ChatBot Sample ASP.NET Core Web Application with Amazon Lex and AWS Lambda

The .NET ChatBot is deployed as a web application using ASP.NET Core; specifically using the ASP.NET Core MVC framework. ASP.NET Core is designed to integrate seamlessly with a variety of client-side frameworks, this application uses the Bootstrap client-side framework to build the ChatBot UI.

The ChatBot application uses Dependency Injection to add an application service for the Amazon Lex service using an interface and service class, and is configured as an ASP.NET scoped service. In addition, the ChatBot uses the ASP.NET Core configuration API to pull the needed Amazon Cognito identity pool information used for authentication to the AWS Service, Amazon Lex, as well as Lambda in order to perform validation for Amazon Lex, OrderFlowers.

What's Here

This sample includes:

- README.md - this file
- appspec.yml - this file is used by AWS CodeDeploy when deploying the web application to EC2
- buildspec.yml - this file is used by AWS CodeBuild to build the web application
- dotnetLexChatBot/ - this directory contains your ASP.NET Core application project files
- scripts/ - this directory contains scripts used by AWS CodeDeploy when installing and deploying your application on the Amazon EC2 instance

Getting Started

These directions assume you want to develop on your local computer, and not from the Amazon EC2 instance itself. If you're on the Amazon EC2 instance, the virtual environment is already set up for you, and you can start working on the code.

To work on the sample code, you'll need to clone your project's AWS CodeCommit repository to your local computer. If you haven't, do that first. You can find instructions in the AWS CodeStar user guide.

1. Install dotnet. See https://www.microsoft.com/net/core

   ❖ Version 1.1.2 was used to create this sample
     https://github.com/dotnet/core/blob/master/release-notes/download-
     archives/1.1.2-download.md

2. Build the application

3. ```$ cd AspNetCoreWebApplication```
4. ```$ dotnet restore```
5. ```$ dotnet build```
6. Run Kestrel server.

7. ```$ dotnet run```
8. Open http://localhost:5000/ in a web browser to view your web app.

CloudFormation Deployment – Windows

Step by step instruction for deploying the entire CD/CI pipeline in your AWS account that includes CodePipeline (CD/CI orchestration service), CodeBuild (which builds the code pulled from GitHub) and CodeDeploy (which takes the built package and deploys in on EC2 instance running Windows), and EC2 instance running Windows with URL of the deployed application as output.

1. Log into GitHub with your own GitHub credentials.
2. Duplicate/Mirror AWS ChatBot code repository to your GitHUb repository by following these steps https://help.github.com/articles/duplicating-a-repository/
3. Log into your AWS account, ensure that you're in use-east-1 (N. Virginia), us-west-2 (Oregon) or eu-west-1 (Ireland) region
4. Click here **Launch Stack** ▶
5. You will be presented with the following screen:

Parameters

Application

ProjectId		Project ID.
AppName		Name of the application.

GitHub

GitHubOwner		The owner of the GitHub repository.
GitHubOAuthToken		The OAuthToken of the GitHub user.
GitHubRepo		The GitHub repository.

Other parameters

InstanceType	t2.micro	The Amazon EC2 instance type to use.
KeyPairName	Search ▼	
	The name of the key pair used to make SSH connections to Amazon EC2 instances.	
OSType	ubuntu ▲▼	Operating system
SubnetId	Search by ID, or Name tag value ▼	
	The name of the VPC public subnet to use for Amazon EC2 instances launched for this project.	
VpcId	Search by ID, or Name tag value ▼	
	The ID of the Amazon Virtual Private Cloud (VPC) to use for Amazon EC2 instances.	

- Stack name and Application section values are up to you
- GitHub section will have preloaded values for owner and repo

- GitHubOAuthToken can be obtained by going to "Settings" of your GitHub account, clicking on "Personal Access Tokens" and creating new token that has the scopes as shown below.

6. Select VPC where you want the solution deployed, Public Subnet (the one that has internet gateway attached) that belongs to that VPC.
7. Select keypair name http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-key-pairs.html
8. Change instance type if necessary, otherwise leave the default value for it.
9. In OSTtype field select the desired Operating System for the target EC2 instance (Ubuntu or Windows).
10. Click "Next"
11. Leave everything as it is (on both Options and Advanced section)
12. Click "Next"
13. On the review screen make sure to tick the box next to: "**I acknowledge that AWS CloudFormation might create IAM resources with custom names."**

14. Watch the full stack and all the necessary resources get created under "Events" tab and wait until "CREATE_COMPLETE" status.
15. Take a look at CodePipeline and see your application go through the whole CD/CI process from GitHub to CodeBuild to CodeDeploy which deploys it on EC2 in the end
16. Once it's successfully deployed, go to outputs tab in CloudFormation and click on the URL value and see the ChatBot application running in your environment

Connect to Visual Studio

1. Get the AWS Toolkit for Visual Studio, you can get it here:
   https://aws.amazon.com/visualstudio/
2. Open Visual Studio. On the Getting Started page, configure your AWS credentials. (You can also do this from Team Explorer.) You can configure your keys here: https://us-west-2.console.aws.amazon.com/iam/home#/users/username?section=security_credentials

Note: Replace "username" in URL above with your user name.

3. In Team Explorer, in the list of Hosted Service Providers, choose AWS CodeCommit.
4. Choose your AWS profile from the list.
5. In Connect, choose Clone, and then choose your project's repository from the list. Choose where to create your local repo, and then choose OK.
6. If you haven't already done so, create and download Git credentials for your IAM when prompted and store them in a safe place. You can create and download them here: https://us-west-2.console.aws.amazon.com/iam/home#/users/username?section=security_credentials

Note: Replace "username" in URL above with your user name.

7. In Project Explorer, expand the tree to find your project and its files. Start working on code.
8. In Team Explorer, open Changes, and stage the files you want to commit. Change your view to Synchronization, and choose Push.

Manual Deployment – Windows

Step by step manual instruction for connecting to GitHub and deploying (via AWS console) the entire CD/CI pipeline in your AWS account that includes CodePipeline (CD/CI orchestration service), CodeBuild (which builds the code pulled from GitHub) and CodeDeploy (which takes the built package and deploys in on EC2 instance running Windows), and EC2 instance running Windows with URL of the deployed application as output.

1. Log in to GitHub https://github.com/ with your user credentials, if you don't have them, please create them and log in.
2. Log into your AWS account, ensure that you're in use-east-1 (N. Virginia), us-west-2 (Oregon) or eu-west-1 (Ireland) region.
3.

Create the EC2 Instance Role

4. Open the IAM console at https://console.aws.amazon.com/iam/.
5. In the navigation pane, choose Roles, and then choose Create new role.
6. On the Select role type page, with AWS Service Role selected, click on EC2 and click "Next: Permissions".
7. On the Attach Policy page, select the box next to the AmazonS3ReadOnlyAccess policy, and then choose Next Step.
8. In the Role name box, give the service role a name (for example, EC2InstanceRole).
9. Click "Create Role".

Create the EC2 Instance

10. Click on EC2 in the service list.
11. In the Amazon EC2 Dashboard, choose "Launch Instance" to create and configure your virtual machine.
12. In this wizard, you have the option to configure your instance features. Below are some guidelines on setting up your first instance.
    - Choose an Amazon Machine Image (AMI): In step 1 of the wizard, select the Microsoft Server 2016 AMI.
    - Choose an instance type: In step 2 of the wizard, select t2.xlarge.
    - Configure Instance Details: In step 3 of the wizard, select default VPC and default Public Subnet and select the IAM role you created in "Create the EC2 Instance Role" previously. Under "Advanced details" paste the following in "User data" field (with "as text" selected):

```
<powershell>
New-Item -Path c:\temp -ItemType "directory" -Force
powershell.exe -Command Read-S3Object -BucketName bucket-name/latest
-Key codedeploy-agent.msi -File c:\temp\codedeploy-agent.msi
Start-Process -Wait -FilePath c:\temp\codedeploy-agent.msi -
WindowStyle Hidden
</powershell>
```

Note: In line "`powershell.exe -Command Read-S3Object -BucketName bucket-name/latest -Key codedeploy-agent.msi -File c:\temp\codedeploy-agent.msi`" please replace "`bucket-name`" value just after "`-BucketName`" (just before /latest) based on the following table:

http://docs.aws.amazon.com/codedeploy/latest/userguide/resource-kit.html#resource-kit-bucket-names

- Add Storage: In step 4 of the wizard, accept defaults and click "Next: Add Tags"
- Add Tags: In step 5 of the wizard, click "Add Tag", then type "Name" for "Key" field and "ChatBot" for "Value" field. Click "Next: Configure Security Group".
- Security group: In step 6, you have the option to configure your virtual firewall. You should see RDP already there, click "Add Rule", select "HTTP" for "Type", leave everything else as default and click on "Review and Launch"
- Launch instance: In step 7, review your instance configuration and choose "Launch".
- Create a key pair: Select "Create a new key pair" and assign a name. The key pair file (.pem) will download automatically - save this in a safe place as we will later use this file to log in to the instance. Finally, choose "Launch Instances" to complete the set up.

Note: It may take a few minutes to initialize your instance.

Create the CodeDeploy Service Role

13. Open the IAM console at https://console.aws.amazon.com/iam/.
14. In the navigation pane, choose Roles, and then choose Create new role.
15. On the Select role type page, with AWS Service selected, click on CodeDeploy, and click "Next: Permissions".
16. On the Attach Policy page, select the box next to the AWSCodeDeployRole policy, and then choose Next Step.
17. In the Role name box, give the service role a name (for example, CodeDeployServiceRole).

18. Note the value of the Policies field. You will need it later when you create deployment groups. If you forget the value, follow the instructions in Get the Service Role ARN (Console) http://docs.aws.amazon.com/codedeploy/latest/userguide/getting-started-create-service-role.html#getting-started-get-service-role-console.
19. Choose "Create role".

Create the CodeDeploy deployment

20. Click on "AWS CodeDeploy" in the service list.
    - Depending on whether you used CodeDeploy before you will be presented with:
        1. "Applications" list, click on "Create application".
        2. Introduction screen, click on "Get Started Now".
            1. On the next screen select "Custom Deployment".
            2. Click "Skip Walkthrough".
21. Provide value for "Application name" and "Deployment group name".

Create application ❓

Create an application and choose a deployment type. Specify the instances to deploy to. Specify the conditions for a successful deployment.

Application name*	100 character limit
Deployment group name*	100 character limit

22. Make sure "In-place deployment" is selected under "Deployment type".
23. Under "Environment configuration" click on "Amazon EC2 instances" tab and provide "Name" and "ChatBot" values for "Key" and "Value" respectively.

Environment configuration

Specify any combination of Auto Scaling groups, Amazon EC2 instances, and on-premises instances to add instances to this deployment group.

Auto Scaling groups	**Amazon EC2 instances**	On-premises instances

You can add up to three groups of tags for EC2 instances to this deployment group. Learn more
One tag group : Any instance identified by the tag group will be deployed to.
Multiple tag groups : Only instances identified by all the tag groups will be deployed to.

Tag group 1

	Key	Value	Instances	
1	Name ▾	ChatBot ▾	1	✖
2	▾	▾		✖

Add tag group

24. You should see the instance you created earlier now listed under "Matching instances".

25. Leave everything as it is (default).

26. Under "Service role" select under "Service role ARN" the IAM role you created in "Create the CodeDeploy Service Role" section.

27. Click "Create application".

Create the CodePipeline deployment

28. Click on "AWS CodePipeline" in the service list.

    - If you never used it before, you'll see a "Get Started" screen, click on "Get Started" button.
    - If you have used it, click on "Create Pipeline"

29. In Step 1 – Name, provide "Pipeline Name" and click "Next Step"

30. In Step 2 – Source, under "Source Provider" select GitHub

31. Click on "Connect to GitHub"

32. Once connected, select the "Repository" and "Branch" where ChatBot .NET Core Code is located.

33. Leave "Advanced" section as it is.

34. In Step 3 – Build, under "Build Provider" select CodeDeploy.

35. Under "Configure your project", select a "Create a new Build Project"

36. Provide a "Project Name".

37. Under "Environment: How to build" make sure "Use an image managed by AWS CodeBuild" is selected.

38. For "Operating System" select "Windows"

39. Next for "Runtime" select ".Net Core"

40. For "Version" select "aws/codebuild/dot-net:code-1"

41. Make sure, under "Build Specification" that "Use the buildspec.yml in the source code root directory" is selected.

42. Click "Save Build Project" and you should get this message when Save actions is complete:

> ❶ **New build project saved**   ✖
>
> Your build project settings have been saved. After the pipeline starts running and AWS CodeBuild starts your build, you can get additional build details and logs through the AWS CodeBuild console.

43. Click "Next Step".

44. In Step 4 – Deploy, under "Deployment provider" select "AWS CodeDeploy".

45. Under "AWS CodeDeploy" section select "Application name" and "Deployment Group" that you created earlier in "Create the CodeDeploy deployment" section.

46. Click "Next stop".

47. In Step 5 – Service Role, under "AWS Service Role" click "Create Role".

48. In the screen that pops up titled "AWS CodePipeline is requesting permission to use resources in your account" just click "Allow" button at the very bottom (right) of the page.

49. You should see "Role name" field now populated with new role created (eg. "AWS-CodePipeline-Service").

50. Click "Next step".

51. You will be presented with the review of your pipeline, click "Create Pipeline" at the bottom of the page.

52. Now you can watch your CodePipeline build and deploy

Source

Source
GitHub

✅ **Succeeded** just now
fb32236

⚙ Source: adding linux stuff

↓

Build

CodeBuild
AWS CodeBuild

🔄 **In Progress** just now
Details

⚙ Source: adding linux stuff

↓

Staging

manualDeploymentGroup
AWS CodeDeploy

No executions yet

53. In the Description tab of the EC2 instance deployed find the value of the Public DNS (IPv4), and paste it in the browser.

It will look something like this:

`ec2-11-111-11-111.us-west-2.compute.amazonaws.com`
*Note that `11-111-11-111` is the same as the public IP of your instance `11.111.11.111`