ZTE中兴

未 来，不 等 待

大型项目CI的服务化解决方案

张佑文

2019年12月

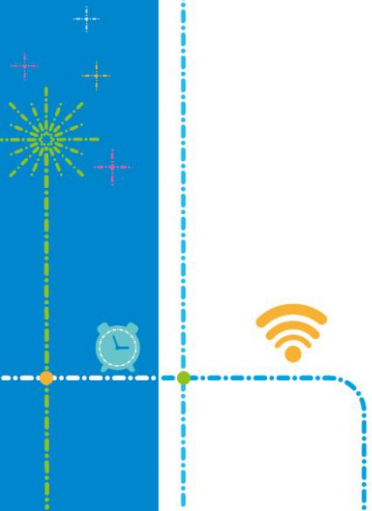
IAS 2019

提 纲

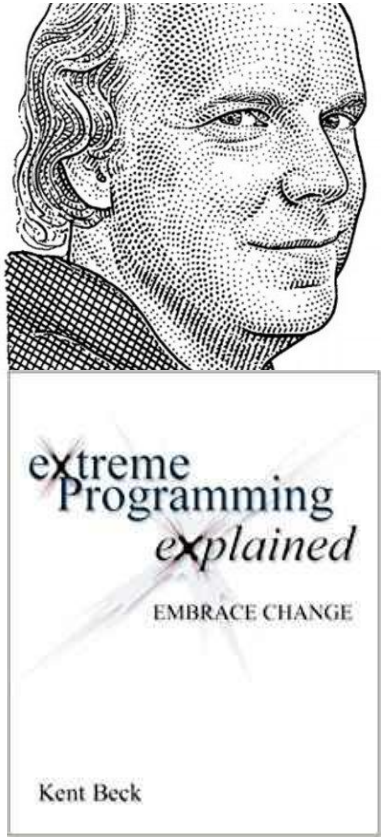
CI与痛点

解决方案

实践成效



Why持续集成



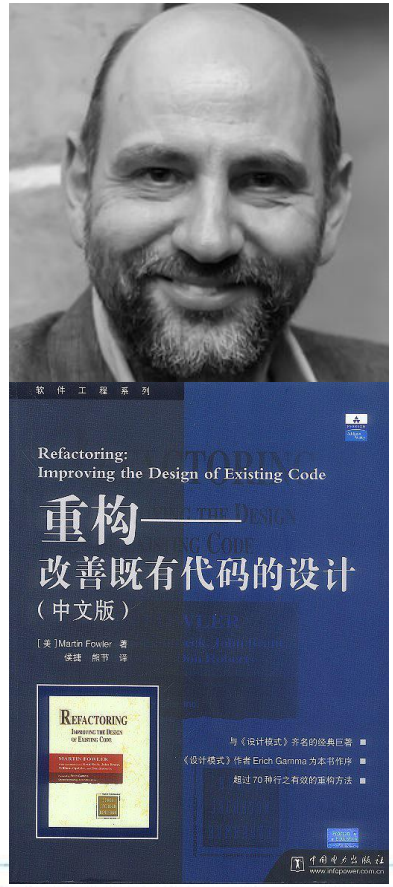
爆炸式集成

- Bug 总是在最后才发现
- 越到后期问题越难解决
- 软件交付时机无法保障
- 程序经常需要变更
- 无效的等待变多
- 用户的满足度低

持续集成

- 降低风险及早发现问题
- 更快地定位和修复问题
- 更快地交付成果
- 减少手工的错误
- 减少了等待时间
- 更高的产品质量

how持续集成

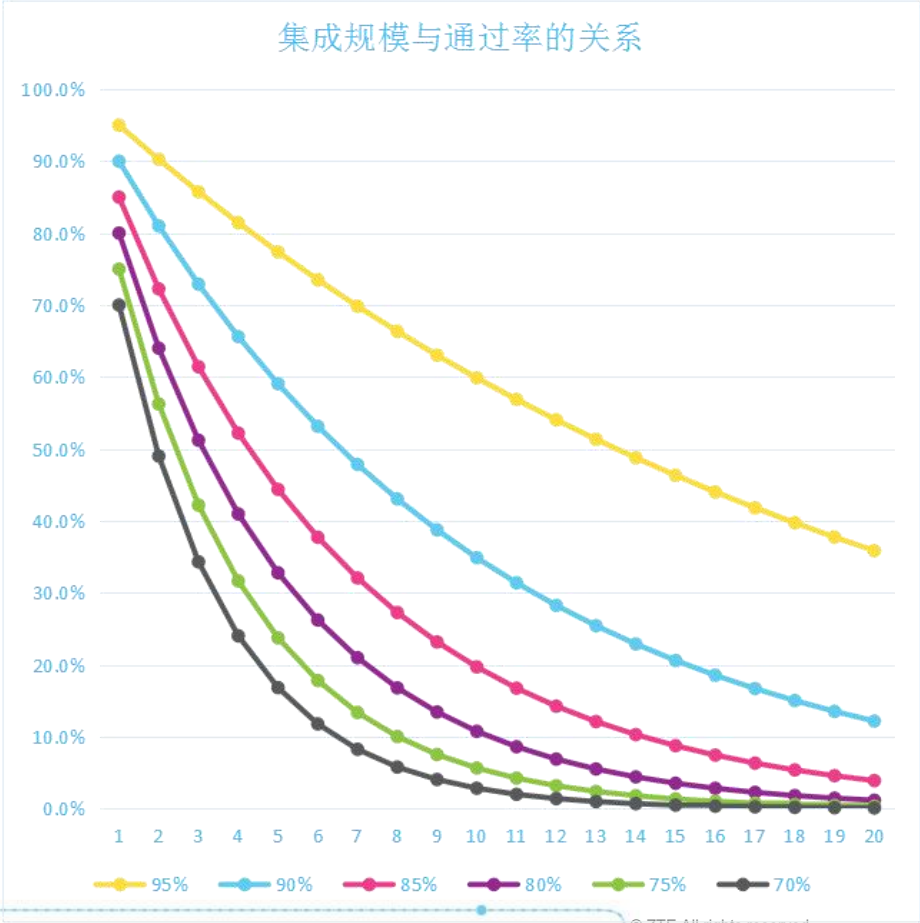


1. 维护统一的源码存储库
2. 自动化构建
3. 构建包含自动化测试验证
4. 每人每天提交代码到主干分支
5. 每次提交都应在主干分支上构建集成
6. 立即修复失败的构建
7. 让构建更快速
8. 在类生产环境中测试
9. 任何人都能轻松获得最新可用版本
10. 每个人都可以看到主干分支构建状态
11. 自动部署多种环境

大规模项目特点

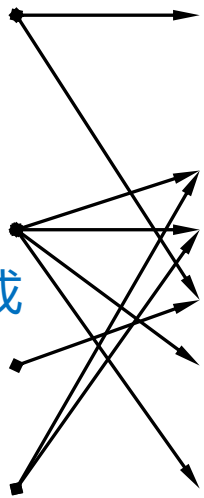
- **产品不确定性多**
  - 输入不确定：标准协议未定稿，客户认知不一
  - 基础不稳定：全新平台、系统架构、全新通讯协议
  - 外部约束多：兼容多种组网、多个规格型号、多个历史版本、安全合规
- **系统复杂程度高**
  - 软件规模大：代码千万~亿行，版本包2G~60G
  - 交付要求高：1个月发布一个版本
- **团队规模大**
  - 并行开发：1000~3000人开发交付
  - 跨地域协同：深圳、上海、西安、南京多地协同
  - 能力参差不齐：新员工多，人员流动性大

大规模项目集成规模与成功率



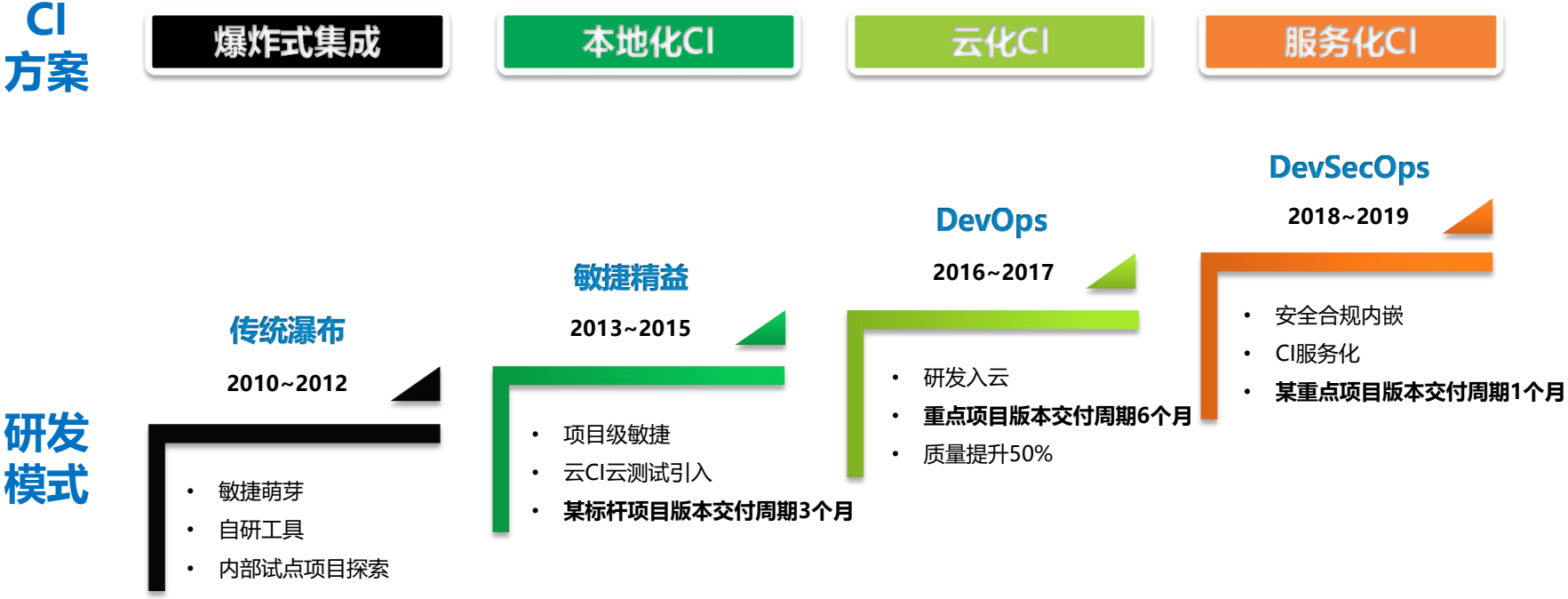
大规模项目CI的挑战与痛点

- 源码/工具/依赖库分散在不同位置
  - 版本分支管理复杂
  - 依赖关系复杂，难以并发构建
- 构建集成时长过长
  - 无法每次入库在主干分支构建集成
- 构建失败定位困难
  - 批量合入，人工定位回滚
- 主干锁库频繁，阻塞代码提交
  - 等待入库，等待测试，开发效率降低



1. 维护统一的源码存储库
2. 自动化构建
3. 构建包含自动化测试验证
4. 每人每天提交代码到主干分支
5. 每次提交都应在主干分支上构建集成
6. 立即修复失败的构建
7. 让构建更快速
8. 在类生产环境中测试
9. 任何人都能轻松获得最新可用版本
10. 每个人都可以看到主干分支构建状态
11. 自动部署多种环境

我们的CI演变历程

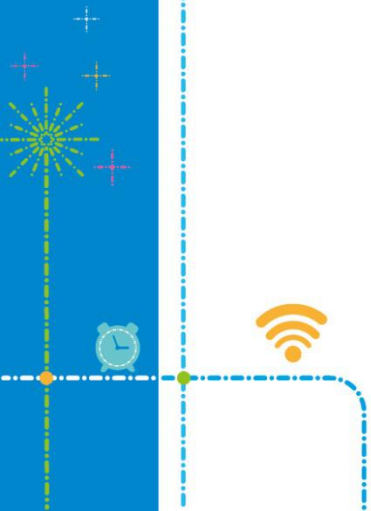


提 纲

CI与痛点

解决方案

实践成效



云化CI方案存在的痛点

痛点

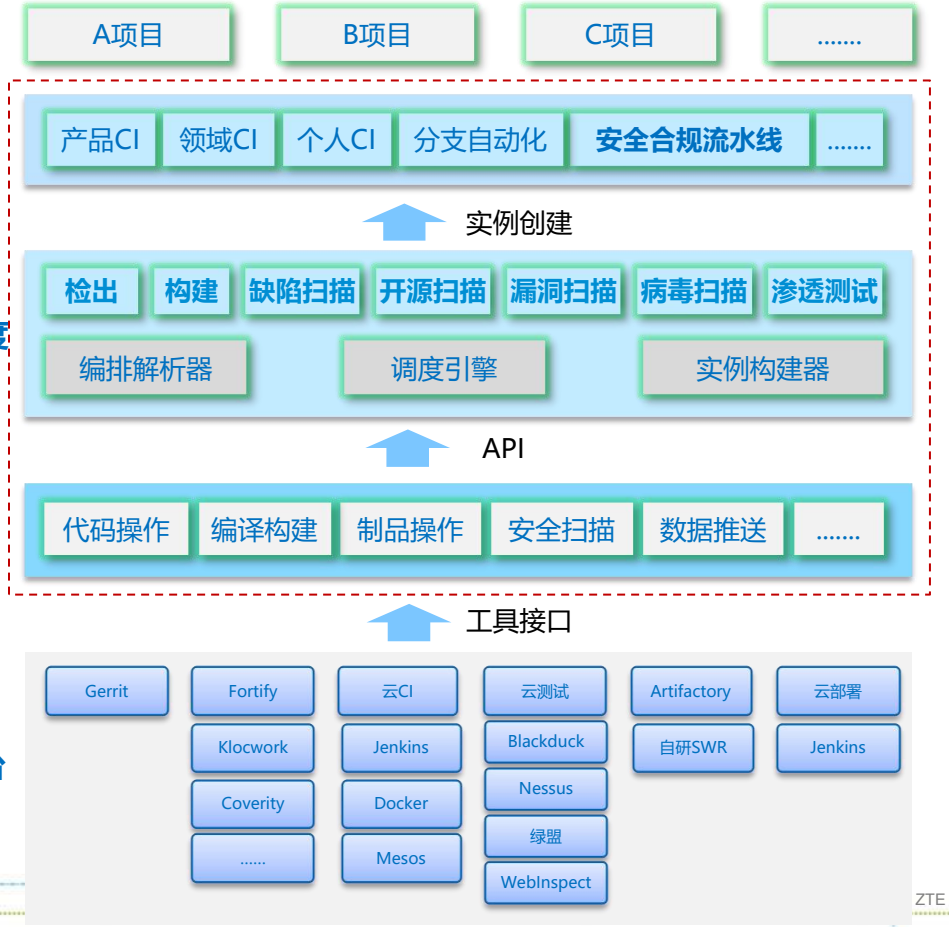
- 项目上线：CI需要DIY，低水平重复
- 安全合规：工具分散，游离在主流程外
- 数据一致性：项目间数据定义差异明显，互通困难
- 成果共享复制：优秀成果复制推广困难

服务化CI的演进思路

目标

- 场景化支撑：可编排支撑多种项目场景，轻量化使用运维
- 服务内嵌：安全合规内嵌、质量内建
- 数据治理：统一业务模型、数据模型，全局展示
- 开源共创：聚焦主航道，工具可插拔和持续演进

服务化CI业务框架：场景化、可编排、可扩展



多样化场景

多级CI/多分支/多项目协同流水
微服务/小产品的多分支管理
多代码库一键拉分支

灵活编排

- 多层次串行、并行编排
- 资源优先级可调配
- 控制点、红线自动管控

开源共创

- 插件式扩展
- 能力集可扩展，**快速复制优秀实践**

数据贯通

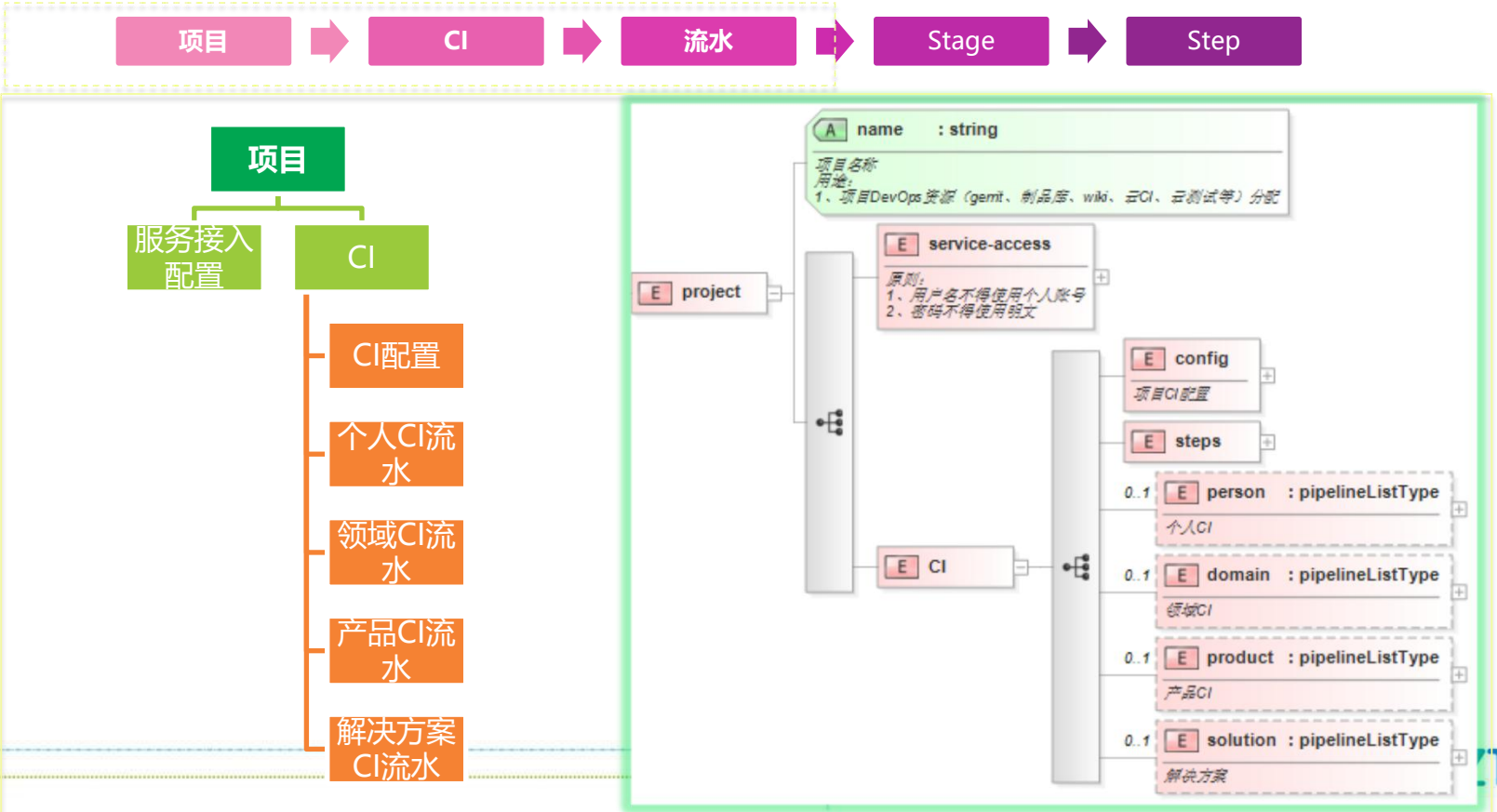
- **统一的数据模型**
- 自动化上报数据中台
- 对通透视图

低成本

- 原生工具封装，简化引用
- 工具部署升级可统一实施

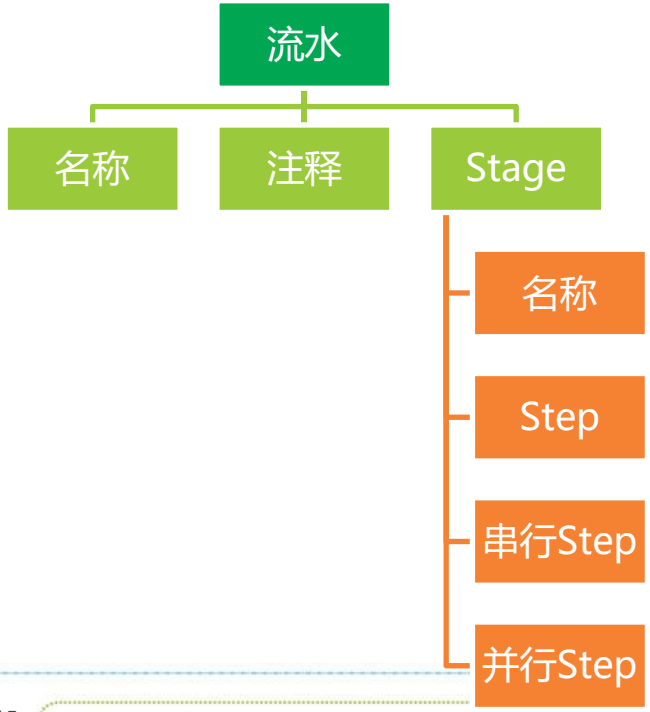
CI服务化方案：基于项目业务提炼通用模型，基于配置灵活调度执行

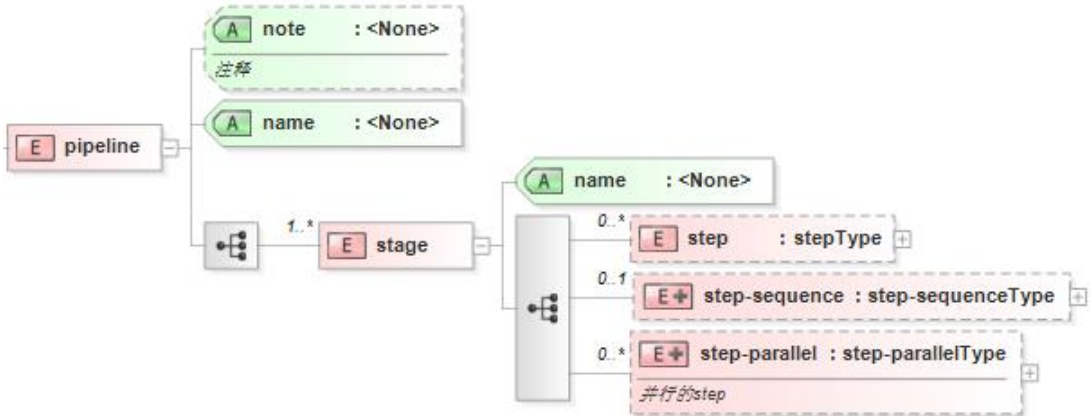
CI模型提炼


项目
CI
流水
Stage
Step
项目
服务接入配置
CI
CI配置
个人CI流水
领域CI流水
产品CI流水
解决方案CI流水
A name : string
项目名称
用途：
1、项目DevOps资源（gemt、制品库、wiki、云CI、云测试等）分配
E project
E service-access
原则：
1、用户名不得使用个人账号
2、密码不得使用明文
E CI
E config
项目CI配置
E steps
0..1 E person : pipelineListType
个人CI
0..1 E domain : pipelineListType
领域CI
0..1 E product : pipelineListType
产品CI
0..1 E solution : pipelineListType
解决方案

CI模型提炼

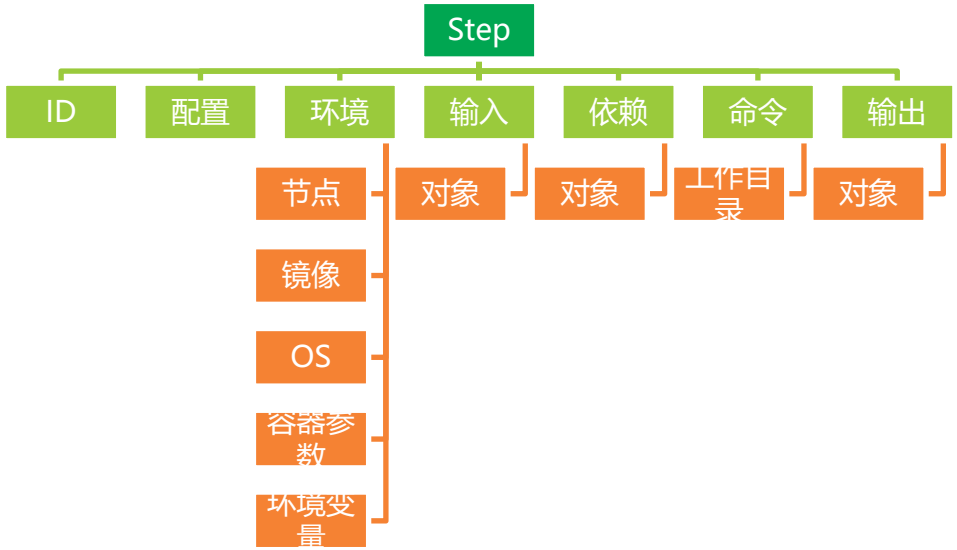
项目 → CI → 流水 → Stage → Step

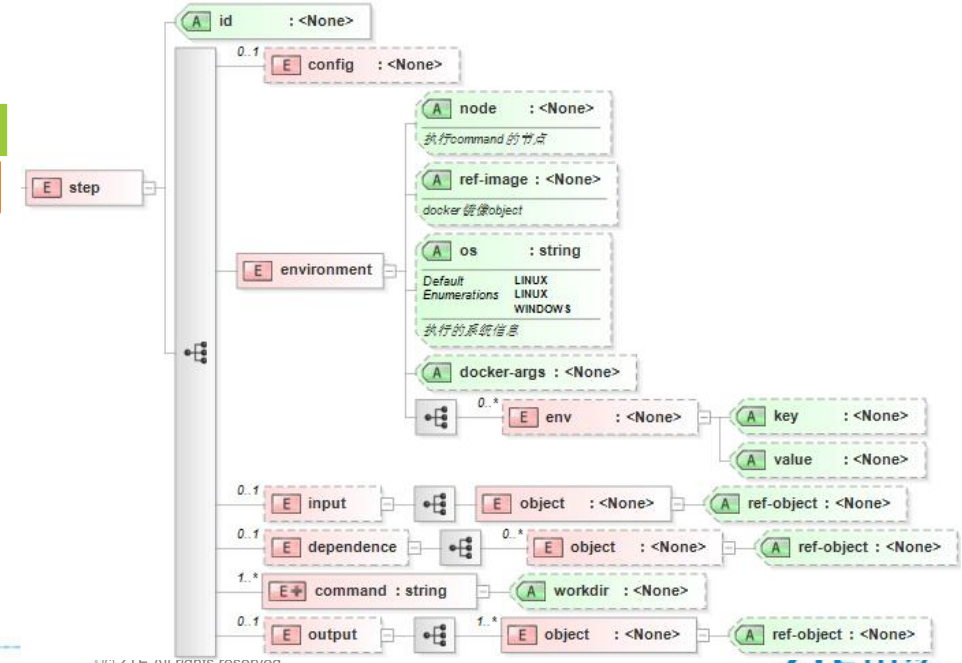




CI模型提炼

项目 → CI → 流水 → Stage → Step



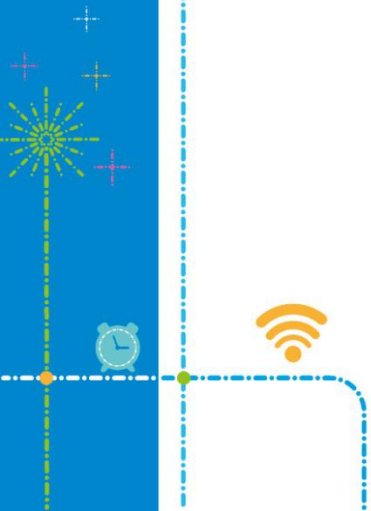


提 纲

CI与痛点

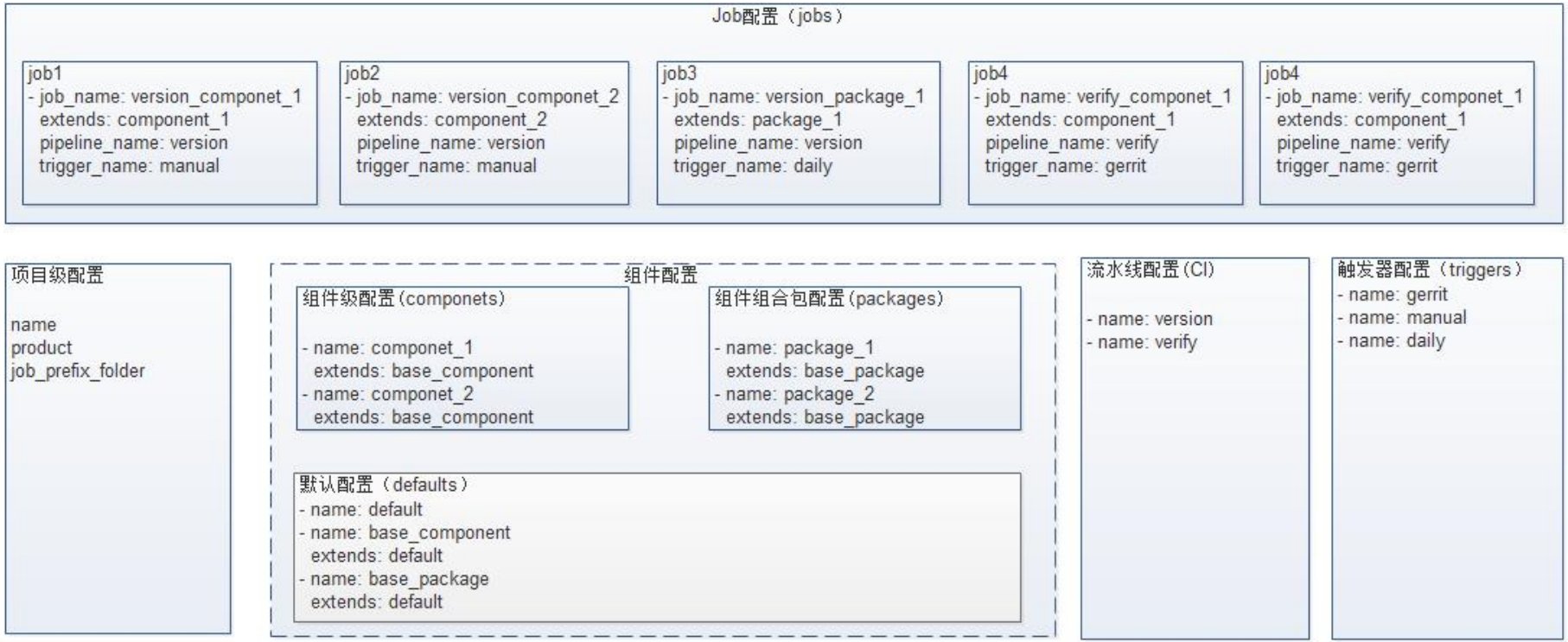
解决方案

实践成效



配置运维轻量化

1. 项目调试上线：新手2周，熟手1周
2. 灵活配置扩展：yaml配置实现新引入工具调用



Job灵活创建

1. Job灵活创建：支持单Job单分支，单Job多分支；单流水单Job，单流水多Job
2. Job间的触发：支持大项目项目群Job之间嵌套自动化调度

```
238 jobs: # ${prefix}/${project_name}/${pipeline}/${base_name}_${branch}
239   - extends_list: [dis, dts, ms_modb, uas, dpf]
240     job_name: ${pipeline_name}_${name}
241     job_folder: ${job_prefix_folder}/${project_name}/${pipeline_name}
242     pipeline_name: version
243     trigger_name: manual # 被触发
244     params:
245       - name: BRANCH_NAME
246         default: master # 默认参数
247         description: "构建分支"
```

```
249   # TODO 自动多分支任务的创建
250   - job_name: dpf_auto_at_night
251     extends: dpf
252     job_folder: ${job_prefix_folder}/${project_name}/${pipeline_name}
253     pipeline_name: version
254     trigger_name: daily_night # 被触发
255     params:
256       - name: BRANCH_NAME
257         default: master # 默认参数
258         description: "构建分支"
```

资源调度提效

1. 多层次节点并发：支持stage内多模块并发/多step并发/step内操作并发，满足各种场景的CI快速反馈需求
2. 资源优先级可调配：支持按照分支/组件/节点方式进行资源的自动化匹配，保障高优先级分支的Job资源要求

```
230     - stage-parallel: sub_components
231       node: "windows"
232       stages:
233         - stage: codes
234           caption: "检出代码"
235           steps:
236             - step-parallel: checkout
237         - stage: prepare
238           caption: "清理&准备"
239           steps:
240             - step: prepare
241             - step-parallel: build_components
242             - step-parallel: download_artifacts
```

```
# phy 环境使用的window节点
- name: windows
  label: windows # 默认规则；
  workspace_root: E:/winshare # 没有配置时使用节点的默认值
  label_filter: # 配置优先从分支匹配选择相应label来运行step
    # 以下是示例，配置默认的Label: windows; 可以根据项目实际情况进行改配
    - branch: master # 正则
      label: windows
    - branch: release/.*
      label: windows
    - branch: feature/.*
      label: windows
```

分支自动化管理

1. 一键拉分支：支持多代码库一键拉分支；可以通过分支配置信息创建/代码库的快照信息创建
2. 支持复杂多分支：支持项目微服务/小产品独立发布的分支自动化管理

构建参数

NEW_BRANCH	sjx_0619
	要创建的新分支名。配置该参数时，DELETE_BRANCH配置无效。 NEW_BRANCH和DELETE_BRANCH都没有配置时，显示加列表。
DELETE_BRANCH	
	要删除的分支名。配置了NEW_BRANCH时，DELETE_BRANCH配置无效。 NEW_BRANCH和DELETE_BRANCH都没有配置时，显示加列表。
BRANCH_NAME	master
	工作分支，运行时使用该分支的配置。 在没有配置COMMIT_FILE时，也从该分支的配置文件中获取库列表。 获取的库列表中，有分支配置库，不会进行操作。
COMMITS_FILE	
	制品库上用于创建分支的文件路径，从repo开始相对路径,可以从制品库中直接拷贝。 配置时，使用commits_file中的信息创建新分支。对于有配置分支的库，不会进行操作。 为空时，使用BRANC_NAME分支中的配置信息来获取库列表。 示例：g5_nr_v2-snapshot-generic/aurora_test/NF_RPF/master/RPF_v2.00.23.00B54-1_20190615150943/commits_rpf.json

自动化上报透视数据

1. 内置数据上报：自动化通过Rest或Jenkins插件上报CI数据
2. 增强CI数据模型：支持CI数据建模全量表的数据模型，可扩展补充step内细节数据

- ci_job_build
- ci_pipeline_stages
- ci_kw
- ci_coverity
- ci_sourcemonitor
- ci_lizard
- ci_flake8
- ci_pylint
- ci_gcov

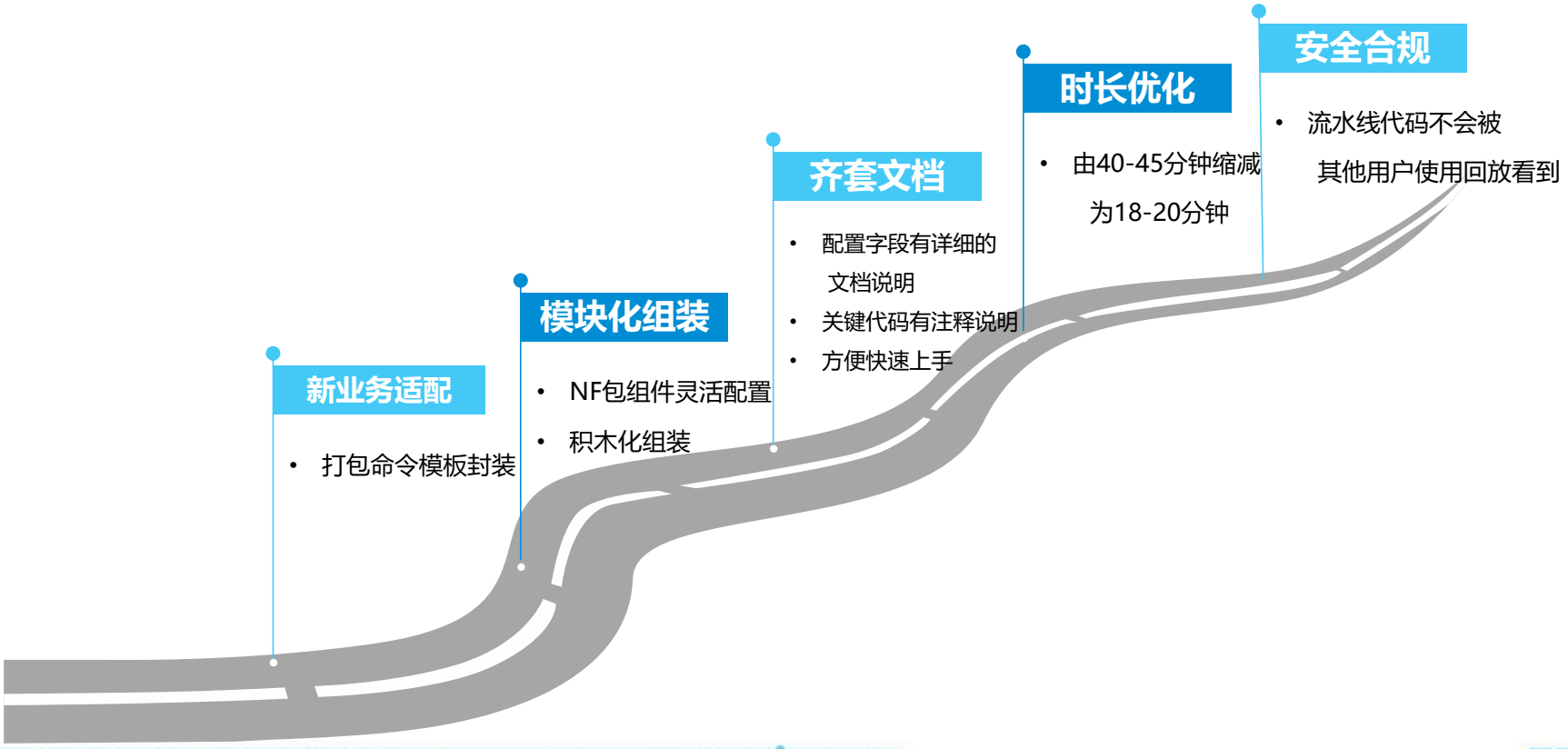
字段英文名	字段类型	键	SCD类型	关联公共维表	源系统	数据源表	取数逻辑
job_name	varchar(512)				CI	接口上报	直取
full_url	varchar(512)	PK			CI	接口上报	直取
build_number	varchar(32)				CI	接口上报	直取
build_start_time	DateTime	PK			CI	接口上报	直取
build_end_time	DateTime				CI	接口上报	直取
build_duration	int(11)				CI	接口上报	直取
build_status	varchar(128)				CI	接口上报	直取
failure_reasons	varchar(128)				CI	接口上报	直取
gerrit_branch	varchar(256)				CI	接口上报	直取
gerrit_event_type	varchar(64)				CI	接口上报	直取
gerrit_project	varchar(256)				CI	接口上报	直取
gerrit_number	varchar(64)				CI	接口上报	直取
gerrit_owner	varchar(64)				CI	接口上报	直取
queue_duration_millis	int(11)				CI	接口上报	直取
pause_duration_millis	int(11)				CI	接口上报	直取
ci_server	varchar(64)				CI	接口上报	直取
ci_type	varchar(64)				CI	接口上报	直取
ci_standard_type	varchar(64)				CI	接口上报	直取
pms_project_name	varchar(64)				CI	接口上报	直取
pms_project_id	int(11)				CI	接口上报	直取
product_name	varchar(64)				CI	接口上报	直取

变量与模板增强编排配置

1. 简化配置，减少配置信息的重复冗余和维护工作量
2. 增强能力，数据可灵活在流水线中传递

```
15    vars: # 变量是由项目自定义
16      cp2out_script: sh ci/nf_tools/packages/cp2out.sh
17      pkg_script: python ci/nf_tools/packages/component_pkg.py
18      pkg_config: ci/config/nfPkgConfig/versionConfig.xml
19      art_prefix_path: aurora_test
20    templates: # 模板是由项目自定义
21      #组件打包产物mv至领域目录模板
22      write_commit_info:
23        write_commit: commits_${task_name}.json
24        dir: out/commits
```

某重点项目使用CI服务化方案收益

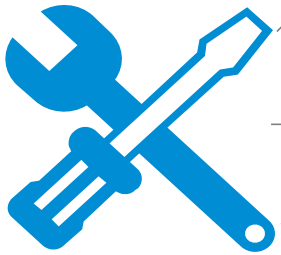


基于服务化方案快速实现安全合规内嵌

检出代码	清理&准备	Coverity扫描	Klocwork扫描	Hub扫描	NSFocus扫描	Nessus扫描	Webinspect扫描	Mcafee扫描

- 基于服务化CI框架，**可一次性快速上线**四大类、七种安全工具，免去**整合安全工具**的人力与时间投入
- **统一输出报告**内容，自动上传制品库并打标签，**100%确保**报告真实准确，达到公司治理要求，并释放汇总工作的人力**每版本0.5人天**
- 简化配置和部署，典型的项目配置**仅100多行**
- **统一安全数据模型**，自动化采集结果并传到数据中台，形成反馈闭环
- 直接对接透视图，**可视化安全扫描数据**

后续改进优化



更高的性能：15分钟

底层框架优化：配置加载、工具性能提升、资源利用率分析

业务并发优化：业务依赖解耦、版本打包调优

更易用的配置：新手1周上手

配置项多：优化和简化，降低学习使用门槛

规则复杂：结合场景，进行标准化和模版化

更快的响应：需求<1周，故障<1天

功能定制：快速简易的自定义功能开发

问题定位：日志易于分析和定位失败

谢谢！

5G 先锋

IAS 2019