

解决LeanCloud定时唤醒失败的流控问题 - 优源博客 | YouYuan Blog

本文转载自[小康博客-《优雅解决LeanCloud流控问题》](#)

1.问题由来

最近好多人在使用LeanCloud时遇到了“因流控原因，通过定时任务唤醒体验版实例失败，建议升级至标准版云引擎实例避免休眠”提示。



jysperm LeanEngine / Node.js 工程师

1 4月14日

我们近期发现有大量云引擎的体验版（免费版）用户创建了在同一时刻触发的定时任务唤醒容器运行，会对服务的负载产生一定影响。因此我们决定根据服务器的负载，对这些定时任务添加流控，通过定时任务唤醒容器有可能会失败。这个改动不会对云引擎的标准版（付费版）用户产生影响。

创建时间

最后回复

2 回复

201 浏览

2 用户



4月14日

4月17日



xiaoxu

4月15日

本主题已置顶，它将始终显示在它所属分类的顶部。可由职员对所有人解除置顶，或者由用户自己取消置顶。



jysperm LeanEngine / Node.js 工程师

4月17日

本主题在创建 3 天后自动关闭。不再允许添加新回复。

看到官方所说：

既然是同一时刻，那么是不是意味着只要搓开时间就可以了呢？

1020-05-14

STDOUT	system	10:00:07	CloudQueue	正在运行	4f061ceb-edc6-4a68-8e88-1acfb726d384: self_wake(null)	
STDOUT	system	10:00:07	CloudQueue	运行失败	4f061ceb-edc6-4a68-8e88-1acfb726d384: self_wake !! {"error": "因流控原因, 通过定时任务唤醒体验版实例失败, 建议升级至标准版云引擎实例避免休眠 https://url1.leanapp.cn/dwAEkxv"}	
STDOUT	system	10:20:03	CloudQueue	正在运行	75c65986-3aaf-4bc4-80ad-111e58fff379: self_wake(null)	
STDOUT	system	10:20:03	CloudQueue	运行失败	75c65986-3aaf-4bc4-80ad-111e58fff379: self_wake !! {"error": "因流控原因, 通过定时任务唤醒体验版实例失败, 建议升级至标准版云引擎实例避免休眠 https://url1.leanapp.cn/dwAEkxv"}	
STDOUT	system	10:40:03	CloudQueue	正在运行	a05745ad-c3e7-4537-ab2a-a4db77d43859: self_wake(null)	
STDOUT	system	10:40:03	CloudQueue	运行失败	a05745ad-c3e7-4537-ab2a-a4db77d43859: self_wake !! {"error": "因流控原因, 通过定时任务唤醒体验版实例失败, 建议升级至标准版云引擎实例避免休眠 https://url1.leanapp.cn/dwAEkxv"}	
STDOUT	system	11:00:07	CloudQueue	正在运行	32ff7782-11a8-4e33-9039-28b3bb3eecb6: self_wake(null)	
STDOUT	system	11:00:07	CloudQueue	运行失败	32ff7782-11a8-4e33-9039-28b3bb3eecb6: self_wake !! {"error": "因流控原因, 通过定时任务唤醒体验版实例失败, 建议升级至标准版云引擎实例避免休眠 https://url1.leanapp.cn/dwAEkxv"}	
STDOUT	system	11:20:03	CloudQueue	正在运行	0e5abb70-906b-4f38-ab04-27b1b1ee4007: self_wake(null)	
STDOUT	system	11:20:03	CloudQueue	运行失败	0e5abb70-906b-4f38-ab04-27b1b1ee4007: self_wake !! {"error": "因流控原因, 通过定时任务唤醒体验版实例失败, 建议升级至标准版云引擎实例避免休眠 https://url1.leanapp.cn/dwAEkxv"}	
STDOUT	system	11:40:03	CloudQueue	正在运行	27c4a7fc-3b7b-42c5-b480-9f74564a9cfc: self_wake(null)	
STDOUT	system	11:40:03	CloudQueue	运行失败	27c4a7fc-3b7b-42c5-b480-9f74564a9cfc: self_wake !! {"error": "因流控原因, 通过定时任务唤醒体验版实例失败, 建议升级至标准版云引擎实例避免休眠 https://url1.leanapp.cn/dwAEkxv"}	
STDOUT	system	12:00:07	CloudQueue	正在运行	a3085910-63ee-4ce3-8021-96cfdc765144: self_wake(null)	
STDOUT	system	12:00:07	CloudQueue	运行失败	a3085910-63ee-4ce3-8021-96cfdc765144: self_wake !! {"error": "因流控原因, 通过定时任务唤醒体验版实例失败, 建议升级至标准版云引擎实例避免休眠 https://url1.leanapp.cn/dwAEkxv"}	
STDOUT	system	12:20:03	CloudQueue	正在运行	40f7e032-b178-48e0-bae6-845b6c03cf2f: self_wake(null)	
STDOUT	system	12:20:03	CloudQueue	运行失败	40f7e032-b178-48e0-bae6-845b6c03cf2f: self_wake !! {"error": "因流控原因, 通过定时任务唤醒体验版实例失败, 建议升级至标准版云引擎实例避免休眠 https://url1.leanapp.cn/dwAEkxv"}	
STDOUT	system	12:40:03	CloudQueue	正在运行	3a3df4e5-ae87-4062-8f5a-5930db8f1875: self_wake(null)	
STDOUT	system	12:40:03	CloudQueue	运行失败	3a3df4e5-ae87-4062-8f5a-5930db8f1875: self_wake !! {"error": "因流控原因, 通过定时任务唤醒体验版实例失败, 建议升级至标准版云引擎实例避免休眠 https://url1.leanapp.cn/dwAEkxv"}	
STDOUT	system	13:00:06	CloudQueue	正在运行	1fc0fa49-34eb-437c-9dc2-efde0256c5bb: self_wake(null)	
STDOUT	system	13:00:07	CloudQueue	运行失败	1fc0fa49-34eb-437c-9dc2-efde0256c5bb: self_wake !! {"error": "因流控原因, 通过定时任务唤醒体验版实例失败, 建议升级至标准版云引擎实例避免休眠 https://url1.leanapp.cn/dwAEkxv"}	
STDOUT	system	13:23:00	CloudQueue	正在运行	1bea1e84-ede2-45b1-a7f9-0ef647bad051: self_wake(null)	
STDOUT	system	13:23:00	CloudQueue	运行失败	1bea1e84-ede2-45b1-a7f9-0ef647bad051: self_wake !! {"error": "因流控原因, 通过定时任务唤醒体验版实例失败, 建议升级至标准版云引擎实例避免休眠 https://url1.leanapp.cn/dwAEkxv"}	
STDOUT	system	13:46:00	CloudQueue	正在运行	9748e398-8386-40da-b3da-99a0d8485351: self_wake(null)	
STDOUT	system	13:46:00	CloudQueue	运行失败	9748e398-8386-40da-b3da-99a0d8485351: self_wake !! {"error": "因流控原因, 通过定时任务唤醒体验版实例失败, 建议升级至标准版云引擎实例避免休眠 https://url1.leanapp.cn/dwAEkxv"}	
STDOUT	system	14:00:07	CloudQueue	正在运行	c6036d60-4994-4956-82d8-2f43c1bd7a06: self_wake(null)	
STDOUT	system	14:00:07	CloudQueue	运行失败	c6036d60-4994-4956-82d8-2f43c1bd7a06: self_wake !! {"error": "因流控原因, 通过定时任务唤醒体验版实例失败, 建议升级至标准版云引擎实例避免休眠 https://url1.leanapp.cn/dwAEkxv"}	
STDOUT	system	14:23:00	CloudQueue	正在运行	41339500-3afa-48bd-a41f-2b9e2f0cf7c1: self_wake(null)	
STDOUT	system	14:23:00	CloudQueue	运行失败	41339500-3afa-48bd-a41f-2b9e2f0cf7c1: self_wake !! {"error": "因流控原因, 通过定时任务唤醒体验版实例失败, 建议升级至标准版云引擎实例避免休眠 https://url1.leanapp.cn/dwAEkxv"}	
STDOUT	system	14:46:00	CloudQueue	正在运行	747cf56e-17b9-4448-b855-d66d25468ed9: self_wake(null)	
STDOUT	system	14:46:00	CloudQueue	运行失败	747cf56e-17b9-4448-b855-d66d25468ed9: self_wake !! {"error": "因流控原因, 通过定时任务唤醒体验版实例失败, 建议升级至标准版云引擎实例避免休眠 https://url1.leanapp.cn/dwAEkxv"}	

原文作者便调整时间尝试了几天，第一天还好，但以后便又出现了流控导致的失败。

因此，调整时间避免的方案治标不治本。还需另寻他法。

2.目前解决思路

- 修改定时任务的唤醒时间

这个方案在上边我已经介绍过了，治标不治本。这里我并不推荐。

- 在博客多加入一条请求。

也就说每一次访问博客时，将 leancloud 唤醒。这种方法可以，这也是我最先想到的，但无疑，这会一定程度上拖慢博客加载速度。

- 第二个方案的变种

为什么说是变种。因为也是加一个请求，只不过不会在你博客加，那么加在哪里呢？请继续往后看。

3.优先解决方案

解决方案其实真的还蛮多的。因为方案很多，我也不可能每种方案都写一篇详细的小白教程，因此发现或者想到新方案后，我会将思路分享给大家。至于具体如何操作，请自己动手，详细过程不可能再会给出教程（特别繁琐除外），本文只会给出一些关键性的代码（脚本），以及代码（脚本）如何使用。

此篇文章详细介绍方案一的做法，其他方案为 2020 年 05 月 18 日后补方案

LeanCloud 的机器唤醒其实还有一种方式。详情请看[休眠策略](#)。

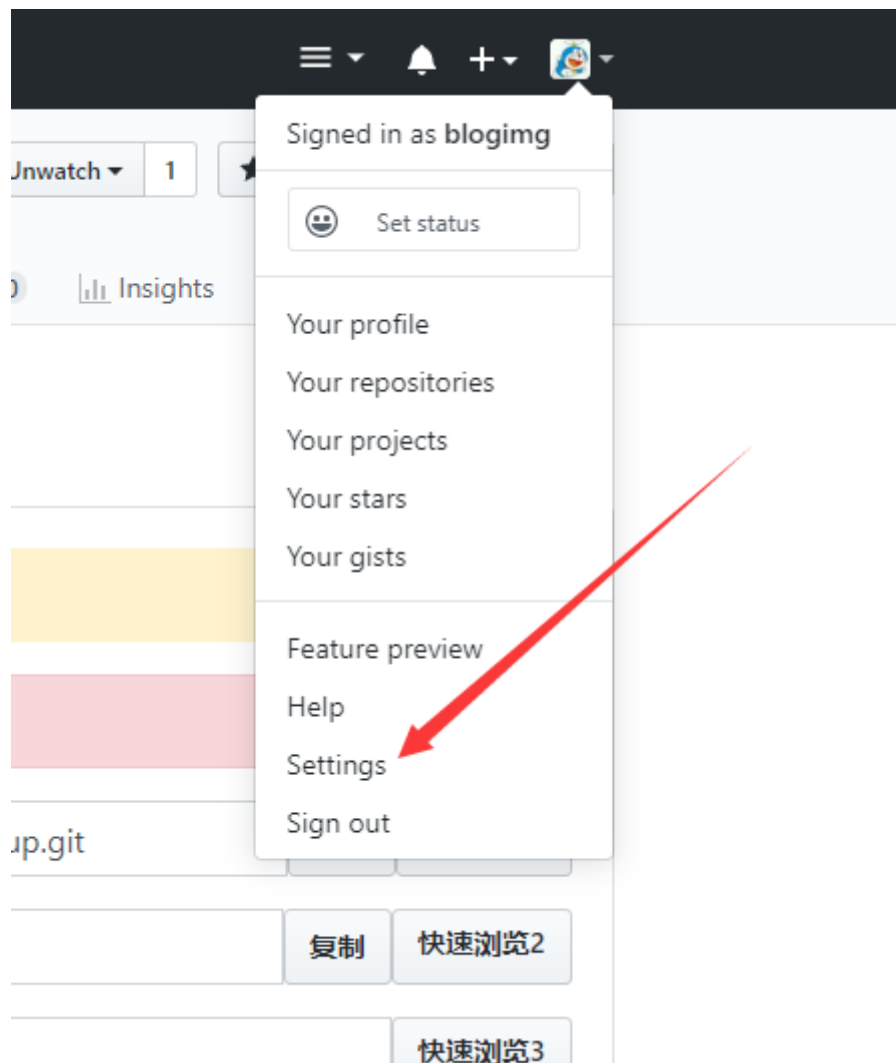
- 如果应用最近一段时间（半小时）没有任何外部请求，则休眠。
- 休眠后如果有新的外部请求实例则马上启动。访问者的体验是第一个请求响应时间是 5 ~ 30 秒（视实例启动时间而定），后续访问响应速度恢复正常。

- 强制休眠：如果最近 24 小时内累计运行超过 18 小时，则强制休眠。此时新的请求会收到 503 的错误响应码，该错误可在 **云引擎 > 统计** 中查看。

那么我们只要每三十分钟之内在外部访问一次不就可以解决了么？

于是我查看了一下 `valine-admin` 的唤醒源代码，自唤醒云函数也是这样实现的。于是便继续开始白嫖。

这里我使用的是 **GitHub+Actions**。是不是很熟悉，自动部署也是这个方案呢。



(1) 鼠标放在右上角，选择 setting

Profile

Account

Security

Security log

Emails

Notifications

Billing

SSH and GPG keys

Blocked users

Repositories

Organizations

Saved replies

Applications

Developer settings

Name

Your name may appear around GitHub where you contribute or are r
remove it at any time.

Public email

Select a verified email to display

You have set your email address to private. To toggle email privacy, c
and uncheck "Keep my email address private."

Bio

Tell us a little bit about yourself

You can @mention other users and organizations to link to them.

URL

Company

You can @mention your company's GitHub organization to link it.

Location

(2) 点击

Developer settings。

(3) 选择 Personal access tokens，添加一个新的 TOKEN。这个 TOKEN 主要使用来启动 actions 和上传结果用的。

Note

GITHUB_TOKEN

What's this token for?

Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes.](#)

☒ repo

Full control of private repositories

☒ repo:status

Access commit status

☒ repo_deployment

Access deployment status

☒ public_repo

Access public repositories

☒ repo:invite

Access repository invitations

☒ security_events

Read and write security events

☐ write:packages

Upload packages to github package registry

<input type="checkbox"/> read:packages	Download packages from github package registry
<input type="checkbox"/> delete:packages	Delete packages from github package registry
<input type="checkbox"/> admin:org	Full control of orgs and teams, read and write org projects
<input type="checkbox"/> write:org	Read and write org and team membership, read and write org projects
<input type="checkbox"/> read:org	Read org and team membership, read org projects
<input type="checkbox"/> admin:public_key	Full control of user public keys
<input type="checkbox"/> write:public_key	Write user public keys
<input type="checkbox"/> read:public_key	Read user public keys
<input checked="" type="checkbox"/> admin:repo_hook	Full control of repository hooks
<input checked="" type="checkbox"/> write:repo_hook	Write repository hooks
<input checked="" type="checkbox"/> read:repo_hook	Read repository hooks
<input type="checkbox"/> admin:org_hook	Full control of organization hooks
<input type="checkbox"/> gist	Create gists
<input type="checkbox"/> notifications	Access notifications
<input type="checkbox"/> user	Update all user data
<input type="checkbox"/> read:user	Read all user profile data
<input type="checkbox"/> user:email	Access user email addresses (read-only)
<input type="checkbox"/> user:follow	Follow and unfollow users
<input type="checkbox"/> delete_repo	Delete repositories
<input type="checkbox"/> write:discussion	Read and write team discussions
<input type="checkbox"/> read:discussion	Read team discussions
<input type="checkbox"/> admin:enterprise	Full control of enterprises
<input type="checkbox"/> manage_billing:enterprise	Read and write enterprise billing data
<input type="checkbox"/> read:enterprise	Read enterprise profile data
<input checked="" type="checkbox"/> workflow	Update github action workflows
<input type="checkbox"/> admin:pgp_key	Full control of public user pgp keys (Developer Preview)
<input type="checkbox"/> write:pgp_key	Write public user pgp keys
<input type="checkbox"/> read:pgp_key	Read public user pgp keys

设置名字为 `GITHUB_TOKEN`，然后勾选 repo，admin:repo_hook，workflow 等选项，最后点击 Generate token 即可。

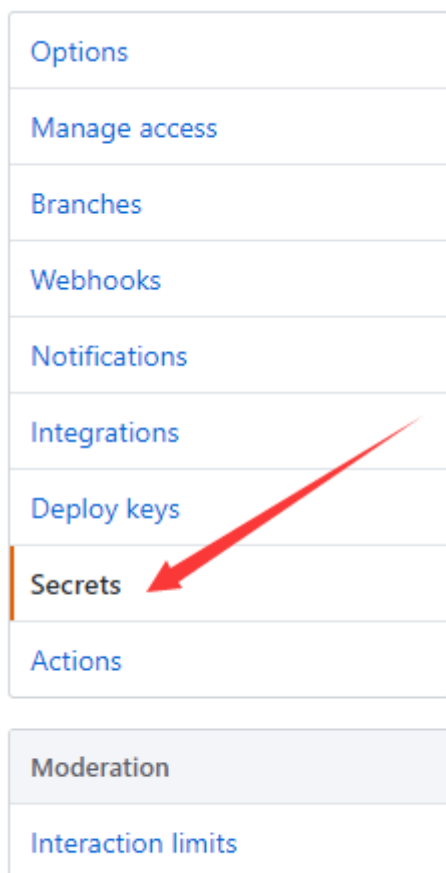
名字请务必使用 `GITHUB_TOKEN`。

(4) 接下来 FORK 项目。

地址：<https://github.com/blogimg/WakeLeanCloud>

如果觉得好用可以点个赞哦！

(5) 成功 FORK 后，进入项目的设置。添加你的 leancloud 的后台地址（也就是评论管理的后台地址）



选择 Secrets，添加你的地址

Name

SITE

Value

https://www.antmoe.com/,https://tzki.cn/

Add secret

其中 Name 的名字必须为 `SITE`，Value 可以是多个地址，用英文逗号分隔。不要中中文逗号，不要用中文逗号，不要用中文逗号

(6) 接下来对自己的项目点个 star 就能启动了，启动后请切换到当前项目的 actions（上方标签，不是左侧标签），看看是否运行成功。

- 成功

那么你就可以关掉了，默认是每天 8:00-24:00 时每 20 分钟运行一次。(GitHub 时间稍有延迟，大概时 2-5 分钟。)

Workflows

All workflows

 Auto Wake Up LeanCloud

All workflows

 Filter workflows

Event ▼ Status ▼ Branch ▼ Actor ▼

✓ Auto Wake Up LeanCloud
Auto Wake Up LeanCloud #322: Scheduled

✓ Auto Wake Up LeanCloud
Auto Wake Up LeanCloud #321: Scheduled

✓ Auto Wake Up LeanCloud
Auto Wake Up LeanCloud #320: Scheduled

✓ Auto Wake Up LeanCloud
Auto Wake Up LeanCloud #319: Scheduled

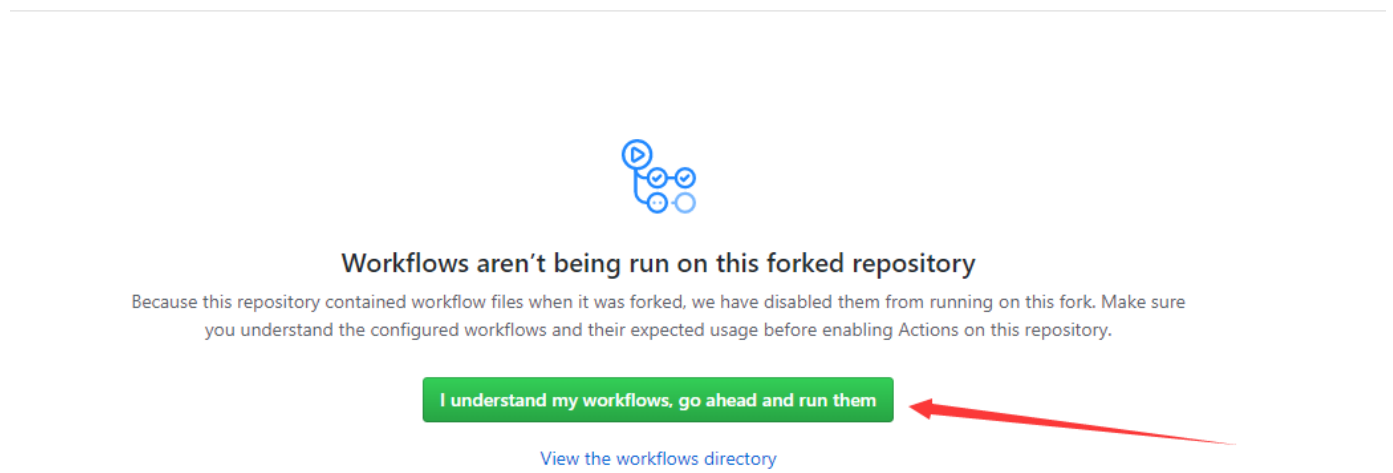
✓ Auto Wake Up LeanCloud
Auto Wake Up LeanCloud #318: Scheduled

✓ Auto Wake Up LeanCloud
Auto Wake Up LeanCloud #317: started by blogimg

✓ Auto Wake Up LeanCloud
Auto Wake Up LeanCloud #316: Scheduled

- 失败

如果你的 GitHub 从来没有用过 actions，那么需要先“了解”。方法很简单，点击绿色的按钮即可。



其他问题：请认真看本教程并且思考为什么这么做，你就能想到你是哪里出了问题。

自己点自己的项目是手动执行一次 actions。是为了测试才设计这个功能的哦！

并不是不点星这个 actions 就不会运行。

(7) 最后，如果觉得好用，请给我点个 star 哦！

4.其他解决方案

这里为 2020-05-18 之后补充的其他方案。

方案二

利用国内的云函数，自己写一个脚本。然后定时监控即可。

或者宝塔、自己服务器的定时任务都是可以的。

方案三

cloudflare 的 Workers 可以在线定义脚本，通过链接即可触发脚本。

因此定义好自己的脚本后，通过监控即可触发来实现唤醒 LeanCloud

javascript:

```
addEventListener('fetch', (event) => {  
  return event.respondWith(handleRequest(event.request));  
})
```



```

const handleRequest = async (request) => {
  const render = (body) => {
    return new Response(`
<!doctype html>
<html>
<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width,initial-scale=1" />
<title>唤醒你的LeanCloud</title>
</head>
<body>${body}</body>
</html>`.trim(), {
    status: 200,
    headers: {
      'Content-Type': 'text/html; charset=utf-8'
    }
  });
}

var date = new Date();
var hour = date.getHours();
var minutes = date.getMinutes();
// 事件控制，因此事件采用utc时间，因此需要手动-8
if (hour >= 0 && hour <= 15) {
  // 列表里添加你的评论管理后台地址
  const Urls = ['https://www.antmoe.com/', 'https://www.tzki.cn/']
  var result = ''
  for(var i=0;i<Urls.length;i++){
    const response = await fetch(Urls[i]);
    const html = await response.status;
    result+=Urls[i]+ "状态: "+html+'<br />'
  }
  return render(`
${result}<br />
`);
} else {
  return render(`当前是休息时间哦! <br />`);
}
}

```


监控平台: <https://uptimerobot.com/> 监控地址就是 Workers 的地址。监控频率看你自己。

另外网友 [track13](#) 评论到: 其实只外部要唤醒一次就可以, 之后都可以交给 leancloud 的定时任务。

方案四

通过 `cron-job` 平台进行监控。注册地址: <https://cron-job.org/en/signup/>

1. 注册



[Welcome](#) [Signup](#) [Members](#) [FAQ](#)

Signup

Sign up for your free cron-job.org account now and start creating your own cronjobs. Signing up is absolutely free and just takes a few minutes. We respect your privacy and do not sell or give away your personal data.

First name (required)

Last name (required)

Email address (required)

Password (required, min. 6 characters)

Repeat (required)


Time zone (required)

Asia/Shanghai

▼

Security check (required)

☐ 进行人机身份验证


reCAPTCHA
隐私权 - 使用条款

By signing up, you agree to our [terms of service](#) and our [privacy policy](#).

Create free account

时区请选择 `Asia/Shanghai`，否则请手动下边的操作请手动换算时间。另外关于最下边的谷歌验证如果出不来，请采用特殊方式。这里不过多介绍。

1. 登录账号

首先去邮箱激活一下账号哦！*邮件可能在垃圾箱哦！*

1. 添加任务

登陆之后依次点击 `Members`, `cronjobs`, `Create cronjob`


Cronjobs


 Create cronjob

	Title, Address	Last execution	Next execution	
You did not create a cronjob yet.				

Note: You can specify advanced settings after adding the cronjob by editing it.


Title, Address

 Title


 http://


☐ Requires HTTP authentication

Username



Password



 Schedule

☒ Every

15

▼

minute(s)

☐ Every day at

0

▼

:

00

▼

☐ Every

1.

▼

of the month at

0


▼

:

00

▼

☐ User-defined


 Notifications

Notify me by email when

☐ execution of the cronjob fails

☐ execution of the cronjob succeeds after it failed before

☒ the cronjob will be disabled because of too many failures

 Common

☐ Save responses

By creating the cronjob, you declare that you are the owner of the website which will be fetched or you have the explicit permission of the website owner to create the cronjob.

Create cronjob

1. 各项配置的解释

字段	简单解释	补充说明
Title	任务名称	

字段	简单解释	补充说明
Address	监控地址	Leancloud 的 Web 主机域名，也就是环境变量的 ADMIN_URL
Schedule	任务周期	分别为每 X 分钟执行、每天 H:MM 执行、每月 DD 日 HH:MM 执行、自定义
Notifications	提醒通知	
Save responses	保存日志	

1. 配置示例

Title, Address

LeanCloud后台唤醒

任务名称，随意填写即可

Leancloud的后台地址

☐ Requires HTTP authentication

Username

Password

这里一般不需要填写，如果你的页面有HTTP的认证，则需要填写

Schedule

☐ Every 15 minute(s)

☐ Every day at 0 : 00

☐ Every 1. of the month at 0 : 00

☒ User-defined 选择自定义

User-defined execution dates

天数，星期和月全选。

小时、分钟按照自己需要即可

Days of month

Days of week

Months

Hours

Minutes

Notice: You can select multiple entires by pressing and holding the Ctrl key while clicking the desired entries. Alternatively, drag over the desired entries while clicking and holding the left mouse button.

Notifications

Notify me by email when

- ☐ execution of the cronjob fails
- ☐ execution of the cronjob succeeds after it failed before
- ☒ the cronjob will be disabled because of too many failures

这里是提醒，根据自己需要选择即可

Common

- ☒ Save responses 保存日志，根据自己需要即可

By creating the cronjob, you declare that you are the owner of the website which will be fetched or you have the explicit permission of the website owner to create the cronjob.

Create cronjob

关于自定义时间，你勾哪个就会在哪个时间段执行。例如五个框里你全选了，那么会每分钟都会执行。因此请各位博主自己想好需要在哪个时间段唤醒，而不是无脑复制。

点下第一个，在按住 `shift` 点最后一个，会权限所有哦！另外 `ctrl` 可以多选

5.问题

1. 修改频率（时间）

修改 `.github/workflows/autoWakeup.yml` 文件中的 `cron` 表达式即可。

```
46 lines (44 sloc) | 1.2 KB
#自动激活Lean Cloud
name: AutoWakeupLeanCloud
on:
  release:
    types: [published]
  push:
    tags:
      - 'v*'
# branches:
#   - master
schedule:
  - cron: "*/18 0-15 * * *"
watch:
  types: [started]
jobs:
  build:
    runs-on: ubuntu-latest
    if: github.event.repository.owner.id == github.event.sender.id # 自己点的 start
    steps:
      - name: Checkout
        uses: actions/checkout@master
```

1. 后台地址会不会暴露

不会的

2. 没有效果

✓

Successful Wake

master AutoWakeUp 8027511

AutoWakeUpLeanCloud

on: watch

✓ build

AutoWakeUpLeanCloud / build

succeeded 1 minute ago in 19s

Search logs

< > ...

▶ ✓ Set up job 3s

▶ ✓ Checkout 4s

▶ ✓ Set up Python 0s

▶ ✓ Install requests 6s

▼ ✓ Waking 3s

1 ▶ Run python run.py ***

8 第0号网址唤醒状态: <Response [200]> 2020-05-15 13:14:47

▶ ✓ Commit 0s

▶ ✓ Push changes 3s

▶ ✓ Post Checkout 0s

▶ ✓ Complete job 0s

请确保你的第五步成功添加了网址，如果没有添加也会定时执行 actions 的动作而不会报错。可以在详情里查看是否监控的你的地址。正常情况下会如下图所以，多个网址会依次排列。如果没有填写网址则会默认访问作者的博客。

1. 每次都会 commit，太多了。

其实可以每天只运行一次，然后其他时间还是依靠定时函数来完成。例如我将 actions 的时间修改为每天早上 8:00 运行一次。而其他时间通过定时函数唤醒。这样**理论上**也是没问题的。

Actions 的时间是按 UTC 时间计算的，因此设置时请手动将时间换算成 UTC 时间哦！

参考文献

*[小康博客-《优雅解决LeanCloud流控问题》](#)

相关文章：

- [使用Jekyll和github pages构建个人wiki知识库 \[2021-01-21\]](#)
- [使用PicGo+GitHub图床 \[2020-06-20\]](#)

- [解决LeanCloud定时唤醒失败的流控问题 \[2020-06-09\]](#)
- [Jekyll不使用插件生成「书单」页面 \[2019-10-27\]](#)
- [获取git博文仓库及Markdown源码 \[2019-10-04\]](#)

「游客及非Github用户留言」：

「Github登录用户留言」：