

远程开发和 CI 一回事

黎志航 腾讯云开发者 2024年10月23日 08:45 北京



微信扫一扫
关注该公众号





研发技术 看腾讯经略 | 腾讯技术人原创集 | 涨研发技


👉 目录

- 1 为什么远程开发和 CI 是等效的
- 2 CNB 远程开发及其原理
- 3 CNB 远程开发为什么启动这么快？
- 4 CNB 远程开发如何在并发场景下做到 100% 增量编译？
- 5 总结

在前面的文章中，我们分别介绍了在 CI 中如何利用 CNB 的 `git-clone-yyds` 插件，实现秒级克隆，以及在 CI 中如何利用 CNB 的 `Volume` 缓存，实现 100% 增量编译以加速构建。

但你有没有想过，这些技术同样可以应用于远程开发中的？

关注腾讯云开发者，一手技术干货提前解锁👉



腾讯云开发者

腾讯云官方社区公众号，汇聚技术开发者群体，分享技术干货，打造技术影响力交... >

925篇原创内容

公众号

想象这样一个场景：你打开 CI 流水线的命令行，用 `vim` 修改了几行代码，然后提交，你无法通过任何本地命令，区分你是在开发写代码，还是在构建。

感觉 `vim` 编辑代码不方便？

改成 `VSCode` 来编辑是不是更顺手一些？

恭喜你，发明了远程开发 ！

CI 和远程开发本质上是等效的。

试想，你打开远程开发的 IDE → 克隆代码 → 安装依赖 → build 构建，然后立马关闭 IDE，是不是和 CI 非常相似？

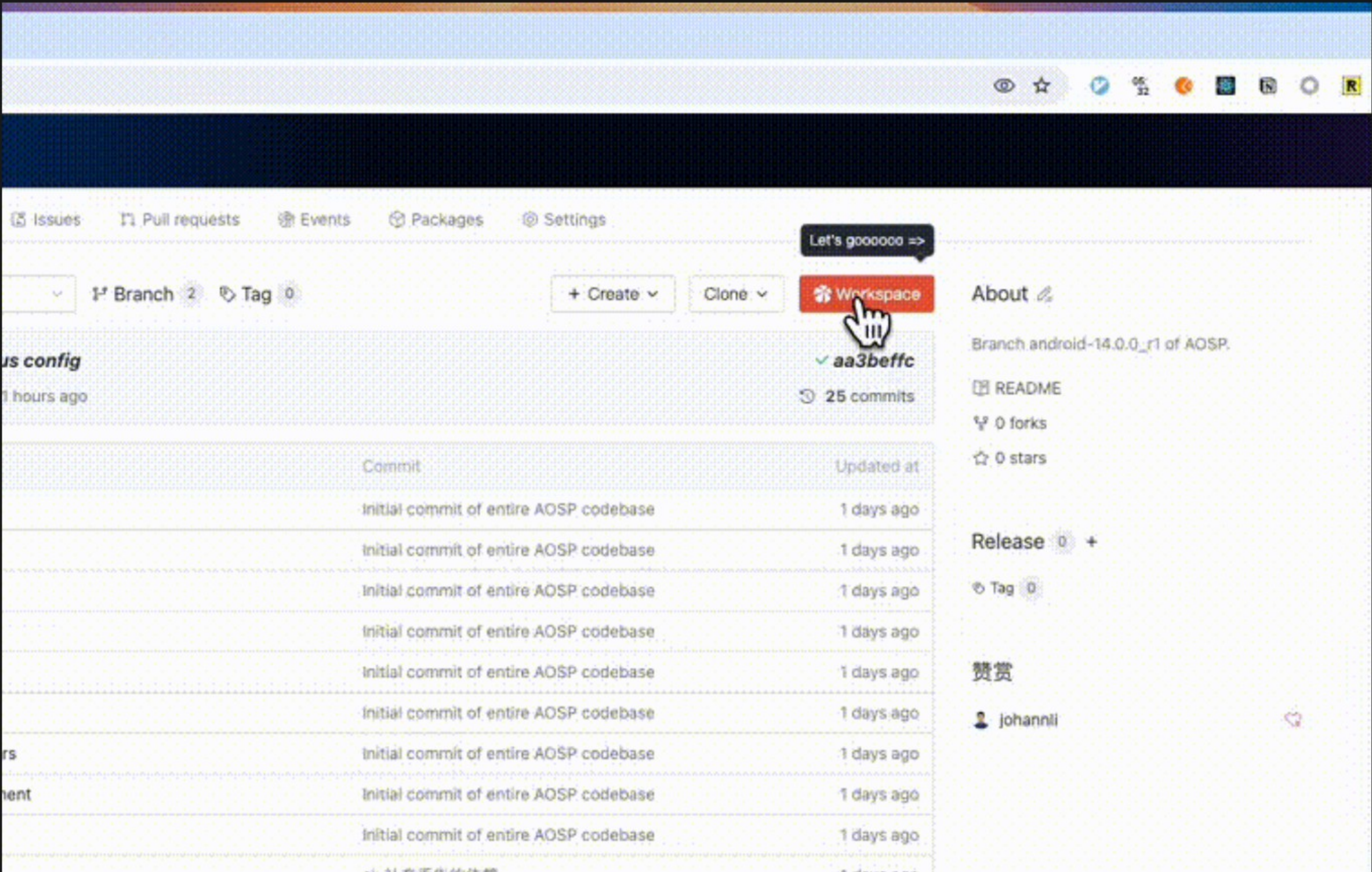
把上面的流程自动化了，就是 CI 。

01

为什么远程开发和 CI 是等效的

在软件开发中，远程开发和 CI 都使用 Git 作为代码管理的基础。因此，不论是在 CI 里面 clone 代码，还是在远程开发环境 clone 代码，然后用 VSCode 进行开发，最后提交代码，都是以 Git 为基础。

所以，如我们前面的文章提到的 CNB（云原生构建 <https://cnb.cool>）在 CI 使用 git-clone-yyds 插件用来做代码「秒级克隆」，秒级完成工作区的准备，在远程开发上同样适用的，这是远程开发秒级启动的基础，所以在代码这一层，远程开发和 CI 是等效的。

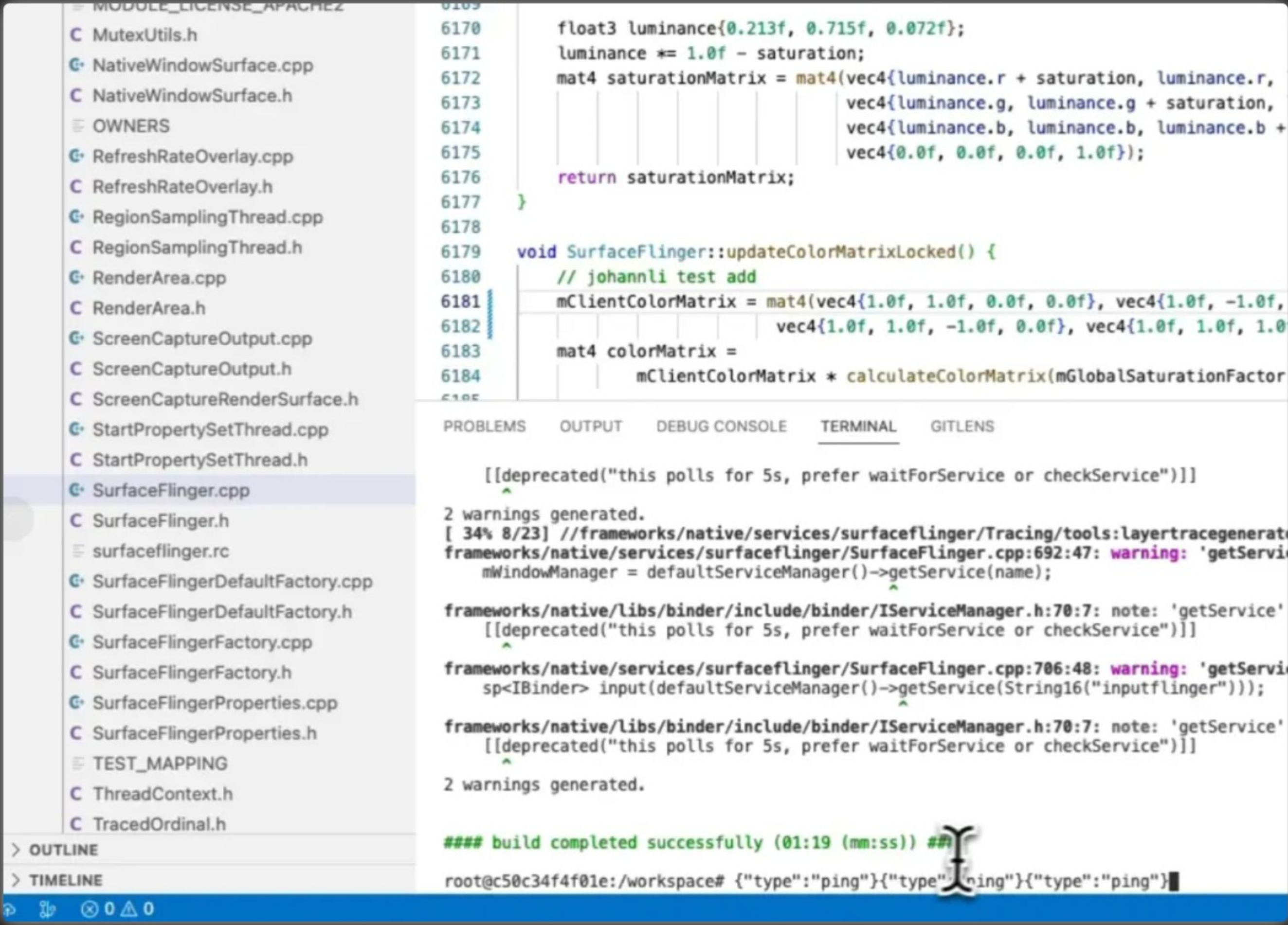


125 G 的 AOSP 项目打开远程开发的速度

远程开发与 CI 的等效性还体现在其对环境配置的处理上。在远程开发上，配置环境并安装依赖以保证代码可以在开发环境运行；而 CI 在每次构建过程中，也会完成相同的环境配置和依赖安装，以便后续进行构建和测试。在 CNB 上用 Dockerfile 来配置开发环境，可以进一步保证构建的一致性，不会出现在本地构建成功，在流水线编译失败的情况。

在编译/测试流程中，远程开发和 CI 的等效性尤为明显。远程开发编码完成需要编译和测试。以 125G 大小的 AOSP 为例，本地编译需要 46 分钟。而在 CI 中，可以使用 CNB 的 Volumes 缓存，将时间缩短到约 1 分钟。这一特性在远程开发中同样重要 —— 没人愿意每次改代码后等待 46 分钟才看到结果。CNB 的 Volume 缓存在远程开发中也适用，且具有并发性，多人同时进行远程开发时，可并发命中缓存。编码完成直接构建能 100% 命中缓存，1 分钟内完成增量编译。

因此，在编译层面，远程开发和 CI 也是等效的。



CNB 远程开发增量编译 AOSP 耗时：1 分 19 秒

02

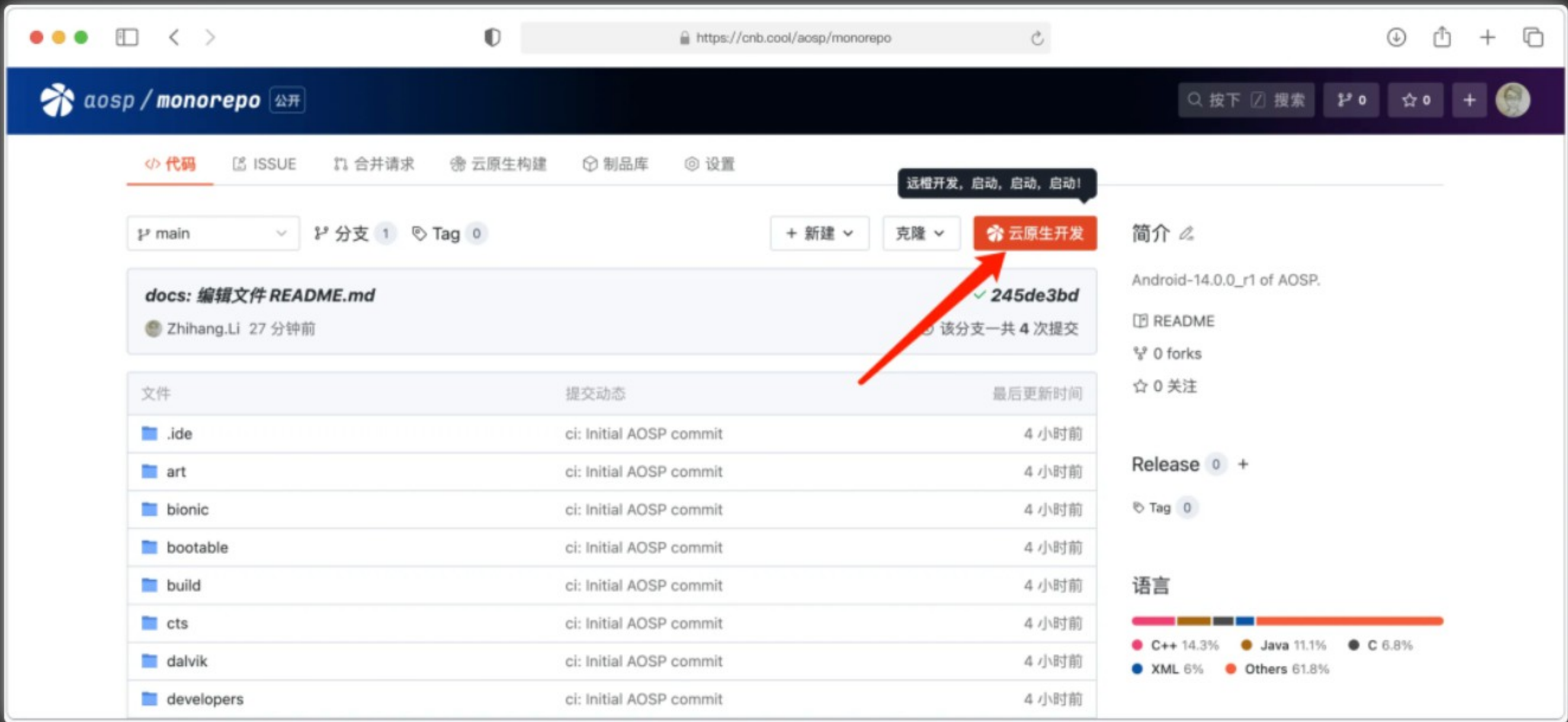
CNB 远程开发及其原理

CNB（云原生构建 <https://cnb.cool>）的远程开发，是基于云原生构建的远程开发解决方案，支持 Web 和 VSCode 客户端进行远程连接。

与传统远程开发相比，CNB 的远程开发有很多不同：

- 声明式：基于 Dockerfile 生态的开发环境，配置与代码同源管理
- 快速启动：即使是超大仓库，也可以数秒准备好代码和工作区。
- Volumes 缓存 ：多人同时开发，也能命中缓存。
- 配置文件漫游: 用户对于自己的远程开发 VSCode 的配置可漫游。
- 按需使用：分支即环境，按需获取开发资源，闲时快速回收，避免资源浪费。

CNB 的远程开发入口在代码仓库上，方便管理和使用，一键启动远程开发。



CNB 的远程开发和 CI 的配置一样，都是声明式的，与代码同源管理。

可在仓库根目录下增加 .ide/Dockerfile 文件，在 Dockerfile 中自由定制开发环境， 启动开发环境时会优先使用 .ide/Dockerfile 构建一个镜像，作为开发环境基础镜像。

然后在 .cnb.yml 的配置中可以自定义创建流程、资源规格等参数：

```
1 # .cnb.yml
2 $:
3   vscode:
4     - runner:
5         # 声明需要的开发资源
6         cpus: 64
7       docker:
8         # 自定义开发环境
9         build: .ide/Dockerfile
10    services:
11      - vscode
12      - docker
13    stages:
14      # 自定义创建流程
15      - name: ls
16        script: ls -al
```

声明式配置有很多好处：

1. 版本控制与审计：Pipeline as Code 允许流水线配置与代码同源管理，提供了版本控制的能力。这是维护项目历史和确保流水线稳定性的关键。
2. 复用性和共享：使用代码形式的流水线定义，可以将通用的模式、模板和实践轻松地在多个项目或流水线之间复用和共享。这大幅提高了项目设置的效率和一致性。
3. 可维护性和透明性：以代码的形式维护流水线配置，使得理解、更新和维护变得更加方便。代码定义的清晰性和可读性促进了团队内外的透明沟通和协作。

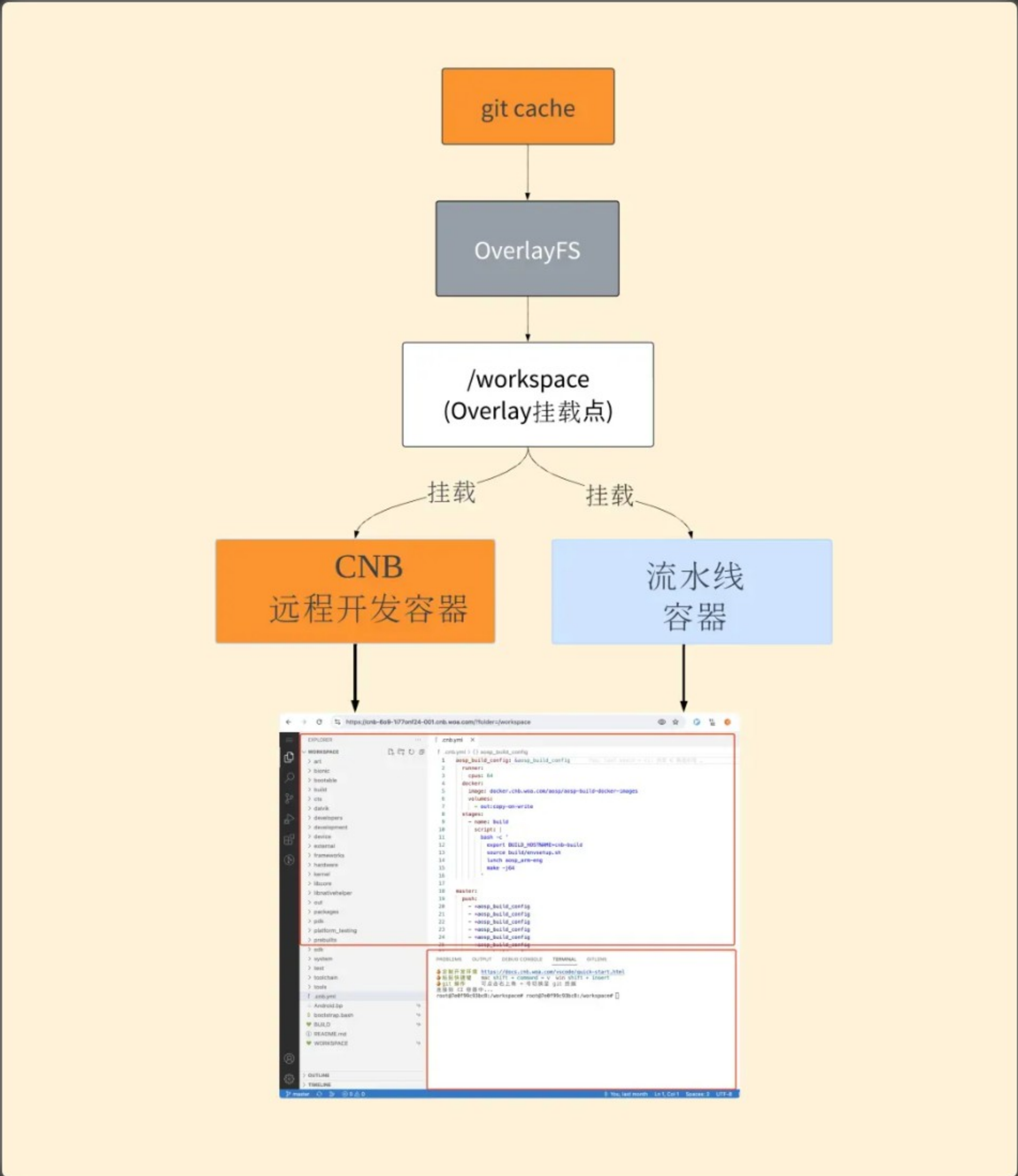
03

CNB 远程开发为什么启动这么快？

CNB 远程开发启动速度快的秘密在于克隆代码快、环境启动快，CNB 远程开发代码准备的逻辑与前面文章我们提到的「秒级克隆」的原理是一样的，也是使用 git-clone-yyds 插件，通过 CoW(Copy-on-Write) 在构建母机上对代码进行缓存，并且通过 volumes 分别挂载在远程开发容器和流水线容器上。

挂载到容器上是具有 CoW 特性的文件夹，修改它不会影响底层的缓存，所以代码的缓存是可以被并发使用的，因为在软件开发中，多人同时开发是客观存在的，所以远程开发的并发性是必须支持的特性。

远程开发容器和流水线容器，均基于 Docker 快速启动。通过在宿主机上缓存 Docker 镜像，进一步提升开发环境的启动速度。



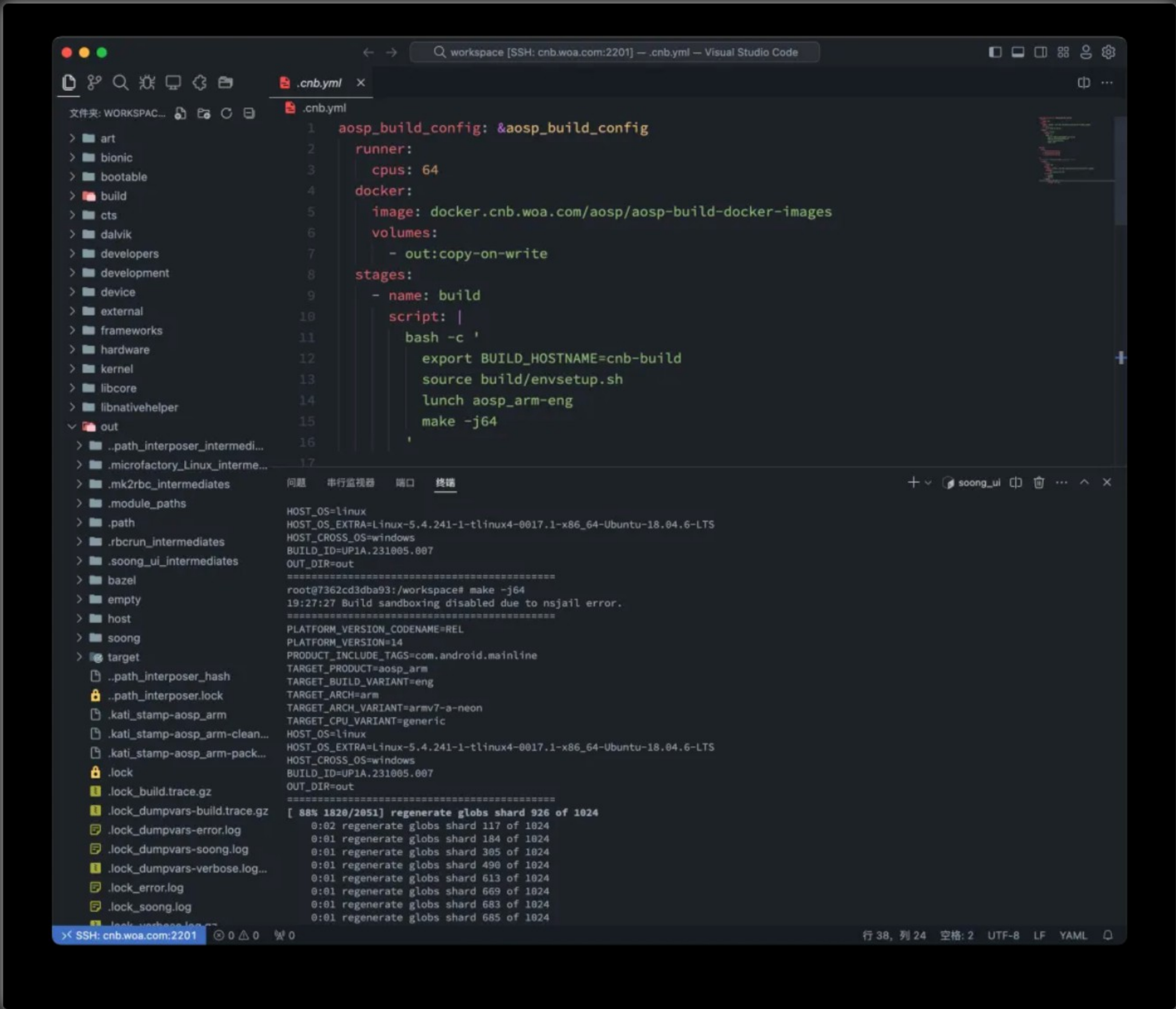
04

CNB 远程开发如何在并发场景下做到 100% 增量编译？

远程开发的可并发缓存，是 CNB 一个重要特性。

首先是远程开发也能使用 Volume 缓存，如前面文章提到在 CI 流水线上，我们可以通过 CNB 的 Volume 缓存，让全量编译转为增量编译，以加快编译速度。

这个特性在远程开发也是同样适用的，通过 Volume 缓存，在代码编写完成后，可以立即在远程开发 VSCode / web IDE 的终端上编译/调试。



另外，如前文提到，在远程开发场景下，多人同时进行同一个项目的开发，这种情况是事实存在的。所以要求在远程开发中，缓存需要具有可并发性。CNB Volume 缓存基于 CoW(Copy-on-Write) 在母机上创建具有 CoW 的文件夹，并且把上层的 merged 目录挂载到容器上，在容器上对缓存的修改，不会影响到 CoW 底层的构建产物的缓存, 多人同时远程开发时，可以并发使用缓存，而不会冲突和干扰。

这种基于 CoW 的缓存机制不仅提高了并发开发的效率，还确保了每个开发者都能享受到快速的增量编译体验。

05

总结

我们探讨了远程开发和CI之间的等效性，以及 CNB 如何利用这种等效性来提供高效的远程开发解决方案。我们分析了 CNB 远程开发的核心优势，包括声明式配置（as code）、快速启动、增量编译和资源优化等。


大家可以看到，Git、CI、远程开发有着密不可分的关系：CI 是消费代码变更、远程开发产生代码变更、生产者和消费者通过 GIT 解耦、远程开发和 CI 是一回事，从而实现统一。

因此，他们在产品形态上应该是一个整体呈现出来。

程序员的时尚单品：CNB

developers after 6 months of
experience

gitlab+jenkins
+vscode+copilot



developers after 20 years of
experience

CNB



-End-

原创作者 | 黎志航

感谢你读到这里，不如关注一下？👉



腾讯云开发者

腾讯云官方社区公众号，汇聚技术开发者群体，分享技术干货，打造技术影响力交流社区。 >
925篇原创内容

公众号

📢欢迎加入腾讯云开发者社群，享前沿资讯、大咖干货，找兴趣搭子，交同城好友，更有鹅厂招聘机会、限量周边好礼等你来~



(长按图片立即扫码)

精品 · 知识推荐

程序员必备Linux性能分析工具和方法

一文搞懂大模型！基础知识、LLM应用

服务端开发必备：9大性能优化秘技

赛博玄学，一键三连少一个Bug!

为好文章



点



收藏

点亮

