KU LEUVEN

GROUP T LEUVEN CAMPUS

This lab requires you to submit a report on Toledo before April 4th, 9 AM.

Distributed Systems: Session 3
Cloud-based deployment with IaaS

Master Electronics/ICT – Software Systems

Lab coaches:

- Bert Lagaisse
- Tianyu Li
- Chaomeng Lu

Introduction

In this practical session we will deploy and test our remote services (RMI, SOAP and REST) on a set of virtual machines in the public cloud. More specifically, we will deploy, run and test the services on Azure Virtual machines in data centers across the globe:

- West Europe,
- US east coast,
- US west coast,
- Japan.

Under the Azure Education program from Microsoft, each KU Leuven student has 100$ credits on Azure, and can also use a broad set of services for free. For example, you can run a basic VM with 1 GB of memory and 1 GB of disk space for 750 hours per month. The free services do not charge your 100$ credits.

Activating your Azure Education account

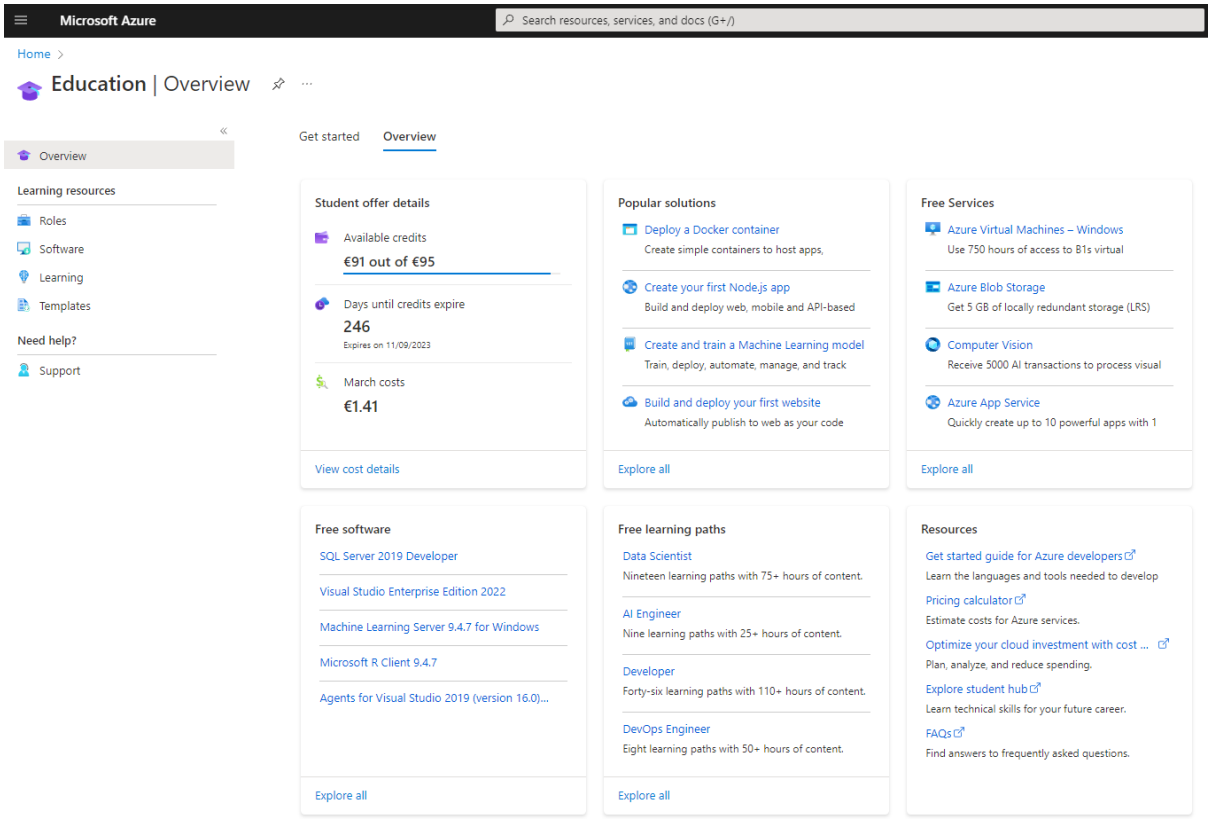
Azure for Education is part of the Microsoft agreement with academic institutions to use certain development tools and Azure cloud services for free. You can find more information about the agreement on

https://icts.kuleuven.be/sc/english/software/microsoft-adtt

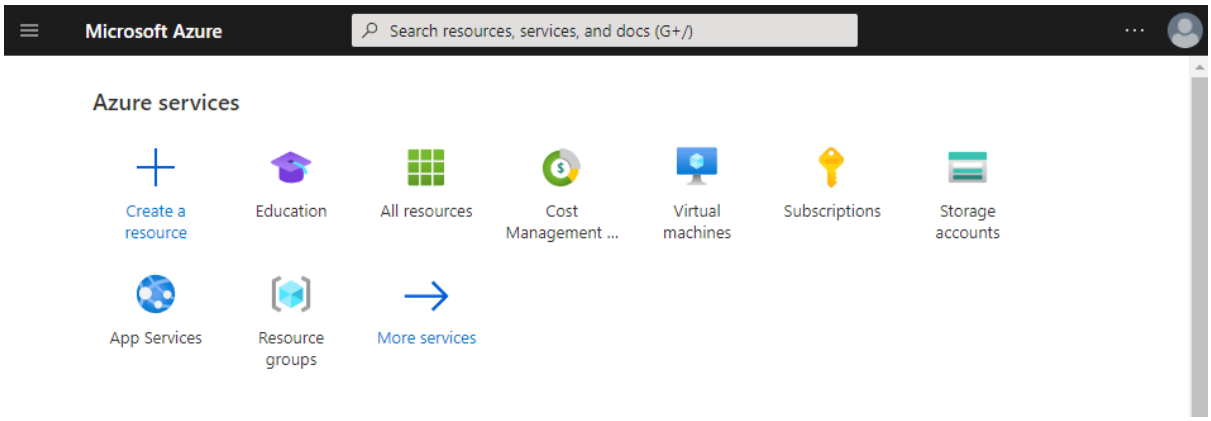
To activate your subscription, you will need to verify as a student with KU Leuven via the startpage:

https://azureforeducation.microsoft.com/devtools

After successfully verifying your association with KU Leuven and logging in, you will be taken to the Education page of the Azure portal:



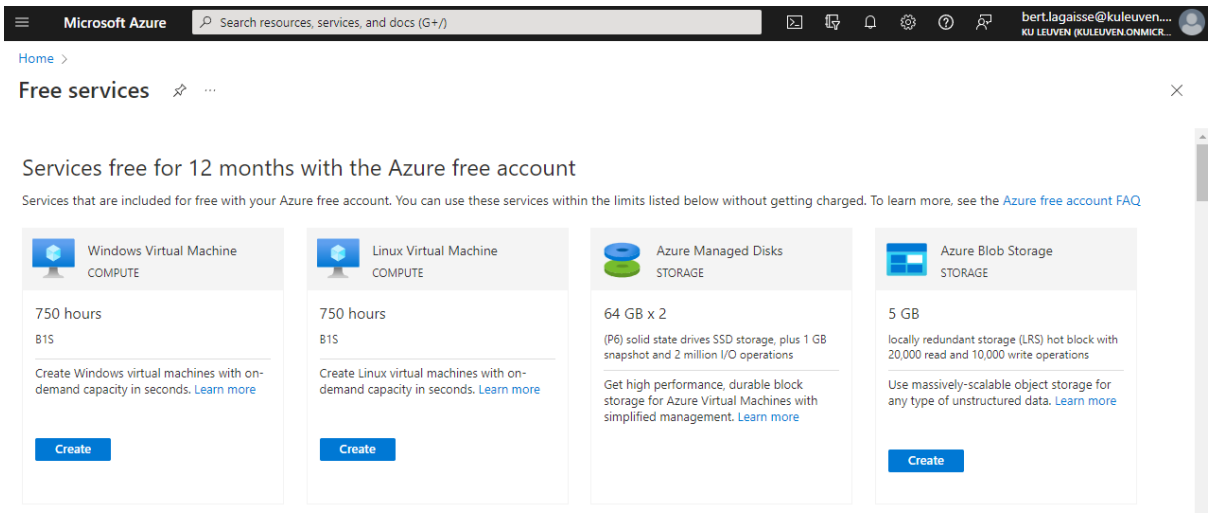
You can always reach this page again by logging in to https://portal.azure.com and navigating to the Education page via the search function or the icon in the services.



On this education page, you will also be given the opportunity to claim the 100$ credits – if you didn't do that yet, which will require you to provide some personal information.

If you navigate to the free services, you will find the opportunity to create a free VM. If you navigate away from the Education page, or from the free services, you can also find some paying services, that will charge your 100$ credits. Keep an eye on your cost management page to know which services you are consuming that will charge your credits. There's an app for that (Azure app).

Under the free services (explore all) we will now create and boot a free Linux Virtual Machine.

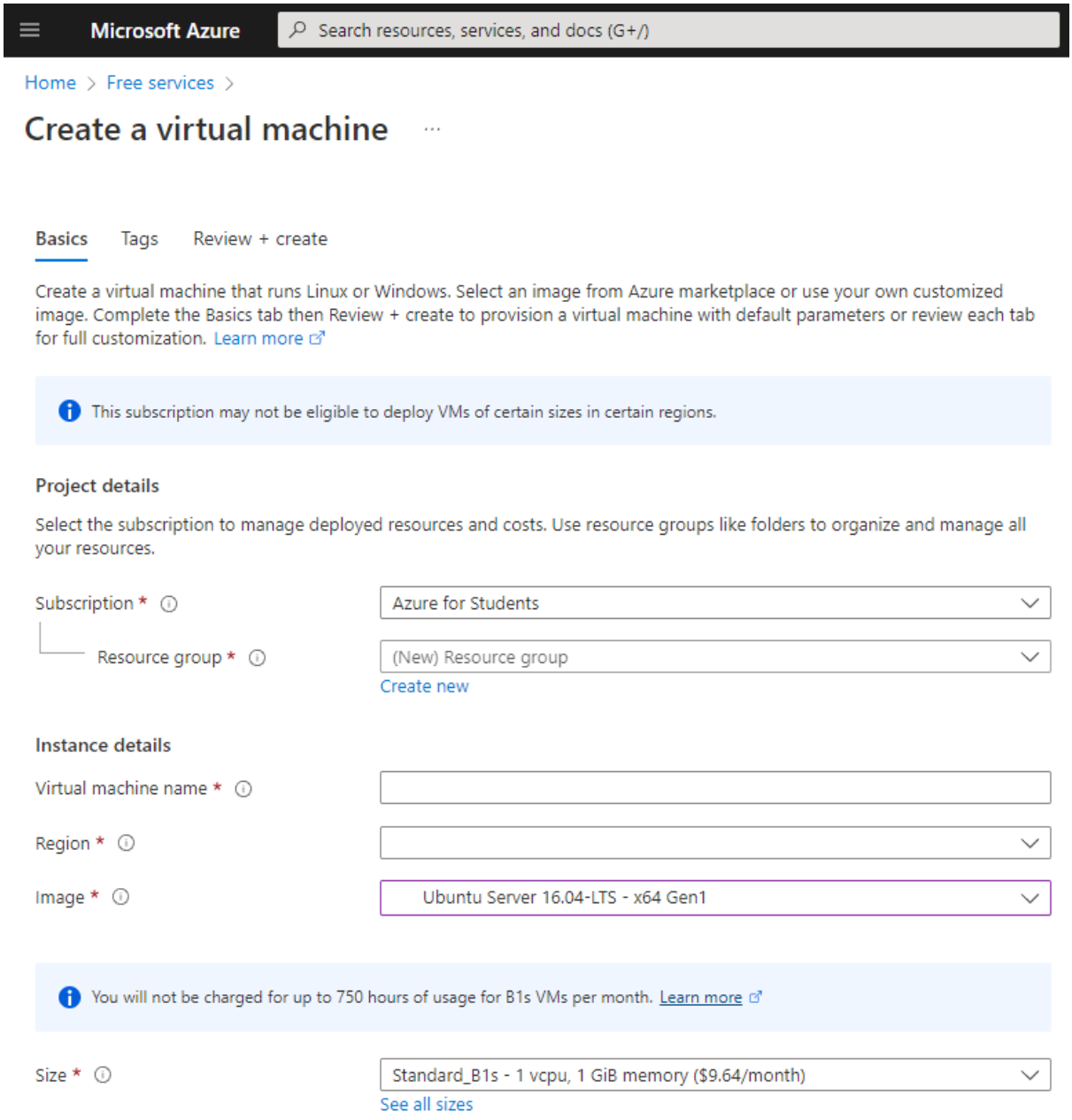


Creating a free VM

Using the webform for creating a virtual machine underneath, you will need to configure the following

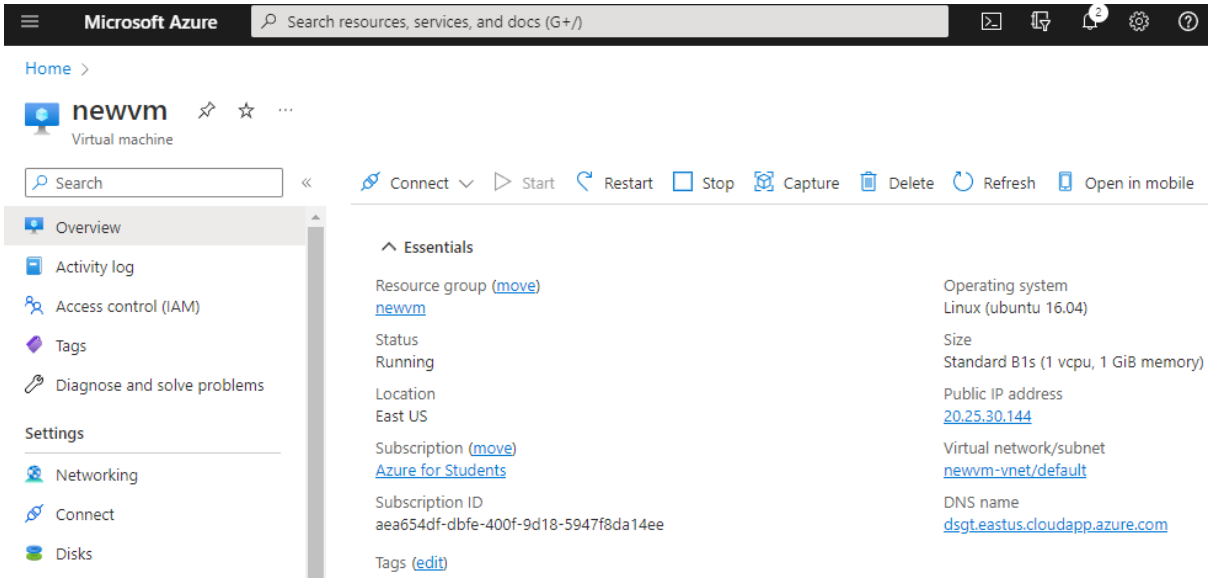
- A name of the resource group for your VM virtual infrastructure (VM, storage account, virtual network, etc will be grouped under that name)
- A name for your VM, e.g. bertvmeastus, or toonvmeurope, or whatever you can remember.
- A region where you want the VM (Azure has data centers all over the world). We will boot some VMs all over the world: West-Europe and US East coast are offering free VMs. Japan and US West coast require a paid VM. Agree with your team members who will boot a VM where.

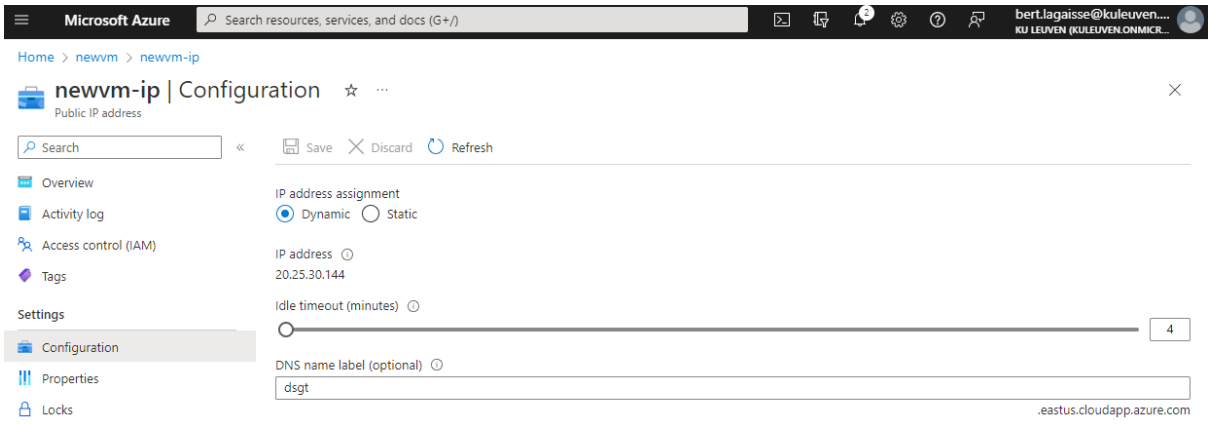
- Choose Ubuntu Server 22 as OS image.
- Leave the size of the VM on the very basic Standard_B1s.
- Choose password as authentication type and pick a username and password you can remember and type easily. The idea is that your team mates can use this password too when necessary.



Choose review and create, and if all input was fine (validation passed), you will now be able to create the VM. The creation might take up to a few minutes.

After the VM has started up, you can go to the resource and configure a DNS name for it, such that you can easily access the VM remotely via ssh.





You should now be able to ssh into your vm with the configured username and password via the configured dns name, e.g:

$ ssh dapp@dsgt.eastus.cloudapp.azure.com

After you have logged in, you can install Java 17 on the VM such that you can deploy and run your remote services (RMI, SOAP and REST).

```
sudo apt-get update
sudo apt-get install openjdk-17-jdk-headless
```

However, to be sure that all your services can run at the same time, you will need to configure and/or recompile your services such that they do not run on the same port (e.g. 8080). Only 1 process can run on a port. Configure ports for:

- Rmi registry
- RMI objects (as parameter of the exportObject function)
- Spring boot SOAP service
- Spring boot REST service

You will also need to open up the ports on the VM, in the networking configuration page of the VM on the azure portal, such that remote clients can access the services on their ports.

In case you export many RMI objects, it's easier to open up all incoming ports, and let RMI generate a port number for the exported RMI objects.

There is one more pitfall. Azure VM's are on an internal network with an internal IP address behind a NAT translation. You already configured a dns and public IP address for the VM, but RMI doesn't know this. It only sees the internal IP address, which is added to the stub in the rmi registry. In order to override this, you can configure the hostname of the RMI server that should be put in the stub that is published in the registry:

```
$ java -Djava.rmi.server.hostname=dsgt.eastus.cloudapp.azure.com server.BookingServer
```

You can set the public ip address or the public dns name of the server as RMI server hostname.

Performance testing

Your main task is now to assess the response time, throughput and performance of your deployed services. With your team, create a testing strategy to **assess response time, throughput and performance**. More specifically, create one or more clients that generate load on your services from different locations in the world, with a different amount of parallel clients, and a different amount of requests per second.

- E.g. from your local laptop to the RMI service in Japan.
- E.g. from your US east coast VM to your US west coast services.
- …

Write a report of 2 to 4 pages, where you describe your testing strategy in terms of scenario's that you tested (client location, service location, parallel clients, requests per second, etc). Be creative and don't settle for a quick and fast solution. Test your services as if they would be running in production, and as if they will be used by many people from all over the world.

The quality manager of your team would for example like to know: how does the REST service in Japan perform when you have an increasing load of requests per seconds from all over the world over the time period of multiple minutes. What is the CPU load and memory usage of the service during that load test ? The quality manager is more interested in statistical results of many requests during multiple test rounds, rather than the result of one single request.

Plot the results in one or more graphs of your choice. Explain the results.

Submit your report on Toledo.

Via the Toledo assignment, each team member should submit the report before April 4th, 9 AM, clearly mentioning:

1) All team members with their student number
2) An active http link to the rest service that lists all meals according to the HATEOAS approach. This rest service should be deployed on one of your free VMs on the public cloud.
3) An active http link to the SOAP wsdl of the SOAP service deployed on one of your free VMs on the public cloud.
4) A clear presentation and critical evaluation of the performance testing strategy and results.