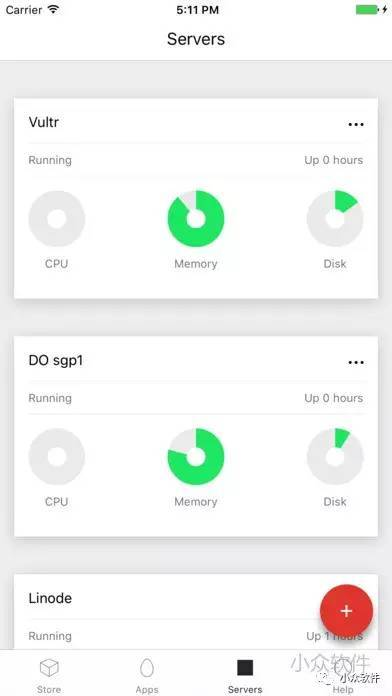
<http://www.sohu.com/a/128748891_464078>

HyperApp 是一款基于 Docker 的自动化部署 Linux 应用的 iPhone 工具，适合敢于尝试的 Linux 零基础用户，无需敲入命令，即可安装那些原本需要命令行才能搞定的 Linux 程序。

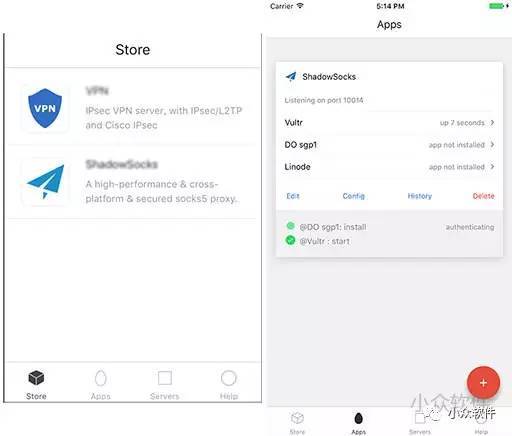


▍背景介绍

这篇文章中出现的一些名词对于零基础用户可能不太好理解，比如 docker、ssh 等，你可以不去管它，并不影响使用。

可以将 HyperApp 理解成可视化的 SSH 工具，它能够看到你的 VPS 信息，包括 CPU 使用率、内存和磁盘占用，以及运行时间，并且还能控制 VPS 重启、关机和查看历史记录（历史记录功能青小蛙木有看到任何信息��）。

通过 Docker虚拟化技术，HyperApp可以很容易的部署一些常见应用，目前有两个应用，青小蛙掩耳盗铃的接了一下图，各位看官老爷们自行补脑：



然后就简单了。

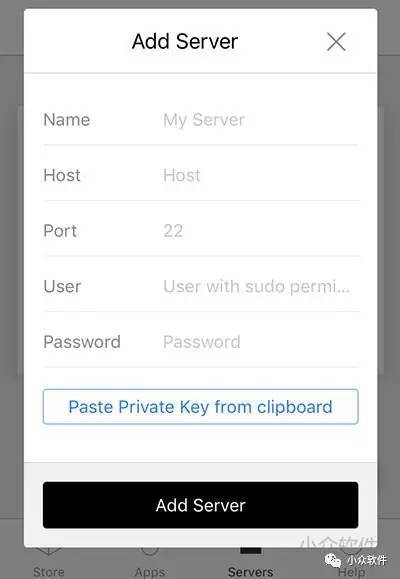
▍使用流程

首先，你需要有一台 VPS：四大 VPS 对比评测：Linode vs. DigitalOcean vs. Lightsail vs. Vultr，由于 Vultr 开启价格战推出了 2.5 刀月费的低端套餐，有需求的同学可以先从 Vultr 开始。

注册地址：http://www.vultr.com/?ref=7112193

当你开通了一台 VPS 后，会得到一个 IP 地址、root 密码，只需要这两个就可以了。

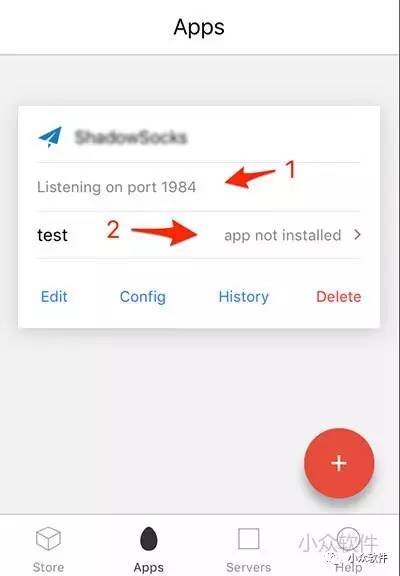
首先需要在 HyperApp的 Servers 功能页添加你的 VPS：



* Name- 为你的 VPS 取个名字，比如 My First VPS。
* Host- VPS 的 IP 地址
* Port- 端口号，一般为 22
* User- VPS 登录用户名，一般为 root
* Password- 密码

最下面那个是密钥登录，不要管了，然后点击 Add Server，就完成了添加。正常情况下，就可以看到与第一张截图类似的状况了，能看到 CPU、内存、硬盘使用情况。

然后，再去 Store 功能页，点击你需要的服务，比如第二个，勾选你要安装的 VPS 名称，保存，再切换到 Apps 功能页，你就会看到上面第二张截图右侧的部分，这里注意了，首先你要点击 Config 按钮来稍微配置一下，如果是 SS 服务，需要填入你想要的端口号（1～65535）、密码和加密方式，保存。



保存后，如果看到上图 1 部分有 Listening on prot xxxx 字样，就没问题了。然后点击 2 位置，选择 Install 就完成了全部设置。

这样就可以去客户端设置了，客户端的设置方法不在本文的讨论范围内，看官老爷们也不要留言问了，毕竟老大哥在看着，分分钟就删了，青小蛙已经尽量通俗易懂了。

根据与开发者的沟通，HyperApp 将来还是会开放更多的 Docker 镜像的，比如 Nextcloud，GitLab，Minecraft Server, RocketChat 等，并且可以输入任意 docker 镜像来进行安装，只是初期隐藏了功能，「是想先上线看看效果的再决定是否花更多精力维护更多镜像。」

<http://www.opendigg.com/p/hyperapp>

项目简介

## hyperapp

HyperApp 是一个用于创建现代UI应用的1kb JavaScript库

## Download

使用npm.

npm i hyperapp

## Usage

嵌入到文档中.

<script src="hyperapp.js"></script>

试试看

const { app, html } = hyperapp

app({

model: "Hi.",

view: model => html`<h1>${model}</h1>`

})

在ES6中.

import { app, html } from "hyperapp"

CommonJS中.

const { app, html } = require("hyperapp")

Browserify

browserify index.js -t hyperxify -g uglifyify | uglifyjs > bundle.js

Webpack

webpack -p --module-bind "js=babel?presets[]=react,presets[]=es2015 index.js bundle.js

## 示例

Hello world

app({

model: "Hi.",

view: model => html`<h1>${model}</h1>`

})

Counter

app({

model: 0,

update: {

add: model => model + 1,

sub: model => model - 1

},

view: (model, msg) => html`

<div>

<button onclick=${msg.add}>+</button>

<h1>${model}</h1>

<button onclick=${msg.sub} disabled=${model <= 0}>-</button>

</div>`

})

Input

app({

model: "",

update: {

text: (\_, value) => value

},

view: (model, msg) => html`

<div>

<h1>Hi${model ? " " + model : ""}.</h1>

<input oninput=${e => msg.text(e.target.value)} />

</div>`

})

Drag & Drop

const model = {

dragging: false,

position: {

x: 0, y: 0, offsetX: 0, offsetY: 0

}

}

const view = (model, msg) => html`

<div

onmousedown=${e => msg.drag({

position: {

x: e.pageX, y: e.pageY, offsetX: e.offsetX, offsetY: e.offsetY

}

})}

style=${{

userSelect: "none",

cursor: "move",

position: "absolute",

padding: "10px",

left: `${model.position.x - model.position.offsetX}px`,

top: `${model.position.y - model.position.offsetY}px`,

backgroundColor: model.dragging ? "gold" : "deepskyblue"

}}

>Drag Me!

</div>`

const update = {

drop: model => ({ dragging: false }),

drag: (model, { position }) => ({ dragging: true, position }),

move: (model, { x, y }) => model.dragging

? ({ position: { ...model.position, x, y } })

: model

}

const subs = [

(\_, msg) => addEventListener("mouseup", msg.drop),

(\_, msg) => addEventListener("mousemove", e =>

msg.move({ x: e.pageX, y: e.pageY }))

]

app({ model, view, update, subs })