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。

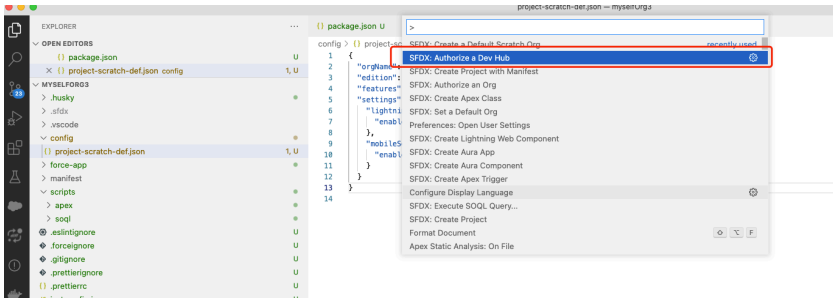
d) 不需要一般外部服务复杂的登录授权工作，该步骤都已经被安全地自动处理了，开发人员只须关注功能实现，Apex class能简单无缝的调用集成各语言的functions服务。

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

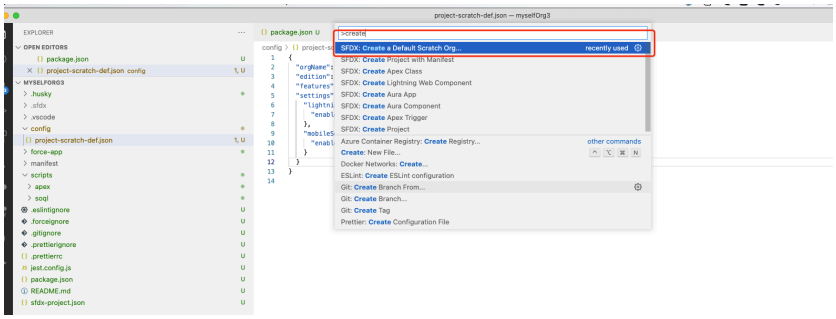


或者用sfdx命令

```
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




```
{
  "features": ["Functions"]
}
```

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

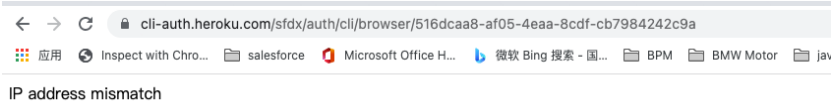
```
MacBook-Pro-2:~ qiang$ sfdx force:user:password:generate --targetusername test-fuioknxcudbk@example.com
MacBook-Pro-2:~ qiang$ sfdx force:user:display --targetusername myselfOrg3_scratchOrg1
```

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
```

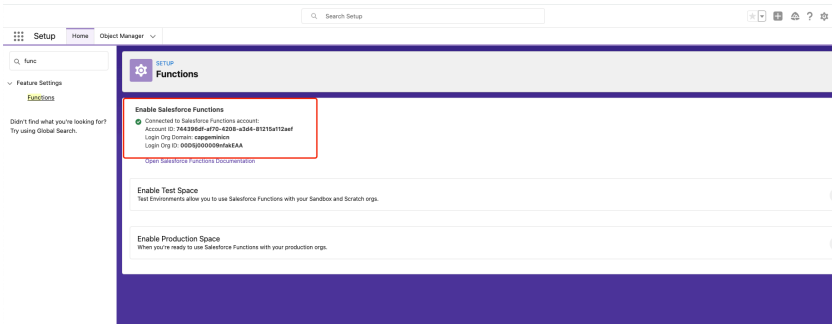
```
MacBook-Pro-2:myself0rg3 qiang$ sf env delete --target-compute MyComputeEnv
```

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。

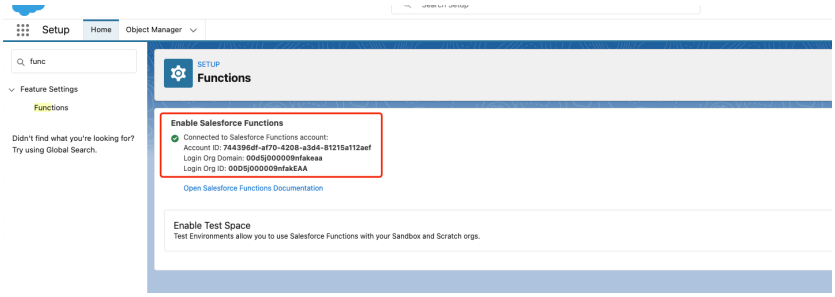


同一个devhub(拥有functions Licenses)和由它创建的scratch org都会自动关联该账户, 但他们都拥有各自独立的代码版本和运行环境

devHub functions信息截图如下

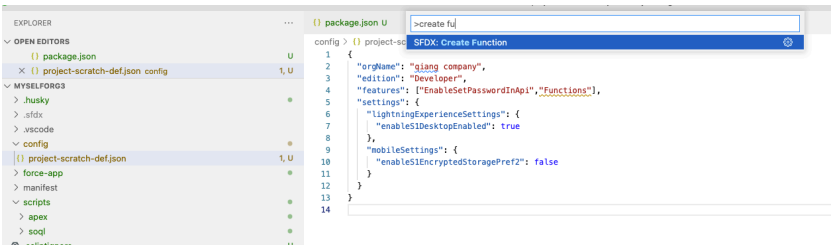


scratch org functions信息截图如下



6. 创建Salesforce Function

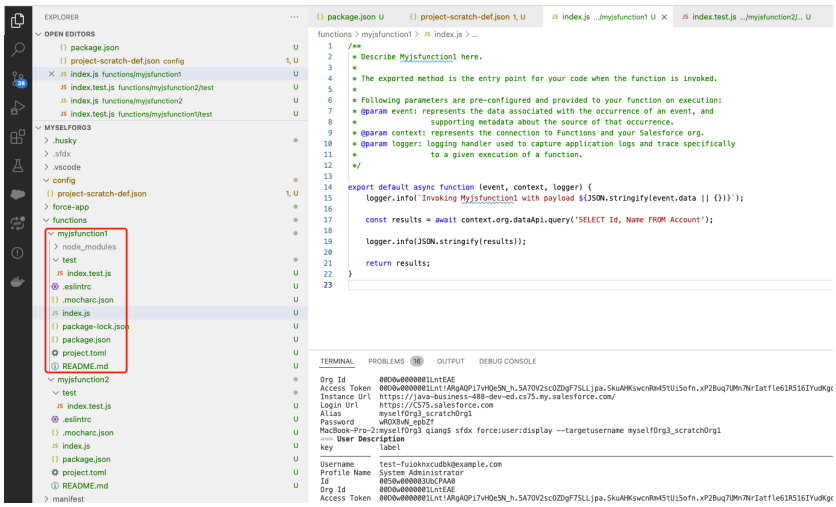
vs code创建



或者用命令行创建

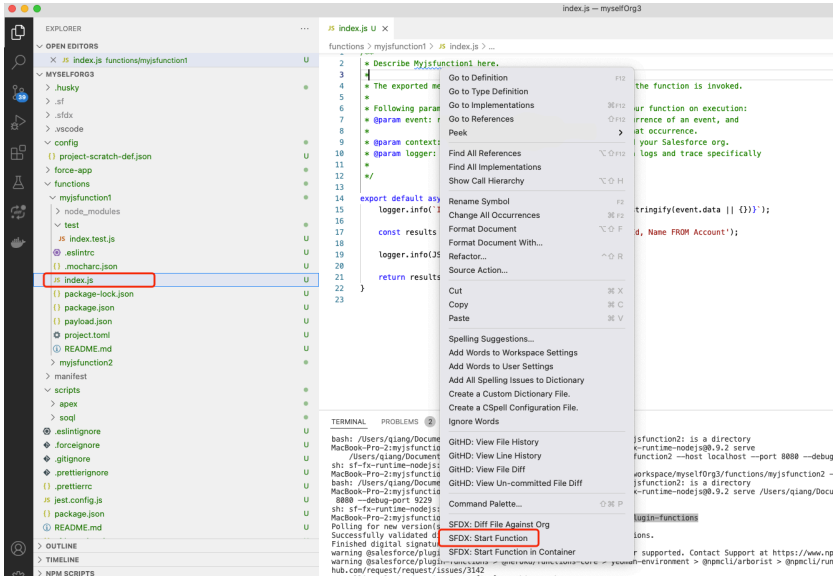
```
MacBook-Pro-2:myself0rg3 qiang$ sf generate function -n myjsfunction2 -l javascript
-n functionName the name of the Function. Names must start with a letter and contain only lowercase letters and
```

```
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）



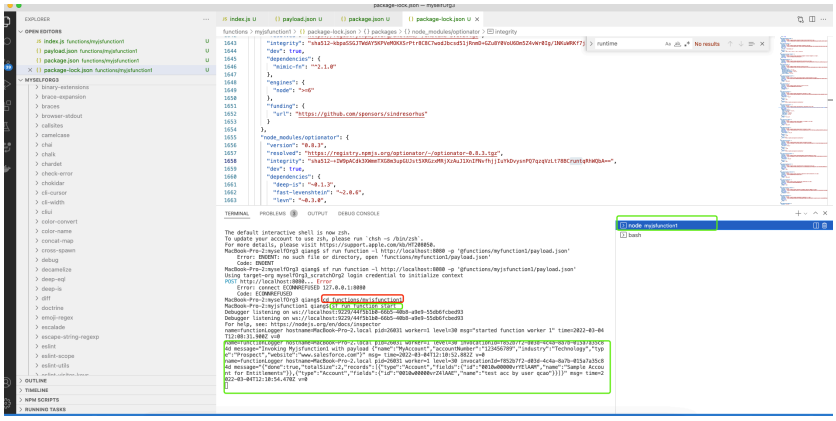
命令行启动

```
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saleforce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:
To https://git.heroku.com/myselfo-00d0w0000001degeam-365.git
 ! [remote rejected]     master -> master (pre-receive hook declined)
error: failed to push some refs to 'https://qiang.cao%40capgemini.com.func:<REDACTED>@git.heroku.com/myselfo-00d0w0000001degeam-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：

```
public with sharing class FunctionApex {
    public static void test() {
        System.debug('Invoking myfunction');

        functions.Function myFunction = functions.Function.get('MyFunctionProject.myfunction');
        functions.FunctionInvocation invocation = myFunction.invoke('{"name":"MyAccount"}');
        String jsonResponse = invocation.getResponse();

        System.debug('Response from myfunction ' + jsonResponse);
    }
}
```

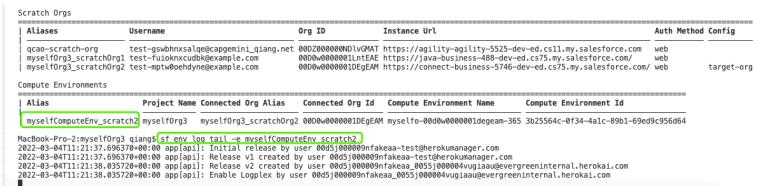
异步调用方法：

```
public with sharing class FunctionApex {
    public static void test() {
        functions.Function myFunction = functions.Function.get('MyProject.myfunction');
        // Pass a callback to Function.invoke to perform an asynchronous invocation.
        functions.FunctionInvocation invocation = myFunction.invoke('{"fields":["Id","Name"]}', new DemoCallb
    }

    public class DemoCallback implements functions.FunctionCallback {
        public void handleResponse(functions.FunctionInvocation result) {
            // Handle result of function invocation
            String jsonResponse = result.getResponse();
            System.debug('Got response ' + jsonResponse);

            // Log error, if applicable
            if (result.getStatus() == functions.FunctionInvocationStatus.ERROR) {
                functions.FunctionInvocationError resultError = result.getError();
                if (resultError != null) {
                    System.debug('Error type: ' + resultError.getType());
                    System.debug('Error message: ' + resultError.getMessage());
                } else {
                    System.debug('Error: UNKNOWN');
                }
                return;
            }
            // Successful response, deserialize the response to an Apex object.
            JSONParser parser = JSON.createParser(jsonResponse);
            Response response = (Response)parser.readValueAs(Response.class);
            // Verify that the Function actually wrote to the org.
            Account account = [ SELECT Name FROM Account WHERE Id = :response.accountId ];
            System.debug('Found account with name ' + account.Name);
        }
    }

    // A plain old data class to which we can deserialize the Function's response.
    public class Response {
        public String accountId;
        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
```



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