

# CLOUD COMPUTING SYSTEMS

## Lab 1

Nuno Preguiça, João Resende

([nuno.preguica\\_at\\_fct.unl.pt](mailto:nuno.preguica_at_fct.unl.pt), [jresende\\_at\\_fct.un.pt](mailto:jresende_at_fct.un.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**
- Set-up git for sharing the project with your group members

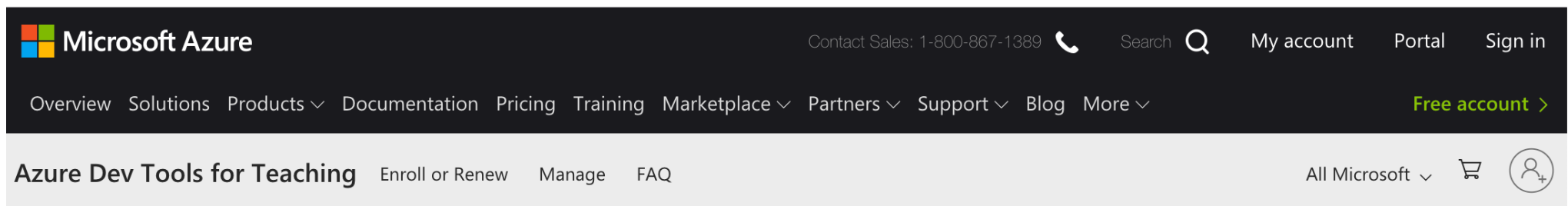
# CREATE A AZURE STUDENT SUBSCRIPTION

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.

You should use you CLIP [id@fct.unl.pt](mailto:id@fct.unl.pt) (not @campus.fct.unl.pt).



The screenshot shows the top navigation bar of the Microsoft Azure website. On the left is the Microsoft Azure logo. To its right are links for 'Contact Sales: 1-800-867-1389', 'Search', 'My account', 'Portal', and 'Sign in'. Below this is a secondary navigation bar with links for 'Overview', 'Solutions', 'Products', 'Documentation', 'Pricing', 'Training', 'Marketplace', 'Partners', 'Support', 'Blog', and 'More'. On the right of this bar is a 'Free account >' link. Below the navigation bars is a section titled 'Azure Dev Tools for Teaching' with links for 'Enroll or Renew', 'Manage', and 'FAQ'. On the right of this section are links for 'All Microsoft', a shopping cart icon, and a user profile icon.

Students—you're almost there! The developer tools and learning resources that were previously part of your Imagine account are now available with Azure Dev Tools for Teaching. Sign in using the button below—you'll be taken to a page requesting you to sign in using a Microsoft Account. Learn about Microsoft Accounts [here](#).

Note: Please use the email you provided for your previous Imagine subscription access when creating a new Microsoft Account.



Sign In

If you are having issues getting access, please reference our [help guide](#). For additional support, please check out [student FAQ](#).

# 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).

The screenshot displays the Microsoft Azure portal interface. The top navigation bar includes the 'Microsoft Azure' logo, a search bar, and user information 'nmp@FCT.UNL.PT'. The left sidebar contains a list of navigation options, with 'All resources' highlighted and circled in red. The main content area is titled 'All resources' and features a toolbar with buttons for 'Add', 'Edit columns', 'Refresh', 'Export to CSV', 'Assign tags', 'Delete' (circled in red), and 'Feedback'. Below the toolbar, there are filter buttons for 'Subscription == all', 'Resource group == all', 'Type == all', and 'Location == all'. The table below shows two resources:

NAME	TYPE	RESOURCE GROUP	LOCATION	SUBSCRIPTION
scc-backend	App Service	scc-backend-rg	West Europe	AzureParaEstudantes
ServicePlane40e1414-3d83-4dc5	App Service plan	scc-backend-rg	West Europe	AzureParaEstudantes

# EXAMPLE: SIMPLE WEB APP

Code available at CLIP: lab1.zip.

Simple web application with two REST resources

- ControlResource
  - REST resource with control methods. There is a single method that prints a version number.
  - This method can be useful to check if the version running at Azure is the latest – deployment may take a few seconds or even minutes.

# EXAMPLE: SIMPLE WEB APP

Code available at CLIP: lab1.zip.

Simple web application with two REST resources

- **MediaResource**
  - REST resource to store and retrieve multimedia BLOBs (images, videos). In this lab, do not store the contents persistently. This will be addressed in lab 2
  - **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`
    2. `byte[] download ( String uid)`
      - Download the contents of the file, given the uid
      - Should produce `MediaType.APPLICATION_OCTET_STREAM`

# EXAMPLE: SIMPLE WEB APP (2)

Code available at CLIP: lab1.zip.

Simple web application with two REST resources

- MediaResource
  - REST resource to store and retrieve multimedia BLOBs (images, videos)
  - **Exercise:** complete the resource for having the following methods
    3. List<String> list()
      - Lists the keys of the resources stored.
      - Should produce MediaType.APPLICATION\_JSON



# SOME NOTES

The code at CLIP is a Maven project.

The Maven project is configured to use Tomcat 10 (with Java 17). 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.

It is possible to launch the server in the local machine before deploying to Azure.

# 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.

We will be using the command line and maven for deploying applications.

<https://docs.microsoft.com/en-us/azure/developer/java/toolkit-for-eclipse/installation>

<https://docs.microsoft.com/en-us/azure/developer/java/toolkit-for-intellij/>

# DEPLOY A SPRING APP

Azure has support for Spring applications. In this course we will support standard Java REST services, using JAX-RS API.

If you prefer to use Spring applications, you are allowed to do it, at your own risk – expect no support on problems.

<https://docs.microsoft.com/en-us/azure/spring-cloud/overview>

# USING MAVEN TO DEPLOY TO AZURE

## Requirements:

- Azure CLI
- Maven

Install Azure CLI on your system

<https://docs.microsoft.com/en-us/cli/azure/install-azure-cli?view=azure-cli-latest>

Follow the steps in the above link to associate your Azure CLI with your Azure account. Without that, you will not be able to deploy applications at Azure.

# INSTALL AZURE CLI (CONT.)

```
(base) LazyMBP:lab1 nmp$ az login
```

The default web browser has been opened at <https://login.microsoftonline.com/common/oauth2/authorize>. Please continue the login in the web browser. If no web browser is available or if the web browser fails to open, use device code flow with `az login --use-device-code`.

You have logged in. Now let us find all the subscriptions to which you have access...

```
[
  {
    "cloudName": "AzureCloud",
    "homeTenantId": "ae7e50a2-ed26-41f7-bd75-f49683f2433a",
    "id": "83abecdf-8b40-49a0-bcae-b5fba4011353",
    "isDefault": true,
    "managedByTenants": [
      {
        "tenantId": "2f4a9838-26b7-47ee-be60-ccc1fdec5953"
      }
    ],
    "name": "Azure para Estudantes",
    "state": "Enabled",
    "tenantId": "ae7e50a2-ed26-41f7-bd75-f49683f2433a",
    "user": {
      "name": "nmp@FCT.UNL.PT",
      "type": "user"
    }
  }
]
```

## USING MAVEN TO DEPLOY TO AZURE (2)

Add the following plugin to your project pom.xml: (the lab1 project is already configured to be deployed)

```
<plugin>  
  <groupId>com.microsoft.azure</groupId>  
  <artifactId>azure-webapp-maven-plugin</artifactId>  
  <version>2.6.1</version>  
</plugin>
```

# USING MAVEN (1)

In your project root directory (i.e., the directory that contains the pom.xml) execute the following command to configure your deployment:

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

# DEPLOYMENT CONFIGURATION (1)

Please choose which part to config [Application]:

\* 1: Application

2: Runtime

3: DeploymentSlot

[Enter your choice:

[Define value for appName [scc-backend-4204]:

[Define value for resourceGroup [scc-backend-rg-4204]:

[Define value for region [westeurope]:

Define value for pricingTier [F1]:

1: B1

2: B2

3: B3

4: D1

5: EP1

6: EP2

7: EP3

\* 8: F1

9: P1v2

10: P1v3

11: P2v2

12: P2v3

13: P3v2

14: P3v3

15: S1

16: S2

17: S3

18: Y1

[Enter your choice:

Please confirm webapp properties

AppName : scc-backend-4204

ResourceGroup : scc-backend-rg-4204

Region : westeurope

PricingTier : F1

OS : Linux

Java : Java 11

Web server stack: Tomcat 9.0

Deploy to slot : false

[Confirm (Y/N) [Y]: y

## *Application*

1. appName, resourceGroup
  - Make sure you change these names – suggestion use your student number as prefix.
2. Region: westeurope

**Exercise:** try a different region to compare latency.
3. Pricing Tier: for now, let's use the free tier (F1).



# DEPLOYMENT CONFIGURATION (2)

Please choose which part to config [Application]:

- \* 1: Application
- 2: Runtime
- 3: DeploymentSlot

Enter your choice: 2

[**WARNING**] The plugin may not work if you change the os of an existing webapp.

Define value for OS [Linux]:

- 1: Windows
- \* 2: Linux
- 3: Docker

Enter your choice:

Define value for javaVersion [Java 11]:

- 1: Java 8
- \* 2: Java 11
- 3: Java 17

Enter your choice: 3

Define value for webContainer [Tomcat 10.0]:

- \* 1: Tomcat 10.0

Enter your choice: 1

Please confirm webapp properties

AppName : scc-backend-XXXXX-YYYYY-ZZZZZ

ResourceGroup : scc-backend-rg-XXXXX-YYYYY-ZZZZZ

Region : westeurope

PricingTier : F1

OS : Linux

Java Version: Java 17

Web server stack: Tomcat 10.0

Deploy to slot : false

Confirm (Y/N) [Y]: y

## *Runtime*

1. OS: Linux
2. Java version: Java 17
3. Server: TOMCAT 10.0

## USING MAVEN (2)

Compile and deploy your application by running:

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

This should launch your application on Azure, printing the URL to access it. Try accessing the **/rest/ctrl/version** endpoint – e.g.:

<https://scc-backend-4204.azurewebsites.net/rest/ctrl/version>

Use some REST client – e.g. Postman – to test other methods (do not forget to complete the download method of the media resource).

# 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
- **Set-up git for sharing the project with your group members**

# GIT

Git is a distributed version-control system for tracking changes in source code.

Designed for allowing a group of developers to collaboratively develop some project.

In the course, we expect all groups to maintain their project in a git repository – you will be asked to share the repository and we expect to see commits from every group member.

P.S.: in the following slides we assume that you will be storing your project in GitHub, but you can use other repositories if you prefer.

# GITHUB: CREATE AN ACCOUNT

Go to [github.com](https://github.com) and create an account (if you still don't have one).

Make sure you apply for the GitHub Students benefits.

# CREATE A REPOSITORY

The screenshot shows the GitHub web interface. At the top, there's a dark navigation bar with the GitHub logo, a search bar, and links for Pulls, Issues, Marketplace, and Explore. On the right of this bar are notification and profile icons. A dropdown menu is open from the plus icon, showing options: 'New repository' (highlighted in blue), 'Import repository', 'New gist', 'New organization', and 'New project'.

On the left sidebar, the user 'preguica' is logged in. Below their name is a 'Repositories' section with a 'New' button and a search bar. A list of repositories is shown, including 'preguica/continuum-2020', 'preguica/sd1920-web', 'albertlinde/papers', 'pfouto/osdi-2020-pfouto', 'pfouto/sd-1920-tp1', 'preguica/sd1920-tester-tp1', and 'mmamarques/TeseMMM47586'. Below this is a 'Your teams' section with a search bar.

The main content area shows recent repository activity. The first entry is 'bieniusa pushed to AntidoteDB/antidote' 8 hours ago, with 2 commits to 'aql-strong-consistency-3-pa'. The second entry is 'bieniusa pushed to AntidoteDB/antidote' 4 days ago, with 1 commit to 'master'. The third entry is 'albsch pushed to AntidoteDB/AntidoteDB-documentation' 8 days ago, with 3 commits to 'master'.


# CREATE A REPOSITORY (2)



## Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner \*

 preguica ▾

Repository name \*

/ scc2021-backend ✓

Great repository names are short and memorable. Need inspiration? How about **solid-spork**?

Description (optional)

☐  **Public**

Anyone on the internet can see this repository. You choose who can commit.

☒  **Private**

You choose who can see and commit to this repository.

Make sure your repository is private

Initialize this repository with:

Skip this step if you're importing an existing repository.

☐ **Add a README file**

This is where you can write a long description for your project. [Learn more.](#)

☐ **Add .gitignore**

Choose which files not to track from a list of templates. [Learn more.](#)

# UPLOAD THE INITIAL CONTENTS

... by running the following commands:

```
git init
git add .gitignore *
git commit -m "first commit"
git branch -M master
git remote add origin URL_of_your_repository
git push -u origin master
```



# SHARING A REPOSITORY WITH OTHER DEVELOPERS

The screenshot shows the GitHub interface for a repository named 'scc2021-backend' by user 'preguica'. The repository is private. The 'Settings' tab is selected in the top navigation bar. On the left sidebar, the 'Manage access' option is highlighted. The main content area is titled 'Who has access' and shows two sections: 'PRIVATE REPOSITORY' and 'DIRECT ACCESS'. The 'PRIVATE REPOSITORY' section states that only those with access can view it, with a 'Manage' link. The 'DIRECT ACCESS' section states that 0 collaborators have access, and only the owner can contribute. Below this, the 'Manage access' section shows a message: 'You haven't invited any collaborators yet' with an 'Invite a collaborator' button.

Options

- Manage access
- Security & analysis
- Branches
- Webhooks
- Notifications
- Integrations
- Deploy keys
- Autolink references
- Secrets
- Actions

## Who has access

**PRIVATE REPOSITORY**

Only those with access to this repository can view it.

[Manage](#)

**DIRECT ACCESS**

0 collaborators have access to this repository. Only you can contribute to this repository.

## Manage access

You haven't invited any collaborators yet

[Invite a collaborator](#)

# CLONE A REPOSITORY

After sharing a private repository with other developer, she can access the repository by cloning it:

```
git clone URL_of_your_repository
```

# GIT COMMANDS

Add all new files:

```
git add *  
git commit -m "some message"
```

Add a new file:

```
git add path  
git commit -m "some message"
```

## GIT COMMANDS (2)

Commit your changes:

```
git commit . -m "some message"
```

Push changes to the server:

```
git push
```

If some change was made to the server, you will need to pull the changes before.

# GIT COMMANDS (3)

Pull changes from the server:

```
git pull
```

If a conflict occurs during a merge, just edit the file and save it. After that, add the file again and commit.

```
git add path
```

```
git commit -m "some message"
```

# EXERCISE

1. Complete REST resource.
2. Set-up the GIT repository for your project.