

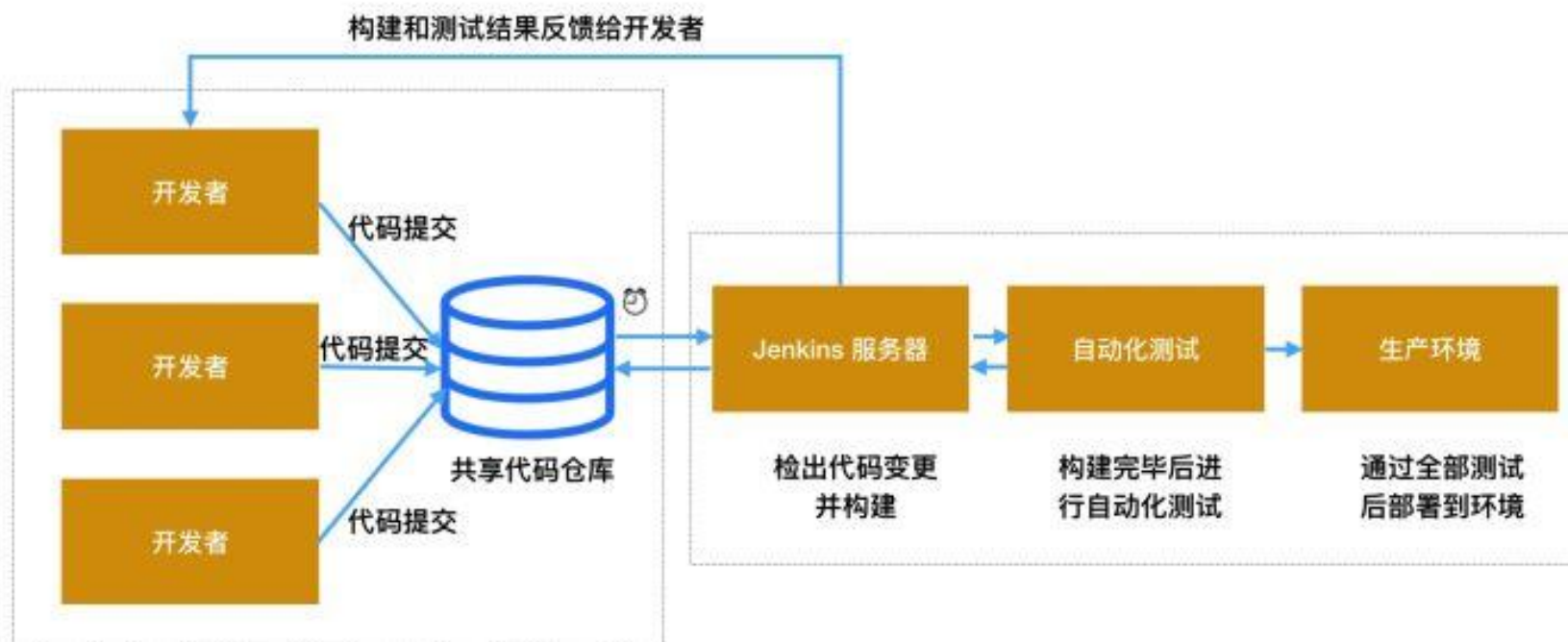
持续集成 (CI) 是每次团队成员提交版本控制更改时自动构建和测试代码的过程。这鼓励开发人员通过在每个小任务完成后将更改合并到共享版本控制存储库来共享代码和单元测试。

2019 DevOps 必备面试题——持续集成篇

Q1: 什么是持续集成?

我会建议你以持续集成的最小定义作为开始来回答这个问题。这是一种研发实践，需要开发人员每天多次将代码集成到共享代码库中。然后通过自动构建来验证每次代码的修改，以便团队尽早发现问题。

我建议你解释一下在以前的工作中是如何实施持续集成的，可以参考以下示例：



知乎 @CODING

在上图中：

- 1、开发人员将代码 **clone** 至私有工作区。
- 2、完成编码后，他们将更改提交至共享代码库中（版本控制仓库）。
- 3、CI 服务器监视代码仓库并在发生更改时检出更改。
- 4、紧接着 CI 服务器提取这些变更进行构建、运行单元以及集成测试。

- 5、CI 服务器会立即告知团队构建成功与否。
- 6、如果构建失败，CI 服务器会向团队发送告警。
- 7、研发团队将尽快解决问题。 8、这个过程会不断重复。

Q2：为什么研发团队需要开发与测试的持续集成？

对于这个答案，你应该关注持续集成的需求。我建议你在回答中提到以下解释： 开发和测试的持续集成通过在完成所有开发之后替换传统的测试实践，来提高软件质量并减少交付耗时。它允许开发团队尽早检测和定位问题，因为开发人员需要每天多次（或更频繁地）将代码集成到代码仓库中，然后自动验证每次集成。

Q3：持续集成的成功因素有哪些？

在这里，你必须提到持续集成的要求，可以在回答中包含以下几点：

- 维护代码仓库
- 自动化构建
- 让构建自我检测
- 每个人每天都确保已将修改提交至基线
- 保持快速构建
- 在生产环境的克隆环境中进行测试
- 研发团队可以轻松获得最新的可交付成果
- 每个人都可以看到最新构建的结果
- 自动部署

Q4：如何将 Jenkins 从一台服务器迁移或者复制到另一台服务器？

我会通过将 jobs 目录从旧服务器复制到新服务器的方式来完成这个事情。有很多种方法可以做到这一点：

- 只需复制相应的 job 目录，即可将 job 从一个 Jenkins 服务器移动到另一个。
- 通过使用其它名称克隆 job 目录来制作现有 job 的副本。
- 通过重命名目录来重命名现有 job。请注意，如果你更改了 job 名称，则需要更改尝试调用该重命名 job 的所有 job 。

Q5：如何在 Jenkins 中创建备份和复制文件？

可以很直接地回答这个问题：要创建备份。你需要做的就是定期备份 JENKINS_HOME 目录。这包含所有构建 job 配置，从属节点配置和构建历史记录。要创建 Jenkins 的备份，只需复制此目录即可，你还可以复制 job 目录或重命名目录。

Q6：如何配置 Jenkins 的 job？

关于这个答案的解决方法是首先提一下如何创建 job：转到 Jenkins 首页，选择 “New Job” ，然后选择 “Build a free-style software project” 。然后你可以设置这个自由式 job 的元素：

- 可选的 SCM，例如源代码所在的 CVS 或 Subversion。
- 用于控制 Jenkins 何时执行构建的触发器。
- 某种构建脚本，用于执行实际工作的构建（ant, maven, shell 脚本，批处理文件等）。
- 从构建中收集信息的可选步骤，例如归档制品、记录 javadoc 和测试结果。
- 配置构建结果通知其他人/系统的步骤，例如发送电子邮件、即时消息、更新问题跟踪器等。

Q7: 列举 Jenkins 中一些有用的插件

下面我将提到一些重要插件：

- Maven 2 project
- Amazon EC2
- HTML publisher
- Copy artifact
- Join
- Green Balls

我觉得这些是最有用的插件，你也可以添加你认为有用的插件。但是请确保首先提到上述插件，然后添加你自己的插件。

Q8: 如何保证 Jenkins 的安全？

- 确保 global security 配置项已经打开。
- 确保用适当的插件将 Jenkins 与企业员工目录进行集成。
- 确保启用项目矩阵的权限访问设置。
- 通过自定义版本控制的脚本来自动化 Jenkins 中设置权限/特权的过程。
- 限制对 Jenkins 数据/文件夹的物理访问。
- 定期对其进行安全审核。

Jenkins+Github 持续集成

由于最近团队代码库从 coding 迁移到 github，在 CI 工具的选型上尝试了 [travis-ci](#) 和 [circle-ci](#)，最后决定自己搭建 CI 服务器，而我也有幸认领了这个任务的调研，因此有了这篇文章。

之前写过一篇文章[浅谈 Jenkins+Node.js 持续集成](#)，那真的是浅谈，Jenkins 包含的东西实在太多了，作为从 hudson 分支出来的开源免费的版本，插件与 hudson 通用，有更快的迭代速度和稳定性。

为什么选择 Jenkins

答案简单：因为免费，学习资料多。

开始吧

安装配置这里就不赘述了，移步[浅谈 Jenkins+Node.js 持续集成](#)

0. 准备

因为要与 Github 通信，所以需要准备一台服务器，该服务器能访问到 Github，Github 能访问到它。

为了这个测试，我特地在[搬瓦工 VPS](#)买了服务器，顺便介绍一下这个高性价比的 vps 供应商，\$2.99 约合人民币 18 元每个月，可一键搭建 shadowsocks。但是记得有个坑就是购买的时候一定要选好机房，之前买过洛杉矶的卡的要死，打条命令之后要等好久才显示，对它失去信心不想用它了，后来听一个朋友说[亚利桑那州 \(Arizona\)](#) 的机房挺稳定的，再给它一次机会，这次买了 Arizona 机房的果然速度挺快的 😊











1. 安装 GitHub Plugin

直接安装 Github Plugin, jenkins 会自动帮你解决其他插件的依赖，直接安装该插件 Jenkins 会自动帮你安装 [plain-credentials](#) 、 [git](#) 、 [credentials](#) 、 [github-api](#)

安装/更新 插件中

准备

- Checking internet connectivity
- Checking update center connectivity
- Success

GitHub API Plugin	 完成
Credentials Plugin	 credentials plugin is already installed. Jenkins needs to be restarted for the update to take effect
SSH Credentials Plugin	 ssh-credentials plugin is already installed. Jenkins needs to be restarted for the update to take effect
GIT client plugin	 完成
SCM API Plugin	 完成
Mailer Plugin	 mailer plugin is already installed. Jenkins needs to be restarted for the update to take effect
GIT plugin	 完成
Plain Credentials Plugin	 完成
GitHub plugin	 等待
重启 Jenkins	 等待

- ➡ [返回首页](#)
(返回首页使用已经安装好的插件)
- ➡ ☒ 安装完成后重启Jenkins(空闲时)

2. 配置 Github 插件

系统管理 >> 系统设置 >> GitHub Plugin Configuration

GitHub Plugin Configuration

Servers configs with credentials to manage GitHub integrations

GitHub Server Config

Manage hooks ☒

Credentials - none -

Add

You can create own [personal access token](#) at GitHub settings.
Token should be registered with scopes:

- **admin:repo_hook** - for managing hooks (read, write and delete old ones)
- **repo** - to see private repos
- **repo:status** - to manipulate commit statuses

In Jenkins create credentials as «Secret Text», provided by [Plain Credentials Plugin](#)

WARN! Creds are filtered on changing custom GitHub url

If you have existed GitHub login and password you can convert it to token automatically with help of «[Manage additional GitHub actions](#)»

(from [GitHub plugin](#))

☐ Custom GitHub API URL

高级...

Verify credentials

Delete GitHub Server Config

首先点击 [personal access token](#) 到 github 上
也就是 github 上用户 Settings >> personal access tokens

New personal access token

Token description

token4JenkinsTest

What's this token for?

Select scopes

Scopes *limit* access for personal tokens. [Read more about OAuth scopes.](#)

☒ repo ⓘ
☐ public_repo ⓘ
☐ user:email ⓘ
☐ write:org ⓘ
☐ write:public_key ⓘ
☐ write:repo_hook ⓘ
☐ gist ⓘ

☐ repo:status ⓘ
☐ delete_repo ⓘ
☐ user:follow ⓘ
☐ read:org ⓘ
☐ read:public_key ⓘ
☐ read:repo_hook ⓘ
☐ notifications ⓘ

☐ repo_deployment ⓘ
☐ user ⓘ
☐ admin:org ⓘ
☐ admin:public_key ⓘ
☒ admin:repo_hook ⓘ
☐ admin:org_hook ⓘ

Generate token

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).



勾选给 Jenkins 的访问权限，Github plugin 的帮助信息里说要 admin:repo_hook、repo 和 repo:status 权限，其实 repo:status 是包含在 repo 里的，详见[这里](#)。
点击 Generate token 创建一个 token

Personal access tokens

Generate new token

Tokens you have generated that can be used to access the [GitHub API](#).

Make sure to copy your new personal access token now. You won't be able to see it again!

✓  

EditDelete

复制这个 token，回到 Jenkins 点击 Add 按钮

The image shows the 'Add Credentials' dialog in Jenkins. It has a title bar with the Jenkins logo and the text 'Add Credentials'. Below the title bar, there are several fields: 'Kind' is set to 'Secret text', 'Scope' is 'Global (Jenkins, nodes, items, all child items, etc)', 'Secret' is a text box containing a masked token, and 'Description' is 'token4JenkinsTest'. There are help icons (question marks) next to the 'Scope' and 'Description' fields. At the bottom left, there are 'Add' and 'Cancel' buttons. At the bottom right, there is an '高级...' (Advanced...) button. Red boxes highlight the 'Kind' dropdown, the 'Secret' text box, and the 'Add' button.

选择 Secret text，粘贴 token，添加描述，点击添加。

点击 Verify credentials 测试 token，显示 Credentials verified for user xxx, rate limit: xxxx，说明配置完成了，这样你的 Jenkins 就具有访问你的 github 的权限了。

3. 创建一个 freestyle 任务

- 填写 GitHub project URL^{[OBJ][OBJ][OBJ]}，也就是你的项目主页
https://github.com/your_name/your_repo_name

The image shows the 'New Freestyle Project' dialog in Jenkins. It has a title bar with the Jenkins logo and the text 'New Freestyle Project'. Below the title bar, there are several fields: '项目名称' (Project Name) is 'jenkins_github_test', '描述' (Description) is a text box, and 'GitHub project' is 'https://github.com/wuyanxin/jenkins_github_test/'. There are help icons (question marks) next to the 'GitHub project' field. Below the 'GitHub project' field, there are several checkboxes: '丢弃旧的构建' (Discard old builds), '参数化构建过程' (Parameterize build process), '关闭构建 (重新开启构建前不允许进行新的构建)' (Disable build (no new builds allowed until build is restarted)), and '在必要的时候并发构建' (Concurrent builds when necessary). There is a '预览' (Preview) link next to the 'Plain text' label. The 'Plain text' label is also present.

- 配置源码管理

源码管理

☐ None

☐ CVS

☐ CVS Projectset

☒ Git

Repositories

Repository URL

Credentials

Branches to build

Branch Specifier (blank for 'any')

源码库浏览器

URL

- 填写项目的 git 地址, eg. https://github.com/your_name/your_repo_name.git
- 添加 github 用户和密码
- 选择 githubweb 源码库浏览器, 并填上你的项目 URL, 这样每次构建都会生成对应的 changes, 可直接链到 github 上看变更详情
- 构建触发器勾选 Build when a change is pushed to GitHub, 这样该仓库的每一次 push 或者 pull request 都会触发 build

构建触发器

☐ 触发远程构建 (例如,使用脚本)

☐ Build after other projects are built

☐ Build periodically

☒ Build when a change is pushed to GitHub

☐ Poll SCM

- 配置构建步骤随后配置构建环境、构建步骤和构建后步骤

安装了 Github Plugin 之后在构建步骤和构建后操作会多两个设置，用于在构建时和构建后同步构建状态到 Github 的，后面有效果图

构建环境

☒ Provide Node & npm bin/ folder to PATH

Installation

v4.2.2

Specify needed nodejs installation where npm installed packages will be provided to the PATH

构建

Set build status to "pending" on GitHub commit

删除

Execute shell

Command

npm install

node app

mocha test/success.js

See [the list of available environment variables](#)

删除

增加构建步骤

构建后操作

Set build status on GitHub commit

高级...

删除

增加构建后操作步骤

4. 配置 Github 仓库的 Webhook:

仓库的创建人在仓库的 Settings >> Webhooos & services 添加我们只需要 push 事件触发就可以了，选中 Just the push event 点击 Add webhook

yes, 与 github 集成的 Jenkins CI 环境就配置好了

Options

Collaborators

Branches

Webhooks & services

Deploy keys

Webhooks / **Add webhook**

We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, etc). More information can be found in [our developer documentation](#).

Payload URL *

Content type

application/json

Secret

Which events would you like to trigger this webhook?

☒ Just the push event.

☐ Send me **everything**.

☐ Let me select individual events.

☒ **Active**

We will deliver event details when this hook is triggered.

Add webhook

5. 效果: 每次 push 都会触发一次 build, pull request 的话还会在该界面直接显示 build 结果

trigger one error build 2

63fa68d

Add more commits by pushing to the **develop** branch on **wuyanxin/jenkins_github_test**.



All checks have failed

1 errored check

[Hide all checks](#)

 **jenkins_github_test** — Build #8 failed in 4.5 sec

[Details](#)



This branch is up-to-date with the base branch

Merging can be performed automatically.



Merge pull request

You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

- trigger one error build 2 ✖ 63fa68d
- fix the error ✔ a56693b

Add more commits by pushing to the **develop** branch on **wuyanxin/jenkins_github_test**.



All checks have passed
1 successful check

[Hide all checks](#)

✔ **jenkins_github_test** — Build #9 succeeded in 5.3 sec

[Details](#)



This branch is up-to-date with the base branch
Merging can be performed automatically.



Merge pull request

You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

最后

整个环境终于搭好了，中间遇到了蛮多大坑小坑的，有些记录了下来，后续整理好再发上来。
这段时间学习 **Jenkins** 收获蛮多的，只是到现在也只学了些皮毛，写出来的东西也颇有些晦涩。
接下来要做的实验是通过 **Jenkins** 实现自动远程部署。