CLOUD COMPUTING SYSTEMS

Lab 1

Nuno Preguiça

(nuno.preguica_at_fct.unl.pt)

GOAL

In the end of this lab you should be able to:

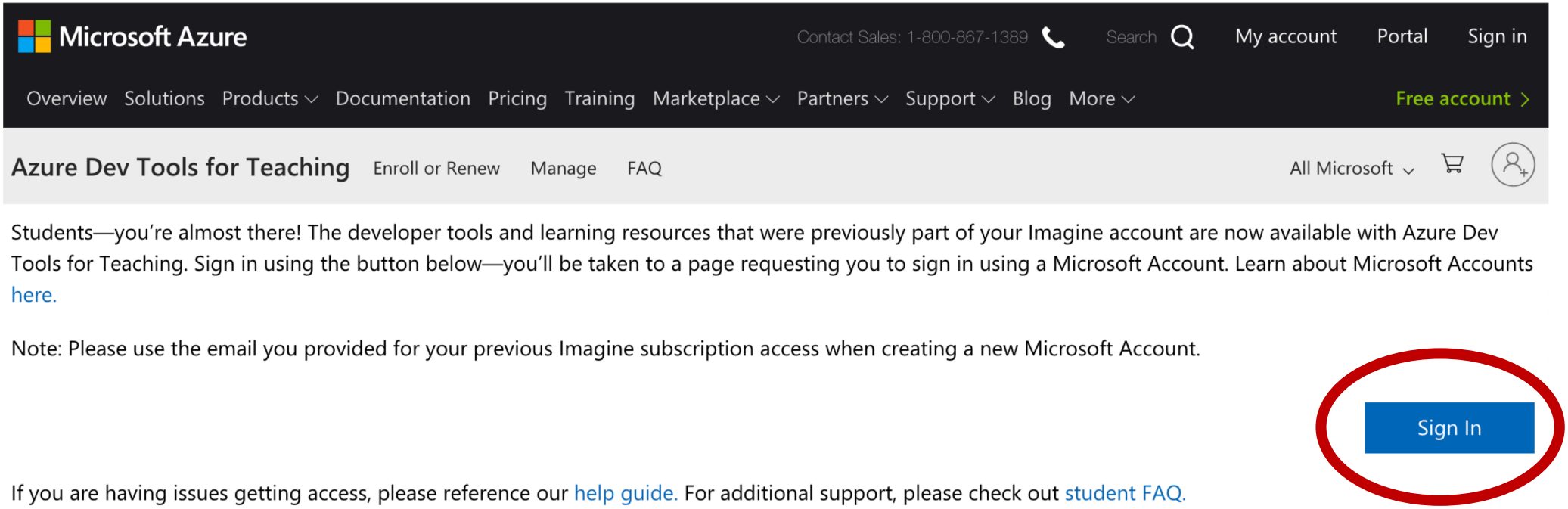
- Create a web application with a REST API and deploy it to the Azure Cloud Platform
- Use Jmeter to evaluate the performance of your REST API (only very trivial invocations)

Create a Azure Student Subscription

Create a Microsoft Azure Dev Tools for Teaching subscription using the FCT protocol

https://azureforeducation.microsoft.com/devtools

NOTE: Use the Sign In button and not the Free Account.



AZURE PORTAL

You can access Azure portal by using the following URL

https://portal.azure.com/#home

You can create and delete resources.

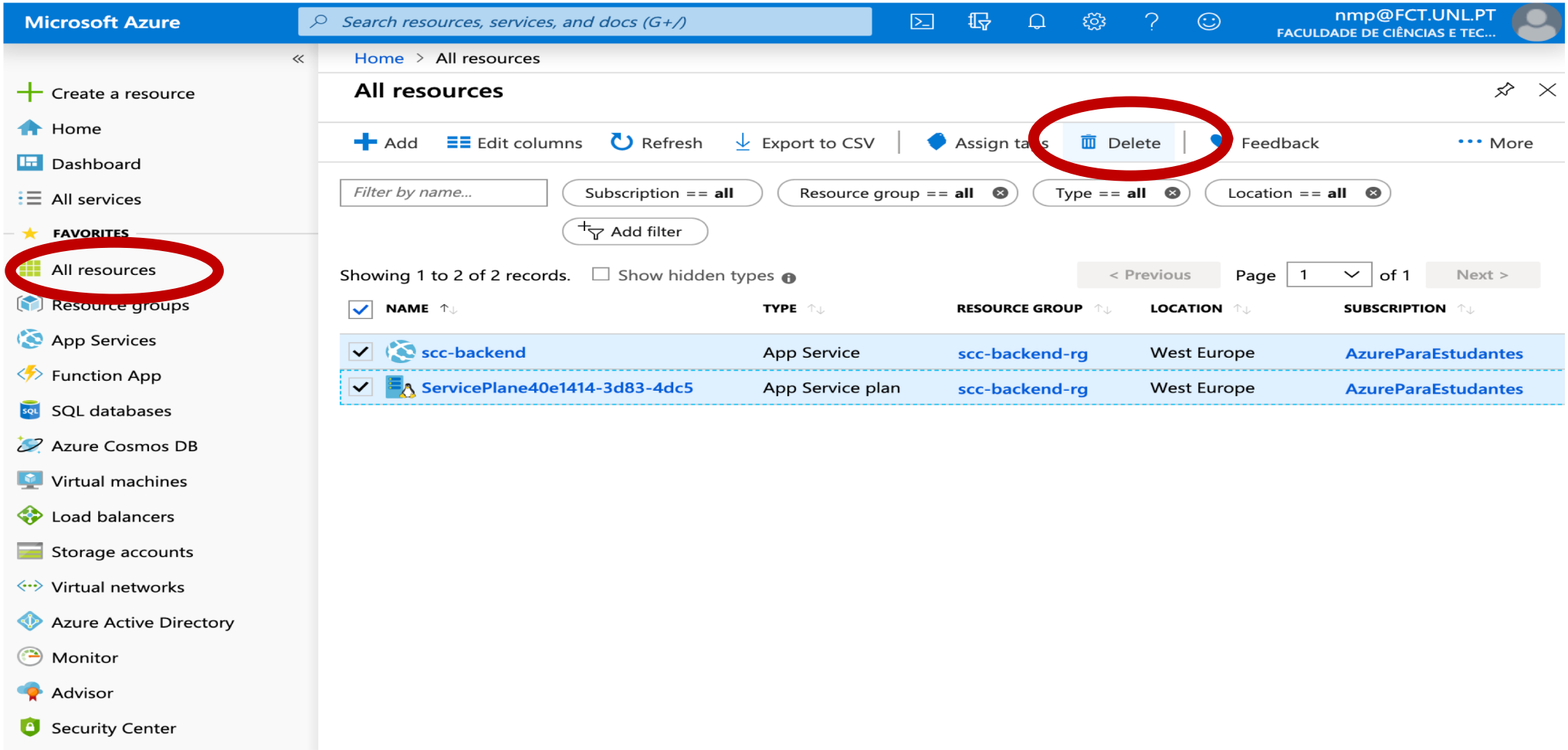
Resources can be VM, Web Apps, Storage accounts, etc.

There are resources that are free and others quite expensive. Use resources carefully to save money.

In this lab, we will only use Web Apps and they will be created using maven. More info on resources in the following labs.

AZURE PORTAL (2)

Delete your resources when you are not using them (to guarantee that you are not spending money).



EXAMPLE: SIMPLE WEB APP

Code available at CLIP: lab1.zip.

Simple web application with a single REST resource

- MediaResource
  - REST resource to store and retrieve multimedia BLOBs (images, videos)
  - **Exercise**: complete the resource for having the following methods
    1. String upload(byte[] contents)
       - Uploads a byte array with the contents of the file
       - Returns a unique identifier for the file (based on an hash of the contents)
       - Should consume MediaType.APPLICATION_OCTET_STREAM
       - NOTE: In this lab, do not store the contents persistently. This will be addressed in lab 2.
    2. byte[] download (String uid)
       - Download the contents of the file, given the uid
       - Should produce MediaType.APPLICATION_OCTET_STREAM

SOME NOTES

The code at CLIP is a Maven project.

The Maven project is configured to use WildFly 14 (with Java 8). This is the version that is available at Azure.

For adding new resources, it is necessary to add the class name to the MainApplication class.

In this course, we will be using standard JAX-RS. Students can use Spring if they prefer to.

Deploy the Web App to Azure

Microsoft has plug-ins for making it easy to deploy Web Apps to the Azure platform using Eclipse (and IntelliJ).

I found the current version of these plug-ins unstable. Use them at your own risk. We will be using the command line and maven for deploying applications.

https://docs.microsoft.com/en-us/java/azure/eclipse/azure-toolkit-for-eclipse-installation?view=azure-java-stable

https://docs.microsoft.com/en-us/java/azure/intellij/azure-toolkit-for-intellij?view=azure-java-stable

Using Maven to deploy to Azure

Requirements:

- Azure CLI
- Maven

Alternatives:

1. Install Azure CLI on your system [preferred]
   https://docs.microsoft.com/en-us/cli/azure/install-azure-cli?view=azure-cli-latest
2. Use the docker image provided

Using Maven to deploy to Azure (2)

Add the following plugin to your project pom.xml:

```
<plugin>
  <groupId>com.microsoft.azure</groupId>
  <artifactId>azure-webapp-maven-plugin</artifactId>
  <version>1.7.0</version>
</plugin>
```

USING THE DOCKER IMAGE

In the parent directory of your project, run the docker image by executing the following command:

```
docker run -it -v $(pwd):/root nunopreguica/ccs-1920
```

NOTE: Replace $(pwd) by the directory path or command for your operating system.

Before configuring and deploying web apps, sign in to Azure by running the command, and selecting the appropriate subscription (if it applies):

```
az login
```

```
az account set --subscription id_returned_in_login
```

Using Maven (Native or inside docker)

Inside the image, change the directory to the project directory.

First, we need to configure the deployment, using the command:

```
mvn azure-webapp:config
```

DEPLOYMENT CONFIGURATION (1)

```
Please choose which part to config
1. Application
2. Runtime
3. DeploymentSlot
[Enter index to use: 1
[Define value for appName(Default: scc-backend):
[Define value for resourceGroup(Default: scc-backend-rg):
[Define value for region(Default: westeurope):
Define value for pricingTier(Default: F1):
1. b1
2. b2
3. b3
4. d1
5. f1 [*]
6. p1v2
7. p2v2
8. p3v2
9. s1
10. s2
11. s3
[Enter index to use:
Please confirm webapp properties
AppName : scc-backend
ResourceGroup : scc-backend-rg
Region : westeurope
PricingTier : Free_F1
OS : Linux
RuntimeStack : WILDFLY 14-jre8
Deploy to slot : false
[Confirm (Y/N)? : Y
```

Application

1. appName, resourceGroup
2. Region: westeurope

   Exercise: try a different region to compare latency.
3. Pricing Tier: for now, let's use the free tier.

DEPLOYMENT CONFIGURATION (2)

```
root@bc58e761a1fa:~/scc-backend# mvn azure-webapp:config
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by com.google.inject.internal.cglib.core.$ReflectUtils$1
int,java.security.ProtectionDomain)
WARNING: Please consider reporting this to the maintainers of com.google.inject.internal.cgl
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access c
WARNING: All illegal access operations will be denied in a future release
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------< pt.unl.fct.di.scc:scc-backend >--------------------
[INFO] Building scc-backend 0.0.1-SNAPSHOT
[INFO] --------------------------------[ war ]---------------------------------
[INFO]
[INFO] --- azure-webapp-maven-plugin:1.7.0:config (default-cli) @ scc-backend ---
Please choose which part to config
1. Application
2. Runtime
3. DeploymentSlot
Enter index to use: 2
[WARNING] The plugin may not work if you change the os of an existing webapp.
Define value for OS(Default: Linux):
1. linux [*]
2. windows
3. docker
Enter index to use:
Define value for javaVersion(Default: Java 8):
1. Java 11
2. Java 8 [*]
Enter index to use:
Define value for runtimeStack(Default: WILDFLY 14):
1. TOMCAT 9.0
2. TOMCAT 8.5
3. WILDFLY 14 [*]
Enter index to use:
Please confirm webapp properties
AppName : scc-backend-1568843803474
ResourceGroup : scc-backend-1568843803474-rg
Region : westeurope
PricingTier : PremiumV2_P1v2
OS : Linux
RuntimeStack : WILDFLY 14-jre8
Deploy to slot : false
Confirm (Y/N)? : Y
[INFO] Saving configuration to pom.
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  20.930 s
[INFO] Finished at: 2019-09-18T21:57:26Z
[INFO] ------------------------------------------------------------------------
```

Runtime

1. OS: Linux
2. Java version: Java 8
3. Server: Wildfly 14

Using Maven (Native or inside docker) (2)

Compile and deploy your application by running:

```
mvn package azure-webapp:deploy
```

This should launch your application on Azure, printing the URL to access it.

GOAL

In the end of this lab you should be able to:

- Create a web application with a REST API and deploy it to the Azure Cloud Platform
- **Use Jmeter to evaluate the performance of your REST API (only very trivial invocations)**

JMeter

Apache JMeter™ application is an open source application designed to load test functional behavior and measure performance.

Download the binaries at:

https://jmeter.apache.org/download_jmeter.cgi

The binaries include scripts to run JMeter easily – located at directory bin.

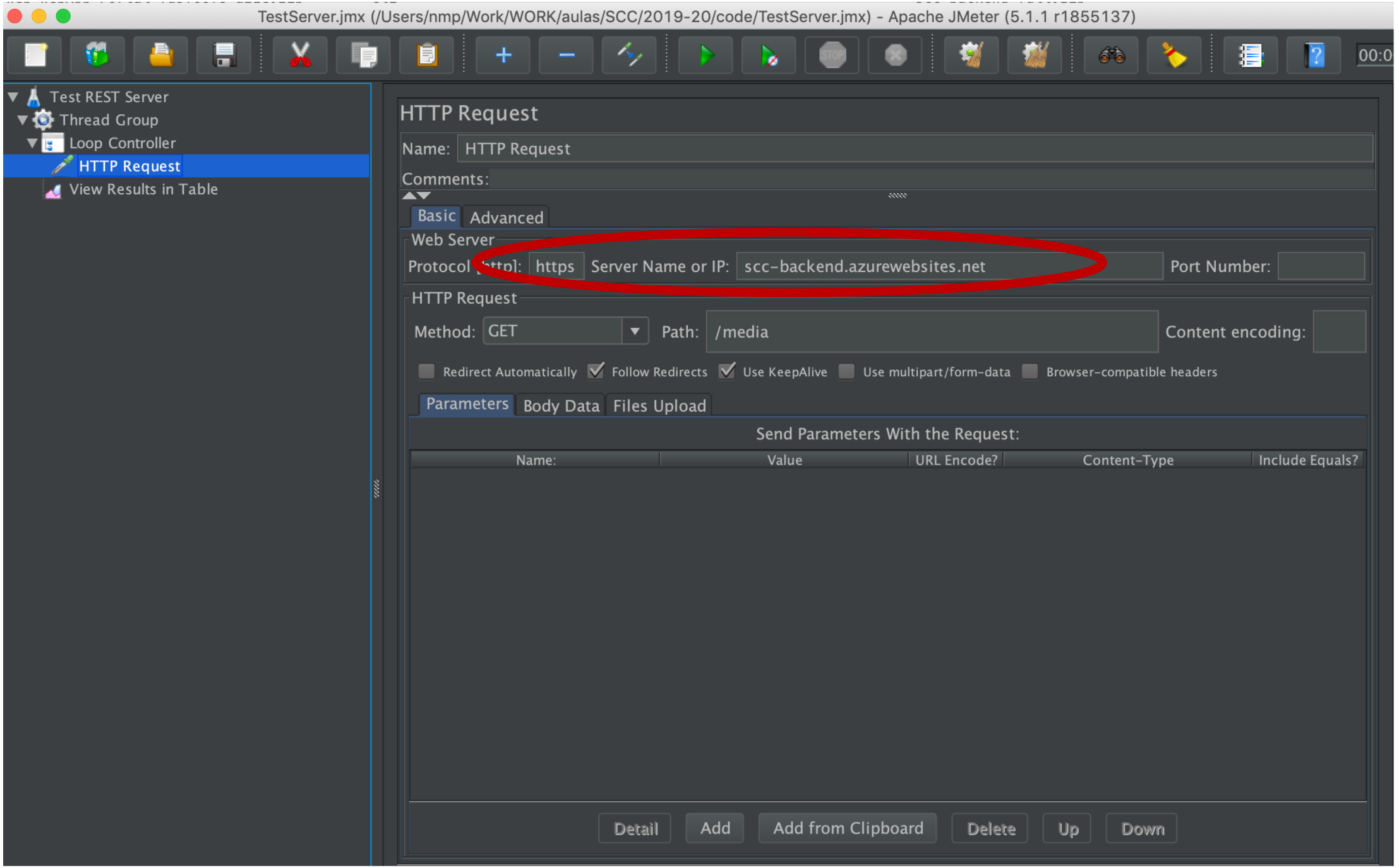
JMeter: some notes

JMeter allows to define a sequence of commands to be executed by a number of concurrent threads.

Today, we will use a simple script that calls a method in your Azure Web App. The script is available at CLIP: TestServer.jmx

The script just call the REST endpoint media 100 times. It displays the results in a table.

NOTE: for accessing your Web App, you will need to change the server address in the HTTP request element.


TestServer.jmx (/Users/nmp/Work/WORK/aulas/SCC/2019-20/code/TestServer.jmx) - Apache JMeter (5.1.1 r1855137)
Test REST Server
Thread Group
Loop Controller
HTTP Request
View Results in Table
HTTP Request
Name: HTTP Request
Comments:
Basic
Advanced
Web Server
Protocol [http]: https
Server Name or IP: scc-backend.azurewebsites.net
Port Number:
HTTP Request
Method: GET
Path: /media
Content encoding:
Redirect Automatically
Follow Redirects
Use KeepAlive
Use multipart/form-data
Browser-compatible headers
Parameters
Body Data
Files Upload
Send Parameters With the Request:
Name:
Value
URL Encode?
Content-Type
Include Equals?
Detail
Add
Add from Clipboard
Delete
Up
Down

JMeter: to know more

Documentation

https://jmeter.apache.org/usermanual/index.html