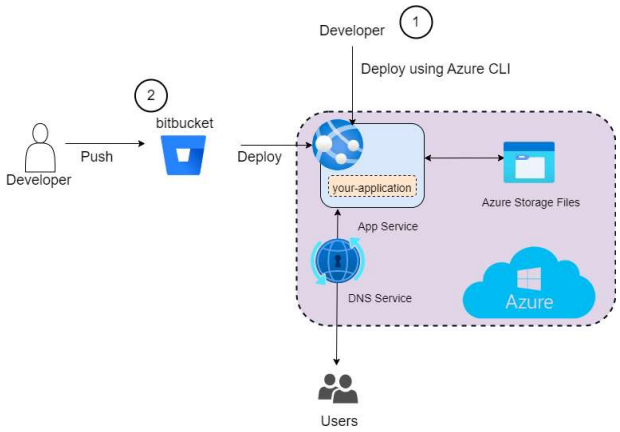
Cloud Computing 2022/23 spring

Practice 3 - Platform as a Service

In this practice session, we will look into the MS Azure Platform as a Service (PaaS). We will look at how to deploy web applications as PaaS applications and explore different features of the PaaS cloud platform.

Applications can be deployed onto PaaS in Multiple ways:

1. Using CLI (Command Line Interface) - Manually deploying the code
2. Git-based (BitBucket, GitHub) - Linking code from your app repository
3. Using docker image registries - Linking a Docker image from a container registry



References

1. Azure Cloud documentation: https://docs.microsoft.com/en-us/azure/?product=featured
2. Azure App Service: https://learn.microsoft.com/en-us/azure/app-service/
3. Azure Storage: https://learn.microsoft.com/en-us/azure/storage/common/storage-introduction

NB! Use ubuntu22.04 if you start a VM in this lab instead of ubuntu 20.04

Exercise 3.1. Getting familiar with the Azure platform and deploying to App Service using docker images

MS Azure cloud can be used with Microsoft account, and students of UT should be able to log in with their existing university account, activate Azure for Students subscription, and use the cloud resources. In this exercise, you will learn about using MS Azure cloud services such as Microsoft Azure App Service.

We will create a docker image for the python flask message-board application (with `data.json` version) and deploy the dockerized application image as a PaaS application in Azure App Service.

Task 3.1.1: Build and push the docker image of the flask message-board application.

- We need a docker-machine to perform this task.

    - You can use your system if docker is installed, or you may create the VM as in Exercise 2.1 of Practise Session 2
        - NB! Use ubuntu22.04 if you start a VM in this lab instead of ubuntu 20.04
- Fork the `https://bitbucket.org/jaks6/cloud-computing-2022-lab-1.git` project to your bitbucket account. (We will update the source code in further exercises)
    - Keep project name as `lab3app`
    - PS!! Make your repository private.
- Clone your project to a local machine or VM
    - PS!! Your account password will not work, so create a separate password to access the projects using the link.
- Build and push the image to the docker hub with tag `your-dockerhub-id/lab3flask1.0`
    - Follow the steps in the previous "Containers: Working with docker" lab

Task 3.1.2: Deploying docker image to Azure App Service.

- Log into Azure **cloud**: https://azure.microsoft.com/en-us/
    - Use your university email and password
    - If asked which subscription type to use, select "Azure for Students subscription."
- Open the Azure **Cloud** portal: https://portal.azure.com/#home
- Familiarize yourself with the available services: https://portal.azure.com/#allservices

Azure App Service is a fully managed platform for building web applications. In this task, we will deploy the message-board application into the App service.

- Search for **App Services** in the azure portal.
- Click on **Create app service**
    - Choose **Resource Group** --> **Create Resource Group**--> Create New --> Name''' : `lab3` as name of group.
    - Choose a name freely (don't use any special characters)
    - Select Publish --> Docker
    - Region --> North Europe
    - Pricing plan : Free F1 (Shared Infrastructure)
- Click on Next: Docker
    - Image Source --> Docker Hub
    - Image and tag --> your-dockerhub-id/lab3flask1.0
- Click on **Review and Create** and than **Create**.
- Now, it will take a few minutes to deploy the application and click on **Go to Resource**
- Go to Overview of your app service and copy the URL (**Essentials-->URL**) and open your application in the browser.
    - It will take a few minutes for the application to run (Ex: 5 to 6 min for me).
    - You can see the logs of the application service **Deployment**--> **Deployment Center**-->**View Logs**
- Once your application is running, then test by inserting a few messages.
- Keep the application running and don't access it until you reach Ex 3.5 (PS!! You can continue with the next exercise, and this is to test the data persistence)

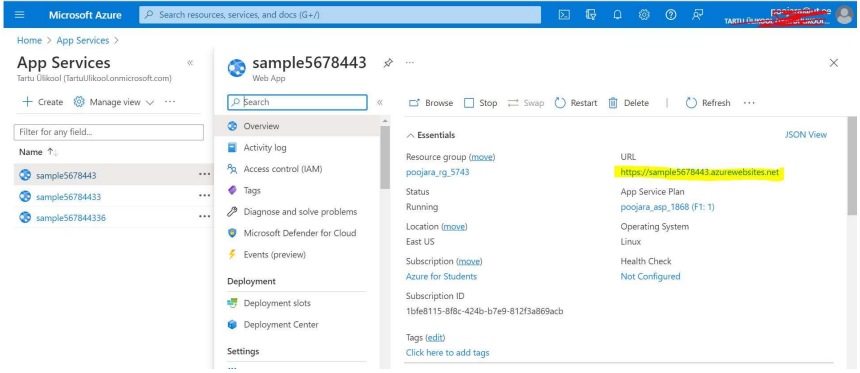
Deliverable: Take a screenshot of the web page of the application. The web page address should be visible.

Exercise 3.2. Deploying python flask application in Azure using CLI

This task mainly helps you to learn the deployment of python applications on the Azure **Cloud** platform CLI. We will deploy our text-file-based Flask message board application.

PS!! You need to use your laptop as a developer machine or a VM to modify the code and to work with Azure services using Azure CLI.

- Let us deploy the flask application into the **Cloud** using Azure CLI. Now, install the Azure CLI on your machine or VM as follows:
    - On Windows:
        - Install Azure CLI using the Windows Package Manager. However, you can use any of the methods used in this document
        - Open Windows Power Shell in administrator mode and run the command `winget install -e --id Microsoft.AzureCLI`
    - On Linux:
        - Install the Azure CLI on Linux machine based on the link document. The straightforward way is to use the command `curl -sL https://aka.ms/InstallAzureCLIDeb | sudo bash`.
- Log into the azure from the command line:
    - Use the command: `az login`
    - Read the response of the command and follow the steps.
- Make sure that you're in the directory `lab3app`.
- Use az cli command to deploy your application into Azure App Service `az webapp up -n <APP_NAME> --sku F1 -g lab3 -l northeurope --runtime "PYTHON:3.9" --logs`
    - In this command, Freely choose the application name, which should be globally unique in azure **cloud**. Ex: shiva198701 (don't use any special characters)
    - The `-sku` specific the link Azure App Service plan. Here, **F1** indicates the Free subscription plan.
    - The `--logs` indicates the logs to be displayed in the terminal.
    - The command `az web up` creates the required resource group and deploys in to the azure app service **cloud**.
    - Now, go to **App Services**.
        - Move to your application. In **Overview** of your application, get the URL (as shown below in the figure) and open the application.



- You should see the flask-based Message application running. Test the application by inserting a few messages.

Deliverable: Take a screenshot of the web page of the application. The web page address should be visible.

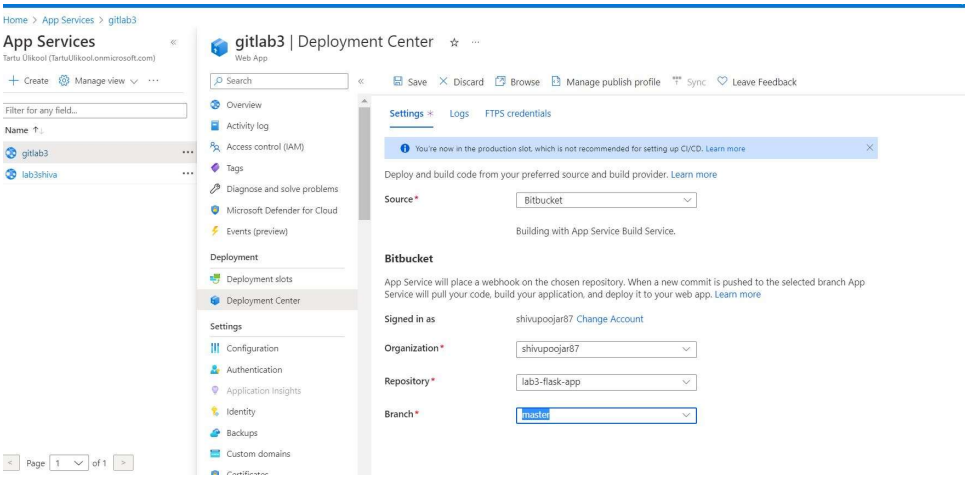
Exercise 3.3. Deploying python flask application in Azure using bitbucket

In this exercise, you will learn to deploy your flask application code from the bitbucket repository to Azure App Service. Here are the following tasks to be performed:

1) Integrate with Azure App Service with bitbucket code.
2) Check for the deployment status.
3) Modify the application, commit to bitbucket code
4) Finally, you should see your application running with modified code.

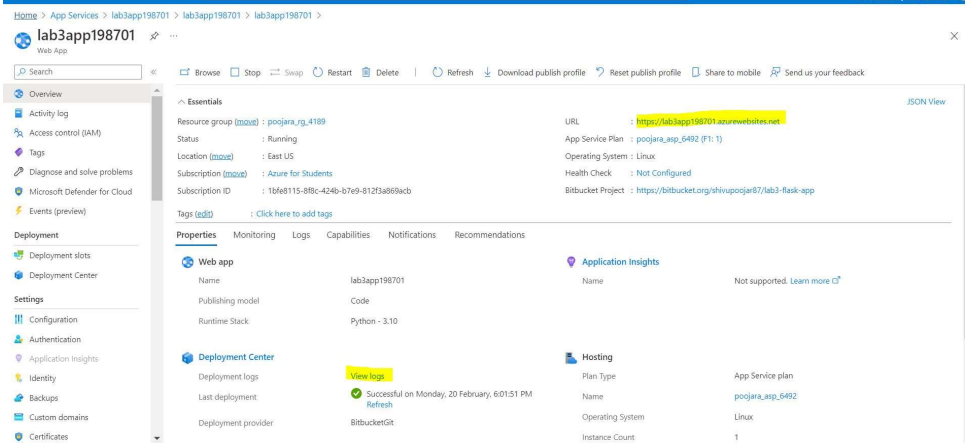
Task 3.3.1: Integrate Azure App Service with bitbucket code.

- Create a New App Service application
  - Resource group: `lab3`
  - Name: `FREELY CHOOSE`
  - Publish: `Code`
  - Runtime Stack: `Python 3.9`
  - Region: `North Europe`
  - Click on Review and Create
- Now, integrate with bitbucket, Go to **Developer Center** in left panel and set the following values
  - **Source**: Select `bitbucket`
  - **Organization** : Provide your bitbucket username
    - Provide permission to access by logging in to bitbucket.
  - **Repository**: Choose your repo `lab3app`
  - **Branch**: `master`
- Finally, Save the configurations by Click on **Save** button at the top.



Task 3.3.3: Check for the status of the deployment.

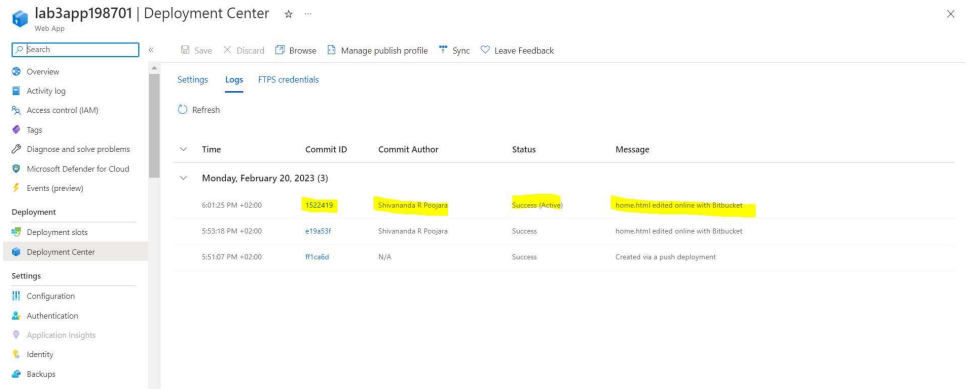
- Click on **View Logs** under Developer Center



- Click on the URL of your project (In overview, as shown in the above figure.), and you should see your application running (It may take a few minutes to get up and running).

Task 3.3.4: Modify the application and commit to bitbucket code.

- Code can be committed using git commands from the system or using bitbucket web interface. However, to commit using git commands with your private repository, needed to setup ssh keys using this document or you can use the bitbucket app password created in the step Ex 3.1 (PS!! Choice is yours)
- Edit the `home.html`, like Practice Session 1, to display your name on the web page (Make something interesting on your web page :)).
- Commit the code.
- Check the logs in App Service in **Developer Center**. You should see the commit status with an update on the application deployment, as shown below.



- You should see your application running with the modified code.

Deliverable: Take a screenshot of the web page of the application. The web page address should be visible.
Question: How long does it take for Azure to update the running application after you commit changes?

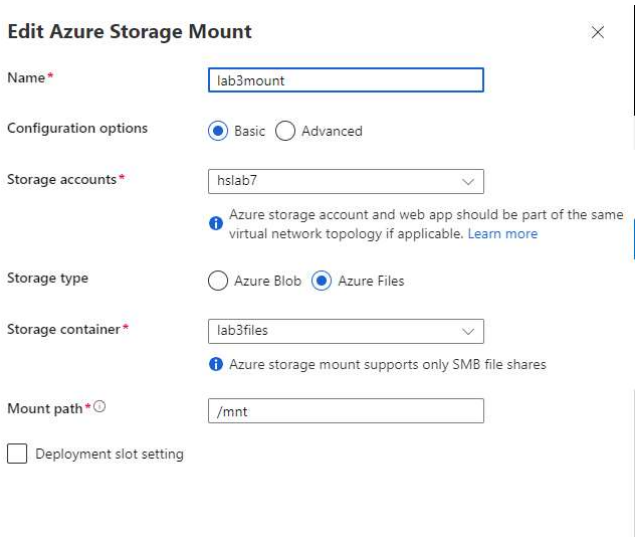
Exercise 3.4. Data persistence using Azure Files

In this task, you will learn to mount external folders and files to the application using the Azure Files service. In the Python Flask application, messages are stored in `data.json` file. To make sure the data is persisted when the container is updated, we will mount an external **cloud** folder into the container and move the `data.json` file there. The application can read and write into files in the mounted folder. Further, we investigate how the data persistence works for two of our applications: deployed in Ex 1 (Docker-based) and Ex 3.3 (bitbucket based).

- Let us create a new Azure Storage Account, which we will use for the persistent **cloud** folders
  - To create the storage account, Search for **storage account** in services overview.
  - Click on create (new storage account) and use the following properties:
    - Storage account name: FREELY CHOOSE a name (PS!! Note it down)
    - **Resource group**: `lab3`
    - **Instance Account Name**: You can FREELY CHOOSE (PS!! Note it down)
    - **Region**: Should be same as bitbucket-based App Service application (By default: North Europe. PS!! This is important)
    - Click Review and Create

- Once the storage account is created, let us create a new File Share inside the storage account.
  - Go to Overview of the just created Storage Account, choose **Data Storage** --> **File shares**
  - Click + **File Share** at top, than add **Name:** `FREELY CHOOSE` (PS!! Note it down) and click on **Create**.
  - Once the file share is created, you must upload the `data.json` file into it. (PS!! data.json can be downloaded from the bitbucket code. It needs to have the same content as in the bitbucket repository. Otherwise, the app will give an error.)
- Now, we will mount the created **File Share** to the App Service application
  - Move on to the bitbucket-based application in App Service and go to Overview page.
  - On left panel, Click on **Settings** --> **Configuration** --> **Path Mappings**
  - Click on + **New Azure Storage Mount** and use the following properties:
    - Name: `FREELY CHOOSE one`
    - Configuration Options: Basic
    - Storage Account: Choose the recently created one
    - Storage Type: Azure Files

        - Storage Container: Name of the File Share created in the previous step
        - Mount Path: `/mnt`
        - Click Ok
    - Click Save



- Let us check the File Share mount in the App Service VM.
    - In App Service Overview, go to **Development Tools**-->**SSH**.
        - Click on **Go**
        - This will open a new window with SSH Connection to App Service VM.
        - In the terminal, list the /mnt files `ls /mnt`. You should see the `data.json`.
- Now, update the bitbucket application source code to modify the path of the data file.
    - In the bitbucket source code, change the path of the `data.json` in - ALL places inside the code - to `/mnt/data.json`.
    - Commit the code and make sure the code is pushed/updated in Bitbucket.
- Now check the application running with modified code and insert a few messages.
- SSH into the Web App Service, and check the content of `data.json` under /mnt.
- Move to the storage account service view and check inside the File Share. Download the data.json for checking whether newly added messages were written there.

Deliverable: Take a screenshot of an overview of File Share created in Azure Storage Account.

Observing the data persistence of the app created in Ex1.

- Now, try to access the application of App Service Created in Ex1, and you may see the messages were deleted because of no new arrivals of user requests and shutting down the container in the background.
- When you access the application later, the container is created with a new data.json file, and previous data was lost.

Observing the data persistence of the app modified in Ex4.

- In this application, `data.json` is mounted to the host file system using Azure files. Hence, data is persistent even though containers were deleted.
- You can observe the behavior by running the application by not accessing it for more than 30 minutes.

Deliverables

- Screenshots from exercises 3.1, 3.2, 3.3 and 3.4
- Answers to question in 3.4:
    - How long does it take for Azure to update the running application after you commit changes?
- Provide application URLs for both of your PaaS applications in the **cloud** (As a comment or as readme.txt file).
- Share your bitbucket repository with the TA (Shivananda, bitbucket (email): shivananda.poojara@ut.ee)
    - Add TA as a member into your bitbucket project with read permissions
    - If this results in an error, you can ignore it. Once TA accepts the invite, error should disappear.
- Delete your VM instance if created in OpenStack **cloud**.

Task: Lab 3 - PaaS

Current submission: ZIP

(23:19 23.02.2023)

If your homework consists of multiple files (for example HTML + CSS + JS) it should be archived before submitting.

File: Choose File | No file chosen

Comment:

How long does it take for Azure to update the running application after you commit changes?

About 1 min 30 sec (measured 2 times)

Submit

In case of issues, check these potential solutions to common issues:

- If you get an error when inviting TA into your Bitbucket repository.
  - You can ignore this error. TA most likely got the invite and needs to accept it before his name is visible there for you. Possibly something recently changed in Bitbucket code and resulting in an error before the invitee accepts the invite. YOu can ask TA by Zulip if you need confirmation your invite succeeded.
- If you get an Error about creating BitBucket code integration (`invalid token, user identity`, ...),
  - Create a brand new App Service from scratch (do not use the previously created CLI version of the App Service)
  - Create a brand new App Service In a **different** region
- If you get an error after running `az webapp up`, like this: `ValueError: invalid literal for int() with base 16: b''`
  - It might be because something failed when fetching app service logs.
  - You should be able to ignore this error. Just check that the App Service was deployed successfully.
- If you can not choose the `Basic` option when creating a `New Azure Storage Mount`
  - Wait for a bit and try again
  - Or use the Advanced option
    - You will need to provide more information manually, including an access key from the Storage Account Azure service. You will find it under Storage account --> Access Keys --> Key1 --> Key (Show)
- If you get the error "We're unable to validate your phone number"
  - It might be a temporary technical issue or rate-limiting block to UT network. Please try again later.
  - What is suggested by the MS portal to do in this case:
    - Check phone number format: Make sure that the phone number is entered in the correct format, including the country code.
    - Check for typos: Verify that there are no typos in the phone number.
    - Verify the phone number: Make sure that the phone number you provided is valid and that you have access to the phone.
    - Check for any blockages: Verify that your phone number is not blocked by your service provider or blocked by any third-party services.
    - Try a different phone number: If the previous steps do not resolve the issue, try using a different phone number to validate your account.
    - Contact Azure support: If you are still unable to validate your phone number, you can contact Microsoft Azure support for further assistance.
  - Can also try Free Trial subscription, but this requires credit card number based validation.
  - If nothing works, contact the lecturer.
- If you get an error when authorizing BitBucket access, something about "Forbidden. CSRF verification failed"
  - Try the same process with a different browser. The issue may have something to do with ad/script blocking in your browser.

The courses of the Institute of Computer Science are supported by following programs: