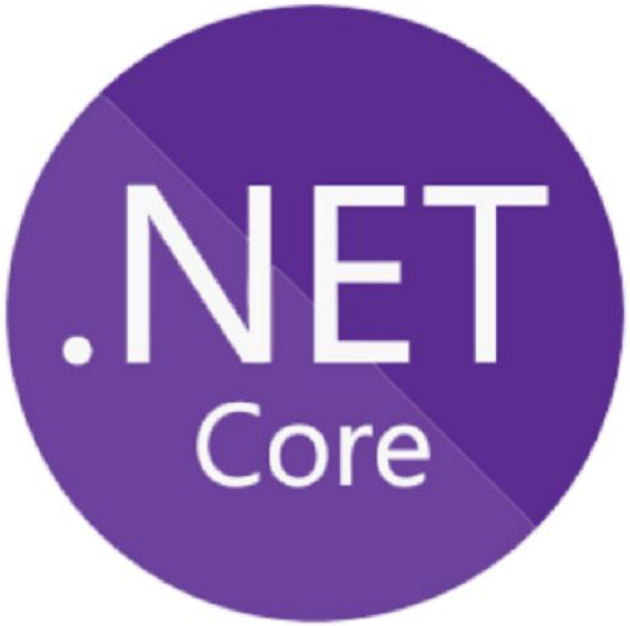

maxcode

maxcode

.NET Core and AWS

Building, deploying & hosting REST APIs

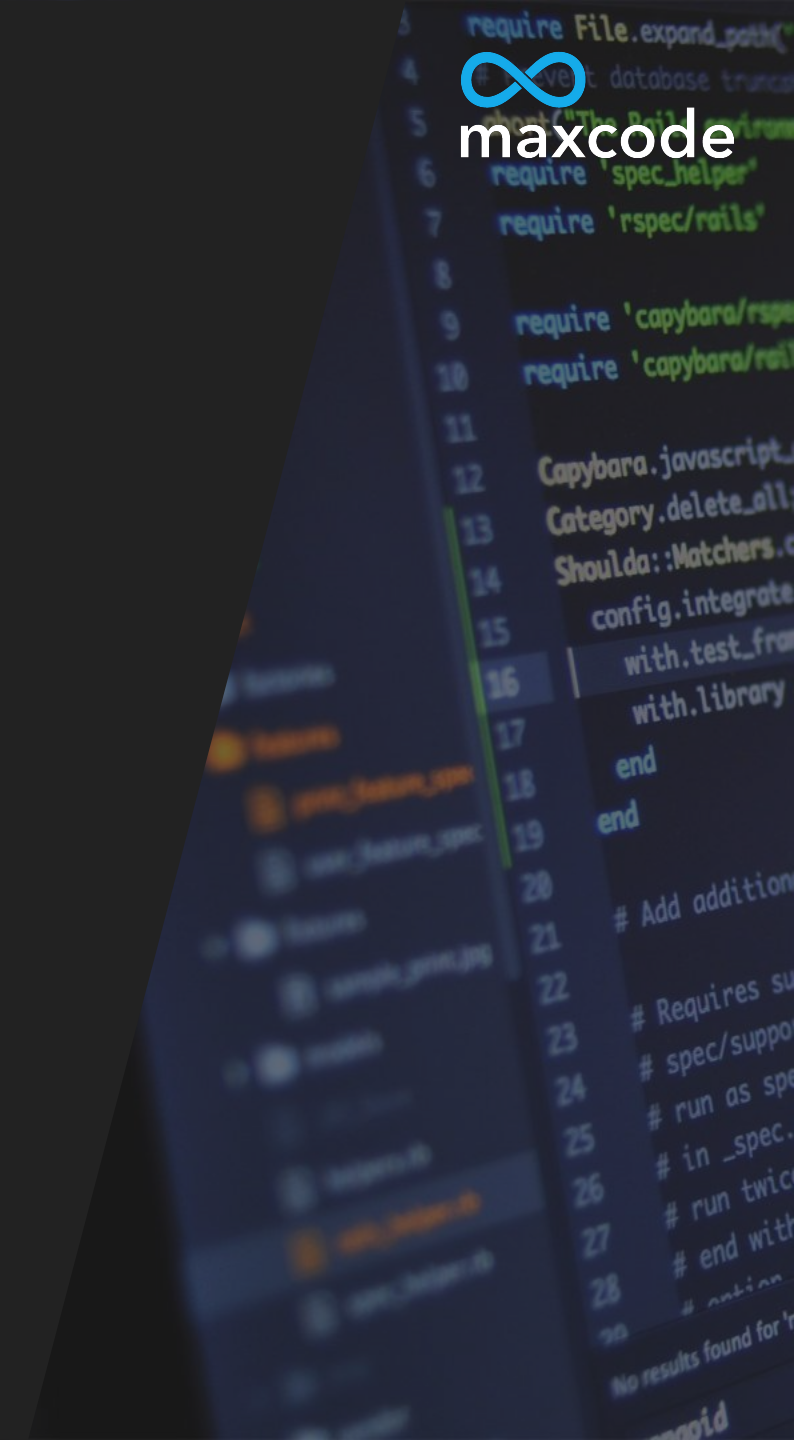

.NET
Core


aws

Disclaimer

#notsponsored

This presentation is not sponsored by AWS.



maxcode

Building

New Project

Visual Studio | Marketplace

Visual Studio > Tools > AWS Toolkit for Visual Studio 2017 and 2019

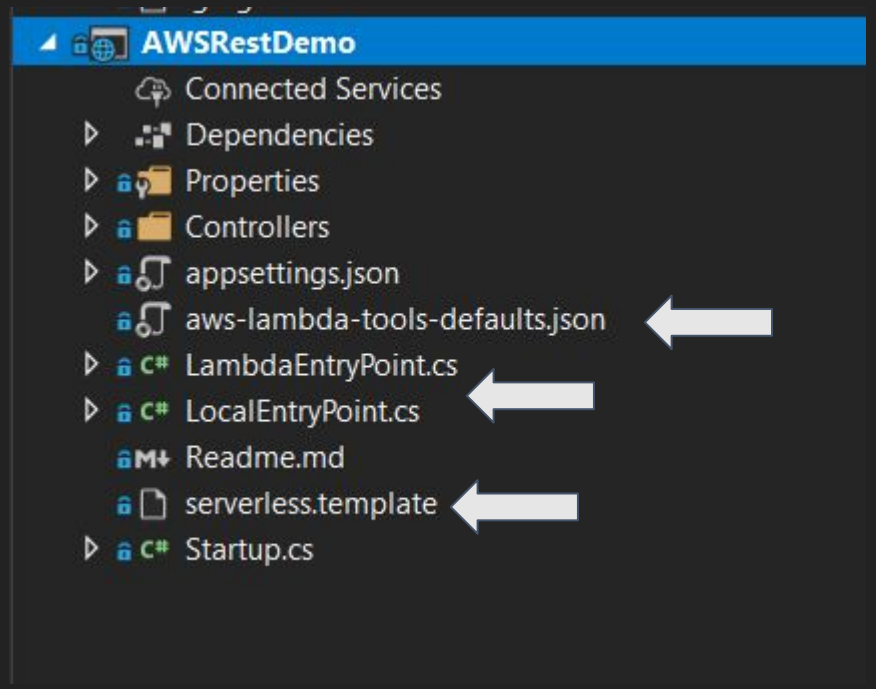
AWS Toolkit for Visual Studio 2017 and 2019

Amazon Web Services | 345,627 installs | ★★★☆☆ (62) | Free

The AWS Toolkit for Visual Studio is an extension for Microsoft Visual Studio on Windows that makes it easier for developers to develop, debug, and deploy .NET applications using Amazon Web Services. With the AWS Toolkit for Visual Studio, you'll be able to get started faster...

Download

Project differences



```
public class LambdaEntryPoint : Amazon.Lambda.AspNetCoreServer.APIGatewayProxyFunction
{
    /// <summary>
    /// The builder has configuration, logging and Amazon API Gateway already configured. The s
    /// needs to be configured in this method using the UseStartup<>() method.
    /// </summary>
    /// <param name="builder"></param>
    0 references
    protected override void Init(IWebHostBuilder builder)
    {
        builder
            .UseStartup<Startup>();
    }

    /// <summary>
    /// Use this override to customize the services registered with the IHostBuilder.
    ///
    /// It is recommended not to call ConfigureWebHostDefaults to configure the IWebHostBuilder
    /// Instead customize the IWebHostBuilder in the Init(IWebHostBuilder) overload.
    /// </summary>
    /// <param name="builder"></param>
    0 references
    protected override void Init(IHostBuilder builder)
    {
    }
}
```

maxcode

```
0 references
public class LocalEntryPoint
{
    0 references
    public static void Main(string[] args)
    {
        CreateHostBuilder(args).Build().Run();
    }

    1 reference
    public static IHostBuilder CreateHostBuilder(string[] args)
        Host.CreateDefaultBuilder(args)
            .ConfigureWebHostDefaults(webBuilder =>
            {
                webBuilder.UseStartup<Startup>();
            }); // IHostBuilder
}
```

```
0 references
public class Program
{
    0 references
    public static void Main(string[] args)
    {
        CreateHostBuilder(args).Build().Run();
    }

    1 reference
    public static IHostBuilder CreateHostBuilder(string[] args) =>
        Host.CreateDefaultBuilder(args)
            .ConfigureWebHostDefaults(webBuilder =>
            {
                webBuilder.UseStartup<Startup>();
            }); // IHostBuilder
}
```

maxcode

serverless.template

- AWS SAM templates are an extension of AWS CloudFormation templates, with some additional components that make them easier to work with
- During deploy it gets transformed to YAML

```
{
  "AWSTemplateFormatVersion":"2010-09-09",
  "Transform":"AWS::Serverless-2016-10-31",
  "Description":"An AWS Serverless Application that uses the ASP.NET Core framework running in Amazon Lambda.",
  "Parameters":{},
  "Resources":{
    "AspNetCoreFunction":... AWS::Serverless::Function ...
  },
  "Outputs":{
    "ApiURL":{
      "Description":"API endpoint URL for Prod environment",
      "Value":{
        "Fn::Sub":"https://${ServerlessRestApi}.execute-api.${AWS::Region}.amazonaws.com/Prod/swagger"
      }
    }
  }
}
```

```
"Resources":{
    "AspNetCoreFunction":{
        "Type":"AWS::Serverless::Function",
        "Properties":{
            "Handler":"AWSRestDemo::AWSRestDemo.LambdaEntryPoint::FunctionHandlerAsync",
            "Runtime":"dotnetcore3.1",
            "CodeUri":"",
            "MemorySize":256,
            "Timeout":30,
            "Role":null,
            "Policies":[
                "AWSLambdaFullAccess"
            ],
            "Events":{
                "ProxyResource":{
                    "Type":"Api",
                    "Properties":{
                        "Path":"/{proxy+}",
                        "Method":"ANY"
                    }
                },
                "RootResource":{
                    "Type":"Api",
                    "Properties":{
                        "Path":"/",
                        "Method":"ANY"
                    }
                }
            }
        }
    }
},
```

maxcode

Lambda Considerations

- Function default memory allocation - 128 MB
- Function default timeout - 3 seconds

- Function max memory allocation - 128 MB to 3,008 MB, in 64 MB increments.
- Function max timeout - 900 seconds (15 minutes)

maxcode

Deploying

maxcode

dotnet publish?

- The deploy package should contain your function's compiled assembly, all of its assembly dependencies along with:
  - proj.deps.json
  - proj.runtimeconfig.json

maxcode

Let’s pack it up!

- AWS offers the Amazon.Lambda.Tools .NET Core Global Tool which allows you to use a simple local tool to deploy and run your lambda
- dotnet tool install -g Amazon.Lambda.Tools
- dotnet lambda deploy-serverless

```
C:\Users\Iulia\source\repos\AWSRestDemo\AWSRestDemo>dotnet lambda deploy-serverless
Amazon Lambda Tools for .NET Core applications (4.2.0)
Project Home: https://github.com/aws/aws-extensions-for-dotnet-cli, https://github.com/aws/aws-lambda-dotnet

Enter CloudFormation Stack Name: (CloudFormation stack name for an AWS Serverless application)
my-rest-api-2
Enter S3 Bucket: (S3 bucket to upload the build output)
iuliadeploybucket
Processing CloudFormation resource AspNetCoreFunction
Initiate packaging of . for resource AspNetCoreFunction
Executing publish command
Deleted previous publish folder
... invoking 'dotnet publish', working folder 'C:\Users\Iulia\source\repos\AWSRestDemo\AWSRestDemo\.\bin\Release\netcoreapp3.1\publish'
... dotnet publish --output "C:\Users\Iulia\source\repos\AWSRestDemo\AWSRestDemo\.\bin\Release\netcoreapp3.1\publish" --configuration "Release" --framework "n
etcoreapp3.1" /p:GenerateRuntimeConfigurationFiles=true --runtime linux-x64 --self-contained false
... publish: Microsoft (R) Build Engine version 16.7.0+7fb82e5b2 for .NET
... publish: Copyright (C) Microsoft Corporation. All rights reserved.
... publish:   Determining projects to restore...
... publish:   Restored C:\Users\Iulia\source\repos\AWSRestDemo\AWSRestDemo\AWSRestDemo.csproj (in 398 ms).
... publish:   AWSRestDemo -> C:\Users\Iulia\source\repos\AWSRestDemo\AWSRestDemo\bin\Release\netcoreapp3.1\linux-x64\AWSRestDemo.dll
... publish:   AWSRestDemo -> C:\Users\Iulia\source\repos\AWSRestDemo\AWSRestDemo\bin\Release\netcoreapp3.1\publish\
Zipping publish folder C:\Users\Iulia\source\repos\AWSRestDemo\AWSRestDemo\.\bin\Release\netcoreapp3.1\publish to C:\Users\Iulia\AppData\Local\Temp\AspNetCore
Function-CodeUri-637392444842904242.zip
... zipping: Amazon.Lambda.APIGatewayEvents.dll
... zipping: Amazon.Lambda.ApplicationLoadBalancerEvents.dll
```

maxcode

AWS SAM

- sam package --s3-bucket my-regional-bucket --output-template-file out.yaml
- sam deploy --template-file out.yaml --capabilities CAPABILITY_IAM --stack-name MyStackName

- sam deploy

Logical ID	Physical ID	Type	Status
AspNetCoreFunction	my-api-5-AspNetCoreFunction-REBQ9Q89FH9A	AWS::Lambda::Function	CREATE_COMPLETE
AspNetCoreFunctionProxyResourcePermissionProd	my-api-5-AspNetCoreFunctionProxyResourcePermissionProd-16X9MHRCIE3AO	AWS::Lambda::Permission	CREATE_COMPLETE
AspNetCoreFunctionRole	my-api-5-AspNetCoreFunctionRole-1LOI7NY0SBY2H	AWS::IAM::Role	CREATE_COMPLETE
AspNetCoreFunctionRootResourcePermissionProd	my-api-5-AspNetCoreFunctionRootResourcePermissionProd-1LFL9871TJX3M	AWS::Lambda::Permission	CREATE_COMPLETE
ServerlessRestApi	y234wcy7ri	AWS::ApiGateway::RestApi	CREATE_COMPLETE
ServerlessRestApiDeploymentcfb7a37fc3	mk055y	AWS::ApiGateway::Deployment	CREATE_COMPLETE
ServerlessRestApiProdStage	Prod	AWS::ApiGateway::Stage	CREATE_COMPLETE

maxcode

Hosting

maxcode

One ugly URL

- The URL generated by the API Gateway will change sometimes if we change the serverless.template and redeploy
- What about exposing it on a custom domain?

maxcode

API Gateway Custom Domain

- With custom domain names, you can set up your API's hostname, to map the alternative URL to your API.
- You must have a registered internet domain name in order to set up custom domain names for your APIs.
- After a custom domain name is created in API Gateway, you must create or update your DNS provider's resource record to map to your API endpoint. Without such a mapping, API requests bound for the custom domain name cannot reach API Gateway.

Route 53

- Amazon Route 53 is a highly available and scalable cloud Domain Name System (DNS) web service.
- Designed to work with other AWS services.
- Allows you to route users directly to a AWS resource (aliases), by using their geographical location, or the latency of response for that particular user.

maxcode

Template changes

- Prerequisites: purchased a domain, have a certificate created in the region of the API Gateway
- Will need to add:
  - API Gateway Custom domain name
  - API Gateway Base Path Mapping
  - Route 53 DNS Record

maxcode

References & Repo

https://github.com/decembrya/awsrestdemo

https://docs.aws.amazon.com/lambda/latest/dg/gettingstarted-limits.html
https://github.com/aws/aws-extensions-for-dotnet-cli#aws-lambda-amazonlambdatools
https://aws.amazon.com/blogs/developer/running-serverless-asp-net-core-web-apis-with-amazon-lambda/
https://docs.aws.amazon.com/serverless-application-model/latest/developerguide/what-is-sam.html
https://docs.aws.amazon.com/serverless-application-model/latest/developerguide/sam-specification-template-anatomy.html
https://docs.aws.amazon.com/lambda/latest/dg/lambda-csharp.html
https://docs.aws.amazon.com/apigateway/latest/developerguide/how-to-custom-domains.html

Thank you!

maxcode