

Azure Web Chat Bot ARM Template

Here we continue our adventure through ARM Templates. The aim of this Get Start is to create a Web App Service and deploy a Web Chat Bot using Azure Services.

But let's understand a bit better how all this will work.

Azure Bot Service

Azure Bot Service and Bot Framework provide tools to build, test, deploy, and manage intelligent bots all in one place. The creation of a Bot using Azure goes beyond just the deployment of our ARM Template. Here we gonna scratch just the surface of what is offered in Azure. Through the use of modular and extensible framework provided by the SDK, tools, templates, and AI services, developers can create bots that use speech, understand natural language, handle questions and answers, and much more.

Azure Bot Service and Bot Framework offer an integrated set of tools and services to facilitate the creation of Bots. You can choose your favorite development environment or command-line tools to create your bot. SDKs exist for C#, JavaScript, and Typescript. (SDKs for Java and Python are under development.) Plus, Azure Bot Service and Bot Framework, provide tools for various stages of bot development to help you design and build bots. From **Plan** to -> **Build** to -> **Test** to -> **Publish** to -> **Connect** and **Evaluate**.

Here some extra documentation if you got hooked by Web ChatBots: [Chat Bot Documentation](#).

It's fair to guess if you came to this Tutorial, you already know what a bot means, but let's just cover some basics:

Bots provide an experience that feels less like using a computer and more like dealing with a person - or at least an intelligent robot. They can be used to shift simple, repetitive tasks, such as making a dinner reservation or gathering profile information, on to automated systems that may no longer require direct human intervention. Users converse with a bot using text, interactive cards, and speech. A bot interaction can be a quick question and answer, or it can be a sophisticated conversation that intelligently provides access to services.

How bots work

A bot is an app that users interact within a conversational way, using text, graphics (such as cards or images), or speech. Every interaction between the user and the bot generates an activity. The Bot Framework Service, which is a component of the Azure Bot Service, sends information between the user's bot-connected app (such as Facebook, Skype, Slack, MS Team and Web - the service we gonna use for this bot - which we call the channel) and the bot. We also gonna need an App Registration where we gonna generate our credentials in order to connect the bot with its channel.

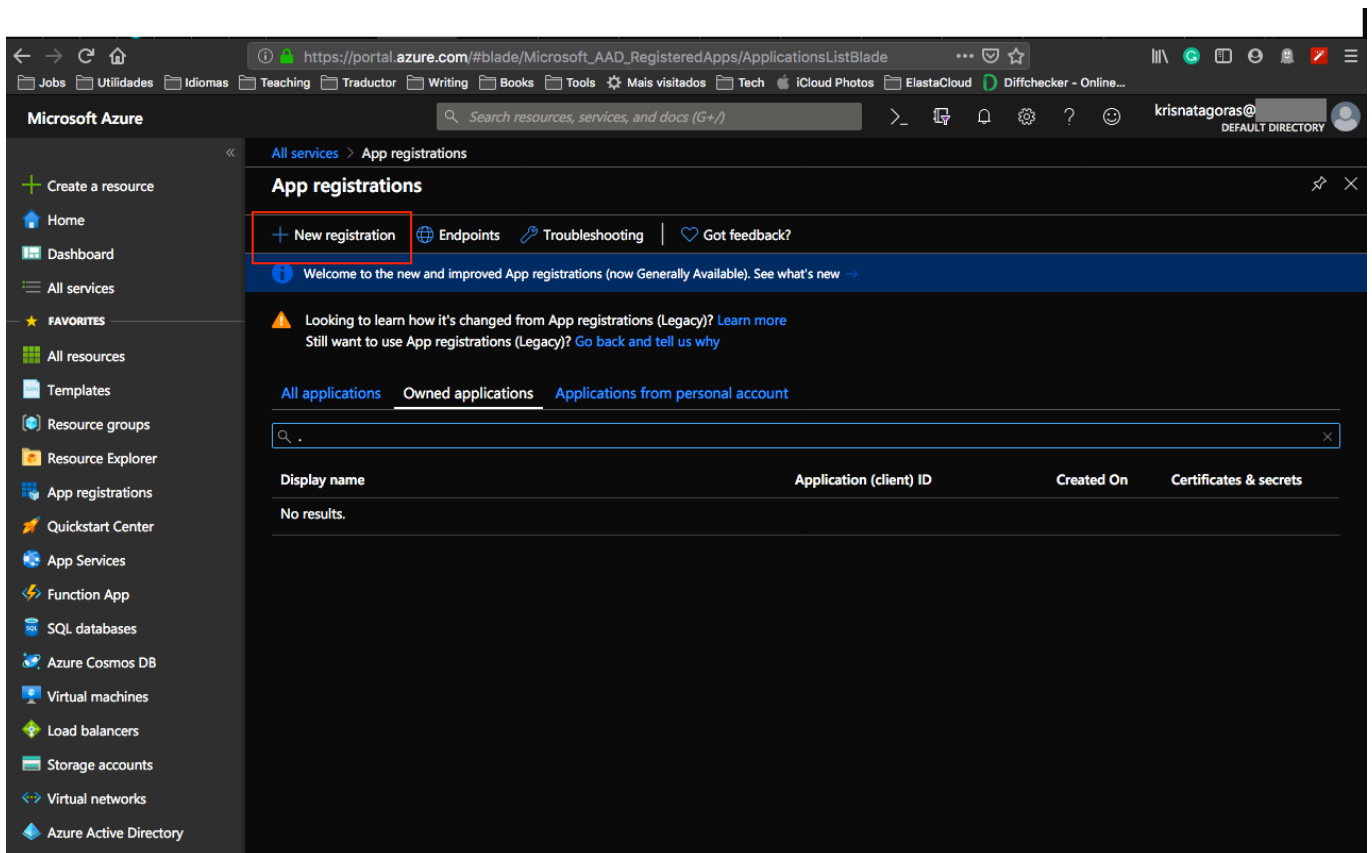
Unfortunately, create credentials aren't yet supported by ARM Templates, so that is a Requisite for our ARM Chat Bot Deployment and we need to do that using the Azure Portal.

Prerequisites:

App Registration

To create your own App ID, follow the steps below.

1. Sign in to the [Azure Portal](#) using your Azure Student Account.
2. Go to **All Services** and search for **app registrations**. You can favorite this service by clicking on the star.



3. Go to the app registrations blade and click **New registration** in the action bar at the top.

Microsoft Azure

Search resources, services, and docs (G+/)

All services > App registrations > Register an application

Register an application

*** Name**
The user-facing display name for this application (this can be changed later).

uks-mychatbot

Supported account types
Who can use this application or access this API?

☐ Accounts in this organizational directory only (Default Directory only - Single tenant)

☐ Accounts in any organizational directory (Any Azure AD directory - Multitenant)

☒ Accounts in any organizational directory (Any Azure AD directory - Multitenant) and personal Microsoft accounts (e.g. Skype, Xbox)

[Help me choose...](#)

Redirect URI (optional)
We'll return the authentication response to this URI after successfully authenticating the user. Providing this now is optional and it can be changed later, but a value is required for most authentication scenarios.

By proceeding, you agree to the [Microsoft Platform Policies](#)

Register

- Enter a display name for the application registration in the Name field and select the supported account types. The name does not have to match the bot ID.

Important

In the Supported account types, select the Accounts in any organizational directory and personal Microsoft accounts (e.g. Skype, Xbox, Outlook.com) radio button. If any of the other options are selected, **bot creation will fail**.

- Click **Register**. After a few moments, the newly created app registration should open a blade. Copy the Application (client) ID in the Overview blade and paste it into the App ID field. (That is a parameter need it in order to deploy our ARM Template)

The screenshot shows the Microsoft Azure portal interface. The left sidebar contains navigation options like 'Create a resource', 'Home', 'Dashboard', 'All services', and 'FAVORITES'. The main area displays the 'uks-mychatbot' app registration details. The 'Overview' tab is active, showing fields for 'Display name', 'Application (client) ID' (highlighted with a red box), 'Directory (tenant) ID', and 'Object ID'. A 'Call APIs' section is also visible, along with a 'Documentation' link.

Now we gonna need to generate a secret for the App Registration:

- Click on **Certificates & secrets** in the left navigation column of your app registration's blade.
- In that blade, click the **New client secret** button. In the dialog that pops up, enter an optional description for the secret and select **Never** from the Expires radio button group.

The screenshot shows the 'uks-mychatbot - Certificates & secrets' blade in the Azure portal. The 'Certificates & secrets' tab is active. A message at the top states 'Update application credentials' and 'Successfully updated application uks-mychatbot credentials'. Below this, there's a section for 'Certificates' with an 'Upload certificate' button. A table shows no certificates have been added. The 'Client secrets' section is also visible, with a 'New client secret' button highlighted. A table below shows a single client secret with the description 'uks-mysecret', an expiration date of '31/12/2299', and a value starting with 'Wwh...'.

Note

The secret will only be visible while on this blade, and you won't be able to retrieve it after you leave that page. Be sure to copy it somewhere safe.

Now that you create your credentials, let's have a look at the Template.

The Template

Don't let the size of the template scares you. The structure is very intuitive and once that you get the gist of it, you gonna see how easier your life will be regarding creating resources on Azure.

The parameters we will manipulate and inform are:

Parameter	Suggested value	Description
Subscription	Your subscription	Select your Azure Student Subscription.
Resource Group	WebAppResourceGroup	You can create a new resource group or choose from an existing one.
Location	The default location	Select the geographic location for your resource group. Your location choice can be any location listed, though it's often best to choose a location closest to your customer. The location cannot be changed once the bot is created.
Web App Name	<i>location-name-enviroment</i> i.e.: uks-mybot-test	The unique URL name of the bot. For example, if you name your bot uks-mybot-test, then your bot's URL will be <code>http://uks-mybot-test.azurewebsites.net</code> . The name must use alphanumeric and underscore characters only. There is a 35 character limit to this field. The App name cannot be changed once the bot is created. I personally like to add the location of the app into the name, and also the environment. That is the reason that my Web App Name is: uks-mybot-test
App ID	Your App ID	The App ID that you've created in the previous section.
App Secret	Complex Password	The secret that you've created in the previous section.

Deployment

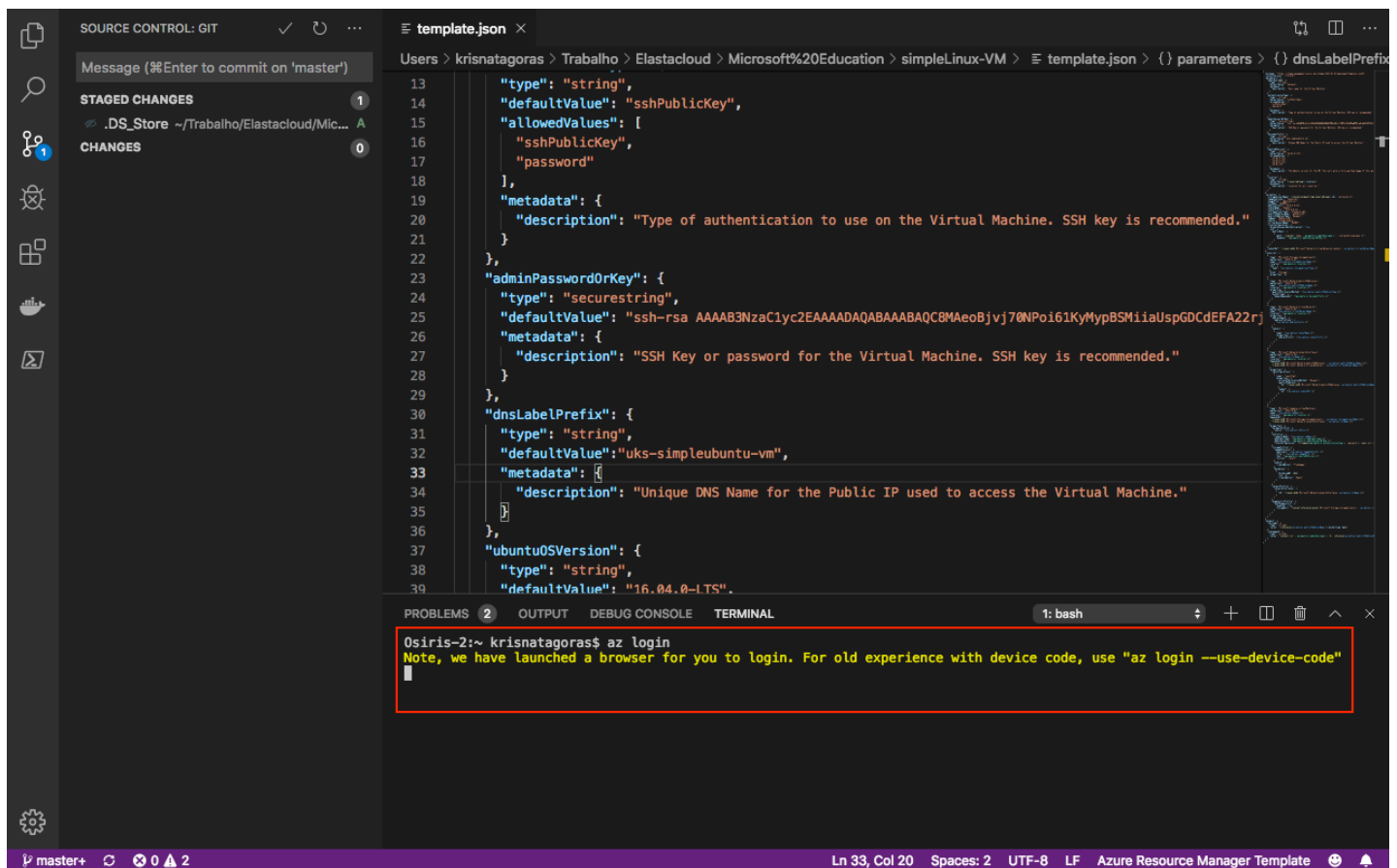
There are a few ways to deploy this template. You can use [PowerShell](#), [Azure CLI](#), [Azure Portal](#) or your favorite SDK.

For this task, we gonna deploy using Visual Code and the portal and a little surprise for you at the end. :D

For Azure CLI I choose to use the Visual Code with Azure CLI extensions, if you like, you can find more information [here](#). But bare in mind that you don't need to use the Visual Code, you can stick with the old good always present **Command Line** on Windows or any **bash terminal**.

Using Azure CLI with Visual Code

type on the terminal windows: **az login**



```
13  "type": "string",
14  "defaultValue": "sshPublicKey",
15  "allowedValues": [
16    "sshPublicKey",
17    "password"
18  ],
19  "metadata": {
20    "description": "Type of authentication to use on the Virtual Machine. SSH key is recommended."
21  },
22 },
23 "adminPasswordOrKey": {
24   "type": "securestring",
25   "defaultValue": "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCA8MAeoBjv770Npoi61KyMypBSMiiaUspGDCdEFA22rj
26   "metadata": {
27     "description": "SSH Key or password for the Virtual Machine. SSH key is recommended."
28   },
29 },
30 "dnsLabelPrefix": {
31   "type": "string",
32   "defaultValue": "uks-simpleubuntu-vm",
33   "metadata": {
34     "description": "Unique DNS Name for the Public IP used to access the Virtual Machine."
35   },
36 },
37 "ubuntuOSVersion": {
38   "type": "string",
39   "defaultValue": "16.04.0-LTS".
```

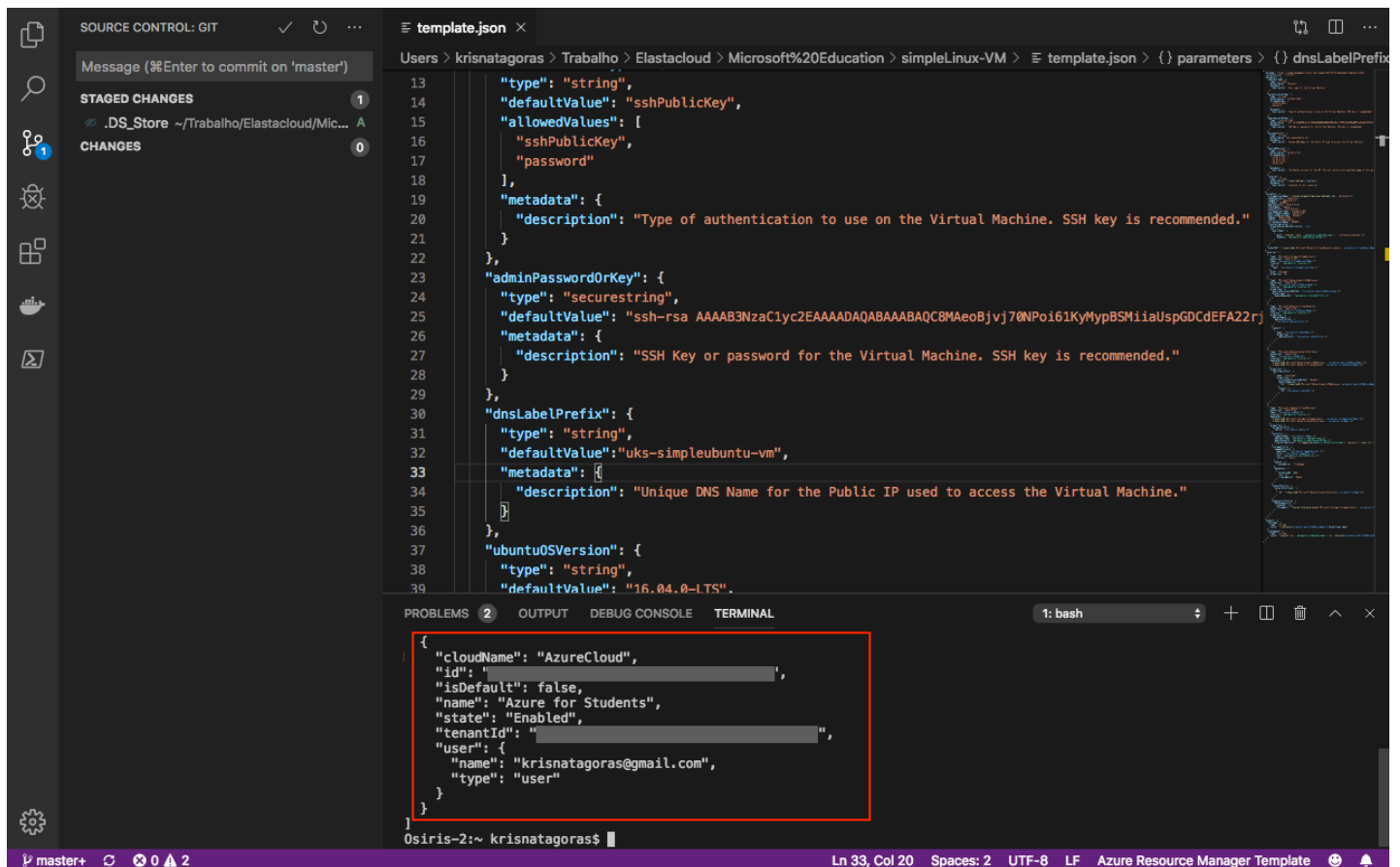
```
Osiris-2:~ krisnatagoras$ az login
Note, we have launched a browser for you to login. For old experience with device code, use "az login --use-device-code"
```

You gonna be redirected to the Azure Portal where you can use your credentials to login into.

After login, you gonna have your credentials.

In order to set the right subscription, you can use the following command:

az account set --subscription "< your subscription id >"



Resource Group

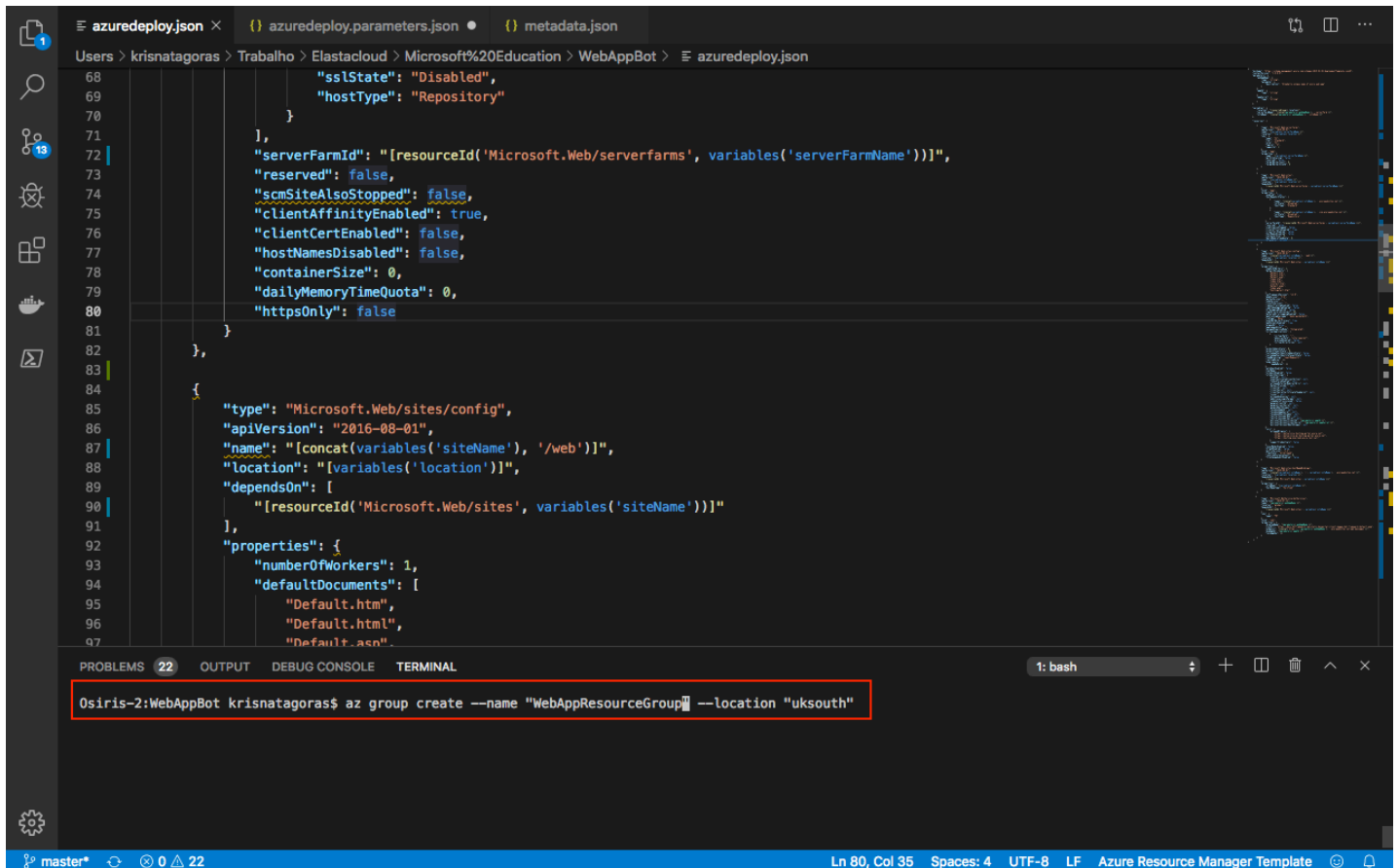
After you logged in, we gonna need to create a Resource Group for our deployment. If you haven't yet created a Resource Group, we gonna do that now! But what is a Resource Group, one might ask. Bare with me! A Resource Group is a container that holds related resources for an Azure solution. The resource group includes those resources that you want to manage as a group. Simply saying, it's like a folder that contains files. Simple as that ;-)

To create a Resource Group, you need a name and the location for your Resource Group.

For a list of locations, type: **az account list-locations**

To create the Resource group, just type the command:

az group create --name < mygroupname> --location < yourlocation >



```
68         "sslState": "Disabled",
69         "hostType": "Repository"
70     },
71     ],
72     "serverFarmId": "[resourceId('Microsoft.Web/serverfarms', variables('serverFarmName'))]",
73     "reserved": false,
74     "scmSiteAlsoStopped": false,
75     "clientAffinityEnabled": true,
76     "clientCertEnabled": false,
77     "hostNamesDisabled": false,
78     "containerSize": 0,
79     "dailyMemoryTimeQuota": 0,
80     "httpsOnly": false
81 },
82 },
83 {
84     "type": "Microsoft.Web/sites/config",
85     "apiVersion": "2016-08-01",
86     "name": "[concat(variables('siteName'), '/web')]",
87     "location": "[variables('location')]",
88     "dependsOn": [
89         "[resourceId('Microsoft.Web/sites', variables('siteName'))]"
90     ],
91     "properties": {
92         "numberOfWorkers": 1,
93         "defaultDocuments": [
94             "Default.htm",
95             "Default.html",
96             "Default.asp"
97         ]
98     }
99 }
```

PROBLEMS (22) OUTPUT DEBUG CONSOLE TERMINAL

1: bash

```
Osiris-2:WebAppBot krisnatagoras$ az group create --name "WebAppResourceGroup" --location "uksouth"
```

Ln 80, Col 35 Spaces: 4 UTF-8 LF Azure Resource Manager Template

Super simple, right? Now that we have our **Resource Group** created, let's deploy our Virtual Machine.

az group deployment create --name "name of your deployment" --resource-group "The group you created" --template-file "./azuredeploy.json"

The screenshot shows the Visual Studio Code editor with the `azuredeploy.json` file open. The file is a JSON template for an Azure Web App. The terminal at the bottom shows the command: `az group deployment create --name mydep --resource-group "WebAppResourceGroup" --template-file ".\azuredeploy.json"`.

```
68      "sslState": "Disabled",
69      "hostType": "Repository"
70    },
71    ],
72    "serverFarmId": "[resourceId('Microsoft.Web/serverfarms', variables('serverFarmName'))]",
73    "reserved": false,
74    "scmSiteAlsoStopped": false,
75    "clientAffinityEnabled": true,
76    "clientCertEnabled": false,
77    "hostNamesDisabled": false,
78    "containerSize": 0,
79    "dailyMemoryTimeQuota": 0,
80    "httpsOnly": false
81  },
82  ],
83  {
84    "type": "Microsoft.Web/sites/config",
85    "apiVersion": "2016-08-01",
86    "name": "[concat(variables('siteName'), '/web')]",
87    "location": "[variables('location')]",
88    "dependsOn": [
89      "[resourceId('Microsoft.Web/sites', variables('siteName'))]"
90    ],
91    "properties": {
92      "numberOfWorkers": 1,
93      "defaultDocuments": [
94        "Default.htm",
95        "Default.html",
96        "Default.asp"
97      ]
98    }
99  }
100}
```

PROBLEMS (22) OUTPUT DEBUG CONSOLE TERMINAL

1: bash

Osiris-2:WebAppBot krisnatagoras\$ az group deployment create --name mydep --resource-group "WebAppResourceGroup" --template-file ".\azuredeploy.json"

master* 0 22 Ln 80, Col 35 Spaces: 4 UTF-8 LF Azure Resource Manager Template

You gonna need to insert the parameters information:

The screenshot shows the Visual Studio Code editor with the `azuredeploy.parameters.json` file open. The file is a JSON template for the parameters of the `azuredeploy.json` file. The terminal at the bottom shows the output of the `az group deployment create` command, including the command being executed and the values for the parameters.

```
1 {
2   "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentParameters.json#",
3   "contentVersion": "1.0.0.0",
4   "parameters": {
5     "botEnv": {
6       "value": "prod"
7     },
8     "botId": {
9       "value": "uks-mychatbot2"
10    },
11    "location": {
12      "value": "UK South"
13    },
14    "sku": {
15      "value": "F0"
16    },
17    "kind": {
18      "value": "sdk"
19    },
20    "siteName": {
21      "value": "uks-mychatbot2"
22    },
23    "createNewStorage": {
24      "value": true
25    },
26    "storageAccountName": {
27      "value": "uksmychatbot2bfaa"
28    },
29    "storageAccountLocation": {
30      "value": ""
31    }
32  }
33}
```

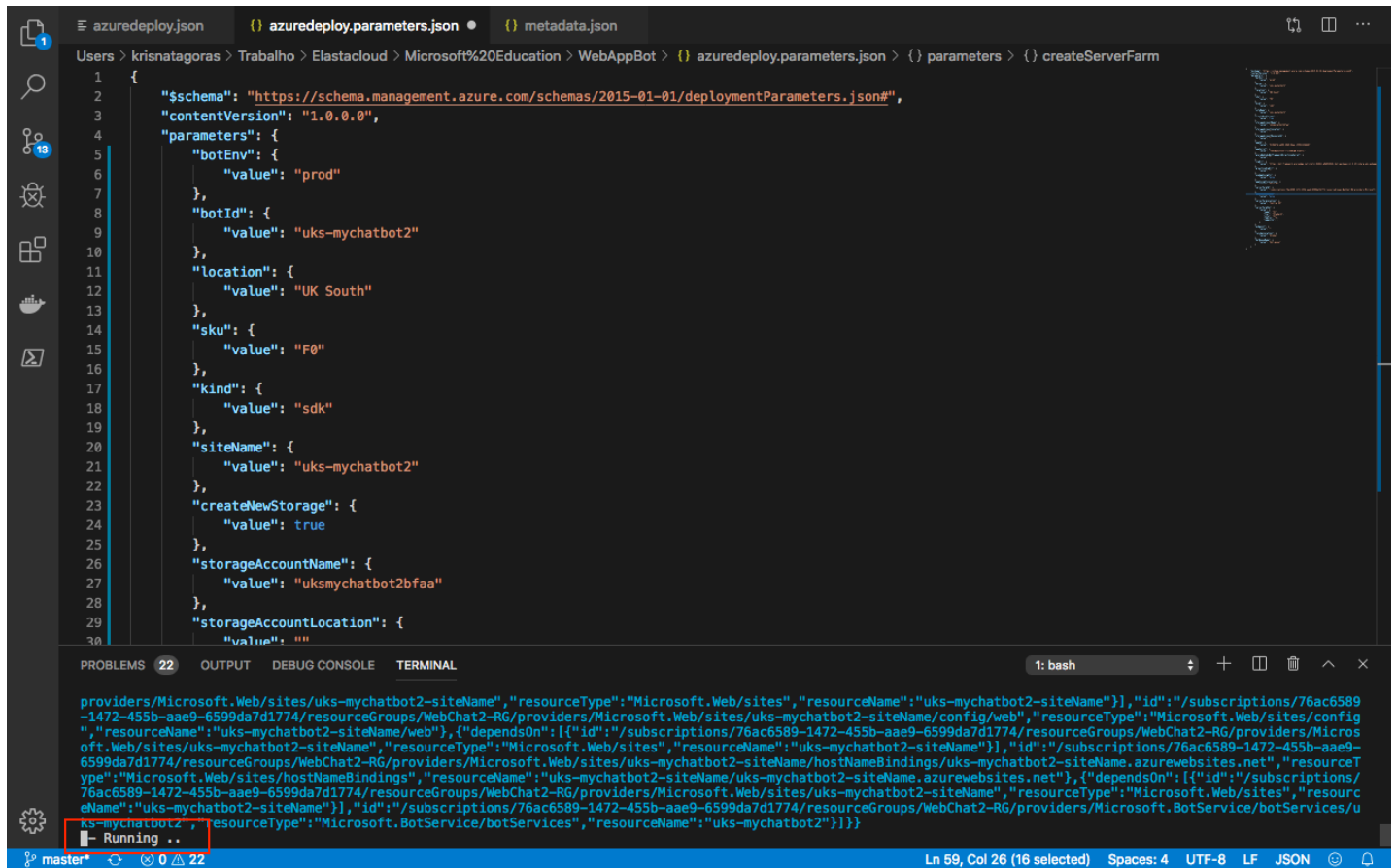
PROBLEMS (22) OUTPUT DEBUG CONSOLE TERMINAL

1: bash

Event: CommandInvoker.OnPostCommandTableCreate [<function register_ids_argument.<locals>.add_ids_arguments at 0x108540680>, <function register_cache_arguments.<locals>.add_cache_arguments at 0x1085409e0>]
Event: CommandInvoker.OnCommandTableLoaded []
Event: CommandInvoker.OnPreParseArgs [<function documentdb.deprecate at 0x109557290>]
Event: CommandInvoker.OnPostParseArgs [<function OutputProducer.handle_output_argument at 0x108427680>, <function CLIQuery.handle_query_parameter at 0x108459290>, <function register_ids_argument.<locals>.parse_ids_arguments at 0x108540710>, <function handler at 0x1096c2290>]
attempting to read file .\azuredeploy.json as utf-8-sig
Please provide string value for 'webAppName' (? for help): uks-mychatbot2
Please provide string value for 'appId' (? for help):
Please provide string value for 'appSecret' (? for help):

master* 0 22 Ln 59, Col 26 (16 selected) Spaces: 4 UTF-8 LF JSON

As you can see, it's running.



```
Users > krisnatagoras > Trabalho > Elastacloud > Microsoft%20Education > WebAppBot > {} azuredploy.parameters.json > {} parameters > {} createServerFarm
1 {
2   "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentParameters.json#",
3   "contentVersion": "1.0.0.0",
4   "parameters": {
5     "botEnv": {
6       "value": "prod"
7     },
8     "botId": {
9       "value": "uks-mychatbot2"
10    },
11    "location": {
12      "value": "UK South"
13    },
14    "sku": {
15      "value": "F0"
16    },
17    "kind": {
18      "value": "sdk"
19    },
20    "siteName": {
21      "value": "uks-mychatbot2"
22    },
23    "createNewStorage": {
24      "value": true
25    },
26    "storageAccountName": {
27      "value": "uksmychatbot2bfaa"
28    },
29    "storageAccountLocation": {
30      "value": ""
31    }
32  }
33 }
```

```
providers/Microsoft.Web/sites/uks-mychatbot2-siteName", "resourceType": "Microsoft.Web/sites", "resourceName": "uks-mychatbot2-siteName"}, {"id": "/subscriptions/76ac6589-1472-455b-aae9-6599da7d1774/resourceGroups/WebChat2-RG/providers/Microsoft.Web/sites/uks-mychatbot2-siteName/config/web", "resourceType": "Microsoft.Web/sites/config", "resourceName": "uks-mychatbot2-siteName/web"}, {"dependsOn": [{"id": "/subscriptions/76ac6589-1472-455b-aae9-6599da7d1774/resourceGroups/WebChat2-RG/providers/Microsoft.Web/sites/uks-mychatbot2-siteName", "resourceType": "Microsoft.Web/sites", "resourceName": "uks-mychatbot2-siteName"}], "id": "/subscriptions/76ac6589-1472-455b-aae9-6599da7d1774/resourceGroups/WebChat2-RG/providers/Microsoft.Web/sites/uks-mychatbot2-siteName/hostNameBindings/uks-mychatbot2-siteName.azurewebsites.net", "resourceType": "Microsoft.Web/sites/hostNameBindings", "resourceName": "uks-mychatbot2-siteName/uks-mychatbot2-siteName.azurewebsites.net"}, {"dependsOn": [{"id": "/subscriptions/76ac6589-1472-455b-aae9-6599da7d1774/resourceGroups/WebChat2-RG/providers/Microsoft.Web/sites/uks-mychatbot2-siteName", "resourceType": "Microsoft.Web/sites", "resourceName": "uks-mychatbot2-siteName"}], "id": "/subscriptions/76ac6589-1472-455b-aae9-6599da7d1774/resourceGroups/WebChat2-RG/providers/Microsoft.BotService/botServices/uks-mychatbot2", "resourceType": "Microsoft.BotService/botServices", "resourceName": "uks-mychatbot2"}]}
```

Running ..

Go grab a cup of coffee, have some fresh air and I'm sure that before you come back you gonna have your Ubuntu Server Virtual Machine ready.

And there we go, our deploy is Succeeded:

Microsoft Azure portal showing the Resource groups page. The left sidebar lists navigation options like Home, Dashboard, All services, and FAVORITES. The main content area displays a list of resources under the selected Resource group. The table shows 3 records:

Name	Type	Location
uks-mychatbot	Web App Bot	global
uks-mychatbot	App Service	Central US
uks-mychatbot-appplan	App Service plan	Central US

The 'uks-mychatbot' Web App Bot is highlighted with a red box.

Click on the Web App Bot.

Microsoft Azure portal showing the 'uks-mychatbot' Web App Bot details page. The left sidebar lists navigation options. The main content area displays the bot's configuration and resources. The 'Test in Web Chat' option is highlighted with a red box.

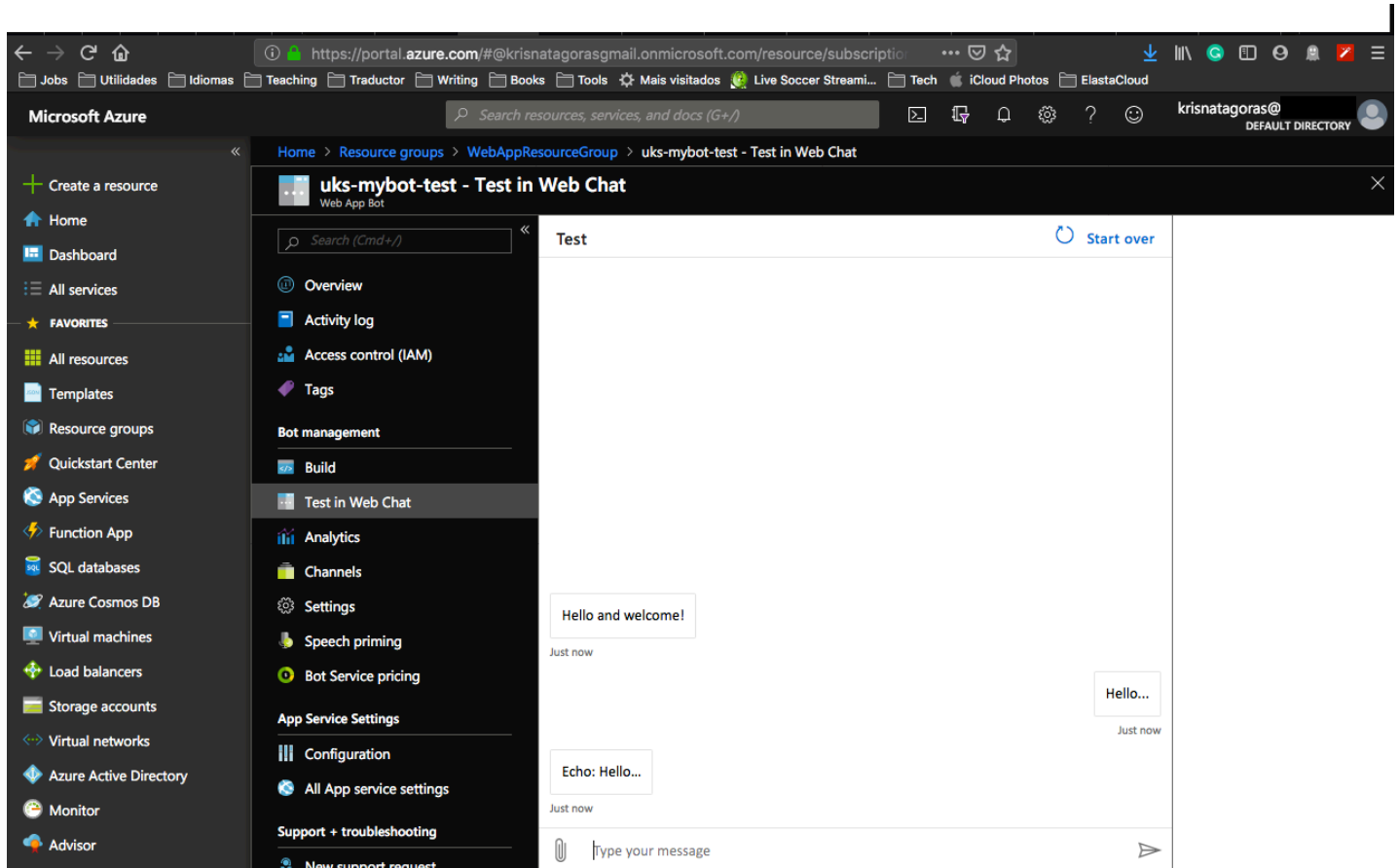
Resources

- Plan:** Review the bot design guidelines for best practices.
- Build:** Download your bot's source code and develop it locally, using your preferred development environment. Add intelligence to your bot with services such as: Language Understanding, QnA Maker, and Dispatch.
- Test:** Test your bot online in Web Chat.
- Publish:** Publish your bot from your local environment directly to Azure or automatically with Continuous Deployment.

On this blade you have all the setups for your Web App Bot. On this overview you have resources from **Plan** to -> **Build** to -> **Test** to -> **Publish** and -> **Connect**. You can download the source code, publish your bot in different environments, connect to different channels, like Facebook, Skype, Cortana and also MS Teams.

All these resources are great, right?

Let's now test our Web ChatBot. Click in **Test in Web Chat**



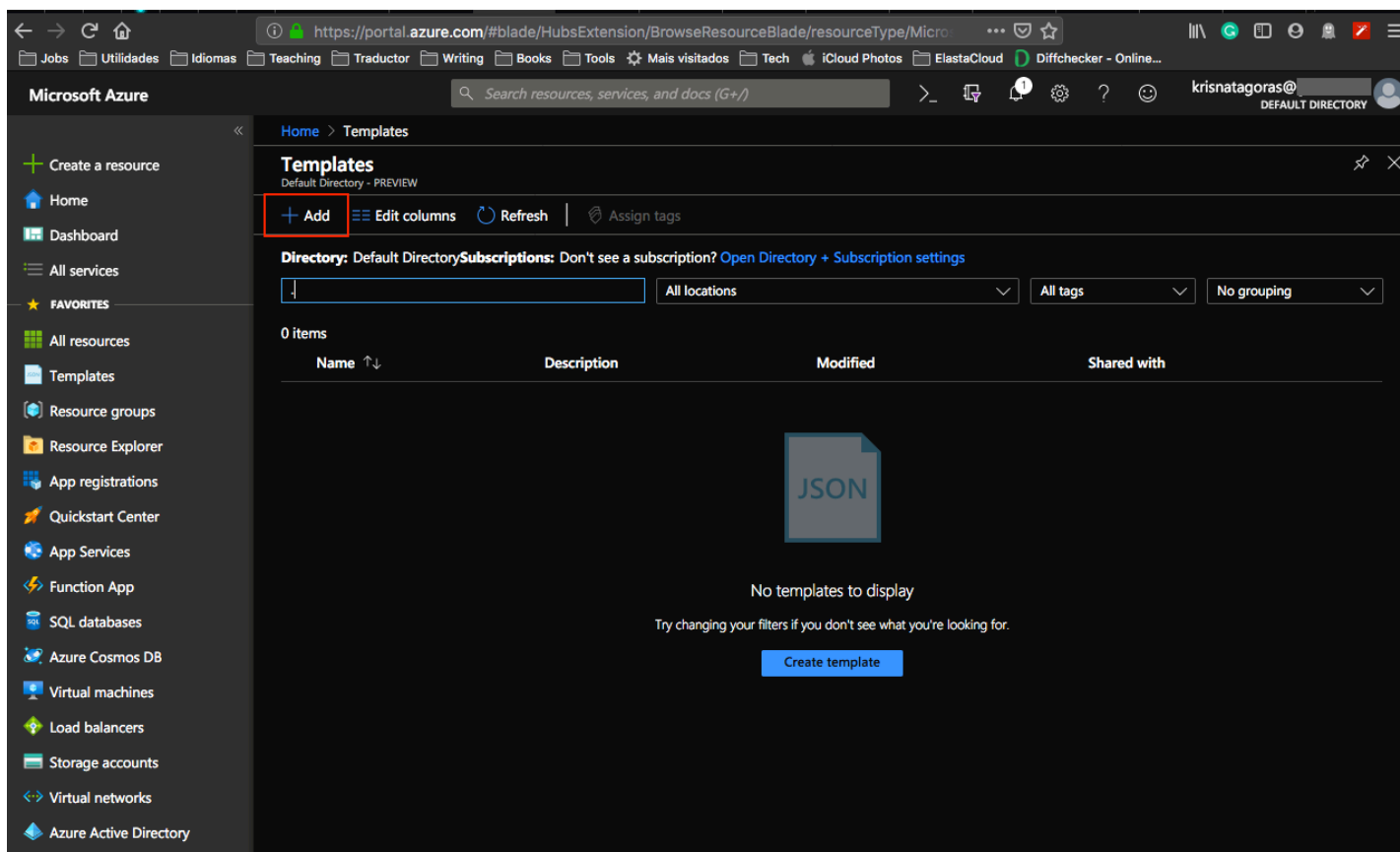
And that is just the tip of the iceberg. There is plenty of documentation to dig in and most important:

Don't forget to have fun!

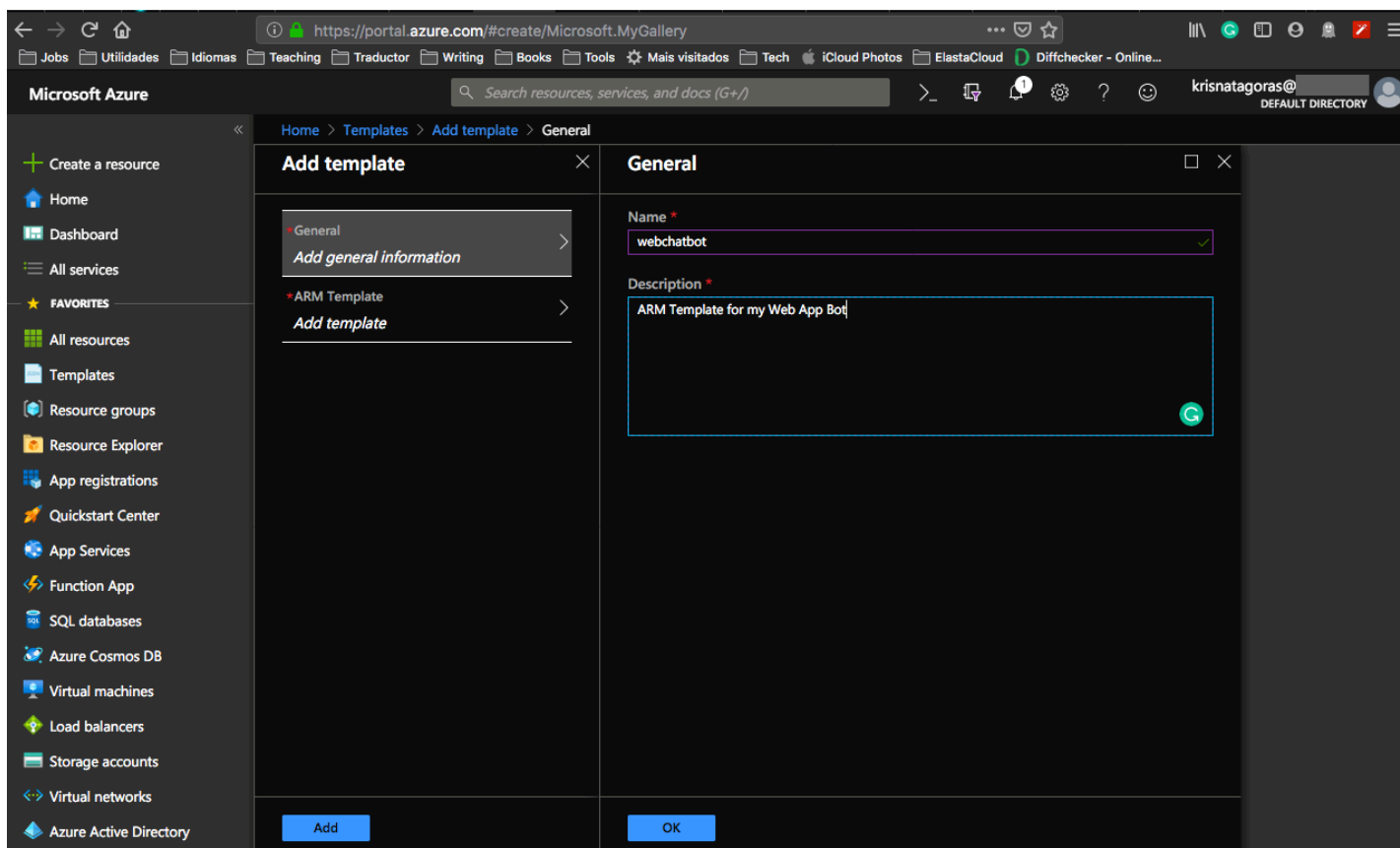
Using the Portal

At the Portal, in All Services look for **Templates**, you can favorite this service.

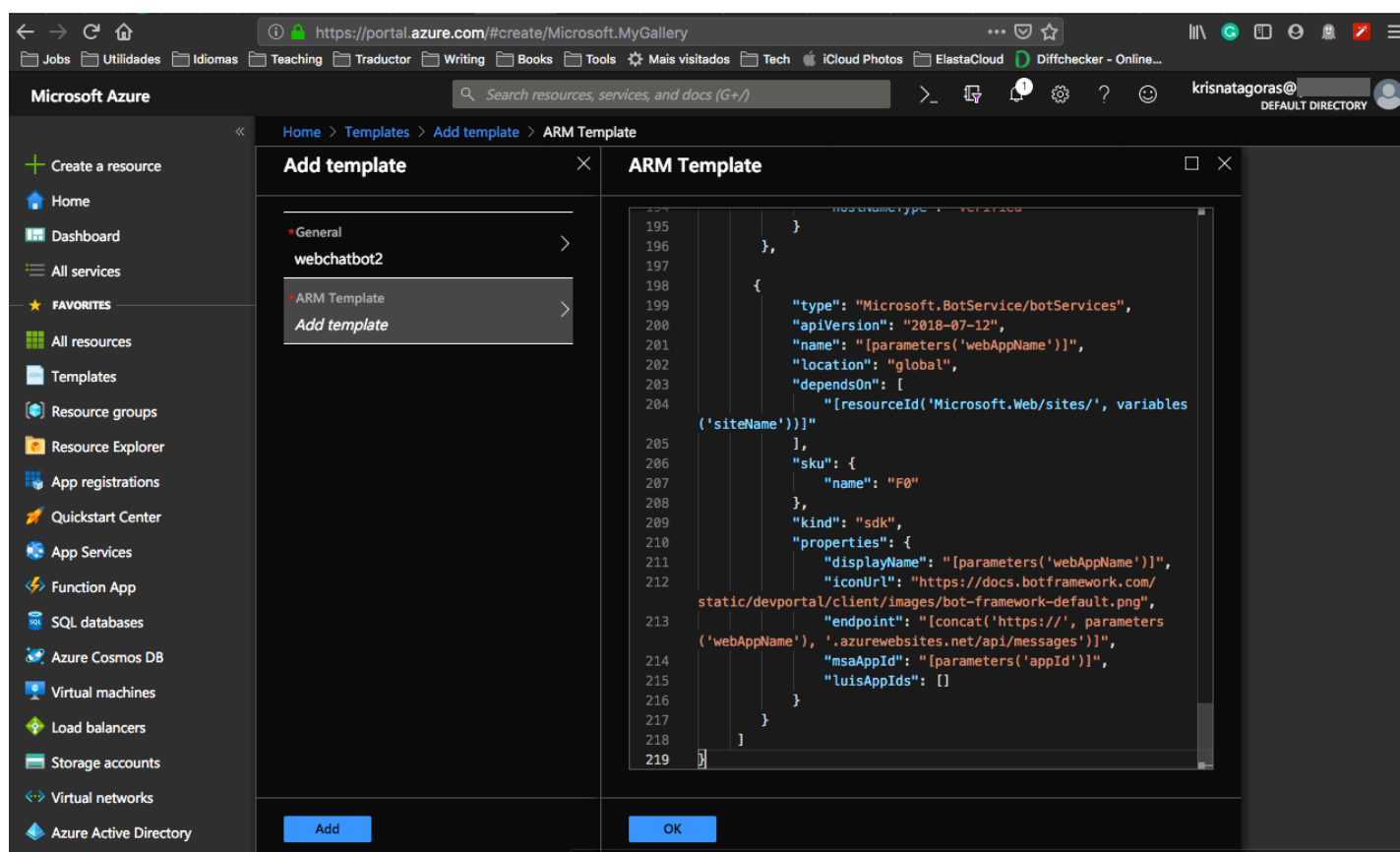
Click in **Add** to add your template:



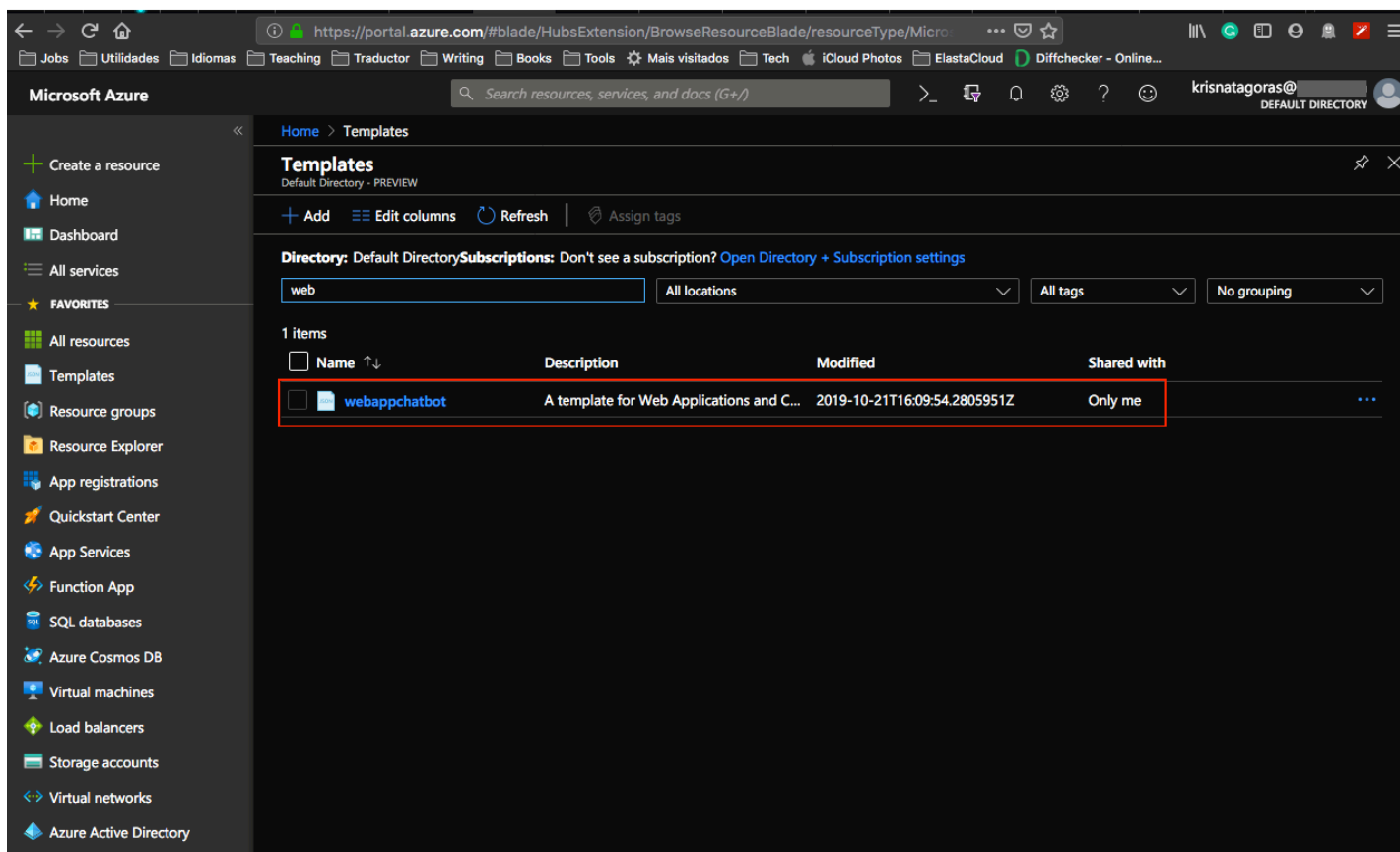
On General, type a name and a description for your template, and click on [OK].



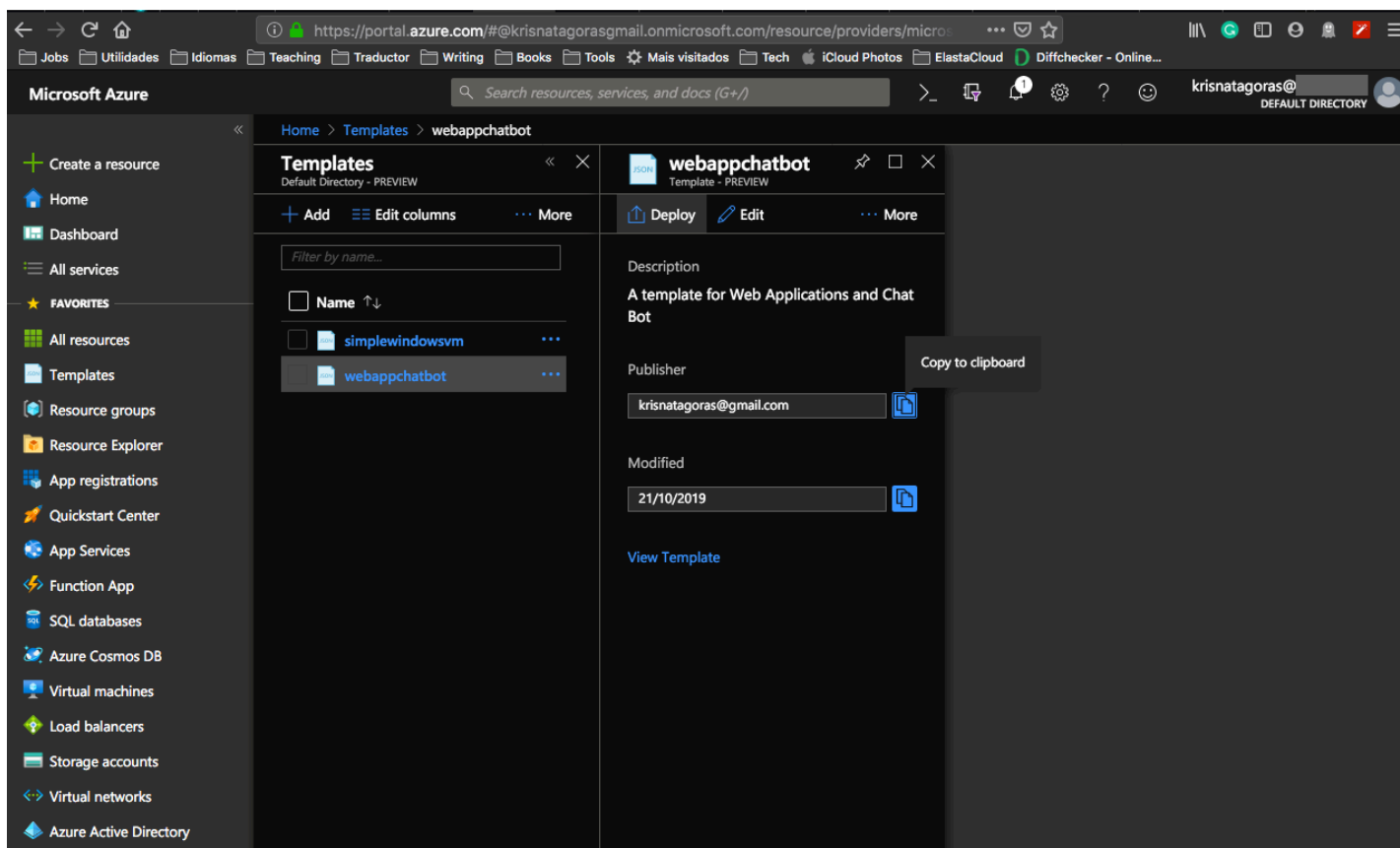
On ARM Template, replace the contents of the template with your template, and click on [OK].



Click on the refresh button and there is your template:

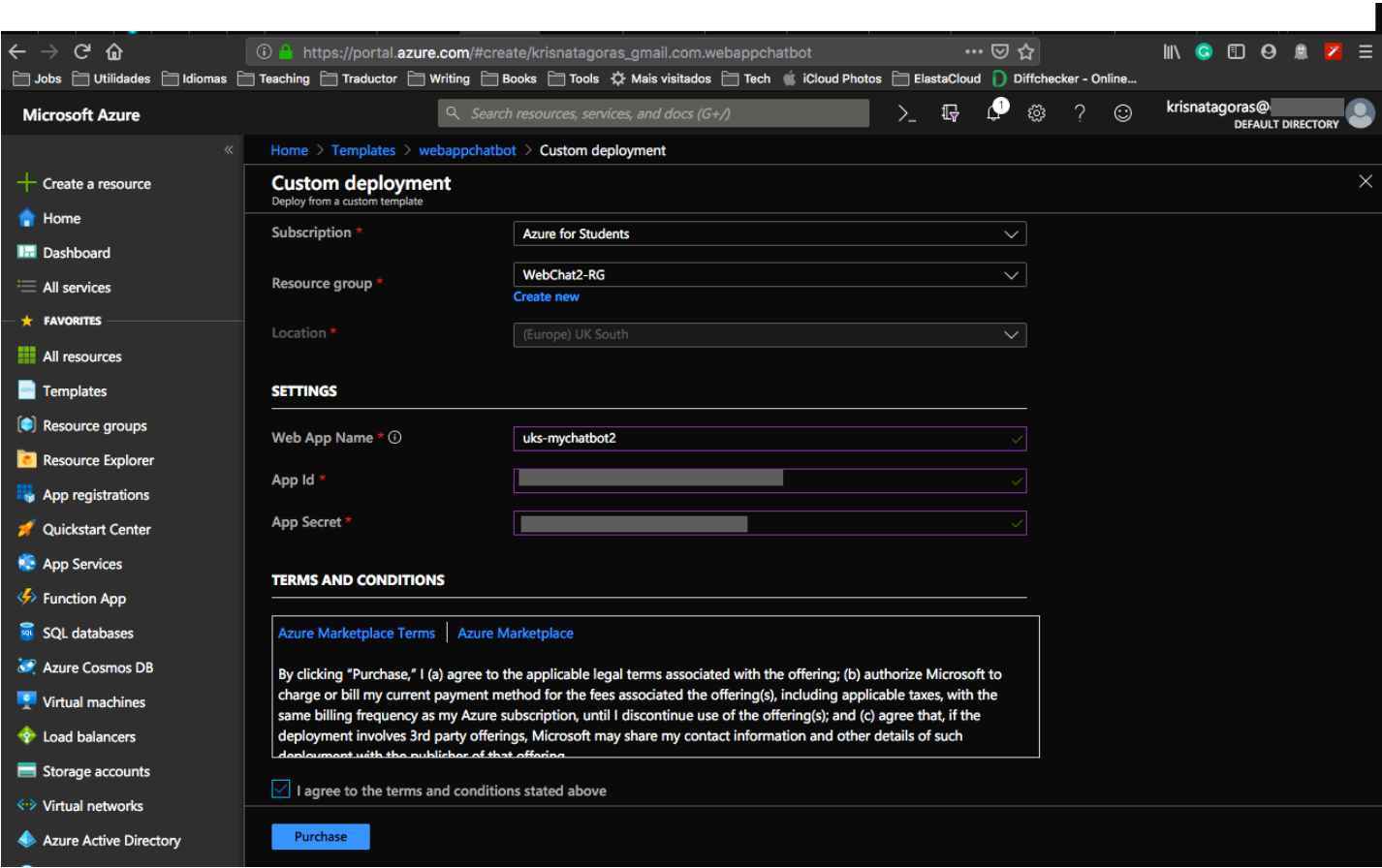


Open the template and click in [Deploy]



On the screen Custom Deployment, insert the information that you must be already familiar with.

Select [I agree] and click on [Purchase].



As you can see, it's deploying.

Microsoft Azure portal screenshot showing the deployment progress of a Web App Bot. The interface includes a sidebar with navigation options, a main content area with "Azure services" and "Recent resources" sections, and a "Notifications" panel on the right.

Azure services

- Create a resource
- Templates
- Resource groups
- App registrations
- Storage accounts
- All services

Recent resources

NAME	TYPE
uks-mychatbot	Web App
WebChat-RG	Resource Group
uks-mychatbot2	Web App
WebChat2-RG	Resource Group
uks-mychatbot	App Service
WebChat3-RG	Resource Group
MyResourceGroup	Resource Group
76ac6589-1472-455b-aae9-6599da7d1774	Subscription
RedHat-Test	Virtual Machine
cf76da84-ca82-4f6d-93c1-c719c8da5e7e	Subscription

Notifications

More events in the activity log → Dismiss all

Deployment in progress... Running

Deployment to resource group 'WebChat2-RG' is in progress.

a few seconds ago

And voilà, you have your Web App Bot deployed.

Microsoft Azure portal screenshot showing the deployment success of a Web App Bot. The interface is similar to the previous one, but the "Notifications" panel now shows a successful deployment message.

Azure services

- Create a resource
- Templates
- Resource groups
- App registrations
- Storage accounts
- All services

Recent resources

NAME	TYPE
uks-mychatbot	Web App
WebChat-RG	Resource Group
uks-mychatbot2	Web App
WebChat2-RG	Resource Group
uks-mychatbot	App Service
WebChat3-RG	Resource Group
MyResourceGroup	Resource Group
76ac6589-1472-455b-aae9-6599da7d1774	Subscription
RedHat-Test	Virtual Machine
cf76da84-ca82-4f6d-93c1-c719c8da5e7e	Subscription

Notifications

More events in the activity log → Dismiss all

Deployment succeeded

Deployment 'krisnatagoras_gmail.com.webappchatbot' to resource group 'WebChat2-RG' was successful.

Go to resource group Pin to dashboard

a few seconds ago

Go to the Resource. Repeat the test you have done before and enjoy your coding.

p.s.: Pretty easy to create resources on Azure, right? But if you are the sort of IT guy that always looks for automating things on the extreme :D Surprise, surprise!. Just click on the button below and it will automatically deploy the VM on your Azure Portal.

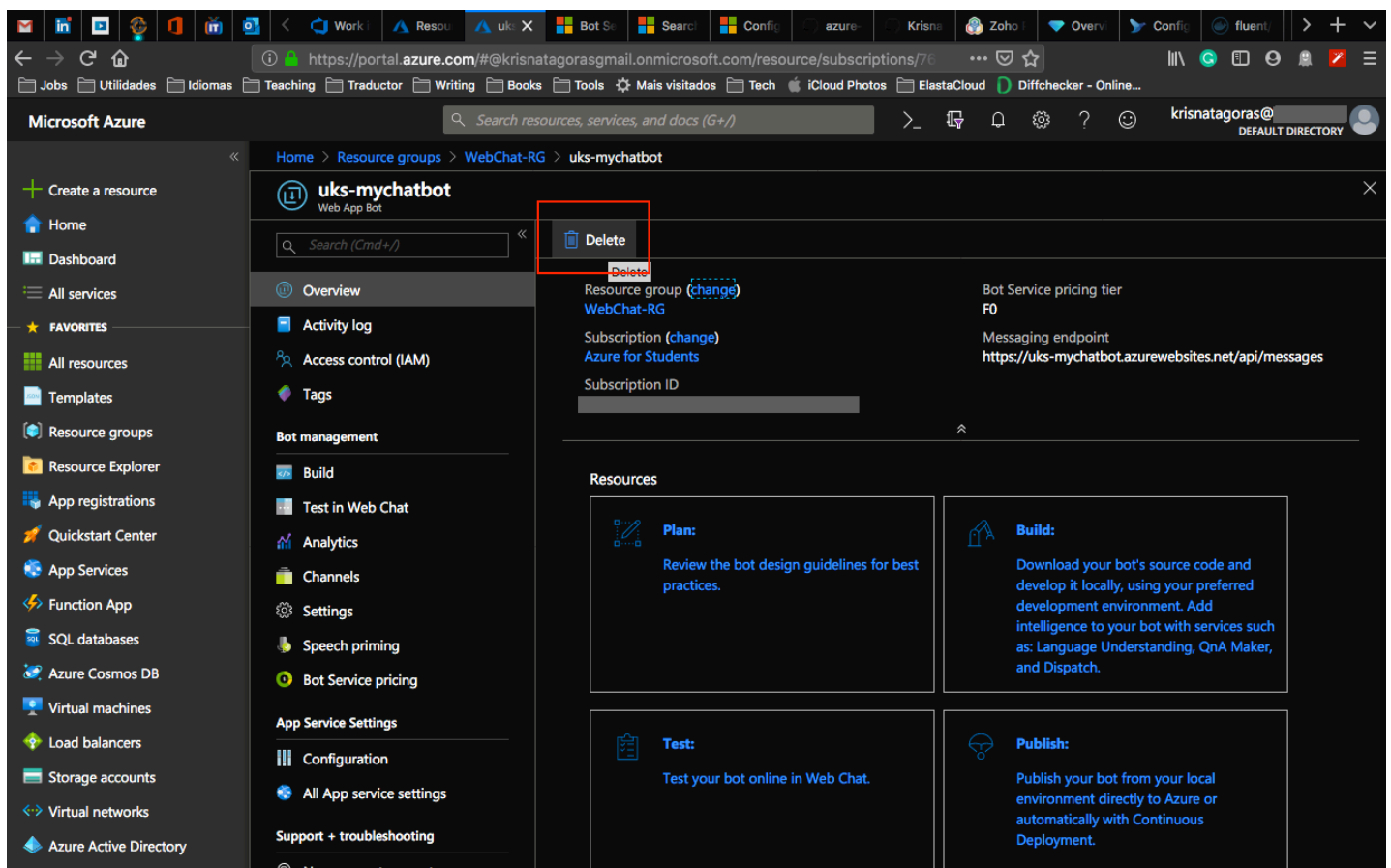


Important disclaimer: For bot Service, we are using the F0 free tier of access, which means that we have a limited amount of access, by the time of writing this guide is 10K of messages that you can use in the Web ChatBot. Anyway if you exceed that access you may be charged, in order to avoid that you can delete the resource and deploy it again, you don't want to finish all your credits at once, right?

How to shutdown your resources:

Using the portal:

On the portal, open your Resource Group, select your Web ChatBot and then just click on the [Delete] Button.



Just refresh your screen and you are good to go.

