Endava DevOps Challenge

Challenge specifications:

https://github.com/Endava-Sofia/endava-devops-challenge

Table of contents

Task breakdown

1. Create a public GitHub Repo
2. Find an appropriate Python app with MySQL DB to deploy to the infrastructure
   - Source 1: https://code.tutsplus.com/tutorials/creating-a-web-app-from-scratch-using-python-flask-and-mysql--cms-22972
   - Source 2: https://github.com/jay3dec/PythonFlaskMySQLApp---Part-1

3. Create a free Azure account
4. Install and configure Azure CLI:
    a. Source: https://docs.microsoft.com/en-us/cli/azure/install-azure-cli-windows?tabs=azure-cli
5. Configure the Azure account:

    ```
    az login
    ```

    and verify:

    ```
    az account list
    ```

6. Deploy the App on a VM
    a. Set up auto-restart on failed services
7. Deploy a MariaDB Azure service
8. Deploy a monitoring system - NetData
    - Source: https://learn.netdata.cloud/guides/deploy/ansible
9. Deploy a Load Balancer service
10. Write detailed documentation of the process
11. Present a working demo

Setup Process

I. Install Ansible

- Source: https://docs.microsoft.com/en-us/azure/developer/ansible/install-on-linux-vm?tabs=azure-cli#ansible-210-with-azureazcollection

Connect to the Ansible host server via SSH

```
ssh root@____HOST____ -p22
```

Deploy the following script to the Ansible master server

```
mkdir ~/ansible
cd $_
cat <<'EOF' >> ~/ansible/setup.sh
#!/bin/bash

# Update all packages that have available updates.
sudo yum update -y

# Install Python 3 and pip.
```

```
sudo yum install -y python3-pip

# Upgrade pip3.
sudo pip3 install --upgrade pip

# Install Ansible.
pip3 install "ansible==2.9.17"

# Install Ansible azure_rm module for interacting with Azure.
pip3 install ansible[azure]
sho
# Adding the Ansible binary to the OS Path
export PATH=$PATH:/usr/local/bin/ansible
EOF
sh ~/ansible/setup.sh
```

Run the Ansible installation script

```
sh ~/path/to/script.sh
```

Note: This script also installs the "azure_rm" Ansible module which is required for completing this task.

Installing additional Ansible modules for the task

Azure collection

- Docs: https://docs.ansible.com/ansible/latest/collections/azure/azcollection/index.html#plugins-in-azure-azcollection
- Source: https://galaxy.ansible.com/azure/azcollection

```
ansible-galaxy collection install azure.azcollection
pip3 install -r requirements-azure.txt
```

Note: The 'requirements-azure.txt' file is not included in this project as it can be found via the source link above

MySQL collection

- Docs: https://docs.ansible.com/ansible/latest/collections/community/mysql/mysql_db_module.html#examples
- Source: https://galaxy.ansible.com/azure/azcollection

```
ansible-galaxy collection install community.mysql
```

Installing the Azure CLI and connecting to the Azure account

Installing Azure CLI

- Source: https://docs.microsoft.com/en-us/cli/azure/install-azure-cli

Note: The installation was completed on a local PC, not the Ansible Master server

Connecting to an Azure account

- Source: https://docs.microsoft.com/en-us/cli/azure/install-azure-cli-windows?tabs=azure-cli

After the installation has been completed, simply run:

```
az login
```

Creating a "Security Principal"

- Source: https://docs.microsoft.com/en-us/powershell/azure/create-azure-service-principal-azureps?view=azps-6.3.0
- Explanation: An Azure service principal is an identity created for use with applications, hosted services, and automated tools to access Azure resources.

```
az ad sp create-for-rbac --name ____NAME____
```

Important: This command will output a password which will be used later. Make sure to store the password somewhere safe as it can't be retrieved afterwards.

Retrieving authentication data

- Source: https://docs.microsoft.com/en-us/cli/azure/create-an-azure-service-principal-azure-cli

Simply run this command:

```
az ad sp list --display-name EndavaProjectServicePrincipal
```

And then find the following keys in the results:

```
az ad sp list --display-name EndavaProjectServicePrincipal | findstr "appOwnerTenantId | appId | servicePrincipalNames"
```

This command should output three results - these are necessary for authenticating the Azure account within Ansible.

Creating environment variables for Azure authentication

Coming from the last two steps, we need to run the following commands on the Ansible Master server via a SSH connection:

```
export AZURE_SUBSCRIPTION_ID=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export AZURE_CLIENT_ID=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export AZURE_SECRET=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
export AZURE_TENANT=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
```

Here we need to replace the "x-es" in the code fragment above with the output from the last two steps, where:

```
AZURE_SUBSCRIPTION_ID = appId
AZURE_CLIENT_ID = servicePrincipalNames
AZURE_SECRET = "The password you received when creating the Service Principal"
AZURE_TENANT = appOwnerTenantId
```

Lastly, we need to assign a "Contributor" role to the "Service Principal" on the local PC:

```
az role assignment create --assignee xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx --role "Contributor"
```

Where you have to replace the "x-es" with the "servicePrincipalNames" ID

II. Cloning the Ansible GitHub repo

Once Ansible has been installed and the Azure connection set up, we need to clone the Ansible project to the Ansible Master server:

```
cd
git clone https://github.com/TodorDimitrovIvanov/Project-Endava
```

III. Running the Ansible playbooks

Setting up the infrastructure

The first thing we need to complete is set up the Azure infrastructure. This includes:

- Resource Group
- Virtual Network
- Subnet
- Public IP Address
- Network Interface
- Network Security Group
- Virtual Machine
- MariaDB Server
- MariaDB Database
- MariaDB Firewall

All of this is done automatically by the "playbook/infrastructure-init.yaml" playbook:

```
ansible-playbook playbook/infrastructure-init.yaml
```

Note: Here you will be asked to provide a password for the VM admin user and the MySQL admin user.

During the process, the script will output the Public IP Address that was assigned to the VM and the Hostname of the MariaDB server.

```
TASK [Output public IP Address] ************************************************************************************
ok: [localhost] => {
    "msg": "The public IP is 20.79.205.82"
}
```

```
TASK [Print the URL of the MySQL server] ***************************************************************************
ok: [localhost] => {
    "msg": "The URL of the MySQL Host server is webapp-mariadb-server.mariadb.database.azure.com"
}
```

Adding the new servers to the Ansible hosts file

Using the information in the last step, we need to set the **/etc/ansible/hosts** file like this:

```
---
azure-app-vms:
  hosts:
    vm1:
      ansible_host: ___VM_IP___
      ansible_connection: ssh
      ansible_port: 22
      ansible_user: todor
      ansible_ssh_private_key_file: /root/endava-keys/endava-webapp

azure-db-servs:
  hosts:
    serv1:
        ansible_host: ___MYSQL_HOSTNAME___
```

Note: You can find a copy of this hosts file within the "configs/hosts" file in the GitHub repo.

As you can see in this extract, the VM uses key authentication (password auth is disabled) so you need to add the following private key to your SSH agent:

```
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAABFwAAAAdzc2gtcn
NhAAAAAwEAAQAAAQEA6GIqPqXHJOaWInKOgyhZTL6npHTlxIPPgdu2ok1CONRgiszWBL7x
ixBaDPxbptGLad40YWPui7REznf5TdKHEttKufb+IZi6vaW853rJn4/DiApLcfjRmpFlzX
iYHbcSQM3YOxWO2TnDNtKJvB7tcZ1sYeJS/nqfmj1MXAO0tnLk16Dsq3c+ZbOjLHgWT/xc
wt5zpFhpSCoAbmssVc1glcPniRkngvXogRRsMG3oY+ObsvbH1QjFLKl3sc7hEqoeepniHv
0jq3bog1pIB8MezsGTf+AAVlgGn/kbTzo51H5P+aLnyNDX0WFqw3pqVhkNZOhRS6HCRpz/
xaAnDMHg0QAAA8idS4menUuJngAAAAdzc2gtcnNhAAABAQDoYio+pcck5pYico6DKFlMvq
ekdOXEg8+B27aiTUI41GCKzNYEvvGLEFoM/Fum0Ytp3jRhY+6LtETOd/lN0ocS20q59v4h
mLq9pbznesmfj8OICktx+NGakWXNeJgdtxJAzdg7FY7ZOcM20om8Hu1xnWxh4lL+ep+aPU
xcA7S2cuTXoOyrdz5ls6MseBZP/FzC3nOkWGlIKgBuayxVzWCVw+eJGSeC9eiBFGwwbehj
45uy9sfVCMUsqXexzuESqh56meIe/SOrduiDWkgHwx7OwZN/4ABWWAaf+RtPOjnUfk/5ou
fI0NfRYWrDempWGQ1k6FFLocJGnP/FoCcMweDRAAAAAwEAAQAAAQEAlvQNlRyDPIk9Ras8
wB5Zw+ylYcM97mXhSe27ubqOJfRvsjpVXEfdCATbOkxEDR1ZfrAFPq53qvCzMZvlqpPzgm
BJLnwvoQthYwhiqcutZaUx0xE86RBEuC9y/gI0bjkZ9lbA+8CJRrqrANXK+xTsET29xegL
GZv2SW9ARxVxMoI7EZzQm1Mxxrj297VFhw/eZMyFpcCndrf5VmJg+nc5cEfwt/ggNl69fm
UD4E4qwfyntiEUVkyClZHtCpn74vOSWuZ9KZHVlt3kSyJuIN3IU8wiJodgjFPtD+FX7aT9
AkOW0mIxvm2b6Co7gzn+kcFagThaDDZr4JPZm2TxX1TvcQAAAIB0AyLce1BUlxXUHUvcTc
ewPPu/RRRN15fCp7LmIp/yq8LURqIRxwi/bv4bS1S+FCXf1dwOobAThbb/UUsT+aYny7T9
7s2cnMYULJsNlEVRJUGuo+88OIEDOlP7VvwCuyDCI1jRVNOSf5KaW2RHY5OdgCumjhkPcH
LyoKIQ+uGvqQAAAIEA99I2ElU49N+gNRoXc/J7UOLoXUufitXG7gZqGkjL2HJmedwEBZVR
owNMctAbpNeOhXLhwECBoBlhQE2iryv7tNNst4doOm7BdwQQa7na06MVVOoC5cbQbtq/SE
/3Newy2vewtlby3I98n3lVcDZI+5oLJ9m6Pr25qR2DMDmbqrsAAACBAPANhioGXP8pymr3
ANim1Zv5fQk5S2s1/gwjYbJchnOHAoAsds/qVCBewuKdI9izwmY2BejFhP+vEpwZfidiV8
```

```
nmVeNq9/3xYHGcfzo2gQZ9HWE69PI5lYkTMUMp/8UoGtCK+NTDb5xO3qNspucNK//6O1v6
SLyq4IkBQ344NSfjAAAADnJvb3RAbG9jYWxob3N0AQIDBA==
-----END OPENSSH PRIVATE KEY-----
```

```
ssh-add ~/path/to/private.key
```

Deploying the MySQL database

This is a very straightforward process - simply run:

```
ansible-playbook playbooks/mysql-setup.yaml
```

Note: You'll be asked to provide the MySQL password you entered during the infrastructure setup along with the MySQL URL hostname from the output.

Deploying the Python web app and a Nginx web server

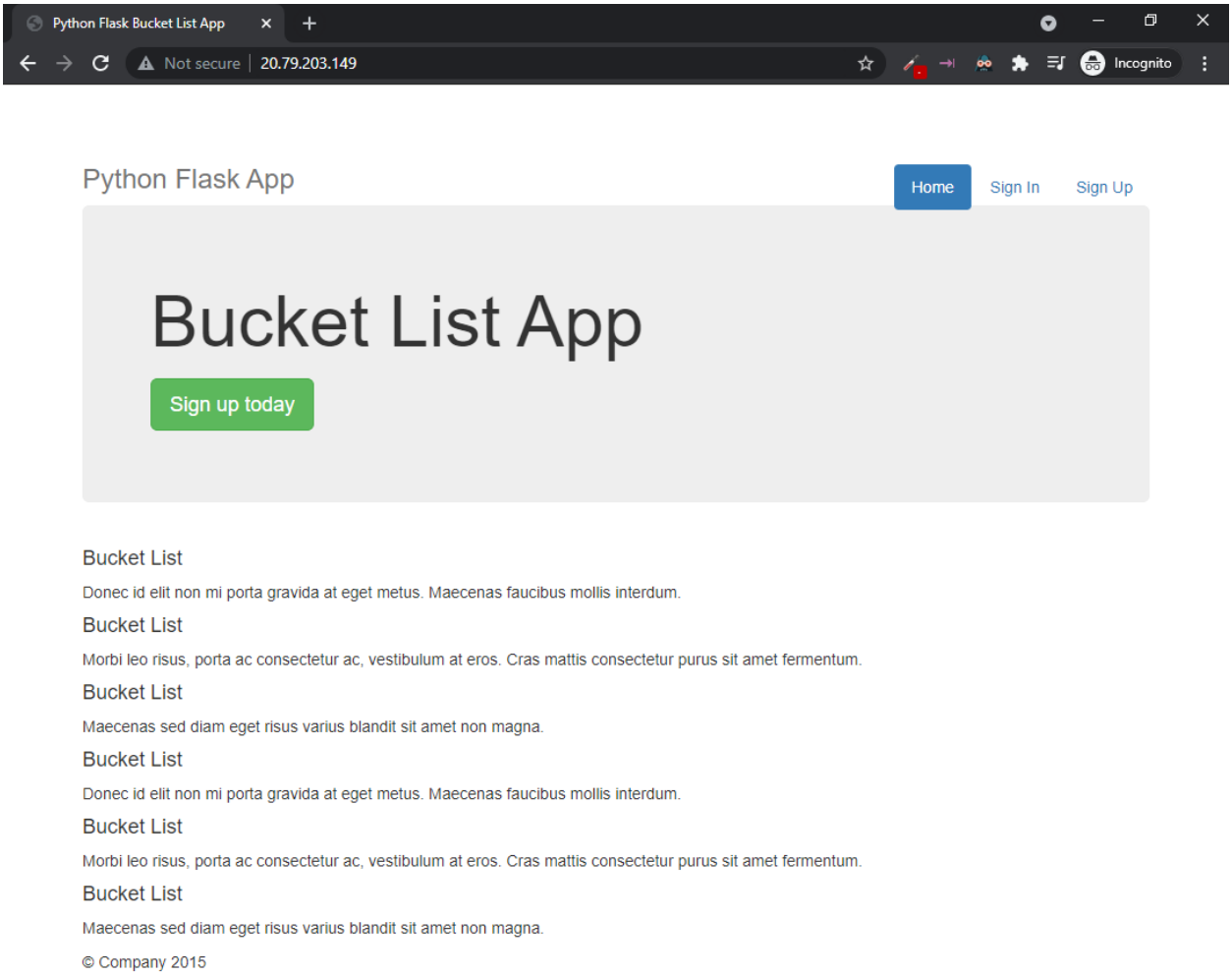
Once again, you simply run the command and provide the requested information:

```
ansible-playbook playbooks/webapp-setup.yaml
```

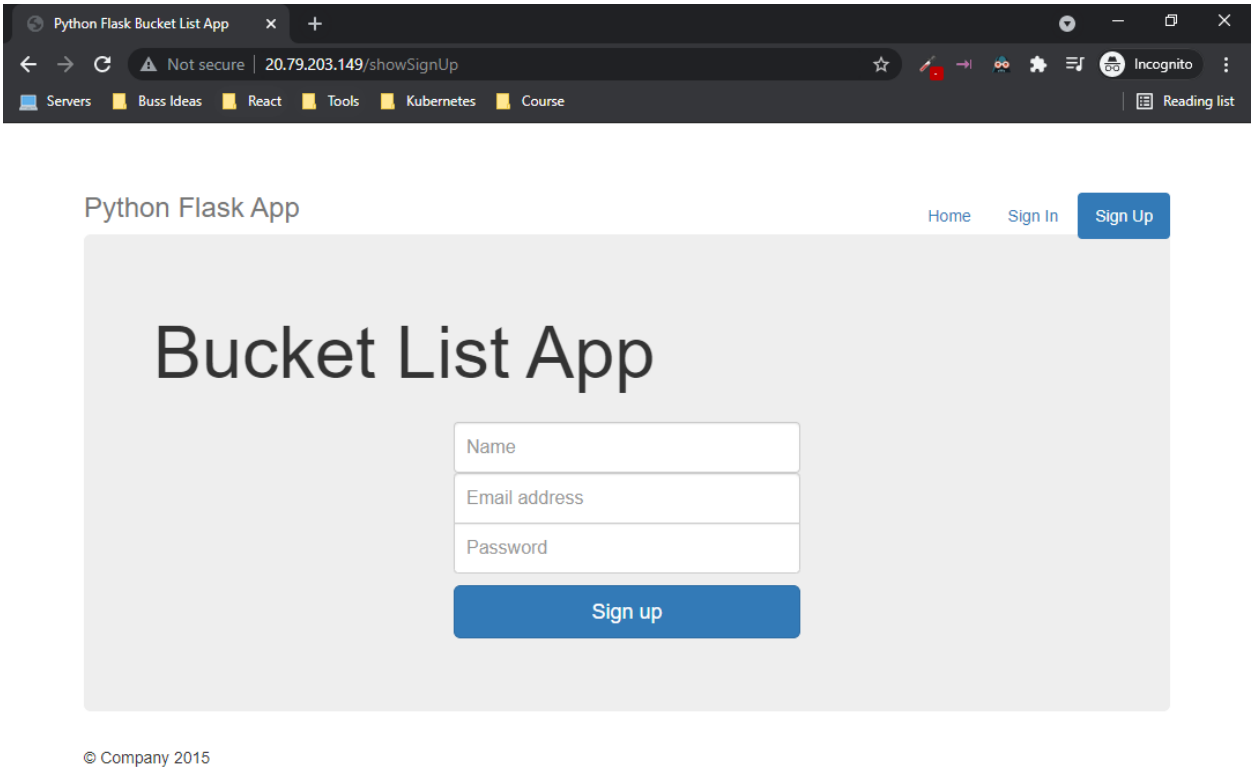
Note: Here you will be asked for the MySQL hostname and password again.

IV. Verifying the results

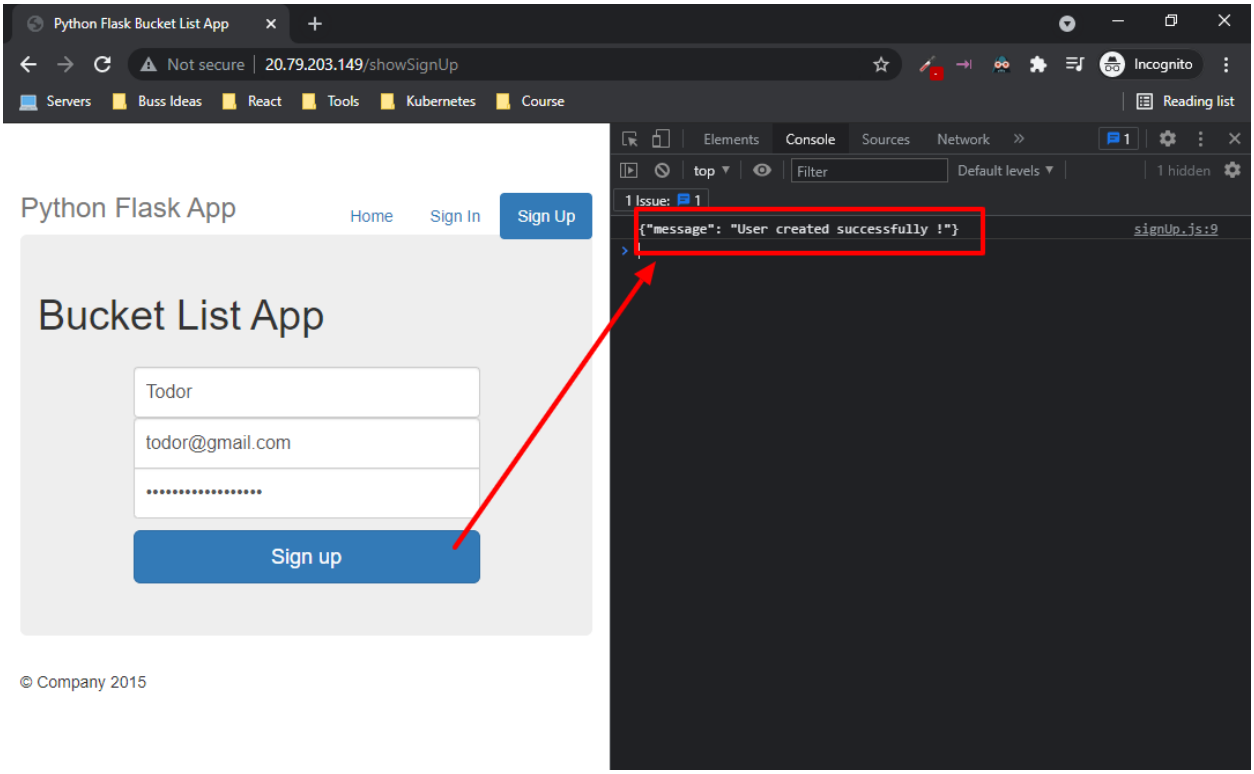
Head on to the public IP Address of the VM that was created and you should see the following:



Here you can use the "Sign Up" button to create an account:



Important: The developer of this web app (not me - I simply edited it in order to work with my project) hasn't added any way to notify the user that his account has been created successfully so this has to be checked with the internet browser's console:



Lastly, we can verify that data was added to the MySQL database by connecting to the MySQL server:

```
mysql -h __MYSQL_SERVER_HOST__ -u todor -p
```

Once the connection is created, you can use the following mysql cli commands to check the data:

```
use webapp_mariadb_db;
select * from tbl_user;
```

Which should result in something like this:

```
MySQL [webapp_mariadb_db]> select * from tbl_user;
+---------+--------------+-----------------+--------------------+
| user_id | user_name    | user_username   | user_password      |
+---------+--------------+-----------------+--------------------+
|       1 | Todor Ivanov | asdas           | asdasd             |
|       2 | Todor        | todor@gmail.com | asdasdasdasdasdasd |
+---------+--------------+-----------------+--------------------+
2 rows in set (0.002 sec)
```

V. Setting up the monitoring system

First, you need to create a NetData account from here:

https://app.netdata.cloud/

Once this is done, we need to edit the "/netdata/vars/main.yml" file and replace the "claim_token" and "claim_rooms" variables with the output you received when signing up on the NetData website

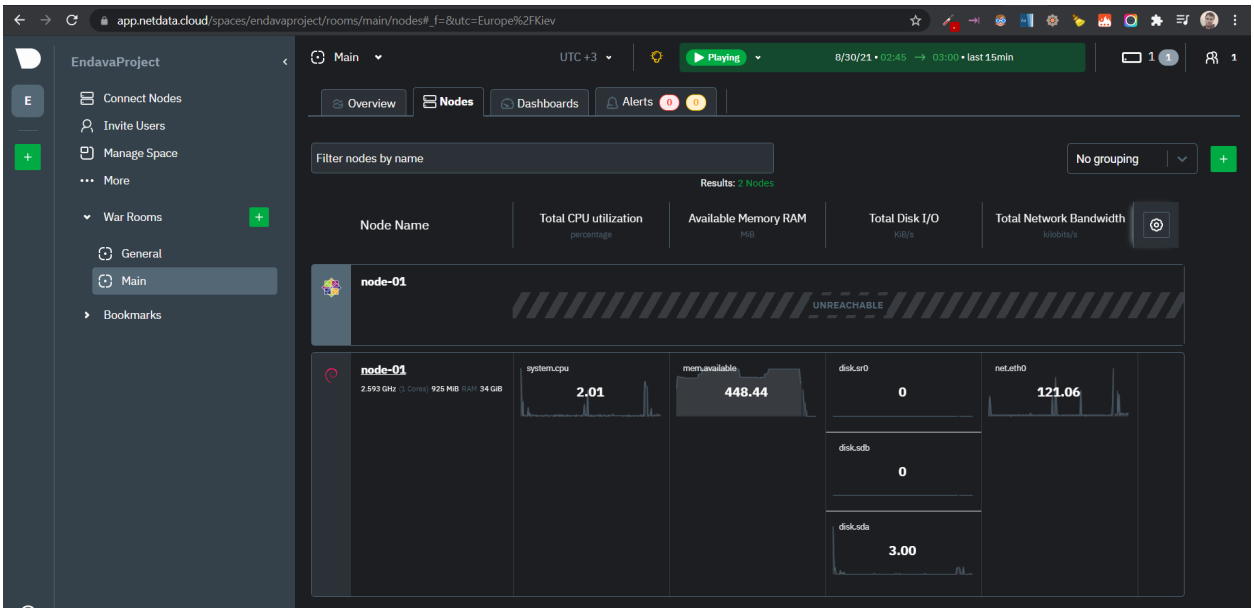
Next, we need to edit the "/netdata/host" file and replace "___VM_IP___" with the VM's public IP address:

```
sed -i.bak 's/___VM_IP___/___ENTER_YOUR_IP_HERE___/g' netdata/hosts
```

Lastly, we have to complete one last playbook:

```
ansible-playbook -i hosts netdata/tasks/main.yml
```

Afterwards, we'll be able to monitor the VM from within the Netdata's dashboard:



VI. Auto-restart on failed services

This step is already completed - the "webapp-setup.yaml" playbook creates a separate systemctl service for the Python App that has the "restart=on-failure" option enabled.

VII. Load Balancer

Didn't have the time to research this topic and provide a solution

VIII. Working demo

You can check the Python Web App that was deployed on my Azure account from the following address:

http://endava-project.todorivanov.net/

IX. Additional information:

Ansible Master server information:

- Provider: Linode
- OS: Debian
- OS version: 10
- CPU: 2 cores
- RAM: 4 GB
- Storage: 80 GB

Deployed VM information:

- Provider: Azure
- OS: Debian
- OS: Version 10
- CPU: 1 core
- RAM: 1 GB
- Temp Storage: 4 GB

Python App Source:

https://github.com/jay3dec/PythonFlaskMySQLApp---Part-1

https://code.tutsplus.com/tutorials/creating-a-web-app-from-scratch-using-python-flask-and-mysql--cms-22972