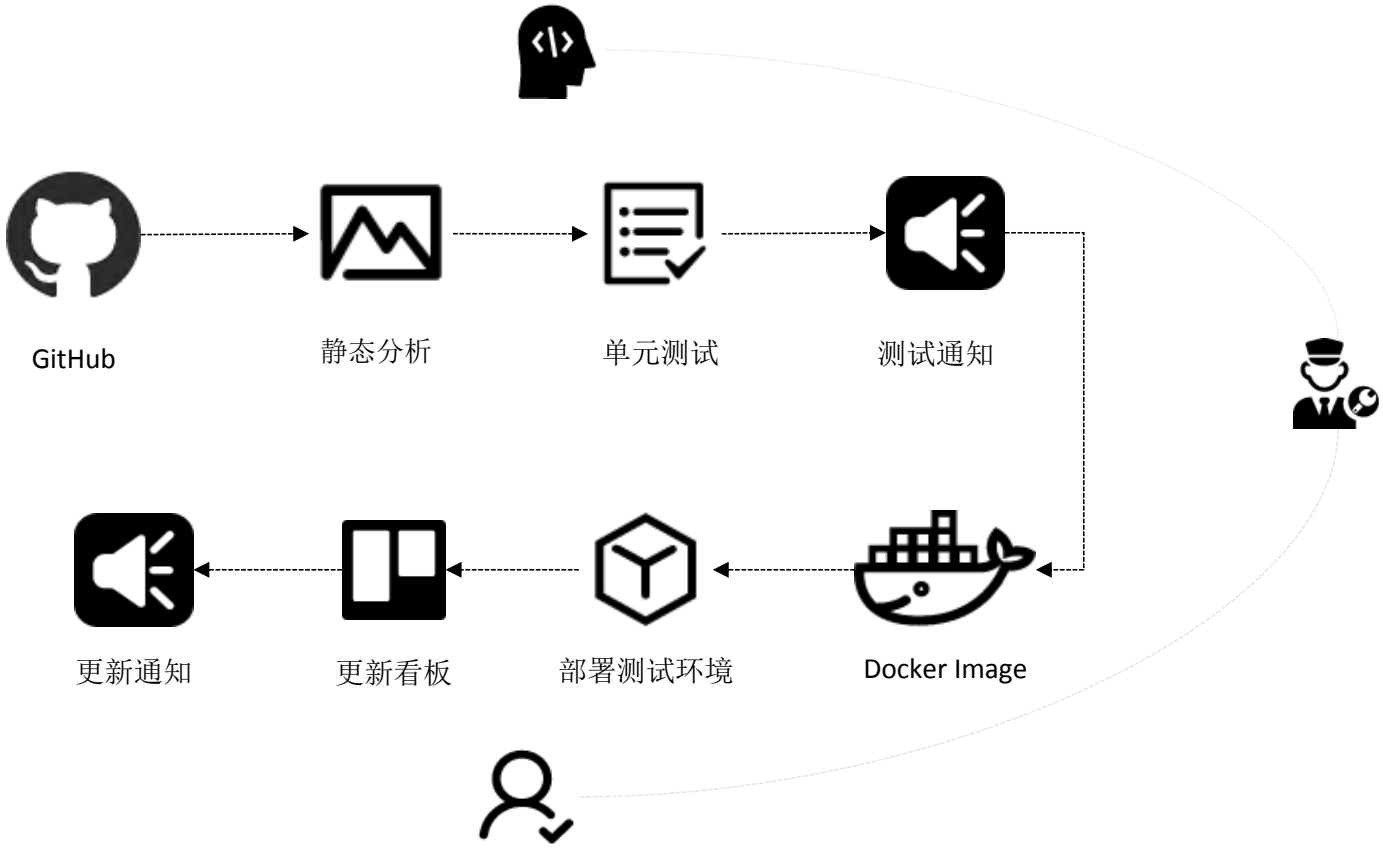
Mock

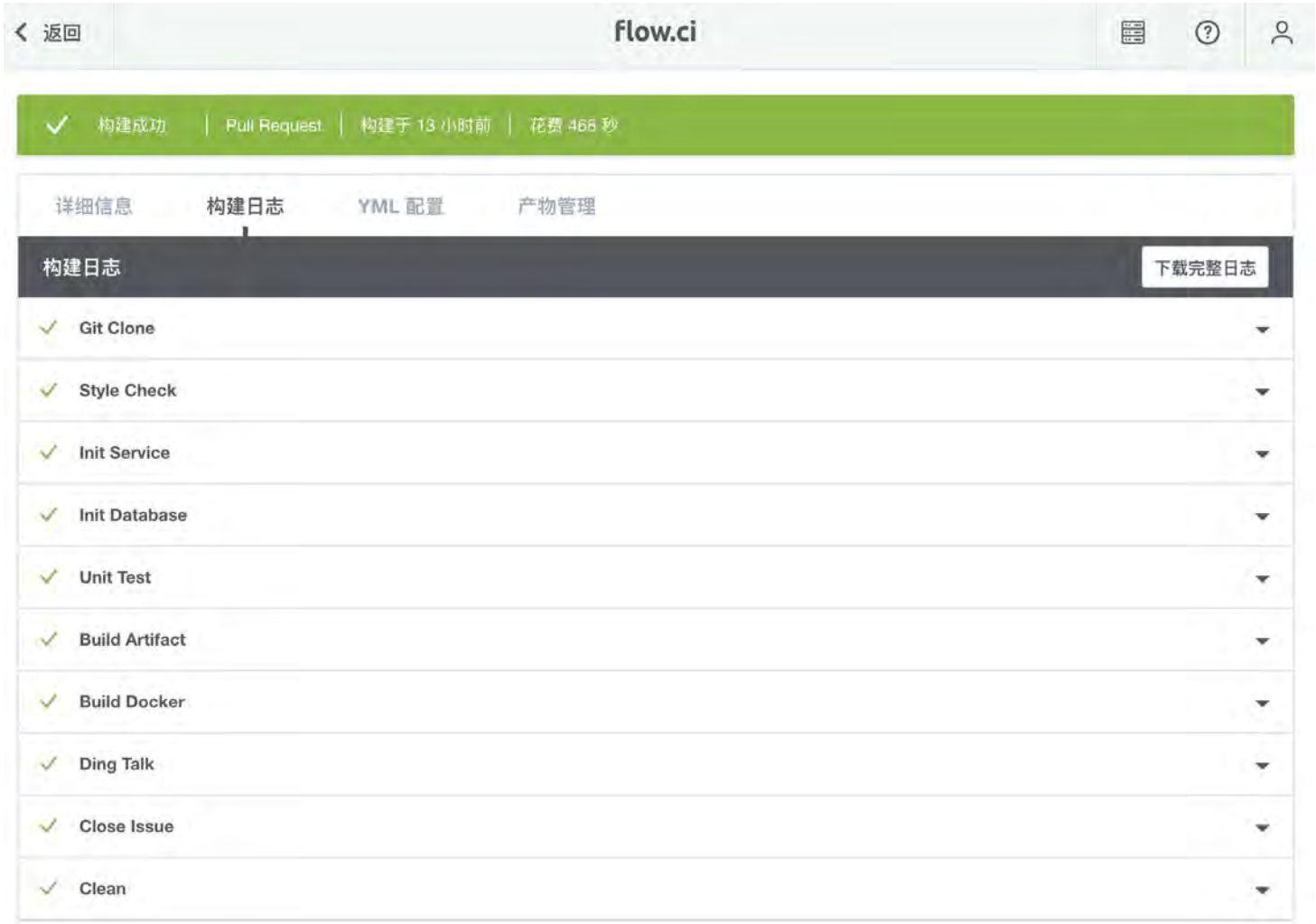
```
@Bean
@Primary
public PluginService pluginService() {
   return Mockito.mock(PluginService.class);
}

…

Mockito.when(pluginService.list(ImmutableSet.of(PluginStatus.INSTALLED)))
   .thenReturn(ImmutableList.of(plugin));
```

及时更新，让产品和开发同步



及时通知，让产品和开发同步

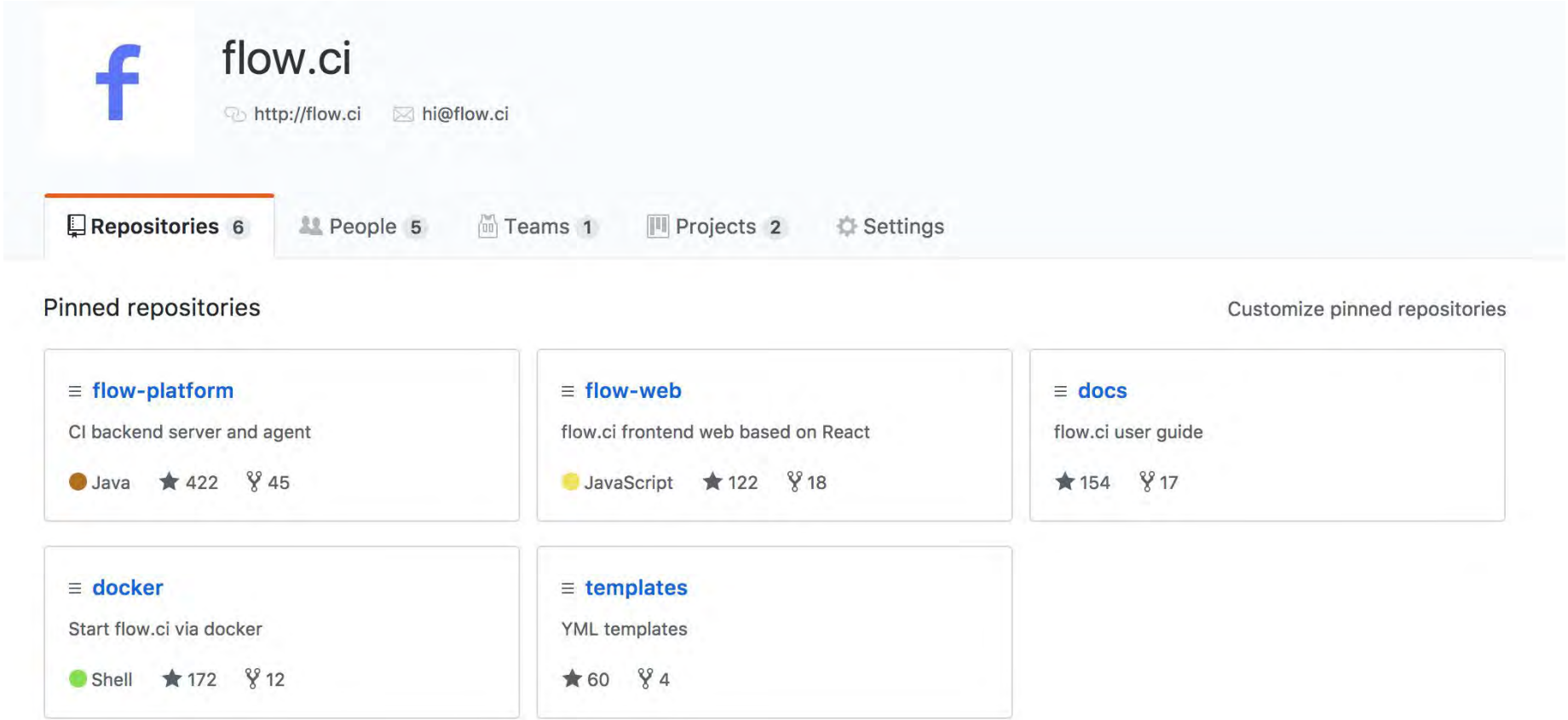



返回
Flow.ci
构建成功
Pull Request
构建于 13 小时前
花费 468 秒
详细信息
构建日志
YML 配置
产物管理
构建日志
下载完整日志
Git Clone
Style Check
Init Service
Init Database
Unit Test
Build Artifact
Build Docker
Ding Talk
Close Issue
Clean

持续集成是一种文化

- 频繁集成
- 集成结果可视化
- 修复失败的集成是最高优先级的任务

敏捷的基石



https://github.com/flowci

只有最适合的，才是最好的持续集成方案


Q
&
A

GIAC | 全球互联网架构大会
GLOBAL INTERNET ARCHITECTURE CONFERENCE

GIAC

全球互联网架构大会

GLOBAL INTERNET ARCHITECTURE CONFERENCE

扫码关注GIAC公众号

2017.thegiac.com