

# Amphitheatre

开箱即用的云开发环境



# 软件开发过程中面临的挑战

## • 技术栈杂多

开发工程师需要在本机安装各种编程语言的运行时和相关的框架以及库，随着业务规模的增长，多语言多版本的维护和管理变得越来越复杂。

## • 研发流程繁长

研发流程长达十几个步骤：安装开发软件、配置环境、克隆代码、开发、本地调试、提交代码、编译构建、自动化测试、部署到测试环境、测试验收、合并代码到主线、部署到生产环境...

## • 复杂的基础设施

为了部署测试，不仅要学习容器化、Kubernetes，还要申请资源安装配置各项中间件，学习成本高，费心费力

一些运行在 Kubernetes 中的复杂微服务架构是 CPU 和内存密集型的，在某些情况下编译或测试可能非常耗时且占用大量资源。然而大多数工程师的标准设备是笔记本电脑，有 CPU 和内存的限制，编译一次的时间估计够喝好几杯咖啡。

你的团队是个大杂烩，Python、TypeScript、Golang 都做，此时有新人进来了，你准备让新人怎么开始？你觉得新人的心情是怎样的？

虽然说我们可以在测试服务器上进行调试，但整个流程也是比较漫长，「提交代码 -> 触发CI/CD -> 等待构建成功」，可能简单的 BUG 我们提交代码打个日志就能解决问题，当遇到复杂的 BUG 时通过这种方式在服务器上调试就非常难受了，太浪费时间了，「提交 -> 等待」，反反复复，始终没有本地开发工具直接调试的方便。

我在本地环境测试过...都正常的呀...为什么一到上线就不行了呢？

对于容器服务用户而言，代码改动操作都要求重新构建镜像和启动容器。改动频率增多造成多次构建镜像和启动容器，大大降低开发效率。

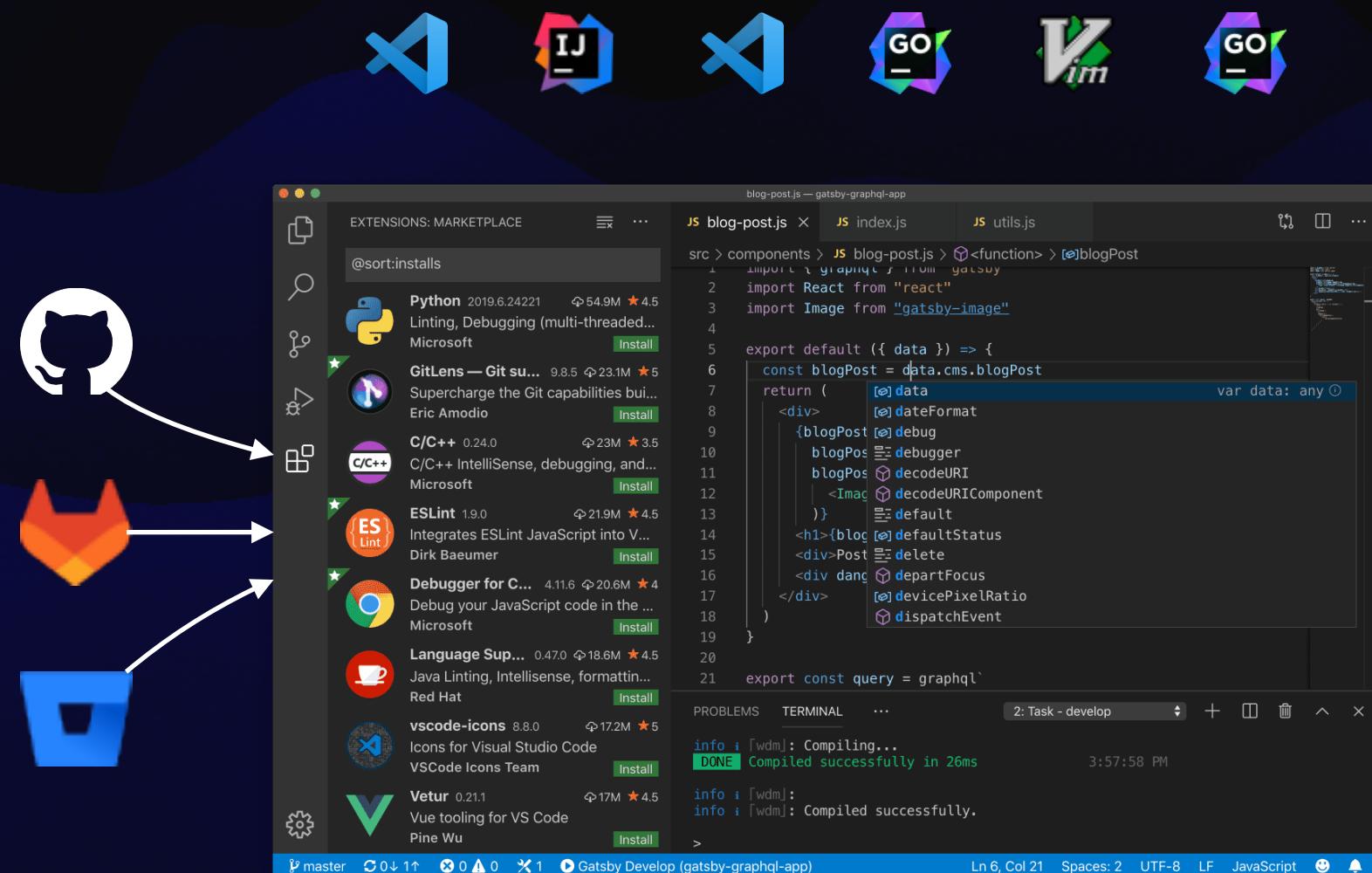
微服务后端开发的最大痛点之一就是调试困难，非常影响我们的开发效率。

对于一些初、中级程序员，想开发并部署一个中小应用（如开源项目的文档、个人博客、个人网站、在线简历和在线 ChatGPT 聊天工具等）还是有一定门槛的，需要先在电脑上装好对应的开发环境（如 Python、Java、Go、NodeJS 等），然后到 GitHub 上创建一个项目，拉到本地，开发完后 push 代码，再到阿里云买云主机，配环境、证书，绑 ssh key，拉代码、编译.....纯前端项目相对方便一点，可以本地编译传 CDN，或使用 GitHub Pages 服务等。总体来说，开发部署应用的过程费时费力，效率较低。

如果我们想与其他微服务进行联动调试，则需要在本地环境中启动对应的微服务模块，这可能需要大量的配置和构建时间，同时也会占用我们本地很多资源，可能还会出现“带不动”的情况。

你作为一位技术大拿，突然被临时指派做一个排期非常紧张的项目，但是你本地没有工作环境，安装那套环境又比较费时。果然...你在配置环境上花费了些许时间，大拿的技术反而没发挥出来，此刻你的心情又是怎样？

# Environment as a Service



主流编辑器 / IDE

与这个世界上最优秀的编辑器配合使用，无论  
是 VS Code、IntelliJ IDEA、Vim、Emacs...

拉取代码

开发代码

即时响应式构建

部署到测试网络

**可观测**

Quality Gate ? Passed

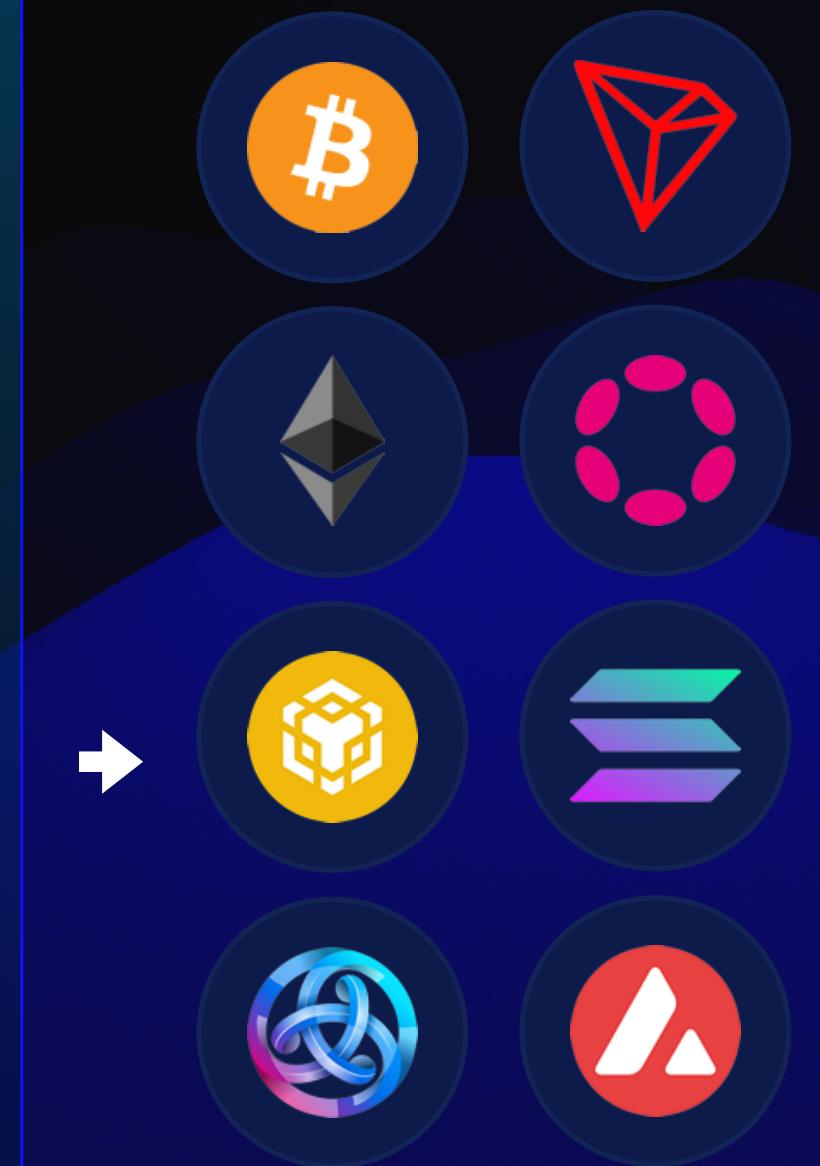
**智能代码分析**

**实时预览**

**远程调试**

**Amphitheatre 云端开发环境**

在云端启动新的开发环境，按需提供并预先配置了所需工具、库和依赖项，适用于 29+ 种编程语言和 4000 多种技术栈，可随时进行编码。



A screenshot of a GitHub repository page for 'amp-example-go'. The repository is public and has 1 branch and 0 tags. The 'Code' tab is selected. A commit by 'wangeguo' is shown, merging the 'master' branch from 'github.com:amphitheatre-app/amp-example-go' into the current branch. The commit was made 2 months ago with 10 commits. The page includes standard GitHub navigation links like 'Issues', 'Pull requests', 'Actions', 'Projects', 'Wiki', 'Security', 'Insights', and 'Settings'.

# 产品演示

演示视频无法播放时，可复制以下链接在浏览器中打开  
<https://amphitheatre.app/videos/demo.mp4>

This is a simple example based on:

- building a single Go file app and with a multistage `Dockerfile` using local docker to build
- tagging using the default tagPolicy (`gitCommit`)
- deploying a single container pod using `kubectl`

Inspired by <https://github.com/GoogleContainerTools/skaffold/tree/main/examples/getting-started>

 SLSA Generic generator [Configure](#)  
Generate SLSA3 provenance for your existing release workflows

 Go [Configure](#)  
Build a Go project.

 SLSA Go releaser [Configure](#)  
Compile your Go project using a SLSA3 compliant builder

[More workflows](#) [Dismiss suggestions](#)

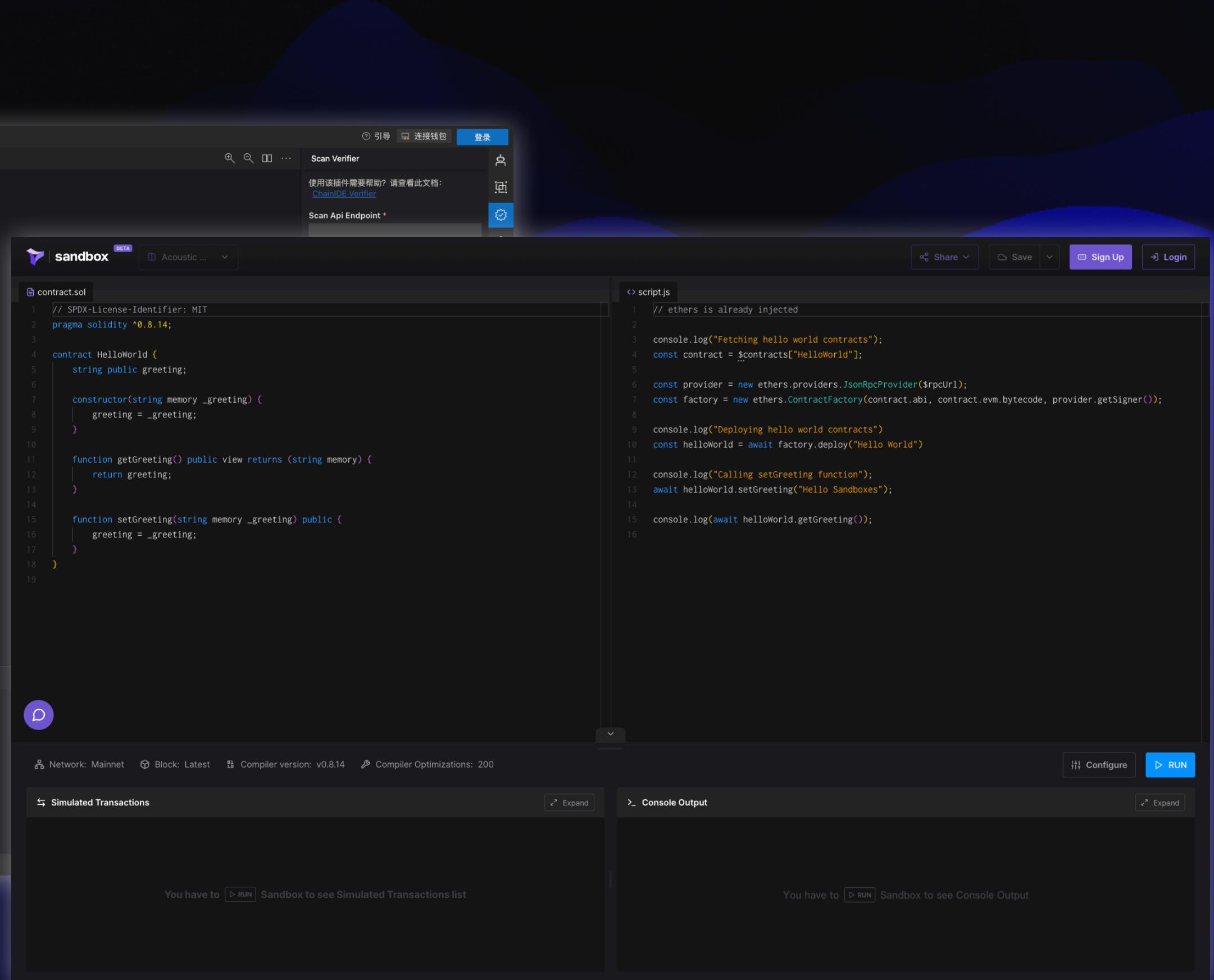
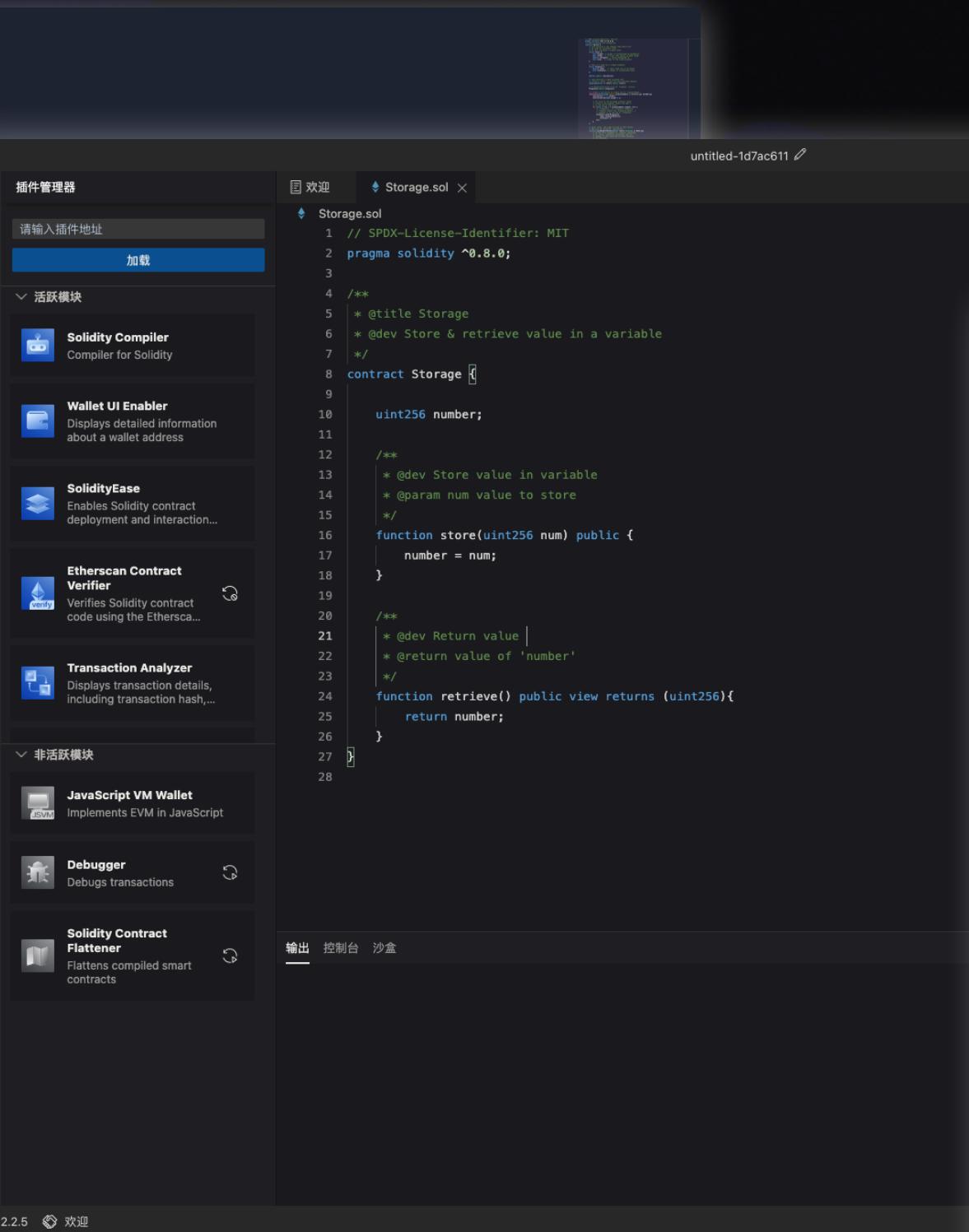
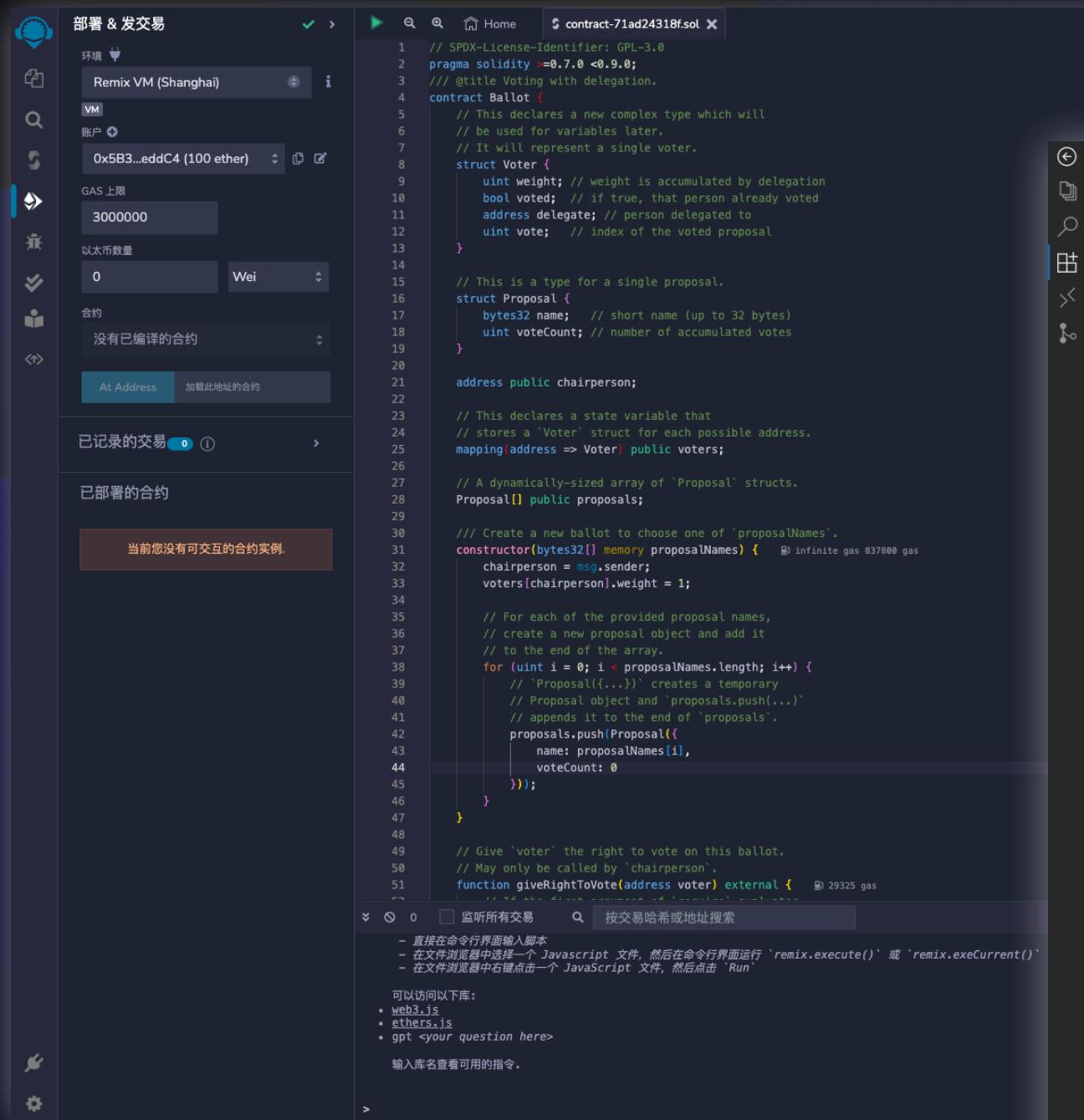
# 云端即时构建，无需繁琐的安装设置，加速开发迭代速度

调试、测试、部署一站式服务，不需要安装特殊工具。只需要增加新的编程语言和工具构建包即可轻松地支持如 Ethereum, Solana、Sui、Aleo 等可编程区块链。



# 打造专属的在线 IDE，用于开发、部署、调试和测试

基于 **Amphitheatre** 多语言编译构建及开箱即用的开发环境能力，您可以在 VS Code 在线版基础上增加自定义的插件和按钮，定制属于自己链的智能合约集成开发环境。



# 在线互动式编程教育与培训

在教学文档中嵌入代码编辑器，学生有机会在真实或模拟的编程环境中进行实践，以便学生能够立即应用所学知识

教学文档

Playground

http://docs.local/examples/hello-world

Save your file with a ".c" extension, for example, "hello.c". Open a terminal, navigate to the file's directory, and compile it with the following command: gcc hello.c -o hello This creates an executable file named "hello".

```
#include <stdio.h>

int main() {
    printf("Hello, world!");
    return 0;
}
```

Hello, world!

Congratulations! You've just written, compiled, and executed your first C program. This is a small but essential step towards mastering the C programming language. Happy coding!

A blue rounded rectangle containing a white icon of an amphitheater. A double-headed horizontal arrow connects this icon to the 'Playground' section of the teaching document.

Amphitheatre Cloud

# 未来发展计划

软件和 SaaS 服务的发展计划，包括新功能、版本升级等



如欲详谈，敬请联系

王宜国 / wangeguo@gmail.com / +86 13911128943