

node接入层应用与实践

魏天亮

个人介绍



魏天亮

2016年起加入腾讯，曾参与开发QQ空间、情侣空间、空间小游戏等线上H5业务，目前主要负责QQ小程序、QQ小游戏等前端相关开发工作。

内容大纲

01 WebSocket多机多进程间通信

02 测试环境搭建

03 实时日志





开发

多机多进程间如何通信



开发

多机多进程间如何通信

产品、测试

怎么体验、测试



开发

多机多进程间如何通信

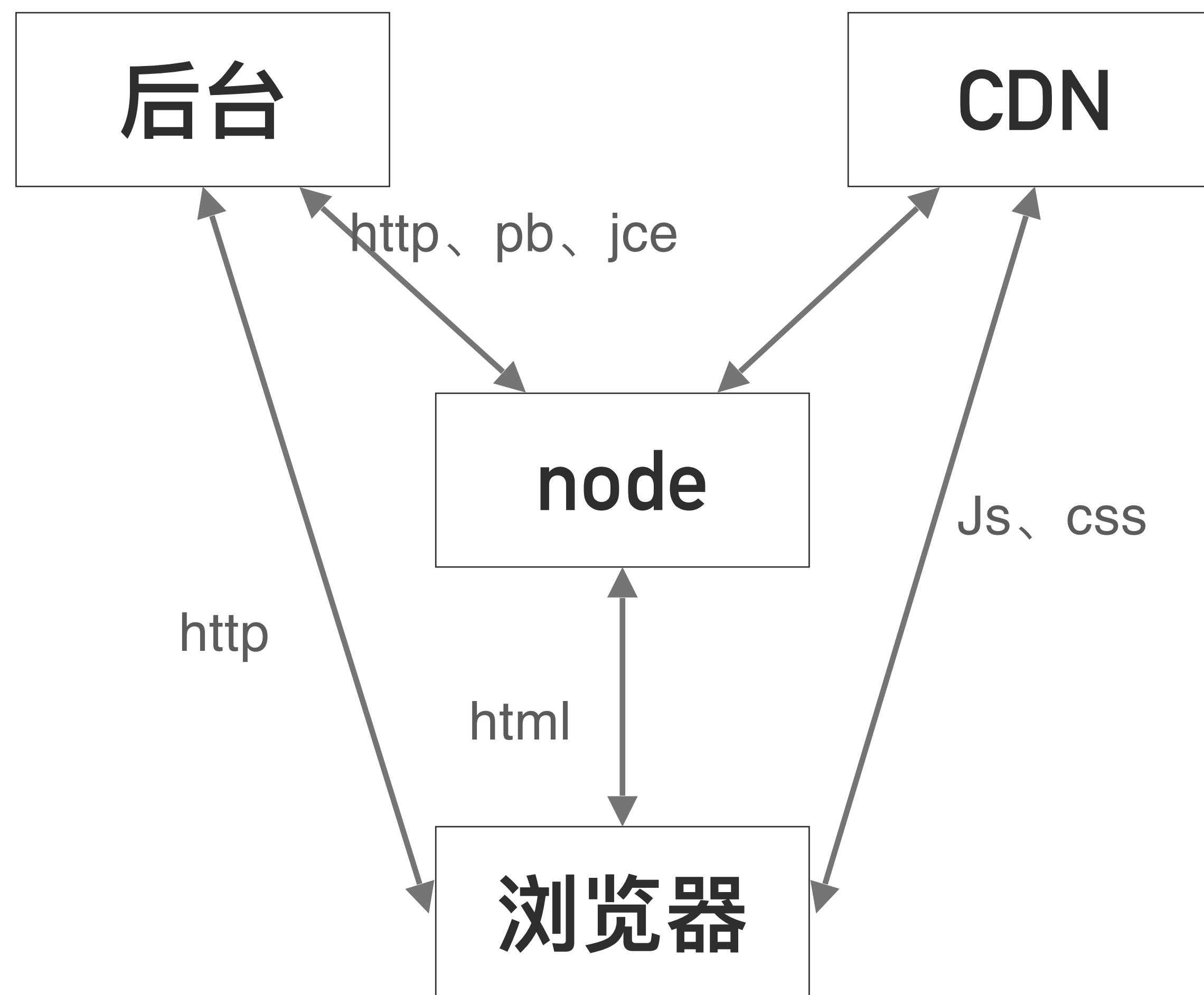
产品、测试

怎么体验、测试

用户

出现了问题如何定位

基础架构



多机通信

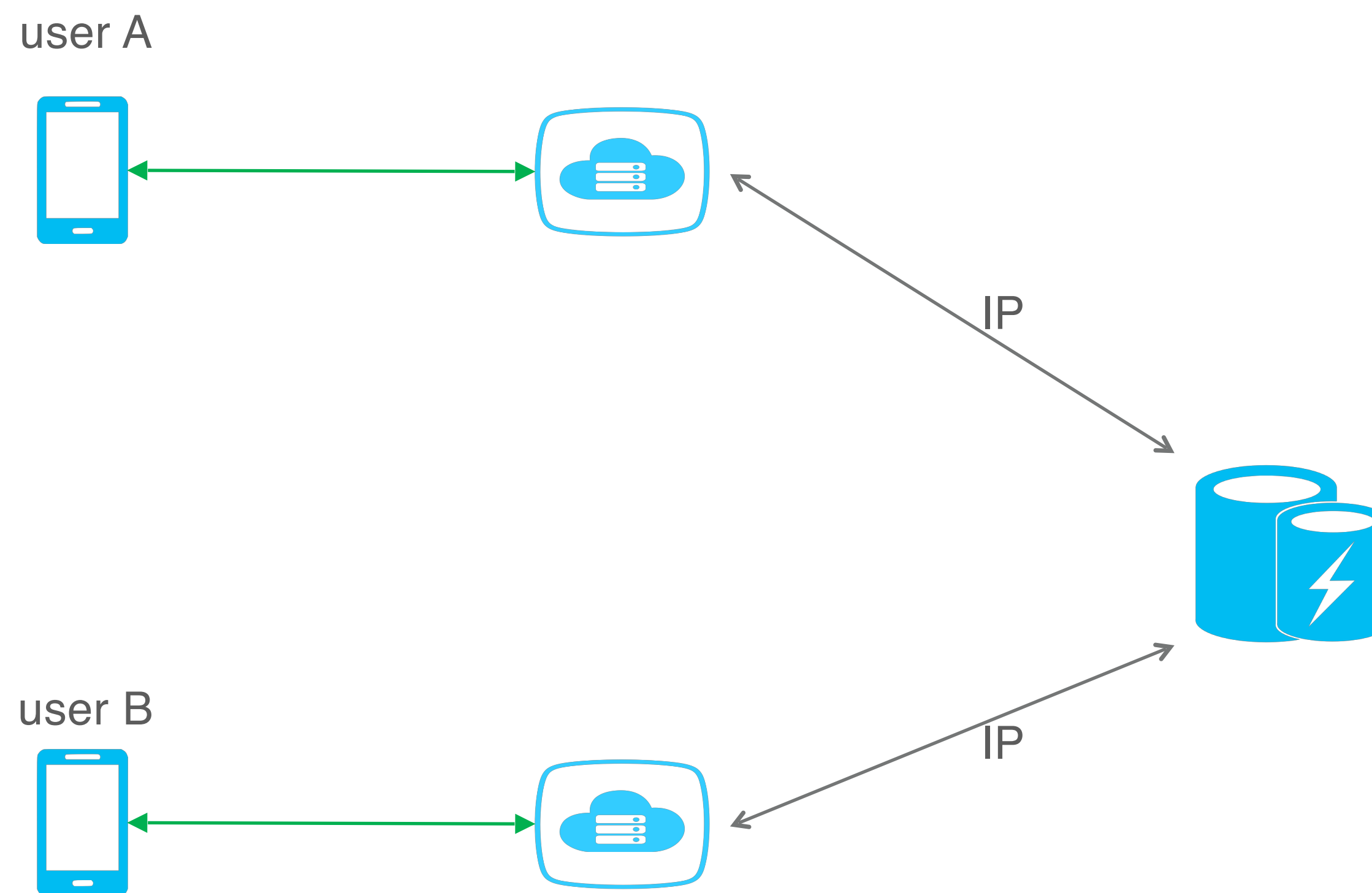
user A



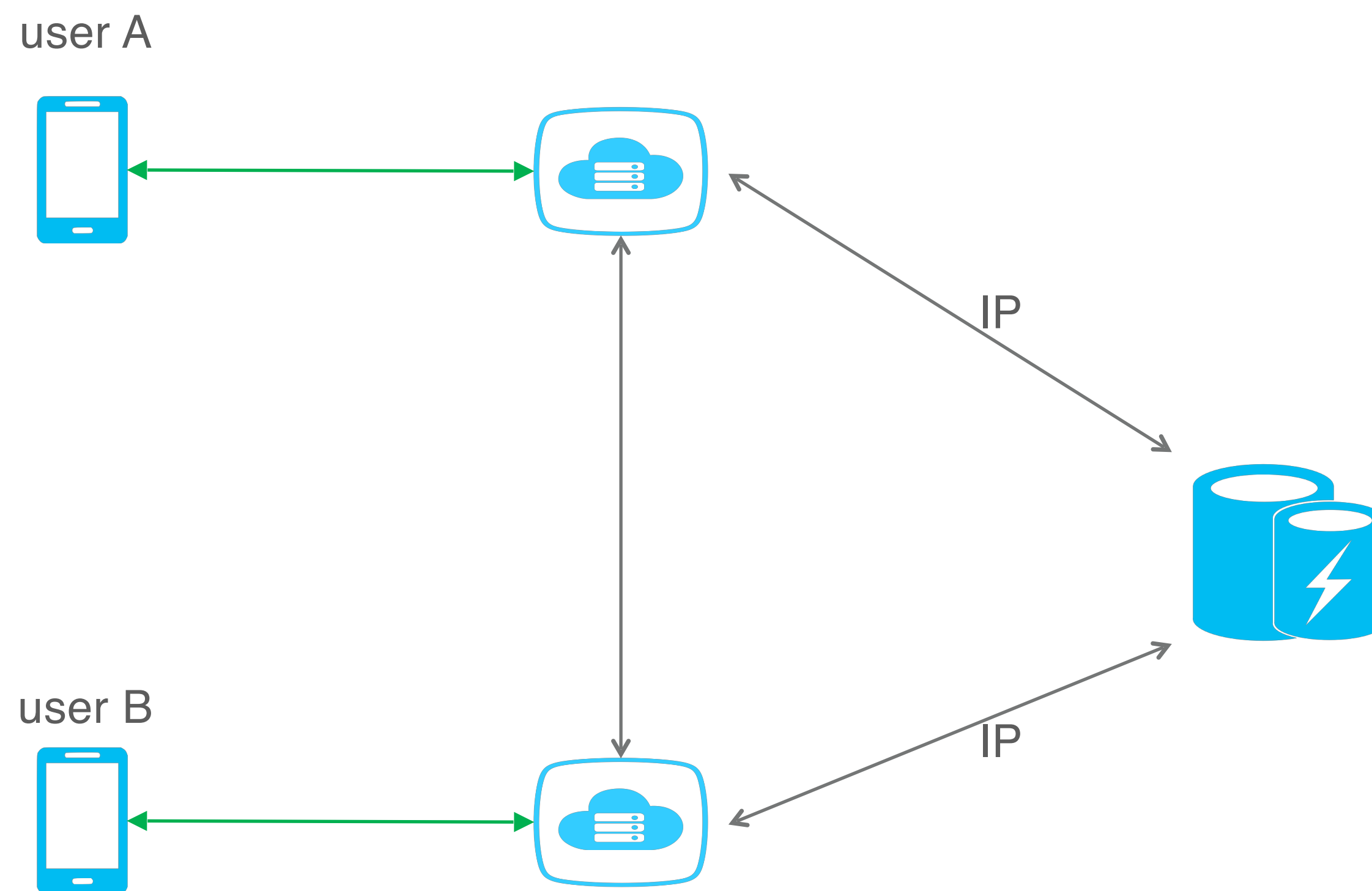
user B



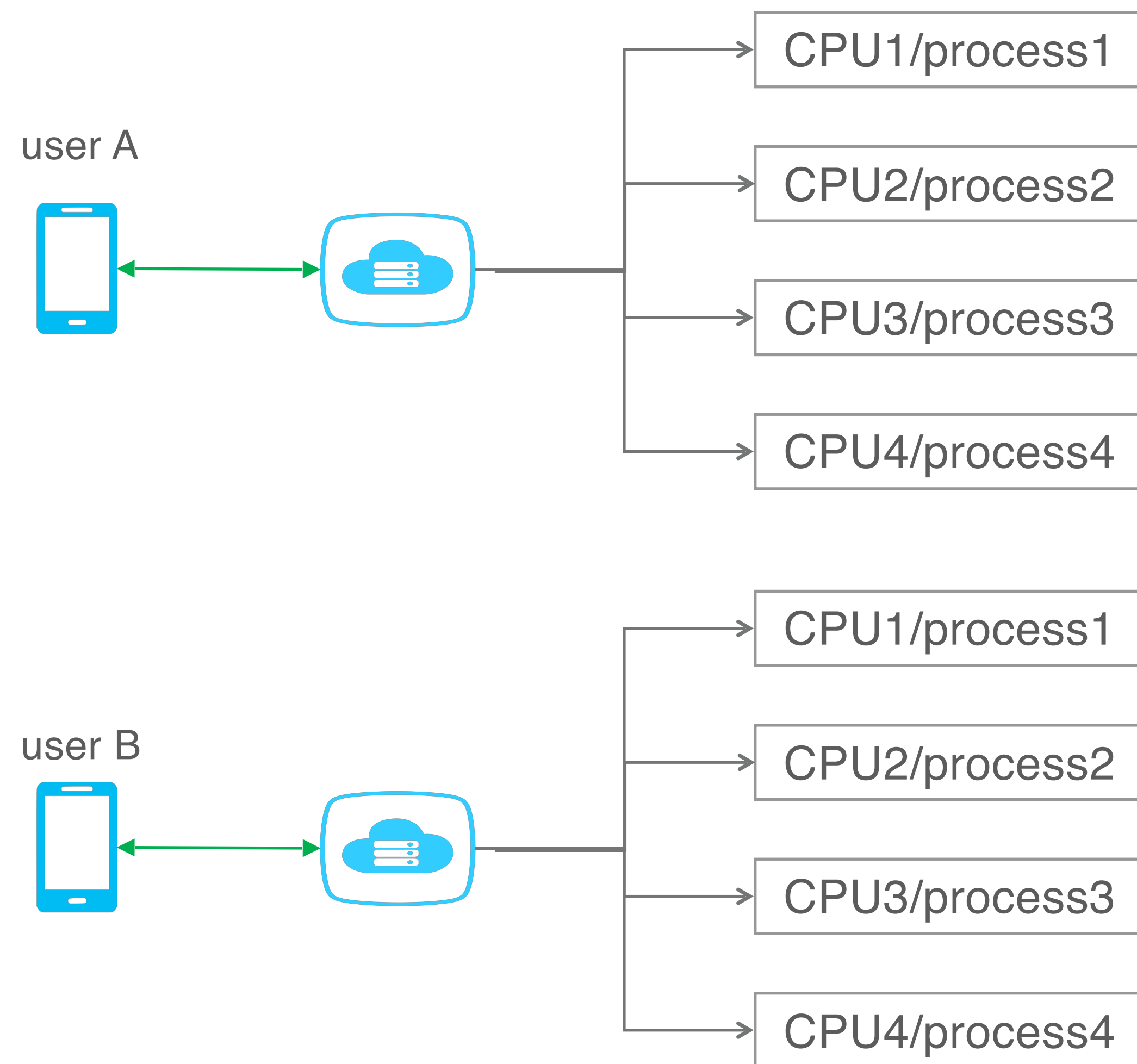
多机通信



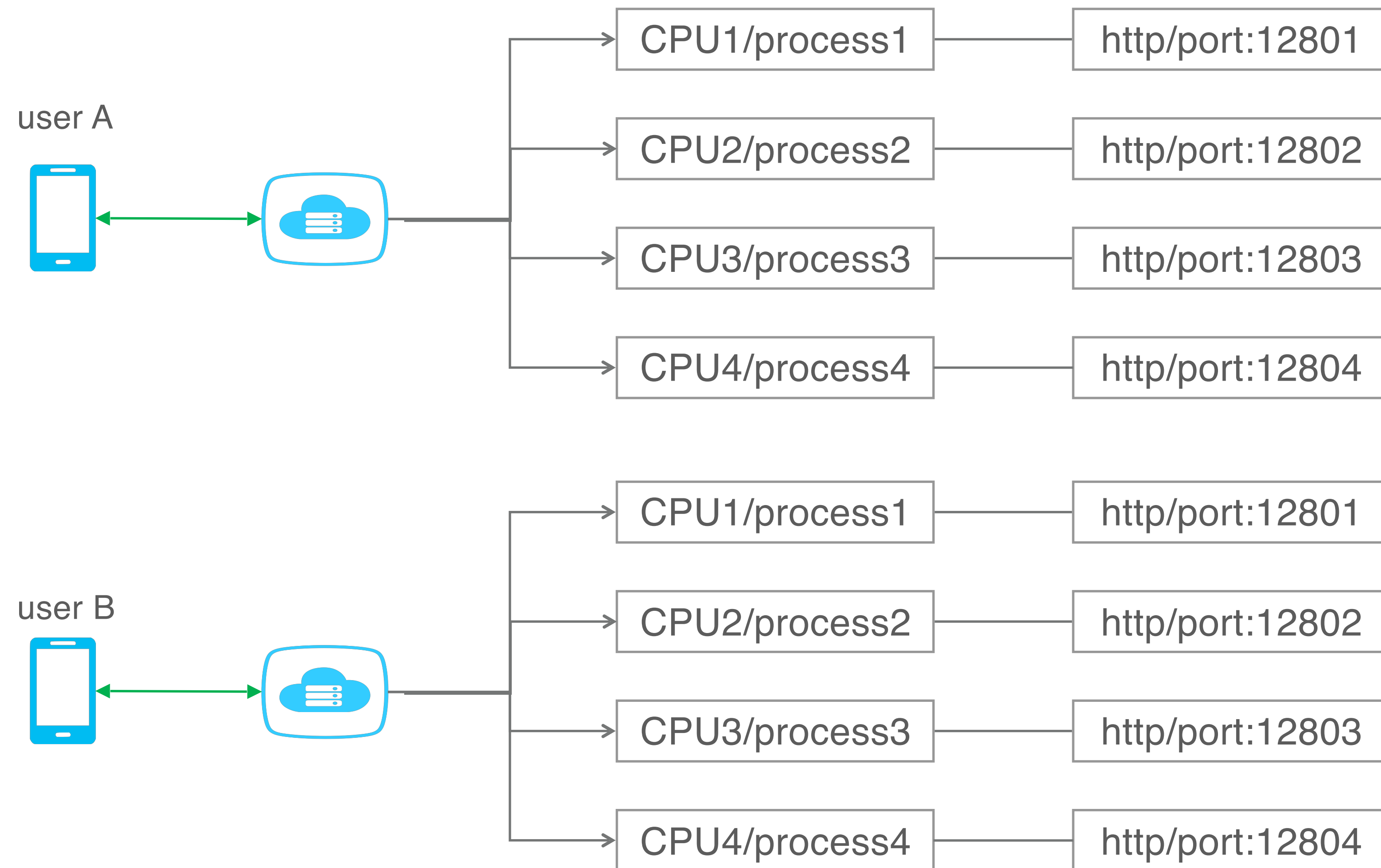
多机通信



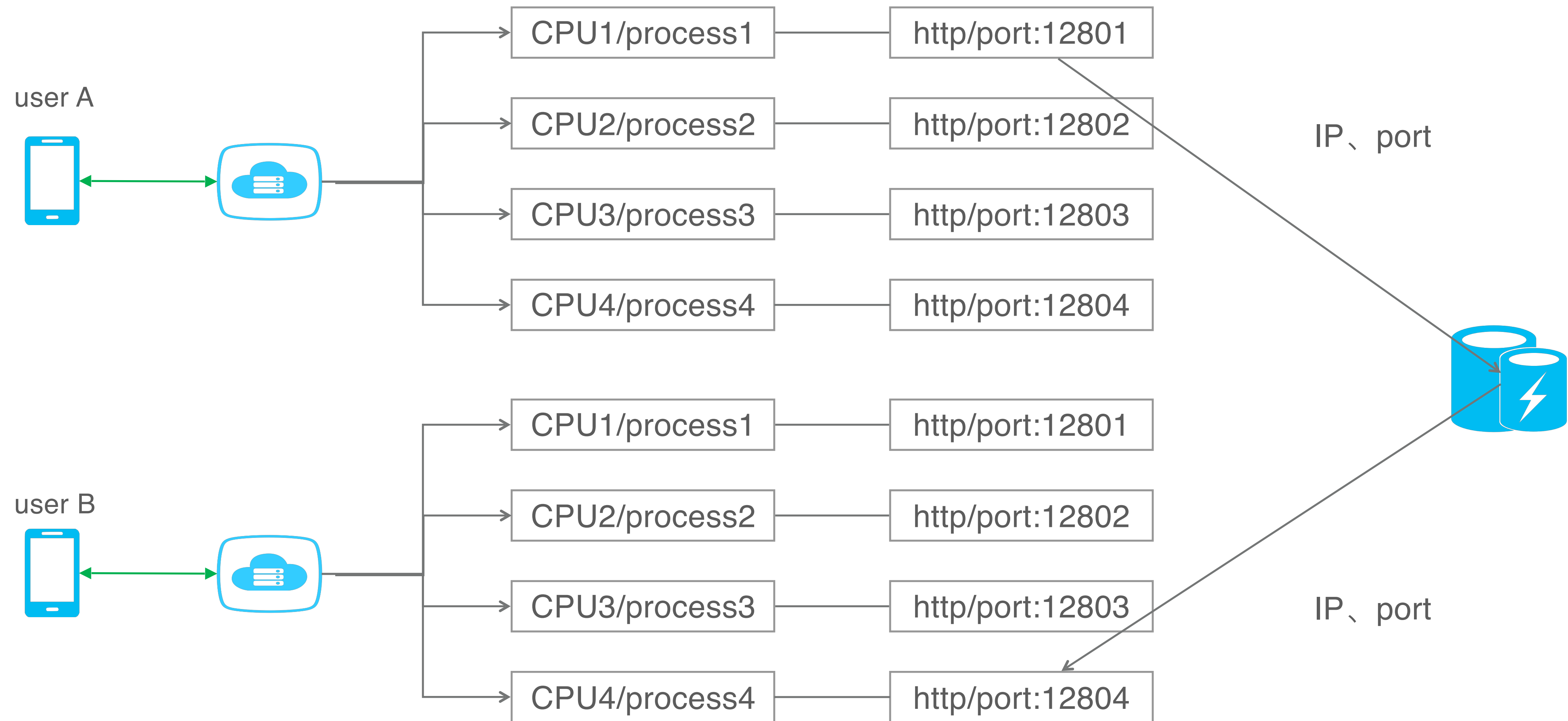
多机多进程通信



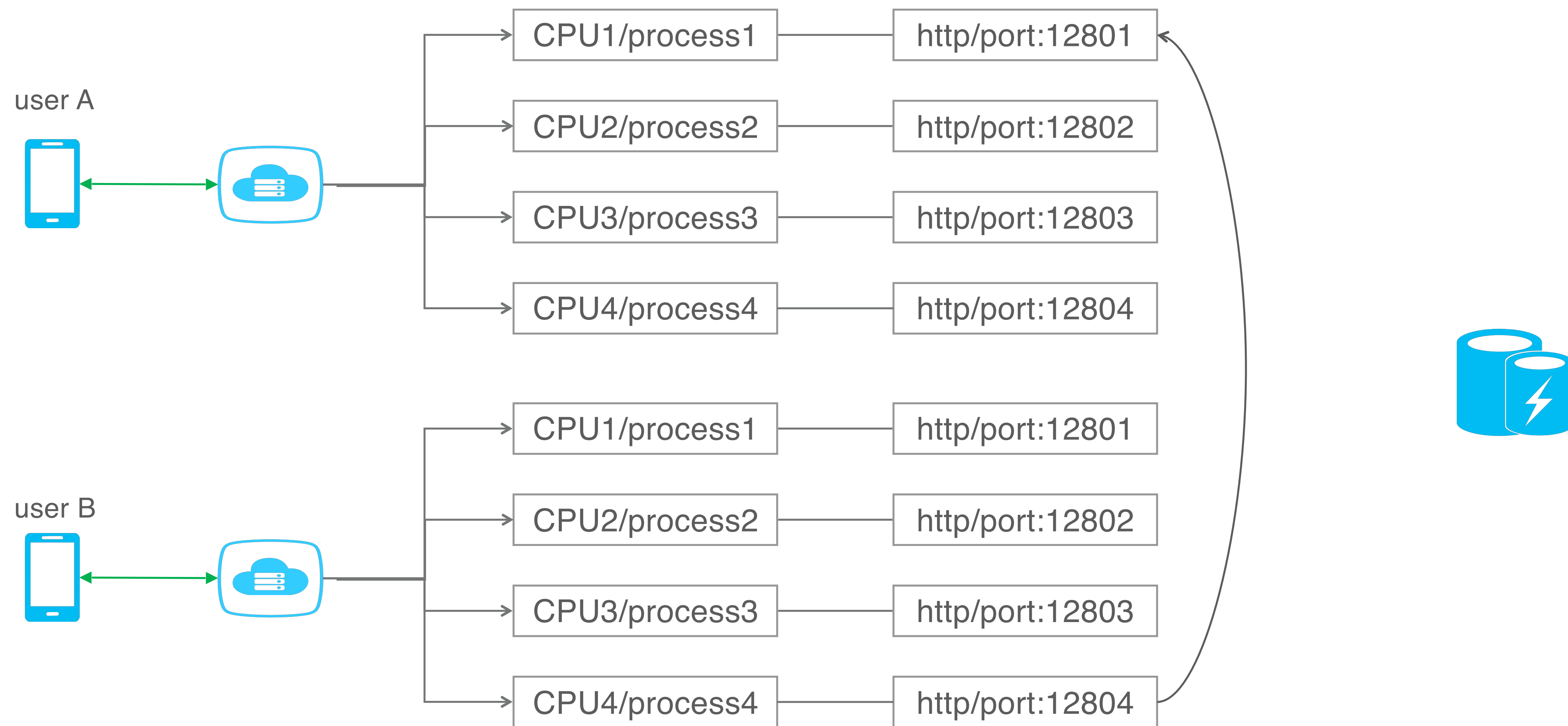
多机多进程通信



多机多进程通信



多机多进程通信



测试环境

测试环境

IP直连

无法带上Cookie、需随时修改IP

测试环境

IP直连

无法带上Cookie、需随时修改IP

测试Host

发布代码需修改Host、多测试环境
机器难维护

测试环境

IP直连

无法带上Cookie、需随时修改IP

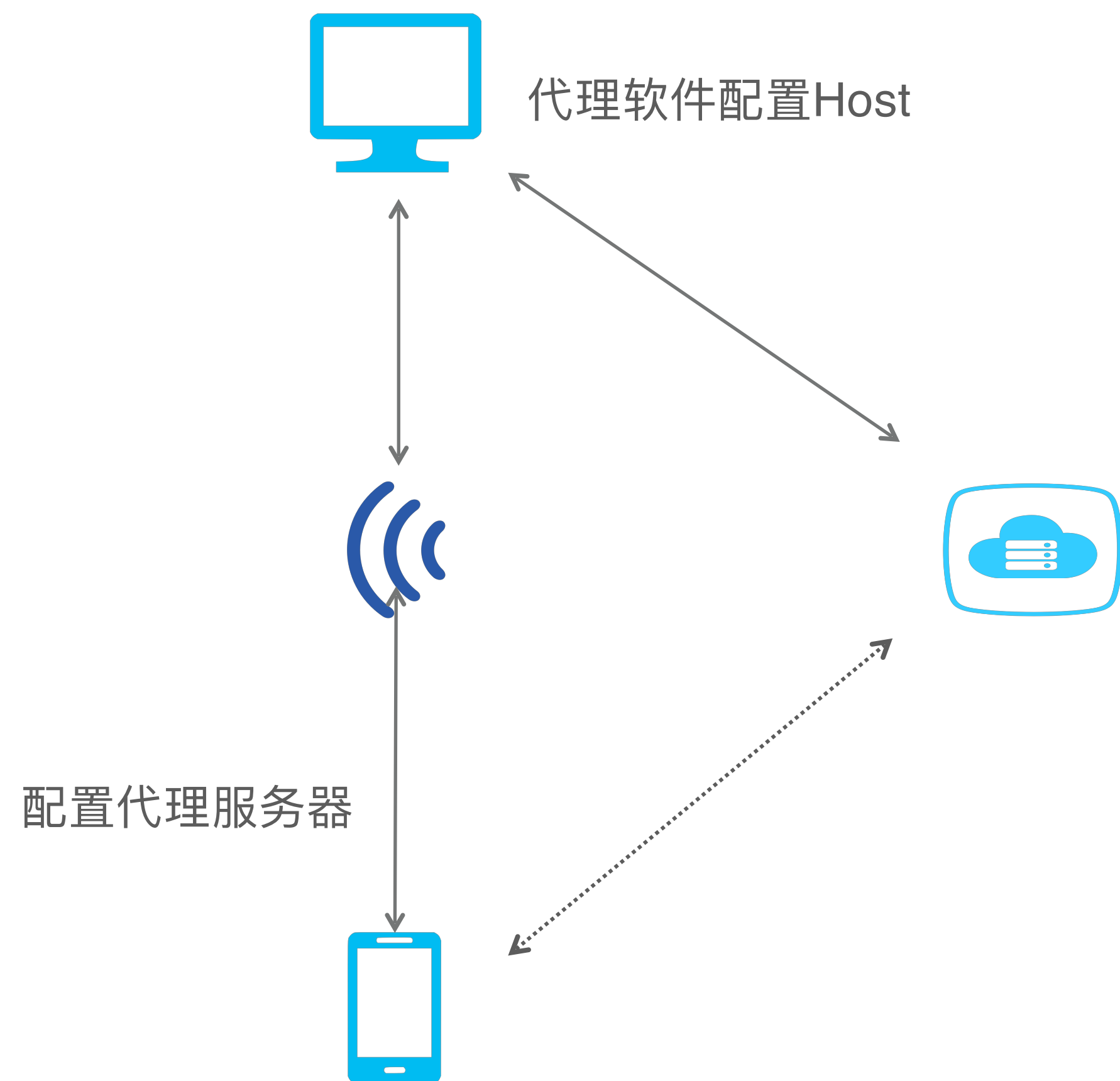
测试Host

发布代码需修改Host、多测试环境
机器难维护

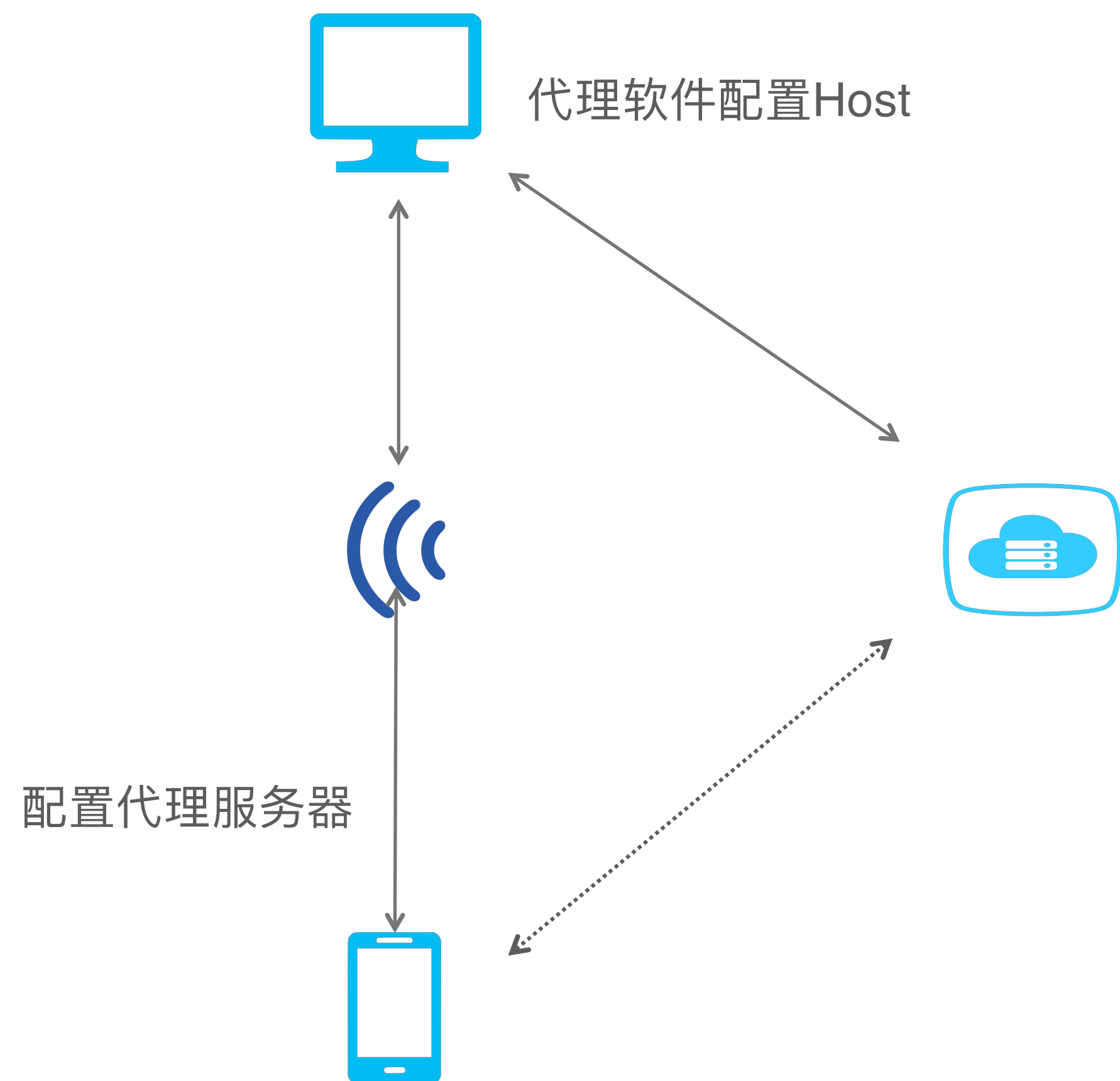
Host代理

对于产品、测试等角色使用门槛高

Host代理

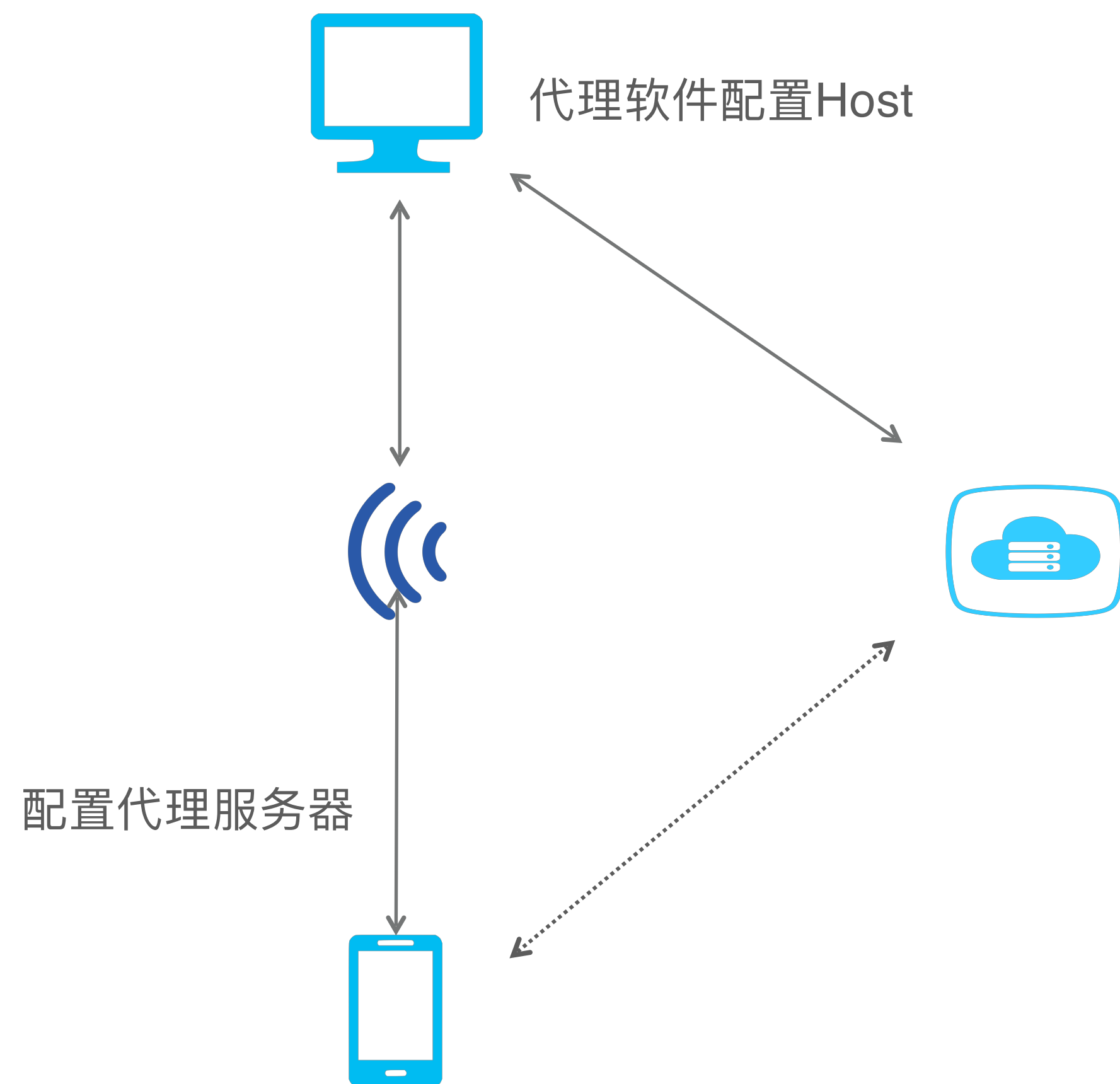


Host代理



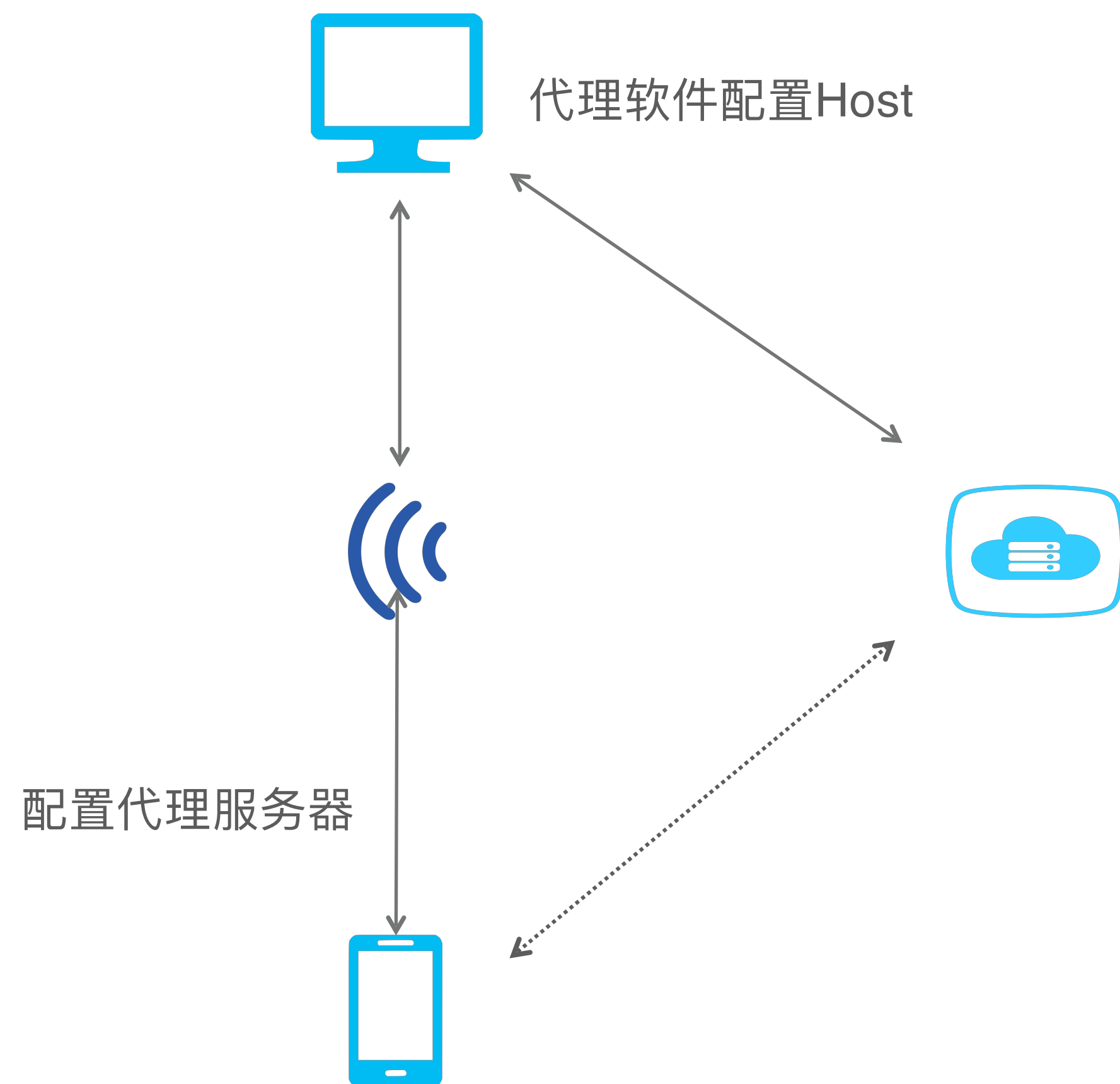
- 0配置难度

Host代理



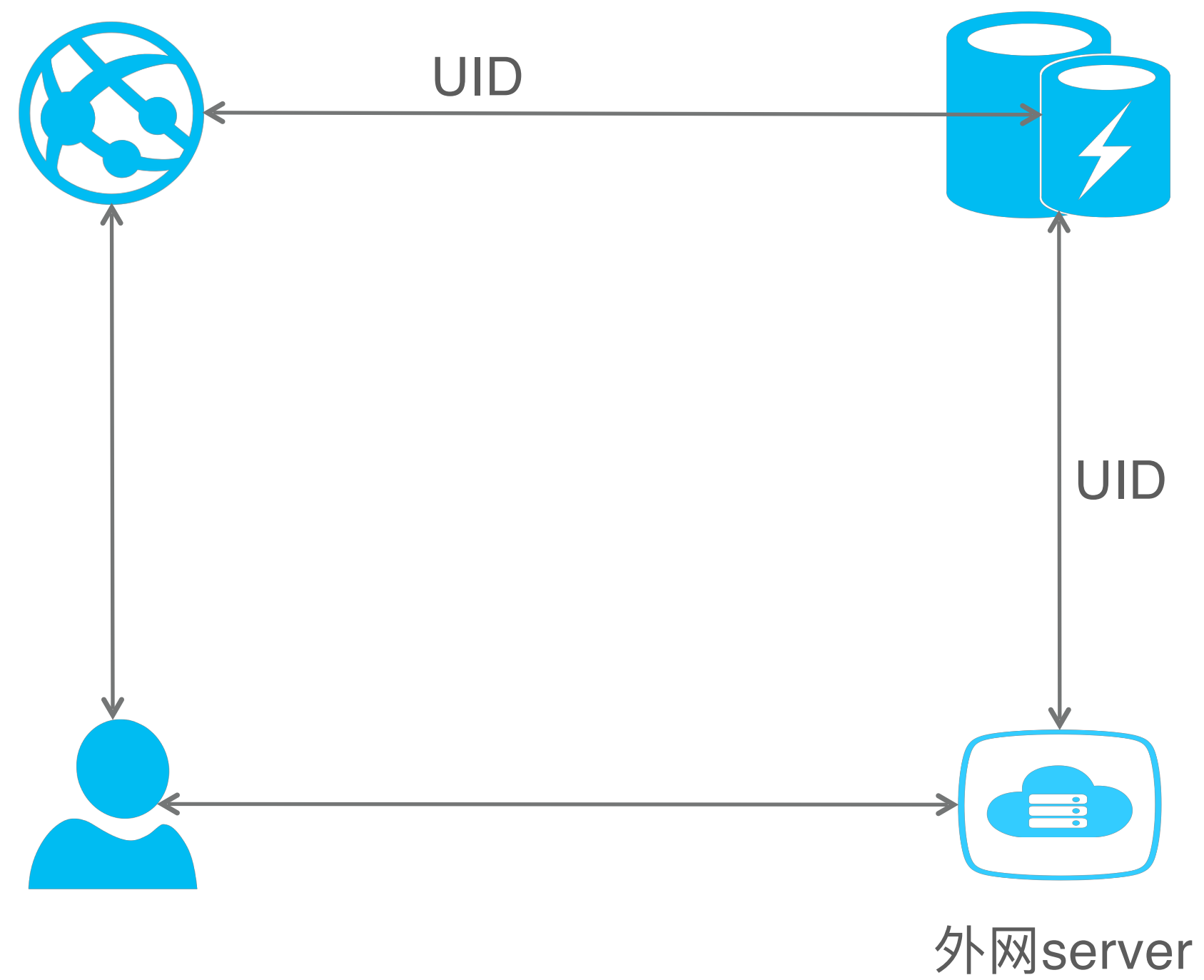
- 0配置难度
- 任意网络环境

Host代理

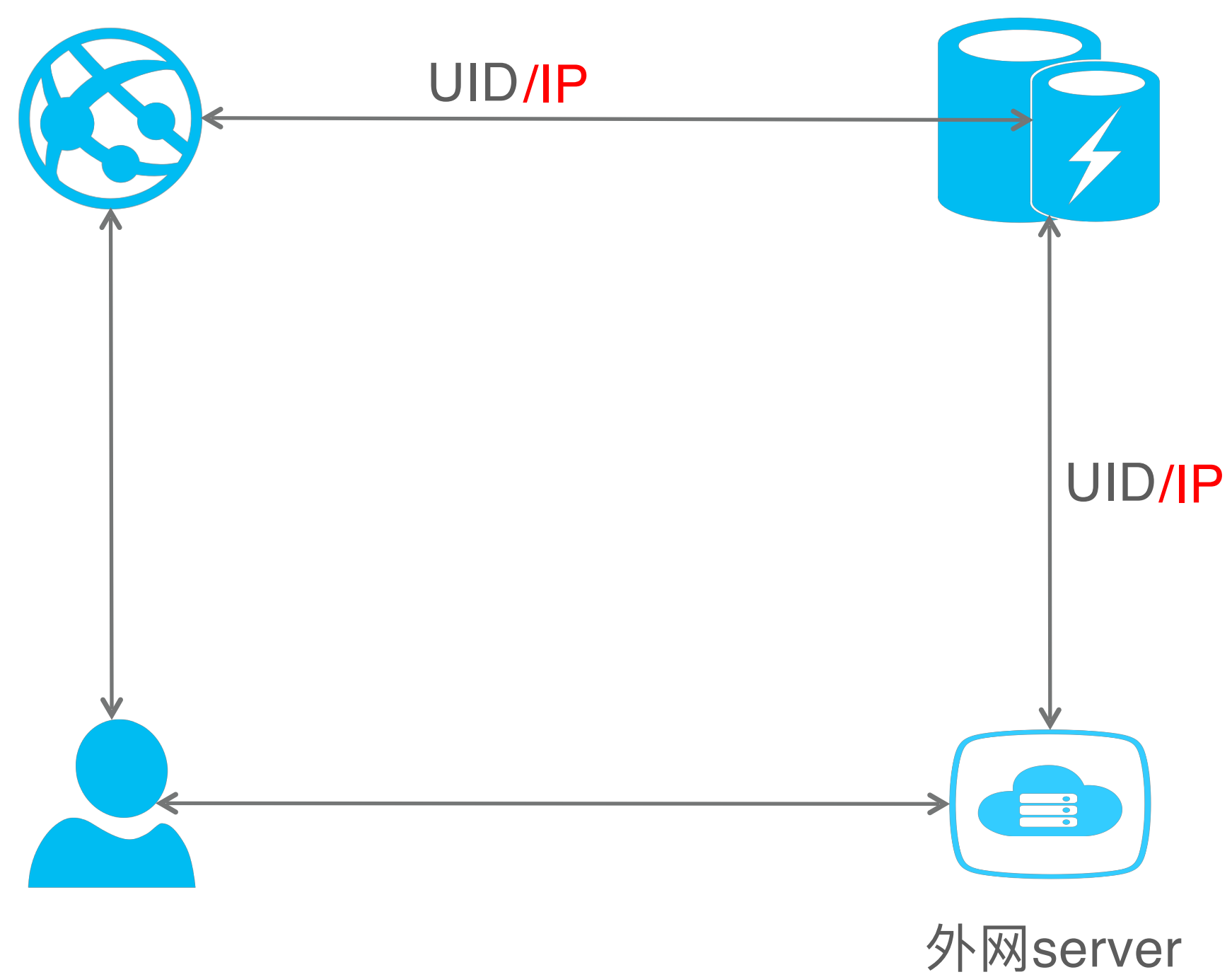


- 0配置难度
- 任意网络环境
- 多测试环境切换

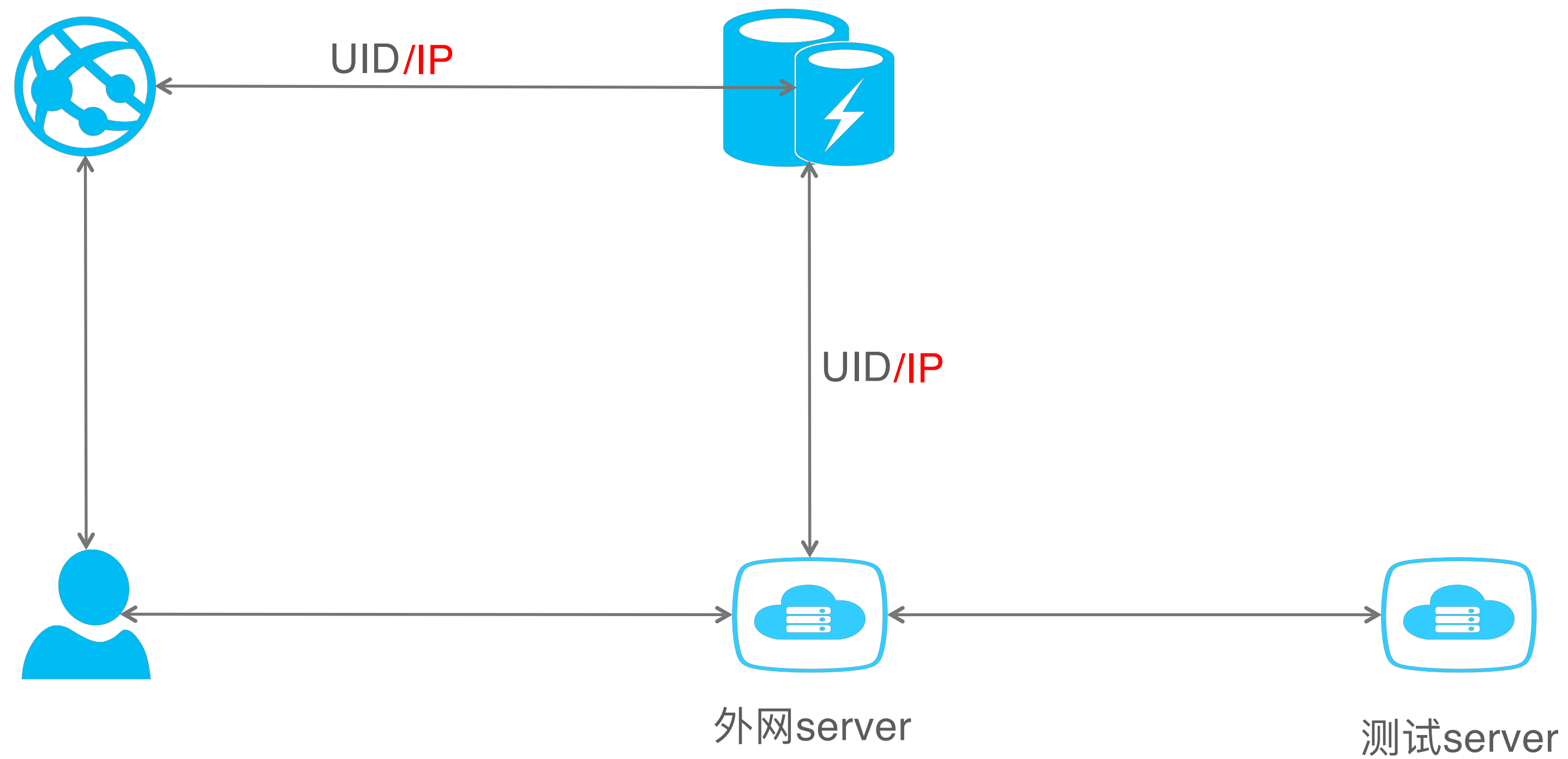
染色



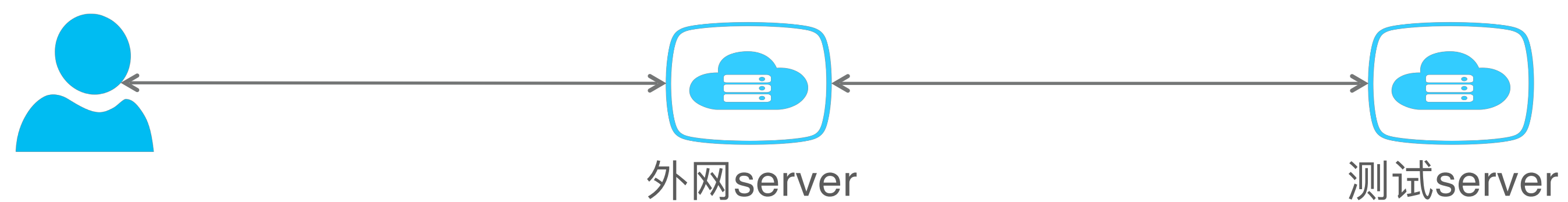
染色



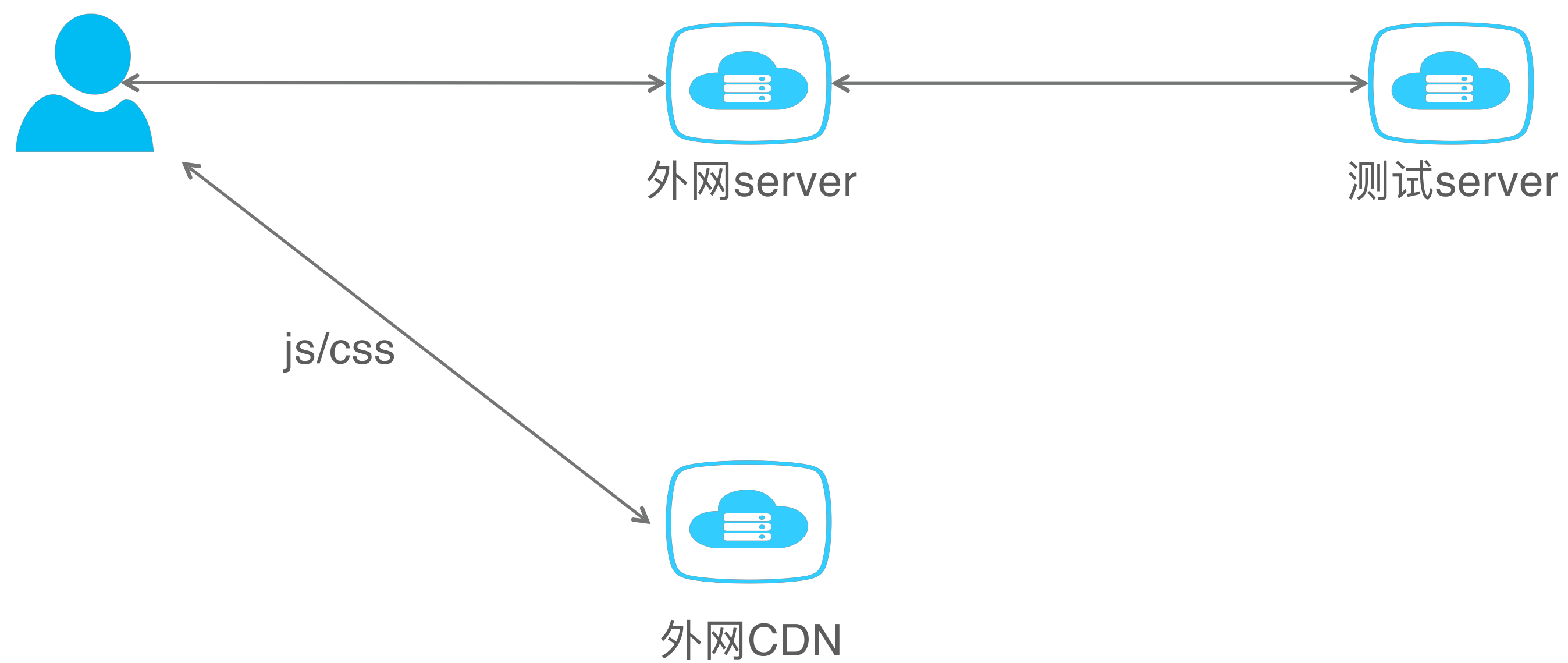
染色



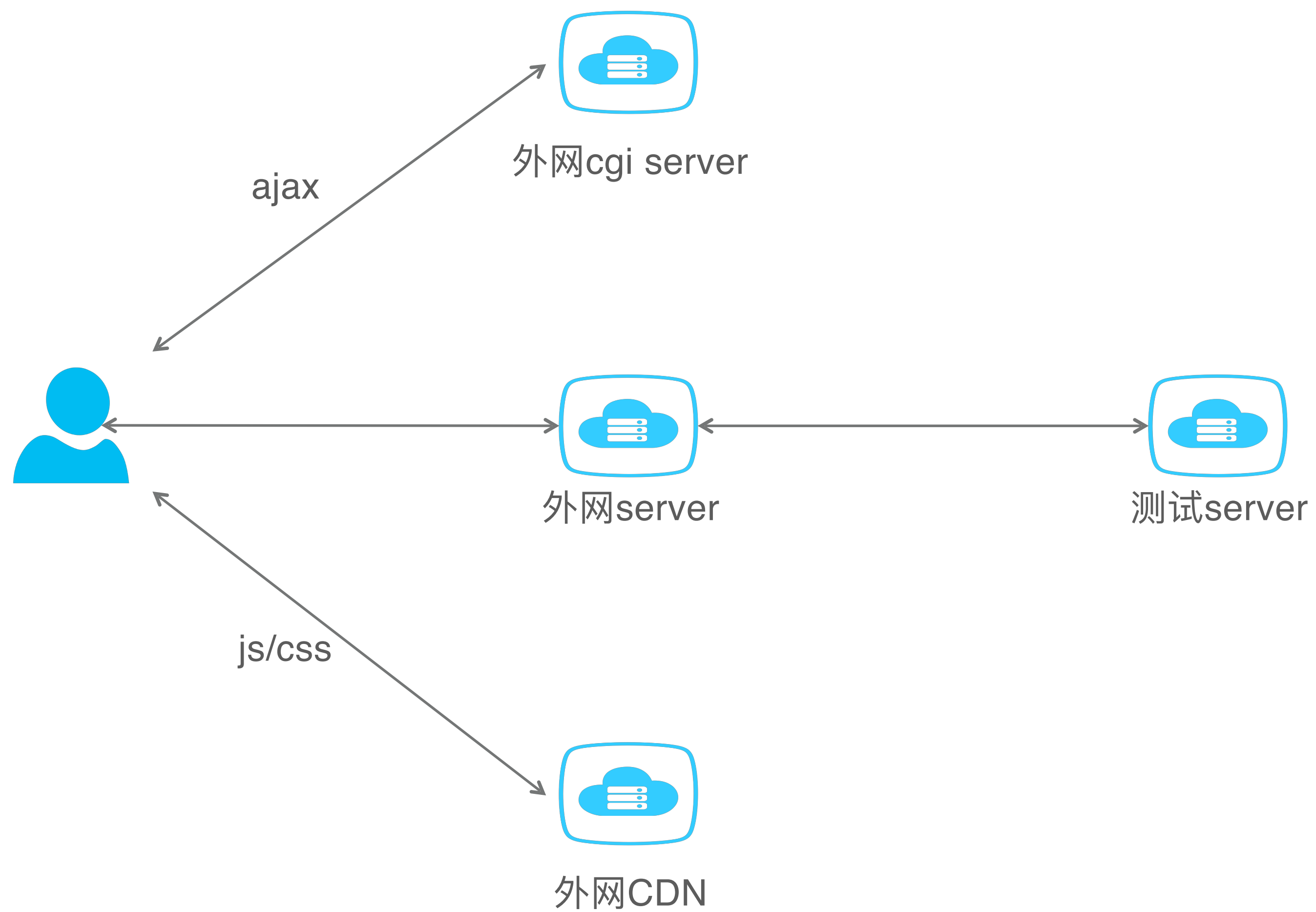
测试环境代理



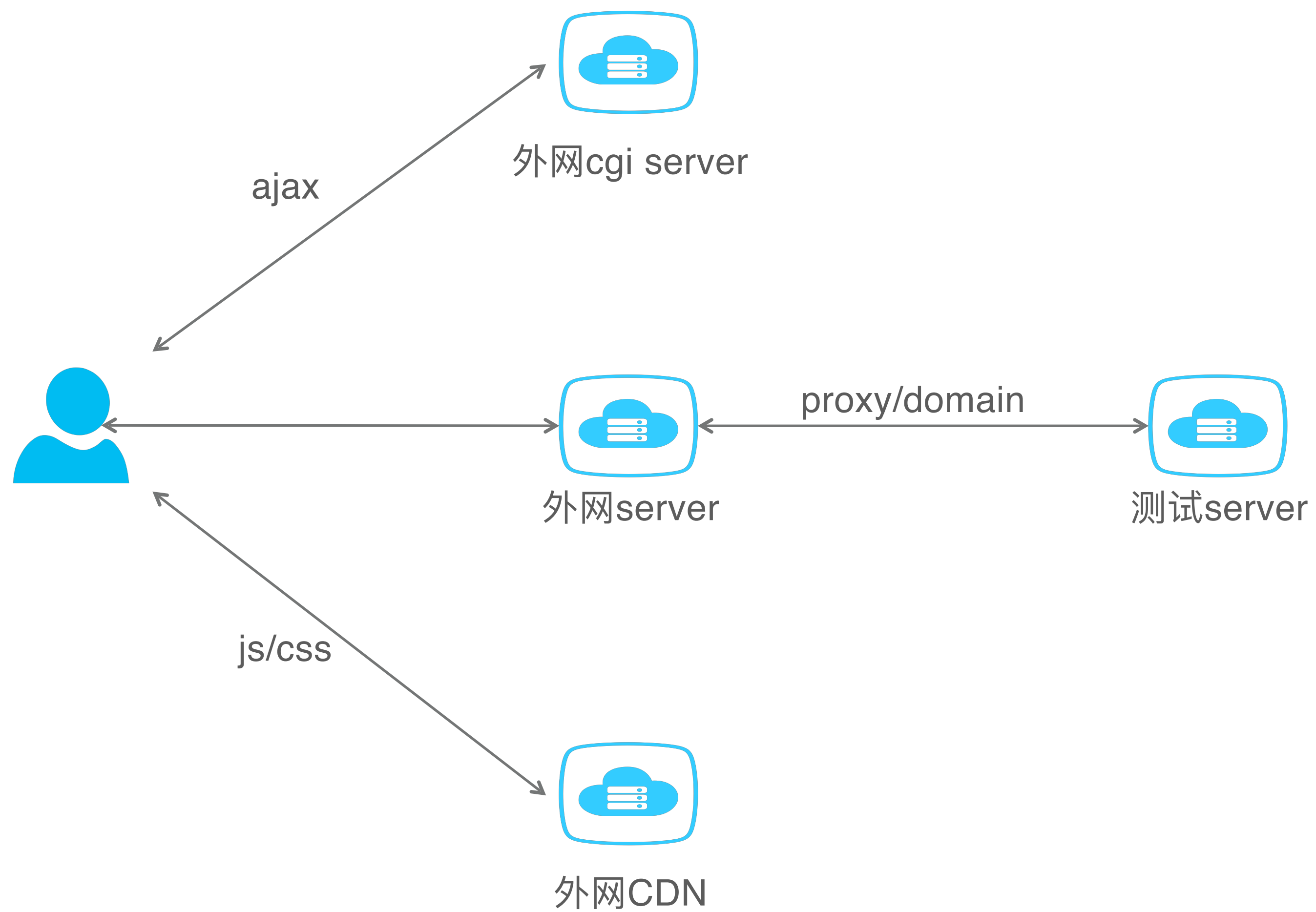
测试环境代理



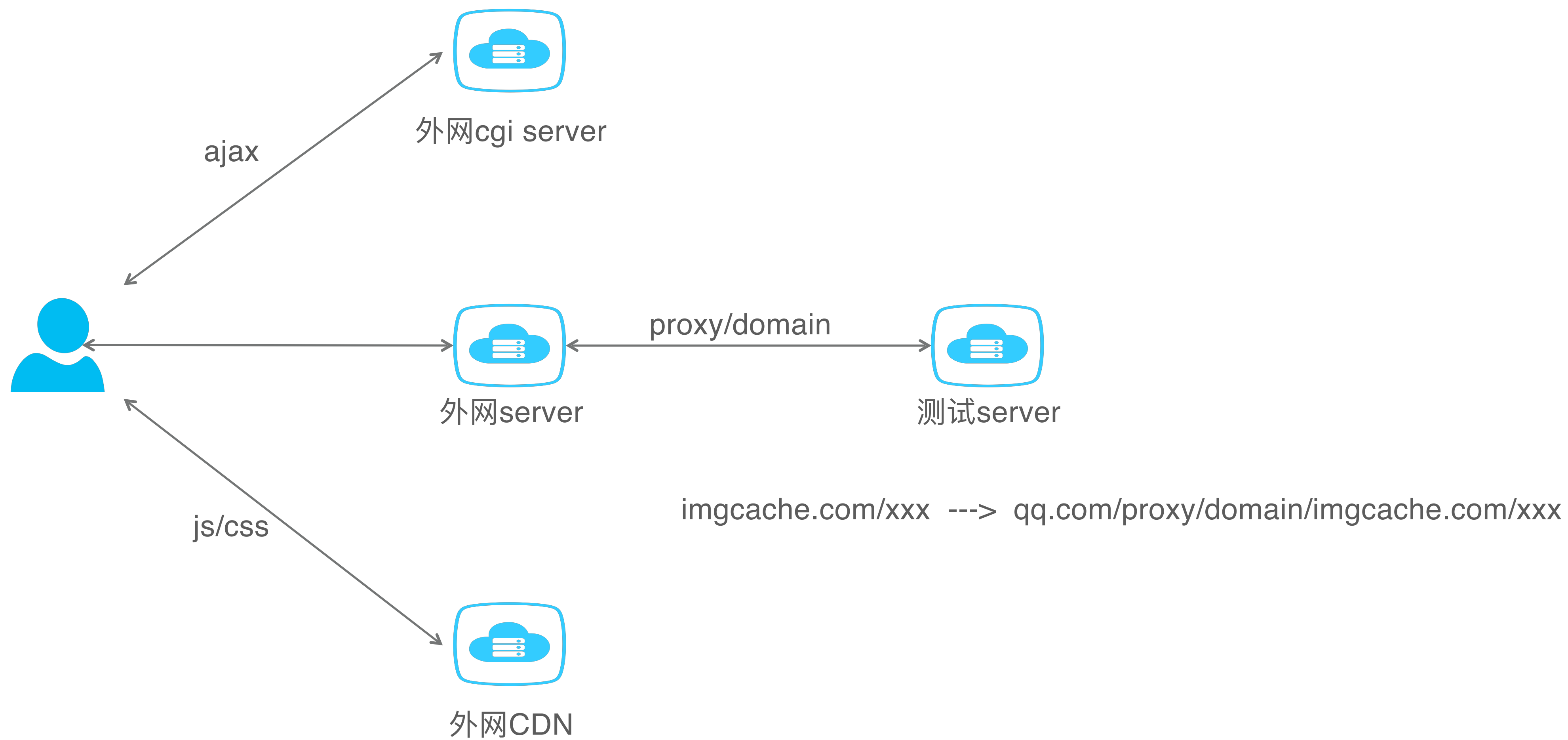
测试环境代理



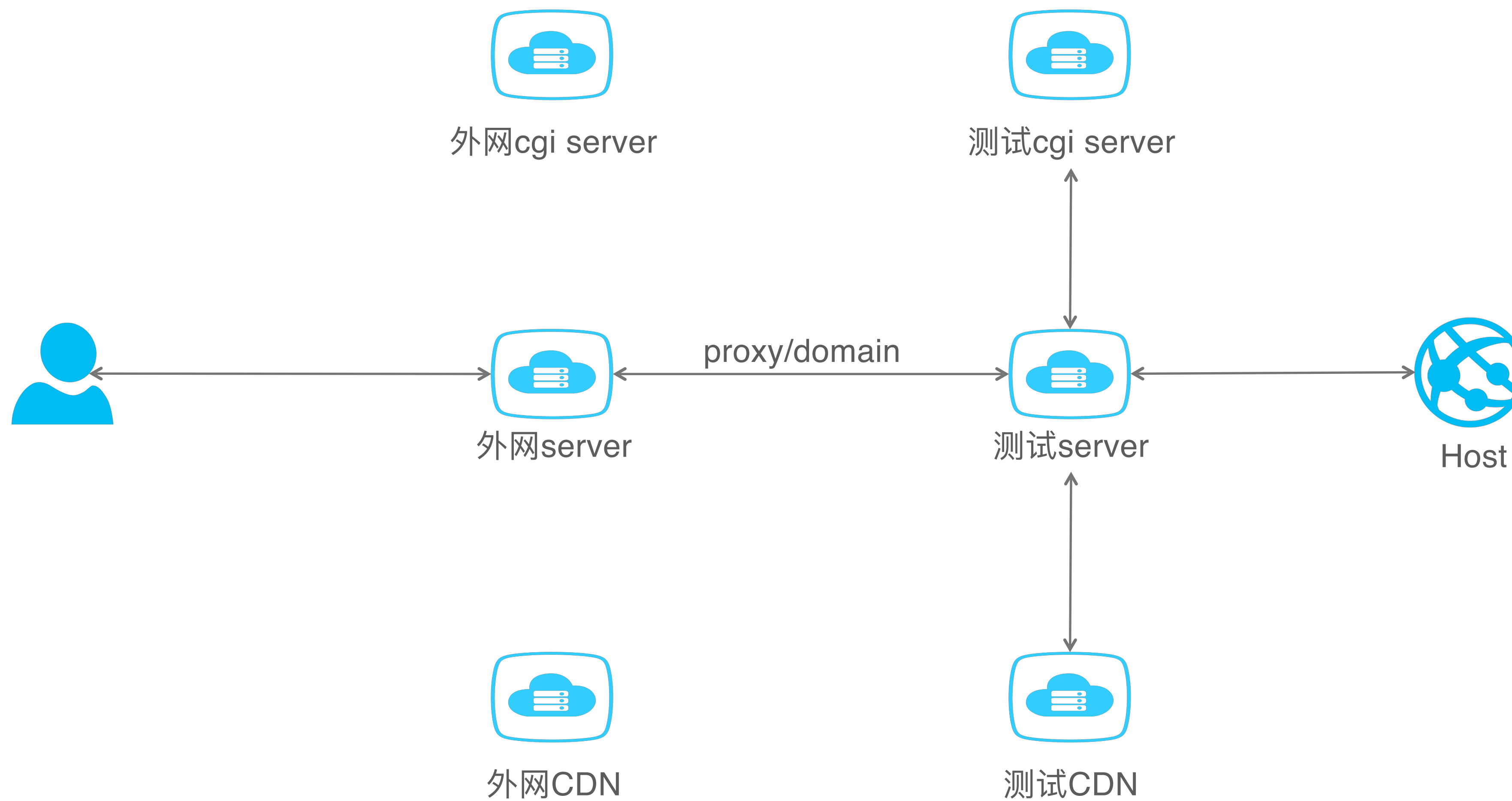
测试环境代理



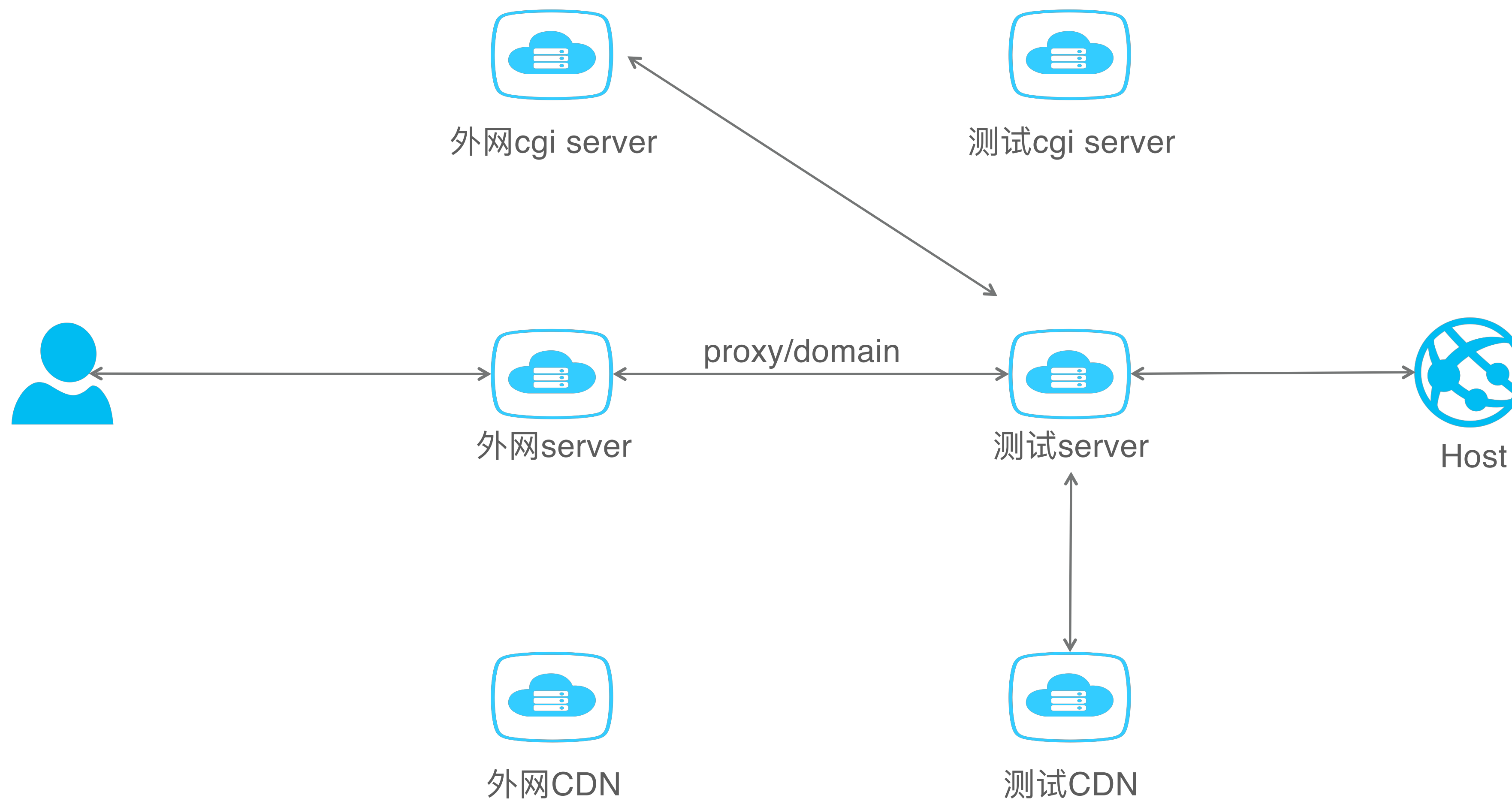
测试环境代理



测试环境代理



测试环境代理



测试环境

测试环境

https://qzone.tswjs.org

#1 选择测试环境

临时染色

alpha

我是2号

10.100

负责人

我是3号

10.100

负责人:

测试环境5号

10.10

负责人:

测试环境8号

10.100

负责人:

测试环境9号

10.100

负责人

我是6号

10.10

负责人

测试环境15号

10.100

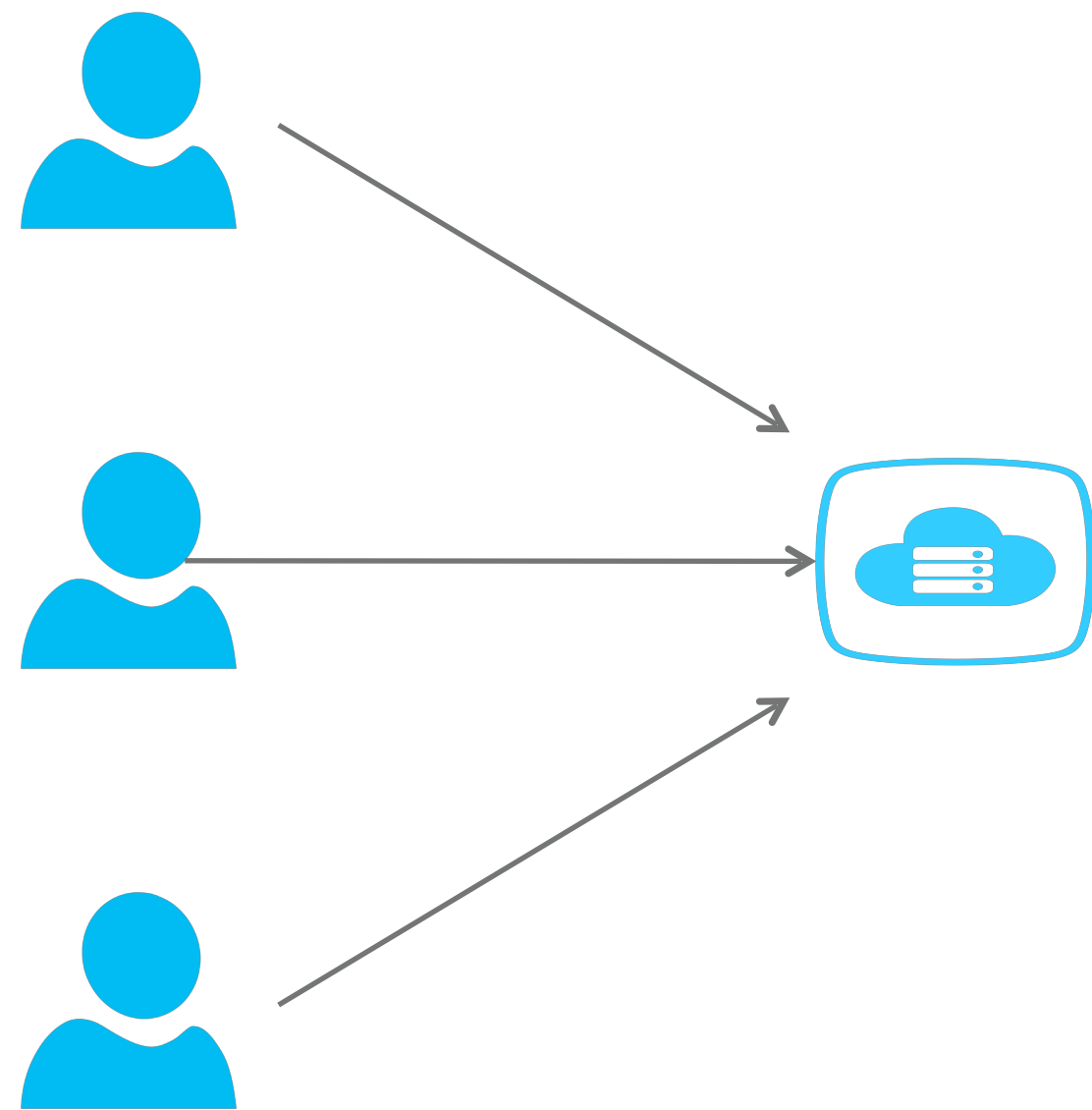
负责人:

#2 输入用户id (支持批量, 换行即可)

添加

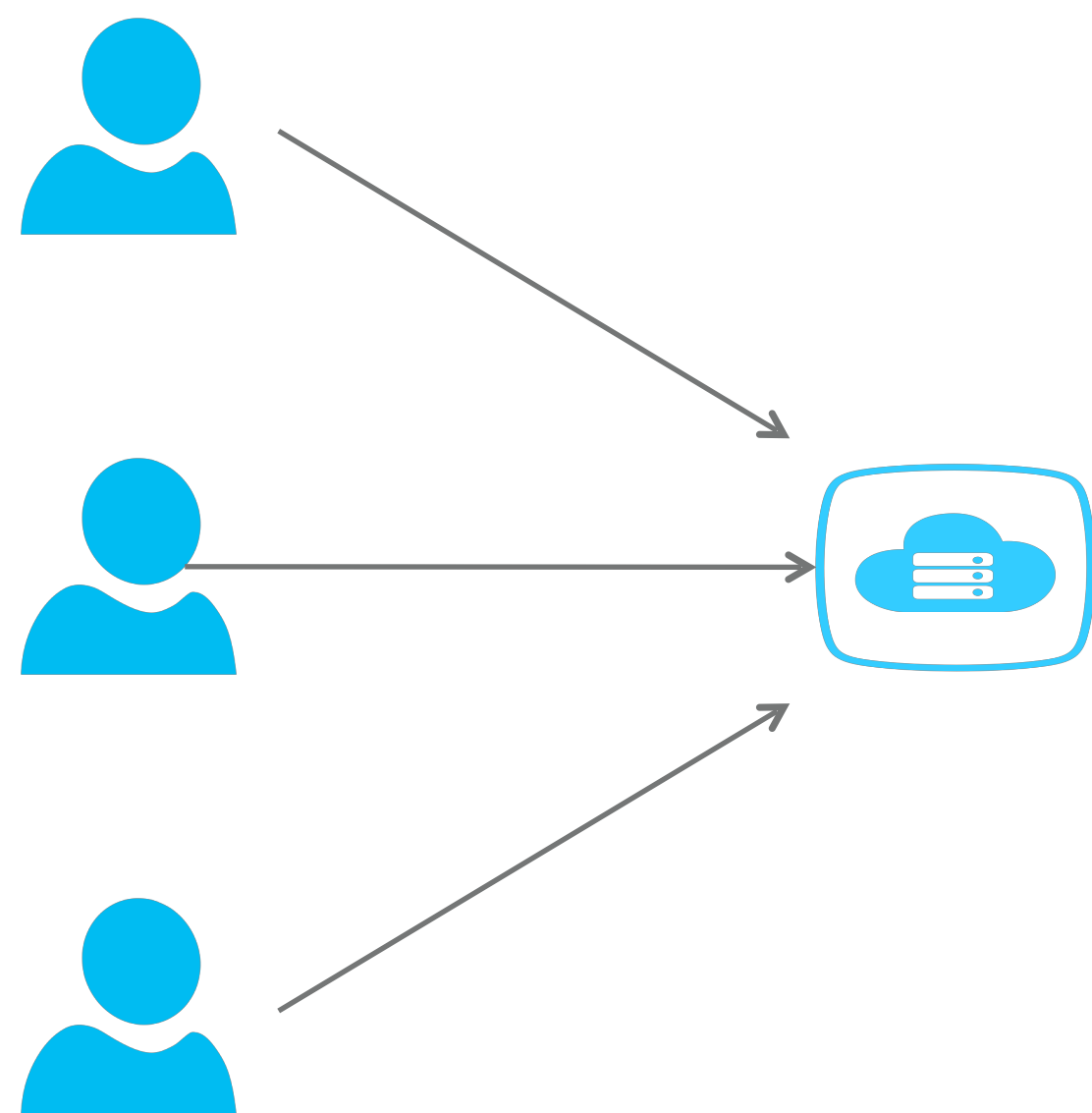
删除

传统日志



```
user1 --- req1 --- start
....
user2 --- req1 --- start
user1 --- req1 --- end
....
user2 --- req1 --- end
user3 --- req1 --- start
....
user2 --- req2 --- start
user3 --- req1 --- end
....
user2 --- req2 --- end
```

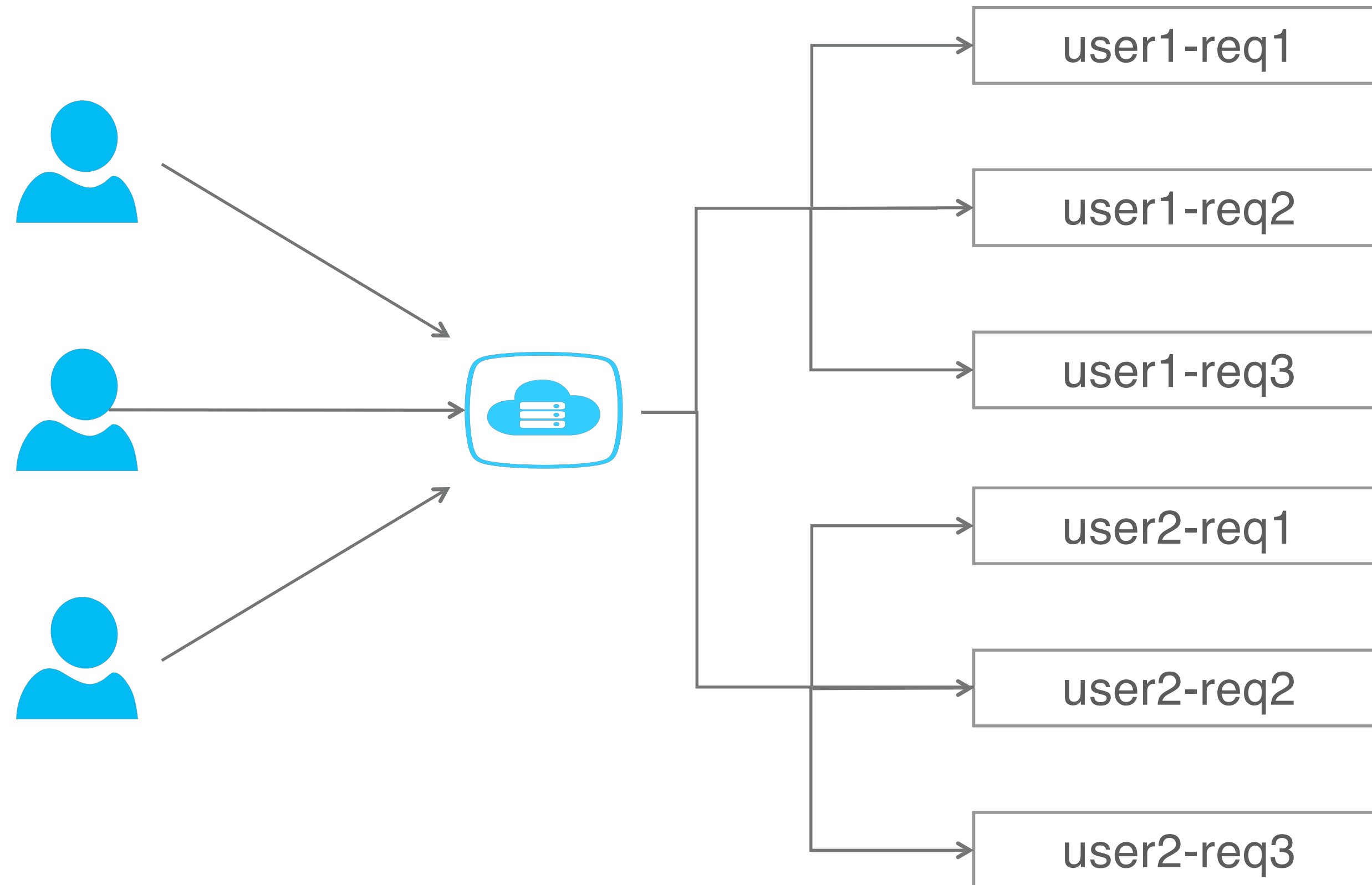
传统日志



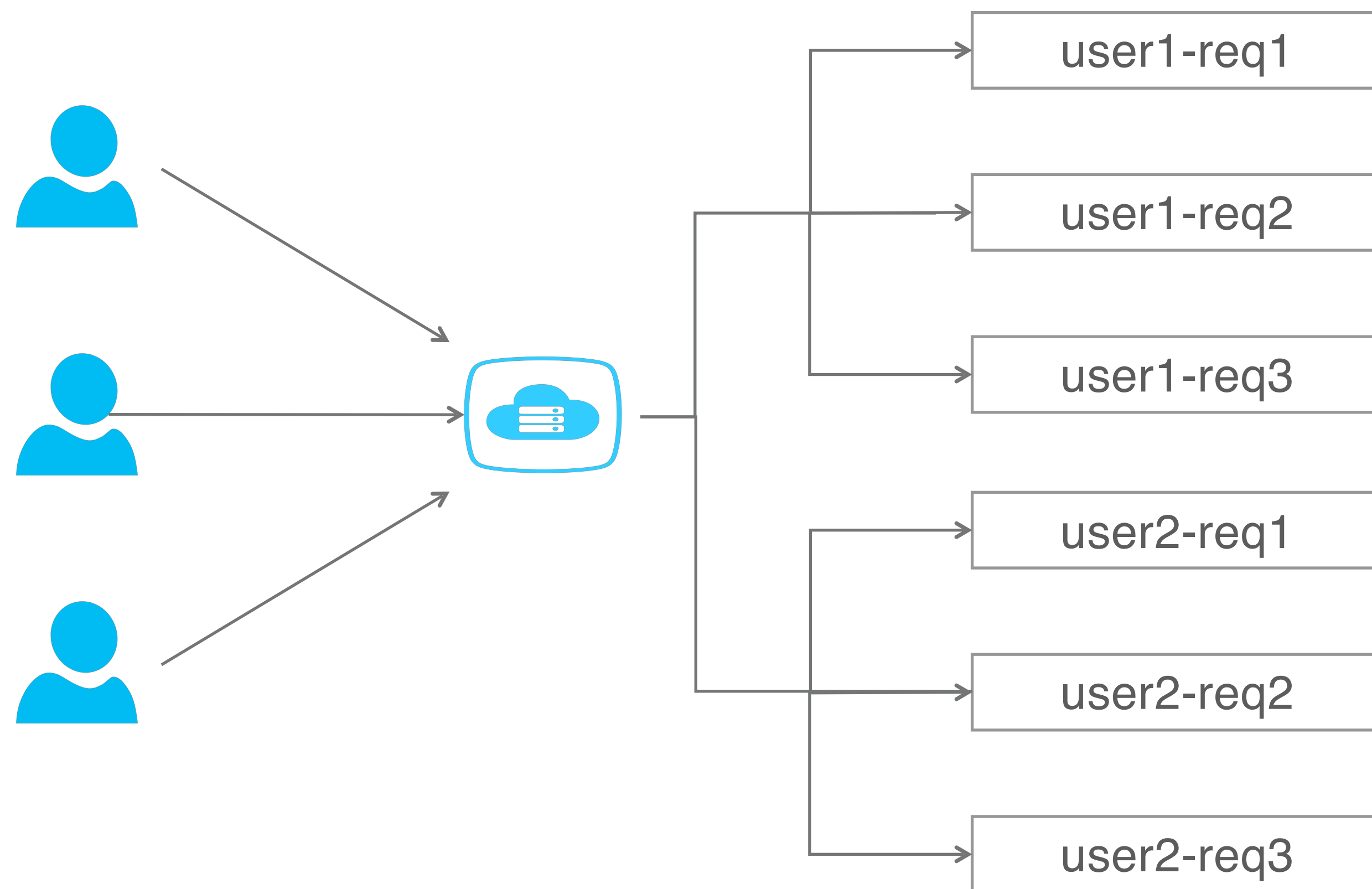
```
user1 --- req1 --- start
....
user2 --- req1 --- start
user1 --- req1 --- end
....
user2 --- req1 --- end
user3 --- req1 --- start
....
user2 --- req2 --- start
user3 --- req1 --- end
....
user2 --- req2 --- end
```

- 按API调用顺序，以单行日志为维度，逐条记录
- 不同请求，不同维度代码穿插执行，看一行日志无法感知上下文
- 上机器查日志，抓包一小时，代码一分钟

日志聚合



日志聚合



- 按请求维度聚合
- 按用户维度聚合
- 实时查询
- 关联衍生请求

```
callback() {  
  .....  
  const handleRequest = (req, res) => {  
    const ctx = this.createContext(req, res);  
    return this.handleRequest(ctx, fn);  
  };  
  .....  
}
```

```
ctx.logger.debug('debug info');
```

```
domain.create().run(function() {  
    ...  
    doRoute();  
});  
  
Object.defineProperty(global, 'logger', {  
    get: function () {  
        return process.domain.logger  
    }  
});
```

日志聚合

XXXXXX

```
setTimeout(function(){
```

```
    //do something
```

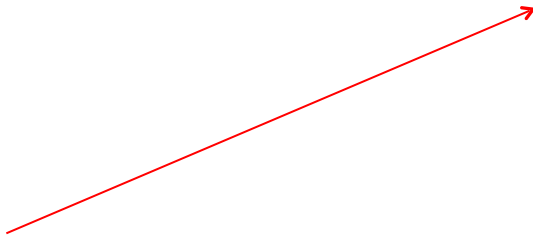
```
}, 2000)
```

XXXXXX

日志聚合

```
XXXXXX  
  
setTimeout(function(){  
  
    //do something  
  
}, 2000)  
  
XXXXXX
```

init

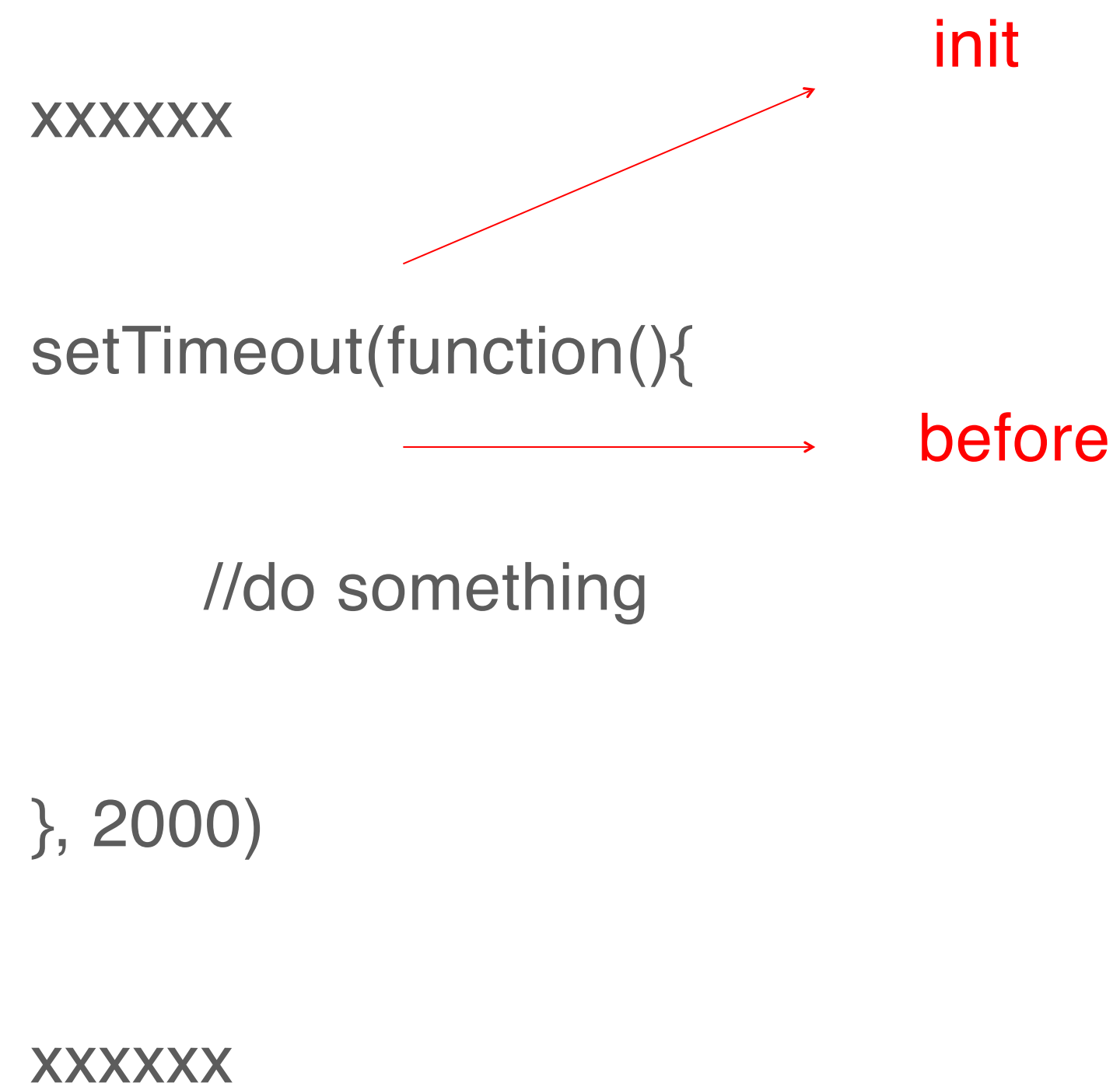


日志聚合

```
XXXXXX  
  
setTimeout(function(){  
    //do something  
  
}, 2000)  
  
XXXXXX
```

init

before



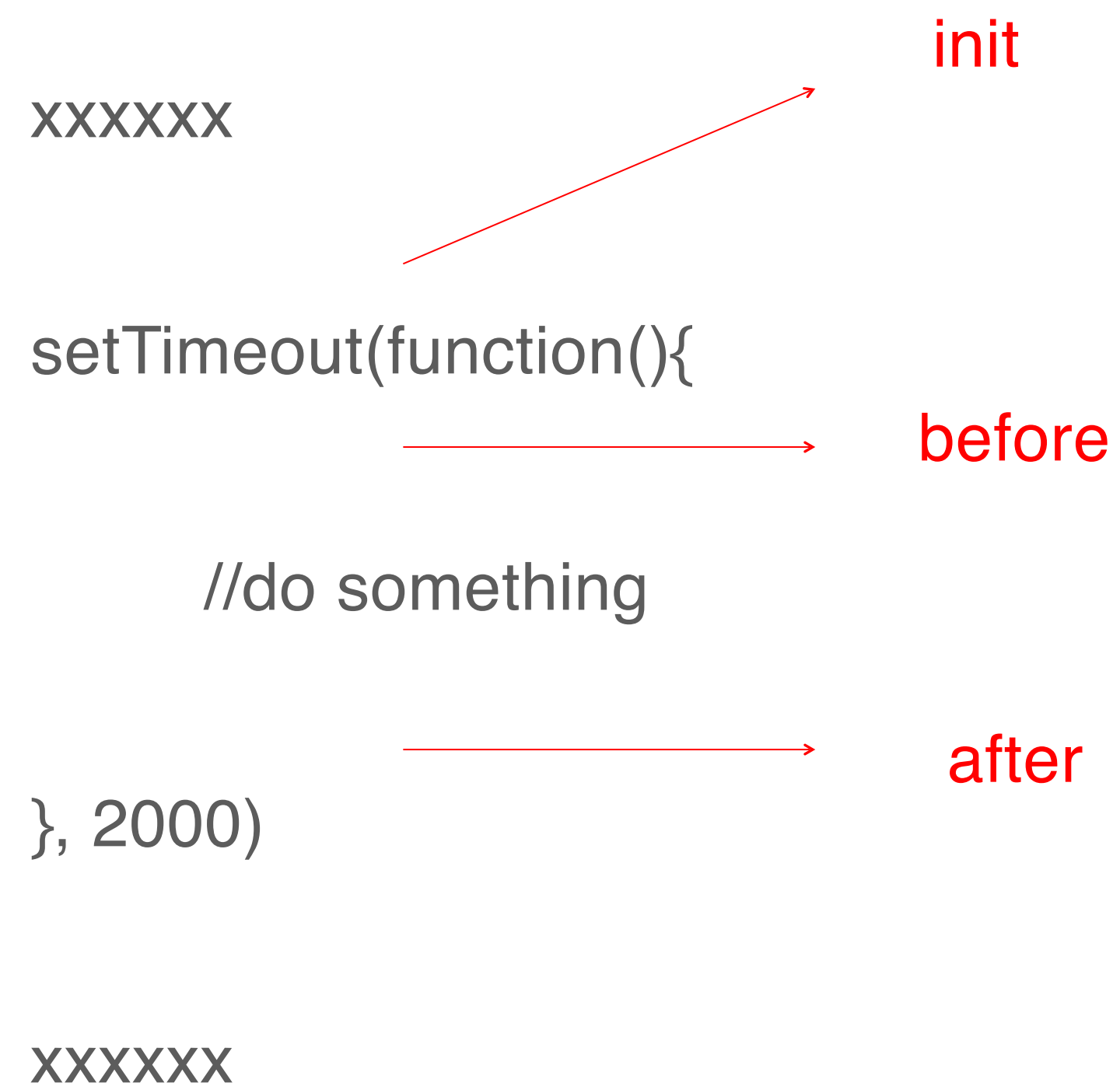
The diagram illustrates log aggregation for a JavaScript code snippet. The code is as follows:

```
XXXXXX  
  
setTimeout(function(){  
    //do something  
  
}, 2000)  
  
XXXXXX
```

Annotations and arrows:

- A red arrow points from the text **init** to the first line of the code block (`XXXXXX`).
- A red arrow points from the text **before** to the opening curly brace of the `setTimeout` function (`{`).

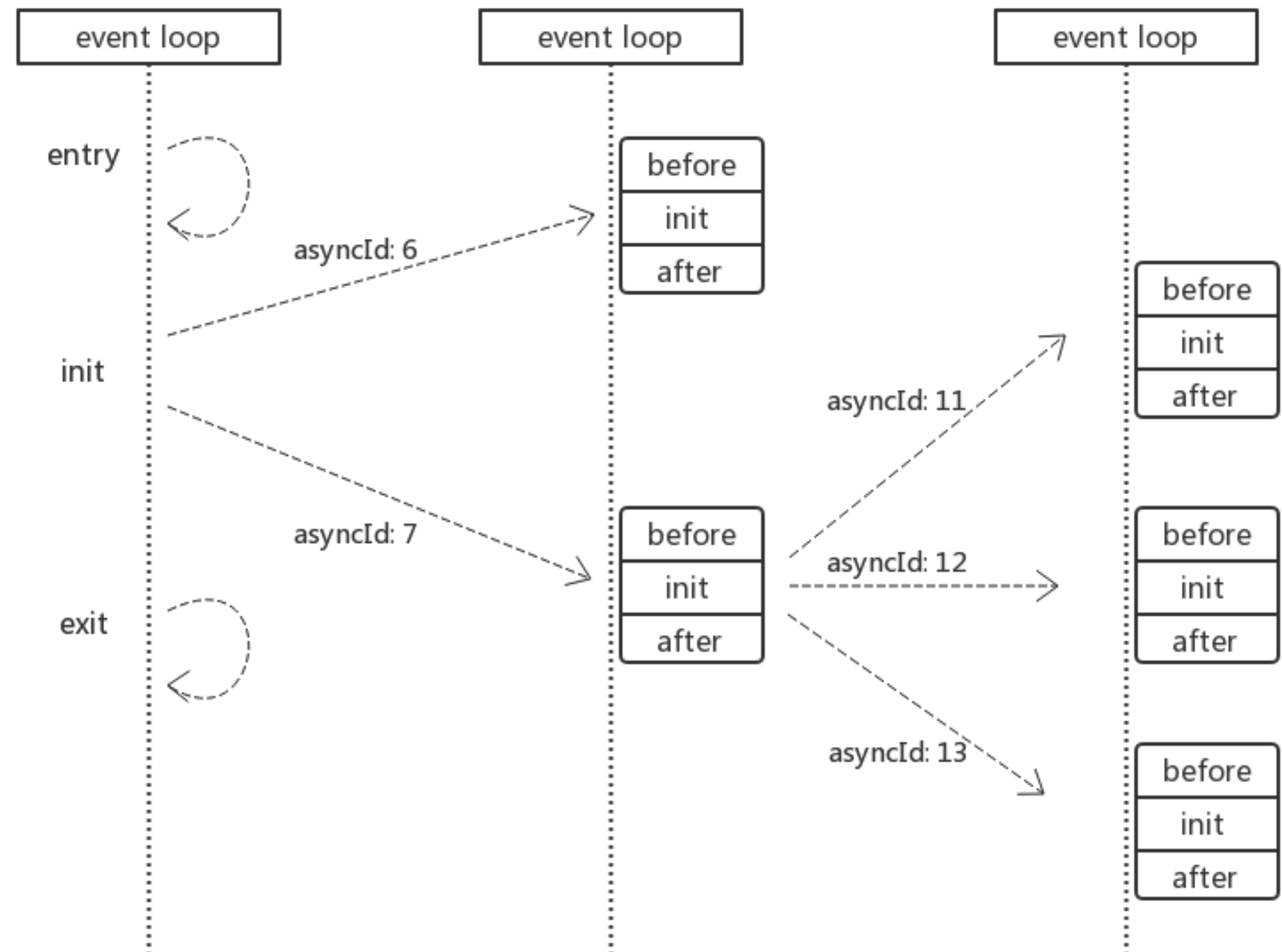
日志聚合



日志聚合

```
XXXXXX  
setTimeout(function(){  
  //do something  
}, 2000)  
XXXXXX
```

init
before
after



■ 日志上报

HTTP

➤ 生命周期短，请求完成即上报

■ 日志上报

HTTP

- 生命周期短，请求完成即上报

WebSocket

- 生命周期长，日志分片上报、定时上报

■ 日志上报

HTTP

- 生命周期短，请求完成即上报

衍生请求

- 支持fiddler、charles、whistle等查看重放

WebSocket

- 生命周期长，日志分片上报、定时上报

■ 日志上报

HTTP

- 生命周期短，请求完成即上报

WebSocket

- 生命周期长，日志分片上报、定时上报

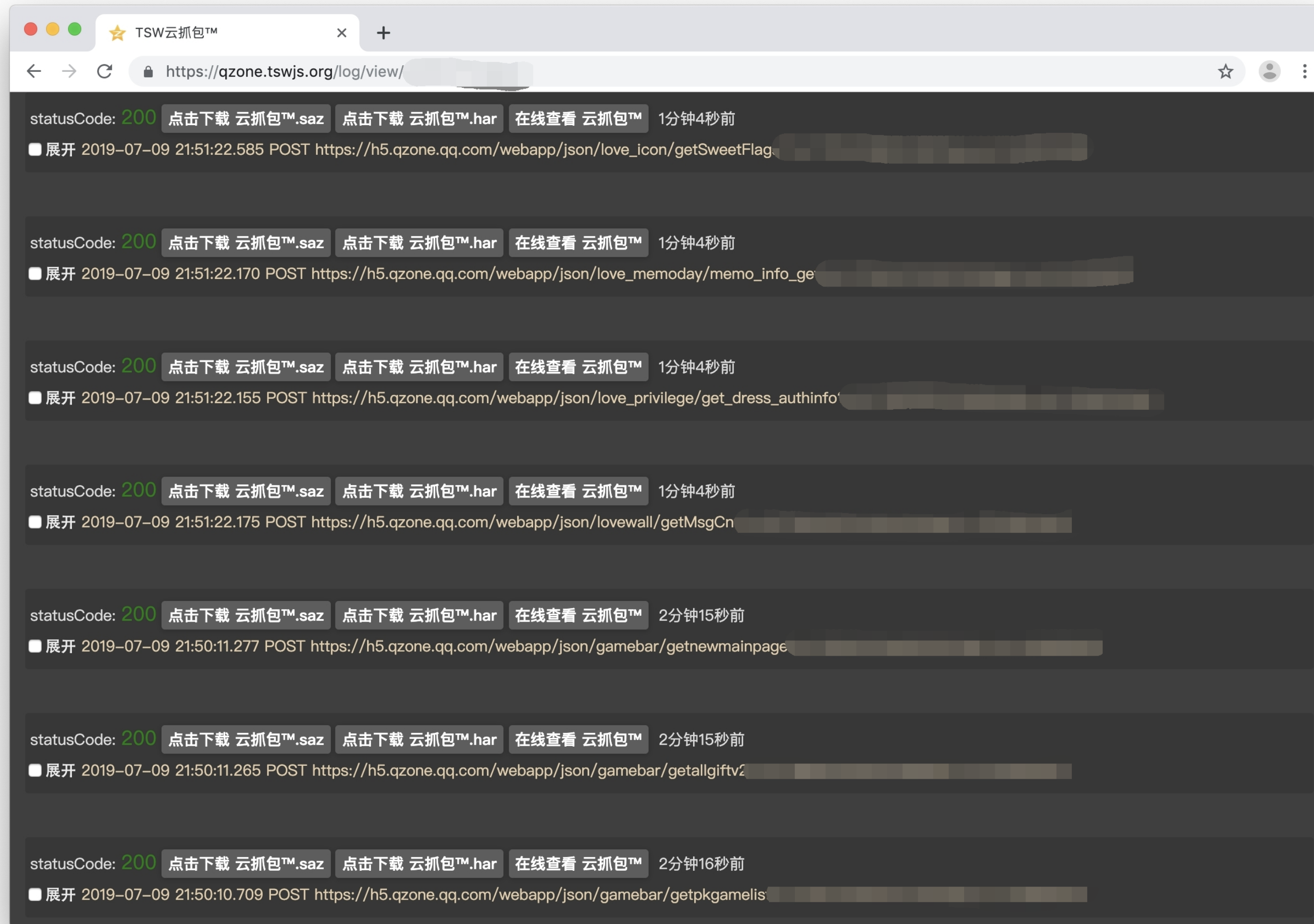
衍生请求

- 支持fiddler、charles、whistle等查看重放

错误日志

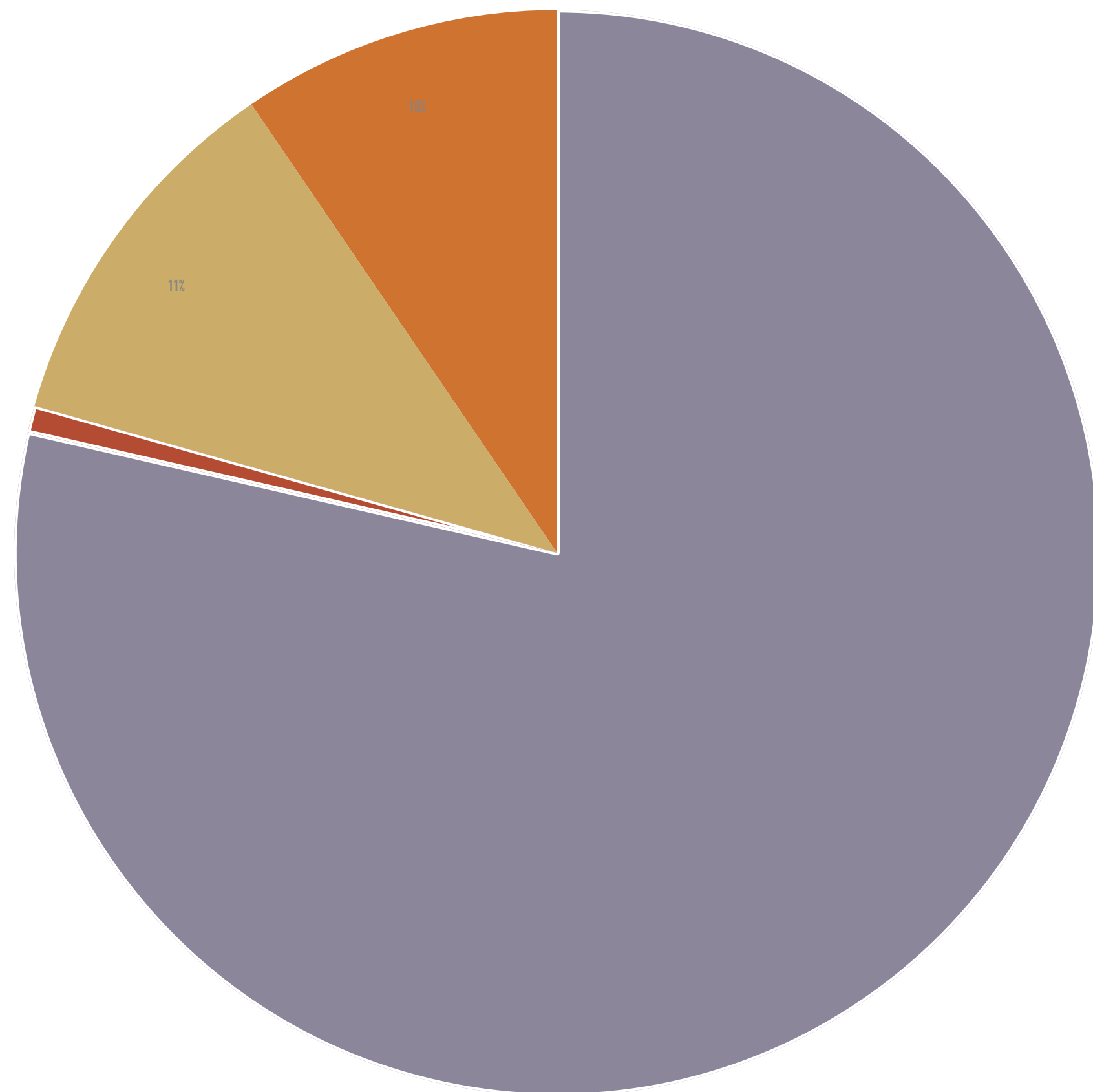
- 邮件告警
- 全量上报，存储一定时间

日志上报



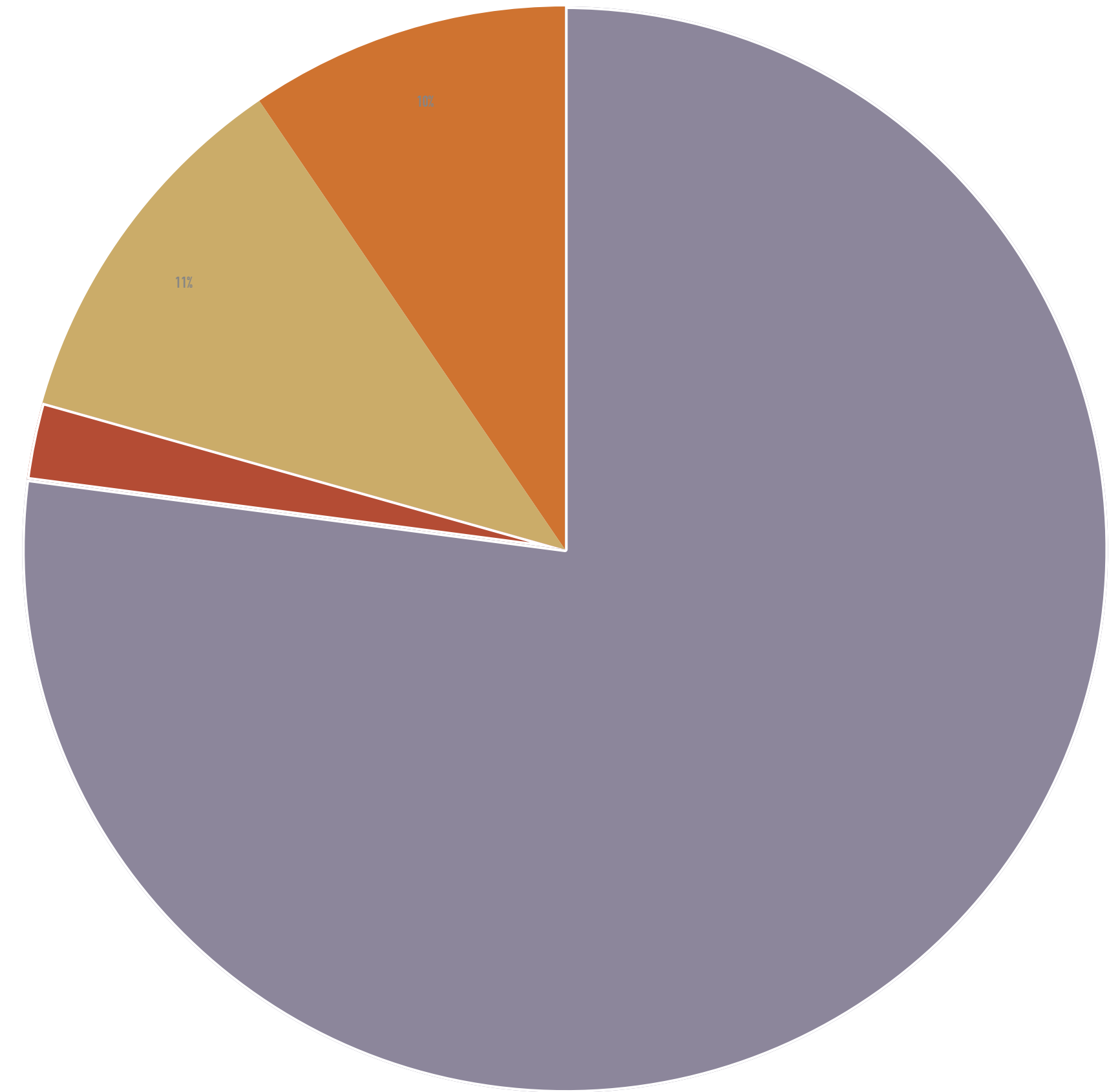
支持率

移动端

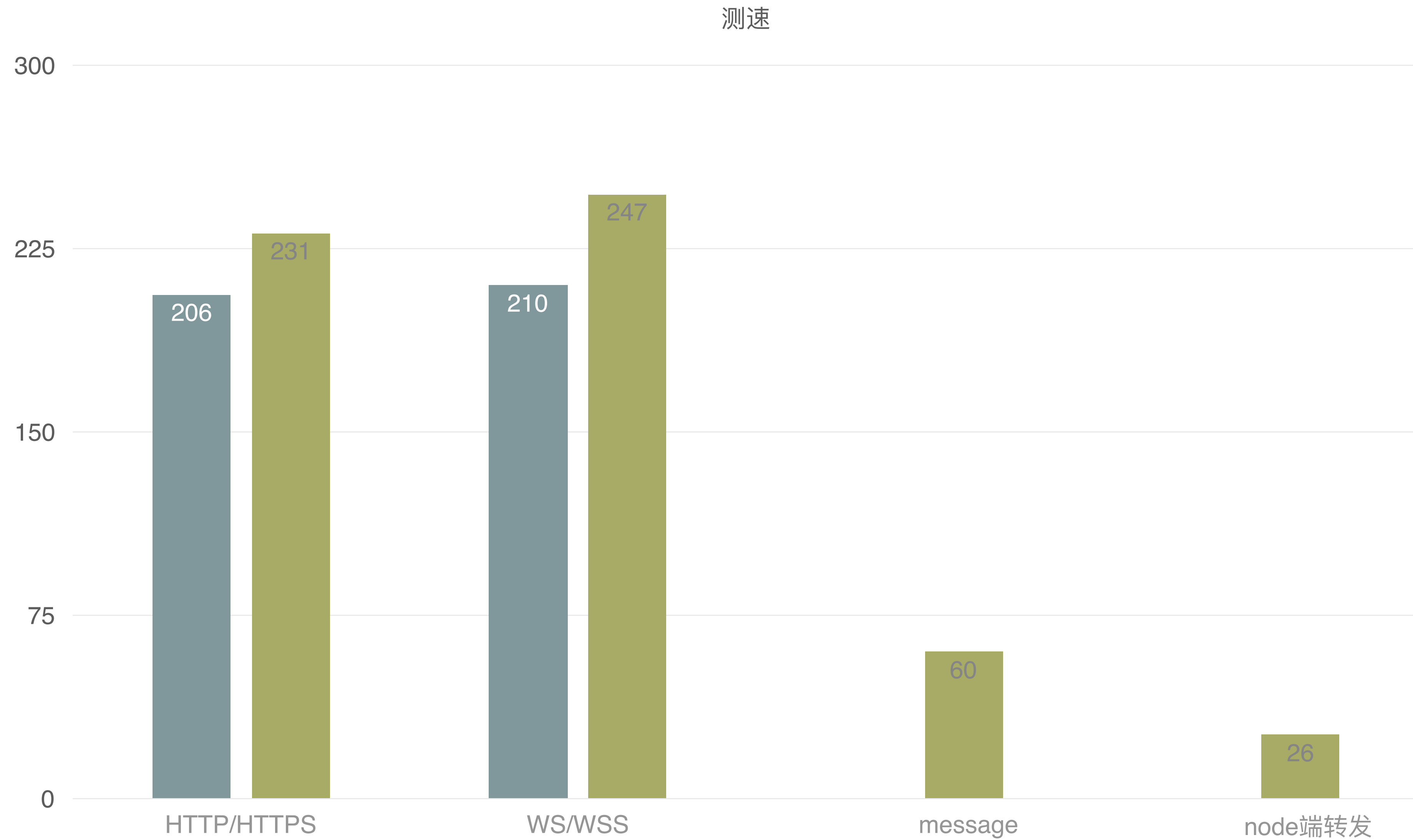


● 支持 ● 不支持 ● ●

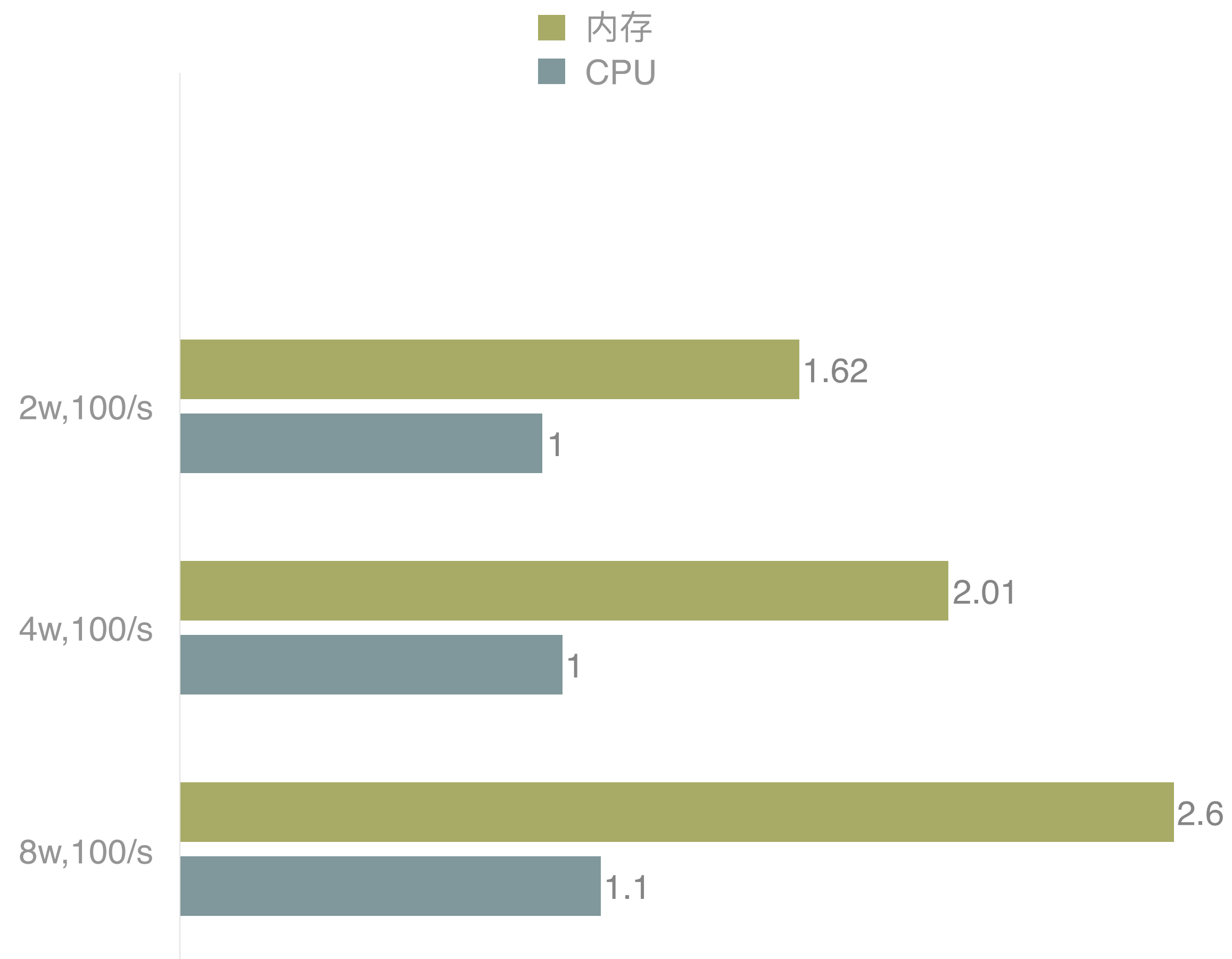
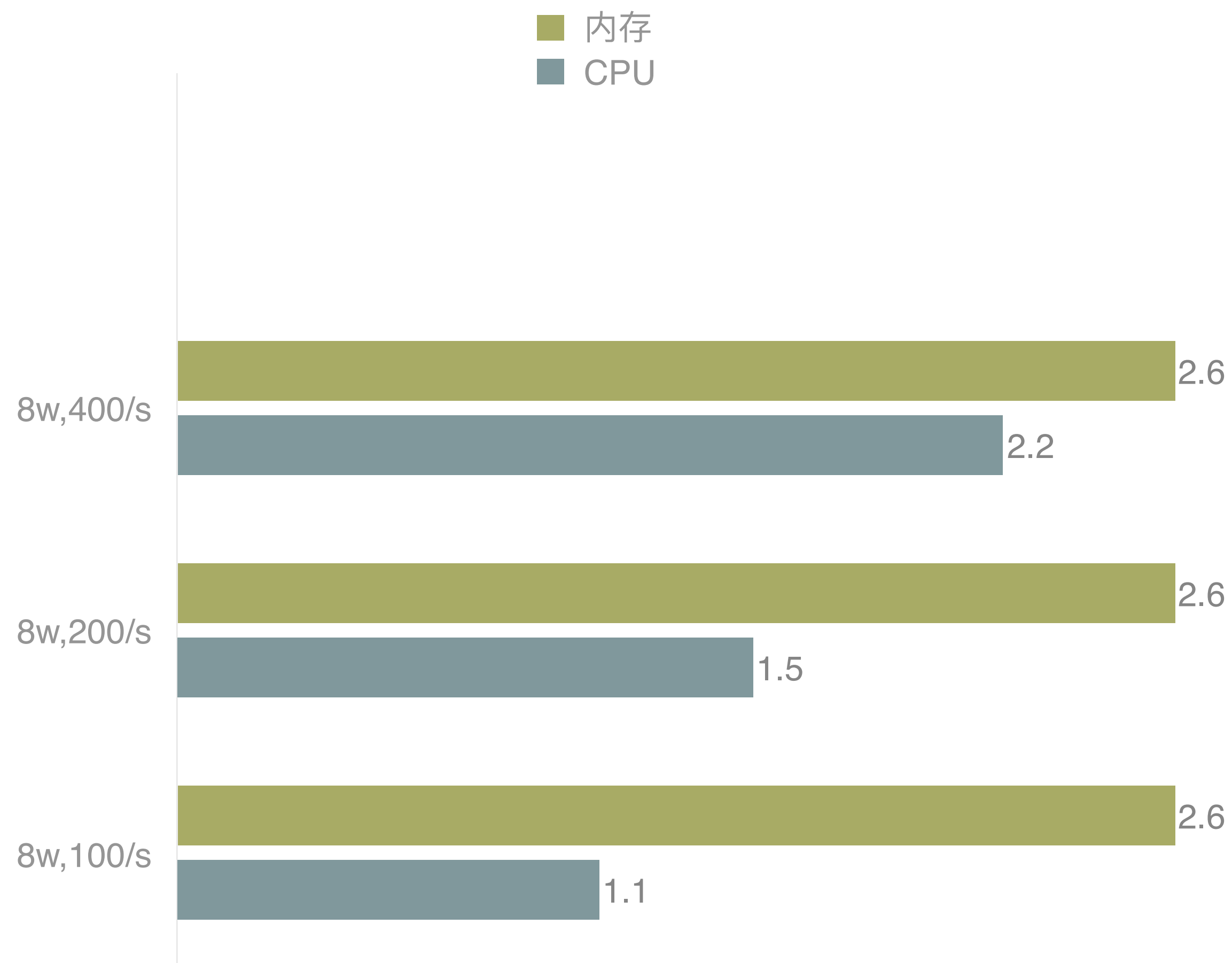
PC端



● 支持 ● 不支持 ● ●



压测



Thank you for your attention!

魏天亮