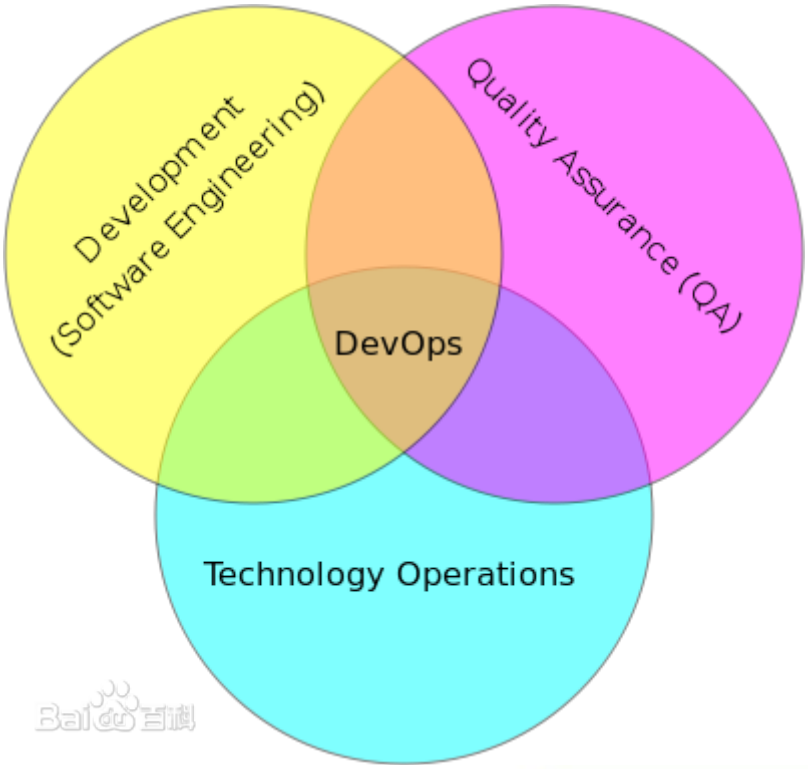
DevOps实战

一、 DevOps概念

1、 DevOps是什么

Development和Operations的组合词；



DevOps能力环

无尽头的可能性：DevOps涵盖了代码、部署目标的发布和反馈等环节，闭合成一个无限大符号形状的DevOps能力闭环。

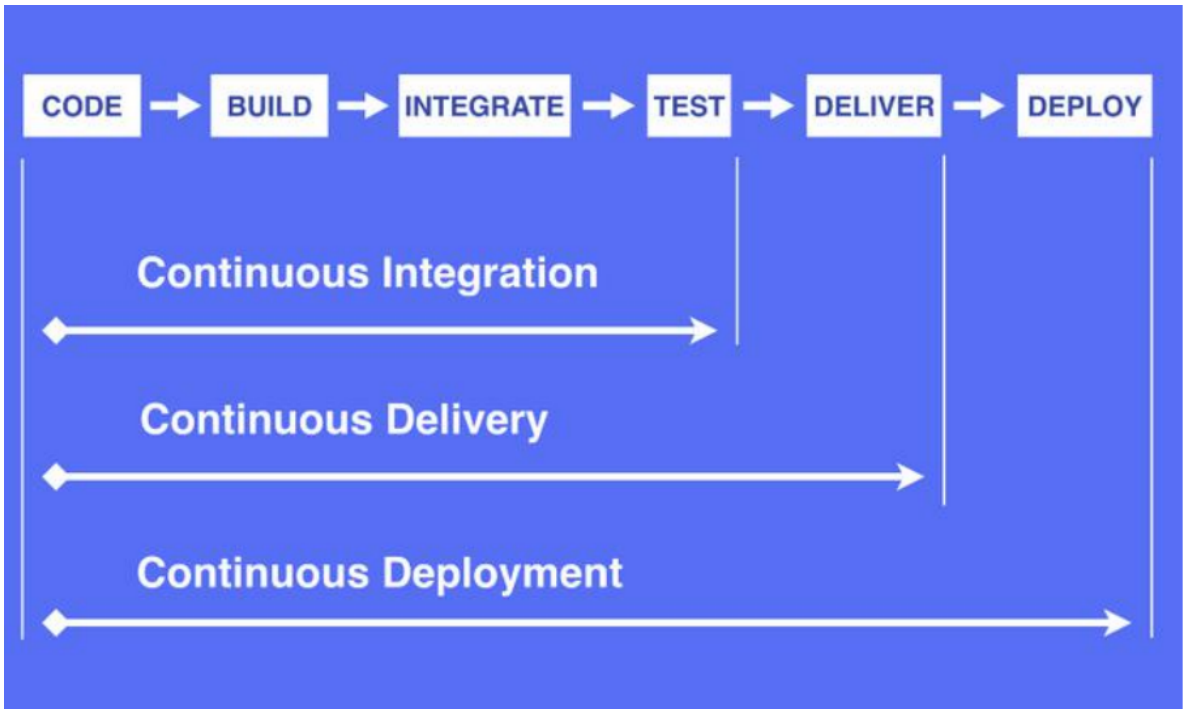
DevOps: Development 和 Operations 的组合

- DevOps 看作开发（软件工程）、技术运营和质量保障（QA）三者的交集。
- 突出重视软件开发人员和运维人员的沟通合作，通过自动化流程来使得软件构建、测试、 发布更加快捷、频繁和可靠。
- DevOps 希望做到的是软件产品交付过程中 IT **工具链的打通**，使得各个团队减少时间损 耗，更加高效地协同工作。专家们总结出了下面这个 DevOps 能力图，良好的闭环可以大大 增加整体的产出。

2、CICD是什么

持续集成　**持续**部署

1、基本理念



1、持续集成（Continuous Integration）

持续集成是指软件个人研发的部分向软件整体部分交付，频繁进行集成以便更快地发现 其中的错误。“持续集成”源自于极限编程（XP），是 XP 最初的 12 种实践之一。

CI 需要具备这些：

- **全面的自动化测试。**这是实践持续集成&持续部署的基础，同时，选择合适的 自动化测试工具也极其重要；
- **灵活的基础设施。容器，虚拟机**的存在让开发人员和 QA 人员不必再大费周 折；
- **版本控制工具。**如 Git，CVS，SVN 等；
- **自动化的构建和软件发布流程的工具**，如 Jenkins，flow.ci；
- **反馈机制。**如构建/测试的失败，可以快速地反馈到相关负责人，以尽快解决达到一个更稳定的版本。

2、持续交付（Continuous Delivery）

持续交付在持续集成的基础上，将集成后的代码部署到**更贴近真实运行环境的**「类生产环境」（production-like environments）中。持续交付优先于整个产品生命周期的软件部署，建立 在高水平自动化持续集成之上。

灰度发布。

持续交付和持续集成的优点非常相似：

- **快速发布。**能够应对业务需求，并更快地实现软件价值。
- **编码->测试->上线->交付**的频繁迭代周期缩短，同时获得迅速反馈；
- **高质量的软件发布标准。**整个交付过程标准化、可重复、可靠，
- **整个交付过程进度可视化，**方便团队人员了解项目成熟度；

- **更先进的团队协作方式。**从需求分析、产品的用户体验到交互 设计、开发、测试、运维等角色密切协作，相比于传统的瀑布式软件团队，更少浪费。

3、**持续部署（Continuous Deployment）**

持续部署是指当交付的代码通过评审之后，**自动部署到生产环境中**。持续部署是持续交付的最高阶段。这意味着，所有通过了一系列的自动化测试的改动都将自动部署到生产环境。它也可以被称为“Continuous Release”。

> “开发人员**提交代码**，持续集成服务器获取代码，执行单元测试，根据测试结果决定是否部署到预演环境，如果成功部署到预演环境，进行整体 *验收测试，如果测试通过，自动部署到产品环境，* **全程自动化高效运转。**

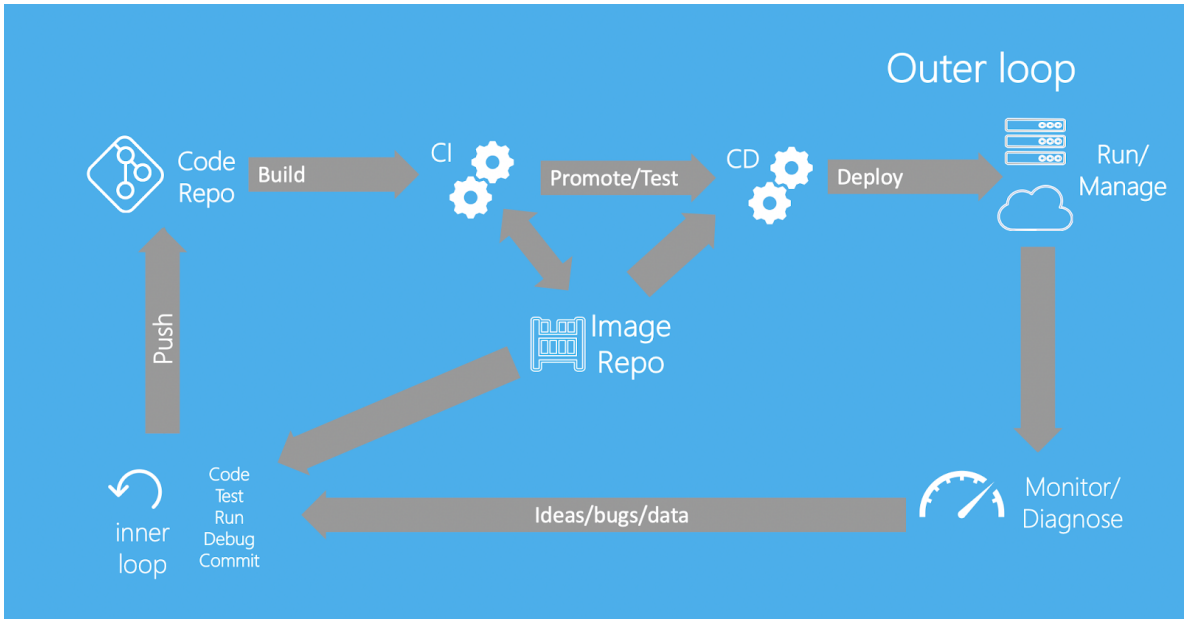
持续部署主要好处是，可以相对独立地部署新的功能，并能快速地收集真实用户的反馈。

> “You build it, you run it”，这是 **Amazon** *一年可以完成 5000 万次部署，* 平均每个工程师每天部署超过 50 次的核心秘籍。
>
> 5000/365 = 15 万次

开发人员代码敲完。可以release的时候，提交代码， 剩下的全部一站式自动搞定

2、最佳实践

1、内循环与外循环

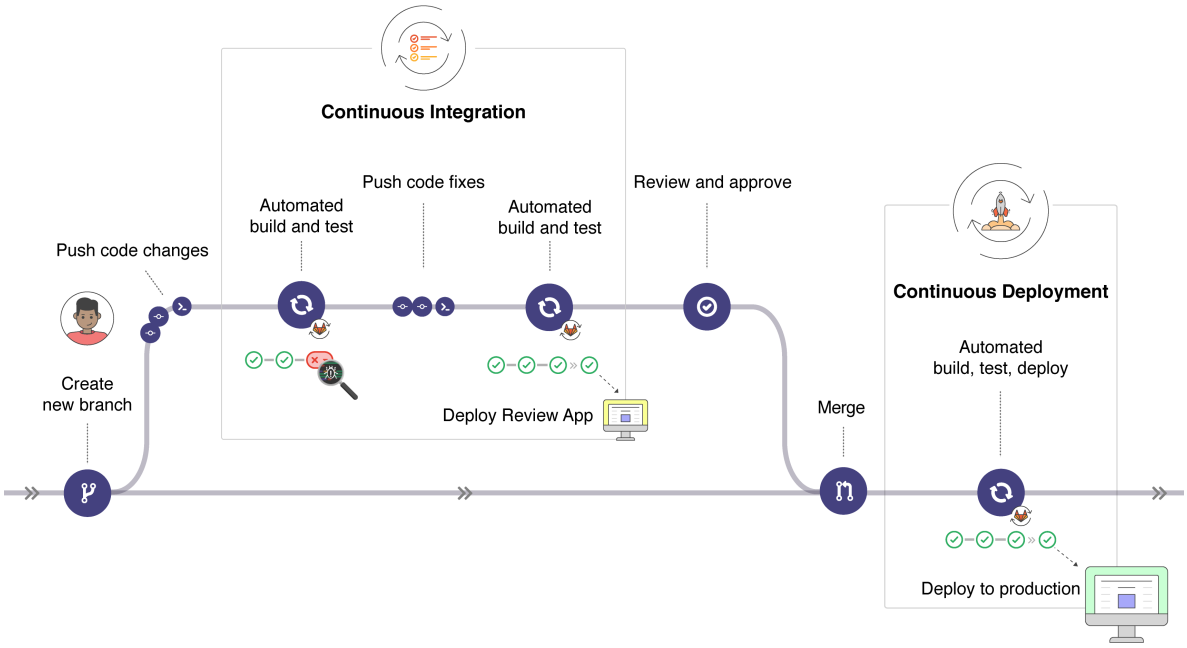


- 内循环（开发要做的事情）：
  - 编码、测试、运行、debug、提交
- 代码推送到代码仓库（svn，git）【代码回滚】
- 进行CI过程（持续集成），万物皆可容器化。打包成一个Docker镜像
- 镜像推送到镜像仓库
- 测试
- 持续部署流程（CD），拿到之前的镜像，进行CD。怎么放到各种环境。uat、test、prod
- 外循环（）
  - 运行时监控

    - 生产环境的管理
    - 监控
    - 线上反馈到开发
- 来到内循环

MVC：Model（bean,entity,to,po.....） View(thymeleaf、前后分离....）Controller（xxxxx）

2、实践流程

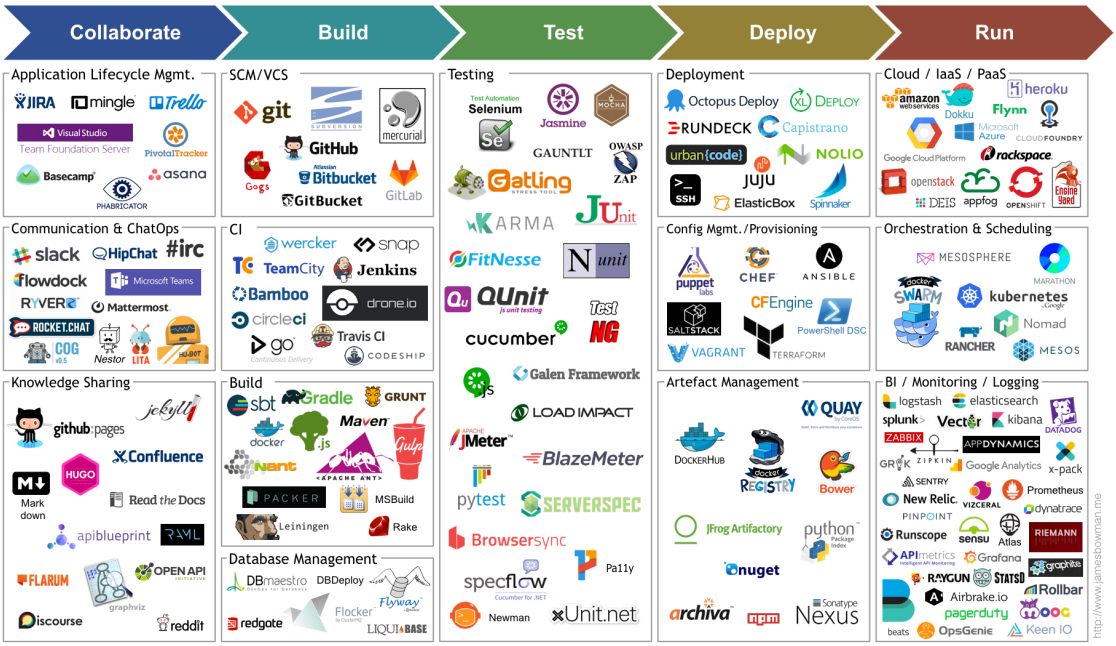


新功能，bug修复。

- 创建分支来做这个事情（开发功能）
- 提交分支的代码改变
- **进入持续集成流程**
    - **当前分支代码功能性自动化构建和测试**
    - **自动工具推送这次提交**
    - **自动化集成测试**
    - **可以看到效果**
    - **人工确认此次功能是否发布到生产环境**
- **代码合并。**
- **进入持续部署流程**
    - 构建、测试、发布.....

3、 CICD LandSpace

禅道



二、Jenkins

```
/var/jenkins_home    jenkins的家目录
包含了jenkins的所有配置。


以后要注意备份   /var/jenkins_home  （以文件的方式固化的）
```

Java

Jenkins镜像用 https://hub.docker.com/r/jenkinsci/jenkins/

驱动我们整个CICD过程的很多工具

1、Jenkins安装

https://www.jenkins.io/zh/doc/book/installing/

```
docker run \
  -u root \
  -d \
  -p 8080:8080 \
  -p 50000:50000 \
  -v jenkins-data:/var/jenkins_home \
  -v /etc/localtime:/etc/localtime:ro \
  -v /var/run/docker.sock:/var/run/docker.sock \
  --restart=always \
  jenkinsci/blueocean


  #自己构建镜像 RUN的时候就把时区设置好
  #如果是别人的镜像，docker hub，UTC； 容器运行时 ， -v /etc/localtime:/etc/localtime:ro


  jenkinsci/jenkins 是没有 blueocean插件的，得自己装
  jenkinsci/blueocean：带了的

  #/var/run/docker.sock 表示Docker守护程序通过其监听的基于Unix的套接字。 该映射允许 jenkinsci/blueocean 容器与Docker守护进程通信， 如果 jenkinsci/blueocean 容器需要实例化其他Docker容器，则该守护进程是必需的。 如果运行声明式管道，其语法包含agent部分用 docker；例如， agent { docker { ... } } 此选项是必需的。

  #如果你的jenkins 安装插件装不上。使用这个镜像【 registry.cn-qingdao.aliyuncs.com/lfy/jenkins:plugins-blueocean 】默认访问账号/密码是【admin/admin】
```

安装插件，并配置用户

2、Jenkins实战

代码在本地修改----提交到远程gitee----触发jenkins整个自动化构建流程（打包，测试，发布，部署）

1、准备一个git项目进行测试

我们以gitee为例，github可能太慢了。需要idea安装gitee插件。或者自己熟悉手动命令也行。

步骤：

- 1、idea创建Spring Boot项目
- 2、VCS - 创建git 仓库
- 3、gitee创建一个空仓库，示例为public
- 4、idea提交内容到gitee
- 5、开发项目基本功能，并在项目中创建一个Jenkinsfile文件
- 6、创建一个名为 devops-java-demo的流水线项目，使用项目自己的流水线

Jenkins的工作流程

1、先定义一个流水线项目，指定项目的git位置

- 流水线启动
  - 1、先去git位置自动拉取代码
  - 2、解析拉取代码里面的Jenkinsfile文件
  - 3、按照Jenkinsfile指定的流水线开始加工项目
    -

> Jenkins重要的点
>
> 0、jenkins的家目录 /var/jenkins_home 已经被我们docker外部挂载了 ；
>
> /var/lib/docker/volumes/jenkins-data/_data
>
> 1、WORKSPACE（工作空间）=/var/jenkins_home/workspace/java-devops-demo
>
> - 每一个流水线项目，占用一个文件夹位置
> - BUILD_NUMBER=5；当前第几次构建
> - WORKSPACE_TMP（临时目录）=/var/jenkins_home/workspace/java-devops-demo@tmp
>
> ```
> 1    JOB_URL=http://139.198.9.163:8080/job/java-devops-demo/
> ```
>
> ```
> 1
> ```

> 2、常用的环境如果没有，jenkins配置环境一大堆操作
>
> 3、jenkins_url： http://139.198.27.103:8080/

2、远程构建触发

期望效果：远程的github代码提交了，jenkins流水线自动触发构建。

实现流程：

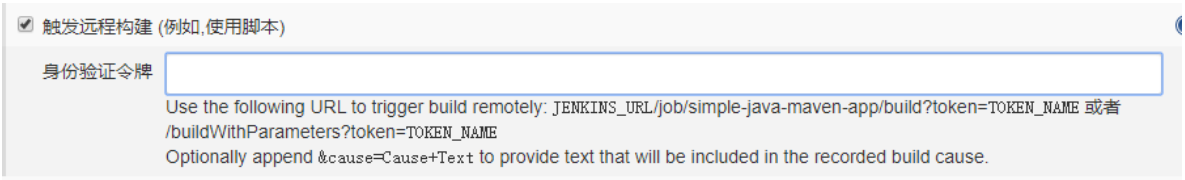
1、保证jenkins所在主机能被远程访问

2、jenkins中远程触发需要权限，我们应该使用用户进行授权

3、配置gitee/github，webhook进行触发

```
#远程构建即使配置了github 的webhook，默认会403.我们应该使用用户进行授权
1、创建一个用户
2、一定随便登陆激活一次
3、生成一个apitoken
http://leifengyang:113620edce6200b9c78ecadb26e9cf122e@139.198.186.134:8080/job/devops-java-demo/build?token=leifengyang

```



远程触发： JENKINS_URL /job/simple-java-maven-app/build?token= TOKEN_NAME 请求即可

6、流水线语法

1、基础格式

```
pipeline {
    agent any
    environment {
        CC = 'clang'
    }
    stages {
        stage('Example') {
            steps {
                sh 'printenv'
                sh 'echo $CC'
            }
        }
    }
}
```

2、环境变量

3、密钥管理

4、自定义agent


```
//需要安装docker、docker pipeline插件

pipeline {
    agent none
    stages {
        stage('Example Build') {
            agent {
                docker 'maven:3-alpine'
                //args 是指定 docker run 的所有指令
                args '-v /var/jenkins_home/maven/.m2:/root/.m2'
            }
            steps {
                echo 'Hello, Maven'
                sh 'mvn --version'
            }
        }
        stage('Example Test') {
            agent { docker 'openjdk:8-jre' }
            steps {
                echo 'Hello, JDK'
                sh 'java -version'
            }
        }
    }
}
```

5、参数化

6、条件表达式

7、修改jenkins插件源

```
1   http://mirror.xmission.com/jenkins/updates/update-center.json
```

jenkins插件中心的插件；

三、 Jenkins On Docker

```
1
```

四、 Jenkins On Kubernetes

```
1
```