**nodejs网络请求使用proxy**

**使用proxy的场景**

业务场景下，部分请求有跨域问题，需要服务端进行中转。

私人场景下，部分接口被墙，需要走梯子访问。

**全局网络代理模式**

所有由node服务发送的网络请求，都会走代理服务器发送

**http全局代理**

const http = require("http");

const https = require("https");

const proxy = "http://127.0.0.1:10886"; // 替换为实际 SOCKS5 代理服务器的地址和端口

const agent = new http.Agent({ keepAlive: true, keepAliveMsecs: 1000, maxSockets: 10, proxy: proxy })

const httpsAgent = new https.Agent({ keepAlive: true, keepAliveMsecs: 1000, maxSockets: 10, proxy: proxy })

// 设置全局 HTTP 代理

http.globalAgent = agent;

https.globalAgent = httpsAgent;

**sockt代理**

需要下载npm包**socks-proxy-agent**

执行命令 **npm i -S socks-proxy-agent**

const { SocksProxyAgent } = require("socks-proxy-agent");

const http = require("http");

const https = require("https");

// 创建 SOCKS5 代理程序

const proxy = "socks5://127.0.0.1:10886"; // 替换为实际 SOCKS5 代理服务器的地址和端口

const agent = new SocksProxyAgent(proxy);

// 设置全局 HTTP 代理

http.globalAgent = agent;

https.globalAgent = agent;

**部分请求代理模式**

配置请求options的agent，发送请求时带上

const http = require('http');

const proxy = 'http://yourproxy.com:8080'; // 替换为实际代理服务器的地址和端口

const url = 'http://www.example.com'; // 替换为实际要请求的 URL

const options = {

method: 'GET',

headers: {

'User-Agent': 'Node.js'

},

// 将代理设置为 options.agent，这里使用 http.Agent

agent: new http.Agent({ keepAlive: true, keepAliveMsecs: 1000, maxSockets: 10, proxy: proxy })

};

const req = http.request(url, options, (res) => {

console.log(`statusCode: ${res.statusCode}`);

res.on('data', (chunk) => {

console.log(chunk.toString());

});

});

req.on('error', (error) => {

console.error(error);

});

req.end();