持续集成 CI/CD

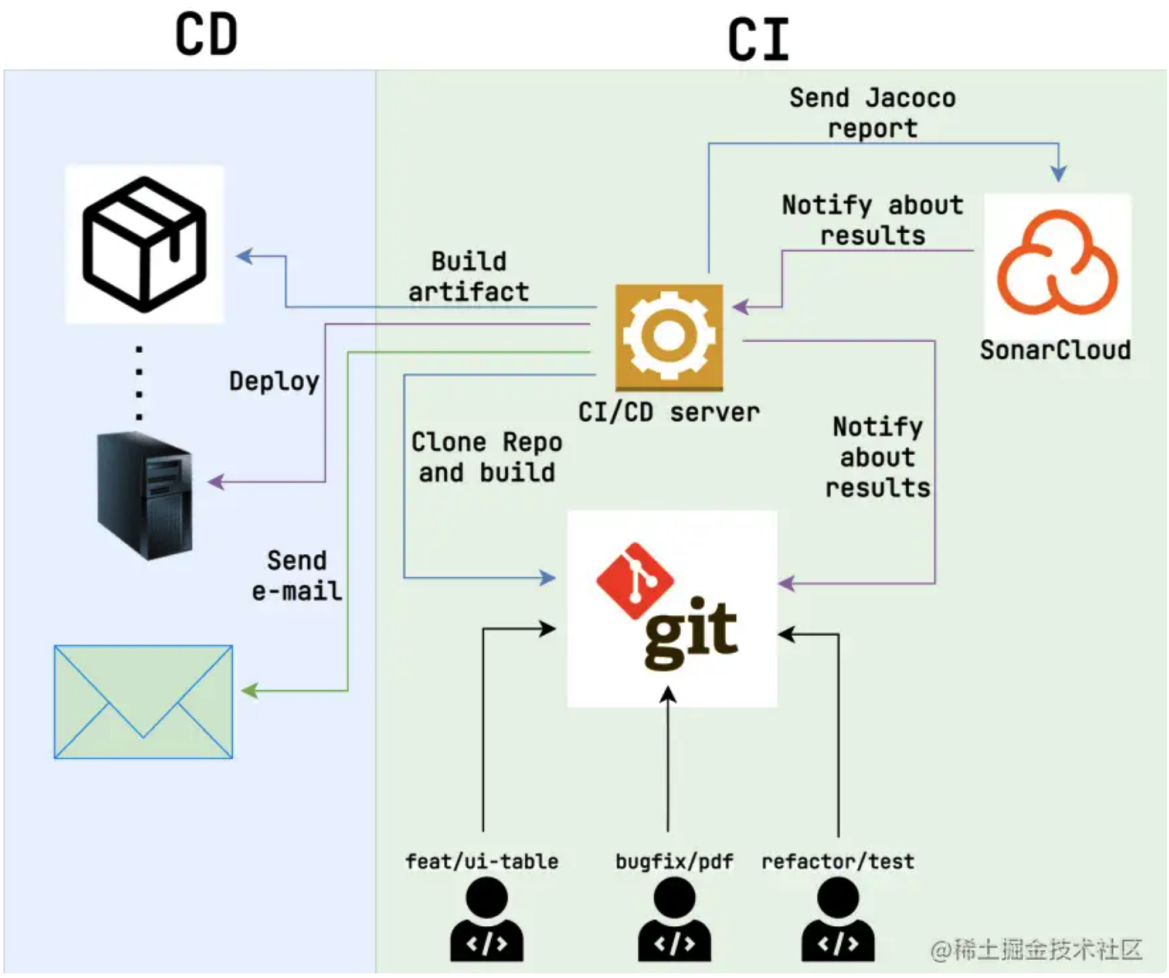
什么是 CI/CD

CI 持续集成（Continuous Integration）

CD 持续交付（Continuous Delivery）

CD 持续部署（Continuous Deployment）

通过一系列自动化的脚本执行，实现开发过程中的代码的交付和部署，能够快速交付，提高团队开发的效率。



什么是持续集成

代码合并，构建，部署，测试都在一起，不断地执行这个过程，并对结果反馈。避免微小的错误累。保证在不断迭代开发。

什么是持续交付

系统可以自动为将代码更改发布到生产环境做好准备。

前端的 CI/CD

具体到前端的 CI 与 CD 任务

持续集成 – CI	持续交付 – CD
代码静态检查 Eslint	部署到应用服务器 (Nginx)
编译与构建（webpack 或 rollup）	部署到软件包管理器 (Npm)
回归单元测试 – 覆盖率报告	Webhook 消息推送 – email /钉钉
BVT 测试 – 集成测试 – 端到端测试	

几种 CI/CD 工具

Jenkins

Jenkins 是一款诞生比较早的老牌的开源 CI/CD 工具，可以自动化执行各种任务，包括构建、测试和部署软件。可以通过配置和编写脚本自动化执行 CI/CD 流程，代替手动 CD/CD 流程，从而节省开发时间，减少错误率，并且会将每一次构建结果记录下来。

CircleCI

CircleCI 是基于云的 CI/CD 工具，可自动执行软件构建和交付过程。它提供快速的配置和维护，没有任何复杂性。由于它是基于云的 CI/CD 工具，因此消除了专用服务器的困扰，并降低了维护本地服务器的成本。此外，基于云的服务器是可扩展的，健壮的，并有助于更快地部署应用程序。

Github Action

Github Action 是 github 官方提供的 CI/CD 服务。那么什么是 Github Action 中的 Action 呢？在 CI/CD 中，比如抓取代码、运行测试、发布第三方服务等，这一个个操作点就是一个个的 Action。

Gitlab CI

和 Github CI 类似，Gitlab CI 是 Gitlab 提供的 CI/CD 服务。你只需要在你项目中的根目录下加上 `.gitlab-ci.yml` 包含构建、测试和部署的脚本即可。GitLab 如果检测到仓库中有该文件，就会使用 `Gitlab Runner` 工具按照顺序运行你设置的构建、测试和部署的脚本。

实战 Github Action 实战前端自动发布

https://github.com/su37josephxia/lego-react

.github/workflows/publish.yml

YAML

```yaml
name: Publish to Aliyun

on:
  push:
    branches: [ main ]
  pull_request:
    branches: [ main ]

jobs:
  build:
    runs-on: ubuntu-latest
    steps:
    # 下载代码
    - uses: actions/checkout@v2
    # 安装Nodejs
    - uses: actions/setup-node@v2
      with:
        node-version: 14
    - run: yarn
    - run: yarn build
    # 部署到阿里云
    - name: Deploy to Aliyun
      uses: easingthemes/ssh-deploy@v2.1.1
      env:
        SSH_PRIVATE_KEY: ${{ secrets.ACCESS_TOKEN }}
        ARGS: "-avzr --delete"
        SOURCE: "build/"
```

```
        REMOTE_HOST: ${{ secrets.SERVER_IP }}
        REMOTE_USER: "root"
        TARGET: ${{ secrets.TARGET }}
```

ssh 登录的过程

- 公钥
- 私钥

客户端/持续集成服务器(私钥）→　服务器（信任列表：公钥）

生成秘钥对

JSON

```
ssh-keygen -t rsa -C action@github.com
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/josephxia/.ssh/id_rsa): ./ssh
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in ./ssh
Your public key has been saved in ./ssh.pub
The key fingerprint is:
SHA256:9H40KvXvJ4j/mFK1tlUpKAGD8B07fYvmuPoUGBeugXg action@github.com
The key's randomart image is:
+---[RSA 3072]----+
|   .. .=.        |
|  . o.o *.       |
| . E + *.....   .|
|  .   *..oo.....|
|     o .So+.o....|
|        =o +..o .|
|       o..oooo o |
|      . ..o..+o .|
|     .oo   o+o+o |
+----[SHA256]-----+
```

- 将公钥放在 应用服务器的 ~/.ssh/authorized_keys 中

- 将私钥放在 Github 项目的 Secret 中