亚太机电资产安全与效能提升项目
技术协议

国际商业机器（中国）有限公司

俞黎敏

YuLimin@cn.ibm.com

2022年11月25日

目　录

1. 实施技术协议内容概览

通过前期的业务调研、形成需求之后进行实施部署，部署完成后进行内容技术协议

1.1 实施任务与描述

任务阶段	描述	目标	技术协议
安装 OpenShift 集群	离线安装 OpenShift 集群	离线安装镜像、上传安装镜像、离线安装 OpenShift 集群	
环境校验和准备	验证基础环境和 OpenShift 集群	熟悉集群环境，网络可互通、可分配持久卷。	
		镜像服务器 Harbor 的搭建、配置和对接。	
		OpenLDAP 服务的搭建，组织和用户创建。有 AD 略过	
		相应用户和项目的创建。	
测试环境 CICD 组件部署	Jenkins 搭建与配置	Jenkins 在 OpenShift 中搭建及初始化	
		相关插件安装与配置	
		Jenkins-slave 导入	
		修改安全权限、全局配置等	
		Jenkins 及 GitLab 权限访问配置	
		Jenkins 及 SonarQube 相关配置	
	GitLab 搭建与配置	GitLab 在虚拟机中搭建及初始化	
		从原先的 GitLab 备份数据、恢复数据和验证	
		代码仓库和 Jenkins 对接	
	Gradle 搭建与配置	Gradle 在 OpenShift 中搭建及初始化	
		Gradle 的相关配置	
	Nexus 搭建与配置	Nexus 在 OpenShift 中搭建及初始化	
		Nexus 仓库相关配置	
		配置代理、常用依赖导入	
	SonarQube 搭建与配置	SonarQube 在 OpenShift 中安装与相关访问权限配置	
		SonarQube 代码扫描相关配置	
	Selenium 搭建与配置	Selenium 在 OpenShift 中搭建及初始化	
		Selenium 各节点的配置	
	组件组合验证	各组件的配合可用性	
Demo 程序在开发测试环境流水线	Jenkins Pipline 设计和验证	设计和编写 Jenkins Pipline	
		调试 Jenkins Pipline	
		测试验证 Jenkins Pipline	
	CICD 组件配置	CICD 其它组件的相关配置	
	OCP 配套 Demo 的对	与 Demo 应用相关的 OCP 环境和支撑、配套对象的配置	

	象配置		
	Jenkins Pipline 复用	复用给其他项目并调试验证	
生产环境 CICD 组件部署	Jenkins 搭建与配置	生产环境的 Jenkins 组件搭建和配置	
	Selenium 搭建与配置	生产环境的 Selenium 搭建与配置	
Demo 程序在生产环境流水线	Jenkins Pipline 设计和验证	设计和编写 Jenkins Pipline	
		各 CICD 组件和 OCP 对象的配置和准备	
		调试 Jenkins Pipline	
		测试验证 Jenkins Pipline	
	Jenkins Pipline 复用	复用给其他项目并调试验证	
认证和授权	统一认证和各应用授权设置	完成 2 个环境中各组件通过 OpenLDAP 实现统一认证，并对 LDAP 服务器中规划出的用户信息进行同步和授权设置。有 AD 略过	
Aspera 服务端	安装 Aspera 服务端、配置 Aspera 服务端	安装 Aspera 服务端、配置 Aspera 服务端	
Aspera 客户端	安装 Aspera 客户端、配置 Aspera 客户端	安装 Aspera 客户端、配置 Aspera 客户端	
培训及文档	文档编写及使用说明	说明手册及讲解	

1.2 代码安全管理

序号	需求	是否满足	备注
1	线上安全访问		只有连接亚太内网才可访问
2	线上代码存储		代码存储在亚太本地服务器
3	线上代码下载		通过 git 和平台网页端可下载线上存储的代码到本地
4	线上代码编辑		通过GitLab 自带Web 编辑器可线上修改代码
5	线上代码编译		四代 ESC 软件编译需要许可证书，在编译时不需要许可证书的项目进行线上编译
6	线上代码审核		变更代码后需审核者审核通过后才能提交
7	代码变更追溯		变更代码更新到线上后会留下变更记录
8	本地代码批量上传		通过 git 命令窗口可批量上传变更代码
9	人员数量与权限配置		通过配置权限可设置代码是否对人员可见
10	项目议题分配、审核、追踪		和 JIRA 功能类似
11	用户上传和下载记录		用户上传会留下记录，通过命令行终端进行查看用户登陆记录

1.3 文件安全传输与提速

序号	需求	是否满足	备注
1	提供图形化客户端工具进行上传下载		类似 FileZilla Client 的工具
2	能够设置不同的带宽占用策略		固定、高、中、低等四种方式
3	提供有自动重试功能，并可以设置重试的最小与最大的时间间隔		能自动重试，重试的最小与最大的时间间隔
4	能够根据规则过滤掉不需要上传/下载的文件		过滤规则自定义
5	支持传输安全加密算法		支持 AES-128、AES-192、AES-256、AES-128-CFB、AES-192-CFB、AES-256-CFB、AES-128-GCM、AES-192-GCM、AES-256-GCM
6	支持传输文件过程中对文件进行设置密码进行保护		传输前的密码保护与传输后采用密码解压
7	不受档案大小、形态、传输距离、网络条件限制，以最高效率迁移各地的数据文件		端到端的传输效率及吞吐量优化，传输性能与带宽成正比，与传输距离无关，丢包率影响甚微，不论距离多远，都能高速传输数据
8	实时地查看传输进度、性能、带宽使用率		实时图形化查看
9	详细的传输历史记录、日志		日志可视化查看
10	比较低的资源占用		在 30%丢包率的情况下，具备少于 0.1%的资源占用
11	应用编程接口（API）支持各种运行平台，并且在各平台上的 API 接口一致		提供 API 接口
12	支持多种开发工具和编程接口，特别是流行的开发工具。如 C、C++、Java、Go、.NET、等等。		实现对这些工具的灵活接


限制登陆、限制查看、
用户记录
限制登陆
配置允许与拒绝的IP地址或IP地址段
限制查看
在限制登陆的基础上
通过用户权限
项目组安全属性设置
Gitlab会将所有的操作记录成日志
production.log
记录每次请求的具体

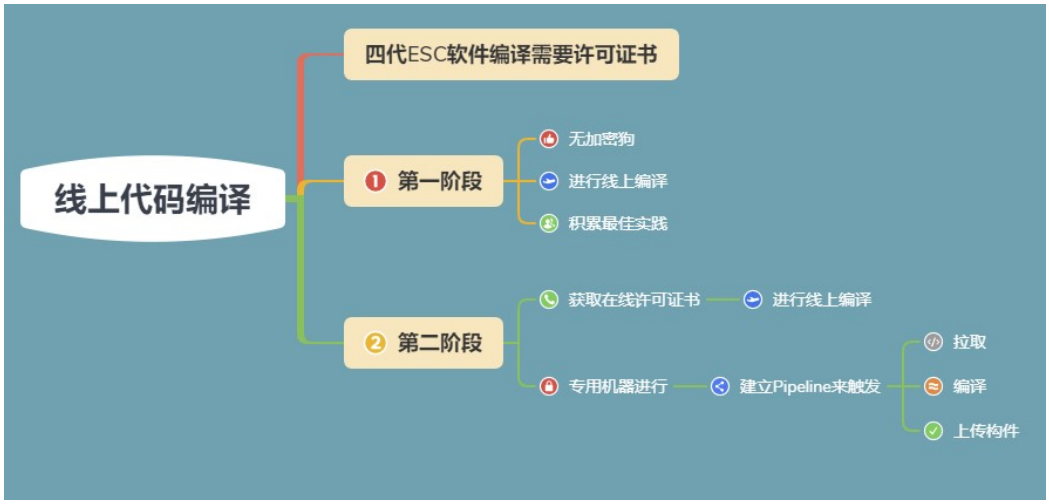
2. 项目实施技术协议资料参考

在实施的过程中，可参考以下相关资料进行。

2.1 代码安全管理

四代 ESC 软件编译需要许可证书，以目前条件难以适配线上编译方案。

第一阶段：在编译时不需要许可证书的项目进行线上编译。



第二阶段：在编译软件能够提供在线许可证书后，再进行线上编译。

在最后生成制品后，通过 Aspera 服务，可以快速下载到边远山区进行实际的汽车构件测试。速度对比如下示例：

2.1.1 开发工具链集成

开发集成 SVN/Git、Jinkens、JIRA、Sonar 等工具，开发对接开发工具链，实现源码管理、持续集成 Job 管理、Issue 跟踪、自动化测试工具等功能集成。提供开发侧事件管理与 DevOps 流水线集成关联，支撑从开发到自动部署的全自动化流程定义。

在持续集成方面，平台主要提供如下几个能力：

- 与持续集成工具/平台进行对接，实现代码自动构建、自动测试，自动输出应用镜像
- 对应用版本、应用分类、构建策略等进行管理
- 与后端持续部署对接，共同实现应用从代码到可用应用系统环境的完整 DevOps 流水线。

具体体现如下：

- 容器镜像能够通过代码、Dockerfile、应用程序包等方式进行自动构建，支持内部（Jenkins）和外部（Bamboo）的构建工具和持续集成服务。
- 支持基于应用程序包和 Dockfile 的 DevOps 流程
- 支持基于源代码（SVN/Git）的 DevOps 流程
- 支持基于 Jenkins 的持续集成对接
- 提供基于 SVN/Git 的对接和自动部署

2.1.2 DevOps 流水线设计与实施

DevOps 的技术以及工作链条有很多选择，同时对于使用容器技术的 DevOps 以及未使用容器技术的 DevOps 在工具选择会有一定不同，选择的原则应基于应用情况发展规划以及团队成员熟悉程度而定。在技术选型后，要做适当技能培训以及最佳实践分享加快团队对工具的熟悉程度更好的提升效率。

本次项目实施供选型的 Devops 技术栈工具包括：

- 持续集成持续交付服务器：Jenkins、Tekton 等
- 源代码管理：GitLab、SVN 等
- 源代码编译：Maven 等
- 仓库管理：Artifactory、Nexus 等
- 自动化测试：JUnit、TestNG、JMeter 等
- 代码质量检查：Sonar 等
- 运维自动化工具：Ansible 等
- 项目管理工具：JIRA 等
- 性能监控工具/APM 工具：Prometheus、Grafana、Instana、Zabbix 等
- API 管理工具：API 管理平台 APIC 等
- 用户目录：OpenLdap、AD 活动目录等
- 日志聚合栈：ELK/EFK 等

PaaS 的 DevOps 平台功能需涵盖从项目的需求，代码版本，代码扫描，镜像仓

库，测试，应用日志监控等。

2.1.2.1 工程初始化

工程指项目开发环节中从需求分析、功能设计、功能开发、测试部署等软件开发全生命周期的定义。我司的 DevOps 平台功能以项目和应用为管理维度。工程的初始化主要指对项目过程中涉及的入口资源，权限，计划进行数据设置。

2.1.2.1.1 基于 JIRA 的项目管理

利用现有的 JIRA 平台进行项目管理，JIRA 是澳大利亚 Atlassian 公司开发的一款优秀的问题跟踪管理软件工具，可以对各种类型的问题进行跟踪管理，包括缺陷、任务、需求、改进等。JIRA 采用 J2EE 技术，能够跨平台部署。它正被广泛的开源软件组织，以及全球著名的公司使用。

JIRA 可完成的主要功能有：

1) 管理缺陷，新特性、任务、改进或者其他任何问题

2) 干净和强大的用户界面

3) 灵活的工作流定制

4) 全文搜索和强大的过滤器

5) 企业级的权限和安全控制

6) 非常灵活的邮件通知配置

7) 可以创建子任务

8) 方便的扩展及与其他系统集成：包括 email 、LDAP 和源码控制工具等；

9) 丰富插件库

10) 项目类别和组件/模块管理

JIRA 在新建项目时，可输入项目基本信息，初始需求信息，人员信息，工具信息，项目角色等。

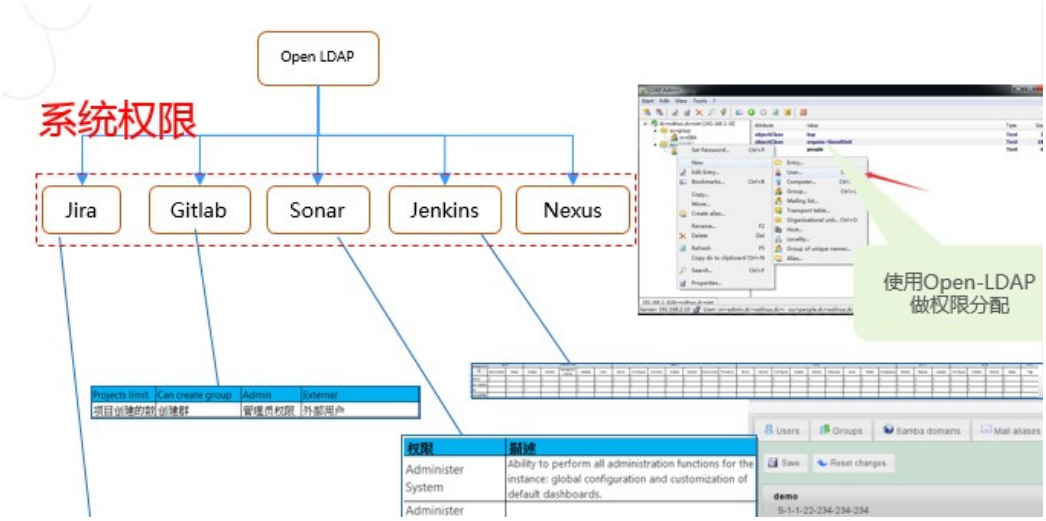
在项目初始化中，可设置项目成员不同的项目角色。

项目创建后，可创建项目需求：

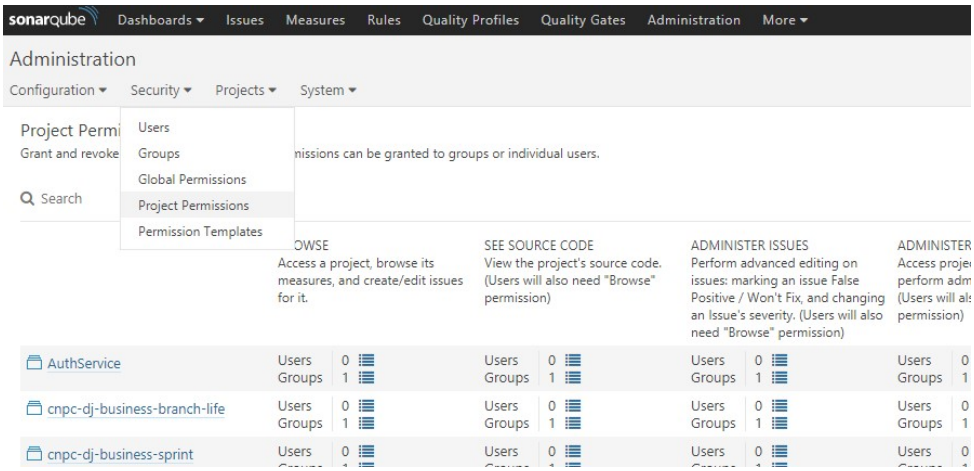
需求创建好以后，为项目成员分配任务：

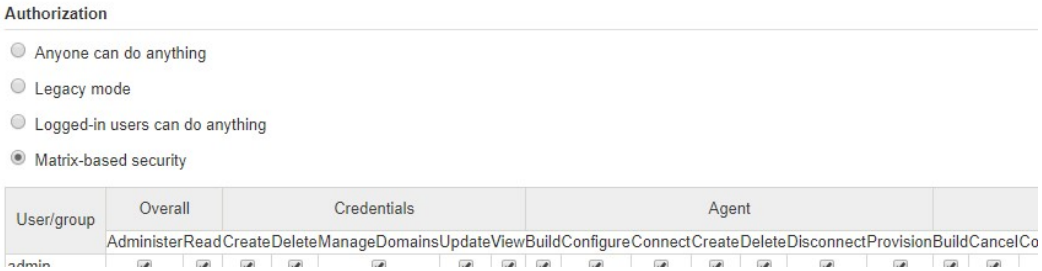
2.1.2.1.2 统一成员权限赋予

在 JIRA 上赋予的项目角色，然后通过集成的中央权限管控来赋权，赋予群组和成员权限给 GitLab，SonarQube，Jenkins。



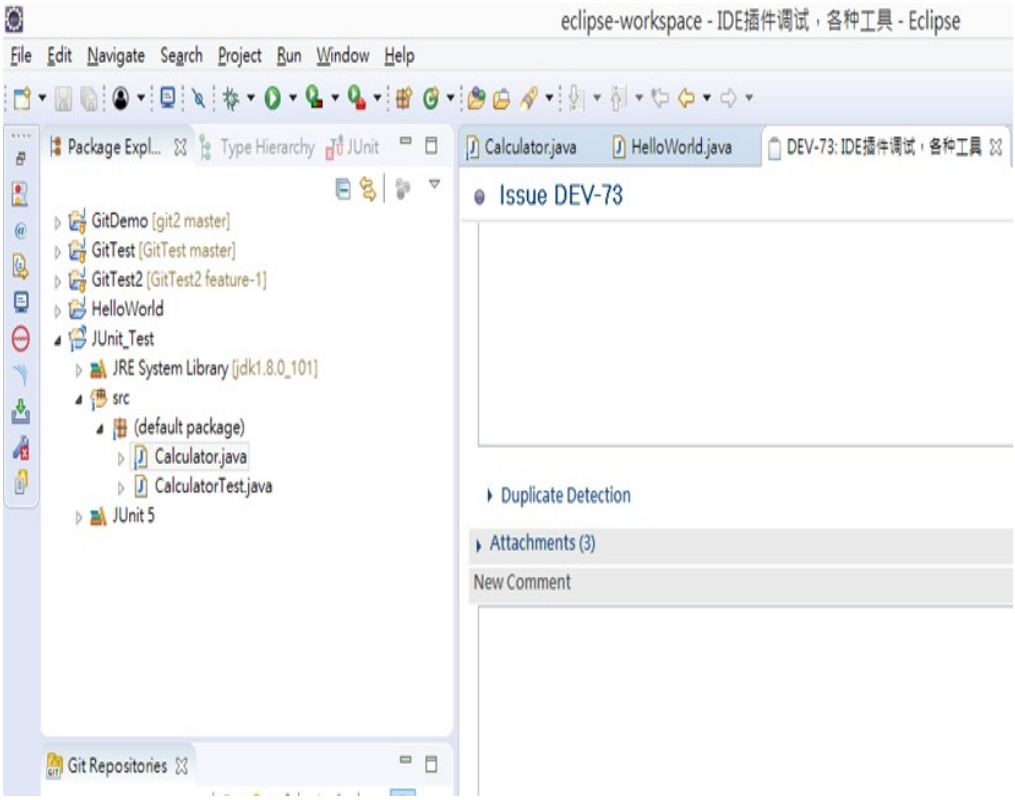
在 SonarQube 的成员信息：



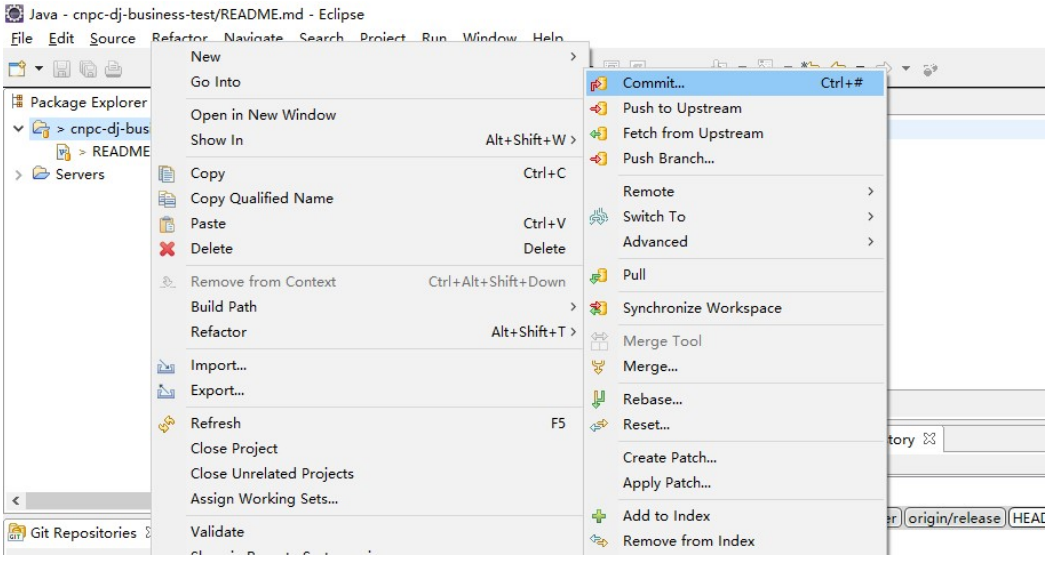


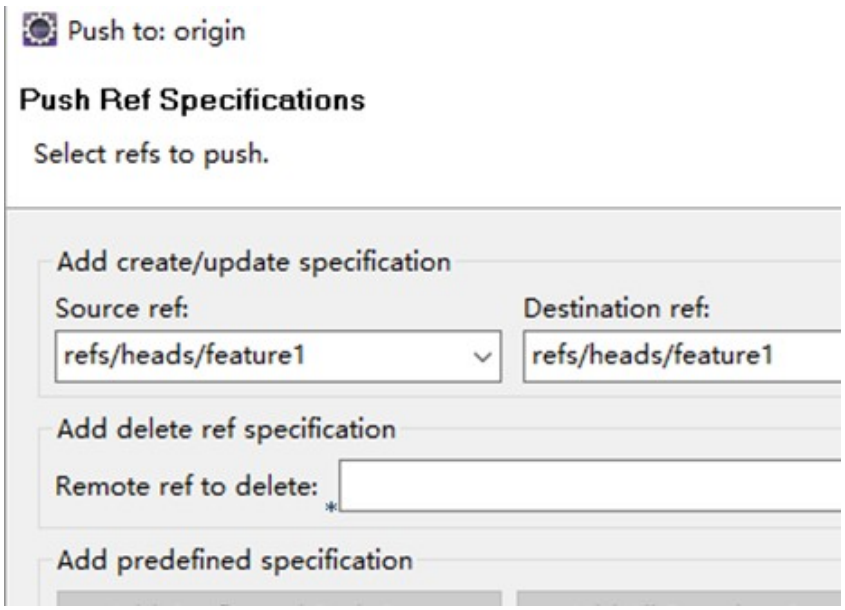
2.1.2.1.3 工程任务传递

在 JIRA 上创建的任务既可以在不同的工具上查看到。下图为在 Eclipse 开发工具上对应 JIRA 上的任务。其他开发工具使用的过程则类似，需要另外进行截图。

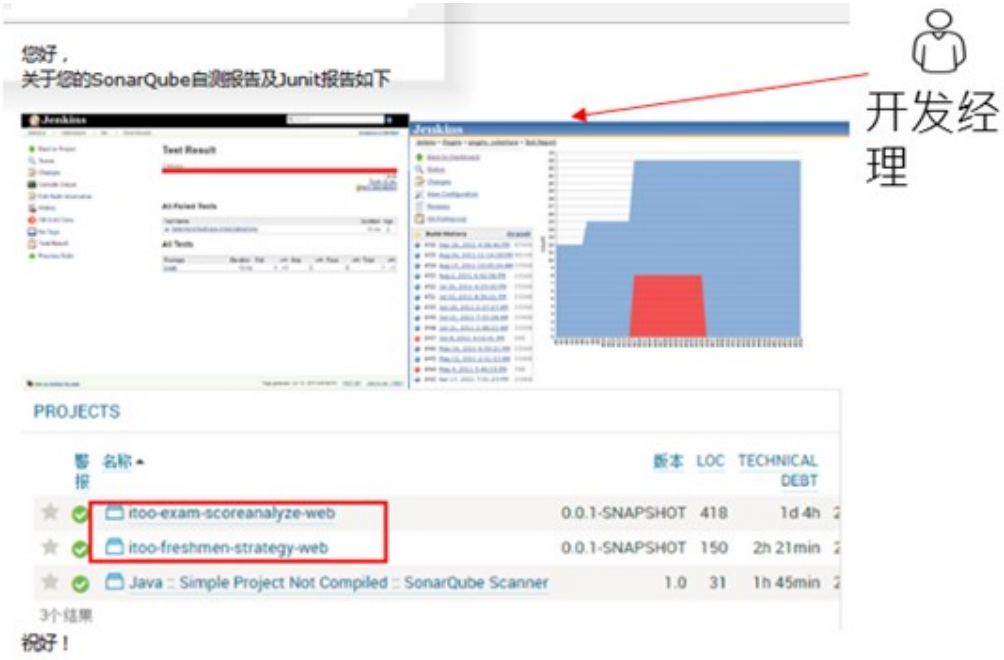


在 Eclipse 的相应任务下，创建分支，并提交代码后，该任务信息在 GitLab 上显示。



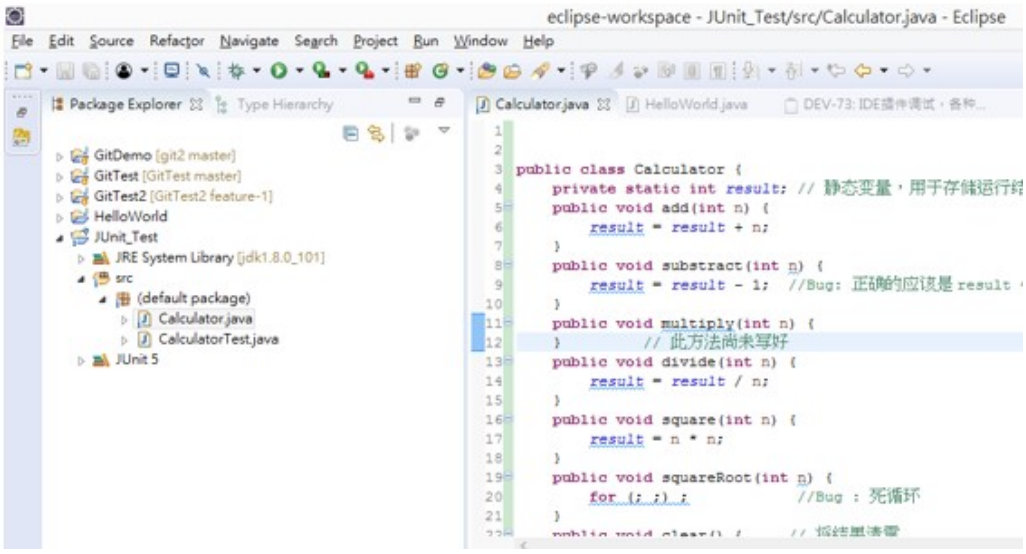

Push to: origin
Push Ref Specifications
Select refs to push.
Add create/update specification
Source ref:
Destination ref:
refs/heads/feature1
refs/heads/feature1
Add delete ref specification
Remote ref to delete:
Add predefined specification

开发经理可收到代码合并的邮件通知

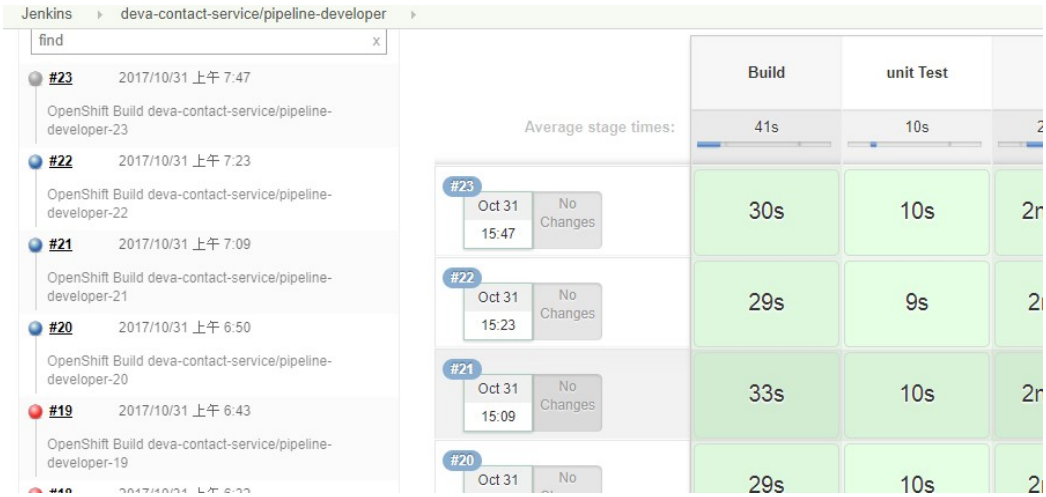


然后开发经理通过 JIRA 的 GitLab 入口，对分支代码合并请求进行审批。

在 Eclipse 开发工具中直接调用 JUnit 测试工具进行测试：



代码提交后，可在 Jenkins 中持续集成



2.1.2.1 持续集成

2.1.2.1.1 持续集成实现方案

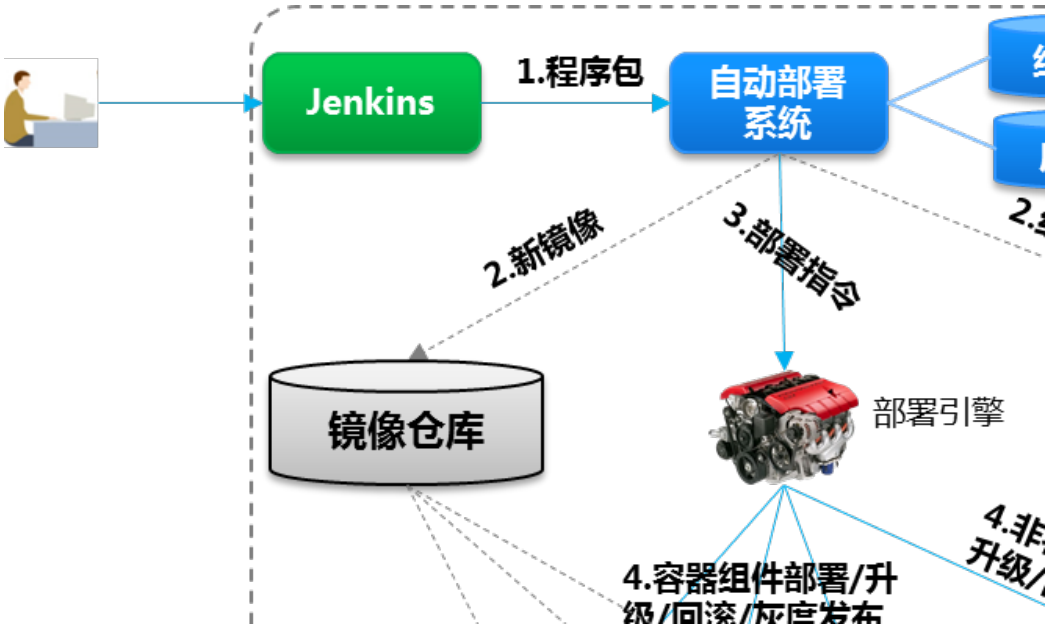
平台支持应用的持续集成和全自动化的升级部署：

平台自身集成了 Jenkins（持续集成和交付服务器），该组件可以实时监控代码以及镜像的变更，当代码或者镜像有改变就可以自动的触发自动集成的工作，而且会根据自动集成的工作量自动的扩展 Jenkins 服务器的数量，提高效率快速响应集成和部署的需求。

平台持续集成的所有组件都通过容器化的方式来实现的，因此在可维护性和扩展性方面有着巨大的优势，并且应用代码本身集成也都是在容器镜像中进行，因此大大降低了持续集成的复杂度提高了灵活性。

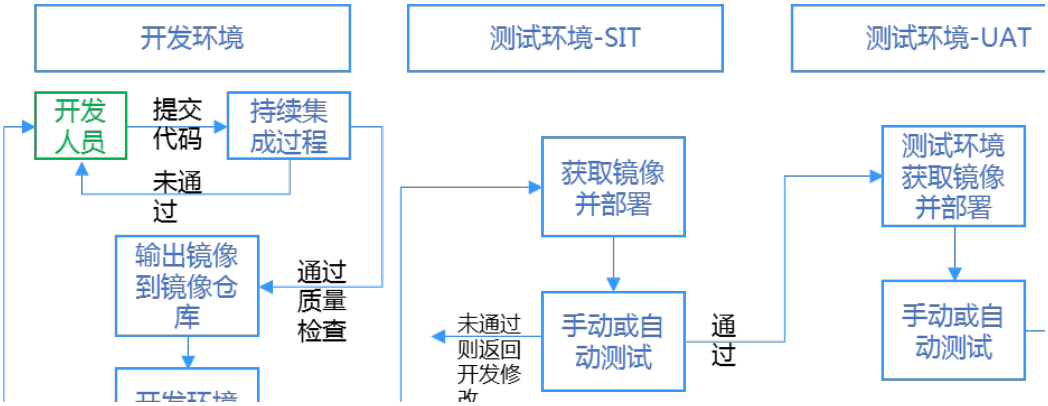
除了自带代码管理平台以外，还可以和 Sonar 集成对代码质量进行管理并作代码规范检查，同时我司将 Sonar 组件实现了容器化具有高可用可扩展灵活高效的特点。

同时平台通过自带 Jenkins，可实现应用的持续集成和全自动化升级，同时支持一键回滚和灰度发布功能。整体功能实现如下图所示：



平台默认集成自动化测试工具，该自动测试工具是一组自动化的脚本，可以帮助开发者做相关测试，还可以通过集成 TestLink 等自动化测试系统帮助开发者做更全面的测试并生成测试报告。

本平台可支持多环境管理，比如开发、SIT、UAT、生产运行环境等。通过不同的环境管理，可对硬件资源、镜像、容器实现隔离处理，如需要相关资源共享，可通过共享同步功能实现不同环境的互通。以镜像作为标准交付件，实现多环境流转，其过程如下：



2.1.2.2 自动化测试

虽然持续交付可以包括由质量保证团队执行的手动测试阶段或最终用户技术协议测试，但是自动化测试是加快交付周期并提高质量的关键功能。通常，持续集成服务器将负责执行大多数的自动化测试，以验证每个开发人员提交的代码。然而，当系统部署到测试环境中时，某些自动化测试可能需要被执行，因此还应该尽可能多的实现自动化测试。自动化测试应该是详尽的，能够覆盖测试应用程序的多个方面，包括单元测试、集成测试、技术协议测试、负载测试、性能测试、模拟测试、上测试软件、冒烟测试、验证新部署环境的状态和完整性、质量测试。

这些测试分布在部署流水线中，随着流水线中的测试越来越详细和价值越来越高，在生产环境中，这些发布候选制品越来越可靠。其目标应该是尽早确定有问题的构建，以避免返工，尽快缩短周期时间和获得反馈。

测试分为单元测试，集成测试，接口测试，UI 测试等，其中，单元测试不需要

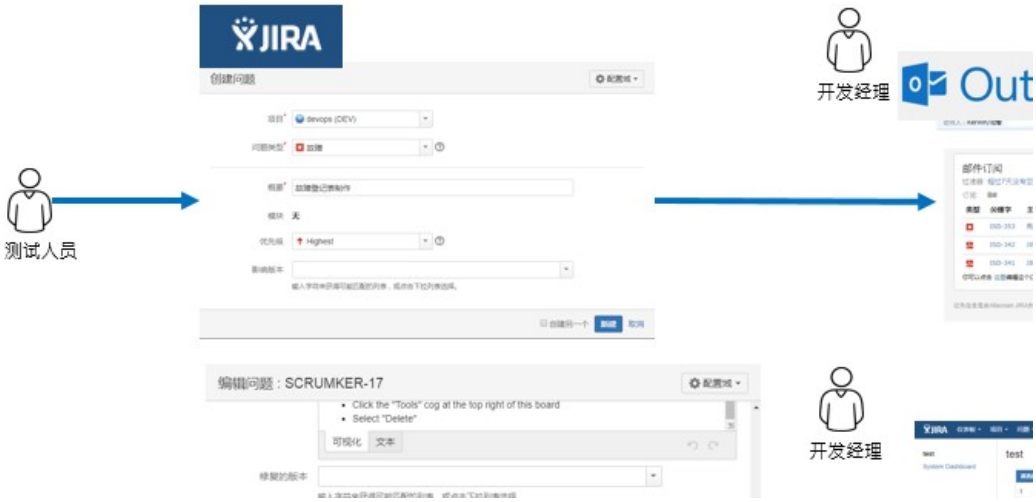
部署环境，直接基于工程仿真即可。测试环境的部署脚本需要应用运维人员编写，包含 SIT 环境，UAT 环境，Stage 环境等。同时，应用运维人员需要为应用配置 Pipeline。

Pipeline 的配置示例如下图：

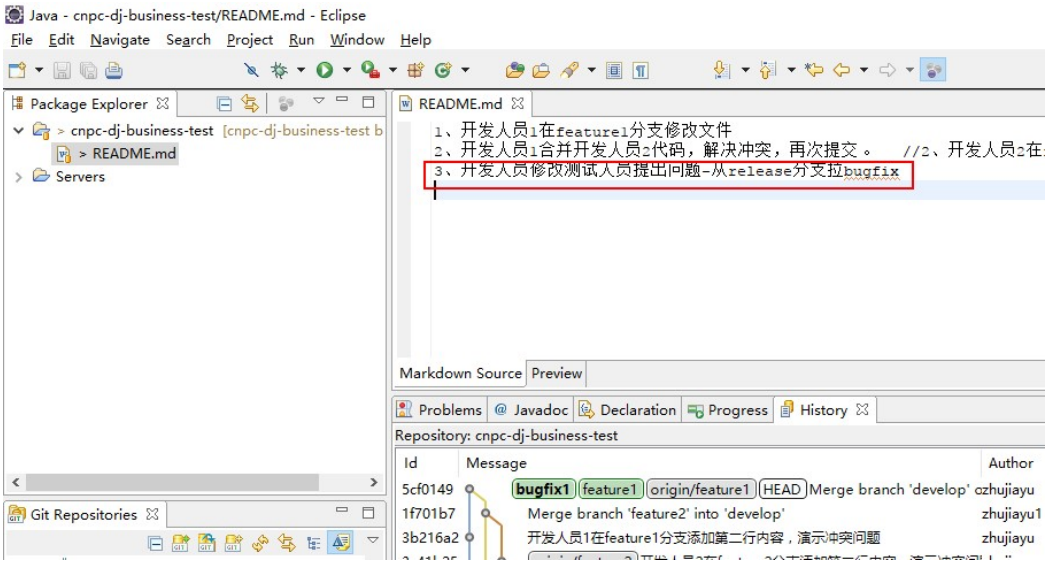
通过 Jenkins 重新构建后，基于环境部署脚本，自动把构建包部署到测试环境。

测试人员测试，提 Bug，通知开发人员，可在 JIRA 上完成。

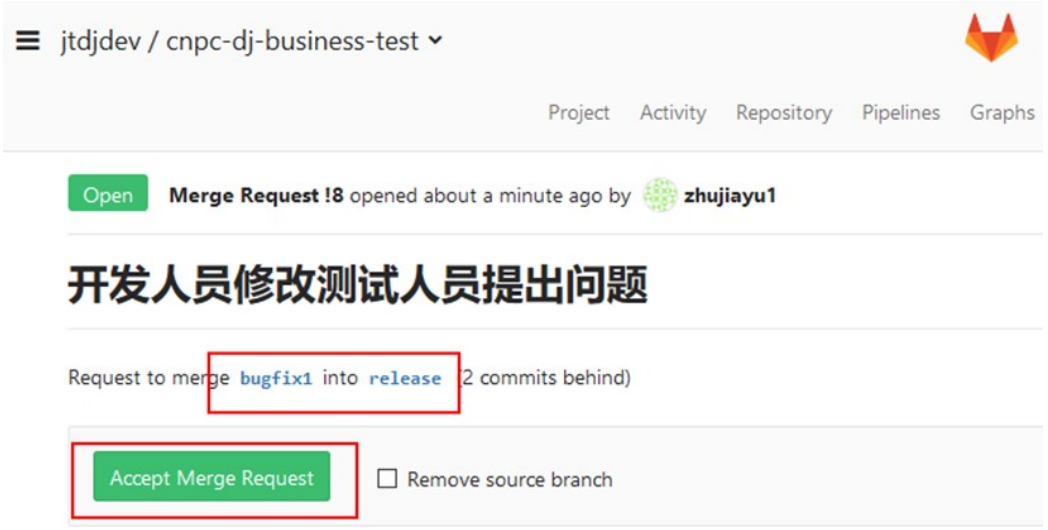
开发主管与开发人员接到邮件和站内消息通知后，主管确认消息紧急程度，开发人员拉取代码修改任务。



开发人员通过 Release 拉 bugfix 分支：



开发人员修改完 BUG，合并回 Release 分支，测试人员回归测试。

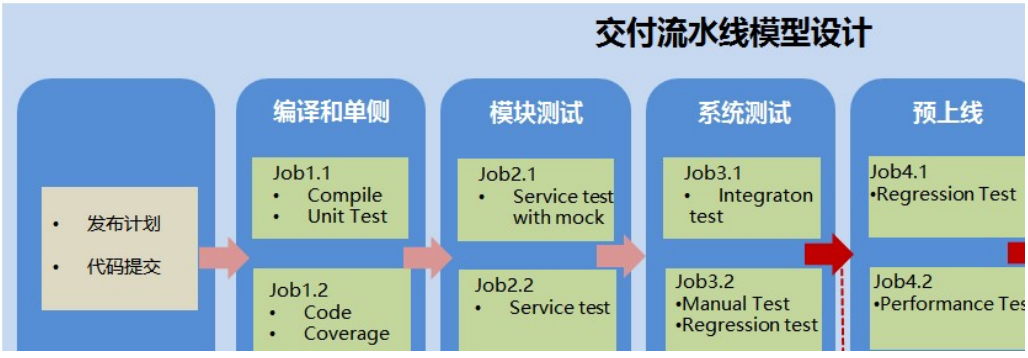


测试人员回测没有问题后，合并至 develop 分支。开发经理收到通知后合并代码到 Master 分支，提起部署到测试环境进行集成测试的申请。

2.1.2.3 持续交付

PaaS 提供完善的 DevOps 持续交付流水线服务。根据研发和运维场景的差异化需求及互联网所需的实际环境，可以实现定制化以及标准化流水线模型，在模型设计的同时，形成应用发布流水线规范和标准。具体来讲，通过 Jenkins 这样的部署流水线工具，甚或在特定的业务场景下可以直接嵌入工作流或诸如 Ansible 这样的自动化运维工具来实现更加负载，更加广泛适应化，直接与 IaaS 可对话的应用部署自动化能力。这样平台才能够根据应用类型自定义各种流程动作，包括与 IaaS 直接耦合的流程动作。例如动作类型可以包括 Jenkins 应用构建、Dockerfile 镜像生成、编排部署、平台阔缩容、监控数据回馈、测试行为等等；源码直接构建出应用镜像将是一个更加理想的选择。从一开始就形成代码至镜像的自动化 DevOps 部署流程，那么可见的未来业务都将不再只有开发者可操作，可维护，而是任何人都可以简单的导航式重构或重启，这将极大的提高互联网软件资产方面的自主权和主动性。源码直接构建出应用镜像的配置可以如下操作：应用按规范提交源码，触发应用构建，应用构建完成，触发定义好的镜像构建，镜像构建完成自动提交到镜像仓库。支持 CI 流水线的编排，每个项目成员可以根据自己的需求配置 CI 流水线；CI 流水线的起点可以来自于代码仓库或者镜像仓库。同时，平台支持有构建历史查看和构建操作的审计日志。

平台发布流水线模型设计具体如下：



模型的设计以 DevOps 理念为基础，构建可靠可重复的交付流水线。通过流水线的设计，可以实现全局过程标准化、自动化、可视化，在关键流程和节点管控，在并行开发过程中的进行协同和管理：

- 划分阶段设计（Stage）：

根据通用的业务情况，部署流水线可划分阶段分为发布计划、代码提交、编译和单侧、模块测试、系统测试、预上线、生产灰度、生产全量几个阶段。各阶段串行执行，当前一个阶段完成后，下一个阶段自动触发进行，也可以进行手动触发。

- 执行阶段设计（Job）：

每个阶段有多个 Job 可以串行或者并行，可以进行定时 Job 执行，在执行中可以自动判断或人工标记 pass 或者 fail。

- 质量门（通过标准）：

在这个流水线设计中，设定各阶段操作标准，包括 pass/Fail 判定标准、测试通过率、代码覆盖率等，保证项目、应用交付质量。

- 决策点（人工干预）：

为适合多场景部署，开设人工干预功能，在流水线中可以配置人工决策，当一键决策后，流水线可以自动执行，也可配置阶段见操作触发模型。

- 自定义流程设计

流水线设计支持自定义流程设计，可以根据客户不同的应用部署场景进行定制化流水线制定，满足客户 5 套应用系统发布流水线的构建需求。

在流水线设计中，采用 GitFlow 开展版本管理，指定版本管理流程和规范。这样能够实现：

1.Feature 的开发不影响整体版本功能

2.多分支管理

3.分支合入管理

4.版本管理

5.线上代码出 Bug，进行快速修复

版本管理分支设计包括如下：

- Production 分支

也就是我们经常使用的 Master 分支，这个分支最近发布到生产环境的代码，最近发布的 Release， 这个分支只能从其他分支合并，不能在这个分支直接修改

- Develop 分支

这个分支是主开发分支，包含所有要发布到下一个 Release 的代码，这个主要合并与其他分支，比如 Feature 分支

- Feature 分支

这个分支主要是用来开发一个新的功能，一旦开发完成，我们合并回 Develop 分支进入下一个 Release

- Release 分支

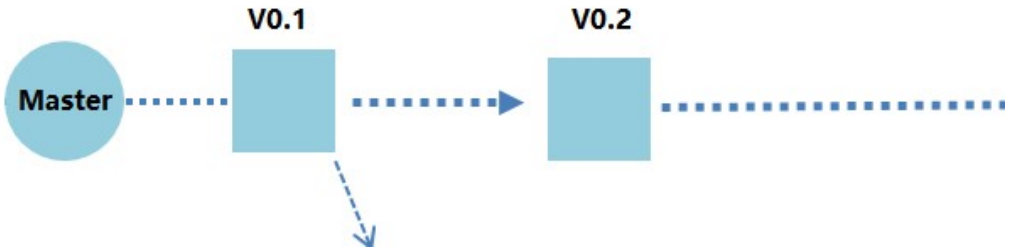
当需要一个发布一个新 Release 的时候，我们基于 Develop 分支创建一个 Release 分支，完成 Release 后，我们合并到 Master 和 Develop 分支

- Hotfix 分支

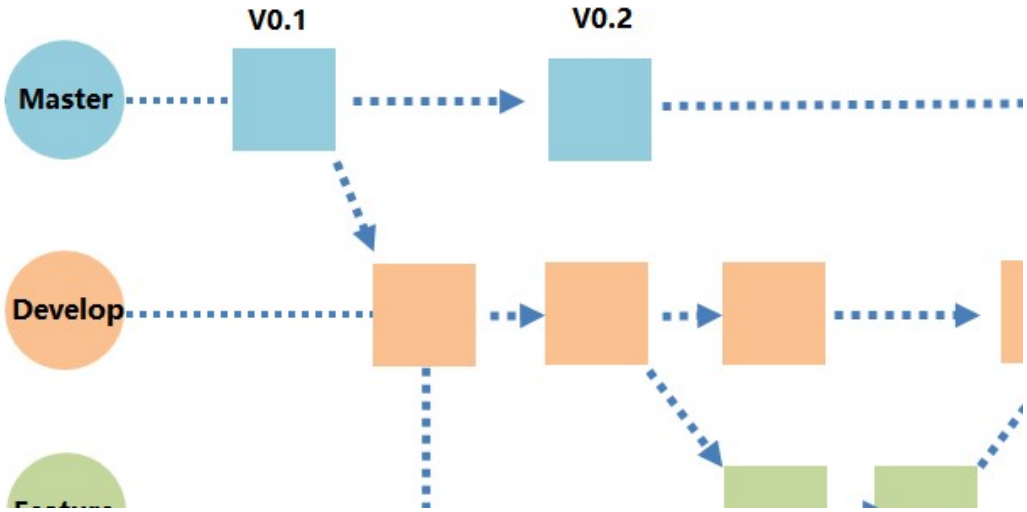
当在 Production 发现新的 Bug 时候，我们需要创建一个 Hotfix, 完成 Hotfix 后，我们合并回 Master 和 Develop 分支，所以 Hotfix 的改动会进入下一个 Release

各工作流设计如下：

初始分支：所有在 Master 分支上的 Commit 应该 Tag



Feature 分支：Feature 分支做完后，必须合并回 Develop 分支，合并完分支后一般会删掉这个 Feature 分支，但是我们也可以保留。



Release 分支：基于 Develop 分支创建，打完 Release 分之后，我们可以在这个 Release 分支上测试，修改 Bug 等。同时，其它开发人员可以基于开发新的 Feature 发布 Release 分支时，合并 Release 到 Master 和 Develop， 同时在 Master 分支上打个 Tag 记住 Release 版本号，然后可以删除 Release 分支了。

平台的 DevOps 流水线设计是一套实践方法，在保证高质量的前提下缩短系统变更从提交到部署至生产环境的时间，其中持续集成和持续交付是流水线设计里面非常重要的一环。持续交付是一组能够帮助软件开发团队极大的提高其软件交付的速度和质量的模式和最佳实践组成。不同于低频率发布相对较大的版本，实施持续交付可以比通常更频繁地将更小批量的变更投入生产， 例如每周，每天或一天之内就能够发布多个版本。

持续交付的关键构建模块为自动化，尽管在持续交付流程中采取手动步骤是非常有效和现实的，但自动化是加快交付步伐和缩短周期时间的关键。自动化模块功能设计如下：

- 自动化构建和打包

实现将开发人员的源代码转换为部署就绪制品。目标是实现单个脚本或命令，能够将版本控制的源代码转换为单个可部署的制品。

- 自动化持续集成

持续集成是持续交付的基本组成部分。它涉及整合多个开发人员的代码，并不

断编译和测试集成的代码库，以便尽可能早地识别错误。此过程将利用自动化构建，从而使持续集成服务器不断地发布包含开发团队集成工作的部署制品，每个构建的结果都是可行的发布候选。

目标：1. 实现持续集成过程就是持续输出一组可用于部署的制品；2. 评估基于云的持续集成产品，以加快您的持续交付进程；3. 通过发布跟踪软件的集成，整合对每个构建所发生变化的详细审计跟踪

- 自动化测试

虽然持续交付可以包括由质量保证团队执行的手动测试阶段或最终用户技术协议测试，但是自动化测试是加快交付周期并提高质量的关键功能。通常，持续集成服务器将负责执行大多数的自动化测试，以验证每个开发人员提交的代码。然而，当系统部署到测试环境中时，某些自动化测试可能需要被执行，因此还应该尽可能多的实现自动化测试。自动化测试应该是详尽的，能够覆盖测试应用程序的多个方面，包括单元测试、集成测试、技术协议测试、负载测试、性能测试、模拟测试、上测试软件、冒烟测试、验证新部署环境的状态和完整性、质量测试。

这些测试分布在部署流水线中，随着流水线中的测试越来越详细和价值越来越高，在生产环境中，这些发布候选制品越来越可靠。其目标应该是尽早确定有问题的构建，以避免返工，尽快缩短周期时间和获得反馈。

- 自动化部署

支持软件团队将发布后续推送到不同部署环境进行不同类别的测试。例如，常见的情况是将软件部署到测试环境进行人为的质量检查测试，然后部署到性能测试环境，进行自动化负载测试。 如果构建通过该测试阶段，则应用程序可能稍后部署到用于 UAT 或 Beta 测试的独立环境中。理想情况下，将任意发布候选制品以及与之通信的其他系统可靠地部署到任意环境中的这个过程应尽可能实现自动化。如果按照计划的速度持续交付，那可能需要每天或每周多次执行，因此它的工作速度和可靠性至关重要。用自动化方式在环境之间移动软件是作为继续交付的团队的主要特性之一，因此这也是继续交付的关键重点。

- 基础架构即代码

当配置方面不一致时候，例如当开发环境与测试不同，或当测试环境与生产不一致时，会发生非常常见的一系列生产事件，错误最终导致返工。平台提供配置管理工具和环境建模工具实现将基础架构和平台提供版本控制代码来避免这种情况，然后将环境自动构建成一致和可重复的状态。

- 容器技术

提供基于容器技术的配置管理，可支持持续交付准备中的多个场景，例如生产环境的首次运行，是创建可重复本地设置的更轻量级手段。

- 自动的生产部署

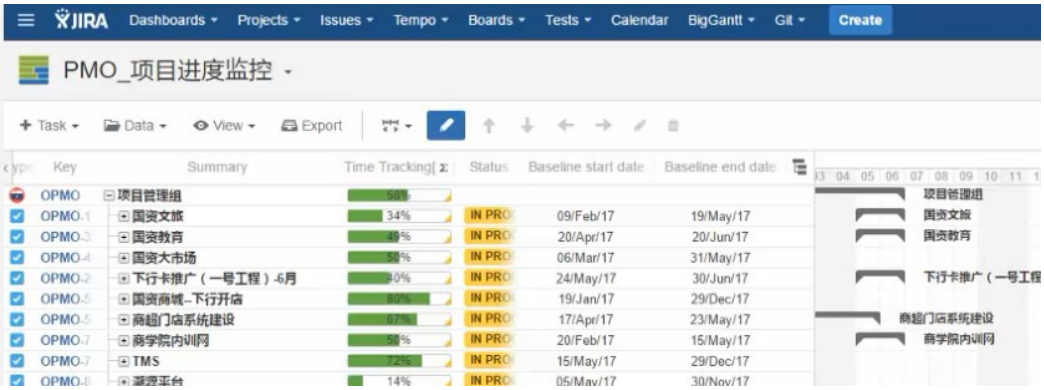
平台提供自动的生产部署设计，包括 1. 完全自动化生产部署过程，使其可以从单个命令或脚本执行；2. 在生产系统生效的同时，可以部署软件的下一个版本，并切换到新版本，而不会降低服务质量；3. 能够使用完全相同的部署到其他环境的流程来部署到生产；4. 实施下面描述的最佳做法，例如金丝雀释放，回滚和监控，以提高生产系统的稳定性。

基于平台提供的流水线服务，可以使用户从迭代时间以及快速发布中收益。自动化会取代很多人工任务，环境和发布会变的一致，发布候选者将使用自动路由和自助服务工具在各个流水线阶段中流动。软件可以近似于时刻处在生产就绪状态，伴随着发布候选人最后从流水线出来的频率远大于从传统途径的频率。平台具备该能力可提升和推进更快的发布周期，而这会使得系统的稳定性更强。

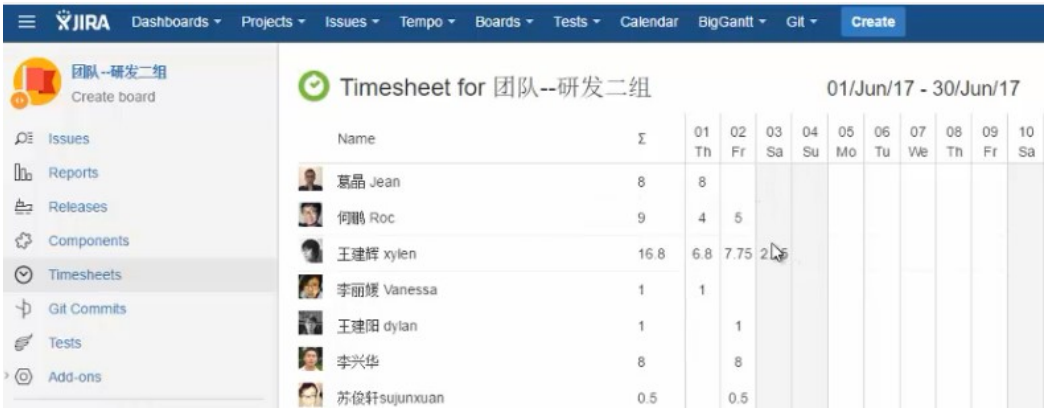
2.1.2.4 报表展示

端到端的 DevOps 报表基于流水线的各个工具完成，用来作为问题分析以及产品运营的基础参照，DevOps 平台提供报表应配置展示统一入口。统计报表包含但不限于以下类型：

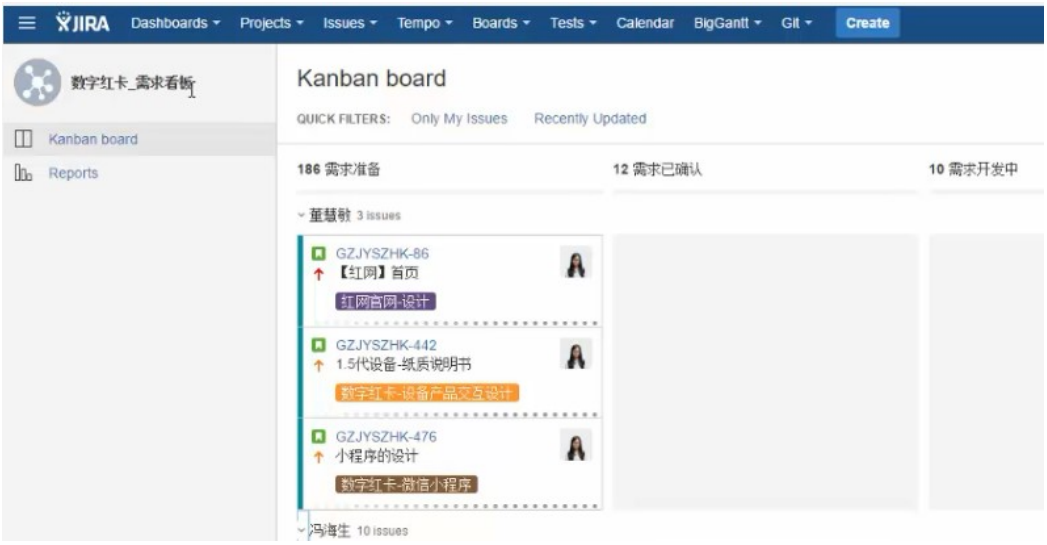
1）基于 JIRA 的项目完成情况统计



2）基于 JIRA 的项目投入统计



3）基于 JIRA 对需求个数、分配情况、完成情况的统计和展示。

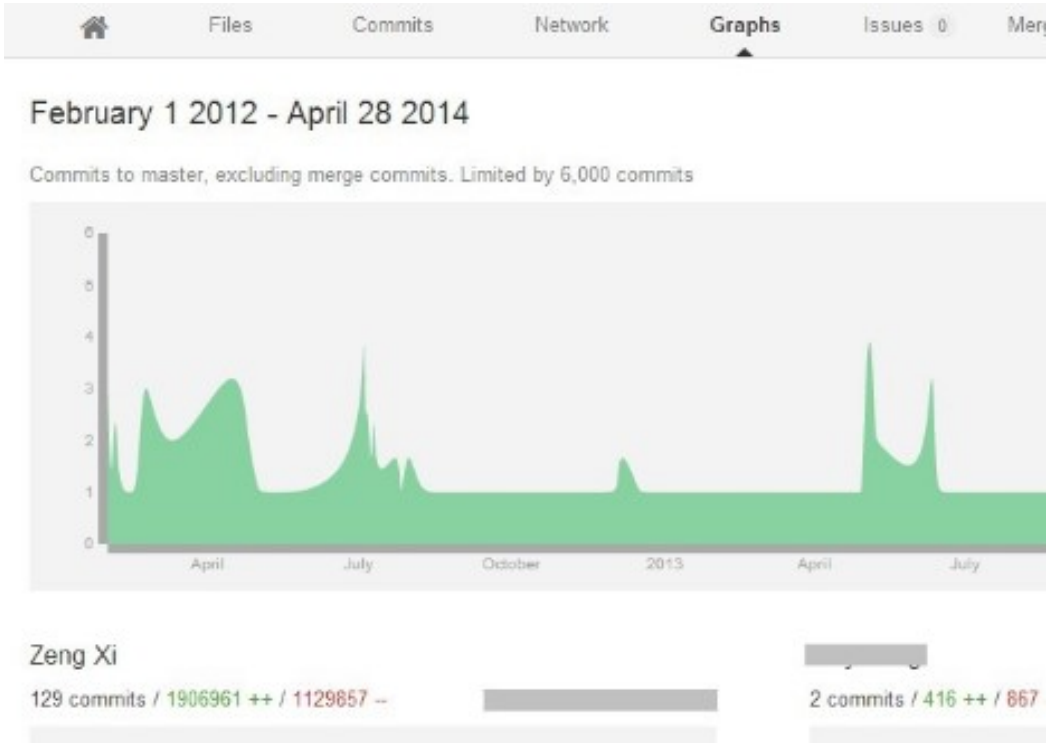


4）基于 SonarQube 的静态代码分析结果。

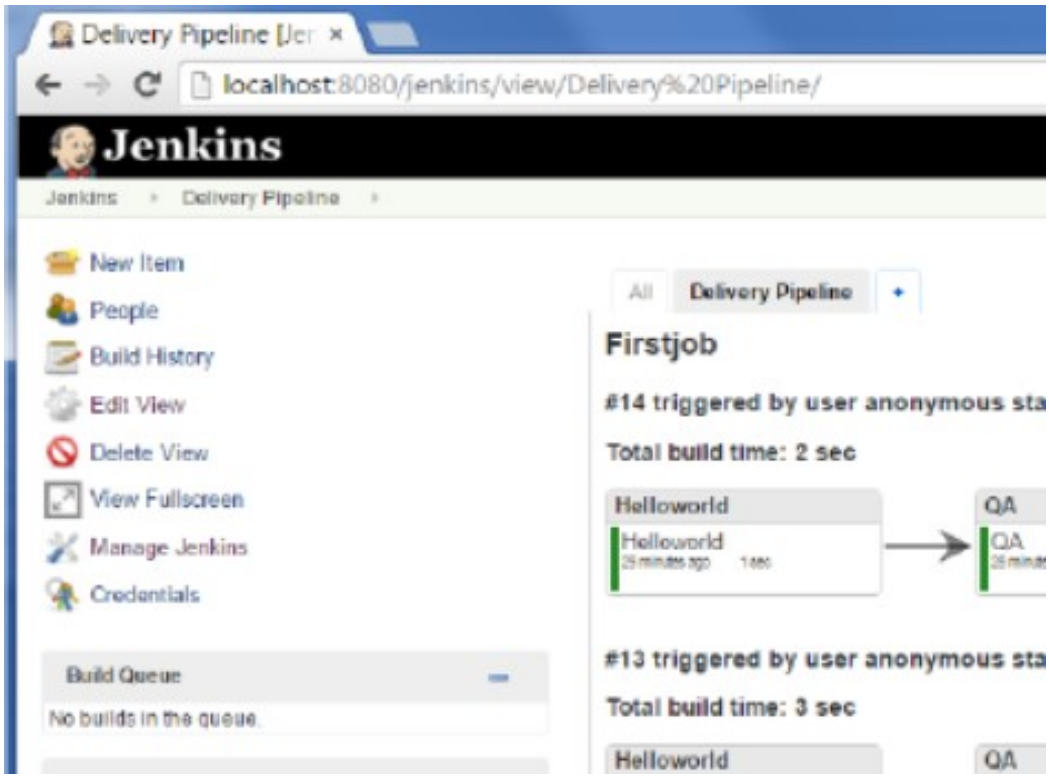
5）基于 SonarQube 的代码安全扫描结果。

6）基于 Junit 的代码的测试覆盖率结果。

7）基于 GitLab 的代码提交统计。

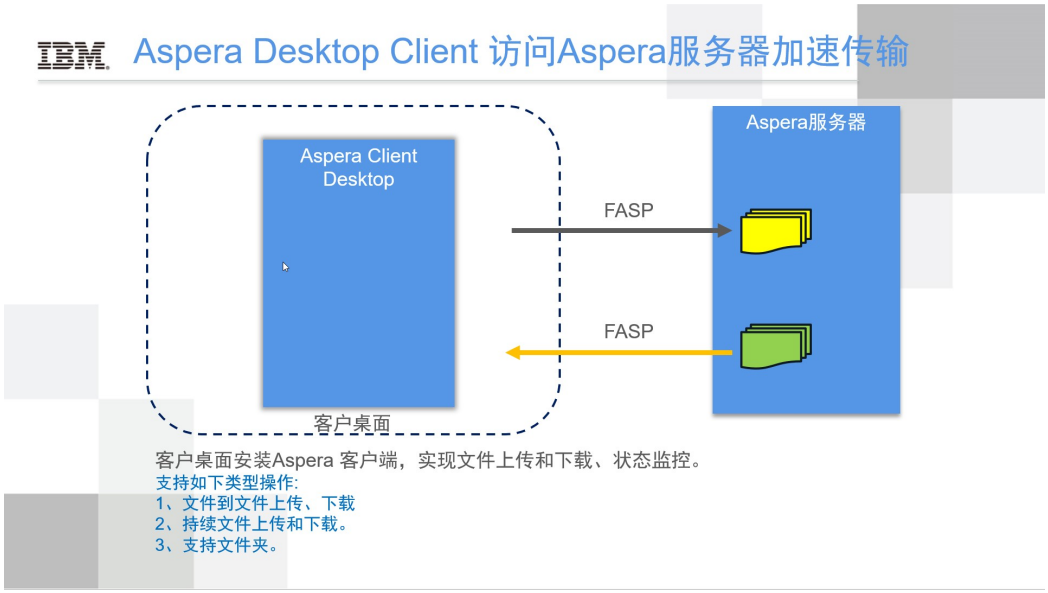


8）基于 Jenkins 工作流的状态统计。



9）基于 GitLab 的效率度量。

2.2 文件安全传输与提速



2.3 安装 Aspera 服务端 for Linux

环境最低要求：
Processor: 2 GHz duo-core CPU, or greater
Memory: 4 GB RAM
Storage: 2 GB disk space
Linux 64-bit: Ubuntu 14.04 LTS, 16.04 LTS, 17.10. RHEL 6-8. CentOS 7-8. SLES 11-12. Debian 7-9. Fedora 26-27. Kernel 2.4 or higher and Glibc 2.5+

2.3.1 配置 Firewall

```
关闭 Linux firewall

systemctl status firewalld.service

systemctl stop firewalld.service
```

High Speed Transfer Server：
Inbound TCP/22 (or other TCP port set for SSH connections): SSH 连接用的端口，建议使用非缺省的 22，可以使用如 TCP/33001 这样的端口
Inbound UDP/33001: FASP 传输用的端口，缺省使用 UDP/33001
Inbound and outbound TCP/8080 and TCP 8443 (or other TCP ports set for HTTP/HTTPS fallback): HTTP fallback 使用的端口
Inbound TCP/80 and TCP/443: HTTP and/or HTTPS web 访问使用的端口
Local firewall: 如果在安装的服务器上有本地 firewall (例如 iptables)，确认没有阻止 SSH 和 FASP 传输端口 (例如 TCP/UDP 33001).

远程客户机：
Outbound TCP/33001: 允许 Aspera Client outbound connections 的 TCP 端口(缺省是 TCP/33001, when connecting to a Windows server, or on another non-default port for other server operating systems).
Outbound UDP/33001 (如果需要，会是一个范围): 允许 Aspera Client outbound connections 的 FASP UDP port (缺省是 33001).
Local firewall: 如果在 client 机器有本地 firewall (such as iptables), 确认没有阻止 SSH 和 FASP transfer ports (例如 TCP/UDP 33001).

2.3.2 安装 Aspera

下载安装文件：IBM_ASPERA_HIGH-SPEED_TRANSFER_SE.zip

解压缩 zip 文件：

```
unzip IBM_ASPERA_HIGH-SPEED_TRANSFER_SE.zip
```

会看到有个 ibm-aspera-hsts-4.0.1.182389-linux-64.rpm 文件

通用 rpm 安装：

```
rpm -Uvh ibm-aspera-hsts-4.0.1.182389-linux-64.rpm
```

如果安装出现“error: Failed dependencies:”这样的错误，说明缺少需要的依赖包，可以通过如下的命令把需要的依赖包一起安装：

```
yum --nogpgcheck install ibm-aspera-hsts-4.0.1.182389-linux-64.rpm
```

在有些 centos 7 的环境中，会出现如下提示：

```
To complete the Connect Server Installation:
- Install the Data::Dumper Perl module
- Install the Digest::MD5 Perl module
```

运行如下的命令安装这两个依赖包：

```
yum install perl-Digest-MD5
yum install perl-Data-Dumper
```

```
以下几个服务是 aspera 通常要用的，确保它们是正常启动的
# systemctl start asperacentral
# systemctl start asperahttpd
# systemctl start asperarund   (manages both asperawatchd and asperawatchfolderd)
# systemctl start asperanoded
```

安装 license 文件

```
#GUI 方式，运行
asperascp

#命令行方式，运行
#拷贝得到的 license 文件内容到下面 license 文件里，保存
vi /opt/aspera/etc/aspera-license

#校验获得的 license
ascp -A
```

2.3.3 配置 SSH 服务

编辑 ssh server 的配置文件：

```
vi /etc/ssh/sshd_config
```

根据需要修改 ssh 的配置文件

```
Port 33001
PubkeyAuthentication yes
PasswordAuthentication yes
PermitEmptyPasswords no

PermitRootLogin yes

MaxAuthTries 30
```

重新启动 ssh server：

```
systemctl restart sshd.service
```

2.3.4 重启 Aspera 服务

如果 asperacentral 停止了，或者修改了 aspera.conf 中<central_server>、<database>小节的内容，需要重启：

```
systemctl restart asperacentral
```

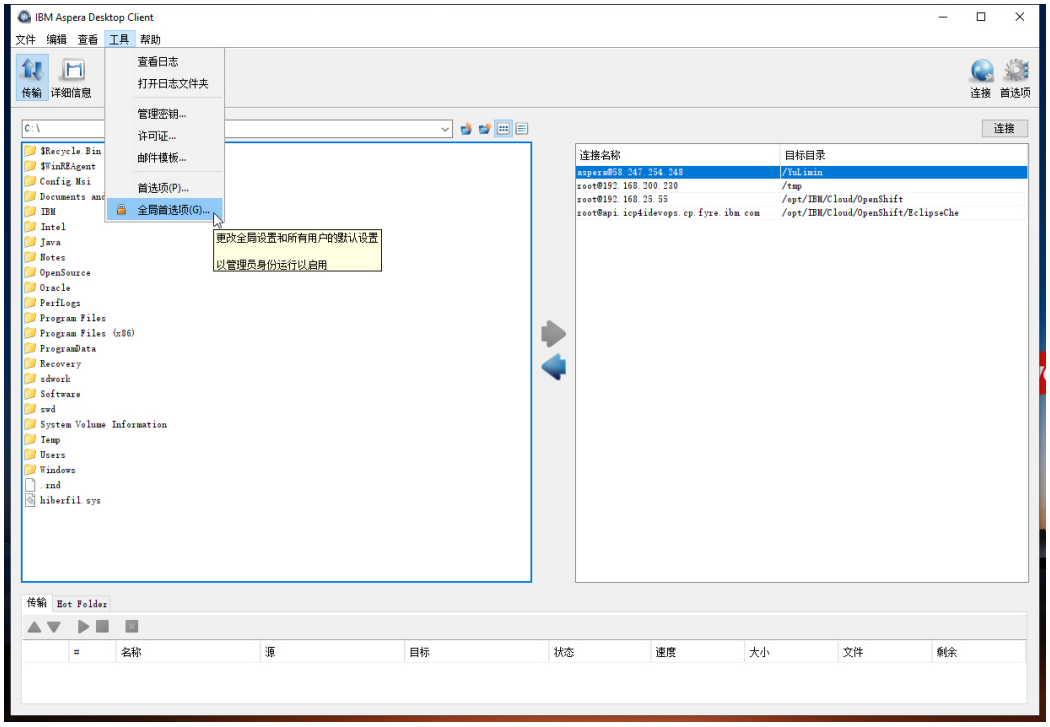
如果修改了 aspera.conf 文件，需要重启：

```
systemctl restart asperanoded
systemctl restart asperahttpd
```

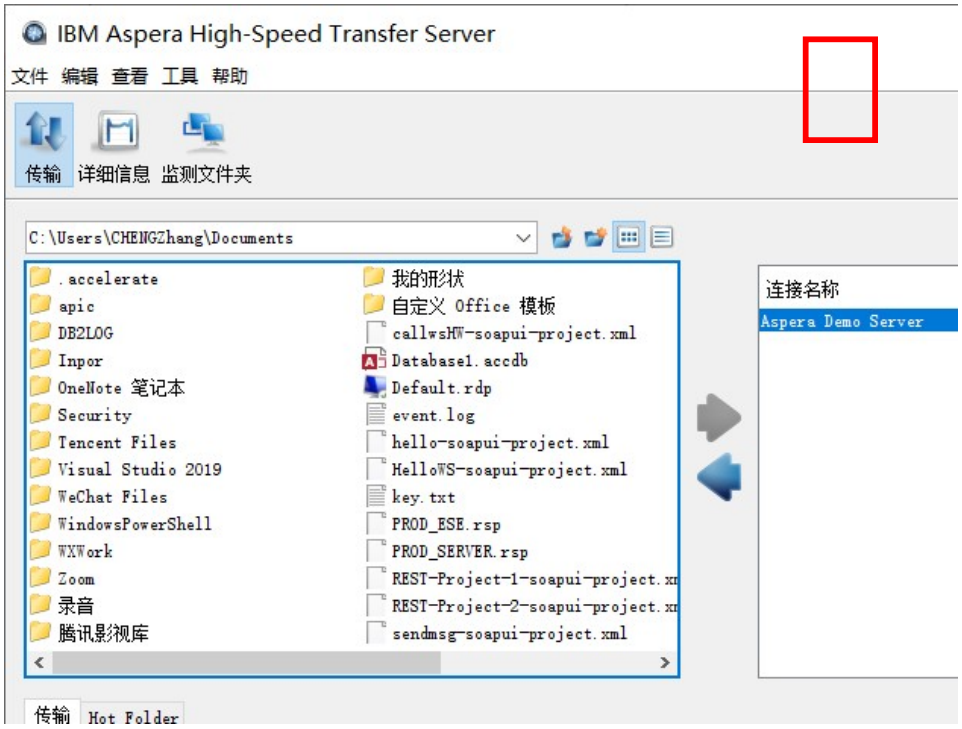
2.4 安装 Aspera 客户端 for Windows

2.4.1 安装 Aspera 客户端

Aspera 客户端是常规的安装软件，根据向导提示直接安装即可运行。



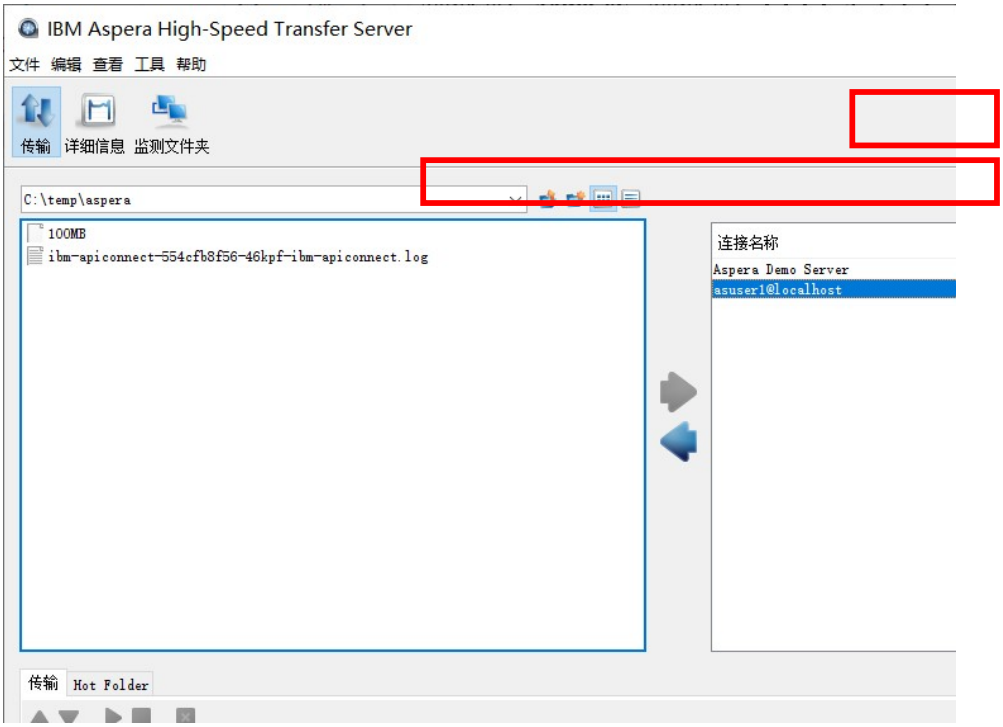
2.4.2 连接当前的 Aspera 服务



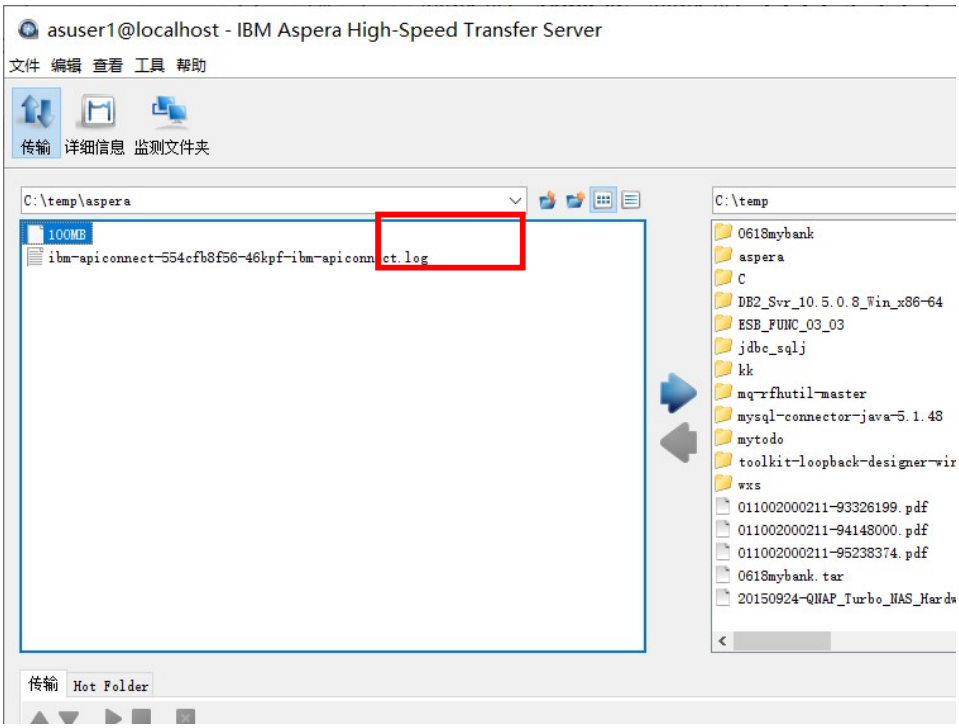
点击“+”添加连接

可以点击“测试连接”来验证是否可以连接 ssh 服务
点击“确定”返回到主界面

选择刚才创建的条目，然后点击“连接”即可连接到本地的 Aspera 服务

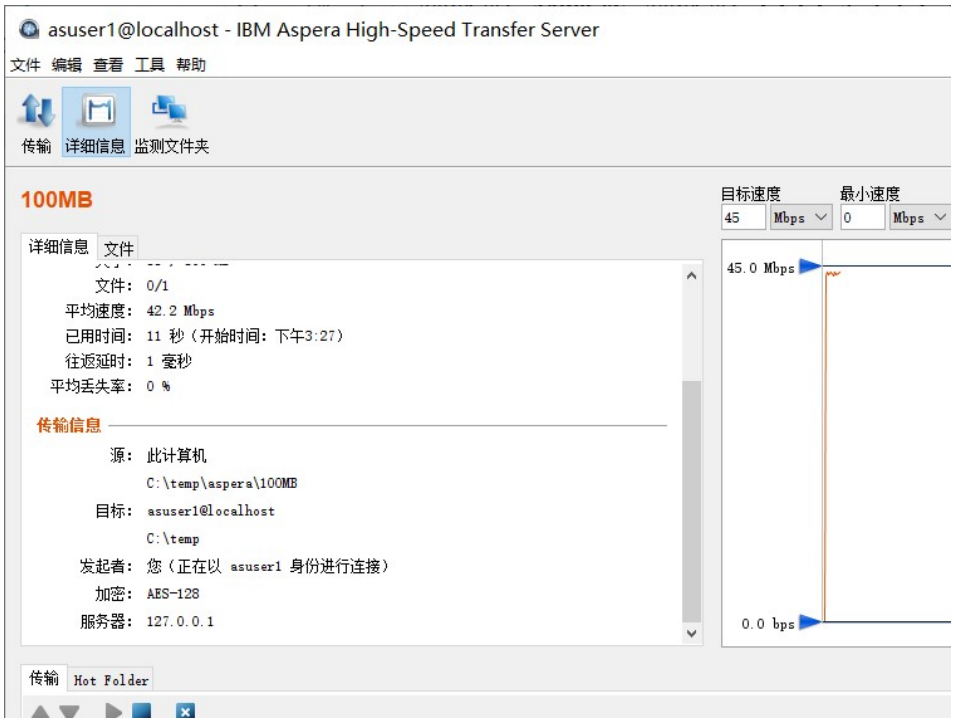


2.4.3 测试发送文件



选择本地的一个文件，选择向右的箭头，进行文件传输

查看实时的传输状态


asuser1@localhost - IBM Aspera High-Speed Transfer Server
文件 编辑 查看 工具 帮助
传输
详细信息
监测文件夹
100MB
目标速度
最小速度
45
Mbps
0
Mbps
详细信息
文件
文件: 0/1
平均速度: 42.2 Mbps
已用时间: 11 秒（开始时间：下午3:27）
往返延时: 1 毫秒
平均丢失率: 0 %
传输信息
源: 此计算机
C:\temp\aspera\100MB
目标: asuser1@localhost
C:\temp
发起者: 您（正在以 asuser1 身份进行连接）
加密: AES-128
服务器: 127.0.0.1
45.0 Mbps
0.0 bps
传输
Hot Folder