

Testing Bots

Mithun Prasad, PhD
miprasad@Microsoft.com



Testing Bots

- What Should We Test?
- How do you test a bot?
- Mocking

Unit/Functional/Load Testing

- Channel Testing

Unit Tests



Unit Tests

- Mock out the Bot Framework
- Perform Unit Testing by reusing the Bot Builder code
 - DialogTestBase.cs
 - FiberTestBase.cs
 - MockConnectorFactory.cs
- Runs locally

What is Mocking?

Simulate the behaviour of real objects

- Used in unit testing
- Used when the unit being tested has external dependencies
- To isolate the behaviour of object you want to test, the other objects will be replaced by mocks that simulate the behaviour of the real objects
- Useful if the real objects are impractical to incorporate into the unit test
- Moq for Bots - mocking framework for .NET

Testing Channels



Testing Channels

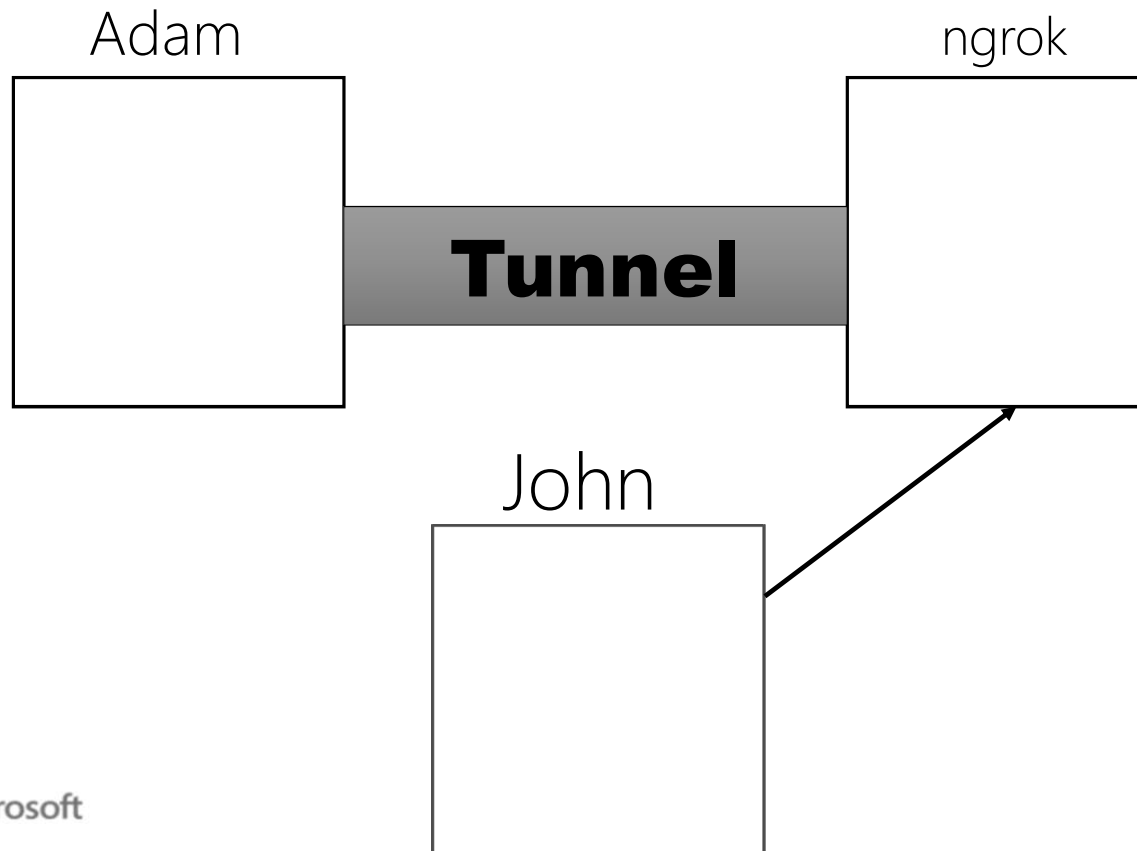
- Host the Bot service on a public URL endpoint
- When designing / building / testing your code, you don't always want to have to redeploy
- Redeploying means paying hosting costs

Ngrok

Secure tunnels to localhost

"I want to expose a local server behind a NAT or firewall to the internet."

What is Ngrok?



Forwarding

ngrok http 1111

```
ngrok                                     <Ctrl+C to quit>
Tunnel Status                           online
Version                                 1.6/1.5
Forwarding                               http://6125081c.ngrok.com -> 127.0.0.1:1111
Forwarding                               https://6125081c.ngrok.com -> 127.0.0.1:1111
Web Interface                            127.0.0.1:4040
# Conn                                   22
Avg Conn Time                            16.86ms
```

Forwarding

- Given the bot is being hosted on localhost:3979, use ngrok to expose it to public internet
- `ngrok.exe http 3979 -host-header="localhost:3979"`
- The forwarding url is the public endpoint for the bot service

ngrok by @inconsreveable

```
Session Status      online
Version            2.2.4
Region             United States (us)
Web Interface       http://127.0.0.1:4040
Forwarding          http://542c7f9f.ngrok.io -> localhost:3979
Forwarding          https://542c7f9f.ngrok.io -> localhost:3979

Connections        ttl    opn    rt1    rt5    p50    p90
                   5      0      0.00   0.01   4.34   42.74

HTTP Requests
-----
POST /api/messages  200 OK
POST /api/messages  200 OK
POST /api/messages  500 Internal Server Error
POST /api/messages  200 OK
POST /api/messages  200 OK
```

Testing Emulator on Public Endpoints

- The forwarding public urls from ngrok can be used in the emulator as shown below

Local Port	Emulator Url	Bot Url
9000	https://e4c7108d.ngrok.io/	http://542c7f9f.ngrok.io/api/messages

User: Cor

Testing Emulator on Public Endpoints

- The forwarding public urls from ngrok can be used in the emulator as shown below

Local Port	Emulator Url	Bot Url
9000	https://e4c7108d.ngrok.io/	http://542c7f9f.ngrok.io/api/messages

User: Cor

Direct Line



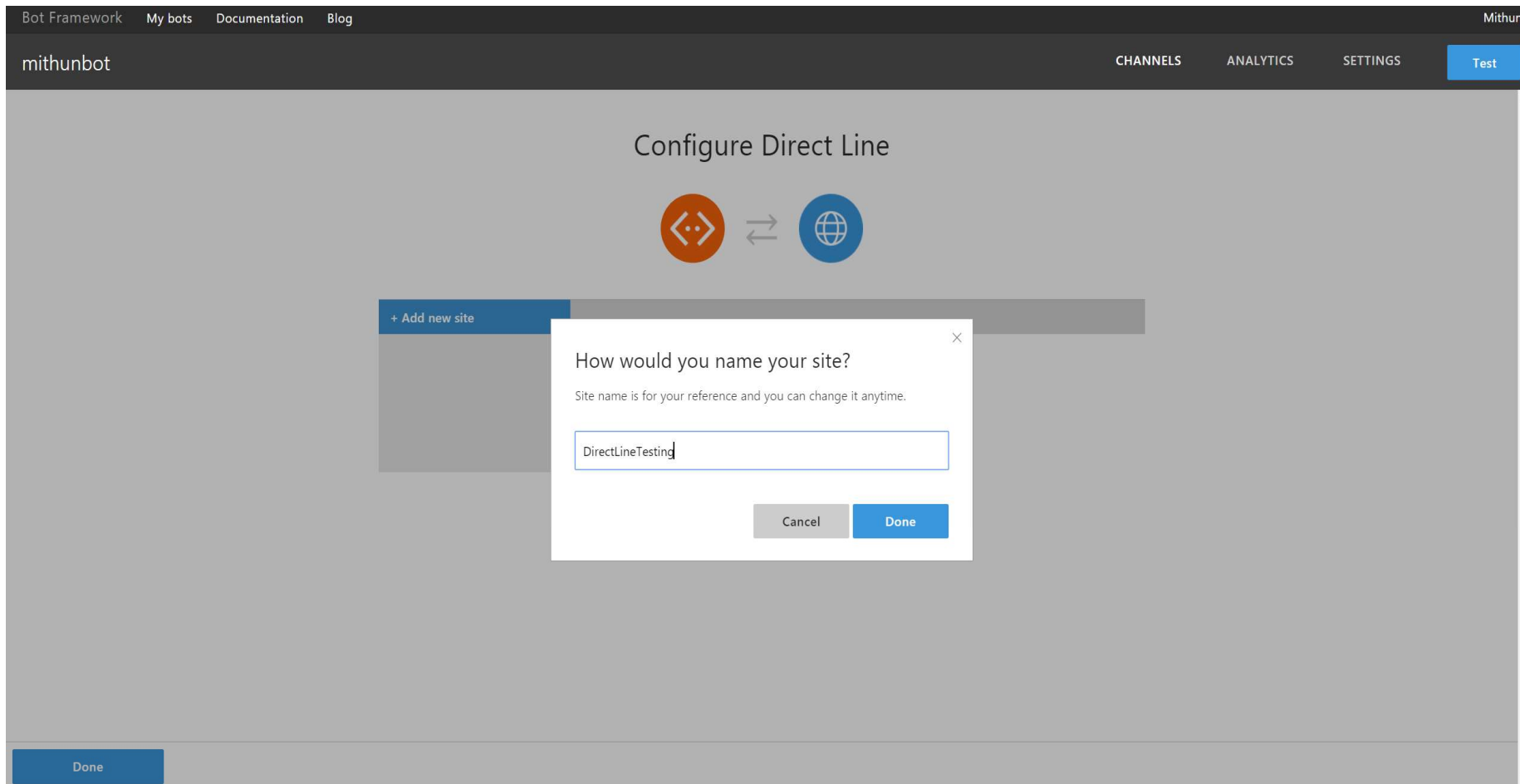
Talk Directly

- A rich list of channels are supported. But what do you do when the channels provided aren't quite enough?
- *Direct Line!*
- Direct Line is a REST API for the bot framework that allows it's users to create their own integrations
- Example: If you have a mobile app that you directly want to integrate a chat bot into. You cannot make your app a channel!

Authentication

- Call the Bot directly from a custom application to test authentication
- Allows you to authenticate a user in your application and securely communicate with the Bot as that user
- Direct Line API requests can be authenticated either by using a secret that you obtain from the Direct Line channel configuration page in the Bot Framework Portal

Direct Line Configuration





Direct Line Configuration

[Bot Framework](#) [My bots](#) [Documentation](#) [Blog](#) Mithun


mithunbot CHANNELS ANALYTICS SETTINGS Test


Configure Direct Line



+ Add new site

DirectLineTesting

DirectLineTesting 

Disable 

Secret keys

dLIRPIK_37l.cwA.xEg.cfBMU9cp1AL5Tt5eJsJvbC-KUANas9860_zxPI2NU_

[Hide](#) [Regenerate](#)

dLIRPIK_37l.cwA.vXE.W2snA75Da1JYnBq81Bi87w1LBduhJq2_uUTy1uEl

[Hide](#) [Regenerate](#)

Version


Select which versions of the Direct Line protocol are enabled on this site. More information about these versions can be found in the [Direct Line reference documentation](#).

☒ 1.1

☒ 3.0 [PREVIEW]

☐ High-speed storage [PREVIEW]

Done

 Microsoft

Start a Conversation

The screenshot displays the ARC REST client interface. On the left, a sidebar contains navigation options: HTTP request (selected), Socket, History, Saved, and Projects. The main panel is titled 'Request' and shows the URL `https://directline.botframework.com/api/conversations`. The HTTP method is set to POST. Under the 'Raw headers' tab, the 'Authorization' header is configured with the value `Bearer dLIRPiK_37I.cwA.xEg.cfBMU9cp1AL5Tt5ejsJvbc-KUANas9860_zxP12NU_s`. A warning message states 'Content-Type header is not defined'. The 'Raw payload' tab is active, showing an empty text area. Below the payload area, the response status is '200 OK' with a response time of '698.00 ms'. At the bottom, the 'JSON' tab displays the response body as a JSON object:

```
{
  "conversationId": "COR95mdLIUIGNaKidxXj2g",
  "token": "dLIRPiK_37I.dAA.QwAwAFIAOQA1AG0AZABMAEKAVQBJAEcAbgBhAEsAaQBkAHgAWABqADIAZwA.E3g1Cp_w0gE.vEcqo0bBVAA.TFJ5P0w6Ikj9Oztks8iSPeyCNZj6Ao5SZmT8ZpH1hk0",
  "expires_in": 1800
}
```

The bottom status bar indicates 'Selected environment: default'.

Direct Line Concepts

- Starting a conversation
- Sending messages
- Receiving messages
- End conversation