Github Codespaces 调研

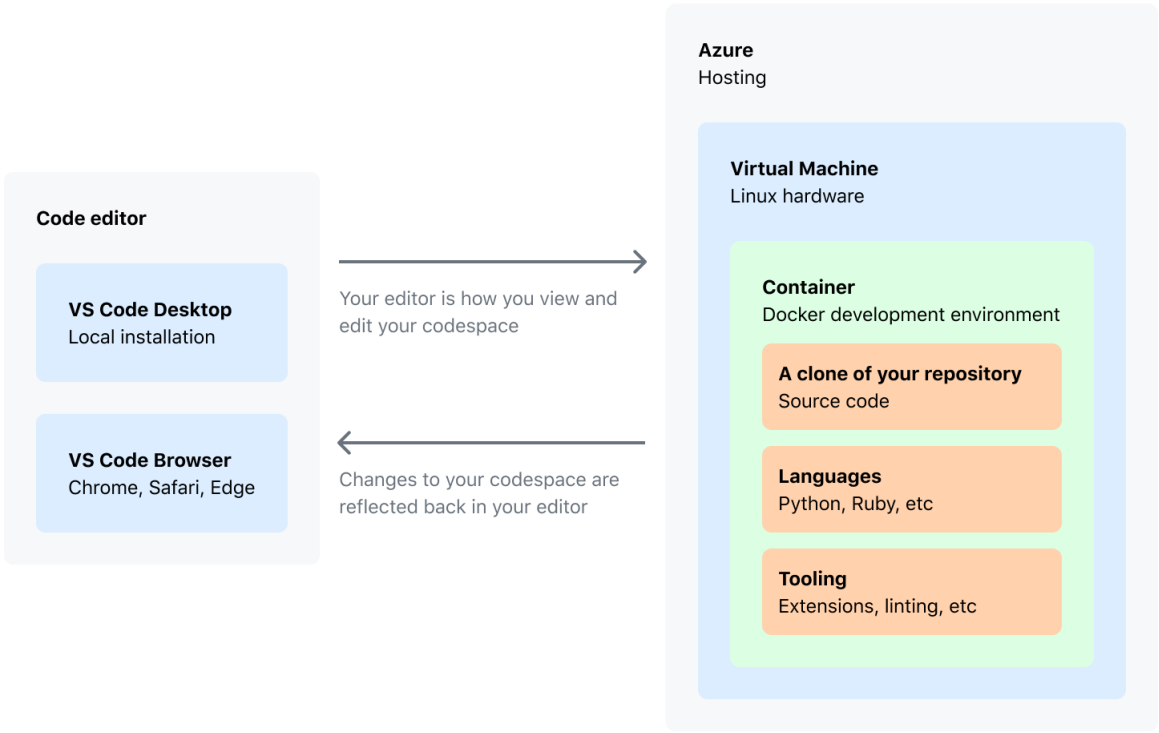
一、简介

CodeSpaces（代码空间）是一个托管在云中的开发环境。 用户可以通过将对应的配置文件提交到代码仓库中来自定义 Codespaces 的项目，这将为项目的所有用户创建可重复使用的代码空间。

Codespaces 运行在 Github 托管的各种基于 VM 的计算资源上，目前个人用户可以免费申请 4c 8g 的环境，同时该功能目前为 Beta 测试阶段，需要进行申请才能使用。

> Build software better, together
>
> Join the Codespaces beta waitlist to get a cloud development environment you can access from anywhere. We'd love to hear your feedback while you're trying this new feature. You can't
>
> https://github.com/features/codespaces/signup

Codespaces 目前是针对 Github 仓库来生成的，具体工作原理如下图所示，生成后用户可以通过桌面端的 VS Code 或浏览器版本的 VS Code 连接到其中。

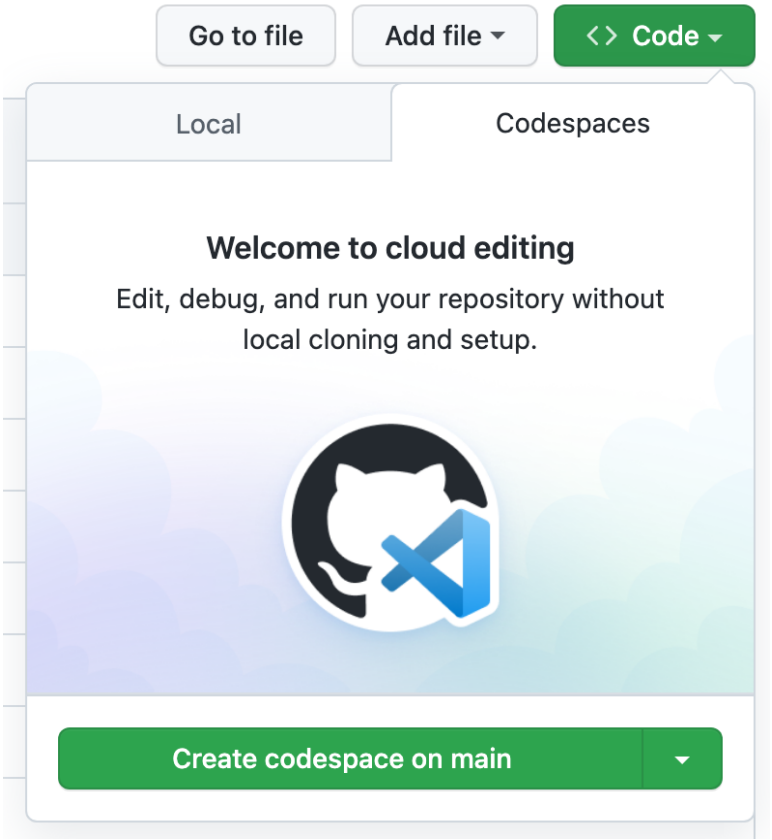


二、上手使用

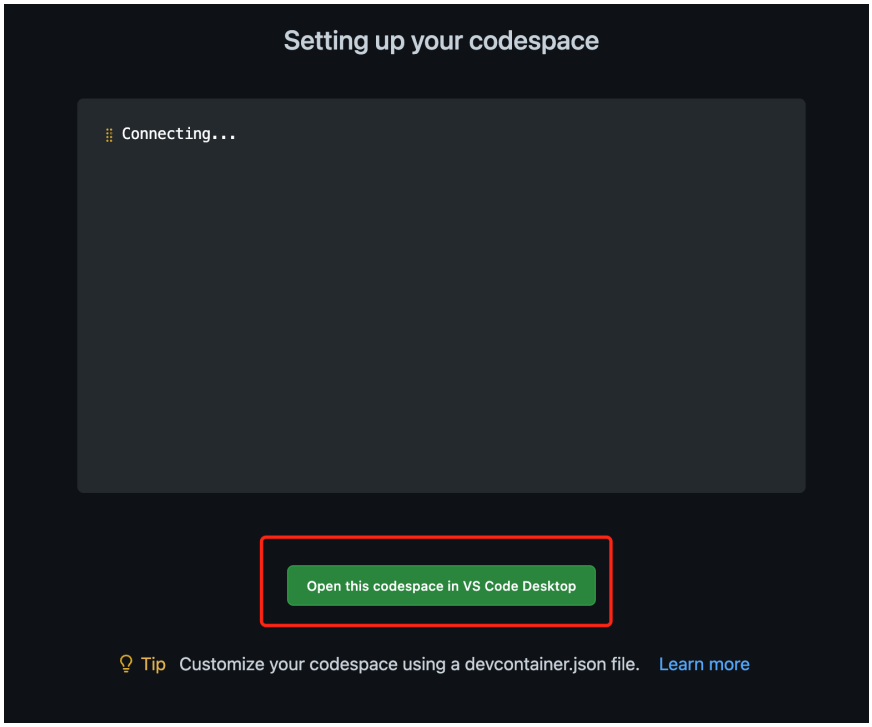
1. 配置环境

默认配置

首先需要打开一个目标的 Github 仓库，使用 **Code** 下拉菜单，然后在 **Codespaces（代码空间）** 选项卡中，单击 **Create codespace on main（在主分支上创建代码空间）**开始创建。



创建完成后就会自动跳转到浏览器版本的 VS Code，并已经连接到该代码空间中，如果希望使用桌面版的 VS Code 则点击下方的按钮即可。



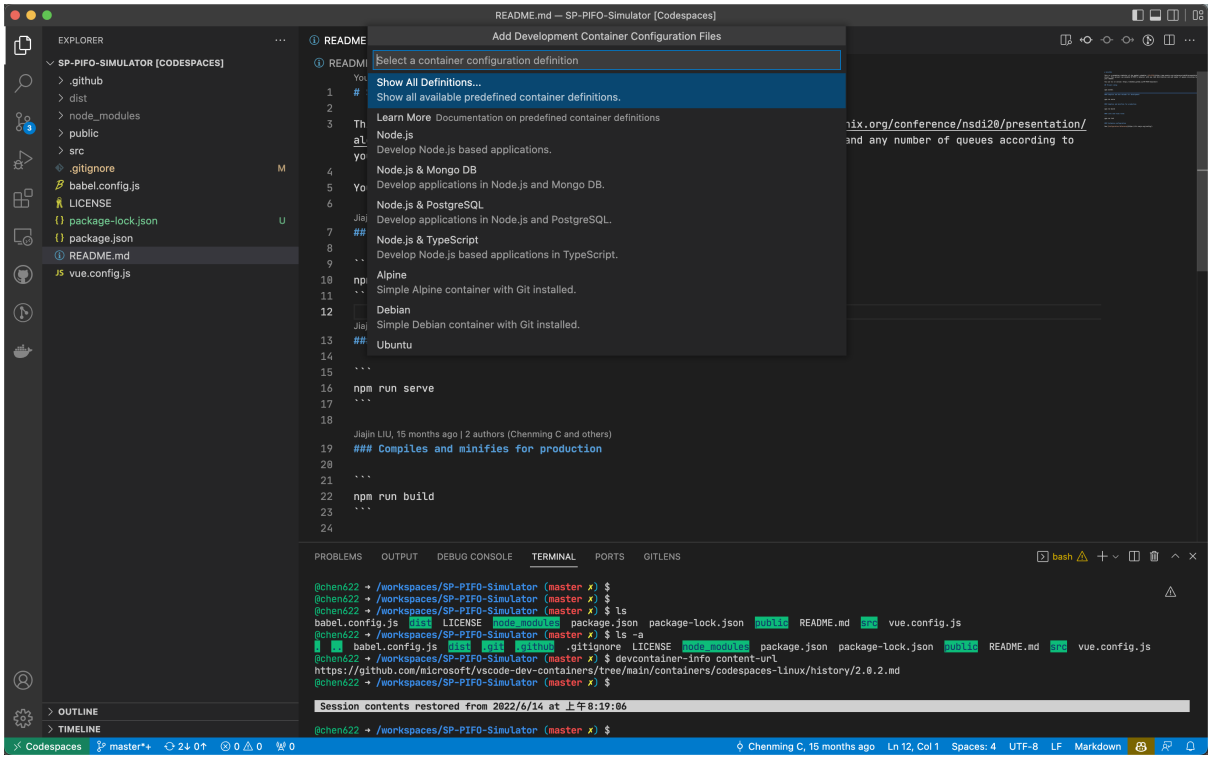
这时候生成的 Codespaces 是基于默认配置的，其中包含在项目开发时可能需要的许多常用工具，具体要查看包含的所有语言、运行时和工具，请在代码空间终端内使用 `devcontainer-info content-url` 命令，其会提供一个链接可以查看当前环境的配置信息。

预定义配置

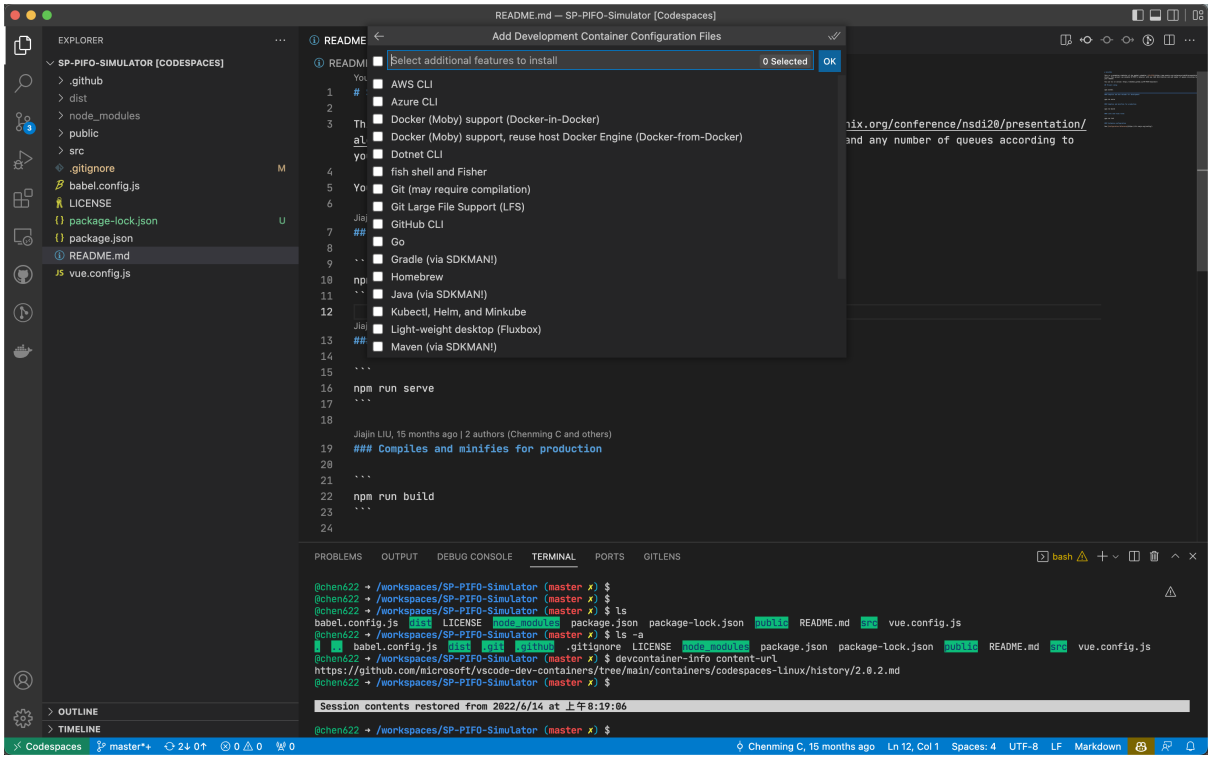
如果需要更改开发环境，可以从一些预定义的配置中选择，这些配置为特定项目类型提供了常见设置，并可以帮助用户快速开始使用已具有相应容器选项、VS 设置和应安装的 VS 代码扩展的配置。

具体配置流程如下：

1. 首先访问 Visual Studio 代码命令面板 (`Shift + Command + P`)，然后开始键入“dev container”。 选择 **Codespaces: Add Development Container Configuration Files...**，如果没有该选项需要先安装 VS Code 扩展 **GitHub Codespaces**。
2. 之后其会出现一些可以使用的预定义配置



3. 还可以选择安装一些其他软件



配置完成后，其会在项目根目录下创建一个 `.devcontainer` 的文件夹，其中包含了该环境的配置信息，主要有两个文件：

- `devcontainer.json` ：其中包含了 Codespaces 的基础配置信息
- `Dockerfile`：其指定了容器的一些具体配置内容，用户可向其中添加进一步的配置

生成完配置信息后，需访问 Visual Studio 代码命令面板，然后开始键入“rebuild”。 选择 **Codespaces: Rebuild Container** 进行重新构建。

自定义配置

这里以一个需要 DPDK 依赖的 CMake 开发环境举例，包括以下配置文件：

`devcontainer.json`

```
{
  "name": "C++",
  "build": {
    "dockerfile": "Dockerfile", // 基于 Dockerfile 进行进一步配置
    "args": {
      "VARIANT": "ubuntu-22.04" // 选择容器的系统版本（可选 Debian·Ubuntu·Mac）
    }
  },
  "runArgs": ["--cap-add=SYS_PTRACE", "--security-opt", "seccomp=unconfined"],

  "settings": { // 同步的 VS Code 配置信息
    "C_Cpp.codeAnalysis.clangTidy.enabled": true,
    "C_Cpp.clang_format_path": "/usr/bin/clang-format",
    "C_Cpp.clang_format_fallbackStyle": "none",
    "C_Cpp.clang_format_style": "Google",
    "C_Cpp.codeAnalysis.clangTidy.path": "/usr/bin/clang-tidy",
    "C_Cpp.files.exclude": {
      "**/.vscode": true
    },
    "C_Cpp.default.cppStandard": "c++11",
    "C_Cpp.default.cStandard": "gnu99",
    "C_Cpp.default.intelliSenseMode": "clang-x64",
    "C_Cpp.codeAnalysis.exclude": {
      "**/build": true
    },
    "C_Cpp.formatting": "clangFormat"
  },

  // 需要安装的 VS Code 插件
```

```
  "extensions": [
    "ms-vscode.cpptools",
    "ms-vscode.cmake-tools",
    "twxs.cmake",
    "ms-vscode.cpptools-extension-pack",
    "mads-hartmann.bash-ide-vscode",
    "redjue.git-commit-plugin",
    "GitHub.copilot"
  ],

  // 需要从云端映射到本地的端口，可用于开发前后端项目
  "forwardPorts": [],

  // 容器创建完成后需要执行的命令
  "postCreateCommand": "gcc -v",


  "remoteUser": "vscode",

  // 需要自动安装的依赖软件
  "features": {
    "git": "latest"
  }
}
```

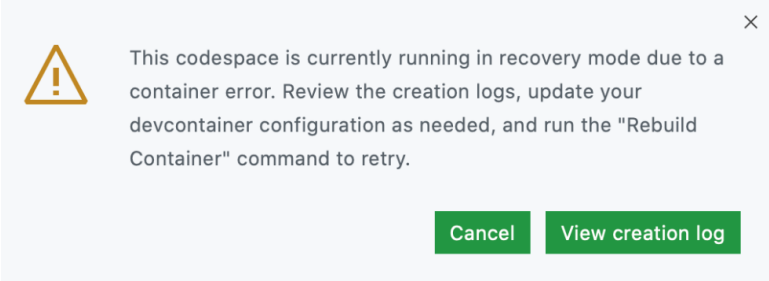
`Dockerfile`

```
# 安装 clang 相关依赖
RUN apt-get update && apt-get -y install clang-tidy clang-format

# 安装 DPDK
RUN apt-get -y install python3 python3-pip
COPY ./install-dpdk.sh /tmp/
RUN chmod +x /tmp/install-dpdk.sh && /tmp/install-dpdk.sh && rm -f /tmp/install-dpdk.sh
```

2. 异常恢复

如果出现因配置错误导致容器创建失败，VS Code 会自动进入恢复模式，用户可以看到错误的相关日志，并根据日志修复配置文件中的错误以进行重新构建。



三、总结

总体来说，Github Codespaces 给了我们一个通用且便捷的开发环境，适合于开发需要 Linux 环境的项目，同时借助浏览器端的帮助，我们甚至可以在 iPad 上进行简单的代码修改。最重要的是，其很适合用于开源项目，当其他人想要对代码进行修改时，无需进行繁琐的环境配置即可编译并运行。