

Amphitheatre

开箱即用的云开发环境



我们为什么要建设这个项目？

开发人员在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？

解决方案：环境即服务

自动为开发人员提供测试环境以测试代码质量并帮助他们同时管理多个临时环境

传统模式（过去）

开发

开发环境配置复杂

资源

环境/资源隔离性差

调试

断点调试成本高

测试

环境部署效率低，
流量调度配置成本高，
流程长，环境不稳定

构建

不稳定、速度慢



资源服务化（现在）

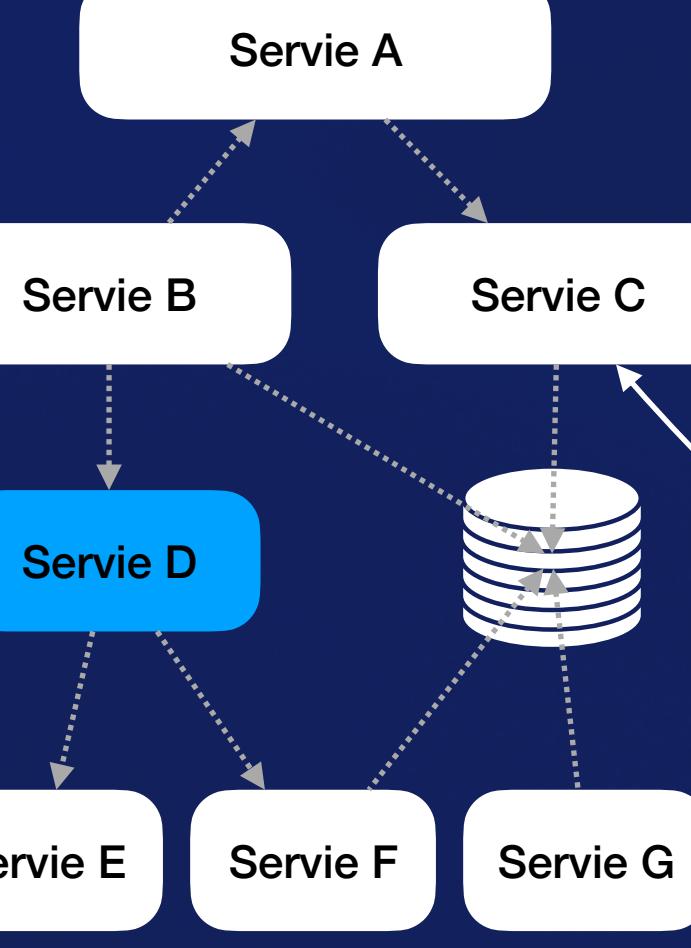
Peter 的工作空间

云上开发环境

代码仓库 Service D
弹性编译

云部署

Amphitheatre Cloud



John 的工作空间

云上开发环境

新服务 Service C
仓库推送、MR、
合并

云部署

集群调度，一键开
启云端开发环境

云端弹性资源可支撑大规
模微服务应用开发和部署

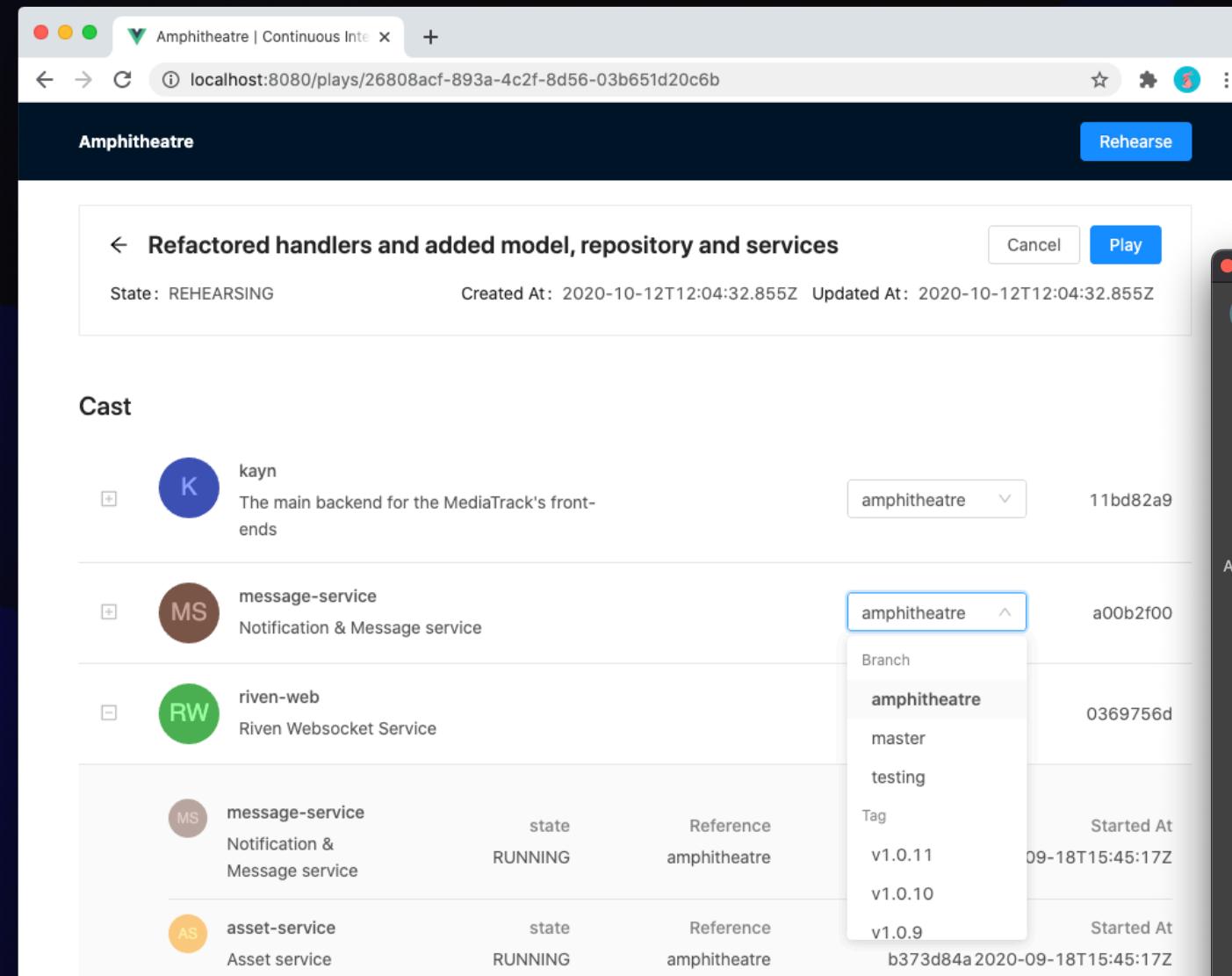
开发测试环境可以
动态按需准备

云端共享环境，
方便联调

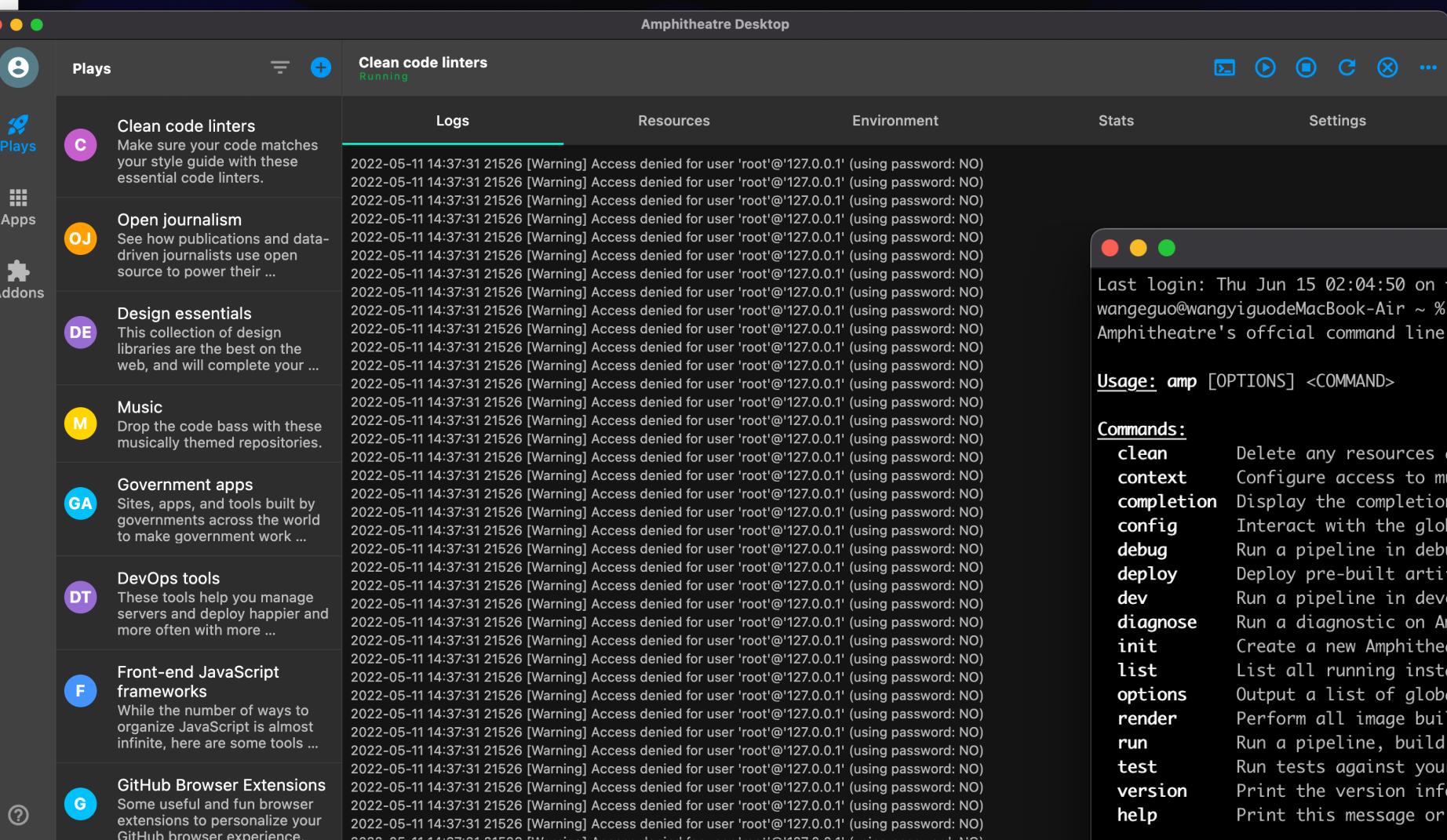
服务级云端一站
式构建与部署

客户端产品演示

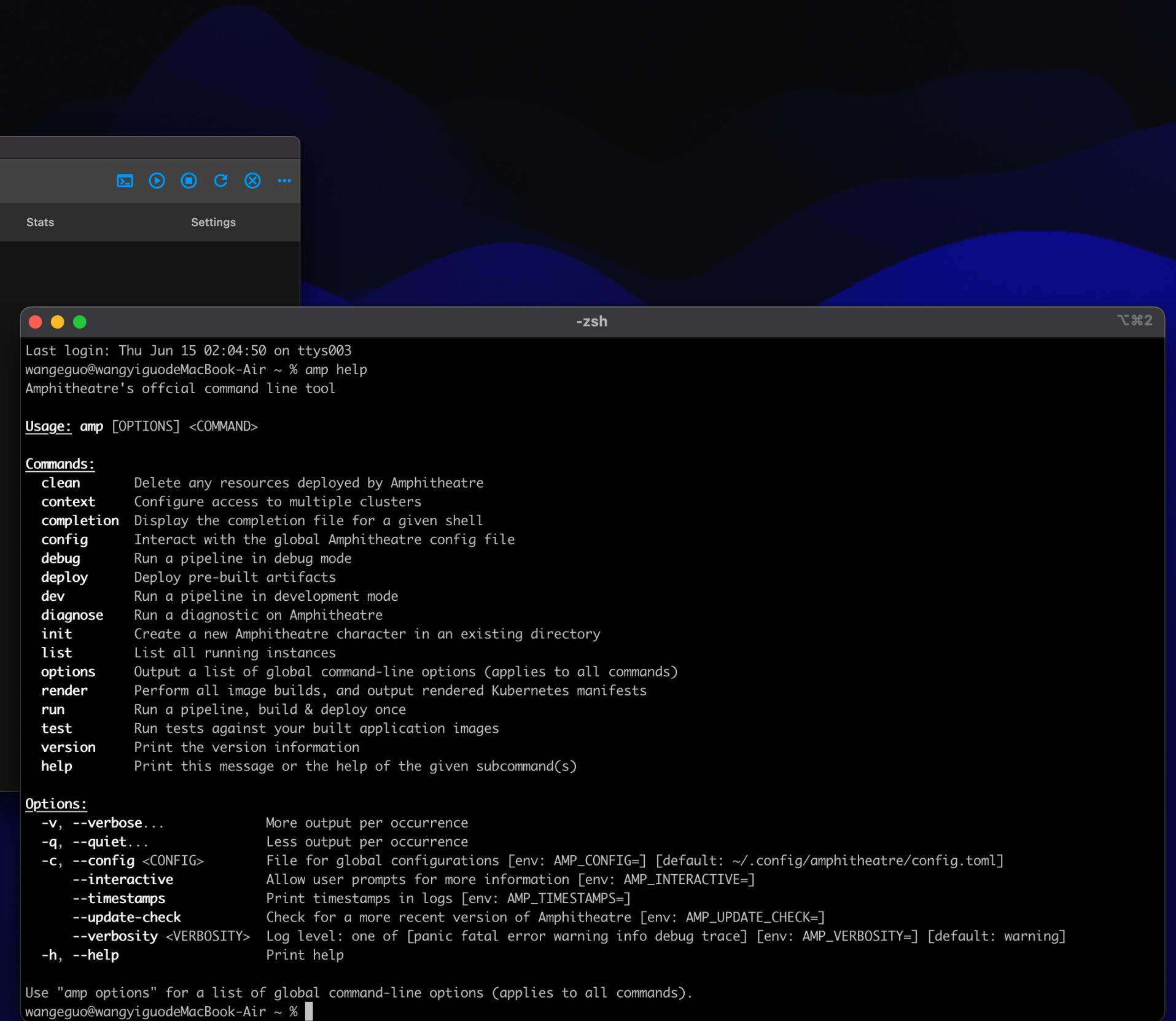
可运行于 Windows、macOS 和 Linux 等多种操作系统



网页版



桌面客户端



终端命令行

运行一个多组件的微服务应用

(Web)

1

申请

A screenshot of a web browser window titled "Amphitheatre | Continuous Inte". The URL is "localhost:8080/plays". A modal dialog box is open, titled "Rehearse a play". It contains fields for "Play's title" (set to "Refactored handlers and added model, repository and services") and "The URL of the Leading Player" (set to "https://git.mter.io/tech/sona"). There are "Cancel" and "OK" buttons at the bottom. In the background, a list of plays is visible, all labeled "REHEARSING".

正在测试中的环境

3

运行

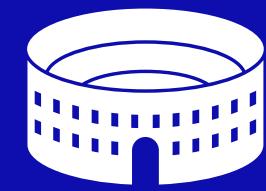
A screenshot of a web browser window titled "Amphitheatre". The URL is "localhost:8080/plays/26808acf-893a-4c2f-8d56-03b651d20c6b". The page shows a play titled "← Refactored handlers and added model, repository and services" in state "REHEARSING". Below it is a "Cast" section listing four components: "kayn" (Backend), "message-service" (Notification & Message service), "riven-web" (Riven Websocket Service), and two instances of "asset-service" (Asset service). On the right, a sidebar shows "amphitheatre" branches "master" and "testing", and tags "v1.0.11", "v1.0.10", and "v1.0.9".

搭档

2

切换版本

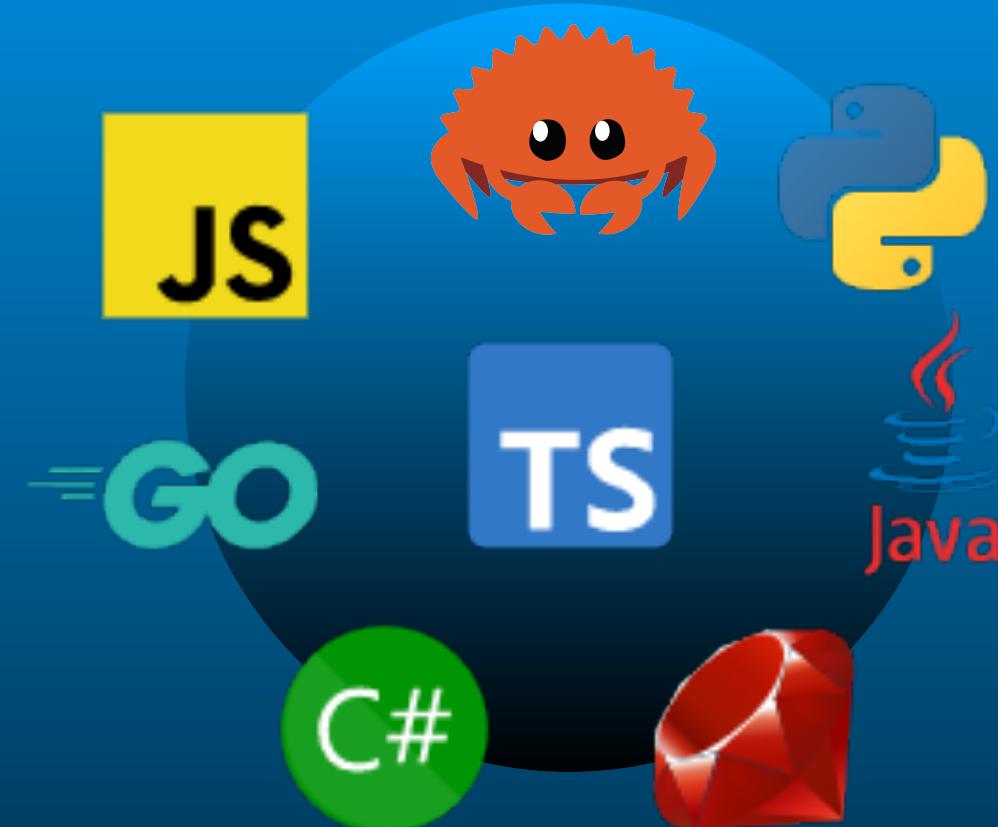
渐进式开发环境



Amphitheatre

在云端立即启动新的自动化开发环境。它提供按需且预先配置好的所有工具、库和依赖项，以确保您能够立即开始编写代码。

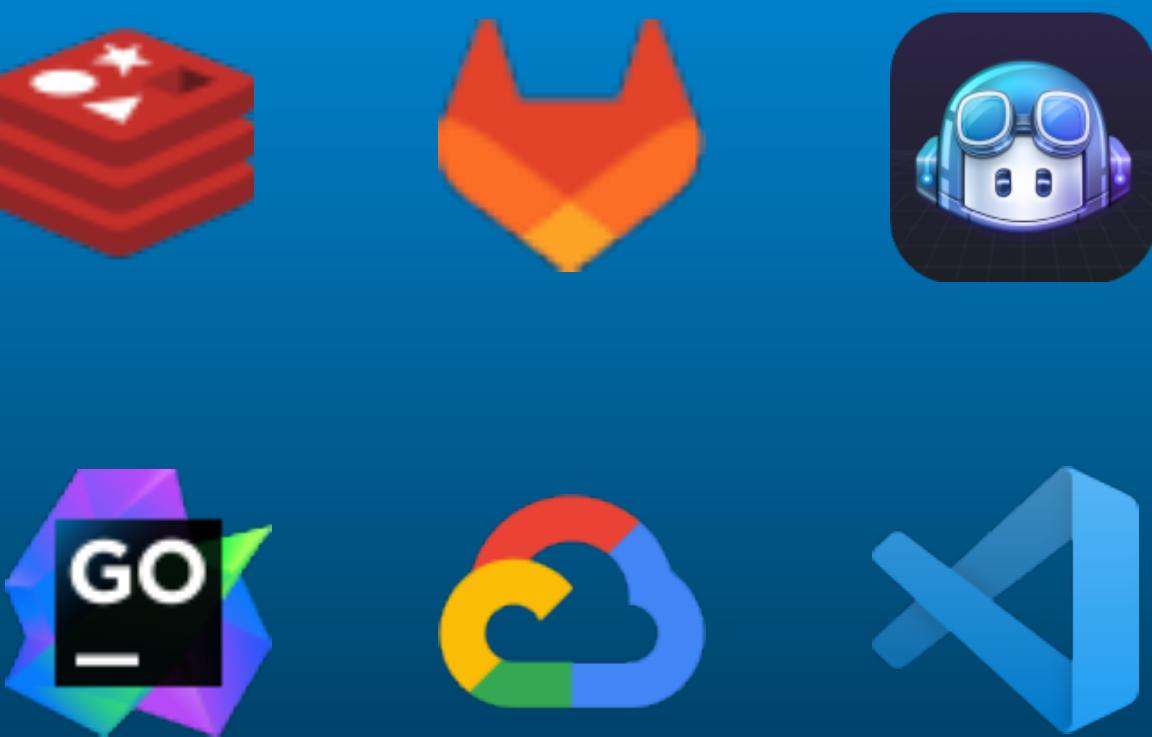
带上你的代码



支持多种流行语言、框架、SDKS 和运行时。

amp init

精简软件集成



与开发工具和任何开发资源配置使用，添加内部和外部依赖关系。

amp build

amp deploy

部署 & 运行



带有容器引擎的
Kubernetes 集群

Amphitheatre Cloud

amp cloud deploy

<http://localhost:3000>

无需配置本地环境， 支持多种编程语言和框架

适用于 29+ 种编程语言，以及
4000+ 技术栈。基于 Buildpacks
标准规范轻松搞定企业开发所
需的自定义需求。



本地开发实时部署到远程集群

可让您完全跳过镜像构建，它使用新代码更新正在运行的容器，只需几秒钟而不是几分钟。即使是编译语言或更改依赖项。



远程调试和可访问终端

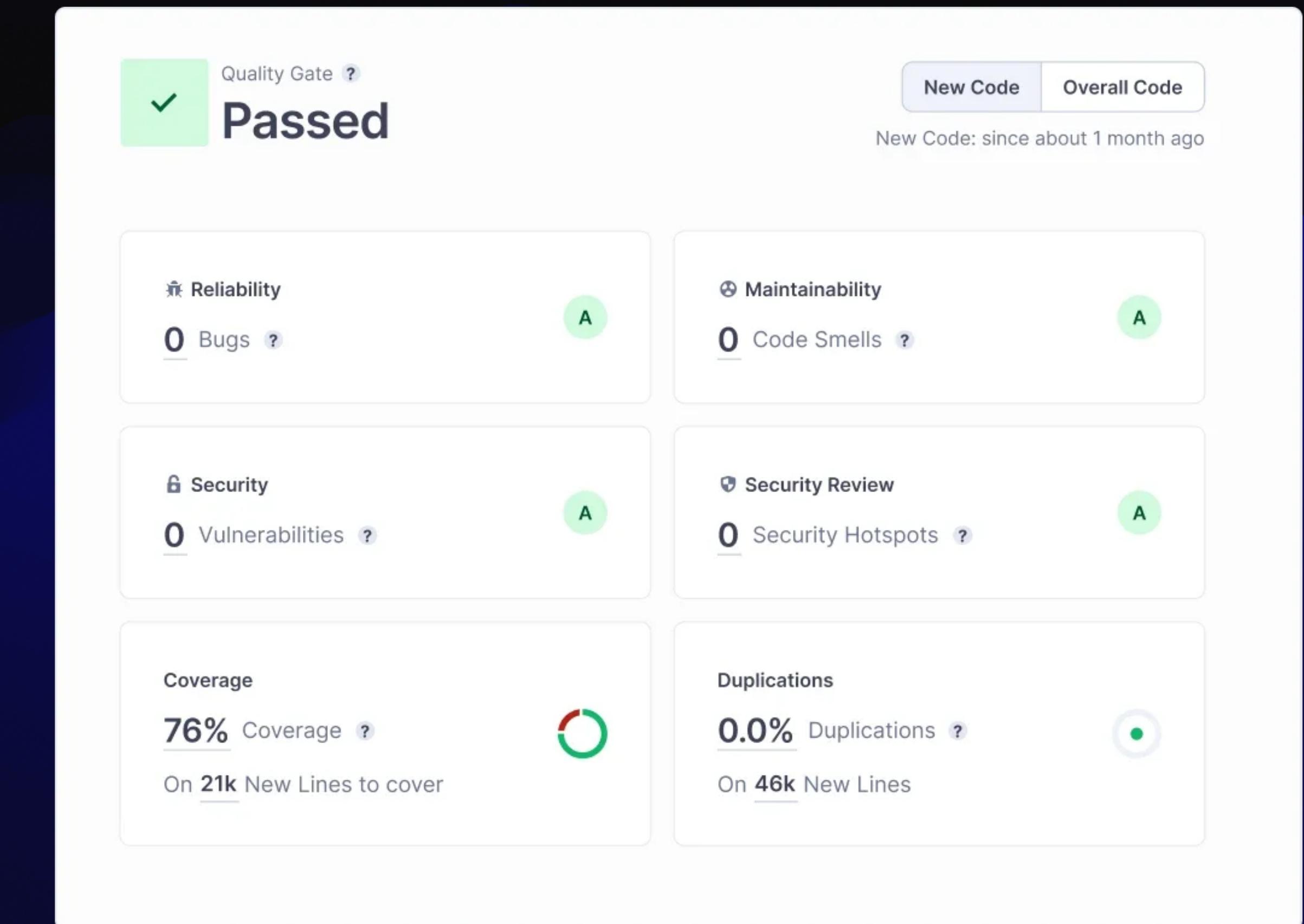
允许使用工具来检测、检查、调试和分析您的应用程序，而无需在本地配置编译设施。另可直接访问运行中的服务器，进一步查看构建上下文等调试信息。



基于 AI 为团队和企业提供干净的代码

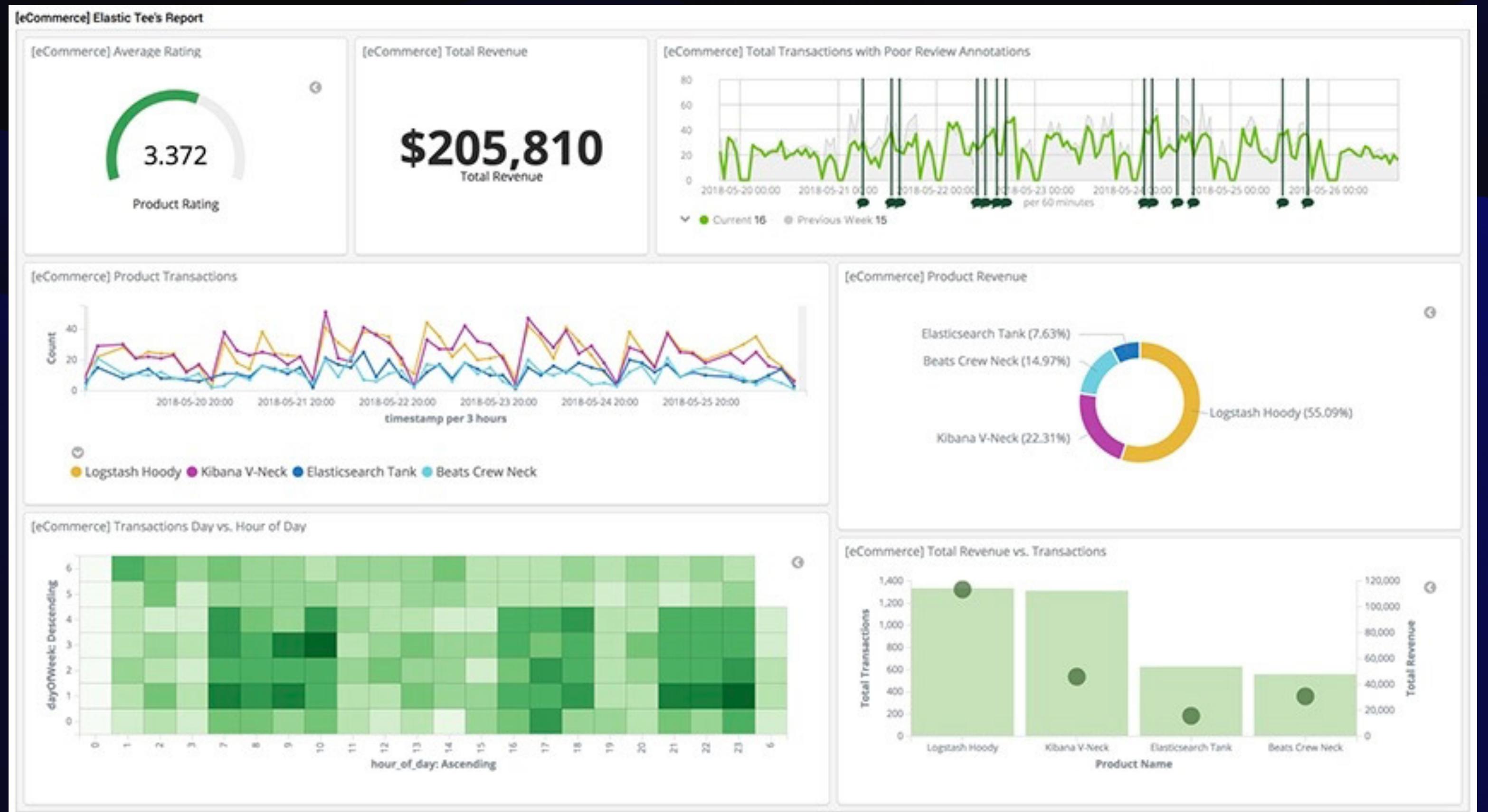
为开发团队提供深度集成到您的企业环境中的代码质量和安全解决方案；使您能够一致、可靠地部署干净的代码。

- ✓ 分析项目中所有语言的代码质量。修补错误、关闭漏洞并遵循单一事实来源的最佳实践。
- ✓ 超快速的分析可以在几分钟而不是几小时内为您提供可操作的干净代码指标。
- ✓ 在正确的地点和时间接收可操作的高精度反馈。受益于 5,000 多个编码规则以及行业领先的 JAVA、C#、PHP、PYTHON、TYPESCRIPT 和 JAVASCRIPT 污点分析。
- ✓ 安全报告、执行汇总和 PDF 报告为大型组织提供了评估其软件资产风险所需的监督。



只需几分钟即可开始分布式跟踪

无需更改代码，使用 OpenTelemetry 和 eBPF 即时监控任何应用程序



✓ 没有代码更改

✓ 与语言无关的自动检测

✓ 保留现有的可观测工具

✓ 与任何 APM 无缝集成

Powered by



Odigos

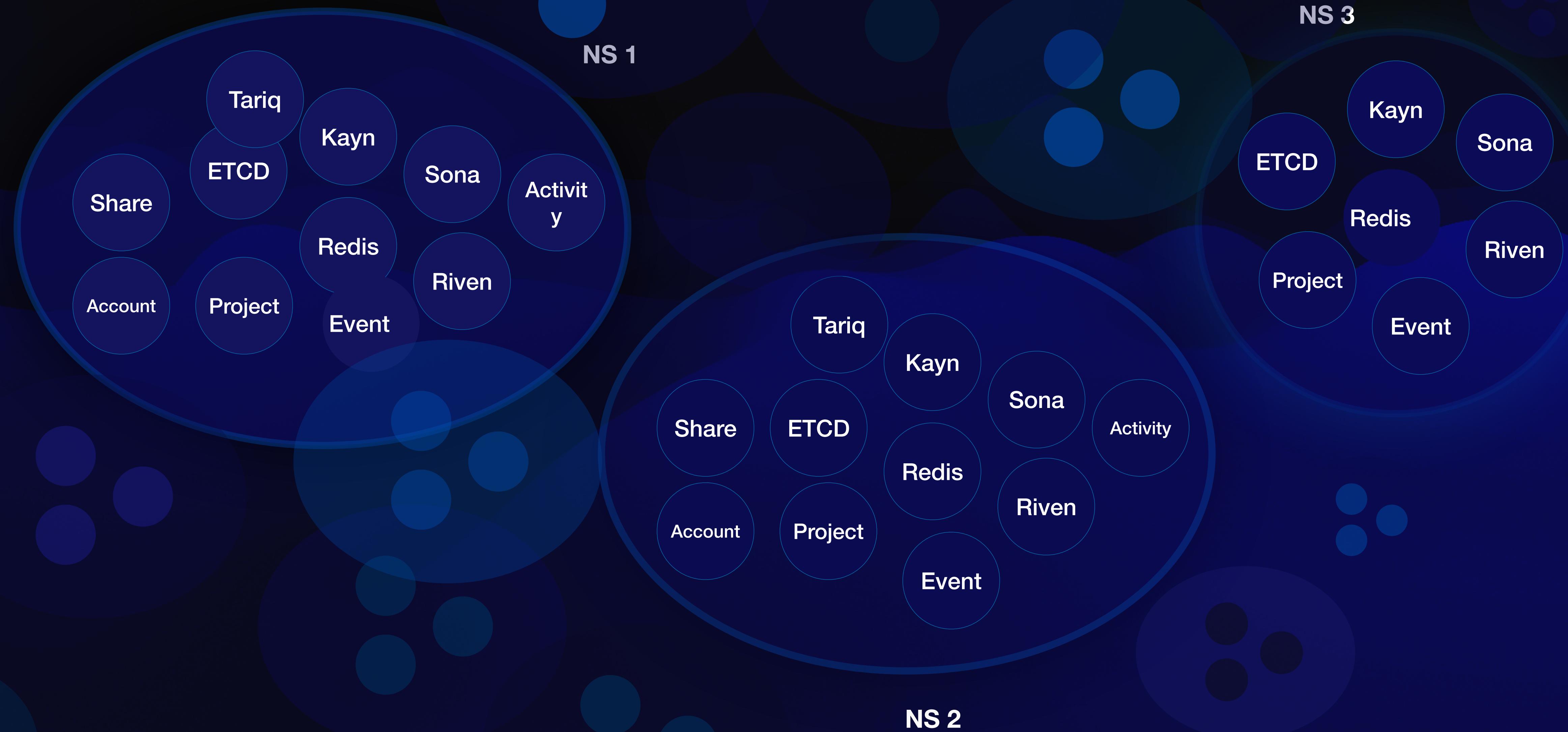
持续集成测试环境

开箱即用的多租户

在任何主机集群上强制实施安全多租户



轻松创建无限量的隔离集成测试环境



可插拔的生态应用市场，全方位提升效率



预览环境

每次提交时保持最新状态

针对每个拉取请求自动创建环境（包括服务、数据库和环境组）的全新副本

The screenshot shows a user interface for managing pull requests. On the left, a table lists four pull requests (PR #1331, PR #1330, PR #1330, PR #1327) across two branches (deployment-7341 and deployment-7344). Each row includes a status indicator (e.g., Deploying...) and a link to view the commit details. A modal window is overlaid on the interface, focusing on PR #1330. The modal displays a circular profile picture of a woman, a green 'Open' button, and the text 'marcleadb wants to merge 1 commit'.

ID	BRANCH
PR #1331	feat: add GroupListPicker component deployment-7341
PR #1330	fix the end-to-end tests deployment-7344
PR #1330	Add default errorHandler() to v... deployment-7323
PR #1327	Update dependency aws-sdk deployment-7320

为您的软件开发人员提供支持

等待访问共享环境会降低生产力（更不用说士气了！）

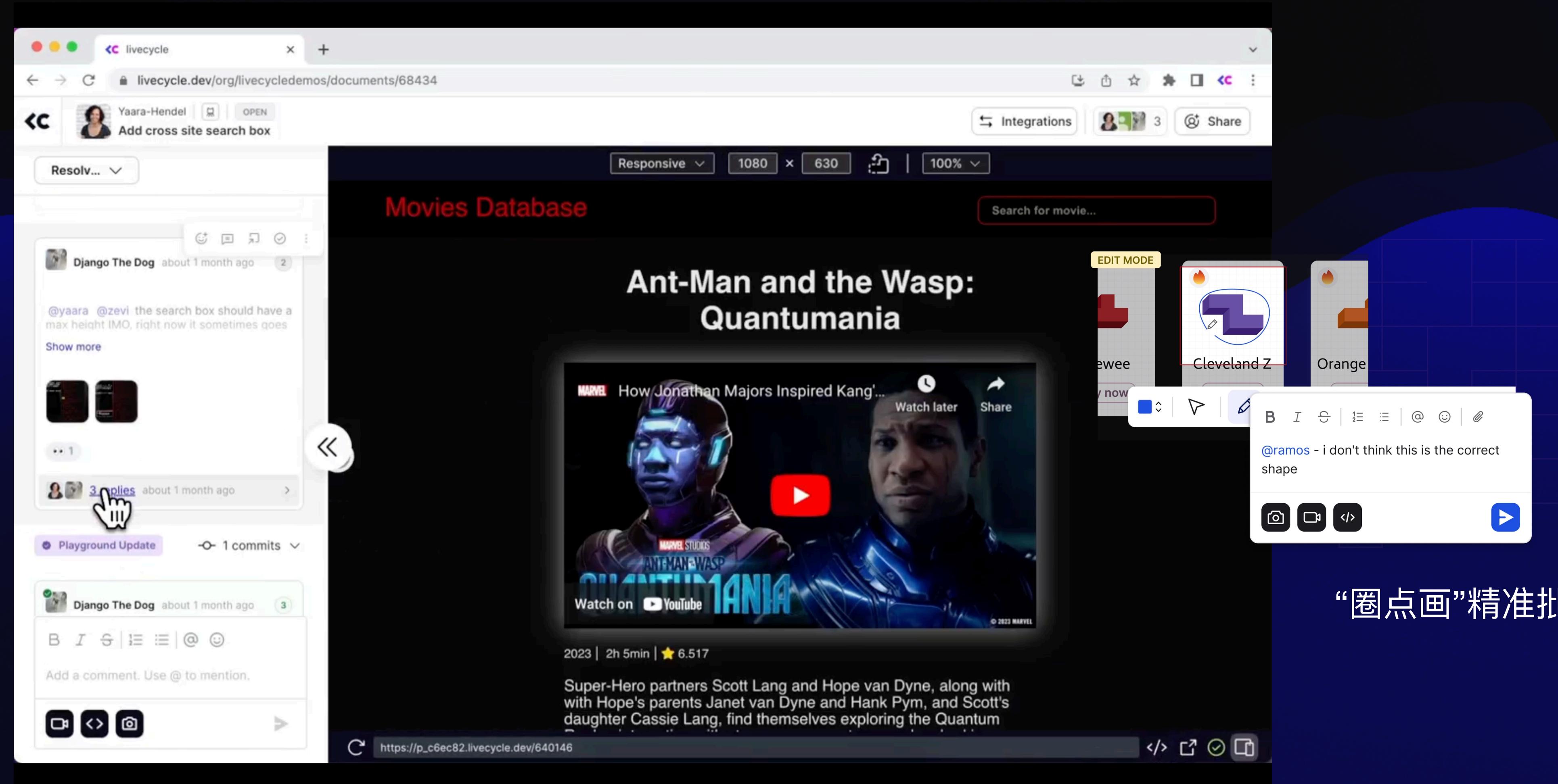
- ✓ 当您的开发人员需要时提供开发环境！只需几分钟即可获得新环境，而不是几天。
- ✓ 在错误合并到生产环境之前捕获它们
- ✓ 针对服务环境运行手动或自动测试
- ✓ 向开发人员提供快速反馈并查看同步的更改
- ✓ 隔离测试以避免污染 QA 环境和数据积累

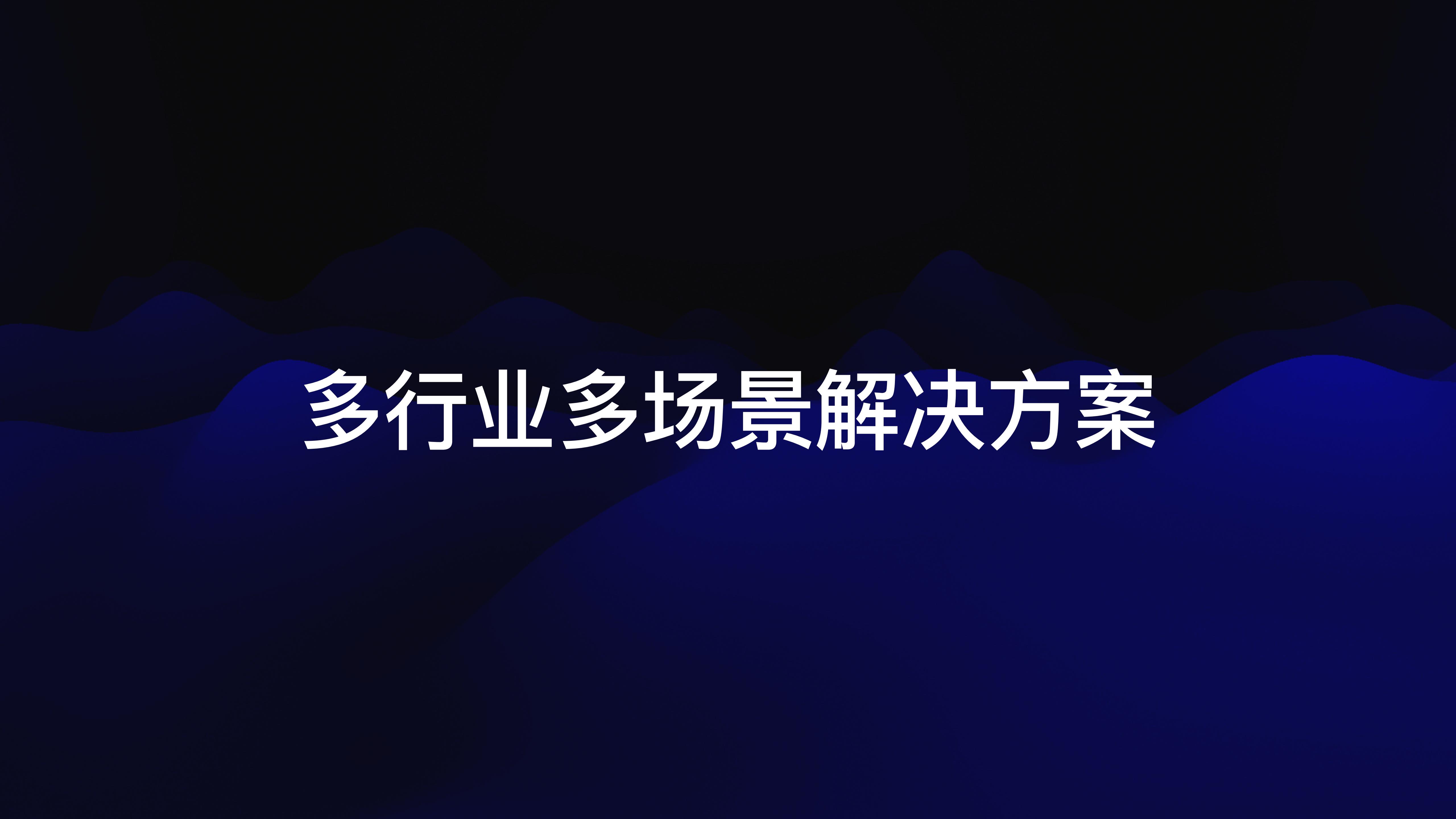
在任何预览环境中的上下文反馈

开发团队收集有关任何预览环境的数据丰富的上下文反馈，并消除工作流程摩擦

所有修改意见按时间码或发表顺序，自动有序汇集。

并与问题画面一一对应，无需在杂乱的聊天记录中进行人工整理。



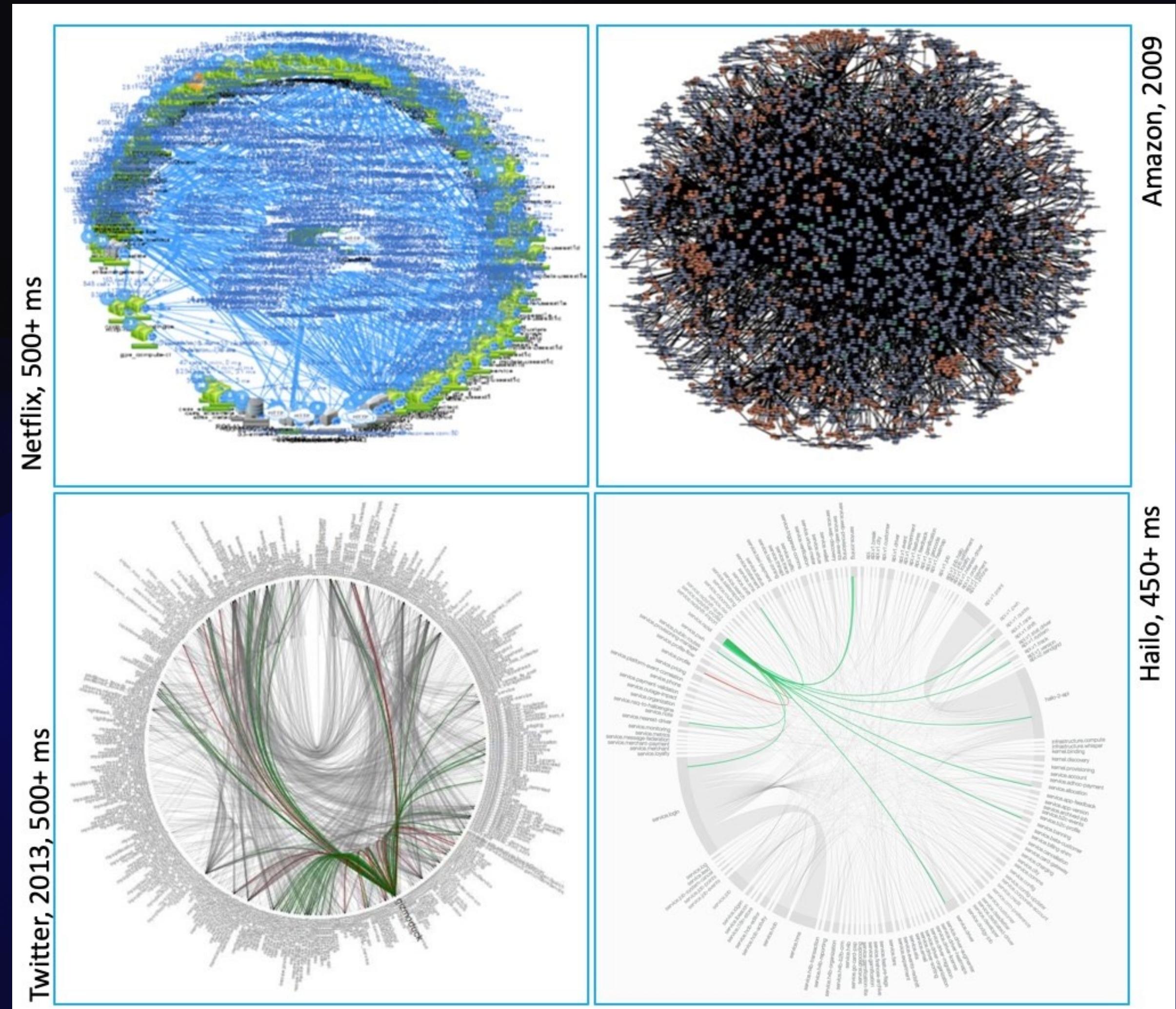


多行业多场景解决方案

微服务架构应用开发

微服务架构的挑战与解决方案

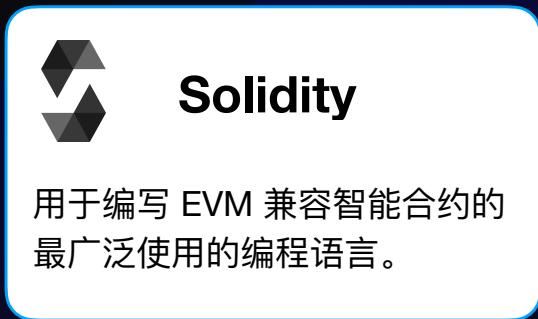
- 服务拆分和边界定义
- 服务间通信和数据一致性
- 服务发现和负载均衡
- 分布式事务和数据管理
- 避免跨多个服务
- 部署和监控



区块链 & Web3 去中心化应用开发

支持区块链编程语言和工具，以快速构建 Web3 项目

智能合约编程语言



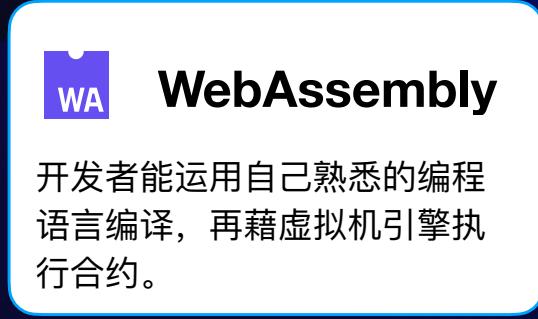
用于编写 EVM 兼容智能合约的最广泛使用的编程语言。

用于编写 EVM 兼容智能合约的最广泛使用的编程语言。



StarkNet 的 ZK 兼容智能合约
编码语言。

StarkNet 的 ZK 兼容智能合约 编码语言。



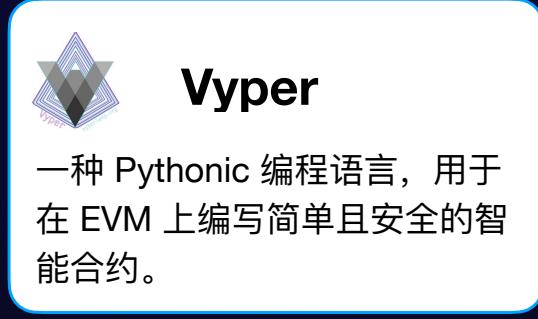
开发者能运用自己熟悉的编程语言编译，再藉虚拟机引擎执行合约。

开发者能运用自己熟悉的编程语言编译，再藉虚拟机引擎执行合约。



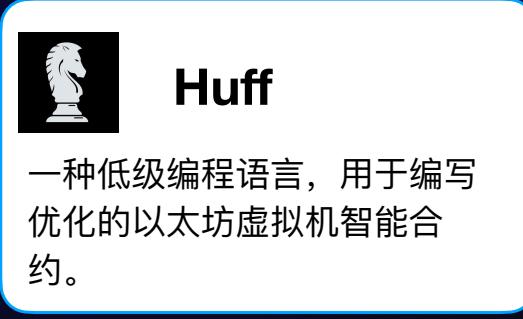
基于原型和头等函数的多范式高级解释型编程语言，可用于编写 Web3 前端程序。

基于原型和头等函数的多范式高级解释型编程语言，可用于编写 Web3 前端程序。



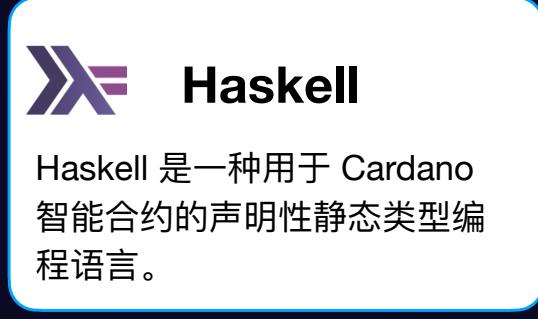
一种 Pythonic 编程语言，用于在 EVM 上编写简单且安全的智能合约。

一种 Pythonic 编程语言，用于在 EVM 上编写简单且安全的智能合约。



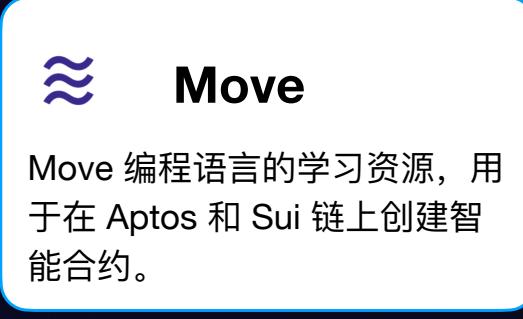
一种低级编程语言，用于编写优化的以太坊虚拟机智能合约。

一种低级编程语言，用于编写
优化的以太坊虚拟机智能合
约。



Haskell 是一种用于 Cardano 智能合约的声明性静态类型编程语言。

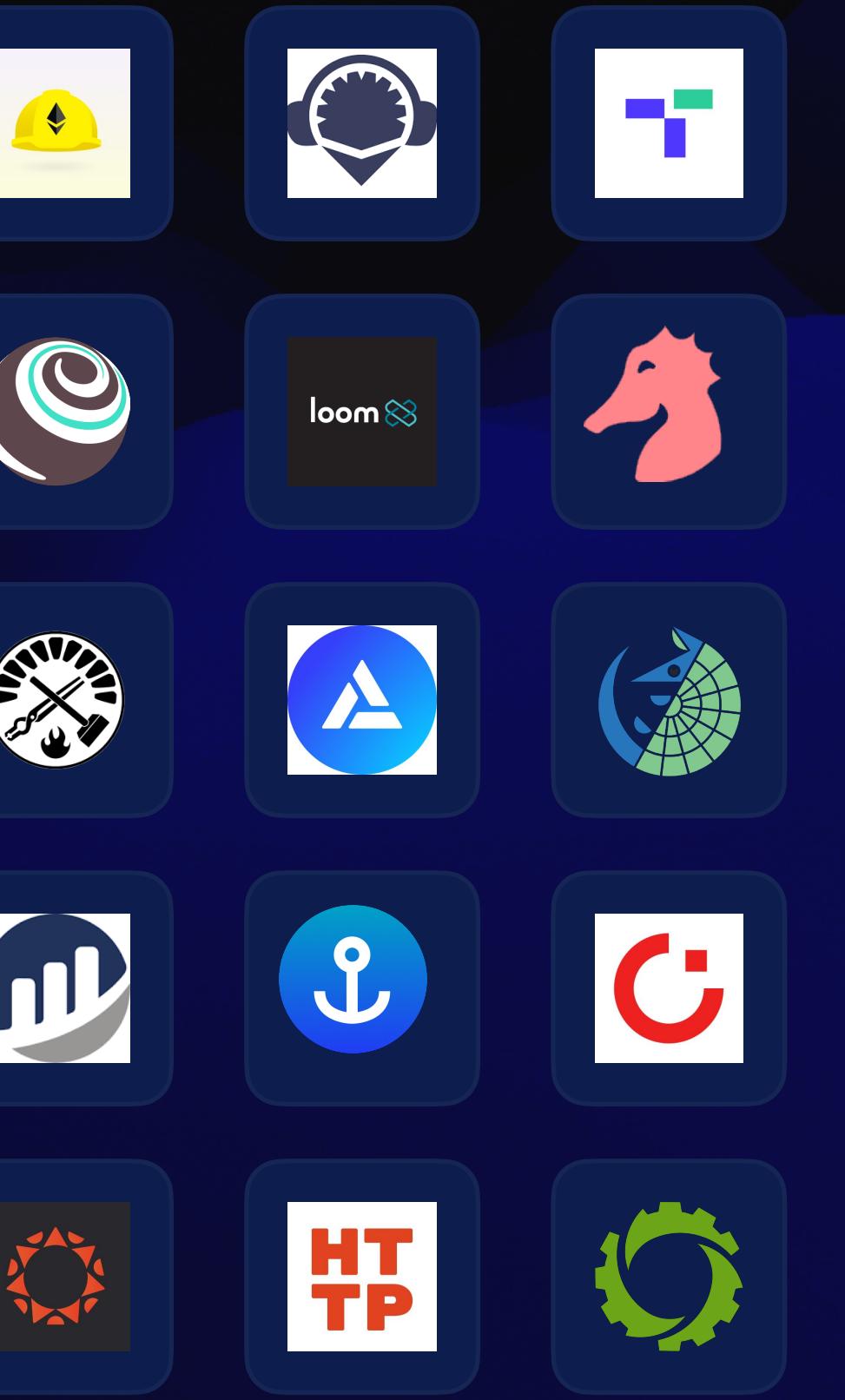
Haskell 是一种用于 Cardano 智能合约的声明性静态类型编程语言。



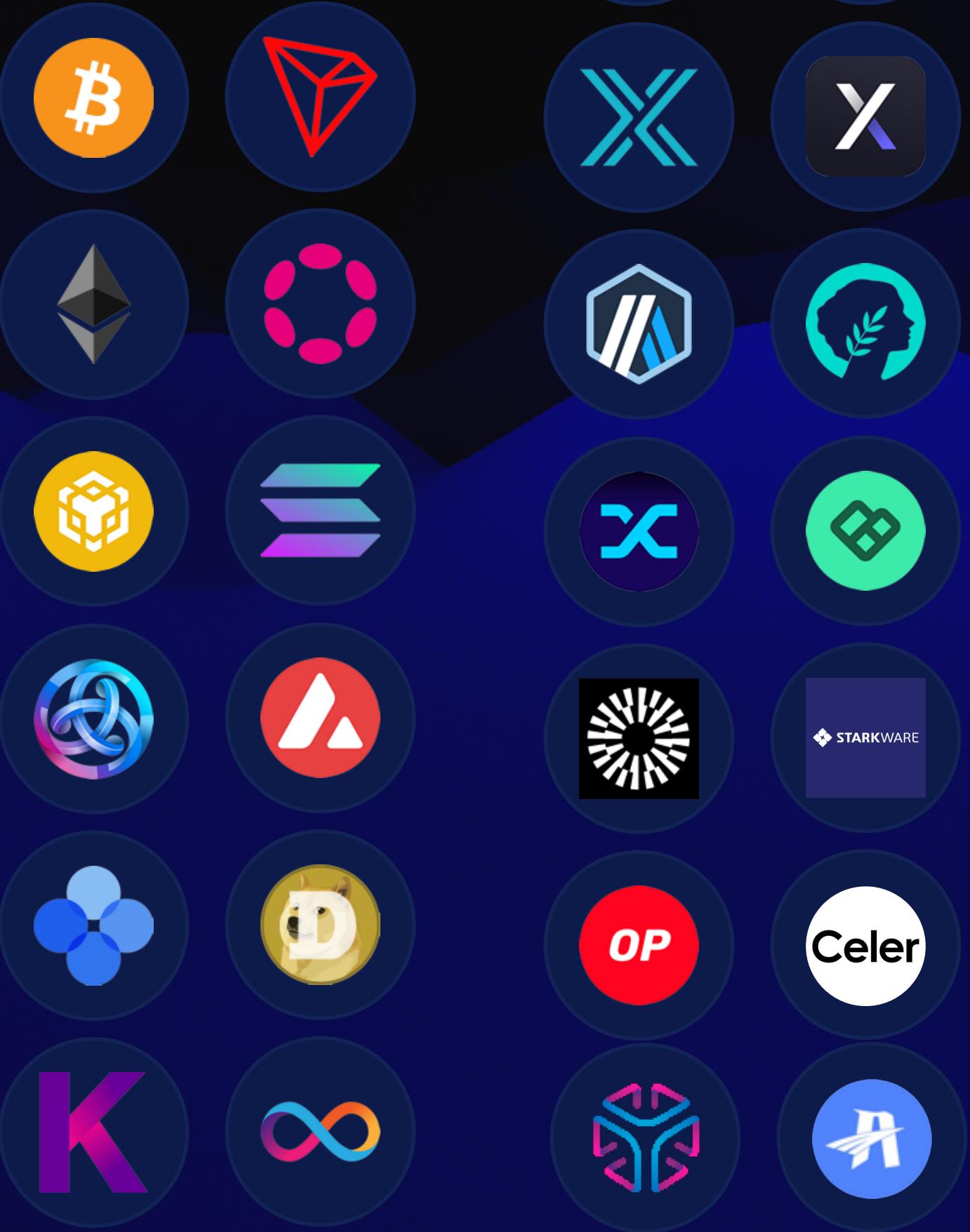
Move 编程语言的学习资源，用于在 Aptos 和 Sui 链上创建智能合约。

Move 编程语言的学习资源，用于在 Aptos 和 Sui 链上创建智能合约。

框架与开发工具



Layer 1



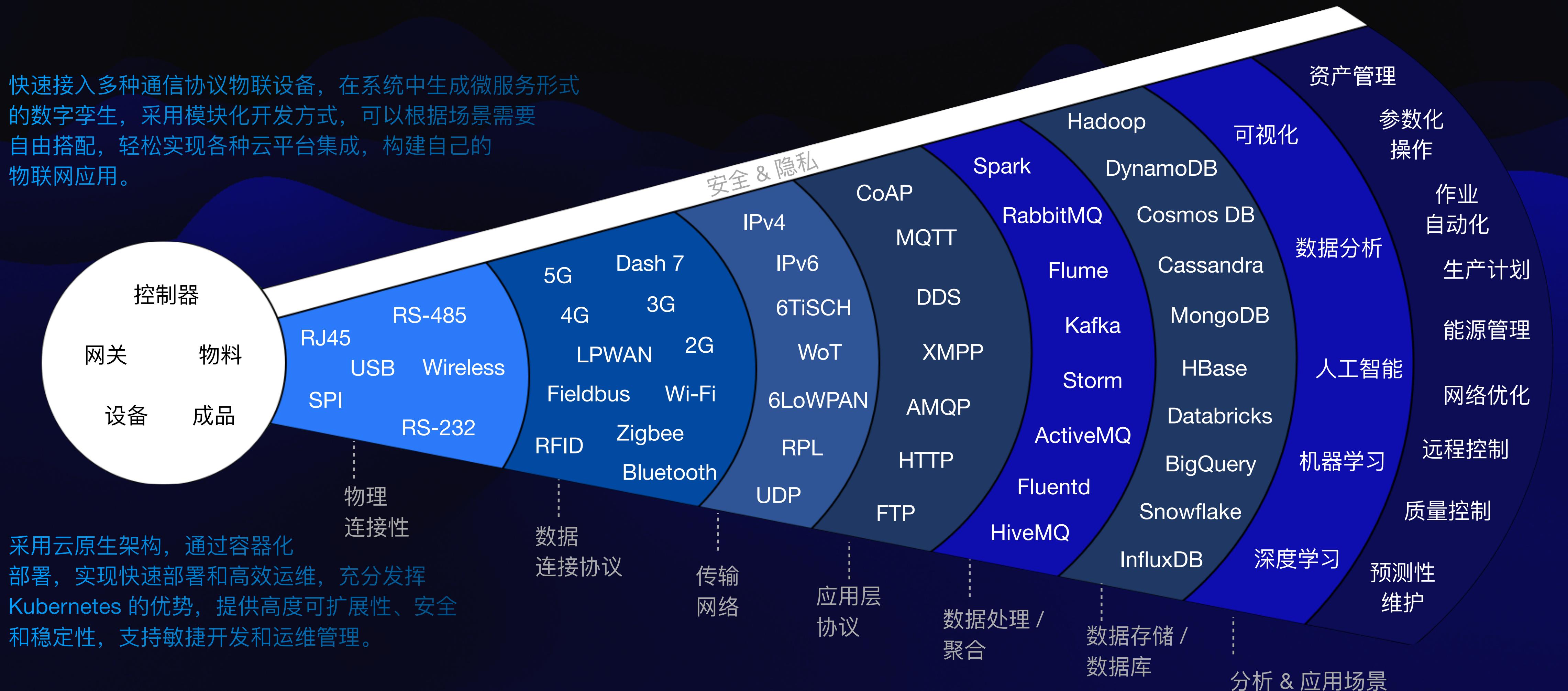
合约代码开发

时响应式构建

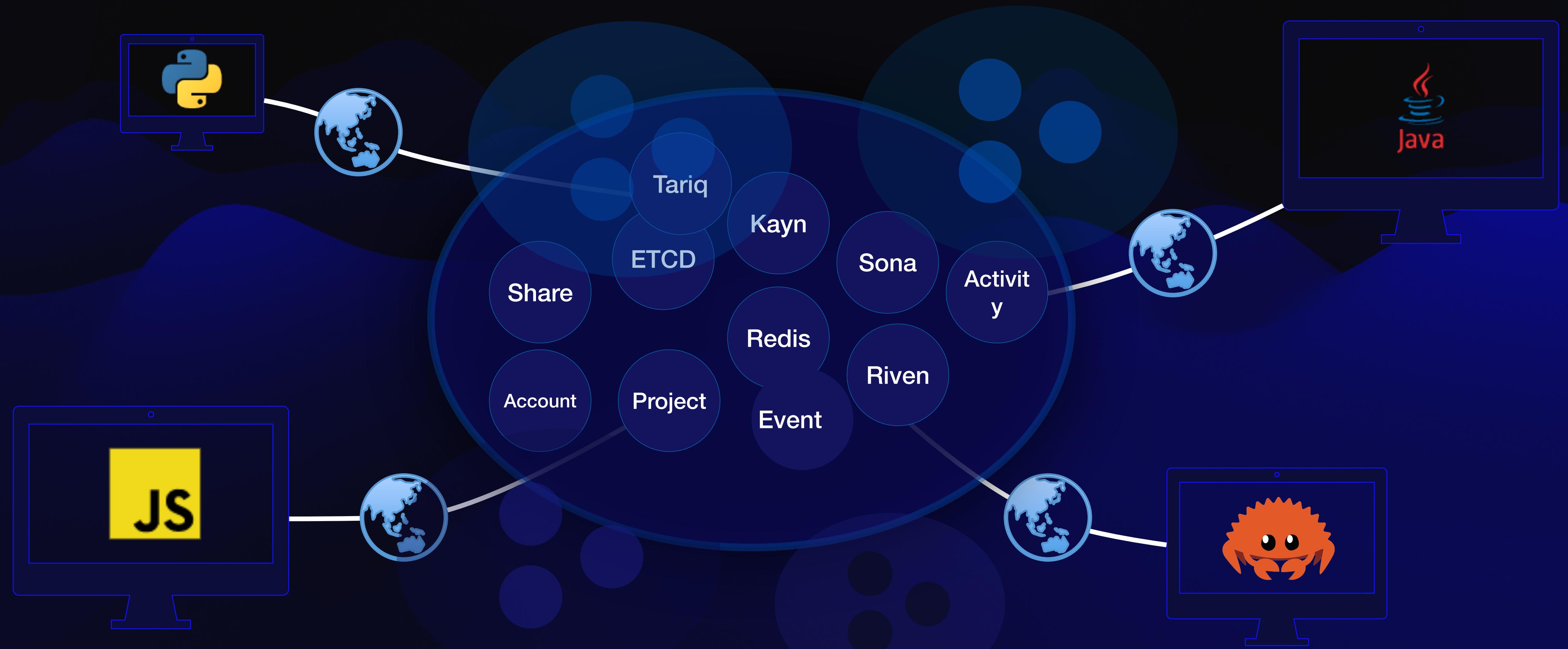
测试网络部署

物联网 & 智能制造 & 嵌入式开发

赋能全场景物联网开发，从通信协议物联设备到微服务形式的数字孪生应用



跨组织/地域/职能协作



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

The diagram illustrates the integration of an interactive code editor (Playground) into a teaching document (Teaching Document). A blue line connects the '教学文档' (Teaching Document) label to a screenshot of a web browser window. The browser window shows a URL: <http://docs.local/examples/hello-world>. The page content includes instructions: "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".". Below the instructions is a code editor area containing a C program:

```
#include <stdio.h>

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

The output of the code execution is shown as "Hello, world!" in the terminal window. A blue dashed box highlights the code editor area. Another blue line connects the 'Playground' label to the bottom of the code editor area. To the right, a large blue icon of an amphitheater labeled 'Amphitheatre Cloud' is connected by a double-headed arrow to the code editor area, indicating a bidirectional relationship or integration between the local playground and the cloud-based platform.

复合条件集成测试

示例：Backports 向后移植是从下一个版本（称为“测试”）中获取的软件包，经过调整和重新编译以在稳定版上使用，利用 Amphitheatre 可进行多种复合测试条件进行构建和测试，以输出报告



软件包

交叉组合编译集群

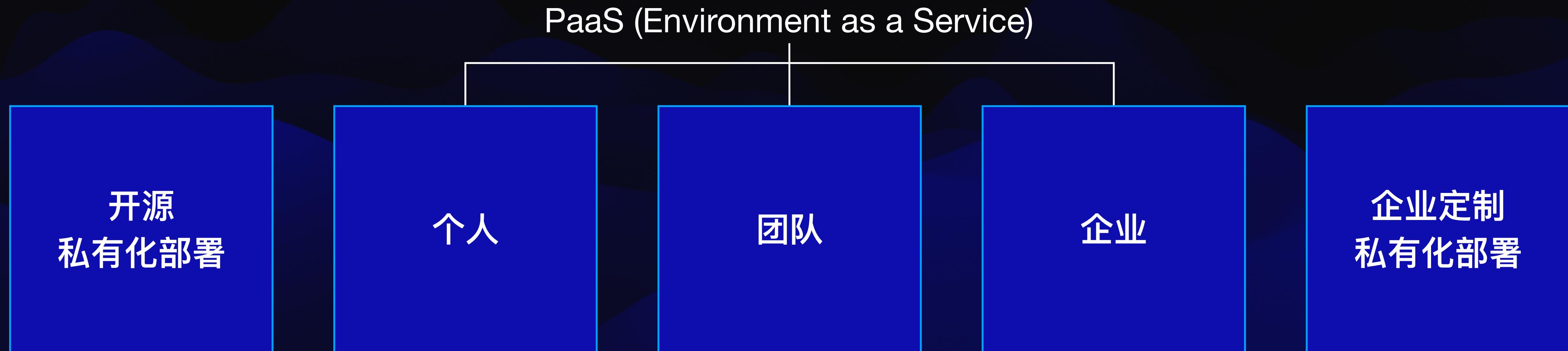
测试报告

竞争格局



商业模式

开源、EaaS 托管、定制部署，满足不同客户需求，创造多渠道收益



MicroK8s



Alibaba Cloud



KUBESPHERE®

支持 CNCF 认证的 Kubernetes 发行版和更多托管服务平台

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