docs Documentation

发布 *Beta0.1.0*

hanzhilong

2020 年 01 月 08 日

目录:

CHAPTER 1

狭义运维论-传统产品运维

CHAPTER 2

广义运维论-平台运维

2.1 广义运维定义

广义运维相对于狭义运维（或称之为传统运维），从涉及的产品线广度及接触的人群及运维职责上有很大区别。最本质的区别是广义运维是对平台进行全生命周期内的运维，重点在于全局质量把控。

2.1.1 主要职责

1. 服务器硬件维护
2. 网络硬件维护
3. 平台部署及代码更新
4. 监测监控数据接入
5. gis3.2 后台维护

2.1.2 接触人群

1. 集团公司各科室
2. 矿端各科室
3. 开发人员

4. gis3.2 研究院
5. 项目经理

2.1.3 职责

硬件

1. 服务器及网络硬件维护

平台

1. 平台前台 web 代码部署
2. 平台前台 web 代码后续更新
3. 平台菜单及用户权限管理
4. 平台问题反馈及修改
5. 平台功能测试
6. 编写平台培训教程

监测监控

1. 监测监控数据接入
2. 监测监控数据检查

后台数据库

1. 数据库后台部署（主库和监测节点库）
2. 所有数据库备份

gis3.2 及一张图

1. gis3.2 一张图用户权限管理
2. gis3.2 一张图上传维护
3. gis3.2 问题反馈给研究院

2.2 平台运维新模式

2.2.1 分工合作

由于平台的复杂性，一个人不可能把系统所有菜单所有细节都完全掌握。所以进行分工合作。新模式注重的是和客户之间的合作、团队内部（运维及开发）的合作、与项目经理的合作。合作的前提是有共同的目标：把平台共同维护好。

2.2.2 合作的基本原则

合作的基本原则是：让专业的人员处理专业的事情。

2.2.3 运维连接器

运维的主要作用：连接器。作为一个胶水职位，对个人知识面的广度有很高要求，但不需要深入了解，只需要有开发者的技术思维即可。需要把问题准确反馈给开发人员和项目经理。同时兼具一部分的修改工作："80% 的简单问题由客户内部处理，10% 的问题由运维处理，5% 的问题由开发人员处理，无法处理的上报给项目经理进行开会讨论"。通过一级一级的筛选和进行问题的处理，把开发人员与客户进行隔离，客户的问题首先经过运维人员的求证和梳理进入到任务调度系统，再由开发人员技术主管分配给各开发人员每周的任务。后续会详细讲解工作流的改进细节。

2.3 服务器管理要求

主要从硬件及软件管理两方面进行管理。

2.3.1 服务器硬件及网络

1. 机房一般由客户管理。
2. 一般集团公司和矿业公司和煤矿都有内网跳板机，跳板机一般设置在集团公司或矿业公司。远程登录时所有开发人员若从外网访问时一般用 vpn 连接内网，建议统一使用跳板机。不过跳板机不利于后期多人登录调试。
3. 网络配置及硬件防火墙也是由用户配置。

2.3.2 密码管理

所有密码只能是开发人员及运维人员内部知晓，登录后不要默认保存密码。操作完要及时退出。

1. 跳板机远程登录密码，一般不修改。
2. 集团主机和矿端主机登录密码，一般不修改。

3. 数据库密码，一般不修改。
4. 平台客户密码，只重置。
5. gis 后台密码，查询数据库查看密码。

2.3.3 杀毒软件管理

需要将龙软的软件添加为白名单。

1. 360 安全卫士和 360 杀毒软件（多用安全卫士扫漏洞，先修补高危漏洞，再修补普通漏洞）
2. 火绒杀毒软件（病毒库升级频率至少一周一次）
3. 卡巴斯基
4. 诺顿

2.4 工作流要求

主要从工作流的分工及操作方法来讲日常的工作流程。

2.4.1 工作流概述

1. 需求分析
2. 任务调度
3. 任务处理
4. 测试
5. 发布

2.4.2 需求分析

只处理平台旧功能修改、bug 修复、新功能添加。若是操作和配置的问题，则不要加到需求分析里，那些属于使用操作不当造成的，不应该提交给开发人员。

1. 旧功能修改（已存在的菜单的功能调整）。
2. bug 修复（平台的框架 bug 和各模块菜单 bug）。
3. 新功能添加（合同和技术协议里没有的功能，若是新增模块则需要重新签订合同，旧模块改动超过 50% 也视为新增模块）。

2.4.3 任务调度

把需求提交到统一的平台如禅道，由开发人员技术主管分配任务。技术主管先检查一遍新需求，剔除操作不当和配置的需求，只保留需要开发人员处理的需求。保证每周的任务能够排满，并预留 20% 的时间预防突发情况和让开发人员进行总结提升。根据重要程度，任务优先级可以这样配置:

1. 集团平台新任务（紧急且重要）。
2. 突发事件（紧急但不重要）。
3. 矿端平台默认新任务（紧急但不重要）。
4. 无关痛痒的新任务（不紧急也不重要）。

2.4.4 任务处理

需求经开发主管设置截止时间分配任务后，会邮件通知各开发人员，各开发人员开始处理。将需求的处理状态分为以下几类，开发人员每周五下班前要及时更新任务状态。

1. 准备（任务已检查过，但未开始执行）。
2. 执行中（任务已分配给开发人员进行修改）。
3. 已处理（任务已由开发人员修改完毕，并经过运维的确认）。
4. 暂停（由于时间安排或需求不明确，暂停中的任务）。
5. 延期（由于时间安排，无限期暂停，最后很大几率是不处理）。
6. 不处理（运维人员和客户沟通后确认不再处理）。

2.4.5 测试

开发人员处理好的任务，邮件通知运维人员。开发人员先进行单元测试，运维人员再进行集成测试。运维人员根据任务需求描述进行确认，若未达到需求描述，则反馈给开发人员重新修改。直到修改完成或暂停处理。

1. 搭建测试环境（部署新的测试环境或者使用指定的测试服务器）。
2. 检查后台数据库写入情况（查看数据是否写入数据库）。
3. 检查平台前台展示情况（查看平台展示数据是否正确）。

2.4.6 发布

其实这里指打包和部署。经过测试的代码部署到生产环境服务器。

1. 开发人员提交代码到 svn 服务器。
2. 运维人员获取 svn 服务器的最新源码。
3. 打包（使用自动化构建工具生成升级包）。

4. 部署（将升级包推送到生产环境服务器进行代码更新）。

CHAPTER 3

资源管理-运维大数据平台

3.1 版本管理

版本控制不是狭义上指开发人员提交 svn 后的版本信息，而是基于基础设施即代码的思想形成的全套版本管理。包括：使用统一的代码库（将所有源文件、配置文件纳入版本控制系统）、代码的依赖项（库、静态内容）、编译器、创建数据库的脚本、搭建生产环境的工具、自动化测试脚本、源码打包和部署脚本、项目文件（需求文档、部署过程、发布说明）、网络防火墙及 ip 配置。这样把构建过程中所依赖的一切都纳入版本控制系统，这样的好处是开发人员无论是在任何时间节点都能在统一的环境下进行开发和测试工作，保证了开发环境的统一和代码部署出问题后能及时进行回滚操作，而且加强对其它底层软件版本进行控制，也会推进运维工作的标准化。

3.1.1 1. 服务器操作系统

win server 2008 r2 x64 或 win server 2012 r2 x64

3.1.2 2. 浏览器

推荐使用 Internet Explorer 11(IE11) 简体中文版或 google chrome 79 版及以后；若无法下载 chrome，则可以使用 360 浏览器 10 版本及以后，开启极速模式。

3.1.3 3. 制图软件

由于制图软件多为离线安装，更新版本较为麻烦，而且更新时费时费力，一个地区或一个单位的制图软件尽量保持版本一致，升级周期长，可一年升级一次，升级前需要做好数据库的备份工作，并在升级前做好功能测试，尽量使用比较新但稳定的版本。由于版本更新时旧版软件必须完全卸载干净，所以请只开启地图服务能浏览但不能编辑，等所有人都确认升级了本机制图软件后，再开启协同服务进行修改，数据库要一天一备份，尽量将版本更新所带来的影响降到最低。

3.1.4 4.svn 版本库

开发人员的 svn 版本尽量做到一天同步一次到生产环境，可以使用自动化构建并部署，svn 版本可实时更新到测试环境下，尽早发现可能存在的问题。

3.1.5 5. 开发人员的 vs 版本

开发人员的 vs 版本尽量保持一致，如 vs2013 及以下。这样做的好处是不会出现使用高版本 vs 编写的带有新特性的代码在低版本 vs 中无法编译通过的情况，保持 vs 版本一致性，就保证了编译环境的一致，构建和打包时也使用统一的 vs 版本进行编译。

3.1.6 6. 打包工具

持续部署和持续集成可以使用自动化工具，如 Jenkins，对构建工具进行版本控制，可对其插件的依赖关系进行管理。源码打包和部署脚本也要有版本号，可以是日期形式的，但不能没有版本信息。

3.1.7 7. 项目文件

需求文档、部署过程、发布说明也需要有版本号。

- 需求文档：在禅道上提交后，若有了内修改，则需要创新新的版本号。
- 部署过程：不同平台的部署方式不一样，对于部署过程也要有相应的说明。
- 发布说明：目前只有开发人员进行 svn 提交时会有备注。

3.1.8 8. 服务器 ip 配置

由运维人员进行服务器 ip 的配置管理，形成统一的表格，一个地区或项目的服务器 ip、用户名、密码要实时更新到该表，并进行更新日期标注。这样同一个项目组人员可共享该表格。

3.1.9 9. 监测接口程序和数据库

监测监控类的接口程序需和数据库保持一致，空数据库要和新的接口程序打包在一起。这样做的好处是实现同步升级更新，而且若数据库出了问题，可以使用旧版的数据库进行恢复。

CHAPTER 4

产品经理-现场事务

CHAPTER 5

开发人员的日常-防脱发指南

5.1 DevOps 导论

都是干货，DevOps 工程师必看。

视频介绍

B 站在线观看视频

5.1.1 软件开发的本质困难

- 复杂性
- 不可见性
- 可变性
- 一致性

5.1.2 网络化和服务化

软件应用特征

功能更复杂，规模更大用户数量急剧增加快速演化和需求不确定分发方式的变化（SaaS）

5.1.3 迭代式软件开发方式

迭代式：大型软件系统的开发过程也是一个逐步学习和交流的过程，软件系统的交付不是一次完成，而是通过多个迭代周期，逐步来完成交付。

5.1.4 DevOps

概念：开发运维一体化

5.1.5 PSP 质量策略

用缺陷管理来替代质量管理高质量产品也就意味着要求组成软件产品的各个组件基本无缺陷

CHAPTER 6

技术中台-自动化运维

6.1 持续部署理论

以前写前端项目打包部署，都是手动运行命令，打包完，然后压缩，再上传到服务器解压。这种方式确实有点 low 并且效率也不高。自从用了 Jenkins 持续集成工具，写前端项目越来越工程化，再也不用担心忘记部署项目，也不用烦躁每次打包压缩后还要部署多个服务器和环境，更开心的是每次家里写完代码，不用远程公司部署项目，提交代码后自动会为你部署。本文基于.NET4.0 的 web 项目和 SVN 的代码仓库以及 Windows(其他系统平台大同小异)，简述 Jenkins 实现自动部署的配置。

6.1.1 名词解释

持续集成

持续集成 (Continous Intergration): 一个大项目是由多个模块组成的，每一个模块都有具体的小组负责开发，但有时候本模块独立测试正常，但与其他模块一起集成测试就会出问题。需要经常把所有模块集成在一起进行测试，尽早发现问题。关注点在于尽早发现项目整体运行问题，尽早解决。在持续集成方面，我们选择 Jenkins。Jenkins 是一款开源软件，拥有众多优秀的插件，依靠这些插件，我们可以完成一些周期、繁琐、复杂的任务。例如我们今天分享的持续发布，虽然 Jenkins 解决了我们繁琐复杂周期性的操作，但是没有解决我们在多种环境下编译构建的需求。而这个场景正是 Docker 的强项。通过 Jenkins 的 pipeline 我们可以实现代码检出、单元测试、编译、构建、发布、测试等流程的自动化，而最终通过 Jenkins 的 Docker 插件将产出物构建成镜像，方便部署到 Docker 环境。

持续部署

部署（deployment）还是发布（release)？部署一般指把应用或者服务“安装”到目标环境（开发、测试或者生产）中，而发布则应指把应用或者服务交付给最终用户使用。尽管这两个动作（尤其是在生产环境中）经常是同时发生的，但它们理应是两个完全不同的阶段。实际上一个好的持续交付流程恰恰应该把“部署”和“发布”解耦，变成两个可以独立控制的阶段。部署的内容包括什么？无论是增量部署还是全量部署，都需要关注其部署的内容是什么，尤其是在广泛讨论微服务的今天。如果从部署角度看，我们把任何可以独立部署的内容称为一个“部署单元”。一个部署单元可以是一个模块，几个模块的联合体或者一个完整的应用，而如何划分则要视具体场景来定。一般来说，划分部署单元的最佳实践为一个可以独立演化、部署且和应用其他部分松耦合的集合。全量部署（full)：全部文件重新拷贝并覆盖。优点稳定性好，但对带宽的要求大，更新时间长。增量部署（min)：更新上个版本与最新版本之间的文件。优点速度快，对带宽的要求小，更新时间短，但若更新失败，则需要全量更新覆盖一次。半增量部署（semi-increment)：只更新近期有更改的文件。保存一个时间范围内的更新文件，最长一个月，集合了全量部署与增量部署的优点。压缩包小，对带宽的要求小，更新时间短，容错率高，可以实现一个月内的增量更新。持续集成让我们新的代码源源不断的构建成了可发布的工程，这些工程经历了单元测试，自动化测试，但还没有接受过测试团队的严格测试。Jenkins 是一个强大的持续集成工具，然而持续部署并不是 Jenkins 的强项，但是 Jenkins 拥有很多强大的插件。

持续交付

持续交付 (Continous Delievery): 用小版本不断进行快速迭代，不断收集用户反馈信息，用最快的速度改进优化。关注点在于研发团队的最新代码能够尽快让最终用户体验到。

作者：张熙链接：https://www.jianshu.com/p/7e1e8d8e8ec5 来源：简书

1、传统我们的项目开发模式是产品调研提出需求，开发团队研究决定开发方案选型。然后开始一个周期的开发，模块开发完成之后开始模块间的联调。联调结束之后打包交付给测试团队。测试团队，系统测试或自动化测试，然后提交 bug，开发团队修复 bug，周而复始。

2、传统的模式中，存在着较多的不确定因素。例如，开发环境、编译环境、测试环境、生产环境，等不确定因素。人为介入打包中的不确定因素，缺乏单元测试和自动化测试的整合。从而导致的结果是，开发-测试-修复的周期较长，而且很多小的问题完全可以由单元测试进行覆盖。

持续交付并不是某个特定的软件，而是一个结果。这个结果要求团队可以随时的发布一个新的准确版本，而且要求在编译发布的过程中进行自动化测试，通过自动化测试可以及时的发现并定位存在的 bug，修复 bug 之后再进行快速的发布到测试环境，测试团队直接进行测试。与传统模式的区别在于持续交付可以提前发现 bug 的存在和快速修复而不必等到测试人员的介入之后才发现。持续交付分解出来就是“持续”和“交付”。 持续: 持续要求任何时，候任何情况都能进行准确的发布，做到准确的发布需要注意以下几个关键点。

持续应该是一个周期性的，可以是每天的某个时间点，也可以是某次代码的提交，或者某次人为触发。所以人工进行构建是不可能的，需要自动化的构建，自动化要求构建的任何一个流程都必须以脚本的形式运行，代码检出、代码构建、各模块代码单元测试、集成测试、UI 自动化测试等。

发布的程序版本不允许是各个模块在开发环境编译出一个版本作为交付，而要求在一个纯净的编译环境中进行构建。

构建的过程应该要求最大可能的固化，例如操作系统的版本，构建环境的版本，相关的依赖等。

避免从网络获取相关的文件，这点以 nodejs 为开发或编译的项目尤其重要，安装 node 的依赖包总是一个漫长的过程，就算有国内的源，一般的项目也需要一两分钟的 node 依赖包，这不符合快速构建。

交付: 在持续编译的过程，使用自动化已经可以避免大多数的错误了。但是还是需要人为介入的系统测试，毕竟自动化的测试一般只能覆盖到 70% 左右。

根据我们团队内部推广这种工作方式的效果来看，持续集成确实让我们工作便利了许多，每次代码的构建和自动化测试让我们及时发现存在的 bug。好的工作模式也需要团队成员的遵守，团队成员应该积极的拥抱这种工作方式。

自动化优点

1. 满足大型项目的需求：有时候开发人员在更新代码时在本地测试是正常的，但放在服务器上运行就会出问题。只有在生产环境下能正常跑起来的代码才算合格，关注点在于项目功能部署至服务器后可以运行。
2. 节约人力成本：开发人员以前每次都是手动复制文件到所有服务器进行更新替换操作，而且每次更新都要需要多次登录多台服务器，容易产生厌倦感。
3. 统一的版本控制：定期执行部署脚本，实现了单一源码统一部署，保证所实施的单位都是同一套代码的统一的版本。而且集成了自动为 js 文件和 css 文件创建 hash 值的功能，每次部署后都能自动更新 cshtml 文件引用 js 文件的版本信息，从而满足不清理浏览器缓存也能更新 js 文件的目的。
4. 部署速度快：每次仅对增量部分进行更新，无论是文件分发还是配置更新的内容都会更少，部署需要的时间也就相对较短。
5. 高安全性：部署时生成的更新包每次只会集成增量更新的文件，不会直接暴露系统的整套代码，避免代码泄露的风险。
6. 高稳定性：由于是使用脚本进行操作，脚本能自行判断是否需要更新，更新时不会重复执行更新脚本；而且使用脚本进行文件替换，操作快速而准确，避免了手工误覆盖造成的平台不稳定。部署时是并发部署，互不影响。
7. 代码检查：若代码有问题，则编译不成功，不会进行推送。增加了手动执行脚本回滚到上一个版本的功能。
8. 集成度高：使用统一的管理平台，把所有操作都写成一套脚本，方便修改。操作都封装好了，不熟悉的人只要按操作步骤执行构建命令即可自动进行部署。
9. 集成邮件通知：源码获取、编译、打包与部署分离；有完善的日志系统，可自动将每次编译打包的日志分发给提交代码的开发人员，并将简要的部署结果按事先编制的模板写入邮件正文；这样由于源码打包和部署是分离的，就明确了开发人员与运维人员的责任划分，开发人员只要关注是否能通过编译即可，运维人员在部署失败后会收到邮件，进行测试，并排查问题。

自动化缺点

1. 第一次需要手动先同步更新一次：无法完成初始化的部署操作，若已部署过的系统，但不知道部署时的版本号，只能先进行一次全量部署，而且若因为网络原因导致部署失败，则没有提示，需要人工操作进行判断。
2. 代码检查：无法满足所有的使用需求，因为有些代码虽然符合语法规范，可以正常编译，但里面的逻辑问题无法检测出来。所以增加了手动回滚到上一个版本的功能。

6.2 Jenkins 部署-概要设计

6.2.1 引言

编写目的

- 本说明书目的在于明确说明 Jenkins 部署的实现方式，指导运维人员进行自动化部署。
- 本说明书的预期读者为：系统运维人员、系统开发人员。

背景

- 待开发软件系统的名称：XX 安全生产共享平台 Jenkins 部署工程
- 此软件系统任务提出者：智慧能源事业部徐州工程三部运维小组
- 此软件系统任务开发者：智慧能源事业部徐州工程三部运维小组
- 此软件系统任务用户：徐州工程三部运维小组成员、徐州工程三部开发小组成员

1. 开发人员少且项目工期紧：开发人员时间、地点不固定，每天还要进行旧模块的修改和新功能的提交，每个开发人员的时间都很宝贵。
2. 项目后期维护人员少但项目多：运维人员每天除了要处理现场工作，还要协助开发人员调试新功能，人员相对较少，但工作量大。
3. 客户的需求一变再变：功能需求是在项目进展过程中持续变化的，在使用中会不断有新的需求出现，这是不可避免的，唯一确定的是需求是持续变化的。只能建立统一的文档，平时由运维人员收集修改建议，论证后再录入待办任务，由部门领导安排优先级，设置好交付时间，具体责任到开发人员进行修改。
4. 传统的运维方式费时费力：每个人的开发水平参差不齐，开发人员对自己的代码比较了解，有时只需替换近期修改的文件即可，但大部分运维人员的开发水平相对较弱（甚至没有开发能力），只能全部进行文件的复制替换，不仅费时费力，而且容易出错，遇到网络和服务器的原因还容易导致更新失败。

基线

需求分析：

1. 自动获取最新源码；
2. 自动打包成全量包（full）和半增量包（semi）；
3. 自动部署到指定服务器；
4. 自动进行测试；
5. 能手动回滚到指定 svn 版本；
6. 对 js、css、图片、html 文件进行压缩；
7. js 和 css 文件加 hash 值，cshtml 引用这些文件时能自动更新这些文件的版本信息；
8. 外网访问：在外网也可以使用 pc 或手机登录到 Jenkins 平台进行构建，手机上也可查看记录；
9. 邮件通知：自动推送构建结果到指定开发人员。

特殊名词定义

- 开发人员：提交代码的人员。
- 运维人员：进行部署的人员。
- Jenkins 管理员：维护代码分发脚本的运维人员。
- 全量包：源码编译后输出的所有文件。
- 增量包：上一个 svn 版本与新 svn 版本之间有更改的文件（包括修改和新增）。
- 半增量包：一段时间内有更改的文件（包括修改和新增）。
- 持续集成部署服务器：部署 Jenkins 的服务器。
- 中转服务器：更新包中转服务器，从持续集成部署服务器获取最新更新包并推送到客户服务器。
- 客户服务器：待更新代码的服务器。

参考资料

属于本项目的其他已发表的文件。本文件中引用的其他文献、资料以及软件开发标准。

1. Jenkins 搭建.NET 自动编译测试并实现半增量部署
2. 使用 gulp 部署 web 前端代码
3. Jenkins 持续发布解决方案
4. << 互联网敏捷 DevOps 和自动化之 5.SCM 和持续集成 >>
5. 基于 Docker 的 Jenkins pipeline 工作流.

6.2.2 总体设计

概述

系统环境描述

系统包括的范围：Jenkins 持续集成部署系统。

运行环境：

1. 软件环境：

- windows server 2008 r2 x64 或 windows server 2012 r2 x64
- visual studio 2012
- IIS 7
- 要求能连接到外网的 svn 服务器
- 要求能连接到外网提供 frp 内网穿透服务的服务器
- 要求能连接到外网的腾讯企业邮箱服务器

2. 硬件环境：

- 超融合服务器，8G 内存及以上，500G 硬盘。

3. 开发环境：

- 服务器软件环境：
- windows server 2008 r2 x64 或 windows server 2012 r2 x64
- visual studio 2012
- IIS 7
- node.js 10 及以上
- ssh

- 服务器硬件环境：
- 超融合服务器，8G 内存以上，500G 硬盘。

- 开发机器软件环境：
- mac os x 10.12.6 及以上、win7 、win10
- IIS 7

- visual studio 2012
- node.js 10 及以上
- ssh

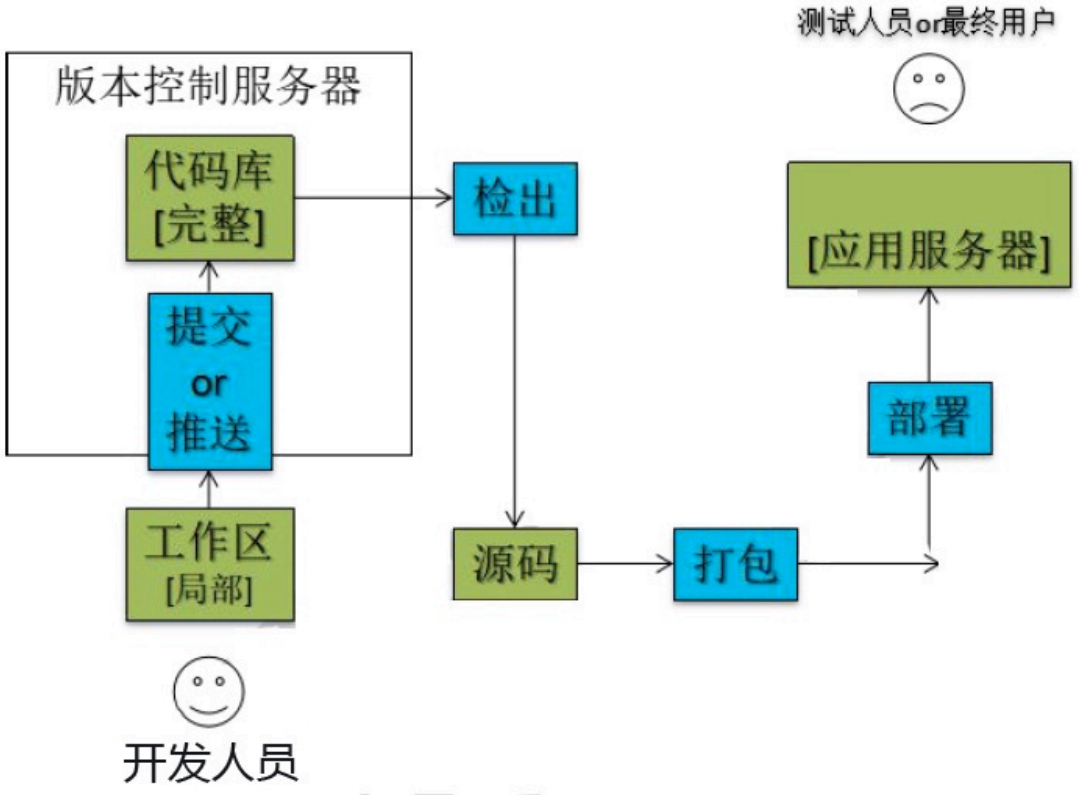
- 开发机器硬件环境：
- 8G 内存及以上，500G 硬盘。

系统总体结构设计

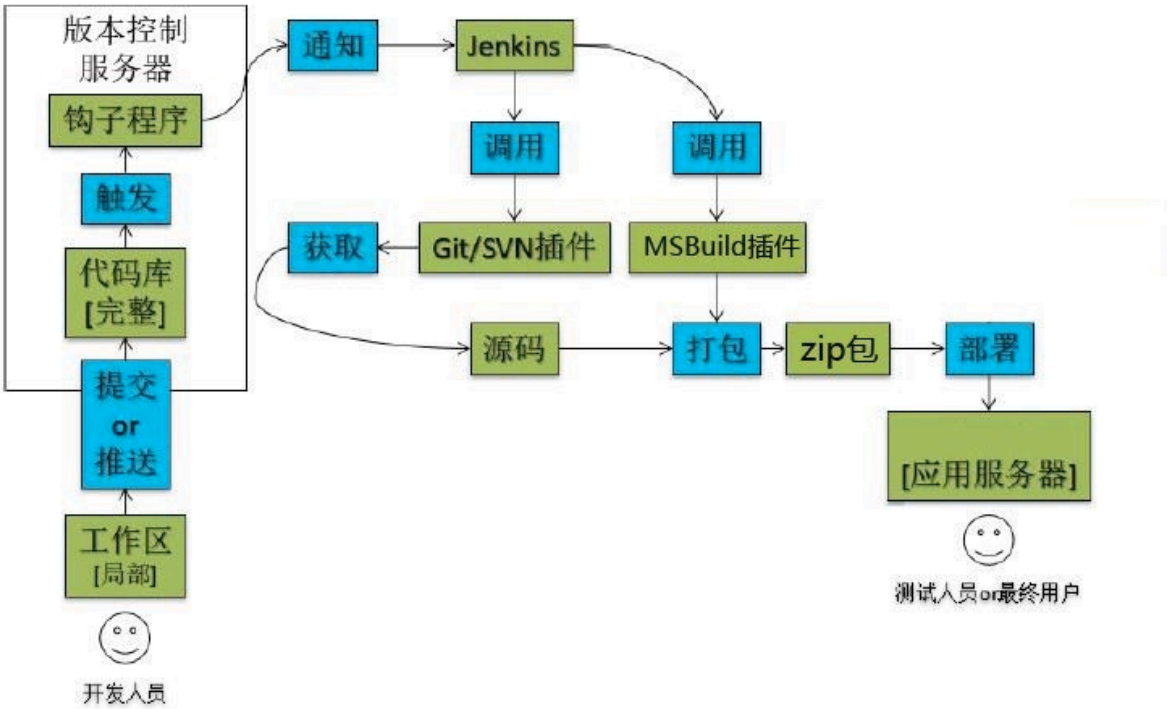
1. 系统业务层次图

- 内容描述：Jenkins 是一款能提高效率的软件，它能帮你把软件开发过程形成工作流，典型的工作流包括以下几个步骤：
- （1）开发
- （2）提交
- （3）编译
- （4）测试
- （5）发布
- 有了 Jenkins 的帮助，在这 5 步中，除了第 1 步，后续的 4 步都是自动化完成的，具体的，当你完成了提交，Jenkins 会自动运行你的编译脚本，编译成功后，再运行你的测试脚本，这一步成功后，接着它会帮你把新程序发布出去，特别的，在最后一步，你可以选择手动发布，或自动发布，毕竟发布这件事情，还是需要人为的确认一下比较好。简而言之

  Jenkins 可以帮你在写完代码后，一键完成开发过程中的一系列工作。
- 使用 Jenkins 的好处显而易见，它减少了你的重复劳动。更重要的是，一个团队的开发流程一开始是不一致的，不一致往往会带来各种各样的问题，最终体现在软件的质量或开发效率不够高，而 Jenkins 会帮你规范大家的行为，从而避免一系列的问题。
- 下图是手动部署的方式图例：



- 下图是自动部署的方式图例：

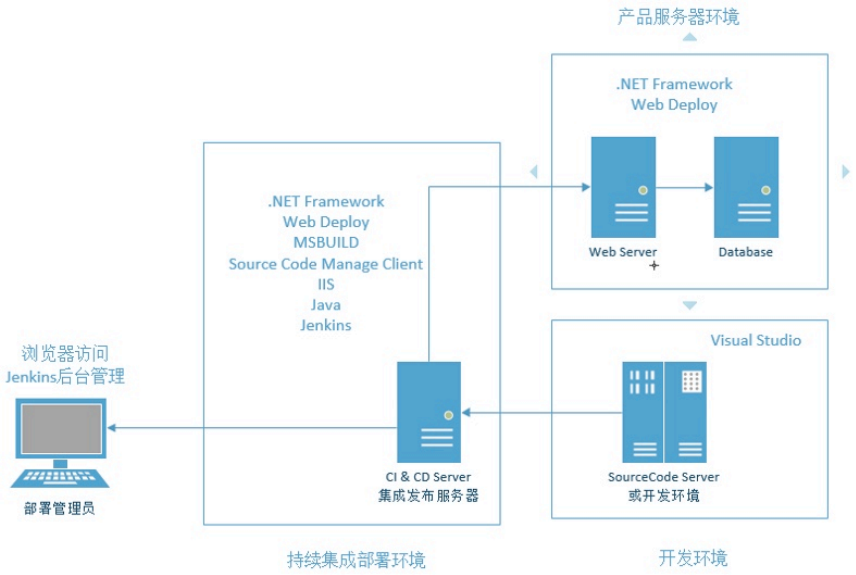


- 而 Jenkins 搭建.NET 自动编译测试并实现半增量部署的方案是基于 Jenkins 平台进行脚本配置的，以下简称 semi-up 方案。

- **semi-up 方案原理：** 进行更新文件的筛选，找出最新文件，压缩并上传到服务器。
- ** 优化后的流程：** 开发人员本地提交源码-自动化服务器定期获取源码并编译-筛查出更新包-同步到其他服务器上进行文件替换。
- 以持续集成部署服务器 A 为例，中转到 B（B 不是必须的），客户端 C。A 主机每天早上 2 点自动获取一次最新的代码，在本地编译完成以后生成一个月内待更新的文件压缩包，B 主机通过网络共享方式获取压缩包并进行分发，C 主机获取上游的压缩包并下载到本地进行解压，通过校验 C 主机本地版本号与 A 主机生成的最新版本号是否一致决定是否进行文件替换，若不一致则把所有 web 目录下指定的文件夹下面的文件替换为最新的文件。
- 功能简介（类似需求分析）：
- 持续集成部署服务器：获取最新源码（默认获取最新，也可手动指定版本号），编译（msbuild 命令行自动编译），sonarqube 测试（代码检查），加 md5 戳（使用 gulp 命令生成 js 文件 md5 映射表，替换 cshtml 文件中的对应引用文本），压缩静态资源（压缩图片、js、css 文件），打包（生成全量包和半增量包），发布（手动或自动发布），邮件通知（通过邮件模板生成邮件并发送给开发人员）。
- 中转服务器：同步打包服务器的全量包和半增量包，分发到客户端服务器。
- 客户端服务器：被动更新（打包服务器发送指令执行更新，可实现回滚操作），主动获取（客户端手动获取全量包并更新，解决半增量包更新失败的情况）。

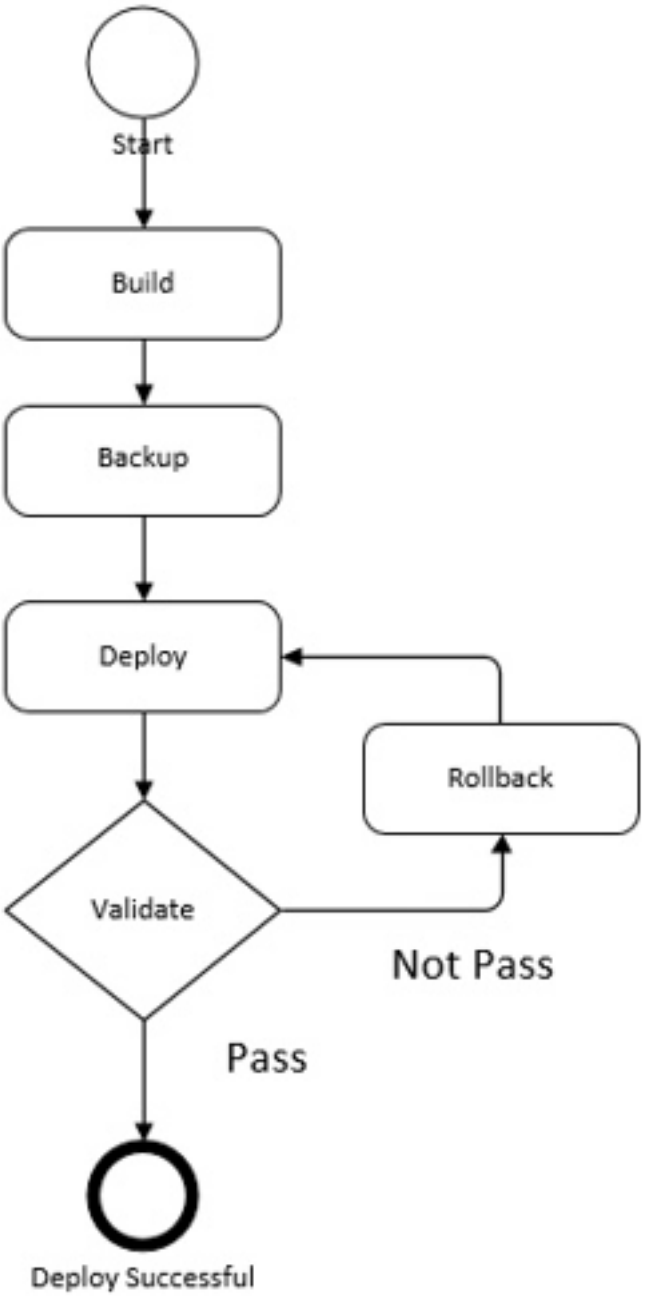
2. 系统架构说明

- 说明整个系统的软硬件架构层次:
- 架构大致分为三个部分：开发环境、产品服务器环境、持续集成部署环境
- 开发环境配置项目部署文件。集成发布环境获取最新代码执行代码编辑及代码发布。产品服务器支持远程程序发布及备份。
- 其中产品服务器需要安装：IIS .NET Framework Web Deploy 等工具；集成发布服务器需要安装 IIS .NET Framework Web Deploy IIS Source Code Client Java Jenkins 等工具。开发环境需要安装 Visual Studio.
- 图例如下：


产品服务器环境
.NET Framework
Web Deploy
Web Server
Database
.NET Framework
Web Deploy
MSBUILD
Source Code Manage Client
IIS
Java
Jenkins
Visual Studio
浏览器访问
Jenkins后台管理
CI & CD Server
集成发布服务器
SourceCode Server
或开发环境
部署管理员
持续集成部署环境
开发环境

-

- 发布流程图:



- 适用于发布 Web 站点，可支持发布失败后回滚，但回滚需要手动指定版本号。

3. 关键技术

- 最新文件筛选：通过使用 xcopy 命令的 d 参数，则可筛选出当前日期前 2 个月内有变化的文件。
- 版本校验：客户端服务器 web 目录中也有版本信息，通过对比 svn 最新的版本信息，判断是否需要更新，若需要更新，则会执行更新脚本。
- 轮询 SCM：可定时检查 svn 服务器是否有最新源码，若有则会自动更新。
- ssh 远程执行：客户端服务器安装 ssh 服务，持续集成部署服务器若需要推送更新，则会使用 ssh 命令

远程执行脚本。

- 回滚：参数化构建，默认为空，若需要回滚，则手动执行工程并录入回滚版本号。

6.2.3 系统功能设计

持续集成部署服务器端

1. 获取源码：最新或手动指定的版本号；
2. 自动或手动触发：可定时执行；
3. 清空缓存；
4. 编译：输出到指定文件夹下；
5. 筛选：根据文件修改日期，获取近期修改过的文件；
6. hash 值：使用 gulp 命令获取 js 文件的 hash 值，并写入 cshtml 文件。若 cshtml 文件是 gb2312 格式，需先使用 iconv 工具转换为 utf-8 格式；
7. 压缩静态资源：压缩图片、js、css、html 文件；
8. 检查版本信息：检查各客户端服务器 web 目录的最新版本信息，判断是否需要更新；更新后再次判断是否更新成功；
9. 推送：把升级包复制到中转服务器；
10. 邮件通知：当前的构建信息、svn 版本信息、要推送的客户端服务器名称、推送结果、构建过程文档添加到附件。

客户服务器端

1. ssh-up 脚本：下载 semi 更新包、清空缓存、解压、覆盖文件、上传最新版本信息。
2. full 包恢复：手动下载 full 更新包，全部覆盖。

6.2.4 尚待解决的问题

1. 若部署时中断 Jenkins 工程，则会下一次重新获取源码，费时很长，而且会导致半增量包按日期筛选的条件失效。
2. js 文件压缩后，集团端平台的一些页面会无法打开（如调度 ppt），原因未知。
3. 目前测试只能人工测试。

CHAPTER 7

项目经理角色-统筹全局

CHAPTER 8

Indices and tables

- genindex
- modindex
- search