Orange Networks GmbH
Sachsentor 26
21029 Hamburg

orange networks
IT-CONSULTING - IT-DEVELOPMENT

Developing Applications - Bots

Scenario 01: Bot Overview
Lab Manual

Lab 1: Bot Overview

Introduction

In this lab, you will create a simple Bot with Bot Framework and Azure Bot Service:

- Explore the available features and functionality.

Objectives

After completing this lab, you will be able to:

- Create a Bot with Azure Bot Service
- Deploy to Azure

Prerequisites

- Browser
- Access to Azure Subscription

Estimated Time to Complete This Lab

30 minutes

For More Information

Create a bot with the Azure Bot Service: https://docs.microsoft.com/en-us/botframework/azure-bot-service-quickstart
Create a bot with the Bot Builder SDK for .NET: https://docs.microsoft.com/en-us/azure/bot-service/dotnet/bot-builder-dotnet-quickstart
Create a bot with the Bot Builder SDK for Node.js: https://docs.microsoft.com/en-us/azure/bot-service/nodejs/bot-builder-nodejs-quickstart?view=azure-bot-service-3.0

Excercise 1: Create a bot with the Azure Bot Service

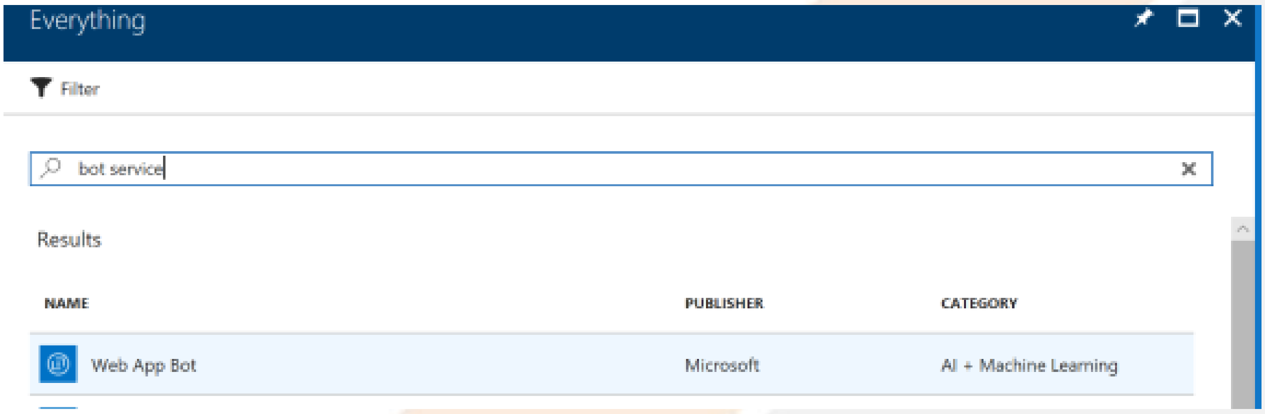
Introduction
After completing this exercise, you will have a simple echo bot deployed in Azure

Objectives
After completing this lab, you will be able to:
- Create bots using the Bot Service
- Test the bot using the Bot Service test tool

Prerequisites
None

Scenario
This will be the starting steps for creating many of your bots.

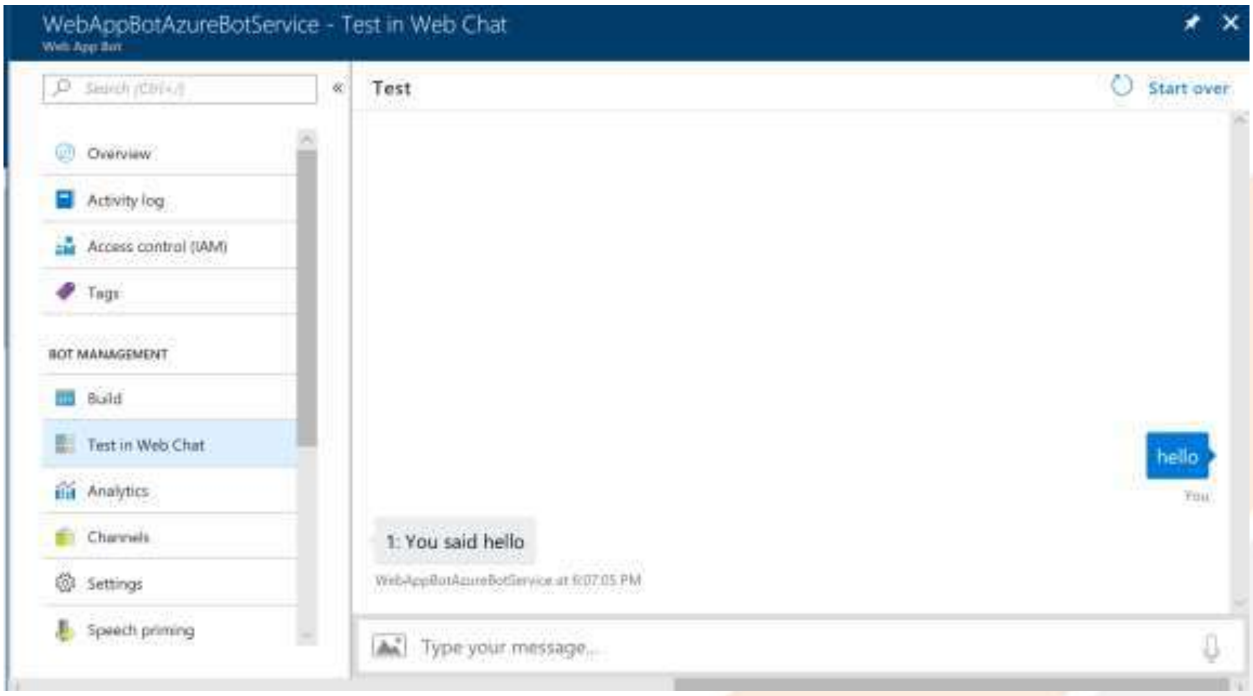
Task 1: Create a bot with the Azure Bot Service
1. Open the Azure portal
2. Start the Bot Service wizard by adding a new Web App Bot and follow the steps to create a Basic NodeJs bot



NAME	PUBLISHER	CATEGORY
Web App Bot	Microsoft	AI + Machine Learning

Task 2: Test with Online Test Tool

1. From the test in webchat tab, send “hello” to the new bot!



hello
You

1: You said hello
WebAppBotAzureBotService at 6:07:05 PM

Exercise 2: Create a bot with the Bot Builder SDK

Introduction
After completing this exercise, you will have a simple echo bot deployed in Azure

Objectives
After completing this lab, you will be able to:
- Create bots using the Bot Builder SDK
- Test and debug the bot using the Bot Framework Emulator and Visual Studio

Prerequisites
- NodeJs (https://nodejs.org/en/)
- Visual Studio Code (https://code.visualstudio.com/)
- Bot Framework Emulator (download available here https://github.com/Microsoft/BotFramework-Emulator/releases)

Scenario
This will be the starting steps for creating many of your bots.

Task 1: Create a bot with the Azure Bot Builder SDK
1. Create a folder for your bot
2. Open Visual Studio Code
3. Open your bot folder in Visual Studio Code
4. Open the Integrated Terminal in Visual Studio Code (View->Terminal)
5. Initialize NodeJs Application and create package.json by running the following command:

   npm init

6. Install the Bot Builder SDK for Node.js by running the following **npm** command:

   npm install --save botbuilder

7. To use any of the Bot Framework channels (or run your bot in the emulator), your bot needs to run an API endpoint. Install restify by running the following **npm** command:

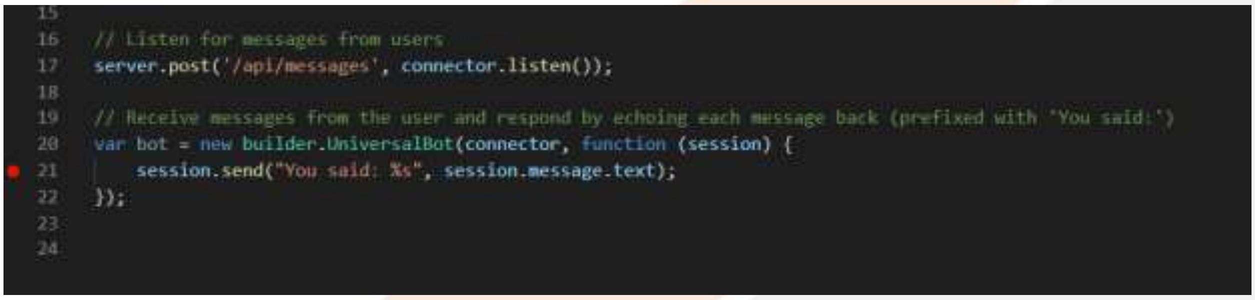
   npm install --save restify

8. Add a new file **app.js** to your folder. Add the following code to this file:

```
var restify = require('restify');
var builder = require('botbuilder');

// Setup Restify Server
var server = restify.createServer();
server.listen(process.env.port || process.env.PORT || 3978, function () {
   console.log('%s listening to %s', server.name, server.url);
});

// Create chat connector for communicating with the Bot Framework Service
var connector = new builder.ChatConnector({
    appId: process.env.MicrosoftAppId,
    appPassword: process.env.MicrosoftAppPassword
});

// Listen for messages from users
server.post('/api/messages', connector.listen());

// Receive messages from the user and respond by echoing each message back
(prefixed with 'You said:')
var bot = new builder.UniversalBot(connector, function (session) {
    session.send("You said: %s", session.message.text);
      });



Save the file. Now you are ready to test your bot in the emulator.
```

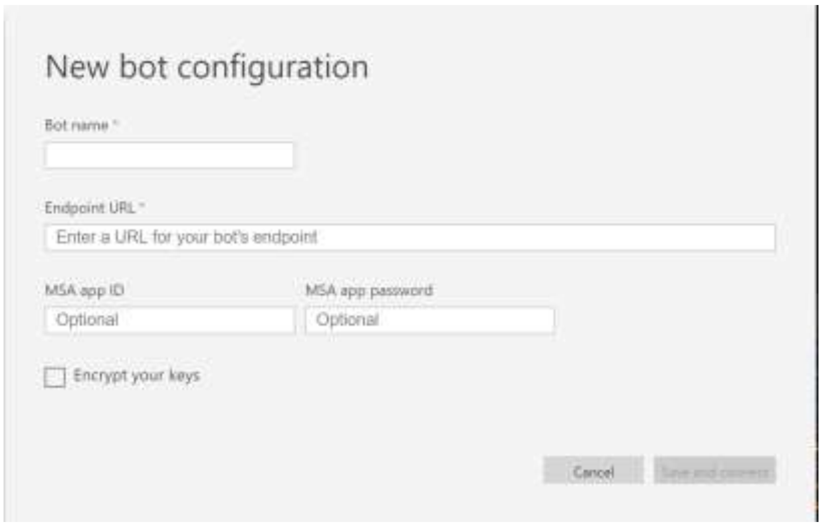
Task 2: Debug using Visual Studio

1. Start debugging in Visual Studio Code and set a breakpoint like the one shown in this picture inside the **app.js** file



```
// Listen for messages from users
server.post('/api/messages', connector.listen());

// Receive messages from the user and respond by echoing each message back (prefixed with 'You said:')
var bot = new builder.UniversalBot(connector, function (session) {
    session.send("You said: %s", session.message.text);
});
```

1. Start your application in debug mode.
2. Copy the URL of your application
3. Launch the Bot framework emulator
4. Under File-> New Bot enter a name and paste the URL and add /api/messages as a suffix.
5. Click on Save and connect



New bot configuration

Bot name *

Endpoint URL *
Enter a URL for your bot's endpoint

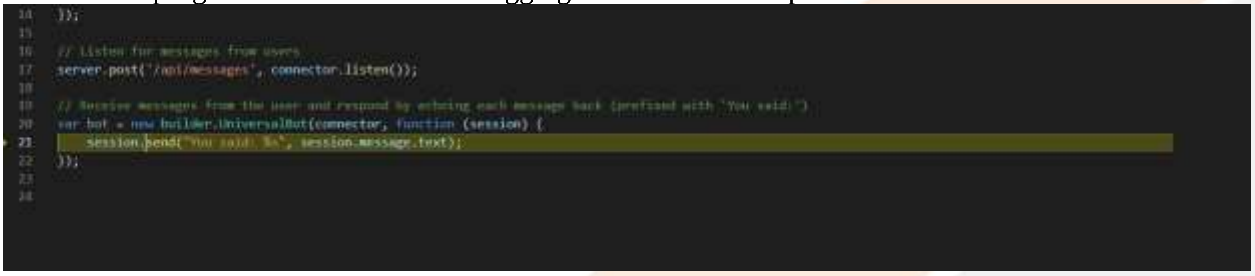
MSA app ID
Optional

MSA app password
Optional

☐ Encrypt your keys

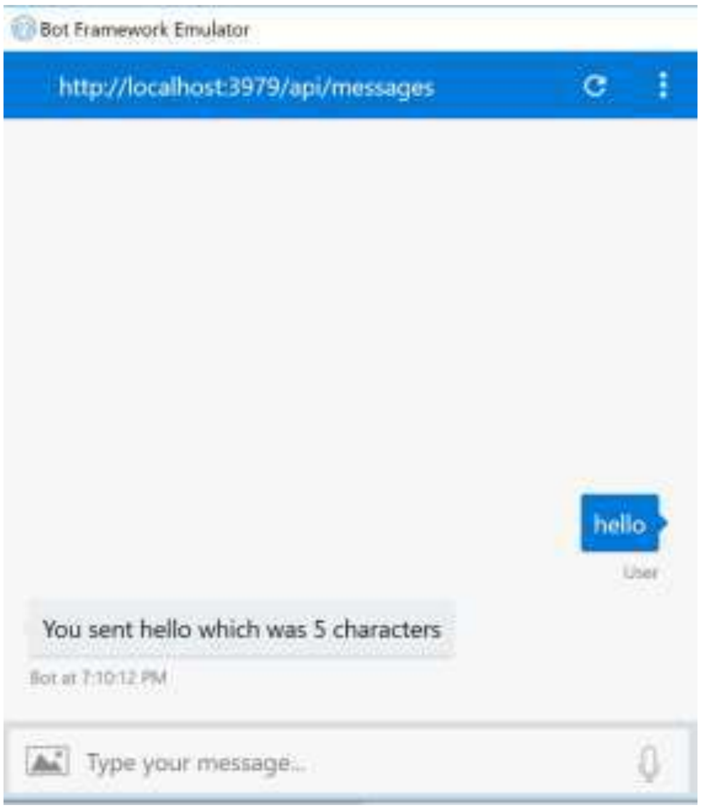
6. Type “hello” in the “Type you message…” input box and press Enter to send the message

7. The program should break in debugging mode at the breakpoint



```
});

// Listen for messages from users
server.post('/api/messages', connector.listen());

// Receive messages from the user and respond by echoing each message back (prefixed with 'You said:')
var bot = new builder.UniversalBot(connector, function (session) {
    session.send("You said: %s", session.message.text);
});
```

8. After continuing execution, you will see the following in the Bot Framework Emulator



hello

User

You sent hello which was 5 characters

Bot at 7:10:12 PM