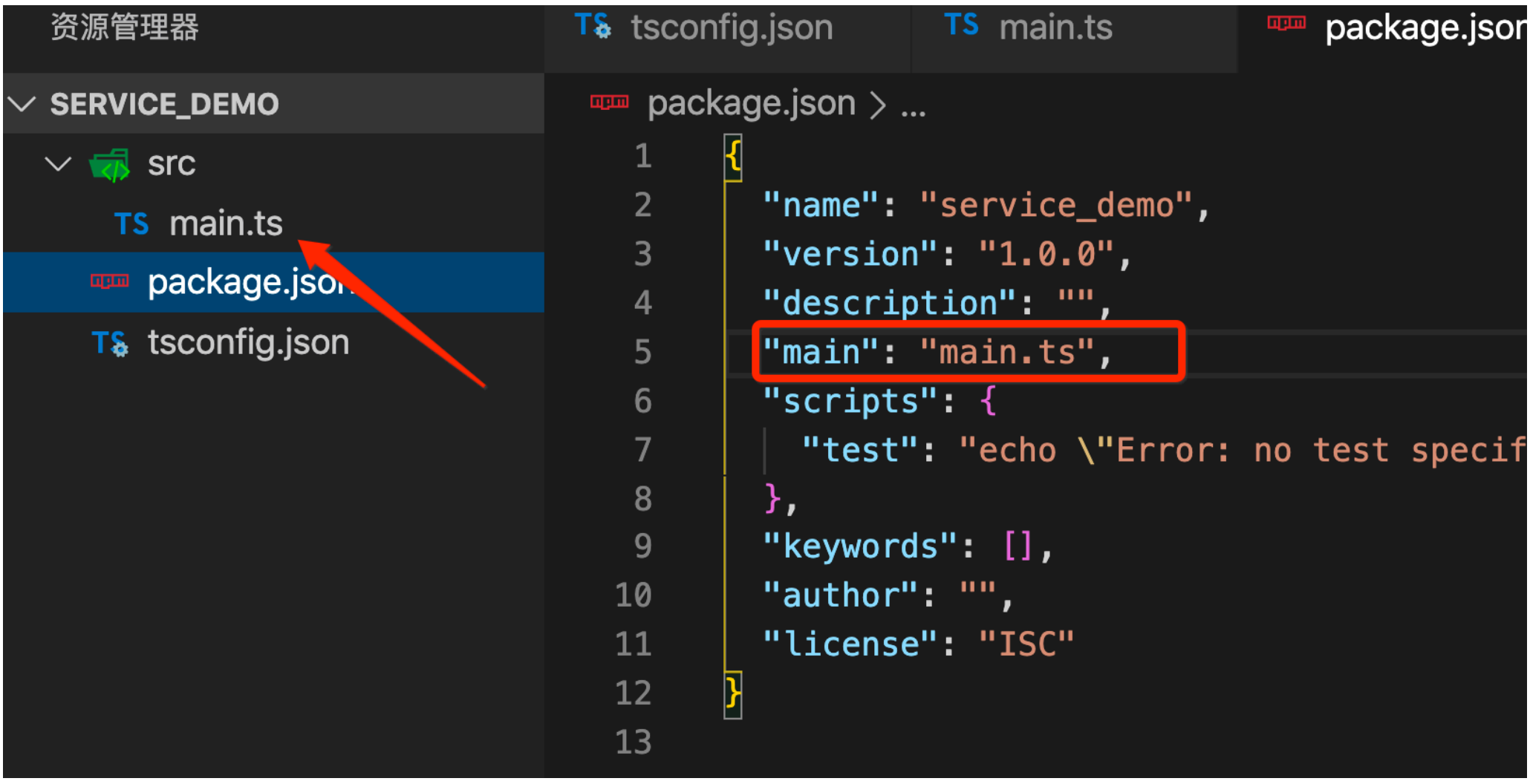
TypeScript 创建第一个后台项目

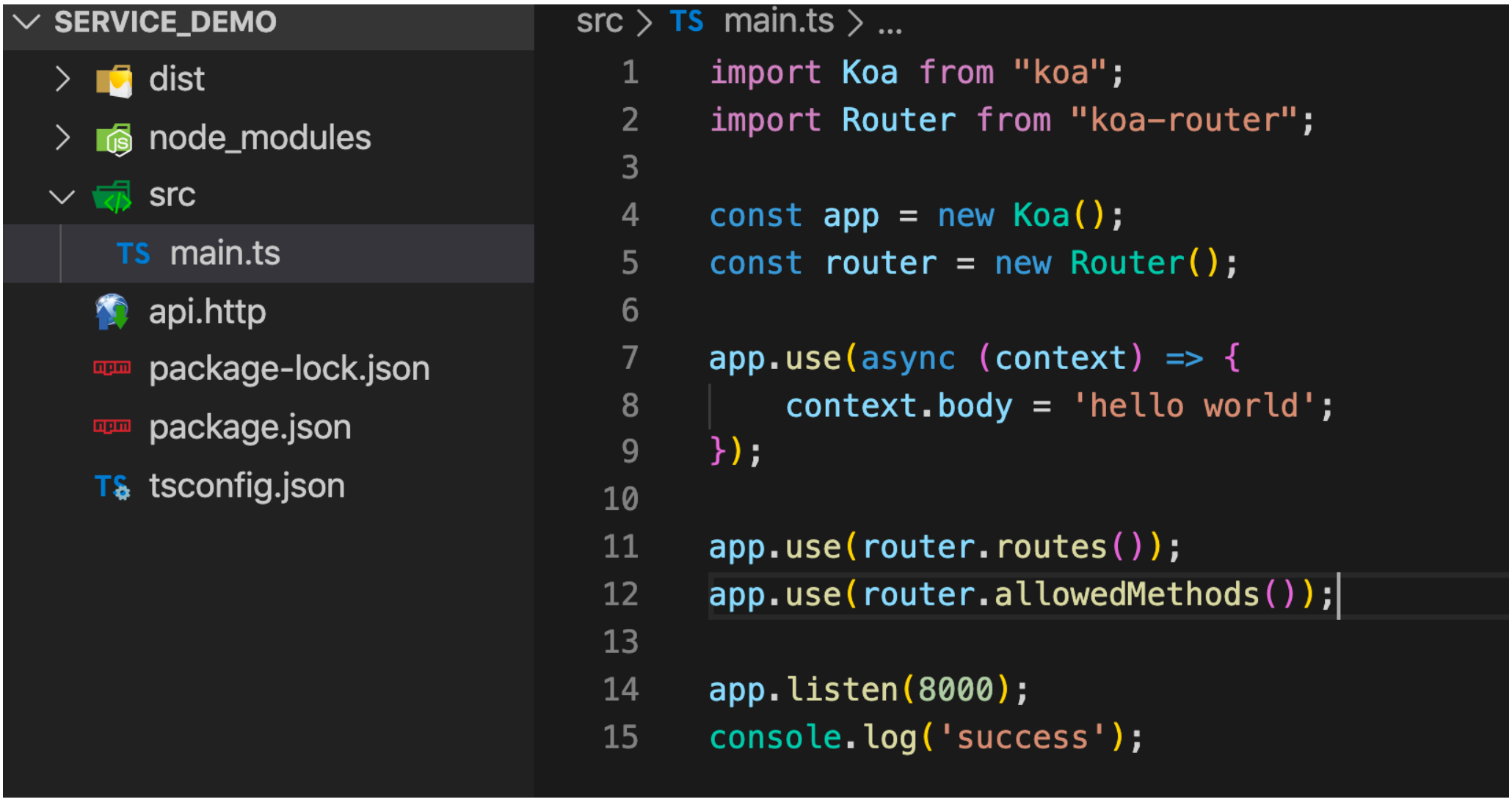
1.创建文件夹service_demo

2.npm init -y

3.tsc --init

4.修改配置文件package.json tsconfig.json

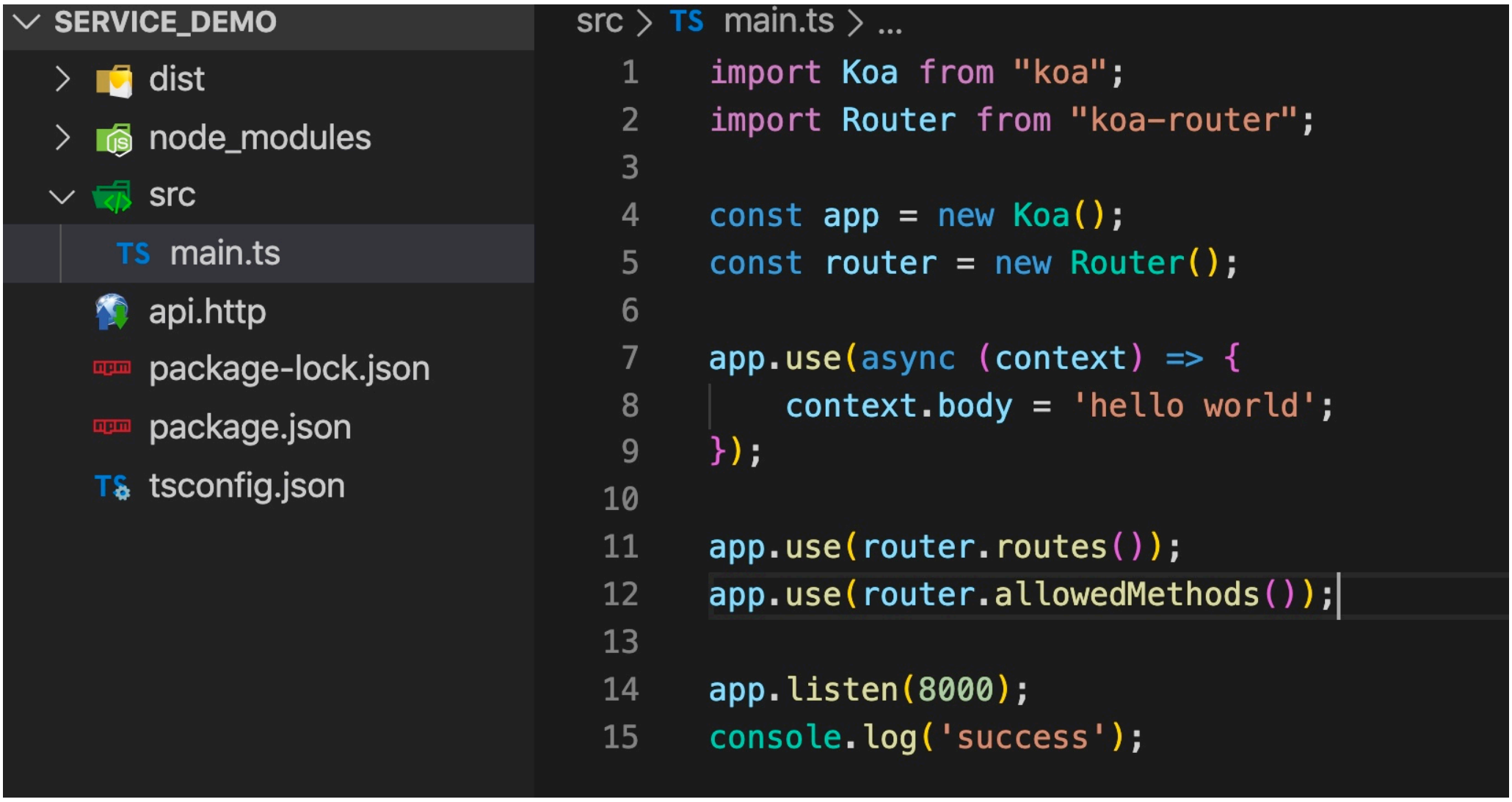




5.安装库

```
(tool)
npm i @types/koa -D
npm i @types/koa-router -D
npm i @types/node -D
npm i dotenv -D
npm i typescript -D

npm i nodemon -g

(dependency)
npm i axios
npm i koa
npm i koa-router
npm i rxjs
```

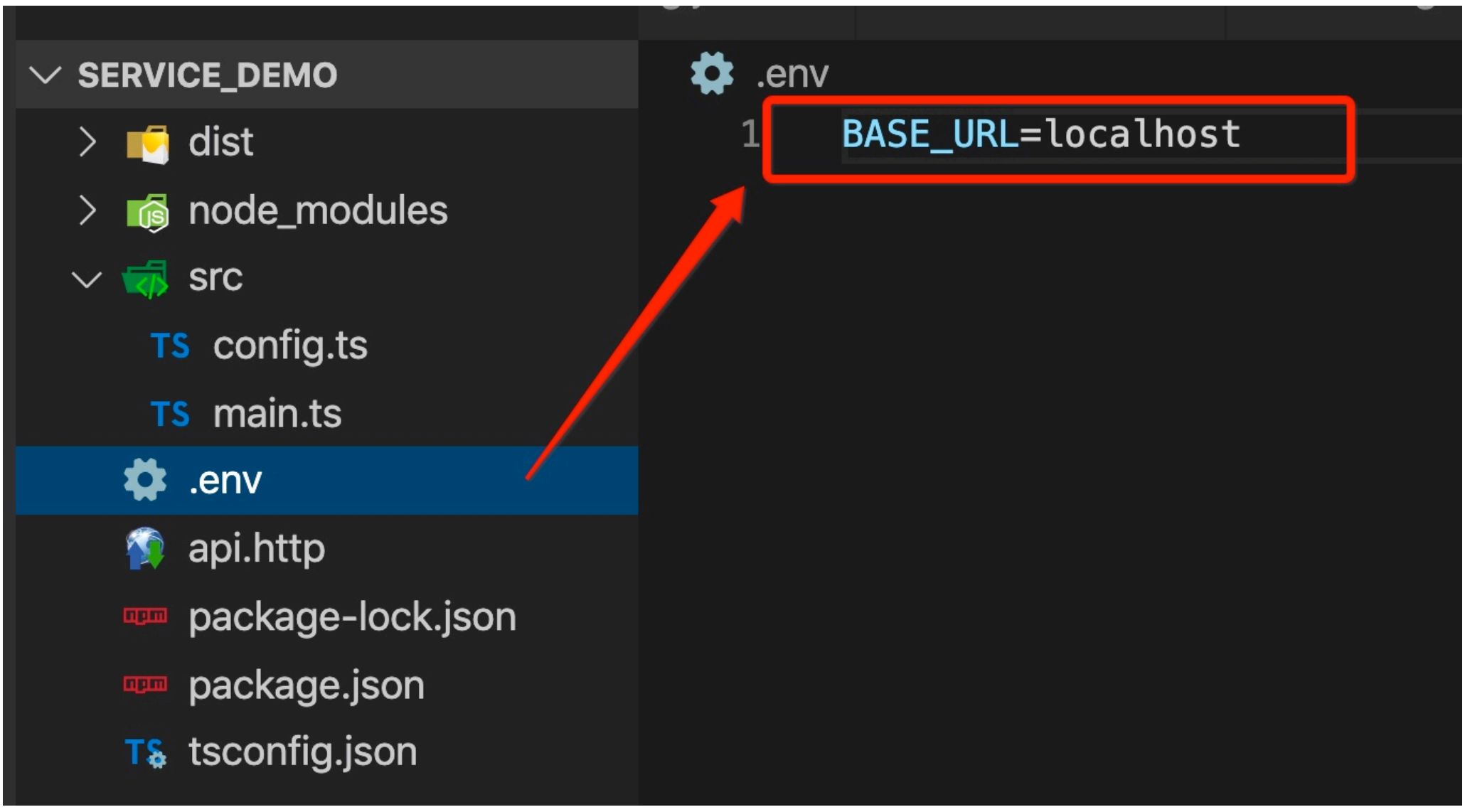
6.编写hello world



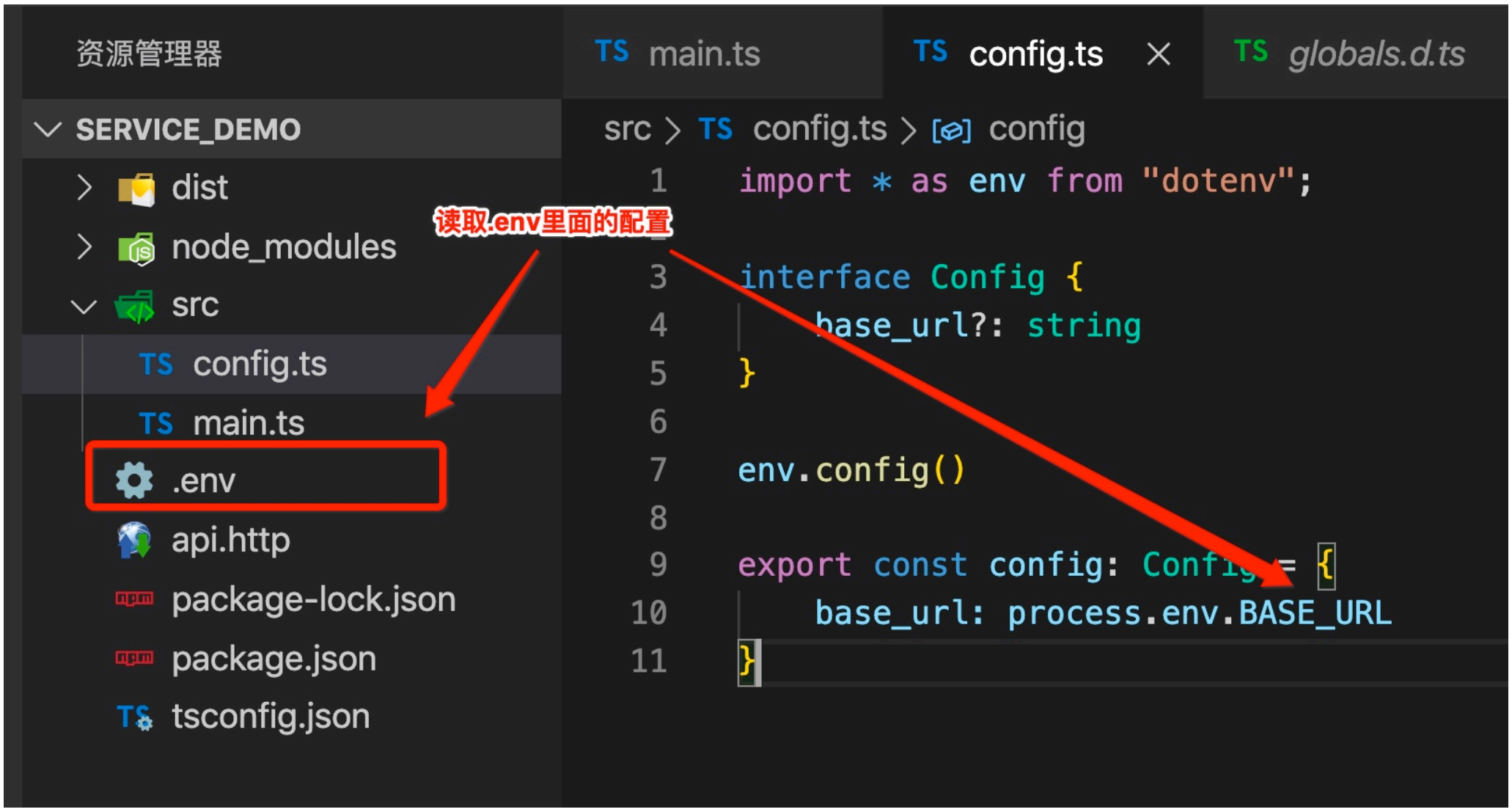
7.执行tsc编译

8.执行 node ./dist/main.js

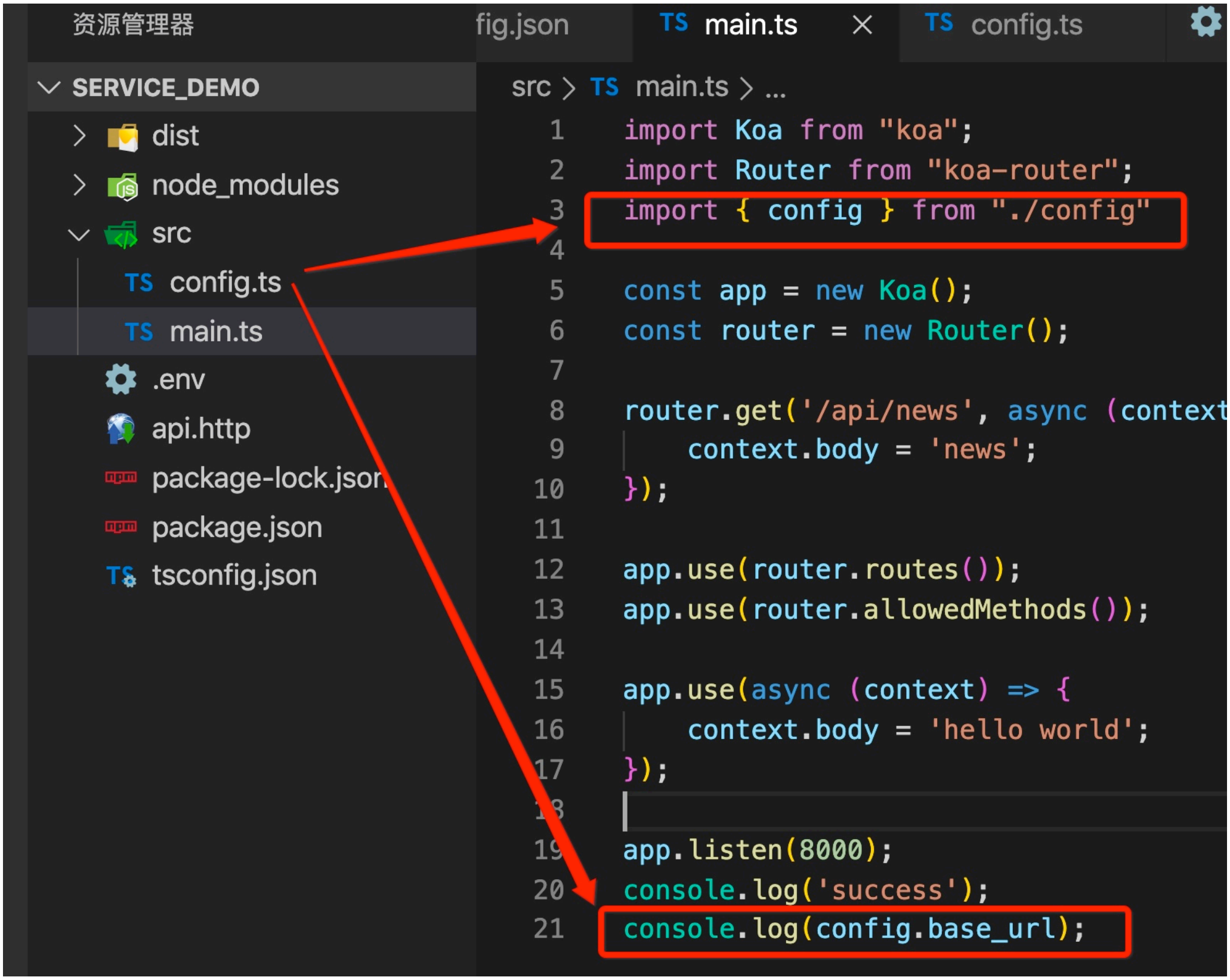
9.配置本地项目env



10.配置本地项目config



11.使用上面配置的config



11.新建service

```
import axios, { AxiosInstance } from "axios";
import { config } from './config'
import { from } from "rxjs";

export class Service {

    async combine() {
        let result = await Promise.all([this.getNews(), this.getDiscover()]);
        return {
            'news': result[0],
            'services': result[1]
        };
    }

    async getDiscover() {
        const host: string = config.discover_url as string;
        try {
            const client = this._axiosFactory(host);
            const res = await client.get('/api/servicemanager/v0/discoverallservices/v1?region=cn&locale=zh-cn
&client=IOS&brand=1', {
                headers: {
```

```
                'AppVersion': '10.3.0',
                'AppKey': '51dee986-d5b2-4af1-b129-4eaedc6c32b6'
            }
        });
        return res.data;
    } catch (e) {
        console.log(e);
    }
}

async getNews() {
    const host: string = config.news_url as string;
    try {
        const client = this._axiosFactory(host);
        const res = await client.get('/api/news', {
            headers: {
                'X-AppKey': '2014_MyBMW837',
                'x-btcapi-usid': '05e4c12d-2182-4862-928a-a08cc79a5dec'
            }
        });
        return res.data;
    } catch (e) {
        console.log(e);
    }
}

private _axiosFactory(host: string) {
    return axios.create({
        baseURL: host,
        timeout: 30000
    });
}
```

12.main.ts文件调用service

SERVICE_DEMO

- dist
- node_modules
- src
  - config.ts
  - main.ts
  - service.ts
- .env
- api.http
- package-lock.json
- package.json
- tsconfig.json

src > main.ts > ...

```ts
import Koa from "koa";
import Router from "koa-router";
import { config } from "./config"
import { Service } from "./service";

const app = new Koa();
const router = new Router();

router.get('/api/news', async (context) => {
    const service = new Service();
    context.body = await service.getNews();
});

router.get('/api/servicemanager', async (context) => {
    const service = new Service();
    context.body = await service.getDiscover();
});

router.get('/api/commine', async (context) => {
    const service = new Service();
    context.body = await service.combine();
});
```

13.使用Rest client查看结果

资源管理器 | main.ts | service.ts | api.http | config.t

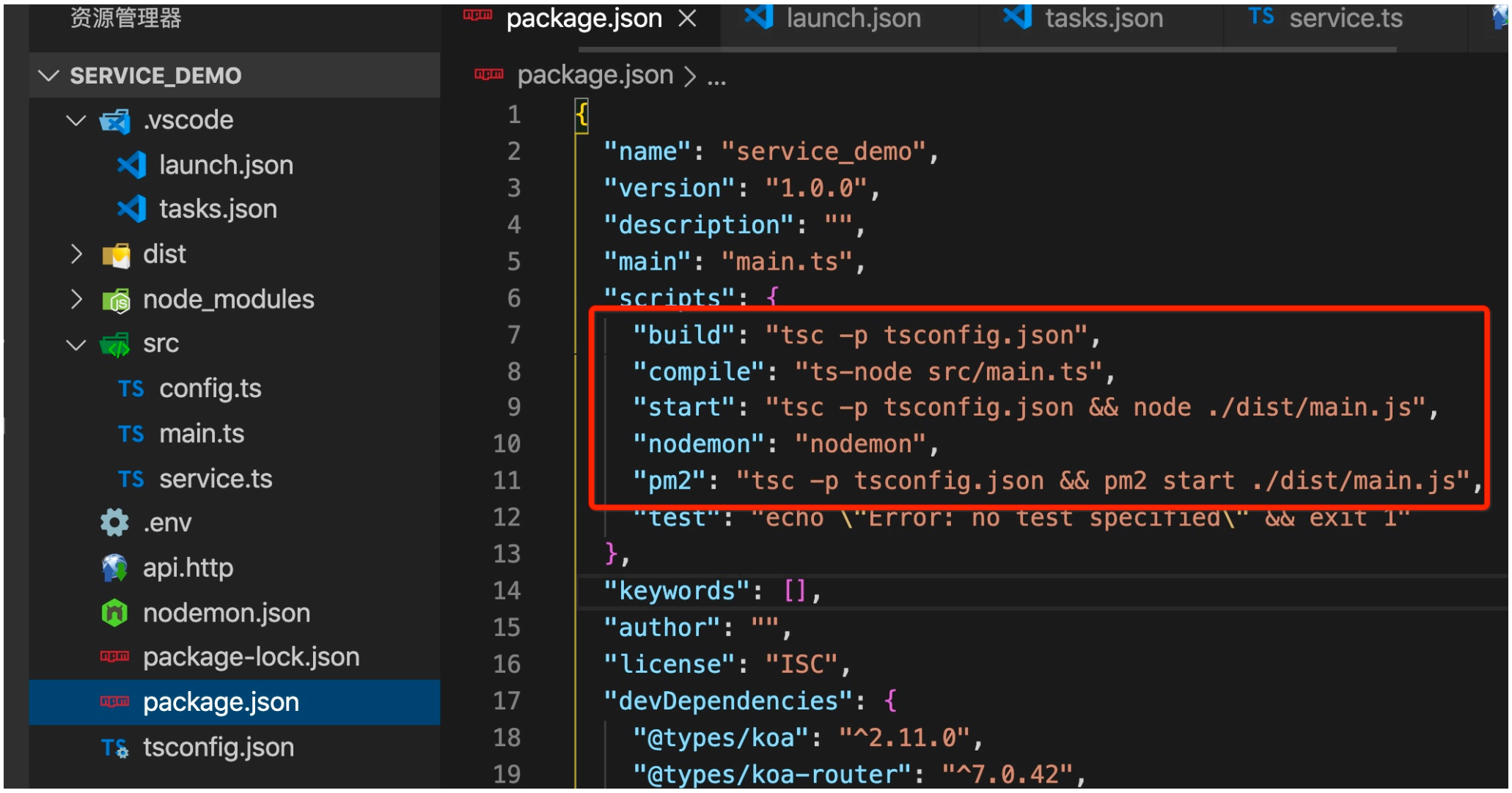
SERVICE_DEMO

- dist
- node_modules
- src
  - config.ts
  - main.ts
  - service.ts
- .env
- api.http
- package-lock.json
- package.json
- tsconfig.json

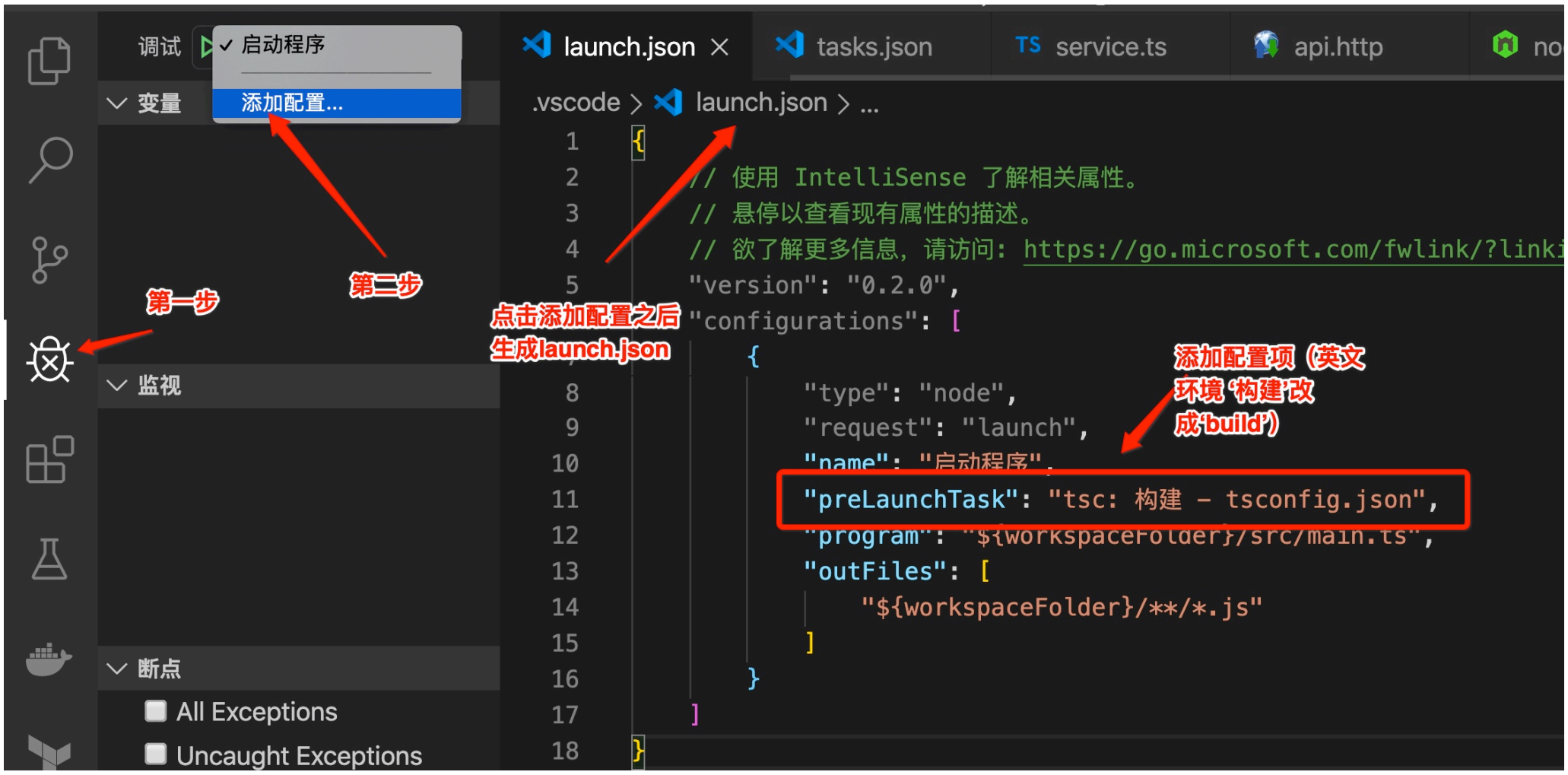
api.http > ...

```http
###
Send Request
GET http://localhost:8000/

###
Send Request
GET http://localhost:8000/api/news

###
Send Request
GET http://localhost:8000/api/servicemanager

###
Send Request
GET http://localhost:8000/api/commine
```

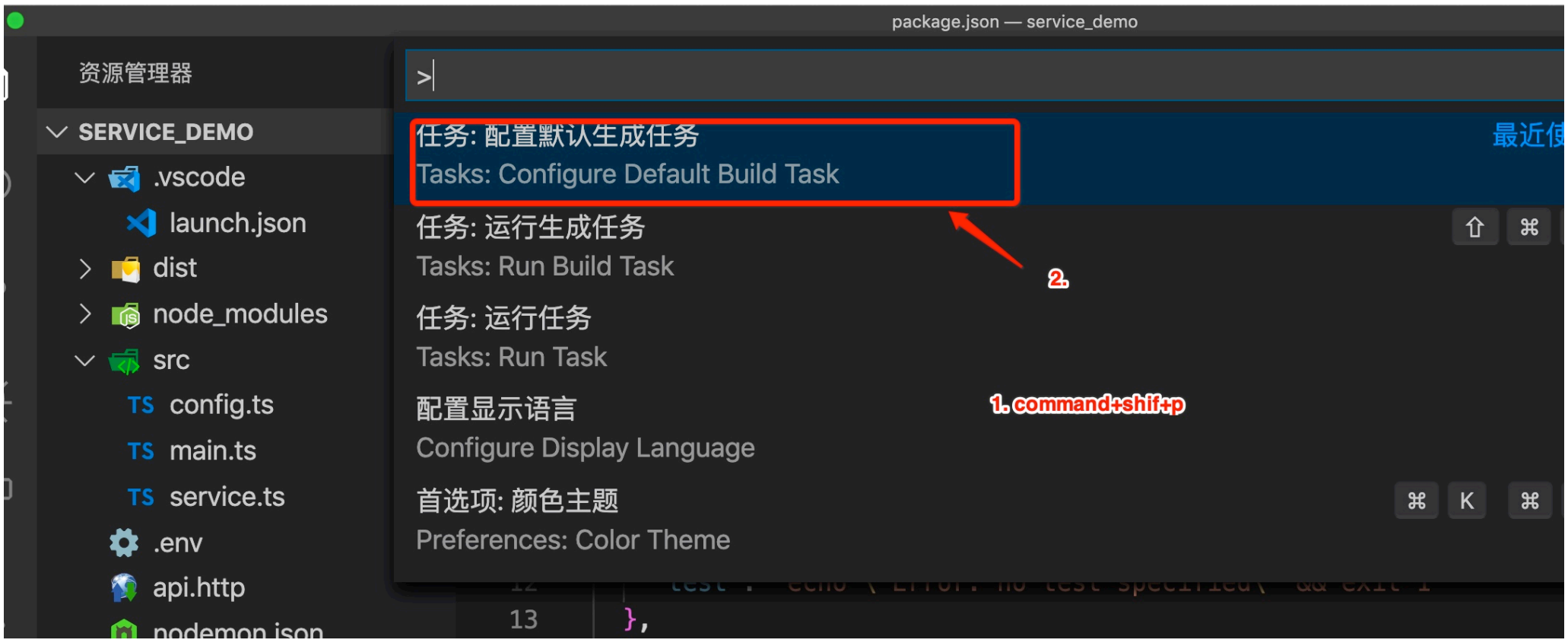
14.配置脚本

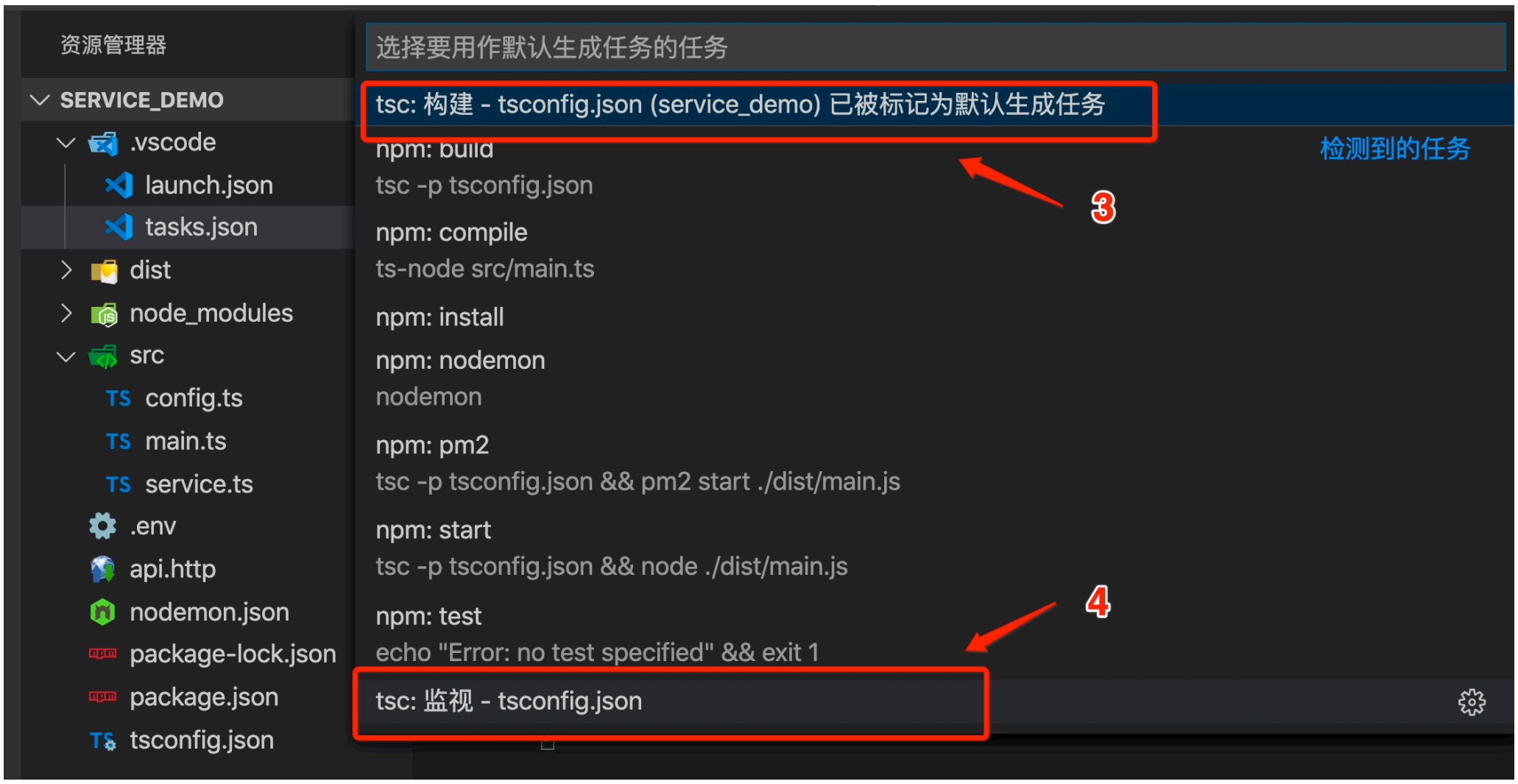


15.配置launch.json



16.配置tasks.json


package.json — service_demo
资源管理器
SERVICE_DEMO
.vscode
launch.json
dist
node_modules
src
config.ts
main.ts
service.ts
.env
api.http
nodemon.json
>
任务: 配置默认生成任务
Tasks: Configure Default Build Task
任务: 运行生成任务
Tasks: Run Build Task
任务: 运行任务
Tasks: Run Task
配置显示语言
Configure Display Language
首选项: 颜色主题
Preferences: Color Theme
2.
1. command+shif+p
13 },


资源管理器
SERVICE_DEMO
.vscode
launch.json
tasks.json
dist
node_modules
src
config.ts
main.ts
service.ts
.env
api.http
nodemon.json
package-lock.json
package.json
tsconfig.json
选择要用作默认生成任务的任务
tsc: 构建 - tsconfig.json (service_demo) 已被标记为默认生成任务
检测到的任务
npm: build
tsc -p tsconfig.json
npm: compile
ts-node src/main.ts
npm: install
npm: nodemon
nodemon
npm: pm2
tsc -p tsconfig.json && pm2 start ./dist/main.js
npm: start
tsc -p tsconfig.json && node ./dist/main.js
npm: test
echo "Error: no test specified" && exit 1
tsc: 监视 - tsconfig.json
3
4

资源管理器

package.json | tasks.json | launch.json | service.t

SERVICE_DEMO

- .vscode
  - launch.json
  - tasks.json
- dist
- node_modules
- src
  - config.ts
  - main.ts
  - service.ts
- .env
- api.http
- nodemon.json
- package-lock.json
- package.json
- tsconfig.json

.vscode > tasks.json > [ ] tasks > { } 0 > abc group

```jsonc
{
// 有关 tasks.json 格式的文档，请参见
    // https://go.microsoft.com/fwlink/?LinkId=733558
    "version": "2.0.0",
    "tasks": [
        {
            "type": "typescript",
            "tsconfig": "tsconfig.json",
            "problemMatcher": [
                "$tsc"
            ],
            "group": "build"
        },
        {
            "type": "typescript",
            "tsconfig": "tsconfig.json",
            "option": "watch",
            "problemMatcher": [
                "$tsc-watch"
            ],
            "group": {
                "kind": "build",
                "isDefault": true
            }
        }
    ]
}
```