

## salesforce functions的优势 (Why Salesforce Functions) :

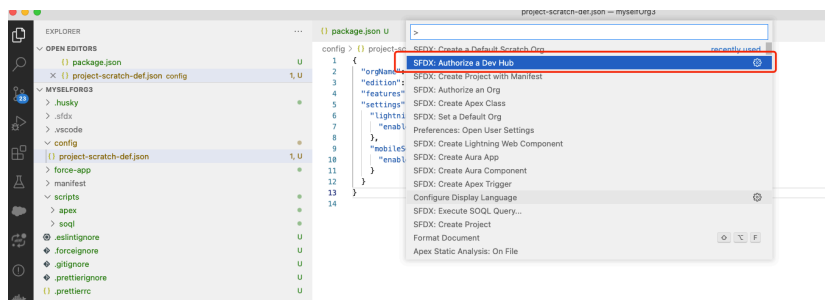
- a) 解决salesforce cpu时间60s限制问题, 把消耗比较大的cpu密集型计算逻辑, 比如复杂计算, 报表生成等, 都交给动态可伸缩的functions计算环境去完成。
- b) 解决Apex无法实现的功能, 如二进制文件的解析等
- c) 可以使用标准语言去实现功能, 如java, JavaScript, typescript, 并能引用该语言的第三方包, 以缩短开发成本。
- d) 不需要一般外部服务复杂的登录授权工作, 该步骤都已经被安全地自动处理了, 开发人员只须关注功能实现, Apex class能简单无缝的调用集成各语言的functions服务。

## 1. 注册一个可以enable devhub功能的org, 并enable该功能

以下示例是注册了30天的试用functions功能的开发账户: <https://functions.salesforce.com/signups/>

## 2. 创建project, 登录授权devhub

通过vs code



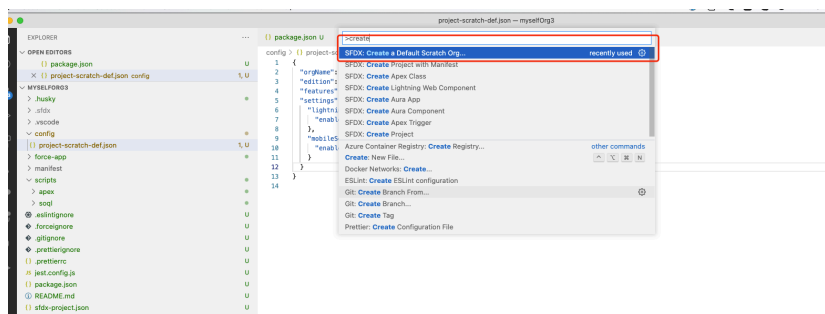
或者用sfdx命令

```
MacBook-Pro-2:~ qiang$ sf generate project -n MyFunctionProject
-n (required) name of the generated project

MacBook-Pro-2:~ qiang$ sf login
-a set an alias for the authenticated org, which you use in subsequent CLI commands
-d set this org as the default Dev Hub for creating scratch orgs
```

## 3. 创建Scratch org

通过vs code创建, 指明利用的config/project-scratch-def.json (配置开启的功能: Functions etc), 过期天数(1-30), 别名等



```
project-scratch-def.json — myselfOrg3
... {} package.json U {} project-scratch-def.json 1, U X
config > {} project-scratch-def.json > ...
1 {
2   "orgName": "qiang company",
3   "edition": "Developer",
4   "features": ["EnableSetPasswordInApi", "Functions"],
5   "settings": {
6     "lightningExperienceSettings": {
7       "enableS1DesktopEnabled": true
8     },
9     "mobileSettings": {
10      "enableS1EncryptedStoragePref2": false
11    }
12  }
13 }
14
```

```
1 {
2   "features": ["Functions"]
3 }
```

或者通过命令行创建

```
MacBook-Pro-2:~ sfdx force:org:create -s -f config/project-scratch-def.json -a MyScratchOrgAlias
```

#### 4. 通过命令行生成scratch org创建时生成的默认admin用户的密码

```
MacBook-Pro-2:~ qiang$ sfdx force:user:password:generate --targetusername test-fuiojknxcudbk@example.com
MacBook-Pro-2:~ qiang$ sfdx force:user:display --targetusername myselfOrg3_scratchOrg1
```

```
TERMINAL  PROBLEMS  OUTPUT  DEBUG CONSOLE
MacBook-Pro-2:myselfOrg3 qiang$ sfdx force:user:display --targetusername myselfOrg3_scratchOrg1
== User Description
key Label
Username test-fuiojknxcudbk@example.com
Profile Name System Administrator
Id 005w0000003UjCPAA0
Org Id 000w00000001LntEAE
Access Token 000w00000001LntEAE0P17vHqE5N_h.5A70V2sc0ZDgF7SLJpa.Sku4HKswcnRm45tU1Sofn.xP2Buq7U6n7Nr1atf1e61RS161YudKge0nB.
Instance Url https://java-business-488-dev-ed.cs75.my.salesforce.com/
Login Url https://cs75.salesforce.com
Alias myselfOrg3_scratchOrg1
MacBook-Pro-2:myselfOrg3 qiang$ sfdx force:user:password:generate --targetusername test-fuiojknxcudbk@example.com
Successfully set the password "x20m477J0Q18" for user test-fuiojknxcudbk@example.com.
You can see the password again by running "sfdx force:user:display -u test-fuiojknxcudbk@example.com".
MacBook-Pro-2:myselfOrg3 qiang$ sfdx force:user:password:generate --targetusername test-fuiojknxcudbk@example.com
Successfully set the password "x20m477J0Q18" for user test-fuiojknxcudbk@example.com.
You can see the password again by running "sfdx force:user:display -u test-fuiojknxcudbk@example.com".
MacBook-Pro-2:myselfOrg3 qiang$ sfdx force:user:display --targetusername myselfOrg3_scratchOrg1
== User Description
key Label
Username test-fuiojknxcudbk@example.com
Profile Name System Administrator
Id 005w0000003UjCPAA0
Org Id 000w00000001LntEAE
Access Token 000w00000001LntEAE0P17vHqE5N_h.5A70V2sc0ZDgF7SLJpa.Sku4HKswcnRm45tU1Sofn.xP2Buq7U6n7Nr1atf1e61RS161YudKge0nB.
Instance Url https://java-business-488-dev-ed.cs75.my.salesforce.com/
Login Url https://cs75.salesforce.com
Alias myselfOrg3_scratchOrg1
Password x20m477J0Q18
MacBook-Pro-2:myselfOrg3 qiang$ sfdx force:user:display --targetusername myselfOrg3_scratchOrg1
```

#### 5. 创建functions功能的运行Salesforce compute environment

```
# dev hub org
MacBook-Pro-2:myselfOrg3 qiang$ sf login org --alias Bob_DevHub --set-default-dev-hub
MacBook-Pro-2:myselfOrg3 qiang$ sf login functions

# This connection can expire after 8 hours and may need to be re-established by running sf login functions again

# 目前试验下来，每次run下面命令会重新创建一个计算环境，同一个org可链接多个不同项目名的计算环境
# 相同项目名，会使用最后创建的那个计算环境
MacBook-Pro-2:myselfOrg3 qiang$ sf env create compute -o MyScratchOrgAlias -a MyComputeEnv

-o Alias of the org the compute environment is connected to.
-a Alias for the newly created compute environment.

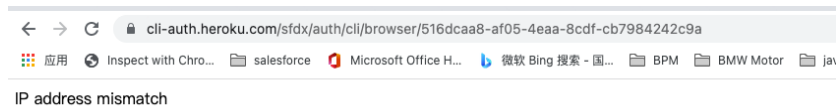
MacBook-Pro-2:myselfOrg3 qiang$ sf env list

# 删除计算运行环境
```

```
MacBook-Pro-2:myselfOrg3 qiang$ sf env delete --target-compute MyComputeEnv
```

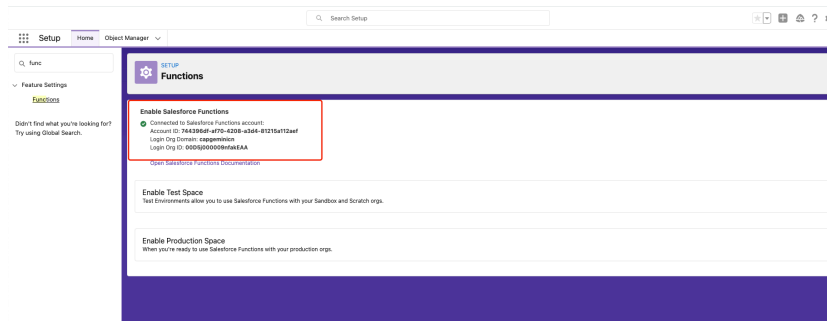
当sf login functions可能会碰到如下错误，该错误是由于国内网络对heroku的限制造成的，当开启代理时会报IP address mismatch，可通过如下步骤解决。

首先开启vpn，run 命令后浏览器自动开启<https://cli-auth.heroku.com/>，等待该页面加载完成。此时可以先关闭vpn(避免后面ip验证)，点击页面上的log in to salesforce后，等待跳转后访问heroku报错（没vpn被墙不能访问heroku），最后开启vpn点击浏览器的重试按钮（再次访问heroku）可成功调整到salesforce，用devhub账号登录后授权。

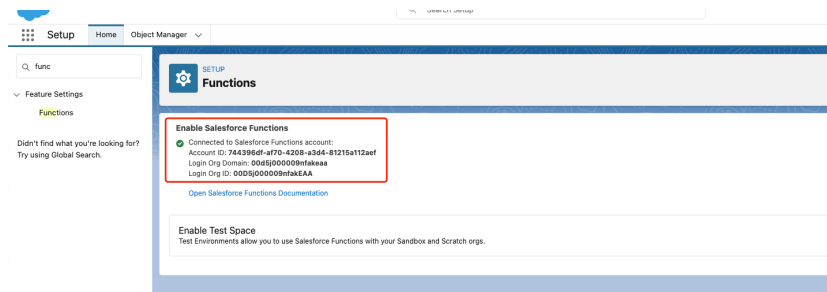


同一个devhub(拥有functions Licenses)和由它创建的scratch org都会自动关联该账户,但他们都拥有各自独立的代码版本和运行环境

devHub functions信息截图如下

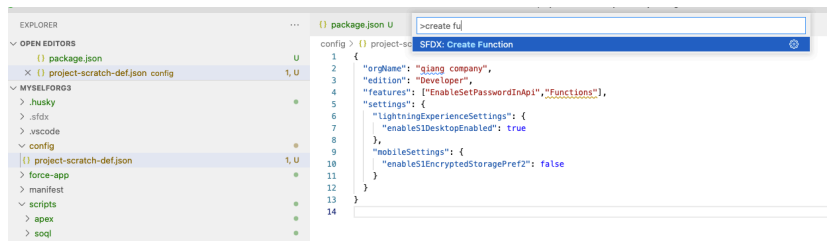


scratch org functions信息截图如下



## 6. 创建Salesforce Function

## vs code创建

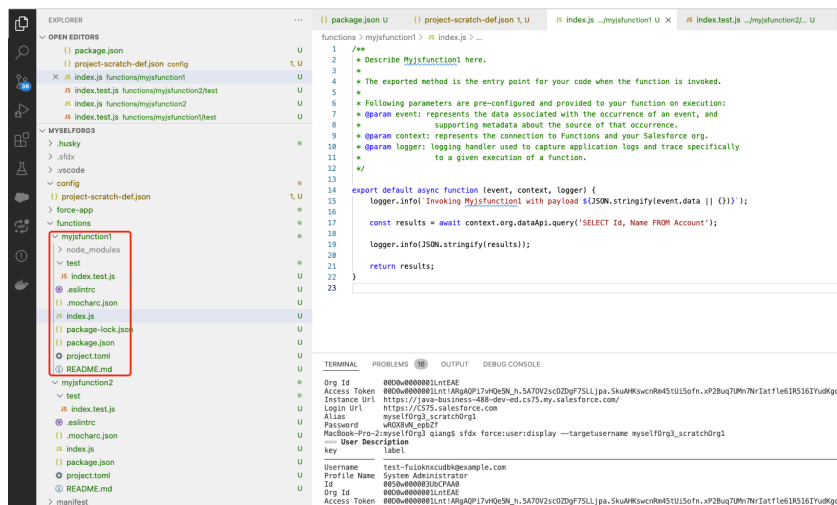


或者用命令行创建

```
MacBook-Pro-2:myselfOrg3 qiang$ sf generate function -n myjsfunction2 -l javascript
-n functionName the name of the Function. Names must start with a letter and contain only lowercase letters and
```

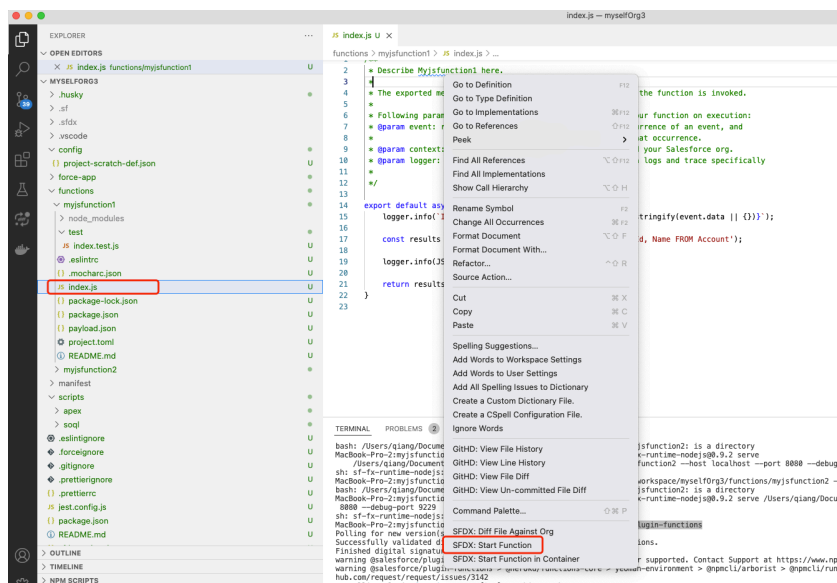
numbers.

-l language the programming language to use.



## 7. 运行本地functions服务

vs code本地运行functions project （停止服务，类似通过index.js文件右击选择SFDX: Stop Function）



命令行启动

```
MacBook-Pro-2:myselfOrg3 qiang$ cd functions/myjsfunction2/
```

```
MacBook-Pro-2:myjsfunction2 qiang$ sf run function start
```

# 如遇错误sh: sf-fx-runtime-nodejs: command not found, 执行sudo sfdx plugins:install @salesforce/plugin-functions  
重装functions plugin

## 8. 调用本地functions服务

开启一个新的命令行工具（开启服务的命令行工具窗口不能关闭，关闭就会停止服务），调用测试接口

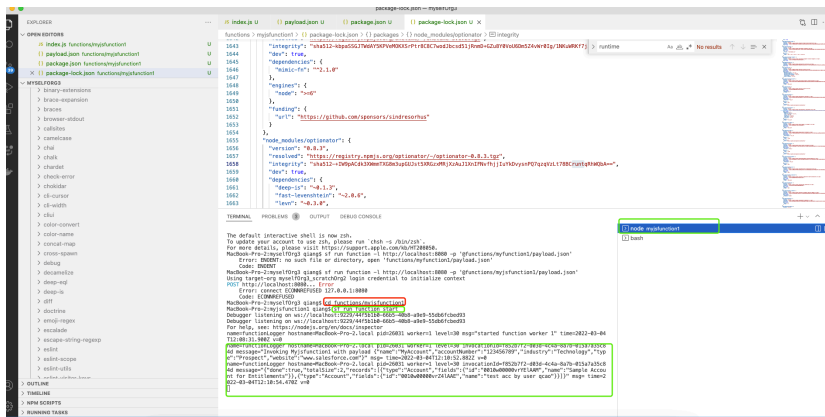
# 在myfunction1建立payload输入数据json文件

```
MacBook-Pro-2:myselfOrg3 qiang$ sf run function -l http://localhost:8080 -p
```

```
'@functions/myfunction1/payload.json'
```

或者

```
MacBook-Pro-2:myselfOrg3 qiang$ sfdx run:function -l http://localhost:8080 -p '{}' -o MyScratchOrg
```



## 9. 部署到salesforce的计算环境（和org绑定）

前置条件：已经创建好该计算环境了，并登录授权在8小时有效期内。

首先要提交整个项目代码到git，才能通过命令行部署，它是根据git修改提交的修改进行增量部署的（git commit只有该project的文件），远端代码库不需要也可以（实际heroku有自己的代码库，部署时会推送到该库），然后通过下列命令进行部署。

# 查看环境是否已创建

```
MacBook-Pro-2:myselfOrg3 qiang$ sf env list
```

```
MacBook-Pro-2:myselfOrg3 qiang$ sf env log tail -e myselfComputeEnv_scratch2
```

# 创建本地git rep，并提交，完成部署前置准备。实际测试

```
MacBook-Pro-2:myselfOrg3 qiang$ git init
```

```
MacBook-Pro-2:myselfOrg3 qiang$ git add .
```

```
MacBook-Pro-2:myselfOrg3 qiang$ git commit -m "Initial project commit after project creation"
```

```
MacBook-Pro-2:myselfOrg3 qiang$ sf deploy functions -o myselfOrg3_scratchOrg2
```

# 首次部署可能需要上传超过500MB的文件，需等待很长时间。

```
The default interactive shell is now zsh.
To update your account to use zsh, please run 'chsh -s /bin/zsh'.
For more details, please visit https://support.apple.com/kb/HT208050.
MacBook-Pro-2:myselfOrg3 qiang$ sf deploy functions -o myselfOrg3_scratchOrg2
Error: Your repo has files that have not been committed yet. Please either commit or stash them before deploying your project.
MacBook-Pro-2:myselfOrg3 qiang$ sf deploy functions -o myselfOrg3_scratchOrg2
remote: !
remote:      Size of checkout and restored submodules exceeds 1 GB. Reduce size and try pushing again.
remote:
remote: To https://git.heroku.com/myselfo-0000000001degeam-365.git
remote: ! [remote rejected]      master -> master (pre-receive hook declined)
error: failed to push some refs to 'https://qiango.cao40cappemini.com:40REDACTED@git.heroku.com/myselfo-0000000001degeam-365.git'
Pushing changes to functions... failed
Error: There was an issue when deploying your functions.
MacBook-Pro-2:myselfOrg3 qiang$
```

上面错误是由于该git仓库包含了太多非本项目的文件，应该只含有该项目的文件。

## 10. Apex调用functions

只需要用Function class就能直接调用，不需要登录和授权。

项目名为创建DX project时的命名（根目录名, sfdx-project.json里的名字）。

同步调用方法：

```
1 public with sharing class FunctionApex {
2     public static void test() {
3         System.debug('Invoking myfunction');
4
5         functions.Function myFunction = functions.Function.get('MyFunctionProject.myfunction');
6         functions.FunctionInvocation invocation = myFunction.invoke({'name': 'MyAccount'});
7         String jsonResponse = invocation.getResponse();
8
9         System.debug('Response from myfunction ' + jsonResponse);
10    }
11 }
```

异步调用方法：

```
1 public with sharing class FunctionApex {
2     public static void test() {
3         functions.Function myFunction = functions.Function.get('MyProject.myfunction');
4         // Pass a callback to Function.invoke to perform an asynchronous invocation.
5         functions.FunctionInvocation invocation = myFunction.invoke({'fields': ['Id', 'Name']}), new DemoCalll
6     }
7
8     public class DemoCallback implements functions.FunctionCallback {
9         public void handleResponse(functions.FunctionInvocation result) {
10             // Handle result of function invocation
11             String jsonResponse = result.getResponse();
12             System.debug('Got response ' + jsonResponse);
13
14             // Log error, if applicable
15             if (result.getStatus() == functions.FunctionInvocationStatus.ERROR) {
16                 functions.FunctionInvocationError resultError = result.getError();
17                 if (resultError != null) {
18                     System.debug('Error type: ' + resultError.getType());
19                     System.debug('Error message: ' + resultError.getMessage());
20                 } else {
21                     System.debug('Error: UNKNOWN');
22                 }
23                 return;
24             }
25             // Successful response, deserialize the response to an Apex object.
26             JSONParser parser = JSON.createParser(jsonResponse);
27             Response response = (Response)parser.readValueAs(Response.class);
28             // Verify that the Function actually wrote to the org.
29             Account account = [ SELECT Name FROM Account WHERE Id = :response.accountId ];
30             System.debug('Found account with name ' + account.Name);
31         }
32     }
33
34     // A plain old data class to which we can deserialize the Function's response.
35     public class Response {
36         public String accountId;
37         public String contactId;
```

```

38     public String opportunityId;
39     }
40 }

```

## 11. 查看Compute Environment Logs

# 查看环境

```
MacBook-Pro-2:myselfOrg3 qiang$ sf env list
```

# 查看实时log, 如401错误需重新登录授权sf login functions

```
MacBook-Pro-2:myselfOrg3 qiang$ sf env log tail -e myselfComputeEnv_scratch2
```

```

Scratch Orgs
=====
| Aliases | Username | Org ID | Instance URL | Auth Method | Config |
|-----|-----|-----|-----|-----|-----|
| qcao-scratch-org | test-gwbnhsalqecaggenini_qiang.net | 000200000000000000000000 | https://agility-agility-5525-dev-ed.cs11.my.salesforce.com | web | |
| myselfOrg3_scratchOrg1 | test-fuzhnxcudh@example.com | 000000000000000000000000 | https://java-business-468-dev-ed.cs75.my.salesforce.com/ | web | |
| myselfOrg3_scratchOrg2 | test-nptwboedhne@example.com | 000000000000000000000000 | https://connect-business-5746-dev-ed.cs75.my.salesforce.com/ | web | target-org

Compute Environments
=====
| Alias | Project Name | Connected Org Alias | Connected Org Id | Compute Environment Name | Compute Environment Id |
|-----|-----|-----|-----|-----|-----|
| myselfComputeEnv_scratch3 | myselfOrg3 | myselfOrg3_scratchOrg2 | 000000000000000000000000 | myselfe-000000000000000000000000 | 3b25564c-0f34-4a1c-0901-60ed9c556d4

MacBook-Pro-2:myselfOrg3 qiang$ sf env log tail -e myselfComputeEnv_scratch2
2022-03-04T11:21:37.696378+08:00 app[api]: Initial release by user 000510000000000000000000nfakeaa-test@herokumanager.com
2022-03-04T11:21:37.696378+08:00 app[api]: Release v1 created by user 000510000000000000000000nfakeaa-test@herokumanager.com
2022-03-04T11:21:38.035728+08:00 app[api]: Release v2 created by user 000510000000000000000000nfakeaa_005100000000000000000000nfakeaa@evergreeninternal.herokai.com
2022-03-04T11:21:38.035728+08:00 app[api]: Enable LogLex by user 000510000000000000000000nfakeaa_005100000000000000000000nfakeaa@evergreeninternal.herokai.com

```

Note:

1. 可以只提交functions目录下的文件（但git提交必须包含functions目录），多个功能时，他们都位于functions目录下，而且可以不同语言混合，如JavaScript和Java，启动成功后，他们都能在Apex class里同时访问执行（每个function在heroku里应该是独立的线程）。
2. 目前function的heroku服务器都在美国，如salesforce服务器在亚太或欧洲，function直接调用salesforce的标准api获取大数据量时，会存在较大时延。

Reference:

<https://developer.salesforce.com/docs/platform/functions/guide/create-dx-project.html>

[https://developer.salesforce.com/docs/atlas.en-us.sfdx\\_dev.meta/sfdx\\_dev/sfdx\\_dev\\_scratch\\_orgs\\_passwd.htm](https://developer.salesforce.com/docs/atlas.en-us.sfdx_dev.meta/sfdx_dev/sfdx_dev_scratch_orgs_passwd.htm)