


Web 开发技术 @ 2025

章许帆

xufan.zhang@outlook.com

Web 持续集成、部署与性能优化



代码版本管理

Git 提交最佳实践

- 保持小且专注的提交，每个提交应该只解决一个特定问题
- 编写有意义的提交信息

```
<type>(<scope>): <subject>
```

type : feat: 新功能; fix/to: 修复bug, 可以是QA发现的BUG, 也可以是研发自己发现的BUG; docs: 文档; style: 格式; refactor: 重构; perf: 优化相关, 比如提升性能、体验; test: 增加测试; chore: 构建过程或辅助工具的变动; revert: 回滚到上一个版本; merge: 代码合并; sync: 同步主线或分支的Bug。

scope : 说明 commit 影响的范围。

subject : subject是commit目的的简短描述, 不超过50个字符。

- 频繁提交, 完成一个小功能或修复就提交一次

代码版本管理

Git 提交信息

Commits on Jun 15, 2025	
fix(deps): update dependency @grpc/proto-loader to v0.7.15 (#4337)	Verified ee5e222
renovate[bot] authored 2 weeks ago · 2 / 5	
fix(deps): update dependency ali-oss to v6.23.0 (#4338)	Verified eee7337
renovate[bot] authored 2 weeks ago · 2 / 5	
feat: create board manager on server ready (#4346)	Verified 65a86bc
czy88840616 authored 2 weeks ago · 2 / 5	
Commits on Jun 3, 2025	
docs: update mqtt.md (#4343)	Verified 7f598ca
miraizhao authored last month · 1 / 1	
Commits on May 26, 2025	
chore: remove aem plugin (#4342)	Verified 5591c6a
czy88840616 authored on May 26 · 1 / 1	
Commits on Apr 28, 2025	
v3.20.5	7ce5728
czy88840616 committed on Apr 28 · 3 / 6	
chore: upgrade bullmq version and update cookie extra options (#4330)	Verified ec27530
czy88840616 authored on Apr 28 · 2 / 5	
Commits on Apr 26, 2025	
feat: support new cookies options (#4329)	Verified 2392502
czy88840616 authored on Apr 26 · 2 / 5	
fix(deps): update dependency @grpc/grpc-js to v1.13.3 (#4324)	Verified dd77c8b
renovate[bot] authored on Apr 26 · 2 / 5	
Commits on Apr 24, 2025	
feat(rabbitmq): add msg into rabbitmq ctx (#4326)	Verified b47e8af
larryzhao and larry authored on Apr 24 · 2 / 5	

测试自动化

在代码提交的瞬间，代码托管平台就可以迅速启动一个自动化流程对代码进行体检

- 开发人员基于 Vitest、Jest 编写了测试脚本

- 前端：

```
"scripts": {  
  "test": "vitest",  
},
```

- 后端

```
"scripts": {  
  "test": "cross-env NODE_ENV=unittest jest",  
},
```

- 流水线实现测试脚本的自动化运行

Github Actions

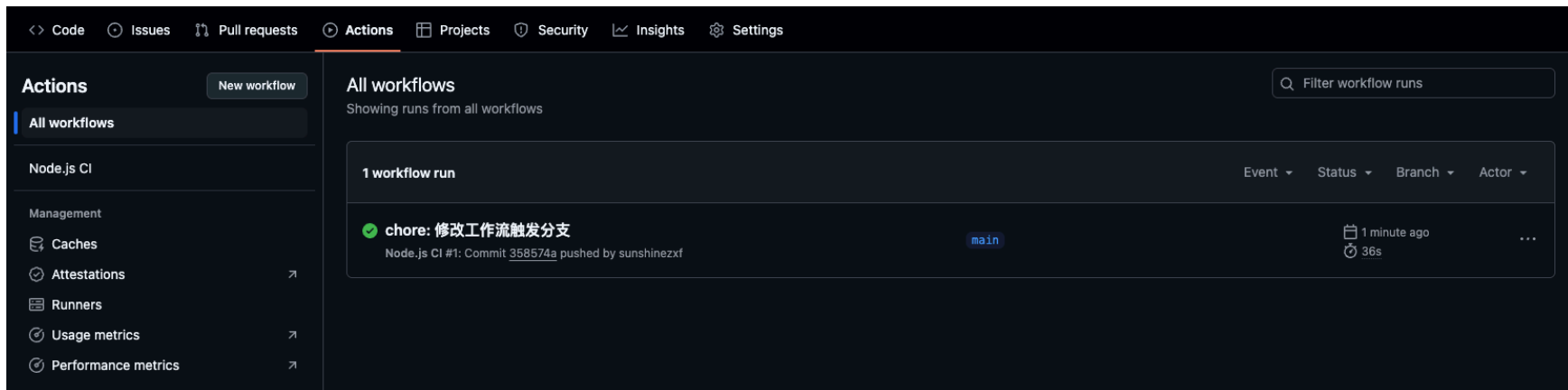
基于 Github Actions，可以快速创建持续集成 workflow

GitHub 提供了一系列的 workflow 模板，可以帮助快速创建工作流。

```
name: Node.js CI
on: # 触发 workflow 的事件
  push:
    branches: [ "main" ]
jobs: # 定义 workflow 中的任务（一个 workflow 可包含多个任务）
  build:
    runs-on: ubuntu-latest # 指定运行环境
    strategy:
      matrix: # 定义矩阵策略，并行运行多环境测试
        node-version: [20.x, 22.x]
    steps: # 具体执行步骤
      - uses: actions/checkout@v4 # 拉取代码
      - name: Use Node.js ${{ matrix.node-version }}
        uses: actions/setup-node@v4
        with:
          node-version: ${{ matrix.node-version }}
          cache: 'npm'
      - run: npm ci
      - run: npm run test
      - run: npm run build --if-present
```

Github Actions

代码仓库添加对应配置(`.github/workflows`), 代码提交到对应的分支后, 将自动触发工作流的执行。

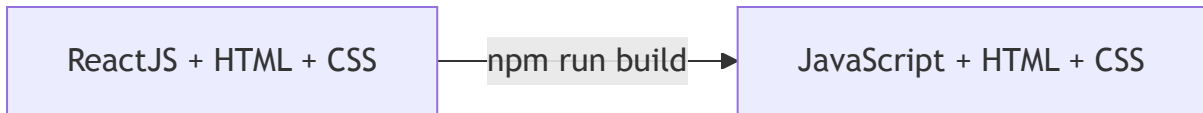


工作流自动帮助完成测试用例的执行、代码的构建。

持续构建

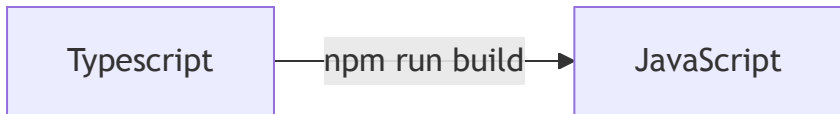
当自动化测试通过后，流水线开始对前后端代码库进行各自的构建

■ 前端



```
"scripts": {  
  "build": "vite build",  
}
```

■ 后端



```
"scripts": {  
  "build": "mwtsc --cleanOutDir"  
}
```


构建产物

■ 前端产物

```
├─ dist
  │─ index.html
  └─ assets
    │─ index.css
    └─ index.js
```

■ 后端产物

```
├─ dist
  │─ config
  │   │─ ...
  │   └─ config.default.js
  │─ controller
  │─ service
  └─ ...
```

后端产物仍然需要依赖 `node_modules` 中的内容才能正常运行。可以基于 `@vercel/ncc` 实现单文件构建。

单文件构建

<https://midwayjs.org/docs/deployment#构建流程>

■ 依赖安装

```
## 用于生成入口
npm i @midwayjs/bundle-helper --save-dev
## 装到全局
npm i @vercel/ncc -g
```

■ 入口文件修改

```
//bootstrap.js
const { Bootstrap } = require("@midwayjs/bootstrap")
Bootstrap.configure({
  imports: require('./dist/index'),
  moduleDetector: false
}).run();
```

■ 脚本命令配置

```
"scripts": {
  "bundle": "bundle && npm run build && ncc build bootstrap.js -o build",
  "bundle_start": "NODE_ENV=production node ./build/index.js"
},
```

单文件构建

<https://midwayjs.org/docs/deployment#构建流程>

■ 配置模式的调整

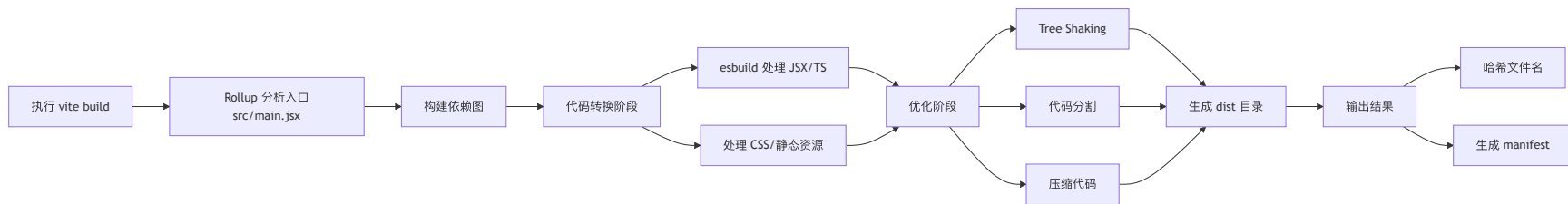
```
1 import DefaultConfig from './config/config.default';
2 import UnitTestConfig from './config/config.unittest';
3 @Configuration({
4   ...
5   importConfigs: [{
6     default: DefaultConfig,
7     unittest: UnitTestConfig
8   }],
9 })
```

■ 数据源相关 entity 设置

```
//config.default.ts
import * as entity from '../entity'
export default {
  ...
  typeorm: {
    dataSource: {
      default: {
        ...
      }
    }
  }
}
```

构建过程

■ Vite 生产构建过程



Rollup 主导整个打包过程，esbuild 仅用于转换（非打包）。优化阶段是生产构建的核心，包括 Tree Shaking 和代码分割。最终输出会进行哈希处理以实现长效缓存。

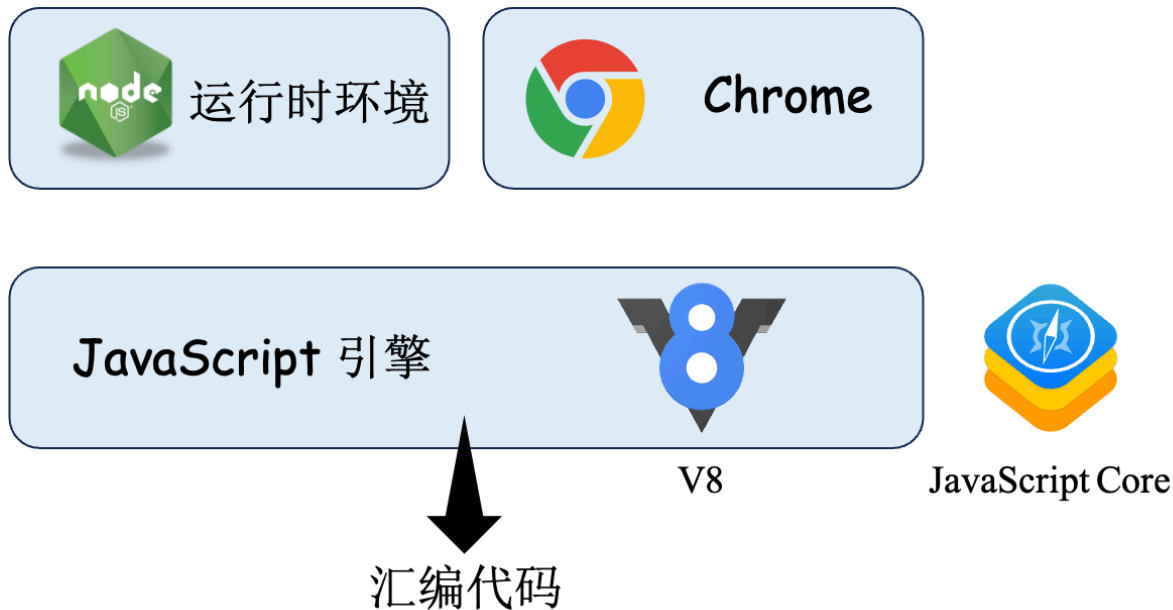
■ MidwayJS 生产构建过程



与前端构建工具不同，后端构建不需要处理资源打包，而是强调类型安全和依赖管理。其构建过程专注于代码编译、依赖处理和部署优化。

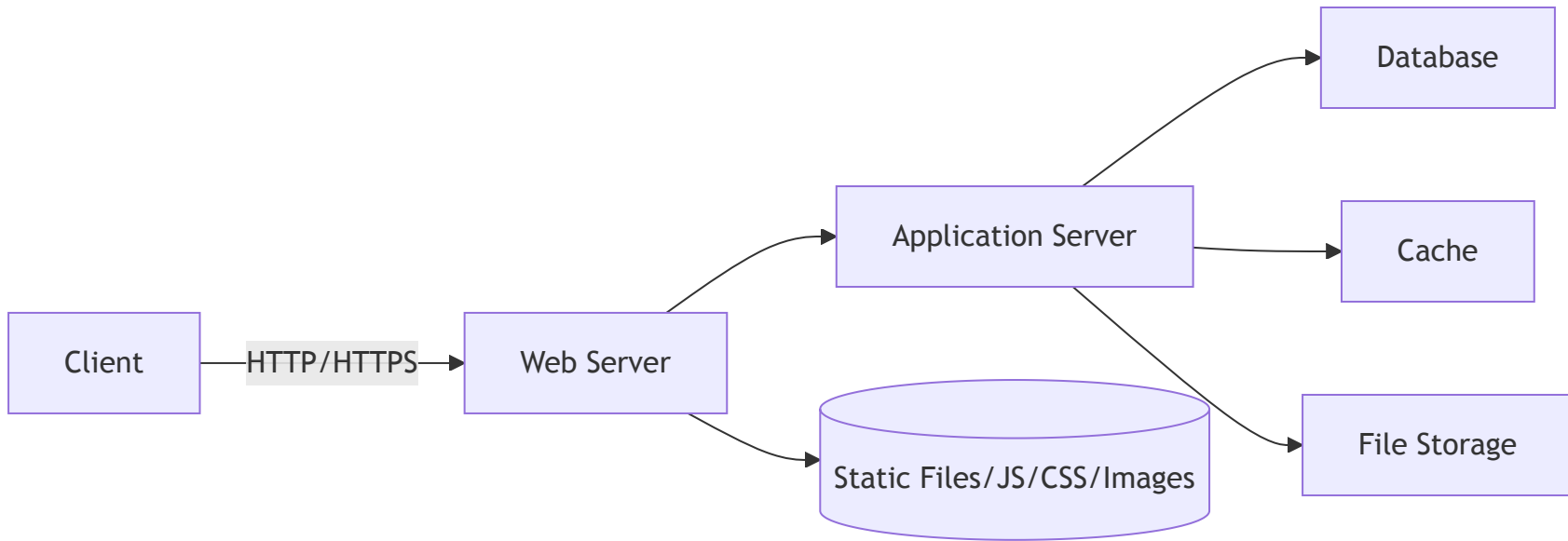
前后端运行方式

基于 JavaScript 实现前后端代码的开发



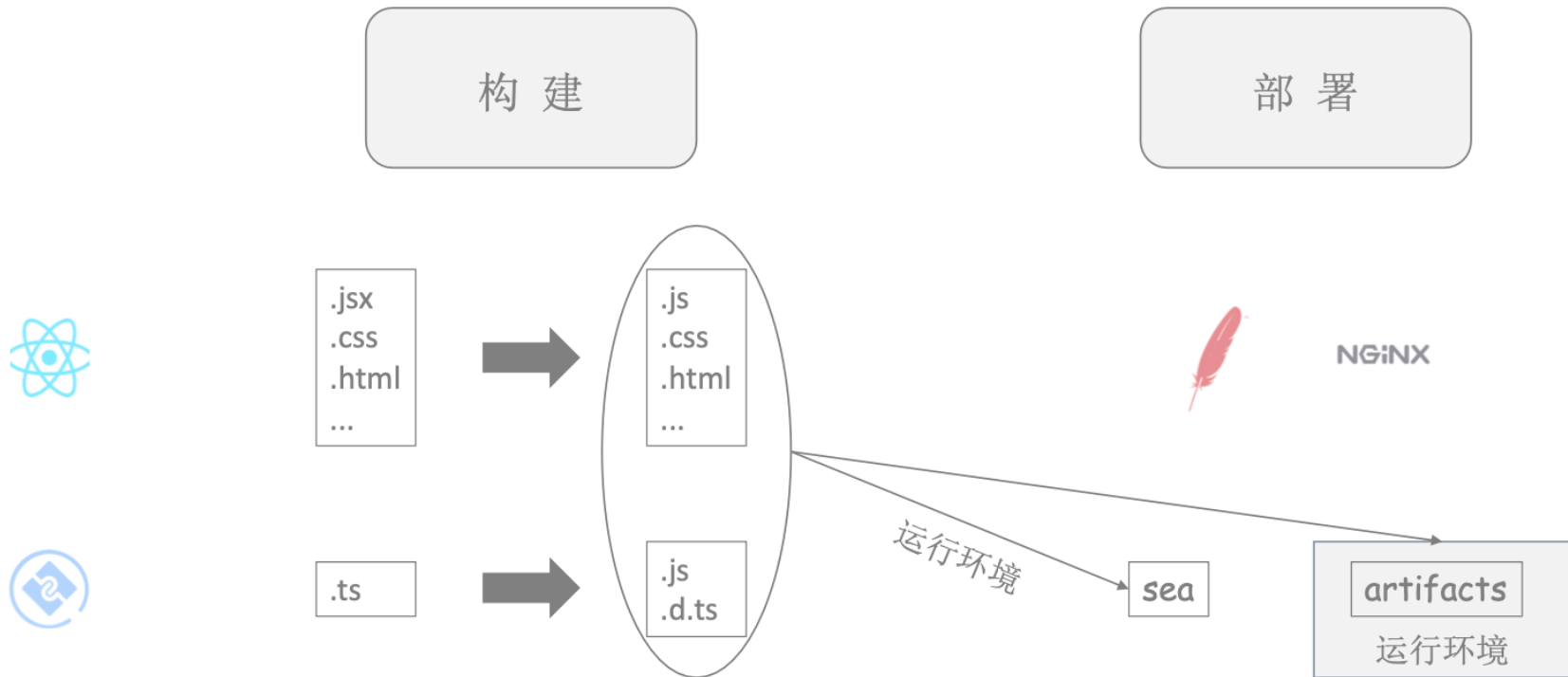
V8 作为 Google Chrome 的一部分，也被用于 Node.js 将 JavaScript 代码编译成汇编代码。

单体架构



- 前端构建产物: `index.html` , `assets/*`
- 后端构建产物: `*.js` , `*.d.ts` , 生产环境依赖

Web 应用构建部署



部署方式

传统服务器部署

- 准备服务器资源（物理服务器/虚拟机）
- 安装运行时环境（例如 nodejs、数据库等）
- Nginx / Apache HTTP 服务配置
- 上传构建产物
- 使用 Node 或者 PM2 等进程管理工具启动服务
- 安全加固（防火墙、证书）
- 服务器维护（日志监控、定期备份）

进程启动

PM2 是一个守护进程管理工具，帮助您管理和守护您的应用程序

PM2 以简单直观的 CLI 命令行方式进行工作，可以实现对 Java、Node.js、Python 等不同编程语言开发的应用进程的管理与守护。

```
pm2 start -n demo bootstrap.js //node.js

pm2 start -n demo java -- -jar ./demo.jar //Java

pm2 start -n demo python_web.py --interpreter python3
```

服务器端 PM2 组件的安装

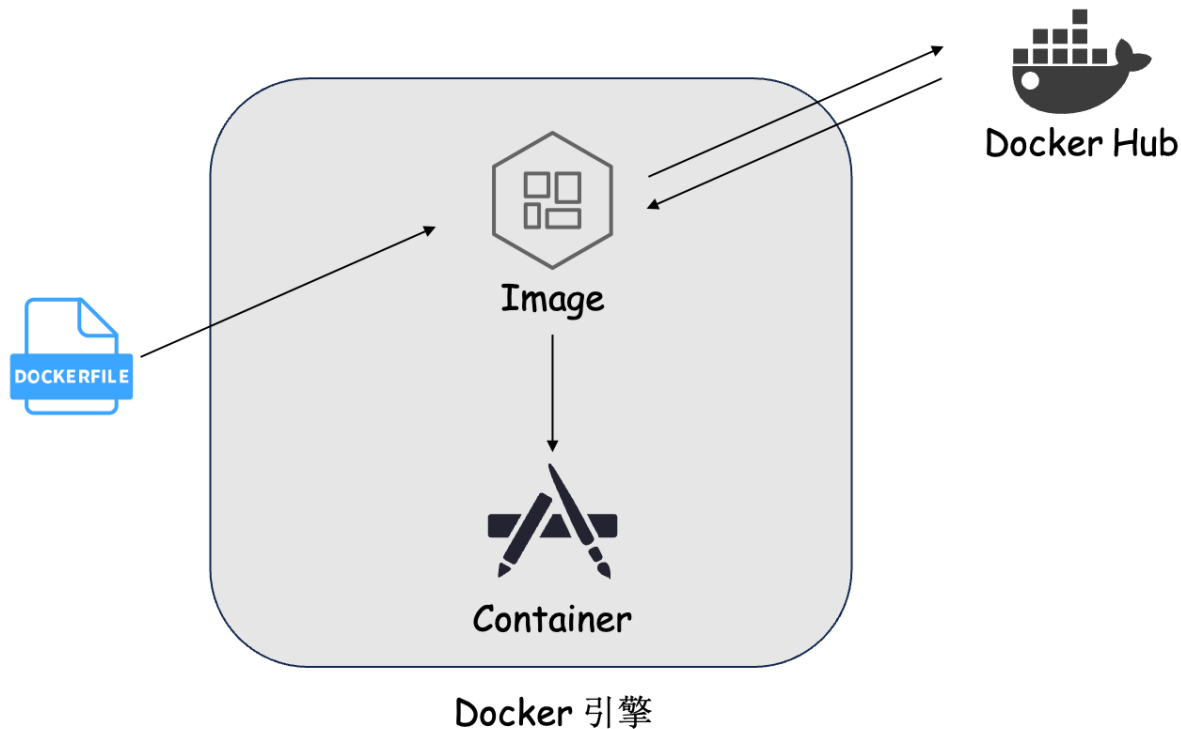
```
npm install -g pm2
```

```
fan@Finleys-MacBook-Pro backend % pm2 list
```

id	name	mode	u	status	cpu	memory
0	backend	fork	0	online	0%	57.6mb

容器化部署

将应用程序与其运行环境打包成一个轻量级、可移植的容器



容器启动

基于 Docker 镜像构建执行容器以运行

Docker Desktop

Search for local and remote images, containers, and more...

🔍

mdock

Containers

Images

Volumes

Dev Environments BETA

Docker Scout

Learning Center

EXTENSIONS

PGAdmin4

Add extension

Containers

Give feedback

Container CPU usage ⓘ
1.06% / 1000% (10 cores allocated)

Container memory usage ⓘ
127.45 MB / 15.1 GB (10 cores allocated)

Show charts ▾

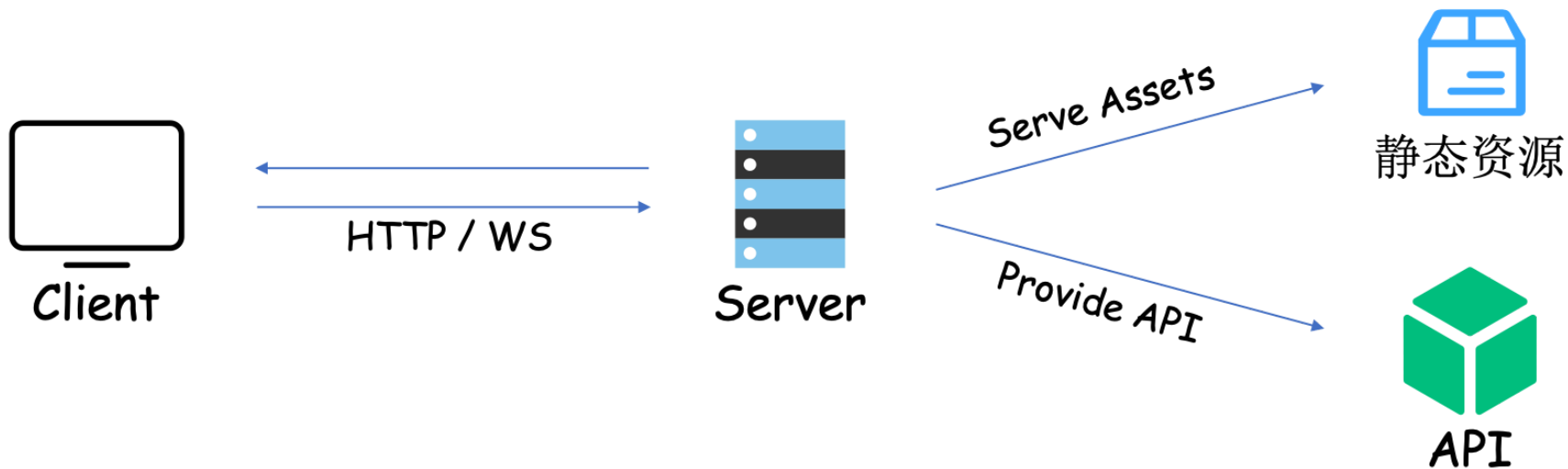
Search

Only show running containers

	Name	Image	Status	CPU (%)	Ports	Actions
<input type="checkbox"/>	<div><div><div></div><div>welcome-to-docker</div><div>f404a400c0ad</div></div></div>	docker/welco	Running	1.43%	52:52	<div><div></div><div></div><div></div></div>
<input type="checkbox"/>	<div><div><div></div><div>nginx</div><div>bad116f4586</div></div></div>	nginx:latest	Running	0%	80:80	<div><div></div><div></div><div></div></div>

文件分发方式

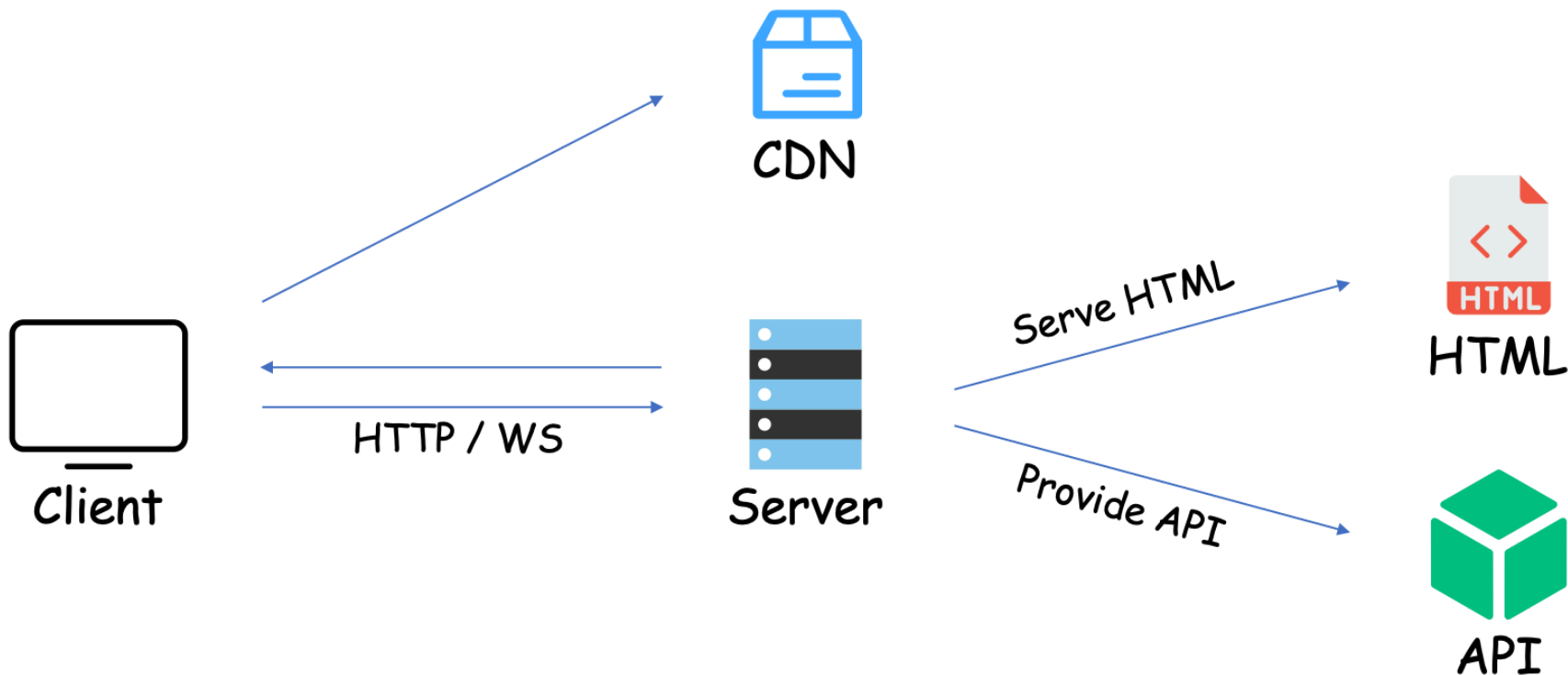
一体化分发



一台 Web Server 同时负责前端资源和后端数据接口的服务提供。

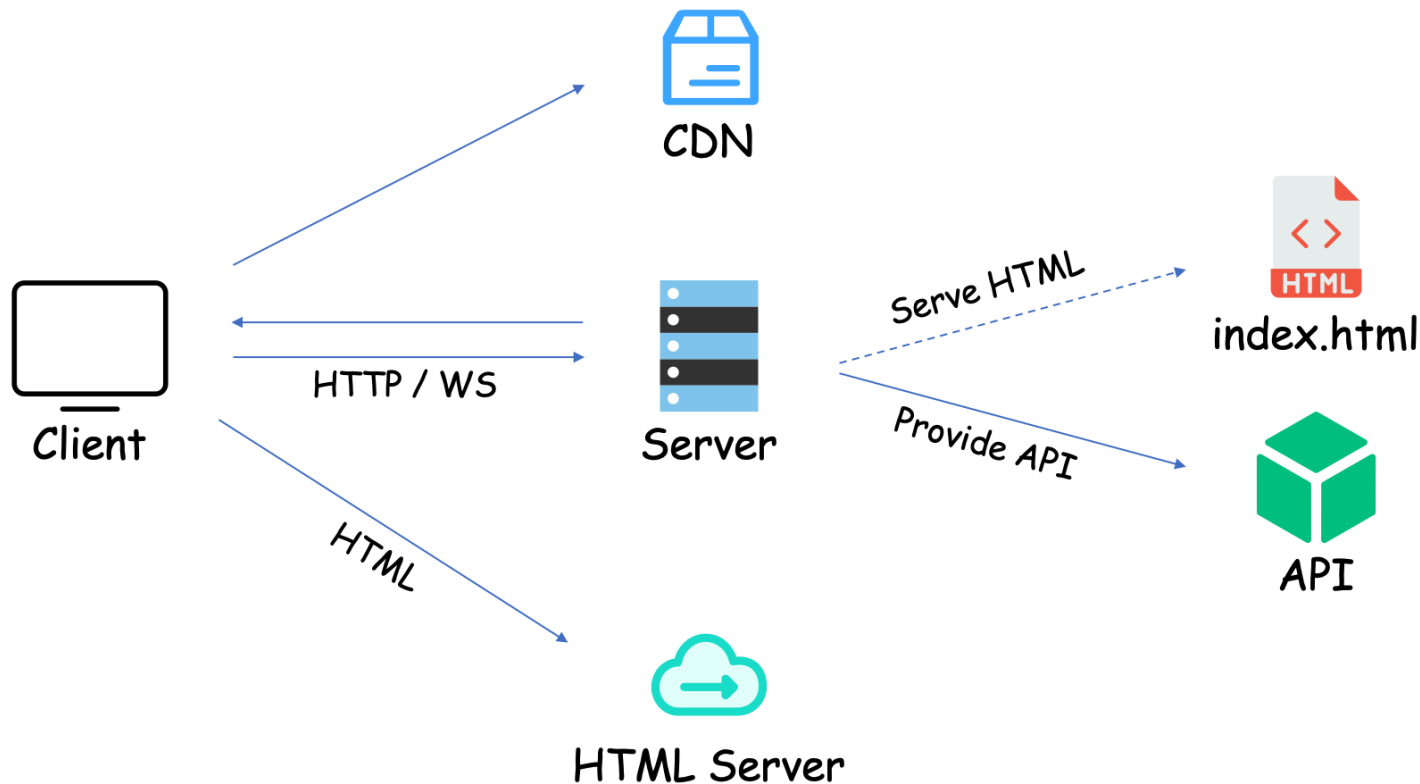
文件分发方式

CDN + Web Server, 将静态资源文件基于内容分发网络进行托管



文件分发方式

CDN + HTML Server + Web Server, 将静态资源文件基于内容分发网络进行托管



静态资源文件托管

将前端构建的产物，交由后端服务托管

后端安装静态资源托管依赖

```
npm i @midwayjs/static-file@3 --save
```

导入配置

```
//configuration.ts
import * as staticFile from '@midwayjs/static-file';
@Configuration({
  imports: [
    ...
    staticFile
  ],
})
export class MainConfiguration {
}
```

```
├─ src
├─ public
│   ├── index.html
│   ├── index.css
│   └─ index.js
```

静态资源文件托管

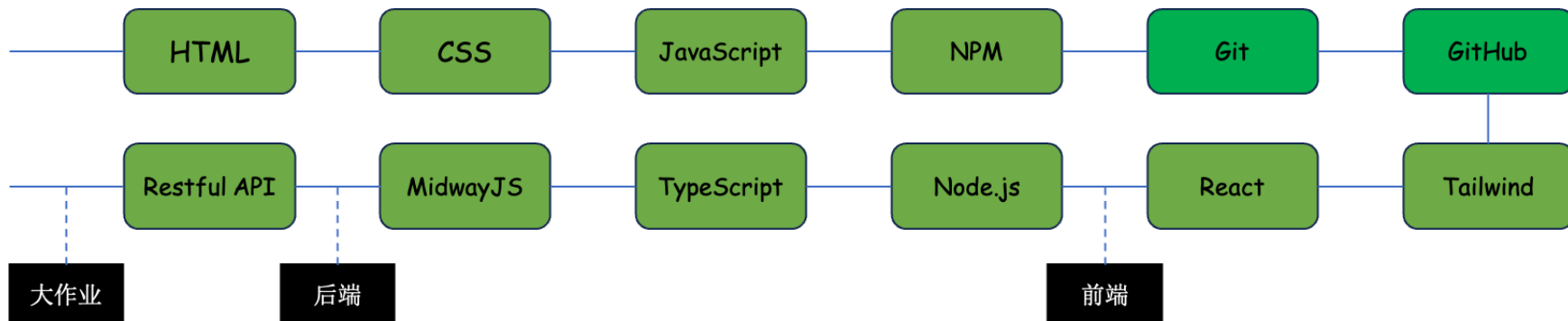
■ 静态资源映射路径配置

```
//config.default.ts
export default {
  staticFile: {
    dirs: {
      default: {
        prefix: '/',
        dir: 'public',
      },
      assets: {
        prefix: '/assets',
        dir: 'public/assets',
      },
    },
  },
} as MidwayConfig;
```

■ 前后端构建内容整合: 在后端构建完成的文件夹内，创建一个 `public` 文件夹，放置前端构建产物。

- ├─ index.js // 后端通过 ncc 打包的 js 文件，嵌入了依赖
- ├─ xxx.db // SQLite 数据库文件
- ├─ build/ //文件夹 .node 文件，由 npm run bundle 时自动生成
- └─ public/ // 文件夹，手动创建，用于放置前端构建产物

内容回顾



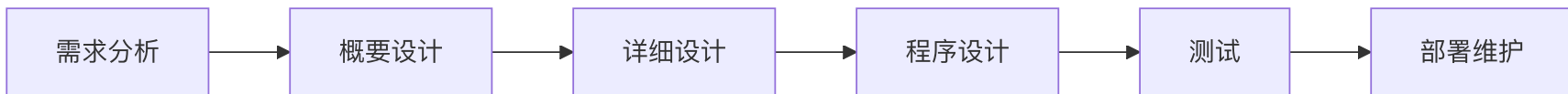
案例分析

一个有时间显示、有数据展示的 Web 界面，无用户交互的情况下，经常会发现时间会出现卡顿和跳变的情况，且当用户点击时，页面响应用户的界面点击操作存在较大的延迟，结合本门课程中提到的 Web 应用相关的技术基础，分析一下，可能是由哪些原因造成的？

- 网络问题
- 前端渲染问题
- 后端数据查询效率低
- 服务器性能问题

AI 辅助软件开发

- AI 的能力可以贯穿整个软件开发生命周期



AI 的诱惑

- AI 负责生成局部的代码片段，需要开发者进行高层设计
- 提出简单、确定性的问题，避免 Garbage In, Garbage Out
- AI 生成的代码实现，是对开发者设计和重构能力的挑战

要想 AI 用得好，系统思维不能少。

AI 辅助软件开发

AI 赋能 Web 开发者的常见场景


- UI 组件生成。开发者规划好组件结构，并：
 - 明确使用的技术栈，指定框架
 - 添加关于组件内的元素、样式和交互逻辑的描述
- 类的定义和方法实现。开发者需要使用自然语言清晰描述：
 - 类内的属性以及其值类型、取值范围
 - 方法的名称、参数、返回值以及执行逻辑
- 代码注释生成
- 测试代码生成

大作业提交内容

- 系统完成功能点使用录屏
- 打包产物（使用单文件构建方式，同时提供 SQLite 文件）
- Readme文件
 - 前后端代码库地址（Github 链接）
 - 打包平台说明: Windows, Linux, MacOS
 - 提供额外实现的功能描述
 - 学习中印象最深刻的内容（可选）
 - 你对课程内容的改进建议（可选）

Web 应用开发学习资源

- MDN Web Docs, https://developer.mozilla.org/en-US/docs/Learn_web_development
- ReactJS, <https://react.dev/>
- Vite, <https://vitejs.dev/guide/>
- TailwindCSS, <https://tailwindcss.com/>
- Node.js, <https://nodejs.org/docs/latest/api/>
- MidwayJS, <https://midwayjs.org/>



谢谢