**技术总结**

工作期间，参与的项目开发与工程实践以及使用的技术，主要如下所示：

1. LightSpeed/LightSpeed Reflexion/LightSpeed App Builder

新的应用层出不穷，为解决应用程序编译、扫描、打包、测试、部署等上线流程，作为DevOps（Development & Operations，理念：开发运维自动化）的产物LightSpeed应运而出。

LightSpeed将每一次的构建作为Pipeline，打通并串联起不同的平台和基础设施建设：

（1）该项目以Bitbucket作为代码托管平台，配置Webhooks检测代码改动以开启自动构建，主要活动为提交代码、创建新的分支、提交的Pull Request被通过而使得分支合并等；

（2）以Jenkins作为构建工具，遵从“约定大于配置”理念，预设默认的参数供给，同时提供Jenkinsfile作为配置接口，可以自由定制整个构建流水线各个阶段的动作，包括使用不同的流水线模板初始化不同技术栈的应用程序、跳过某些构建阶段、指定需要部署的分支和部署的目标环境、提供不同阶段的生命周期钩子等；

（3）以Docker作为应用容器引擎，通过编写Dockerfile配置应用程序的打包逻辑，包括使用的基础镜像、指定工作目录、依赖的安装、权限的更改、容器启动的入口点等；

（4）以发行于Kubernetes的OpenShift作为容器服务平台，提供Web页面和命令行两种方式用来管理容器，授权登录之后，可查看分配的CPU及内存等资源的使用和限制配额、所有Pod的运行状态、网络资源的路由和服务、构建状态和构建配置等；

（5）以Artifactory作为依赖管理平台，存储Docker Image、NPM Package、Maven JAR等多种类型的依赖程序包；

（6）以UrbanCode Deploy作为应用程序部署平台，将Application作为项目级别，包含不同的环境映射用以指向目标部署的环境，比如：开发环境Dev（Development）、用户可接受测试或验收测试环境UAT（User Acceptance Test）、质量保证环境QA（Quality Assurance）、生产环境PROD（Production）、容灾备份或业务可持续环境COB（Continuity Of Business）等。一般得，不同的应用程序部署需要不同的Application Process，每个应用程序会有Component与之对应，在Component Process里可以指定更具体的部署逻辑。通过资源将组件与实际可访问的应用程序相互链接，所有对象均可以配置属性以提供需要的设置，不同对象但相同的属性读取存在访问优先级。

该项目在提供正向的构建工作流的同时，通过Python Django Web框架实现LightSpeed Reflexion以提供反向的反馈流，用以上传构建日志、更新JIRA Activity、邮件通知构建结果等。

使用LightSpeed，通过编写Jenkinsfile、Dockerfile、Deploy Properties，便可以将整个构建过程自动化；开启对应的参数配置，可以将开发的应用程序部署到目标服务器上，默认为云服务器ECS（Enterprise Container Service, Elastic Compute Service, Elastic Cloud Server）。

但是，配置的过程对用户并非友好。

基于此，LightSpeed App Builder提供应用模板初始化，可快速生成用户期待的应用程序，并完成最终部署等操作。

该项目以Angular为前端UI框架提供操作页面，以Java Spring Boot为后端Web框架提供服务，核心逻辑通过Job Engine和Workflow Engine完成模板初始化。Job Engine与接口交互，Workflow Engine提供三方面的模板生成器：Assemble UI、Cookbook Boot、App Builder等；每一方面包含多个步骤，比如：Create Target Pipeline、Call Assemble UI、Call App Cookbook、Checkout Builder、Build App File、Create App、Push To Pipeline、Notify User等。

本人在LightSpeed项目中，增加了Jenkins Library，完善了构建自动化功能；在LightSpeed Reflexion项目中，参与了性能改进，扩展项目的应用面；在LightSpeed App Builder项目中，参与了后端核心逻辑的开发与程序健壮性的改善。除此之外，为了提高应用程序的响应速度以使用户体验良好和监控应用程序运行的健康程度，在云服务器上搭建了缓存数据库Redis、作为监控系统和时间序列数据库的Prometheus以及数据可视化工具Grafana等，同时搭载OSE Monitoring和ITRS Gateway用以实时告警。

2. Valuation Service for RiskAppsClients in EQRMS

参与股票风险管理系统做估价服务的性能增强和功能优化。

该项目服务于欧美、亚洲等地区的股票、证券等金融业务；采用C# .Net Framework和Windows Server作为软件开发工具；通过C/S架构对接CVD、CBFair、CitiQuote、LAS（Trade Capture/871）、BBG、Borrow Cost等客户，根据请求的Security/FirmAccount/Position/PositionNode、以Grid/DLL为Consumer、以Area51/Khayyam/DataBase为数据源、以Tibco Rendezvoud为消息中间件，进行实时风险计算处理并将结果交由上游或发布到RV Network。

3. Release Integration

在公司内部，现有的应用程序部署平台支持基于操作界面的手动程序提交，每一个项目的上线都对应一个部署程序。经常出现的问题是每周或者每个月的项目交付都需要多个不同的部署程序，都需要由运维团队进行大量的手工提交，这会导致以下的问题：

（1）多任务部署容易出现手动提交或者操作上的失误，经常导致不必要的应用部署事故。

（2）由于新版本的发布和部署大多都安排在周末，这就导致了应用部署排期的延迟，使整个部署推进延迟，给项目交付造成风险。

（3）现有的应用程序部署平台只支持单个应用程序的自动部署，不支持多个应用程序。

（4）部署后的应用程序上线评测和问题修复也需要更多的人工干预。

对此，我们设计并开发了应用程序发布集成系统——Release Integration。

作为系统架构师、前端后端数据库开发者、项目部署与维护者，将其落地并实践。

该系统以React为UI框架，搭配AG Grid/Axios/React Router/Redux/Material UI等组件库，提供前端页面，用以展示上线的部署任务并提供便利地所见即所得地数据操作方式；以Node.js/Express为Web框架，提供后端服务，接收前端请求并作出处理用以数据聚合等；以SQL Server为数据库，提供数据持久化存书。该系统通过将预上线的部署任务打包，在指定时间窗口执行部署作业，从而完成多个应用程序的自动部署工作。

一般地，该系统部署在云服务器上，无需人工干预，在预设的时间会自动执行部署任务的上线作业；同时提供手动触发工作方式，用于超过预设时间而需要上线的紧急的作业等业务需求。

该系统通过以下几点提供解决方案：

（1）所有每周或者每个月计划上线的部署应用程序直接预先定义好，由预先定义好的自动化程序一次性全部提交，每一个要计划的部署应用程序会按照预设的时间进行自动部署，规避了手动提交而引起的潜在手工操作风险。

（2）通常发布新版的上线是有时间限制的，程序的上线部署和检验测试一般要在规定的时间内完成。Release Integration 也规避了程序部署被延迟或者延期的风险。

（3）发布新版本后的检验可以通过自动化邮件及时更新状态。

（4）部署问题的核查和项目发布的回滚速度依赖于运维人员的响应时间和当前工作量，多任务自动化的部署也解决了运维人员的工作量瓶颈。