

Azure Functions App

This time we gonna deploy a **Function App**. Azure Functions is a serverless compute service that lets you run event-triggered code without having to explicitly provision or manage infrastructure.

With a **Function App** you also deploy a hosting Web Plan and a Storage account. The WebPlan is settled for Consumption. For more information about hosting Plans click [here!](#)

And the Storage Account is for General Purpose **Standard_LRS**, for more information about Storage Accounts, click [here!](#)

How Functions App works in Azure?

Azure Functions is a solution for easily running small pieces of code, or "functions," in the cloud. You can write just the code you need for the problem at hand, without worrying about a whole application or the infrastructure to run it. Functions can make development even more productive, and you can use your development languages of choice, such as C#, Java, JavaScript, PowerShell, and Python. Pay only for the time your code runs and trust Azure to scale as needed. Azure Functions lets you develop serverless applications on Microsoft Azure.

What can I do with Functions?

Functions are a great solution for processing data, integrating systems, working with the internet-of-things (IoT), and building simple APIs and microservices. Consider Functions for tasks like image or order processing, file maintenance, or for any tasks that you want to run on a schedule.

How much does Functions cost?

Azure Functions has two kinds of pricing plans. Choose the one that best fits your needs:

- **Consumption plan** - When your function runs, Azure provides all of the necessary computational resources. You don't have to worry about resource management, and you only pay for the time that your code runs.
- **App Service plan** - Run your functions just like your web apps. When you are already using App Service for your other applications, you can run your functions on the same plan at no additional cost.

The Template

The only parameter we need to inform is:

Parameter	Suggested value	Description
funcName	<i>locationnameenviroment</i> i.e.: uksmyfunctionst	The unique name of Function App. I recommend you to use the notation above, which helps to create a unique name for your Web Application. 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.

Deployment

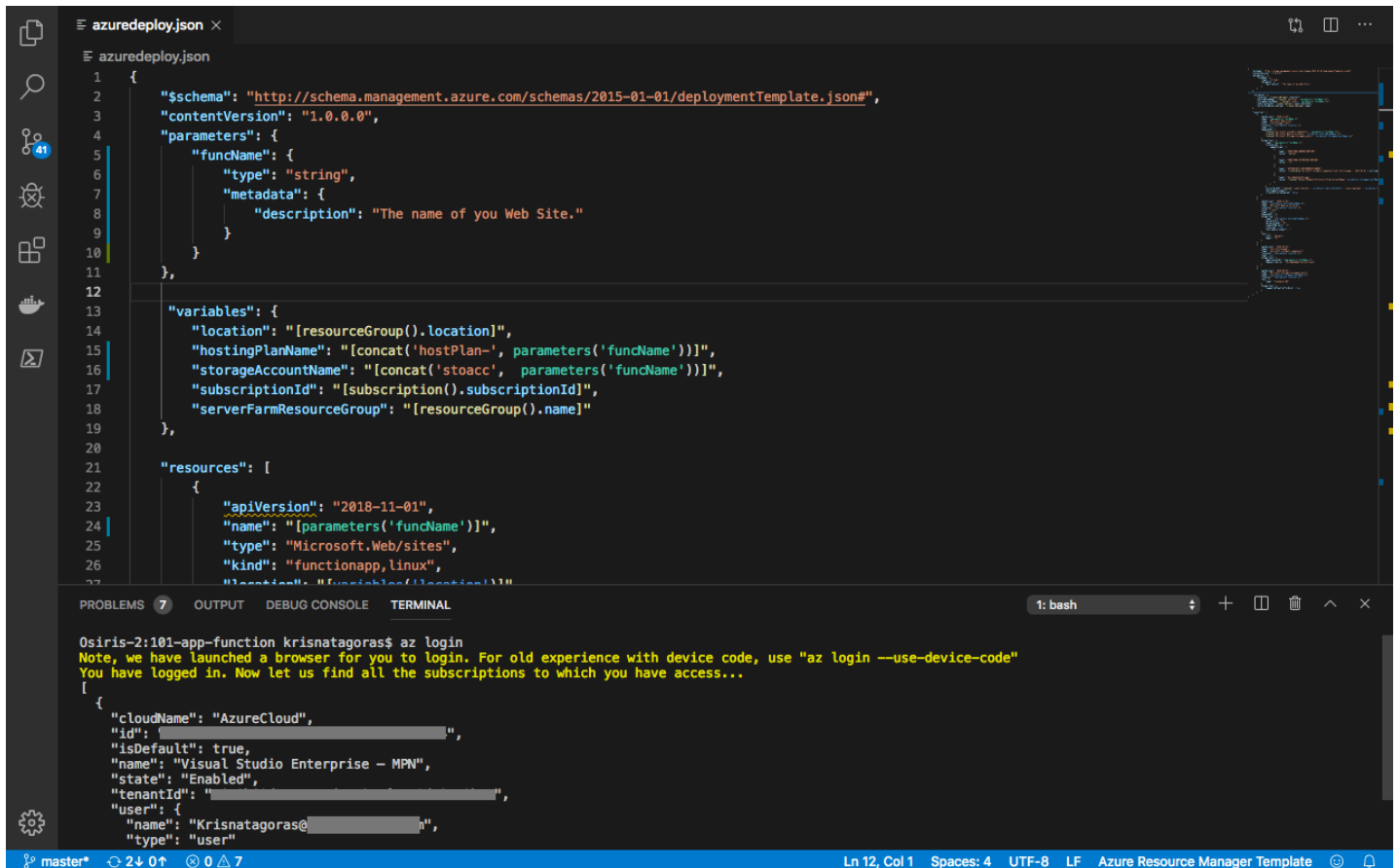
There are a few ways to deploy this template. You can use [PowerShell](#), [Azure CLI](#), [Azure Portal](#) or your favorite SDK.

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**



```
1 {
2   "$schema": "http://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
3   "contentVersion": "1.0.0.0",
4   "parameters": {
5     "funcName": {
6       "type": "string",
7       "metadata": {
8         "description": "The name of you Web Site."
9       }
10    },
11  },
12  "variables": {
13    "location": "[resourceGroup().location]",
14    "hostingPlanName": "[concat('hostPlan-', parameters('funcName'))]",
15    "storageAccountName": "[concat('stoacc', parameters('funcName'))]",
16    "subscriptionId": "[subscription().subscriptionId]",
17    "serverFarmResourceGroup": "[resourceGroup().name]"
18  },
19  "resources": [
20    {
21      "apiVersion": "2018-11-01",
22      "name": "[parameters('funcName')]",
23      "type": "Microsoft.Web/sites",
24      "kind": "functionapp,linux",
25      "location": "[variables('location')]"
26    }
27  ]
28 }
```

PROBLEMS 7 OUTPUT DEBUG CONSOLE TERMINAL 1: bash

Osiris-2:101-app-function 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 have logged in. Now let us find all the subscriptions to which you have access...

```
{
  "cloudName": "AzureCloud",
  "id": "00000000-0000-0000-0000-000000000000",
  "isDefault": true,
  "name": "Visual Studio Enterprise - MPN",
  "state": "Enabled",
  "tenantId": "00000000-0000-0000-0000-000000000000",
  "user": {
    "name": "Krisnatagoras@00000000-0000-0000-0000-000000000000",
    "type": "user"
  }
}
```

master* 24 0 7 Ln 12, Col 1 Spaces: 4 UTF-8 LF Azure Resource Manager Template

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 >"

The screenshot shows the Visual Studio Code editor with a file named `azuredeploy.json` open. The file contains an ARM template for a function app. The `parameters` section defines a `funcName` parameter of type `string` with a description "The name of you Web Site.". The `variables` section defines several variables: `location` (from `resourceGroup().location`), `hostingPlanName` (concatenated with `funcName`), `storageAccountName` (concatenated with `funcName`), `subscriptionId` (from `subscription().subscriptionId`), and `serverFarmResourceGroup` (from `resourceGroup().name`). The `resources` section defines a `Microsoft.Web/sites` resource of type `functionapp,linux`, using the variables defined above. Below the editor, the `TERMINAL` tab is active, showing a JSON object for the user and the command `az account set --subscription "872bc1ca-bf67-40dd-8a7f-da1792592284"` being executed. The command is highlighted with a red box.

```
{
  "$schema": "http://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "funcName": {
      "type": "string",
      "metadata": {
        "description": "The name of you Web Site."
      }
    }
  },
  "variables": {
    "location": "[resourceGroup().location]",
    "hostingPlanName": "[concat('hostPlan-', parameters('funcName'))]",
    "storageAccountName": "[concat('stoacc', parameters('funcName'))]",
    "subscriptionId": "[subscription().subscriptionId]",
    "serverFarmResourceGroup": "[resourceGroup().name]"
  },
  "resources": [
    {
      "apiVersion": "2018-11-01",
      "name": "[parameters('funcName')]",
      "type": "Microsoft.Web/sites",
      "kind": "functionapp,linux",
      "location": "[variables('location')]"
    }
  ]
}
```

```
{
  "cloudName": "AzureCloud",
  "id": "872bc1ca-bf67-40dd-8a7f-da1792592284",
  "isDefault": true,
  "name": "Visual Studio Enterprise - MPN",
  "state": "Enabled",
  "tenantId": "872bc1ca-bf67-40dd-8a7f-da1792592284",
  "user": {
    "name": "Krisnatagoras@872bc1ca-bf67-40dd-8a7f-da1792592284",
    "type": "user"
  }
}
```

```
Osiris-2:101-app-function krisnatagoras$ az account set --subscription "872bc1ca-bf67-40dd-8a7f-da1792592284"
```

Resource Group

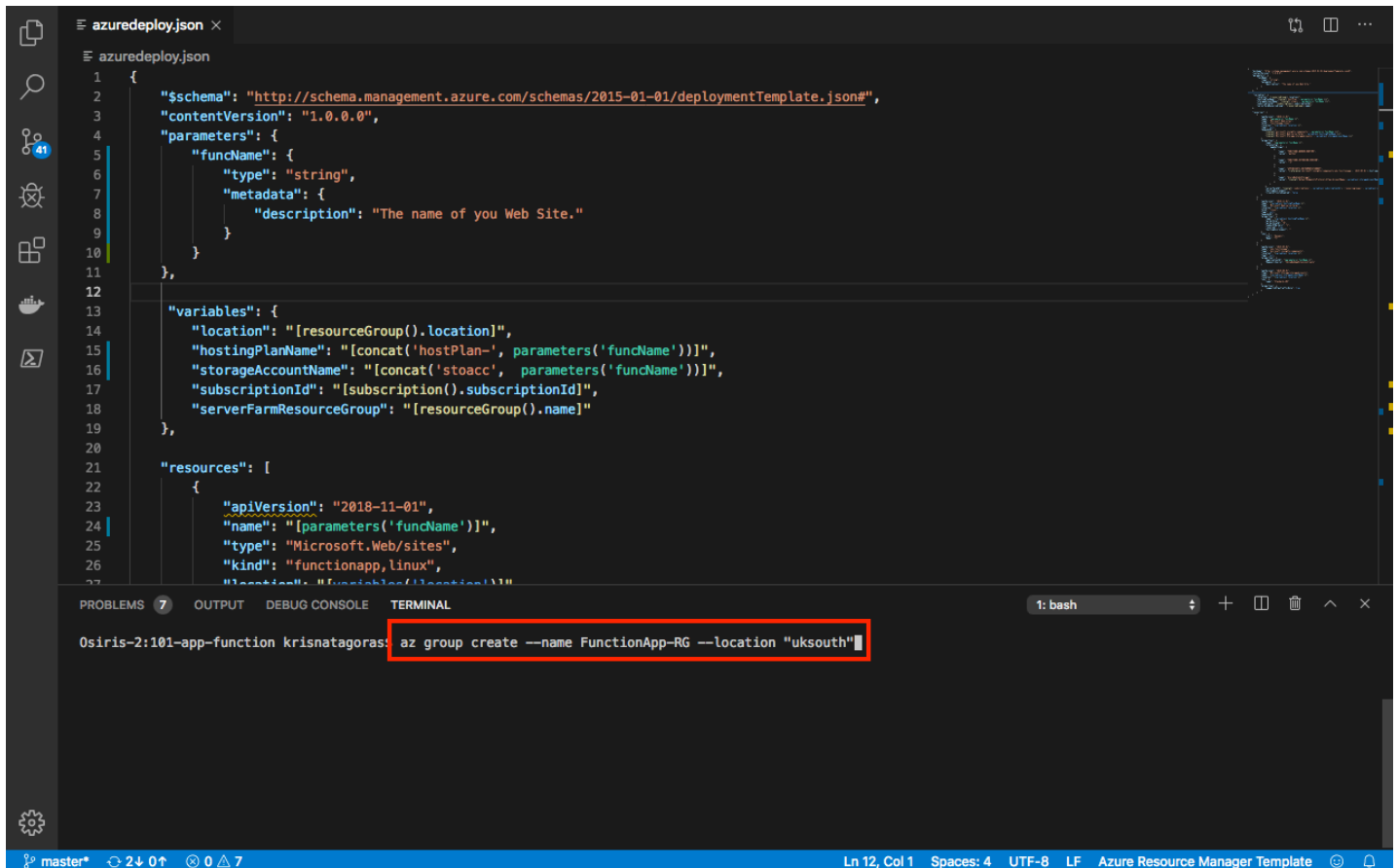
After you logged in, we gonna need to create a Resource Group for our deployment. If you haven't yet created a Resource Group, we gonna do that now! But what is a Resource Group, one might ask. Bare with me! A Resource Group is a container that holds related resources for an Azure solution. The resource 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 FunctionApp-RG --location < yourlocation >

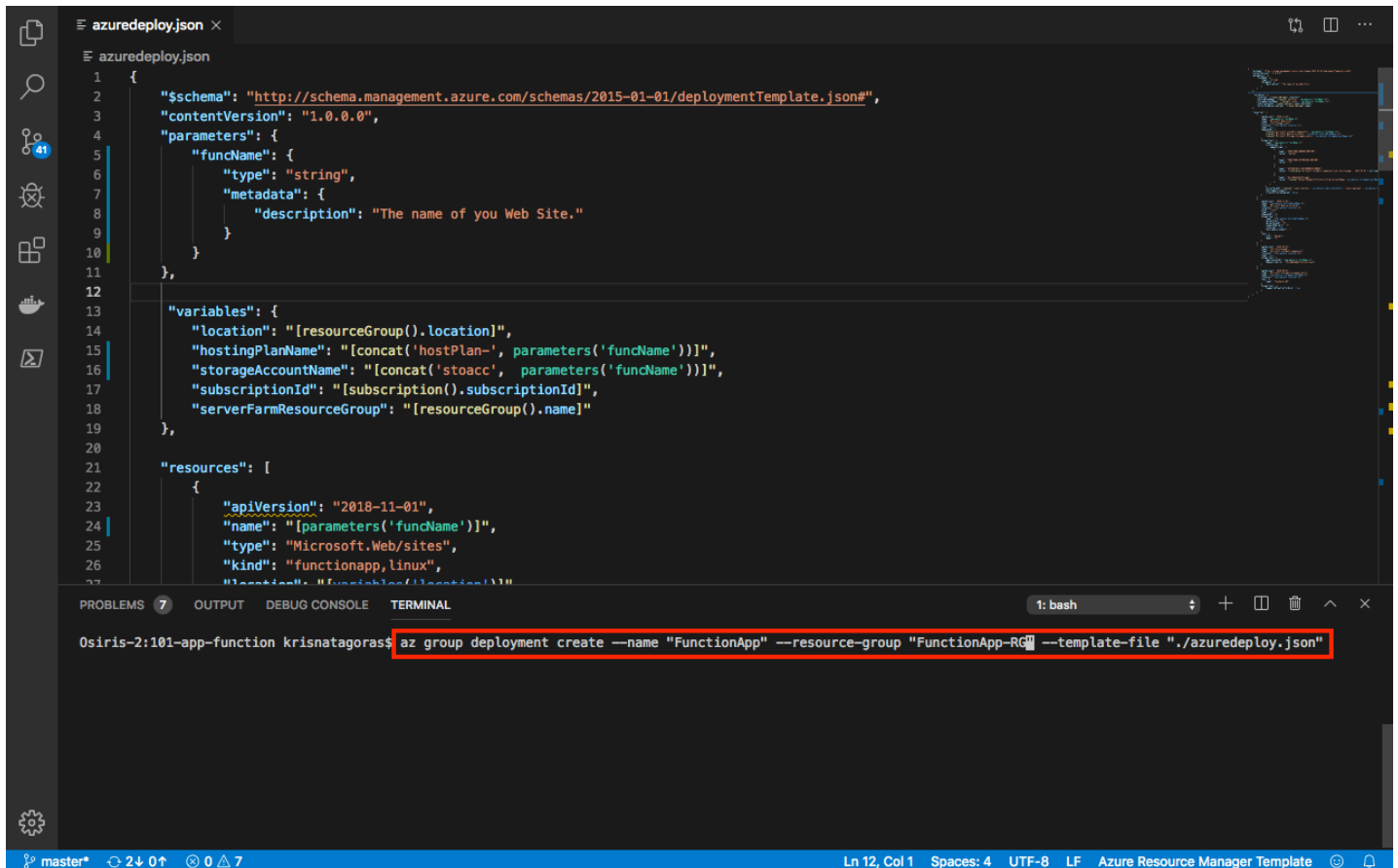


```
1 {
2   "$schema": "http://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
3   "contentVersion": "1.0.0.0",
4   "parameters": {
5     "funcName": {
6       "type": "string",
7       "metadata": {
8         "description": "The name of you Web Site."
9       }
10    },
11  },
12  "variables": {
13    "location": "[resourceGroup().location]",
14    "hostingPlanName": "[concat('hostPlan-', parameters('funcName'))]",
15    "storageAccountName": "[concat('stoacc', parameters('funcName'))]",
16    "subscriptionId": "[subscription().subscriptionId]",
17    "serverFarmResourceGroup": "[resourceGroup().name]"
18  },
19  "resources": [
20    {
21      "apiVersion": "2018-11-01",
22      "name": "[parameters('funcName')]",
23      "type": "Microsoft.Web/sites",
24      "kind": "functionapp,linux",
25      "location": "[variables('location')]"
26    }
27  ]
28 }
```

```
Osiris-2:101-app-function Krisnatagorasa az group create --name FunctionApp-RG --location "uksouth"
```

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

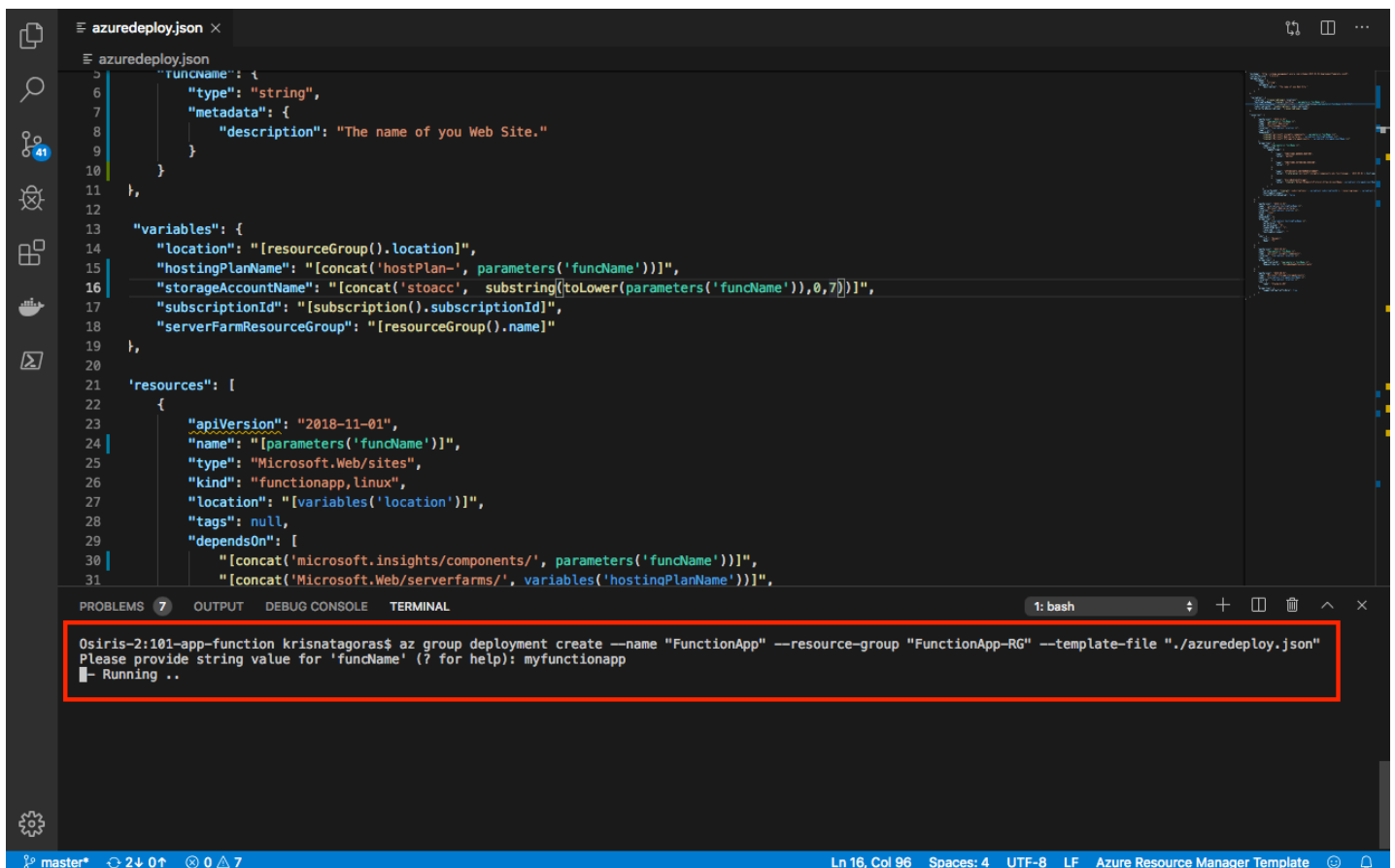
az group deployment create --name "name of your deployment" --resource-group "The group you created" --template-file "./azuredeploy.json"



```
az group deployment create --name "FunctionApp" --resource-group "FunctionApp-RG" --template-file "./azuredeploy.json"
```

The screenshot shows a Visual Studio Code editor with a file named `azuredeploy.json` open. The file contains an ARM template for a Function App. The `parameters` section defines a `funcName` parameter of type `string` with a description "The name of you Web Site.". The `variables` section defines several variables including `location`, `hostingPlanName`, `storageAccountName`, `subscriptionId`, and `serverFarmResourceGroup`. The `resources` section defines a `Microsoft.Web/sites` resource of type `functionapp,linux`. The terminal window at the bottom shows the command `az group deployment create --name "FunctionApp" --resource-group "FunctionApp-RG" --template-file "./azuredeploy.json"` being executed.

You gonna need to insert the parameter information. As you can see, it's running.

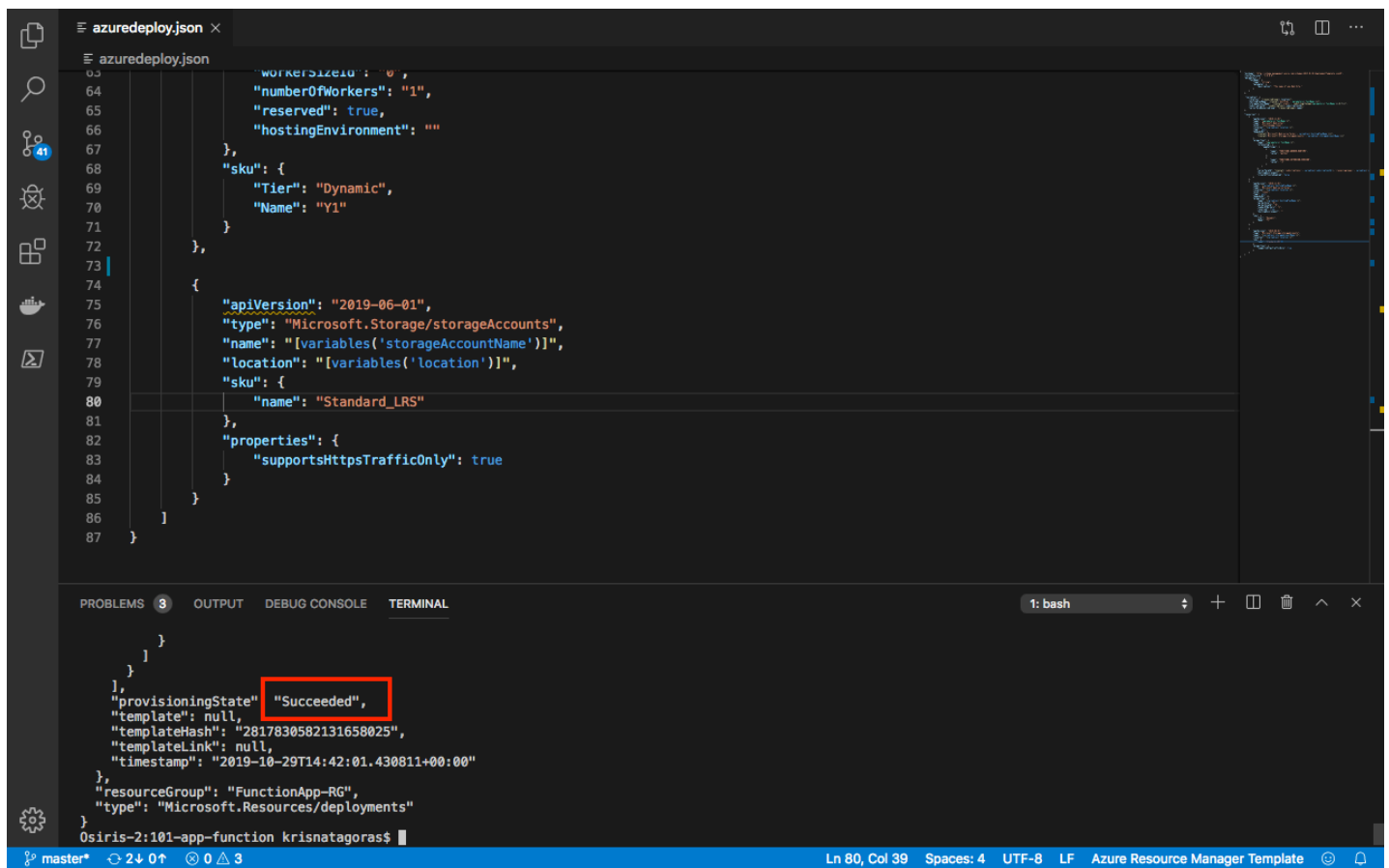


```
az group deployment create --name "FunctionApp" --resource-group "FunctionApp-RG" --template-file "./azuredeploy.json"
```

The screenshot shows the same Visual Studio Code editor with the `azuredeploy.json` file. The terminal window shows the command `az group deployment create --name "FunctionApp" --resource-group "FunctionApp-RG" --template-file "./azuredeploy.json"` being executed. The terminal output shows the command running, followed by a prompt: `Please provide string value for 'funcName' (? for help): myfunctionapp`.

Go grab a cup of coffee, have some fresh air and I'm sure that before you come back you gonna have your Web App with GitHub Account will be deployed.

And there we go, our deploy is Succeeded:



The screenshot shows a Visual Studio Code editor with a file named `azuredeploy.json` open. The file contains a JSON configuration for an Azure deployment. The configuration includes a `workersizeid` of `0`, `numberOfWorkers` of `1`, `reserved` set to `true`, and `hostingEnvironment` set to an empty string. It also defines a `sku` with `Tier` `Dynamic` and `Name` `V1`. A storage account is configured with `apiVersion` `2019-06-01`, `type` `Microsoft.Storage/storageAccounts`, and `name` `[variables('storageAccountName')]`. The `location` is `[variables('location')]`, and the `sku` is `Standard_LRS`. The `properties` object includes `supportsHttpsTrafficOnly` set to `true`.

Below the editor, the `TERMINAL` tab is active, showing the output of the deployment. The `provisioningState` is `Succeeded`, which is highlighted with a red box. Other details include `templateHash` `"2817830582131658025"`, `timestamp` `"2019-10-29T14:42:01.430811+00:00"`, `resourceGroup` `"FunctionApp-RG"`, and `type` `"Microsoft.Resources/deployments"`.

Let's go and check the resource at the [Azure Portal](#).

On the portal, go to Resource Groups. On this blade, you can see the Resource Group we've created.

Click on the Resource Group and there it's your resources **Resources**:

- App Service plan
- Storage Account
- App Service

The screenshot shows the Microsoft Azure portal interface. On the left is a navigation pane with options like 'Create a resource', 'Home', 'Dashboard', 'All services', and 'FAVORITES'. The main area displays the 'FunctionApp-RG' resource group. Under the 'Overview' tab, there's a table of resources:

Name	Type	Location
hostPlan-uksfunctionapptst	App Service plan	UK South
stoaccmyfunct	Storage account	UK South
uksfunctionapptst	App Service	UK South

The 'uksfunctionapptst' resource is highlighted with a red box. Above the table, there are filters for 'Type == all' and 'Location == all'. Below the table, it says 'Showing 1 to 3 of 3 records'.

Click on the App Service and you will have an overview of your **Function App**.

The screenshot shows the Microsoft Azure portal interface for the 'uksfunctionapptst' Function App. The 'Overview' tab is selected, displaying the following details:

- Status:** Running (indicated by a green checkmark)
- Subscription:** Visual Studio Enterprise – MPN
- Resource group:** FunctionApp-RG
- Subscription ID:** [Redacted]
- Location:** UK South
- URL:** https://uksfunctionapptst.azurewebsites.net
- App Service plan / pricing tier:** hostPlan-uksfunctionapptst (Consumption)

At the bottom, there's a section titled 'Configured features' with a lightning bolt icon, listing 'Function app settings' and 'Configuration'. A message at the bottom states: 'You have created a function app!'.

And that is just the tip of the iceberg. Now you can practice with your functions.

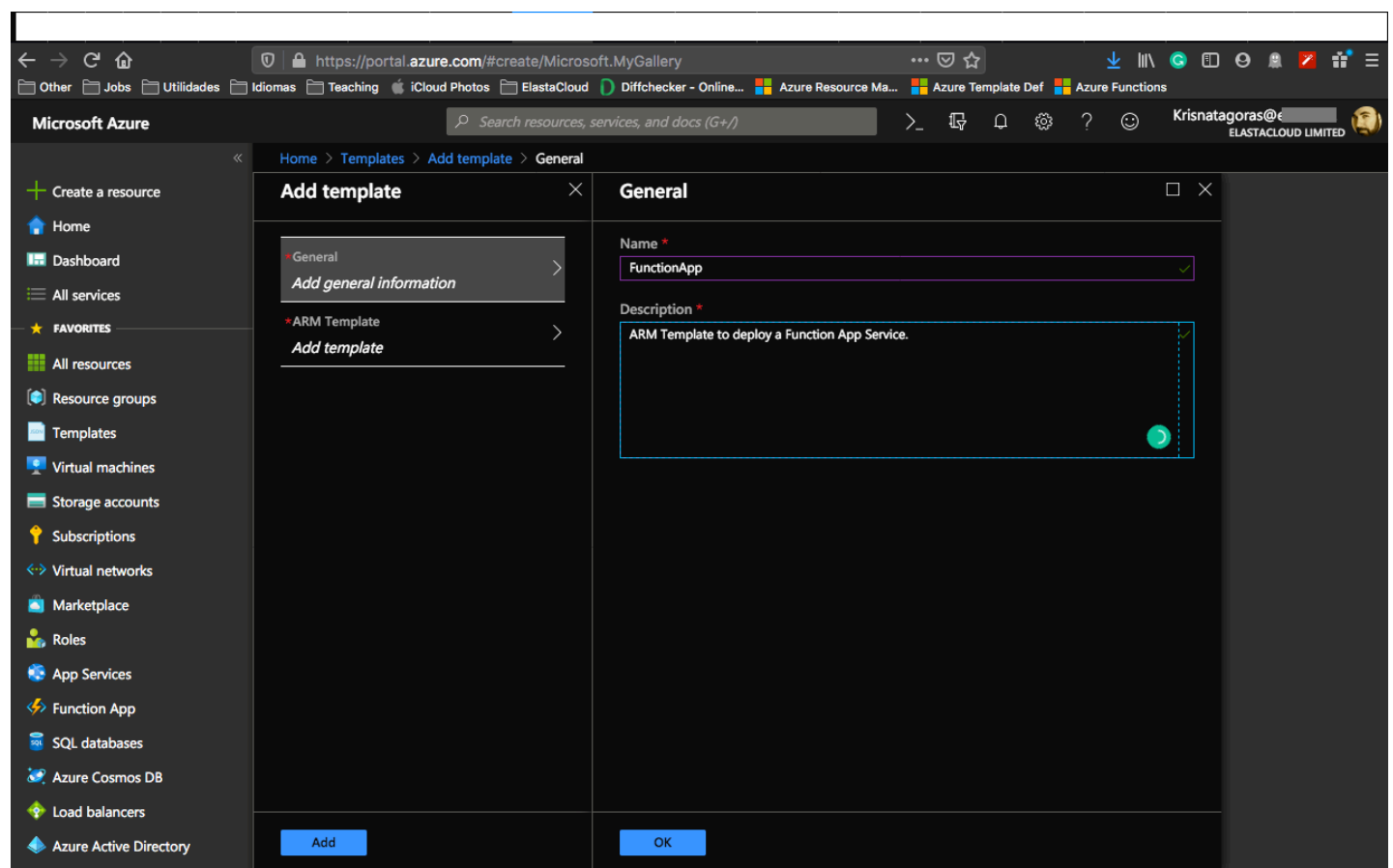
And the most important, don't forget to have fun!

Using the Portal

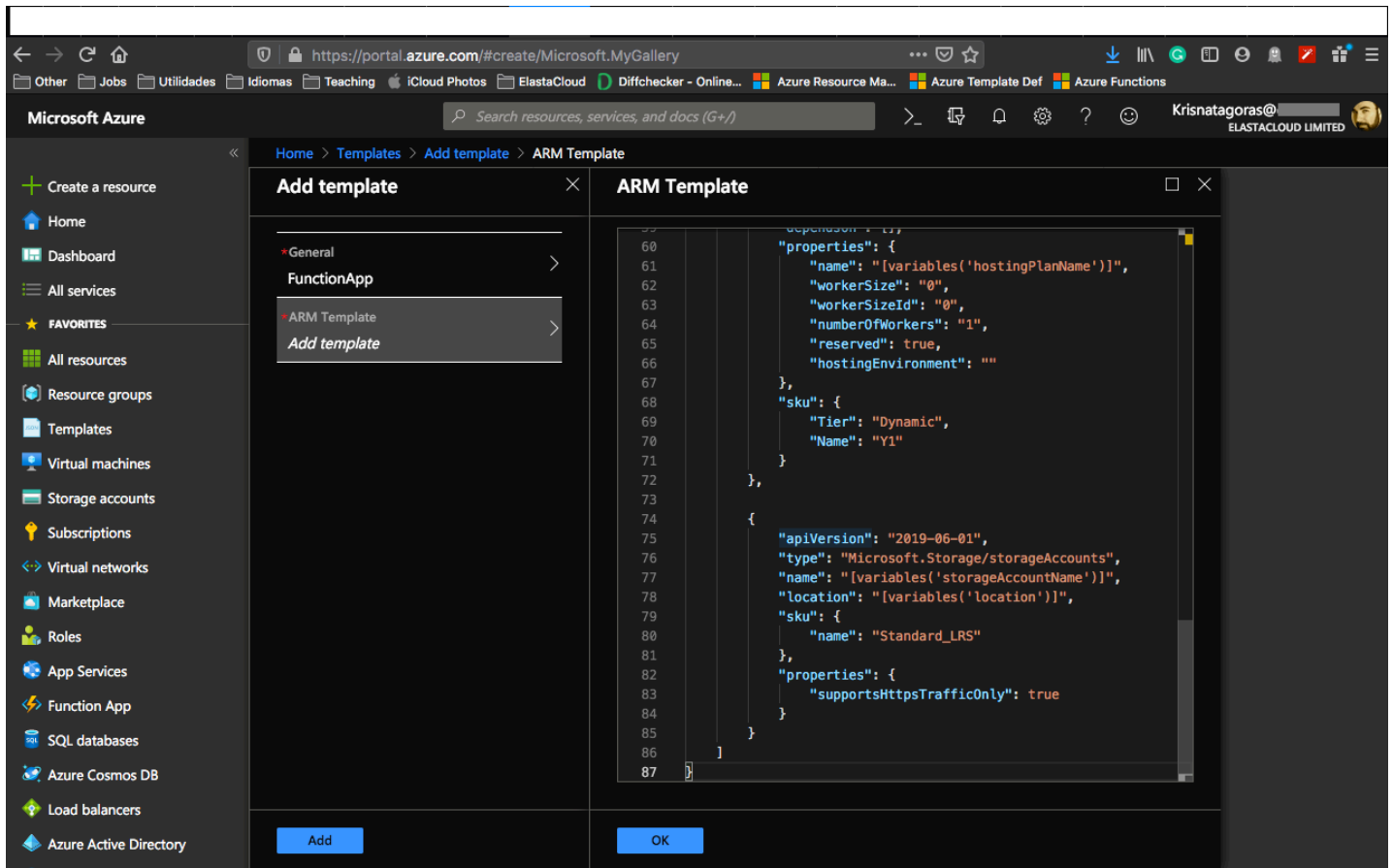
At the Portal, in All Services look for **Templates**, you can favorite this service.

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

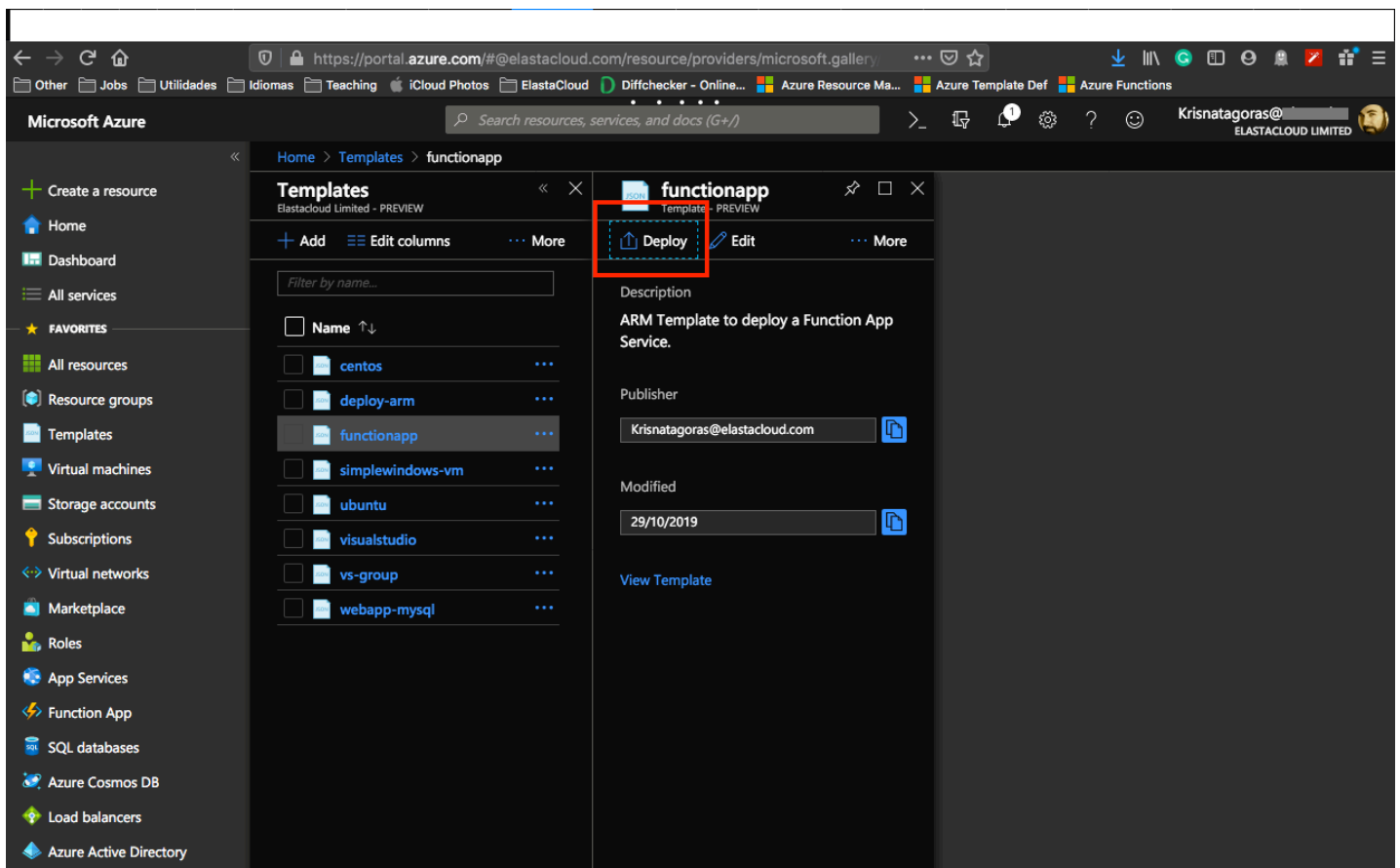
On General, type a name and a description for your template, and click on [OK].



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

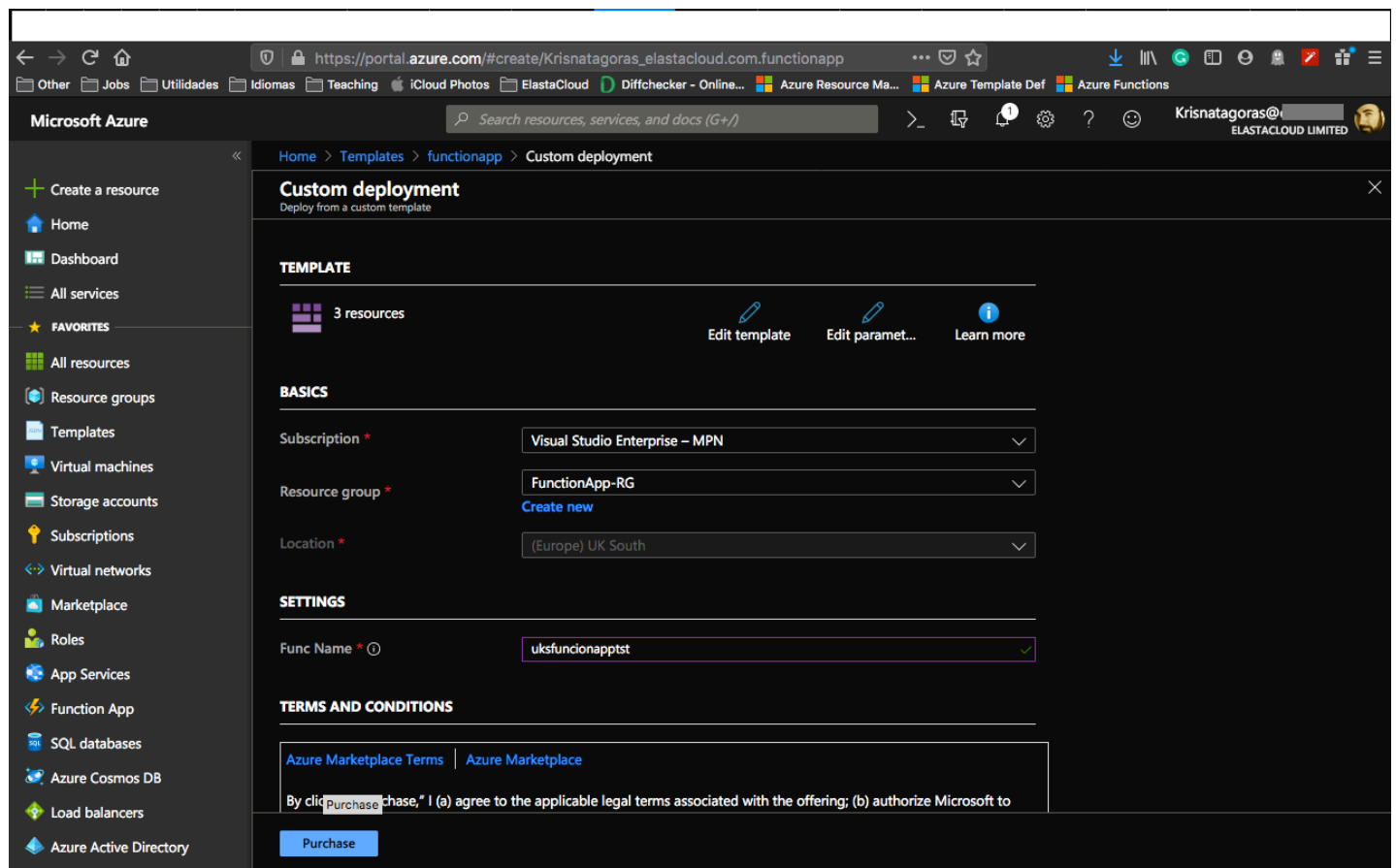


Click on the refresh button and you will find your template. Click on it and then click in [Deploy]



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

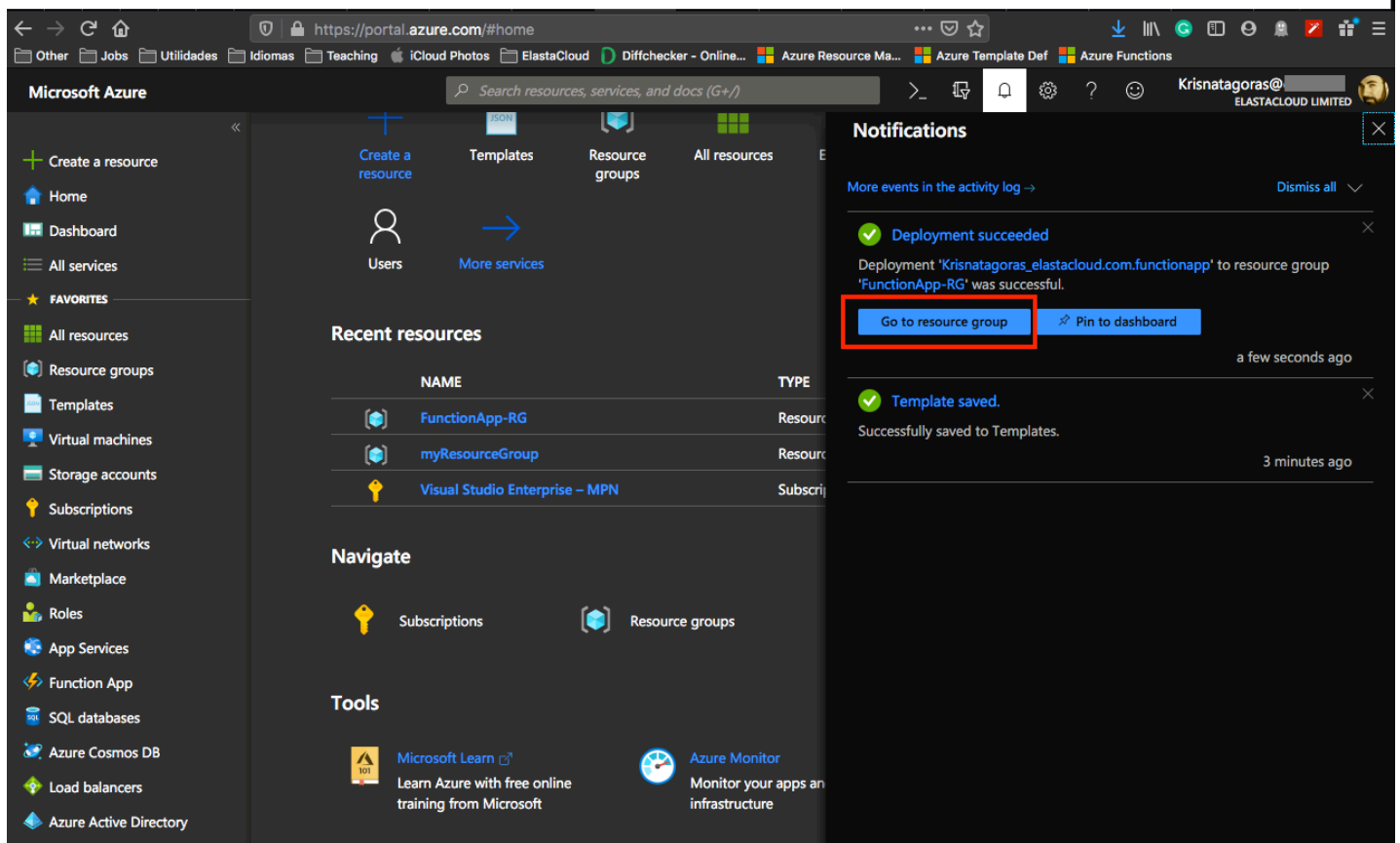
Select [I agree] and click on [Purchase].



The screenshot shows the Microsoft Azure portal's 'Custom deployment' page for a Function App. The page is titled 'Custom deployment' with the subtitle 'Deploy from a custom template'. It features a sidebar with navigation options like 'Create a resource', 'Home', 'Dashboard', 'All services', and 'FAVORITES'. The main content area is divided into sections: 'TEMPLATE' (showing 3 resources with links to 'Edit template', 'Edit paramet...', and 'Learn more'), 'BASICS' (with dropdowns for 'Subscription' (Visual Studio Enterprise – MPN), 'Resource group' (FunctionApp-RG), and 'Location' ((Europe) UK South)), 'SETTINGS' (with a 'Func Name' field containing 'uksfunctionapptst'), and 'TERMS AND CONDITIONS' (with links to 'Azure Marketplace Terms' and 'Azure Marketplace'). At the bottom, there is a 'Purchase' button and a checkbox for 'I agree'.

As you can see, it's deploying.

After a couple of minutes, voilà, you have your Function App deployed.



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: Azure charge you for the resources you are using, and you don't want to finish all your credits at once, right? So, for not running out of credit, don't forget to stop the Function App at the portal or even delete the Resource Group you create to avoid any unnecessary charges.

How to shutdown your resources:

Using the portal:

On the portal, open your Resource Group, if you will not use the Web App anymore, you can just click on the [Delete] Button.

You can also just stop the Web App in case you gonna need the resource. Open the resource and click on Stop.

Microsoft Azure portal interface showing the details of a Function App named **uksfunctionapptst**.

The interface includes a sidebar with navigation options (Home, Dashboard, All services, FAVORITES, All resources, Resource groups, Templates, Virtual machines, Storage accounts, Subscriptions, Virtual networks, Marketplace, Roles, App Services, Function App, SQL databases, Azure Cosmos DB, Load balancers, Azure Active Directory).

The main content area displays the **uksfunctionapptst** Function App details, including:

- Overview** and **Platform features** tabs.
- Stop** and **Delete** buttons (highlighted with red boxes).
- Swap**, **Restart**, **Get publish profile**, **Reset publish profile**, and **Download app content** actions.
- Status**: Running (indicated by a green checkmark).
- Subscription**: Visual Studio Enterprise – MPN.
- Resource group**: FunctionApp-RG.
- Subscription ID**: [Redacted]
- Location**: UK South.
- URL**: <https://uksfunctionapptst.azurewebsites.net>.
- App Service plan / pricing tier**: hostPlan-uksfunctionapptst (Consumption).

The **Configured features** section shows:

- Function app settings
- Configuration

A large blue lightning bolt icon is displayed next to the text: **You have created a function app!**

Just refresh your screen and you are good to go.