在Azure Pipeline中显示代码覆盖率

Azure Pipeline是微软云上全线DevOps工具集中负责自动化流水线的托管服务，具备完备的流水线管理功能，还方便和各种测试、构建和部署工具集成。代码覆盖率是自动化测试中的一个重要指标，统计测试过程中被执行的源代码占全部源代码的比例，进而间接度量软件质量。如果项目代码配置了导出代码覆盖率，Azure Pipeline可以采集相应的数据并存储下来。不过默认情况下，这些数据只提供下载，再使用其它工具查看。本文简单介绍如何在Azure Pipeline的控制台上可视化地显示代码覆盖率。

# 准备项目源码

我们以dotnet new xunit来初始化一个标准的.NET测试用例项目。在VS Code中显示如下：

Text

Description automatically generated

要实现代码覆盖率，核心的配置在NetCodeCoverage.csproj这个文件中的下面这段

<PackageReference Include="coverlet.collector" Version="3.1.2">

<IncludeAssets>runtime; build; native; contentfiles; analyzers; buildtransitive</IncludeAssets>

<PrivateAssets>all</PrivateAssets>

</PackageReference>

如果没有，可以自己手动添加上去。

然后执行 dotnet build和dotnet test命令，可以看到以下输出。

A screenshot of a computer

Description automatically generated

默认创建的测试使用里面没有任何代码逻辑，当然是测试通过啦。如果有问题，还可以从我的GitHub直接fork源码。

<https://github.com/xfsnow/NetCodeCoverage>

# 创建Azure Pipeline

在 Azure DevOps控制台找到Pipeline下的Pipeline，右上角New pipeline创建一个新的yaml格式的pipeline。

Where is your code 先GitHub。连接到前面刚刚提交的GitHub源码库。

Configure页选Starter template。最后来到Review页，我们看到以下一个起始的流水线样子。

Graphical user interface, text, application, email

Description automatically generated

把现有的 2 条script 任务删掉。点右上角的Show assistant，在Tasks中点击 .NET Core，如下图所示，先添加一个build任务。

Graphical user interface, text, application

Description automatically generated

然后Command菜单选择test

Graphical user interface, text, application

Description automatically generated

注意保持Publish test results and code coverage选中不动，再添加一个测试任务。

最后添加一个发布代码覆盖率任务。

最后点击右上角Save and run按钮，按提示commit到源码库并执行。

准备Open ID URL。使用Azure全球区域时，此URL规则是

https://login.microsoftonline.com/{Tenant-ID}/v2.0/.well-known/openid-configuration

使用Azure中国区域时，此URL规则是

https://login.partner.microsoftonline.cn/{Tenant-ID}/v2.0/.well-known/openid-configuration

在API Management控制台，打开我们已经准备好的API。在Design界面，Inbound policy点击Add policy。

Graphical user interface, application

Description automatically generated

再向下翻页，点选Validate JWT。

Header name输入Authorization。

Failed validation error message输入Unauthorized. Access token is missing or invalid.

点击Add claim，name输入aud。再点击Add value，Values输入前述记录下的Azure AD中注册的后端应用的client ID.

点击 Add Open ID URL，输入前述准备好的Open ID URL。点击最下面的Save按钮保存。

Graphical user interface, text, application, email

Description automatically generated

到此为API启用OAuth 2.0 验证配置完成。现在我们直接访问这个API，返回的是验证失败的错误了。

curl https://snowpeak.azure-api.cn/contonso/name

{ "statusCode": 401, "message": "Unauthorized. Access token is missing or invalid." }

# 验证测试

## 使用Azure AD中的客户端应用获取访问token

我们使用OAuth 2.0的标准流程，组织请求。使用前述Azure AD 的Tenant ID 组织目标地址。使用Azure全球区域时，目标地址是

https://login.microsoftonline.com/{Tenant-ID}/oauth2/v2.0/token

使用Azure 中国区域时，目标地址是

https://login.partner.microsoftonline.cn/{Tenant-ID}/oauth2/v2.0/token

参数说明如下：

|  |  |
| --- | --- |
| 参数名 | 参数值 |
| grant\_type | client\_credentials |
| client\_id | 前述客户端应用的Application (client) ID，如6bfe51f7-ea34-4096-aa7c-7e8df1043bbd |
| client\_secret | 前述客户端应用的密钥，如Lz36Zrilxa~01234567890#ABCDEFGH |
| Scope | 以后端应用ID按 {backend-application-Application-ID}/.default 格式组织  如95ac855d-724d-4bf7-95b1-93244f4e2e5b/.default |

组织好请求地址和参数后，我们以curl发送请求，并得到响应，注意再加一个Content-Type: application/x-www-form-urlencoded的请求头。

curl -X POST -H "Content-Type: application/x-www-form-urlencoded" -d 'client\_id=6bfe51f7-ea34-4096-aa7c-7e8df1043bbd&client\_secret=Lz36Zrilxa~pB3d6LAewX~33~6rWREFLC.&scope=95ac855d-724d-4bf7-95b1-93244f4e2e5b/.default&grant\_type=client\_credentials' 'https://login.partner.microsoftonline.cn/78c60b1a-3a93-4e0e-99ad-00dc772cfb6f/oauth2/v2.0/token'

{"token\_type":"Bearer","expires\_in":3599,"ext\_expires\_in":3599,"access\_token":"eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6Il9DSkFPdHlzWVZtNXhjMVlvSzBvUTdxeUJDUSJ9..2PsH5TO79q7Ix7FrkGP\_klLPjpkSC14mPhheBLWTig8d9UX565replym7baKbVX7VuXorwkVHdqLQr\_1AxzgylwMA7FWWTKeBEYWdfM7vcf58Ub46Q4Y2o0HD1jAqlRzyX2iLOvFsA5o0Hi4\_fYrJ6Nt8XW1HXdg2-PS8hdFTmudW\_NHVRD2B7FbmoAUwBV56MdMxmMuDY\_L6isBHCcaeqH9YmzTAosNg-ValNd67h8BLtStIlZOC3\_q6pzlhJB\_ox4wLsYel8tXPUzElli-S72Oo9WLVTB9UOhOTQfuknkpD2rPrGB\_IcQPkFUwo8HsaZ2azSzII5MJdfgUqzUMuQ"}

浏览器打开<https://jwt.ms/>，我们把返回的access\_token解析看一下。

Graphical user interface, text, application

Description automatically generated

可以看到解析出详细的token信息，其中 iss 就是token发布者的URL。这就是前面为后端应用设置accessTokenAcceptedVersion为2的效果。

最后我们添加请求头，再访问已经加上验证的API。

curl -H'Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzIlvSzBvUTdxeUJDUSJ9..2PsH5TO79q7Ix7FrkGP\_klLPjpkSC14mPhheBLWTig8d9UX565replym7baKbVX7VuXorwkVHdqLQr\_1AxzgylwMA7FWWTKeBEYWdfM7vcf58Ub46Q4Y2o0HD1jAqlRzyX2iLOvFsA5o0Hi4\_fYrJ6Nt8XW1HXdg2-PS8hdFTmudW\_NHVRD2B7FbmoAUwBV56MdMxmMuDY\_L6isBHCcaeqH9YmzTAosNg-ValNd67h8BLtStIlZOC3\_q6pzlhJB\_ox4wLsYel8tXPUzElli-S72Oo9WLVTB9UOhOTQfuknkpD2rPrGB\_IcQPkFUwo8HsaZ2azSzII5MJdfgUqzUMuQ' 'https://snowpeak.azure-api.cn/contonso/name?name=test'

{"message": "Hello, test! Welcome to Azure Function!", "timestamp": "2022-09-05, 07:50:57"}

可以看到通过通过验证，并返回了正常的结果。