设计研究与应用

基于 DevOps 的软件开发管理模式

耿泉峰[1]，李 曦[2]，葛 维[3]，葛云龙[3]，卢潇潇[1]

（1. 国网河北省电力有限公司电力科学研究院，河北 050021；2. 国网汇通金财（北京）信息科技有限公司，北京 100053；3. 国网河北省电力有限公司，河北 050022）

摘 要： 文章简述了 DevOps 概念、基于 DevOps 的软件开发流程，并以 D 平台为例对开发流程的每一步给出了示例。DevOps 本质上是将 IT 开发与运维合并，借助强大的管理工具，引入容器化技术使当前系统环境部署过程由 IAAS 转向 PAAS，用以规范系统间调用关系，并基于 PAAS 基础之上整改配置项及数据，使其与系统服务分离，实现灵活配置，快速实施。DevOps 是软件开发管理未来的发展趋势。

关键词： DevOps 开发与运维；DevOps 软件开发流程

中图分类号： TP391. 41 **文献标识码：** B **DOI：** 10.3969/j.issn.1003-6970.2019.01.020

本文著录格式： 耿泉峰，李曦，葛维，等. 基于 DevOps 的软件开发管理模式[J]. 软件，2019，40（1）：93–96

Software Development Management Mode Based on DevOps

GENG Quan-feng[1], LI Xi[2], GE Wei[3], GE Yun-long[3], LU Xiao-xiao[1]

(1. *State Grid Hebei Electric Power Co., Ltd. Institute of Electric Power Science, Hebei* 050021, *China*; 2. *Beijing huitong jincai information technology co., LTD., Beijing 100053, China*; 3. *State Grid Hebei Electric Power Co., Ltd., Hebei* 050022, *China*)

【Abstract】: This paper briefly introduces the concept of DevOps and the software development process based on DevOps, and gives an example of each step of the development process with the D platform as an example. DevOps is essentially a combination of IT development and maintenance. With the help of powerful management tools and container technology, the current system environment deployment process is changed from IAAS to PAAS to standardize inter-system call relations. Based on PAAS, the configuration items and data are adjusted to separate them from system services, achieve flexible configuration and fast implementation. Shi. DevOps is the future trend of software development management.

【Key words】: DevOps development; Operation and maintenance DevOps software development process

0 引言

DevOps（Development 和 Operations 的组合词）是一套完整的面向 IT 运维的工作流，[1]以 IT 自动化以及持续集成（CI）、持续部署（CD）为基础，来优化程式开发、测试、系统运维等所有环节。

它重视“软件开发人员”和“IT 运维人员”间的协作。基于自动的持续的软件交付流程，使编译，打包，发布，测试等行为能够更高效，更稳定。

敏捷迭代的概念愈发明显，为支持敏捷二字，传统的软件发布模式已经感到乏力，因此 DevOps 的概念应运而生。图 1 为 DevOps 的概念图。

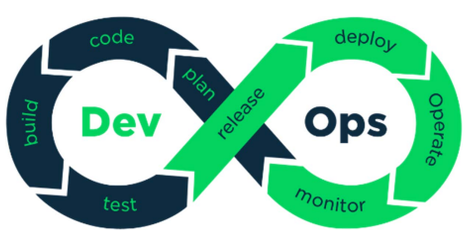

图 1 DevOps 的概念图
Fig.1 Concept Map of DevOps

DevOps 串联的是技术与人的问题。它包含许多技术方案，比如像持续集成这类概念已深入人心，持续集成要做好，不但需要持续集成服务器及配套的自动化集成和测试程序，还需要与版本控制紧密

作者简介： 李曦(1987–)，男，配置管理工程师，多年从事持续集成以及配置管理工作；耿泉峰(1986–)，男，国网河北省电力有限公司电力科学研究院，工程师，主要从事电力营销信息技术研究与应用；葛维(1981–)，女，国网河北省电力有限公司，工程师，主要从事电力营销信息化建设与运行维护、信息系统深化应用等工作；葛云龙(1983–)，男，国网河北省电力有限公司，工程师，主要从事电力信息化与网络安全工作；卢潇潇(1987–)，男，国网河北省电力有限公司电力科学研究院，工程师，主要从事电力营销信息技术研究与应用。

结合。除此之外，还需要注重各岗位之间协作的关系。把开发与运维相融合，这是很重要的思维转变。

DevOps 最主要的优势是可以持续的高效的进行交付，这也正是这个概念兴起的原因。使用 DevOps 的高效 IT 公司平均每年可以完成 1460 次部署。与其他未使用 DevOps 的 IT 公司相比，前者的部署频率为后者的 200 倍，前者投产速度比后者快 2555 倍，前者故障恢复速度比后者快 24 倍。在开发及运维工作分配过程中，后者要多花 22%的时间用在为规划好或者重复工作上，而前者却可以多花 29%的时间用在其他类型工作上面。可以看出利用 Devops 不紧提高了产品产出效率，也提高了员工的工作质量。

DevOps 还有一个优势就是可以使每一个员工都理解和参与整体的产品生产过程，提高了员工的满足和成就感，使员工产出更为高效。

可能你会认为快速部署和提高投产质量是自相矛盾的。其实不然，快速部署可以使投产过程中的问题今早暴露，产品可以更快交付，更快得到用户反馈，从而更快的进行响应优化。“小步快跑”是 DevOps 的形式，每次集成带来的变化是比较小的，出现问题偏差较小，修复优化也相对容易。

DevOps 是开发人员，运维人员和质量人员之间沟通协作的“桥梁”。将传统产品发布模式及运维模式进行改变，提高效率，降低成本。

DevOps 目前处于急速增长状态，在大型企业中尤为明显，经过调查，DevOps 在企业中的接受度大幅提高。%74 的受访者接受并认可了 DevOps。目前，88%的大型企业开始接受并使用 DevOps，中小型企业的使用占比也达到了 75%。

目前采用 DevOps 的公司有很多，比如：Walmart、Sony、Adobe、Amazon、Apple、Facebook、LinkedIn、Airbnb、Ebay、Etsy、NASA、Starbucks、Netflix、Target，等。

大型企业正在自下而上接受 DevOps，其中业务部门（35%）以及项目和团队（32%）已经实施 DevOps。不过，只有 22%的大企业在整个公司范围内使用了 DevOps[2]。

实用工具方面，DevOps 工具的用量大幅。Puppet 和 Chef 是最常用的 DevOps 工具，使用率均为 35%。Docker 增长量也非常亏，使用率翻倍。Ansible 的用量也显著增加，使用率从 15%增长至 31%。

1　DevOps 流程

DevOps 的流程主要划分为以下几个部分：持续的管理计划，持续的集成与测试，持续的交付于部署，持续的监控与运维，持续的分析与计划[3]。

DevOps 流程中涉及的角色为：产品业务人员，开发人员，测试人员，运维人员，项目管理人员（Scrum Master）

1.1　持续的管理与计划

业务产品人员制定功能并于开发，测试与运维人员一起制定产品交付计划[4]。每一个角色根据自身工作内容做出建议。项目管理人员在这个阶段制定进度，并在流程每一个节点进行状态跟踪。整体计划及状态应该在项目管理平台（比如 JIRA）中对所有人为可见并实时更新状态。

1.2　持续集成与测试

开发人员在开发业务功能同时需同时编写针对功能的单元测试代码，运维人员在编译功能代码的同时编译并运行单元测试模块，并针对结果进行判断是否需要修改或向测试人员流转。测试人员依据功能描述和接口文档编写接口测试用例，运维人员在编译部署通过后集成接口测试并反馈测试报告。整个过程的状态应该是对所有参与人员可见的。

1.3　持续交付与部署

针对代码开发实施每日构建和集成，可以使用集成工具如 jenkins 进行定时构建并将构建产物部署至仿真环境，将部署结果告知测试人员进行验证。图 3 为 jenkins 定时示例。

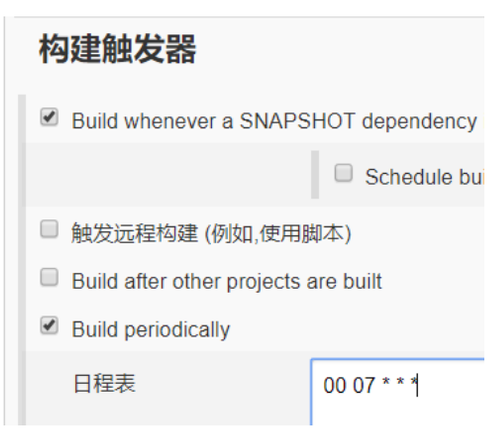

图 2　jenkins 定时示例
Fig.2　Jenkins timing example

1.4　持续的分析与计划

针对项目管理平台（JIRA）的每项工作状态进行分析，发现突破及瓶颈，并针对性的对计划进行变更。加入每天站立会进行整体进度汇总报告并吸收参与人员的建议。

2 DevOps 常用工具

以下是以某电子系统的 D 运维自动化管理平台（以下简称”D 平台”）为例的 DevOps 常用工具。D 平台的 DevOps 工具中包含了项目管理平台 JIRA、开发工具 GIT、持续集成与测试、持续交付与部署、持续监控与运维这几个部分。

2.1 项目管理

JIRA[5]是基于 JAVA 开发的项目缺陷跟踪管理平台，由 Atlassian 公司开发，JIRA 在项目管理系统中功能和稳定性一向比较出色，而且易用性也比较好。同时用户购买 JIRA 系统的同时，也获得了其源代码，可以做二次开发。JIRA 功能强大，界面简洁，配置灵活，可扩展性强。

2.2 开发工具

GIT 是一套分布式代码管理系统，可以高效的处理代码版本问题。主要特性表现为远程分支和本地分支的结合使用来解决多功能并行开发的需求。

2.3 持续集成与测试

Maven[6]是管理开发代码间依赖的利器。Maven 对开发人员提供了一套完整的生命周期框架。Maven 可以集成所有开发过程，包括编译、打包、测试、发布、归档等操作。Maven 的原则是约定大于配置，可以让开发、测试、运维人员专注于岗位工作，而不用浪费时间在各节点衔接的约定配置上面。

CsperJS:CsperJS[7]是基于 PhantomJS 编写的脚本处理和测试工具，提供了完成常见场景的测试方法。D 平台使用 CasperJS 编写平台测试案例，并与 JENKINS 集成形成自动化测试体系。

Jenkins: Jenkins[8]是基于 JAVA 开发的持续集成平台，并且已开源，用于解决重复的工作，比如编译、冒烟测试、发布等操作。Jenkins 扩展性极强，可以与大部分框架集成，包括 Maven, kubernetes, CsperJS 等，在大多数企业中被广泛使用。

2.4 持续交付与部署

kubernetes，简称 K8s，是用 8 代替 8 个字符“ubernete”而成的缩写。是一个开源的，用于管理云平台中多个主机上的容器化的应用，Kubernetes 的目标是让部署容器化的应用简单并且高效（powerful），Kubernetes 提供了应用部署，规划，更新，维护的一种机制。

传统的应用部署方式是通过插件或脚本来安装应用。这样做的缺点是应用的运行、配置、管理、所有生存周期将与当前操作系统绑定，这样做并不利于应用的升级更新/回滚等操作，当然也可以通过创建虚拟机的方式来实现某些功能，但是虚拟机非常重，并不利于可移植性。

新的方式是通过部署容器方式实现，每个容器之间互相隔离，每个容器有自己的文件系统，容器之间进程不会相互影响，能区分计算资源。相对于虚拟机，容器能快速部署，由于容器与底层设施、机器文件系统解耦的，所以它能在不同云、不同版本操作系统间进行迁移。

容器占用资源少、部署快，每个应用可以被打包成一个容器镜像，每个应用与容器间成一对一关系也使容器有更大优势，使用容器可以在 build 或 release 的阶段，为应用创建容器镜像，因为每个应用不需要与其余的应用堆栈组合，也不依赖于生产环境基础结构，这使得从研发到测试、生产能提供一致环境[9]。类似地，容器比虚机轻量、更“透明”，这更便于监控和管理。图 4 为 kubernetes 架构。

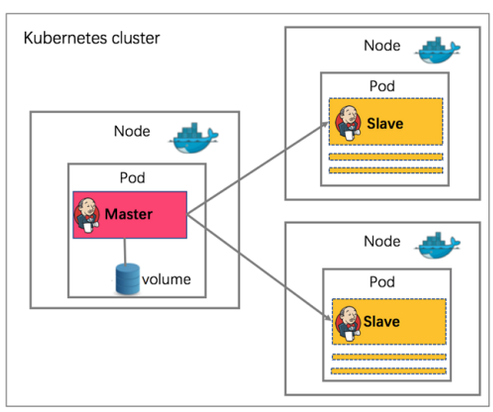

图 3 kubernetes 架构
Fig.3 Kubernetes Framework

2.5 持续监控与运维

Zabbix[10]：Zabbix 是基于 WEB 界面提供分布式系统及网络监视功能的开源框架。

3 DevOps 实例

D 运维自动化管理平台 DevOps 开发实例。

3.1 制定计划

由业务产品人员制定 D 平台本年度的 b 需求点并添加至 JIRA 平台。项目管理人员在计划阶段确定大概的需求及完成时间，在 JIRA 平台中确定 Sprint。每个项目团队的成员在 Sprint 下建立相应的工作任务，使 JIRA 平台 sprint 状态中任务情况一目了然。项目管理人员跟踪 Sprint 及 Sprint 下任务的分配情况。

3.2 定义工作内容

业务产品人员、开发人员、测试人员、运维人员讨论需求细节。业务产品人员完成设计图，测试人员编写测试用例，开发人员进行详细设计，运维人员制定上线后负载及监控配置。

相关的文档等放在 GIT 中进行管理。

根据讨论情况，业务产品人员在 JIRA 中更新需求，开发人员、测试人员、运维人员也需要在 JIRA 中去更新自己的任务及完成状态。

项目管理人员在全程跟进 JIRA 中的更新情况并提出相应的分配建议。

3.3 开发

开发人员运用 Maven 来管理代码的生命周期。所有生命周期定义在 POM.XML 文件中。如下示例：

```
<dependencies>
    <dependency>
        <groupId>D-SYS</groupId>
        <artifactId>TEST</artifactId>
    <version>1.0.0-SNAPSHOT</version>
        </dependency>
    </dependencis>
```

开发人员在 GIT 主干中提交自己的模块代码，Jenkins 中的触发构建保证开发环境的代码不断被提交到测试环境，进行严格的自动化测试并反馈测试结果。

开发人员在完成自己的开发任务后将会在 JIRA 工具中更新自己的任务状态。

3.4 构建——部署——测试

Jenkins 定期从 GIT 目录取得最新的代码进行构建，保证开发人员的代码持续被发布至测试环境中进行自动化测试。

测试人员会提前在 Jenkins 中新建测试模块，放入已编写好的 CasperJS 测试脚本。以下是一个 CasperJS 测试案例示例：

```
// The suite callback will get the current Tester instance as its first argument:
casper.test.begin("EightCasein", 1, function(test){
// this._result_cap_str = "../../TestCapture/Test_SixCase/";
// this.logLevel = 'debug';
// 从这里开始
  casper.start("http://some.url.other.com/");
// 截屏文件保存路径
  casper._result_cap_str = "../../TestCapture/Test_EightinCase/";
// 全局变量
  casper._inlandORinter = '';
  casper.then(_CaseOption2).then(function(){_Delete_LS.call(this)});
  casper.then(function(){_TEST_T.call(this, 'BJS', 'CTU', '2015-02-23')});
  casper.then(GoTo_WoXie).then(LOGININ3);
  casper.then(D_A_T_Flt_Check2).then(function(){
    Choose_O_Flt2.call(this, 4, 1)
  });
  casper.then(in_Check_BillEdit_Elements).then(in_Fill_Passenger).then(Fill_Other_Info).then(Bill_Fill_Edit_next2);
  casper.then(function(){in_Pay_with_Money3.call(this, "Credit")}).then(_Result_Shoot2);
  casper.run(function (){
    this.echo("No Money to Pay, so Stop here. Bye~");
    test.assertResourceExists('html',"ok?\n");
    test.done();
    casper.exit(0);
  });
});
```

此段脚本模拟用户登录，基于 JENKINS 调用并返回结果。减少人工测试环节，且结果稳定。

3.5 部署——测试——发布

当测试环境的版本测试通过以后，运维人员自动化部署该版本至生产环境。

Jenkins 中配置的构建任务将自动执行测试脚本，若测试通过，则发送邮件宣布该版本投产。

3.6 D 平台运维

D 平台后台通过 Zabbix 监控。监控结果会实施更新并将问题通知至运维人员。

4 结语

DevOps 的目的是解决敏捷交付的一系列问题，实现了开发环境、测试环境和生产环境的一致性、灵动性、稳定性，以自动化的全生命周期配置维护管理，从而适应现代 Web 环境的高变动性、高可用性、高可靠性。

参考文献

[1] 荣国平, 张贺, 邵栋. DevOps: 原理、方法与实践[M]. 北京: 机械工业出版社, 2017.
[2] 兰洋. 持续集成实践[M]. 北京: 电子工业出版社, 2015.
[3] 刘建伟, 王育民. 网络安全技术与实践[M]. 北京: 清华大学出版社, 2017.
[4] 王海鹏. 持续集成[M]. 北京: 机械工业出版社, 2008.
[5] 刘淼. JIRA实践基础[M]. 北京: 清华大学出版社, 2015.
[6] 许晓斌. MAVEN实战[M]. 北京: 机械工业出版社, 2011.
[7] 邹辉. 软件自动化测试开发[M]. 北京: 电子工业出版社, 2017.
[8] 郝树伟. Jenkins权威指南[M]. 北京: 电子工业出版集团, 2016.
[9] 龚正. Kubernetes权威指南[M]. 北京: 中国工信出版集团, 2017.
[10] 吴兆松. Zabbix企业级分布式监控系统[M]. 北京: 机械工业出版社, 2012.