Web 持续集成实践

王集鹄 2017年03月11日

背景

创业公司的工作方法

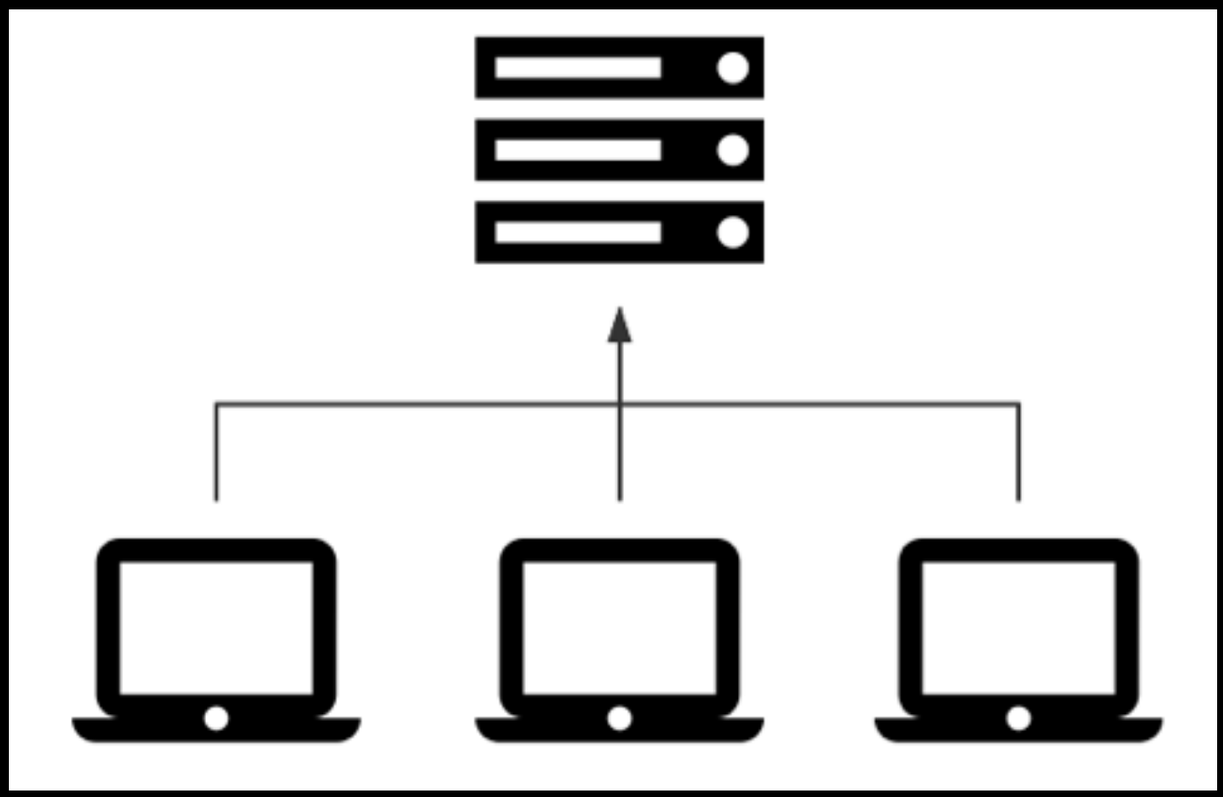
打开冰箱做一顿饭，有什么就得用什么做。

What is Continuous Integration?

持续集成是通过平台串联各个开发环节，实现和**沉淀**工作**自动化**的方法。

HTML5 DREAMWORKS IT大咖说 不止于技术

为什么要做持续集成

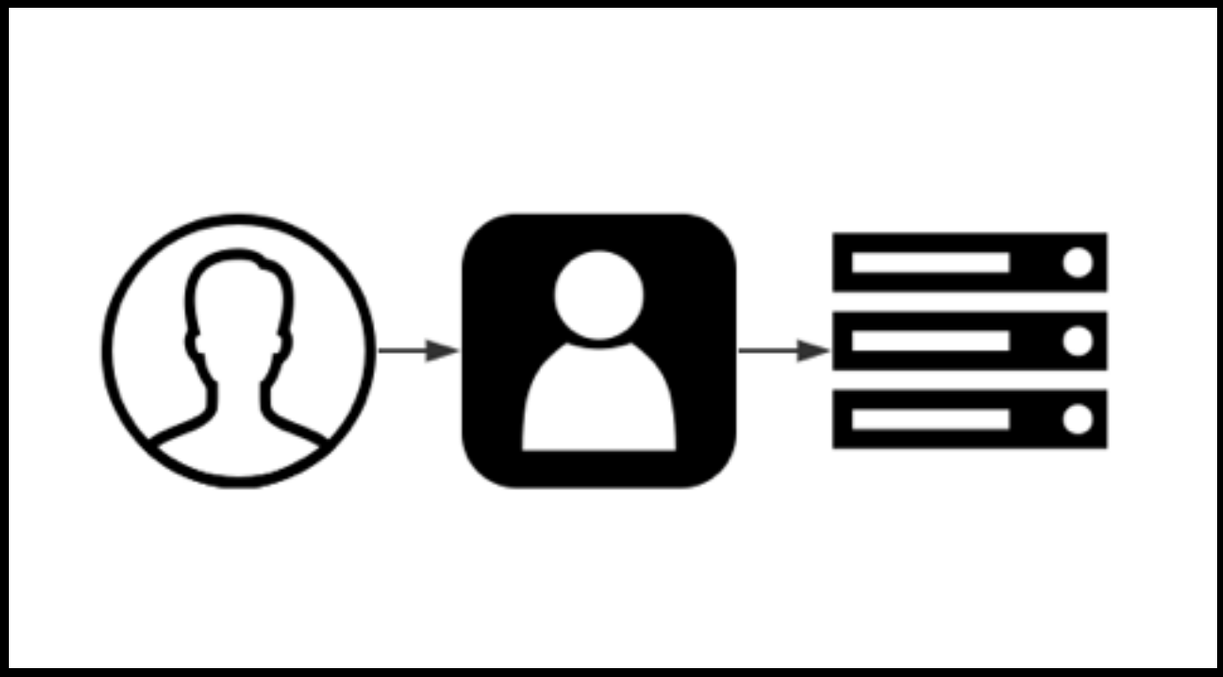


线上代码和代码仓库不同步

加盟公司后，我发现上线部署是通过 FTP 直接上传代码，使用文件比较工具进行代码合并。由于配置不一样，修改的人不一样，经常导致代码仓库和线上代码不统一。每次上线之前代码都要做一次线上线下手工合并。

缺少自动化测试

每次上线仅仅依赖人工测试，测试用例难以覆盖所有被影响的功能，
常常出现初级的接口问题，直到产品上线用户反馈后才能发现问题。



只有程序员才能上线

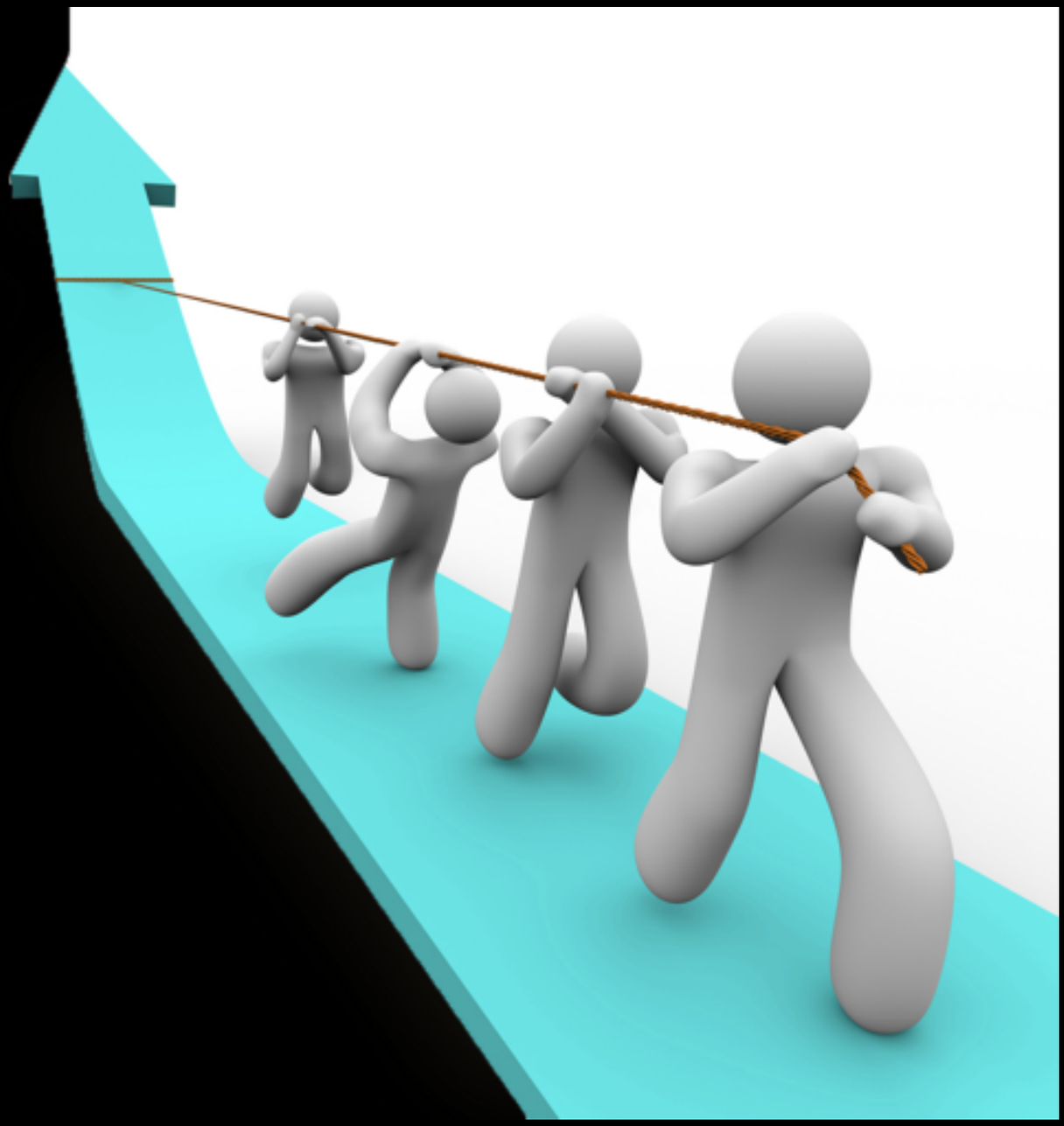
活动页面的文案需要运营同学反复推敲，频繁修改习以为常。可每次修改文案都需要研发同学介入才能部署生效。为修改一个字，研发就需要陪运营熬到很晚。

自动化的需求

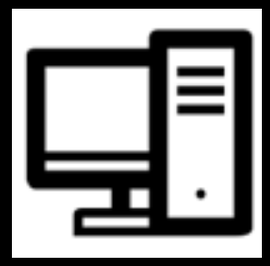
- 自动编译
  - 自动引入各种依赖（开发依赖、包依赖、配置依赖）。资源自动转码、合并、压缩。自动处理配置文件。
- 自动部署
  - 静态资源自动上传 CDN 服务器。应用文件自动上传和同步到应用服务器。
- 自动测试
  - 自动进行单元测试、集成环境测试。
- 自动监控
  - 构建异常、测试异常、运行异常自动通知相关负责人。

团队协作的需求

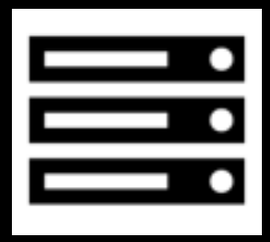
- 成员快速体验最新版本
  - 第一时间部署内测版本，并自动通知团队成员
- 策划直接修改文案上线
  - 面向用户的说明文档，如仅文案修改不需要介入研发人力，即可完成线上更新。
- 设计师直接更新素材
  - 设计师可以直接更新图片资源，图片自动切割、转码、上线
- 数值工程师直接更新配表
  - 数值工程师指游戏场景中设计装备、属性和等级数值关系的人。数值配置通常是一份 Excel 文件。需要自动编译、更新和推演。



适配各种运行环境



- 本机环境 local
  - 应用能最少依赖在本机运行。能够及时修改和预览代码。能够模拟运行环境（接口或数据）。



- 开发环境 develop
  - 一般 Web 项目上线前，都会有一个局域网的开发环境供团队成员测试和体验。开发环境有完整的沙盒数据与线上隔离。方便打印完整日志、提供特权。

- 线上环境 online
  - 线上环境也叫生产环境，直接面向用户。访问的是真实数据，测试和体验时需非常谨慎。通常会上线多个版本，方便测试和回滚。

敏捷开发的需求

- 时间上：小步快跑，推进每次迭代速度，**沉淀**工作方法
- 空间上：将各个岗位的工作汇集和串联实现**自动化**

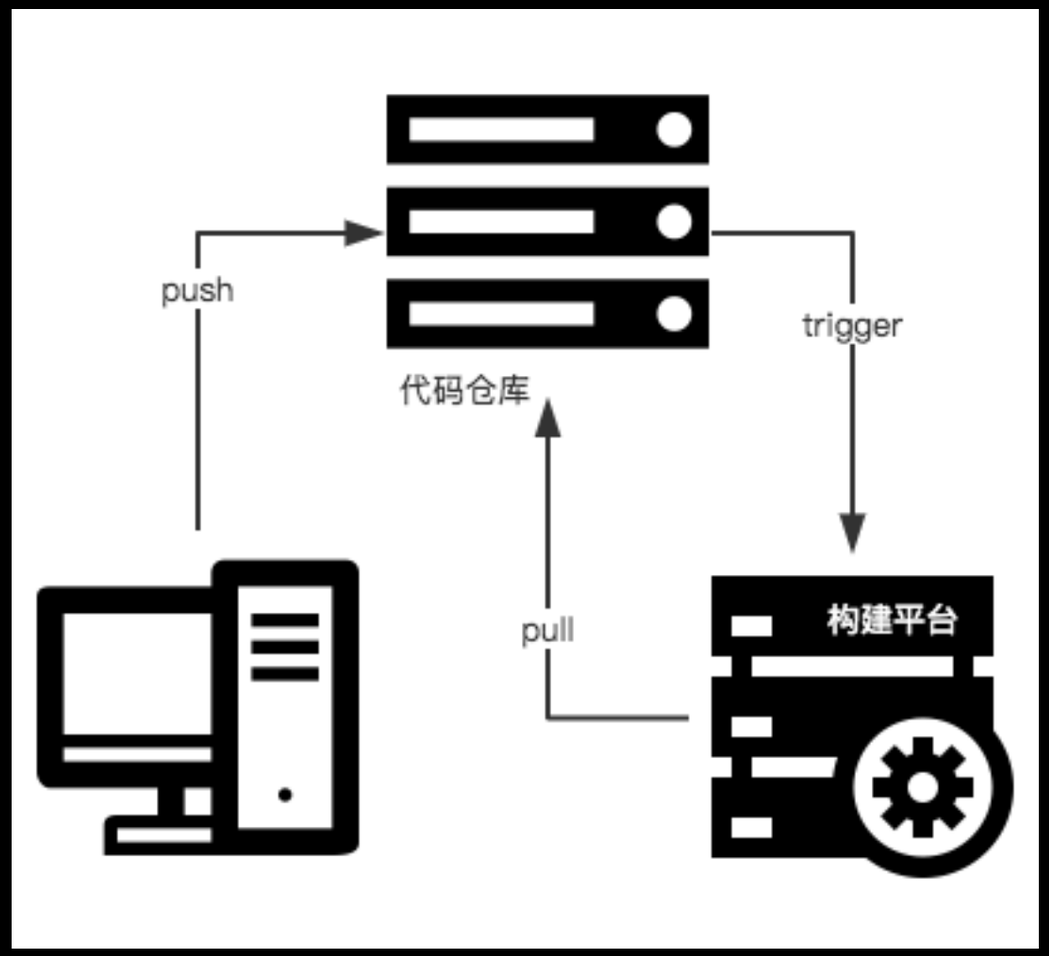
怎么做持续集成

CI 需要的工具

- 统一的代码仓库 GitLab
- 构建平台 Jenkins、Travis CI
- 构建工具 Gulp、FIS3
- 部署工具 rsync

创建 CI 项目

- 基础设置
- 指定代码仓库和相关授权用户
- 设置构建触发器
- 设置构建脚本
- 设置构建异常通知



构建脚本


```
export PATH=$PATH:/usr/local/bin:/usr/local/git/bin

# 避免错误构建
node -e 'if(require("./package.json").JOB_NAME!==process.env.JOB_NAME){console.error("JOB_NAME
    Error.");process.exit(1)}'

yes | cp -R /home/ci/jhtmls.com/* ./

npm install --registry=https://registry.npm.taobao.org
bower install

npm run develop
npm run online # 进入内测阶段后开启
```

构建实例

Edit File Preview Changes

master public/index.html

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4    <title>情人节</title>
5  </head>
6  <body>
7    <div>
8      活动奖励：
9      <p>
10       每日充值满 30 元即可获得白玫瑰 1 只。
11     </p>
12   </div>
13 </body>
14 </html>
```

Edit File Preview Changes

master public/index.html

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4    <title>情人节</title>
5  </head>
6  <body>
7    <div>
8      活动奖励：
9      <p>
10       每日累计充值满 30 元即可获得白玫瑰 3 只。
11     </p>
12   </div>
13 </body>
14 </html>
```

简单文案更新由运营同学完成

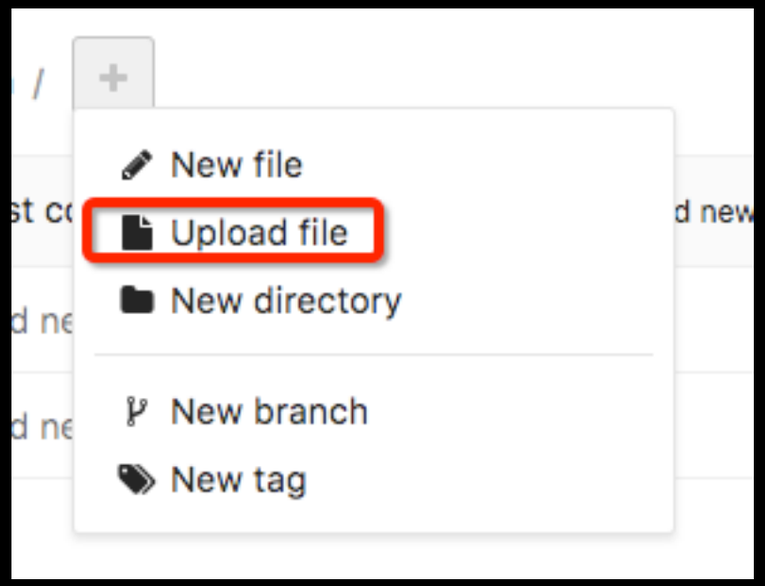
运营同学不需要安装其他软件，直接在浏览器中修改 GitLab 项目文件（通常是 HTML 中的文案），保存即刻更新上线。

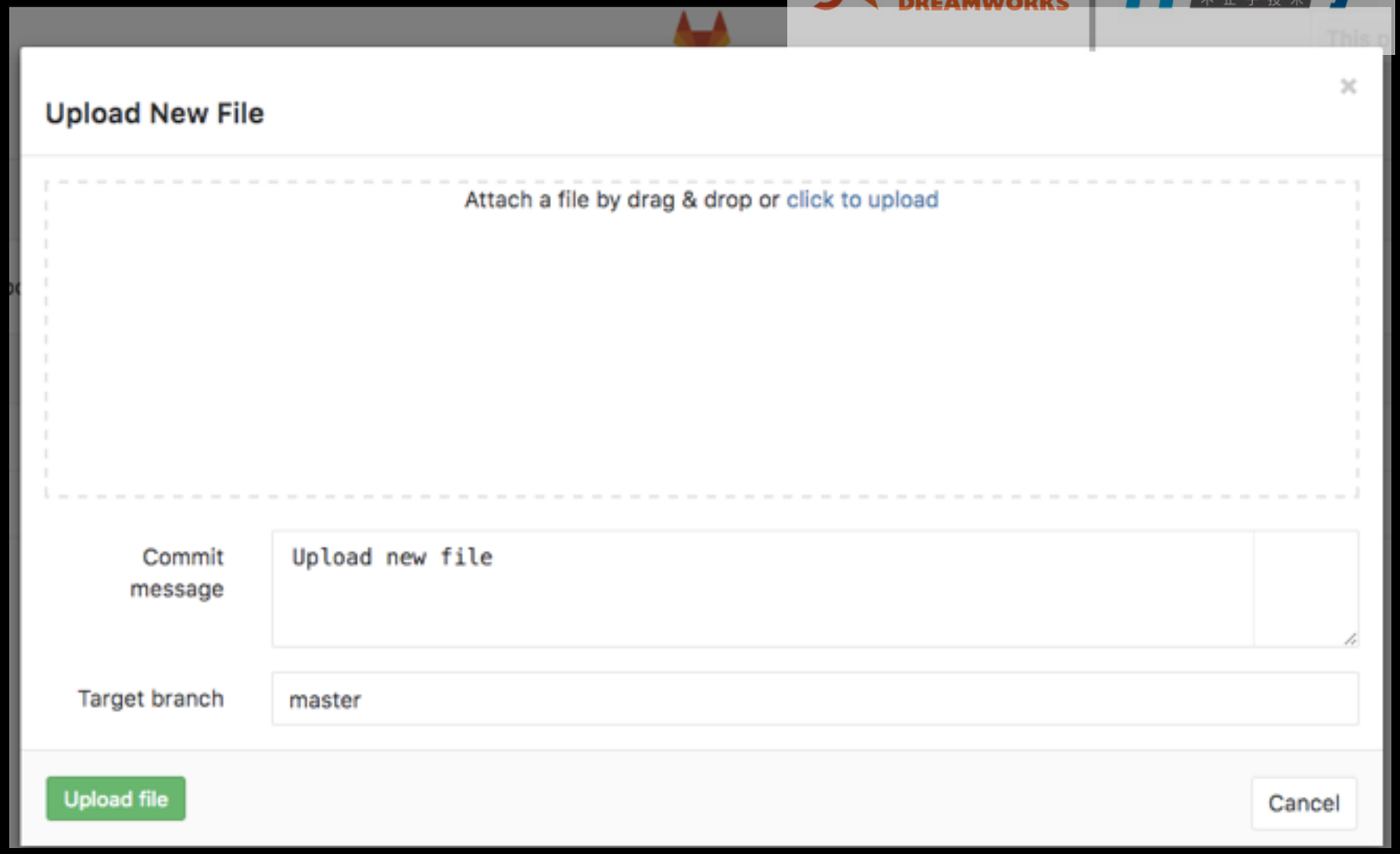
	A	B	C	D	E
1	道具	等级	攻击	防御	闪避
2	小李飞刀	1	100	2	1.00%
3	小李飞刀	2	120	5	2.00%
4	小李飞刀	3	150	10	2.50%
5	小李飞刀	4	200	20	3.50%
6	小李飞刀	5	300	30	3.80%
7	小李飞刀	6	350	50	4.00%
8	小李飞刀	7	500	80	4.20%
9	小李飞刀	8	1000	100	4.50%
10	小李飞刀	9	1200	200	4.80%
11	小李飞刀	10	1500	400	5.00%

```json
{
  "10001": {
    "name": "小李飞刀",
    "level": {
      "1": [100, 2, 1.00],
      "2": [120, 5, 2.00],
      "3": [150, 10, 2.50],
      "4": [200, 20, 3.50],
      "5": [300, 30, 3.80],
      "6": [350, 50, 4.00],
      "7": [500, 80, 4.20],
      "8": [1000, 100, 4.50],
      "9": [1200, 200, 4.80],
      "10": [1500, 400, 5.00]
    }
  }
}
```

数值工程师配表

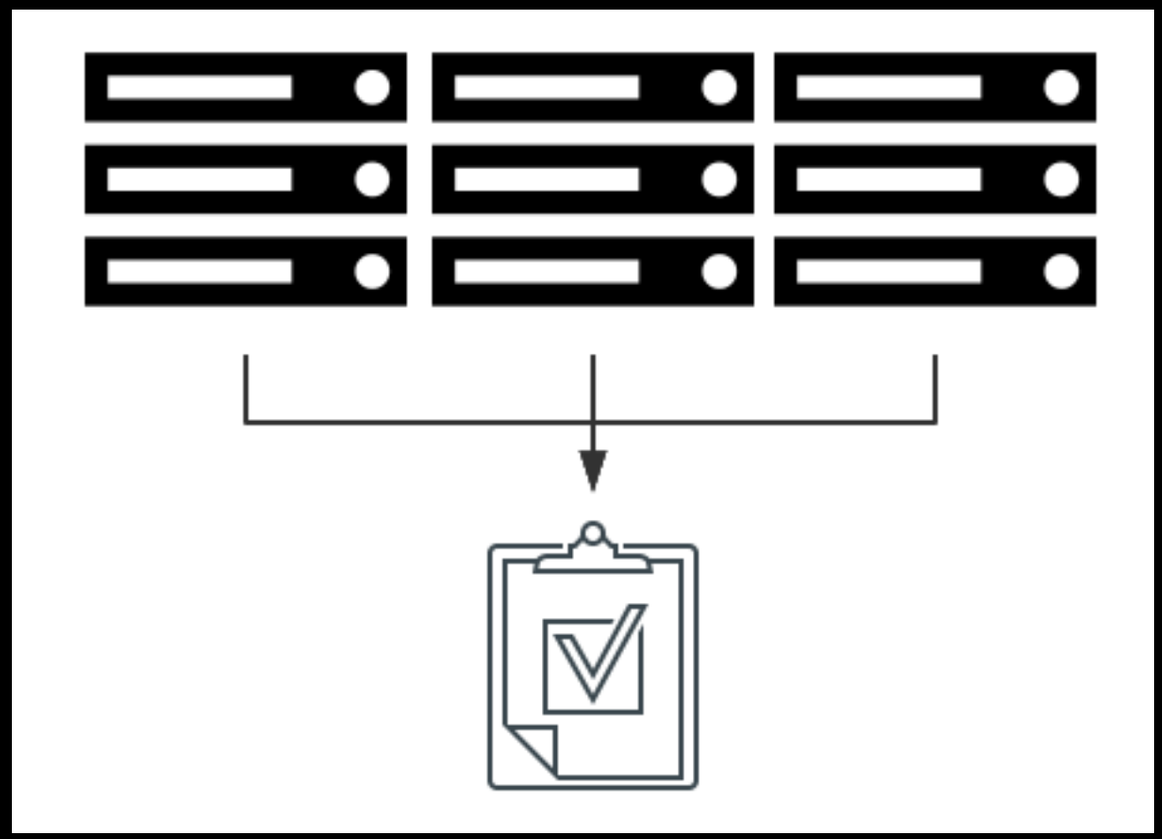
数值工程师通过修改 Excel 文件，更新数值配置。





设计师发布 CDN 资源

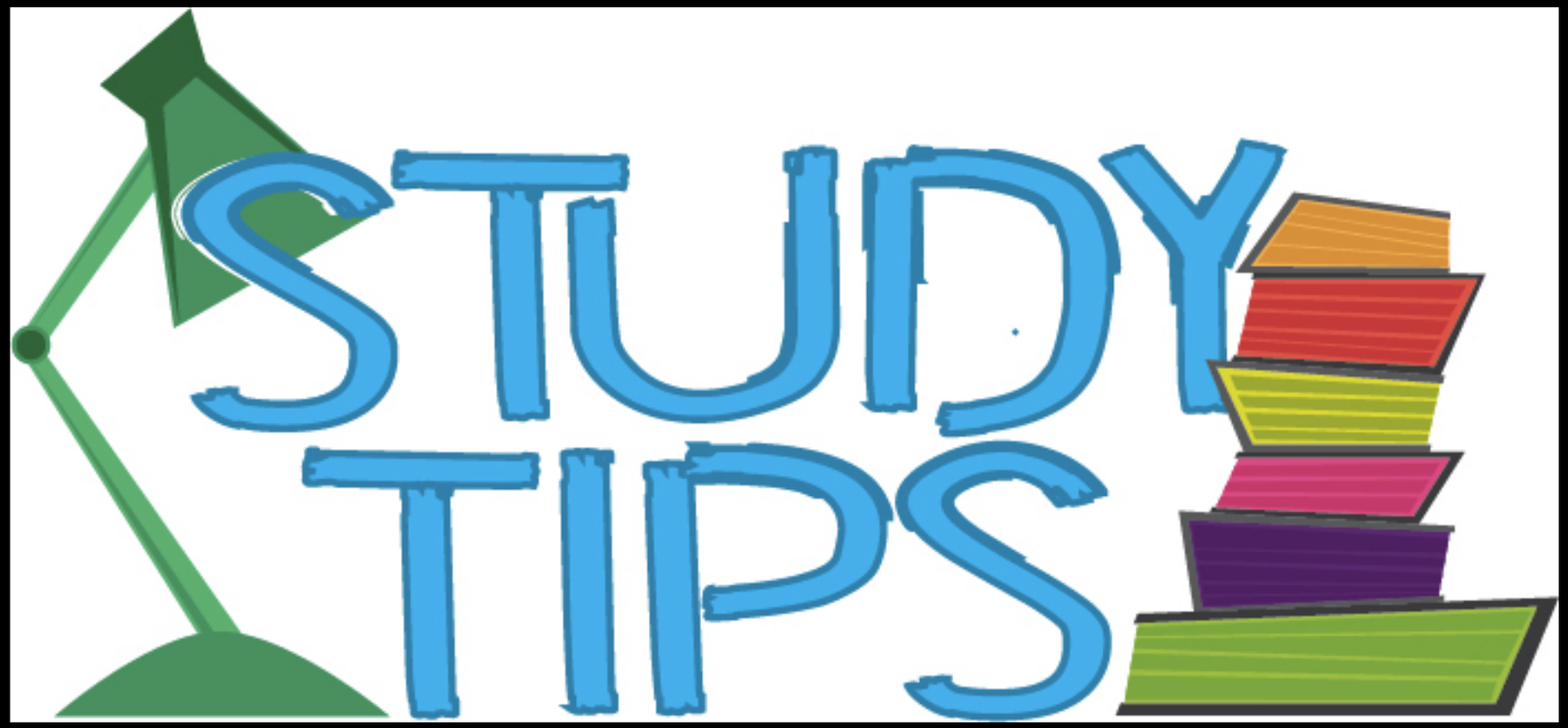
在 GitLab 中可直接拖拽文件上传。转码、部署自动完成



集群服务自动部署和测试

高并发的 Web 应用，通常都有很多分片（可以理解为多个主机）。代码需要同步到各个分片上，而各个分片可能有微小差异，不一定每次代码迭代全都能正常运行。我们将每一个分片提出一个测试端口，上线前各个分片均做一次测试用例覆盖，确保集成服务的稳定性。

使用成本

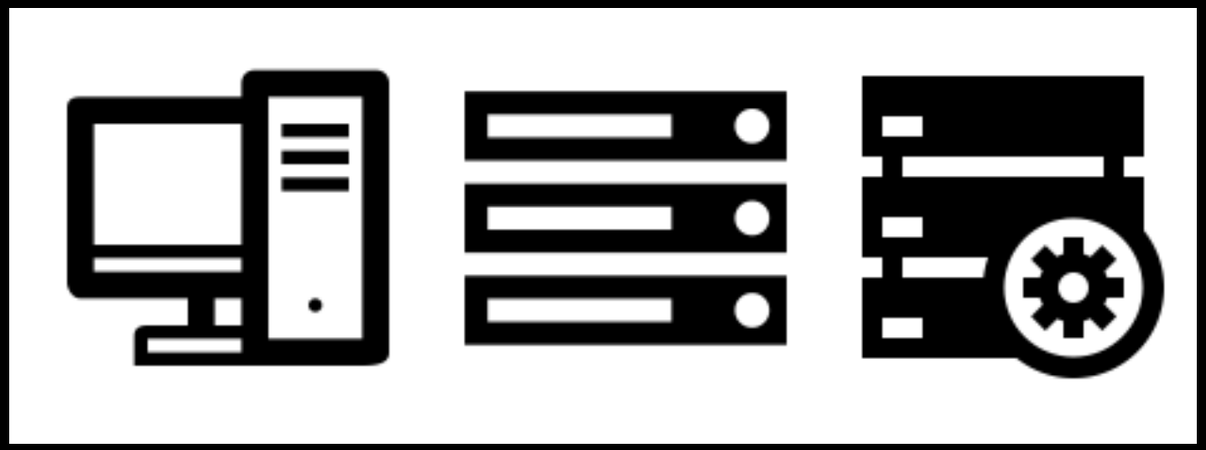


学习和使用成本

持续集成几乎覆盖了开发环节和运行环境方方面面，普通项目组成员不一定都能接触。
所以我给组内的同学下放更多的内网环境权限。当然我们也可以自行安装相关环境。

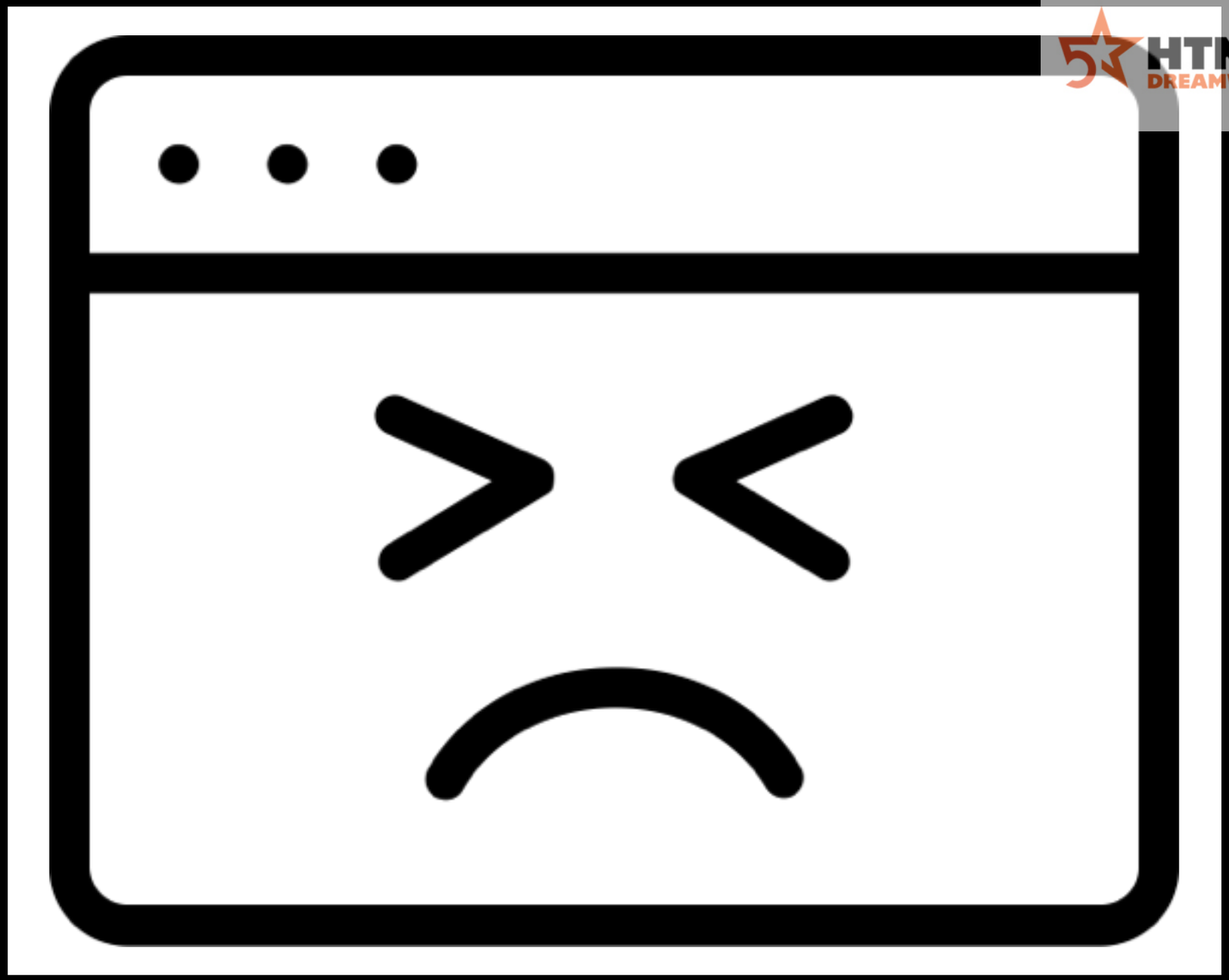
线上环境隐私保护

线上环境的操作需要十分谨慎，一些配置有很高的保密性。包括不限于：第三方支付授权码、第三方应用授权码、文件部署授权码、数据库用户身份，即：各种重要的私密配置。



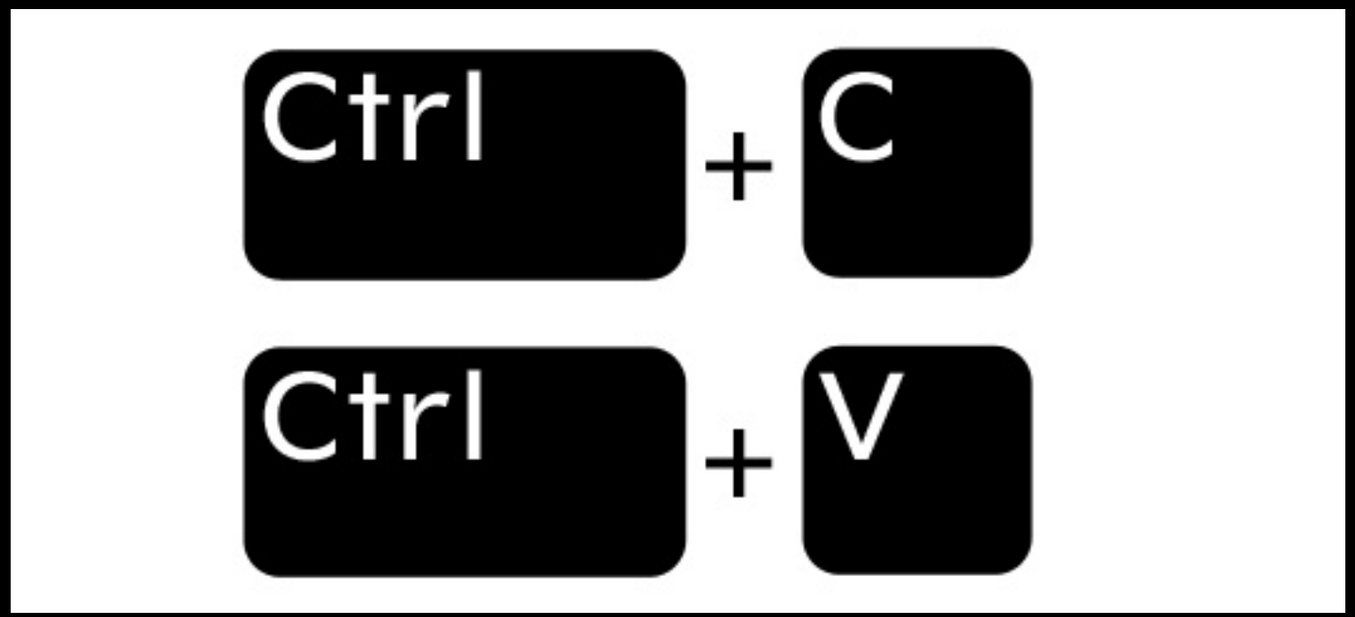
区分不同运行环境

本机运行、开发环境（个人开发环境、稳定版、开发版）、线上环境（预上线、灰度上线），都需要通过配置或环境变量区分。



构建过程自身异常

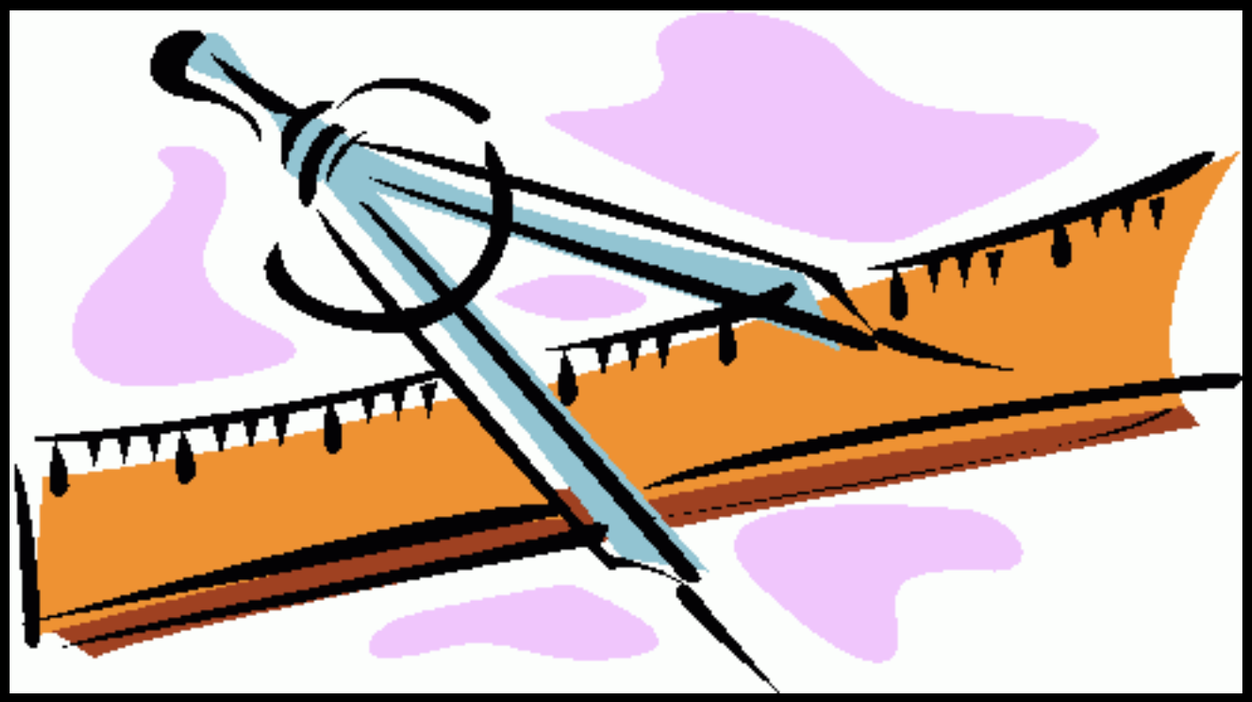
就构建本身也可能出现异常。如：构建机器软硬件异常（网络中断、磁盘满了、编译依赖升级失败）。还有节假日不在办公环境。



错误覆盖线上项目的风险

克隆构建任务也是有风险的，因为有相同的部署配置，处理不好会覆盖之前的线上代码，导致线上事故。

实践经验



项目规范

无论是前端项目还是后端项目（PHP、NodeJS、Go），我们均用 package.json 来定义。方便统一项目名称、版本、构建脚本的来源。

构建过程使用跨平台的脚本

可以选用 PHP、NodeJS、Python 等跨平台的脚本，方便运行到各种环境中。不建设使用 VBScript 或 JScript，仅能在 Windows 直接运行的脚本。

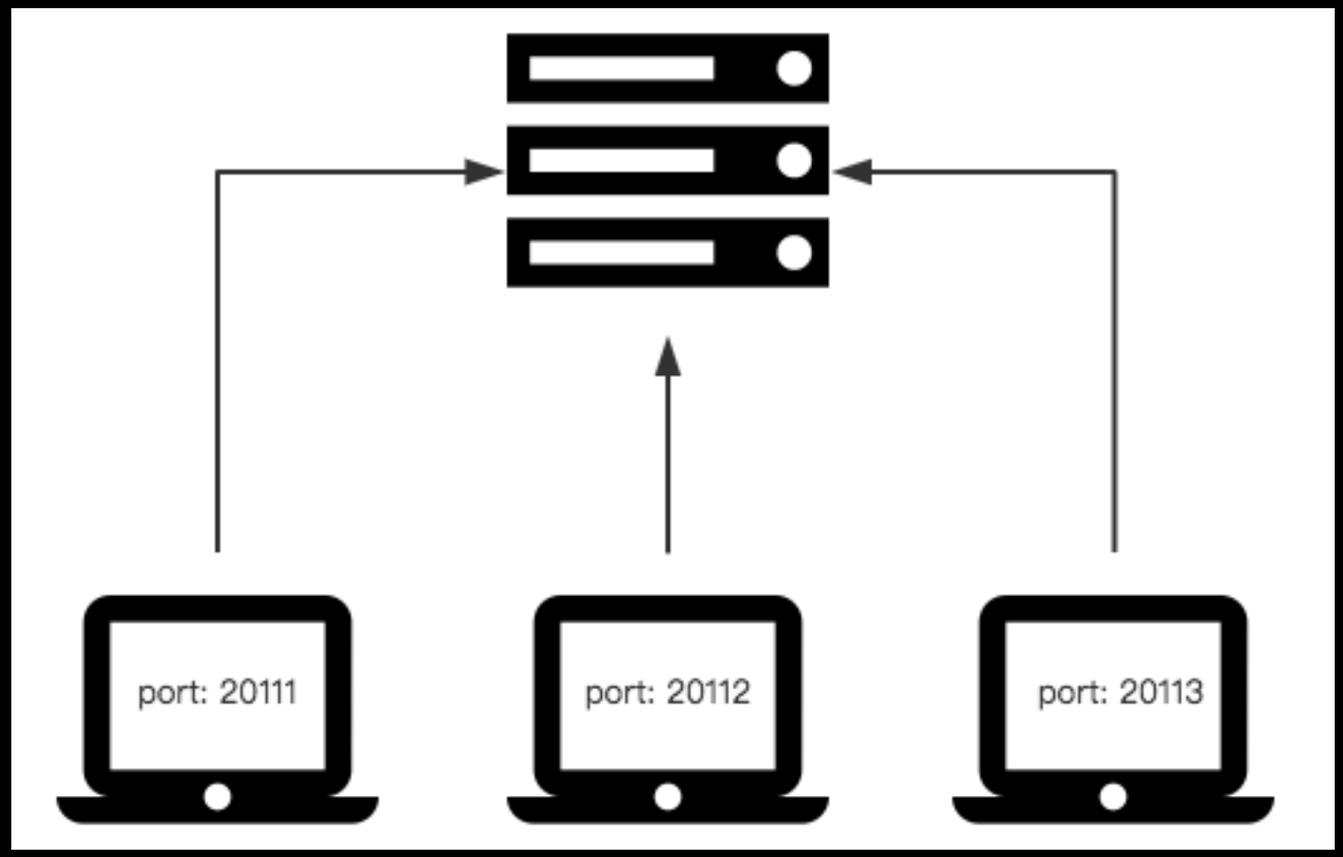
```
if (calc(1, 2, 3) !== 6) {
  process.exit(1);
}
```

```
$ cat test.sh
node -e 'console.log("1");process.exit(1)'
node -e 'console.log("2");process.exit(1)'
node -e 'console.log("3");process.exit(1)'

$ sh -e test.sh
1
```

编写小成本测试用例

编写测试用例也不一定要引入重型的测试框架，其实只要进程以非零状态退出就可以中断构建过程。NodeJS 用 process.exit(1);，PHP 用 exit(1);



手动构建

为避免开发期成员部署项目互相干扰，特给每个成员分配一个性端口。代码不需要提交到仓库，通过手动部署相应项目。

总结

HTML5 DREAMWORKS
IT大咖说 不止于技术