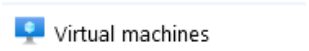
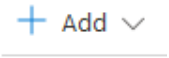
Laborator 14

Scopul acestui laborator este crearea unui mini-cluster Docker în infrastructura de Cloud Azure și a avea un server web load-balanced.

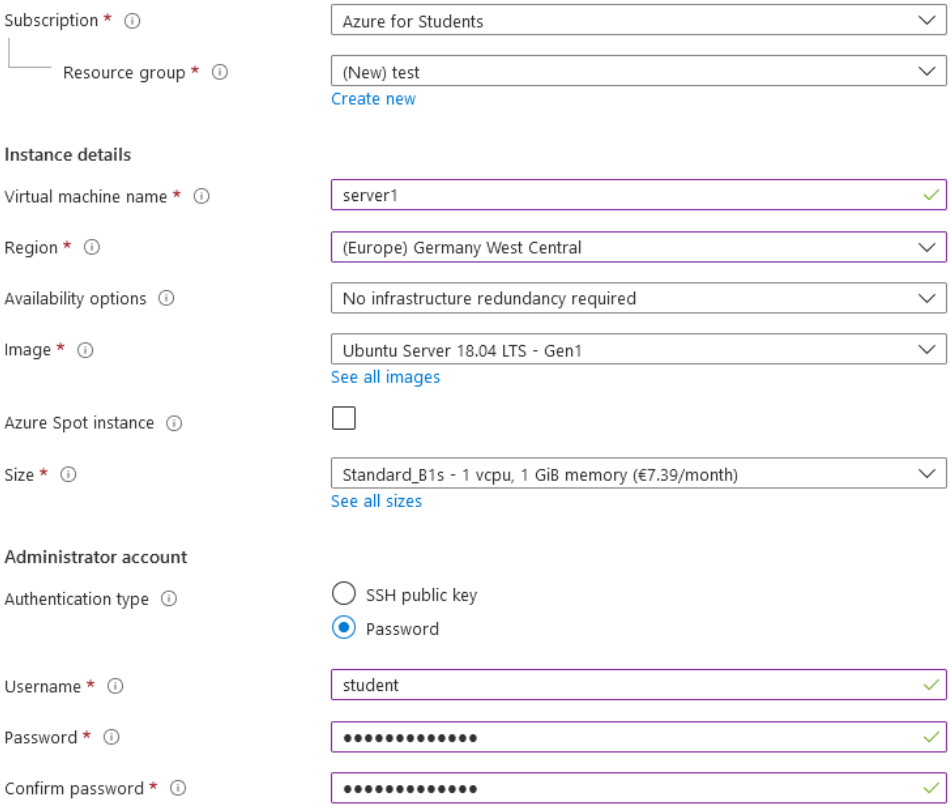
Atenție la copy-paste unele simboluri gen "-" se copiază greșit.

Exerciții

1. Reminder lab trecut: Faceți un cont de student pe Azure.
   - Se va folosi contul de e-mail mta.ro. (Pe alt cont **NU** se poate activa versiunea pentru studenți).
   - Parola de la e-mail e diferită de cea pentru wiki.
   - Acesta nu necesită card. Astfel, nu sunteți expus unui risc de cost.
   - Aveți la dispoziție 100$.
2. Se vor porni 3 mașini virtuale Linux.
   - Se va extinde meniul. 
   - Se va selecta Virtual Machines. 

   - Se va adăuga Mașină. 

   - Se vor crea 3 mașini cu următoarele caracteristici:



   - **ATENȚIE de la *Select inbound ports* dorim și SSH și HTTP de această dată.**
   - Se va merge la Review+Create și se va apăsa Create.
   - Nu uitați, vrem 3 astfel de mașini. Celelalte se vor numi server2 și server3.
3. Prin putty conectați-vă la cele 3 mașini (un terminal putty pentru fiecare).
   - IP-ul îl puteți găsi în pagina de informații a mașinii.
   - Veți folosi student@IP_public dar notați și IP-ul privat al celor 3 mașini. **E important ca toate să se afle în aceeași locație, și în același grup.**
4. Testați conexiunea între cele 3 mașini.
   - Pe una se va porni `nc -l -p 5000`
   - Pe celelalte două se va porni `nc IP_PRIVAT_PRIMA 5000`
   - Trebuie să se poată apoi scrie mesaje ce ajung de la o mașină la alta (ca un chat).
   - Testul se va face separat pentru fiecare pereche de mașini.
   - Dacă aveți nevoie de root puteți scrie `sudo su`
5. Instalați pe toate 3 mașinile Docker.
   - Sunt necesare mai multe comenzi complexe, acestea le găsiți în fișierul **installDocker.sh**. Pot fi probleme la rulare, executați comandă cu comandă.
   - **Verificare**: `docker`

Orice greșeală faceți legată de container puteți să vizualizați containerele și să le ștergeți folosind:

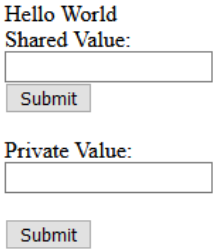
   - `docker container ls -a`
   - `docker container rm -f CONTAINER_NAME`

Deocamdată vom lucra doar pe Server1:

6. Vom avea nevoie de o rețea virtuală specială pentru aceste containere:
   - Având o astfel de rețea putem adresa containerele folosind numele lor în loc de IP-uri aceasta având suport DNS built-in.
   - `docker network create MY_NETWORK_NAME`
7. Aplicația care va rula pe serverul web necesită un in memory data store de tip **redis**:
   - Acesta este un server des folosit și astfel are o imagine pe docker hub.
   - `docker container run --network MY_NETWORK_NAME --name myredis -d redis`
     - Containerul redis va trebui să fie în aceeași rețea virtuală cu containerul apache.
     - Este important ca acest container să aibă numele **myredis**. Acest nume este folosit în fișierul index.php. Îl puteți modifica doar dacă îl modificați peste tot.
     - În final -d setează containerul să ruleze ca un daemon, în background iar redis este imaginea luată de pe dockerhub.

- **Verificare**: `docker container exec -it myredis /bin/bash` O dată intrați în container puteți da `redis-cli` și apoi comenzi redis gen `keys *` pentru listarea tuturor cheilor, `set a b` pentru scrierea lor apoi, `get a` pentru citire. Din `redis-cli` și din container se poate ieși folosind `exit` .

8. Construcția imaginii pentru serverul web:
   - Acest container este unul particularizat pentru acest laborator, astfel imaginea pentru el va trebui construită din Dockerfile și cod-ul ce va rula pe acesta, în index.php .
   - Analizați fișierul ***Dockerfile*** .
     - Acesta pornește de la imaginea php pentru apache de pe dockerhub.
     - Se instalează suport php pentru a putea comunica cu un server redis.
     - Se copiază fișierul index.php în /var/www/html/ . Acesta va fi rulat când accesăm serverul.
   - Va trebui să copiați fișierele din schelet pe server 1. Puteți folosi WinScp.
   - Construiți imaginea: `docker build -t IMAGE_TAG_NAME .` **NU uitați de punct, aceasta este calea către Dockerfile.**
   - **Verificare**: `docker image ls` .
9. Rulare container server web:
   - Puteți porni de la comanda: `docker container run IMAGE_TAG_NAME`
   - Dacă pur și simplu ați dat comanda de mai sus când veți lista containerele veți descoperii că acesta deja s-a oprit. Vedeți la începutul laboratorului cum puteți șterge containerul creat.
   - Bazat pe documentație comenzii de mai sus trebuie să adăugați următoarele:
     - Conexiune la rețeaua creată la punctul 6. (Necesită și parametru)
     - Rularea containerului în background.
     - Asignarea unui nume containerului. (Necesită și parametru)
     - Publicarea portului http (`-p 80:80`)
   - **Verificare**: Din browser intrați pe adresa publică a server1.



10. Restart container server web:
    - În browser introduceți valori pentru ambele câmpuri.
    - Dați stop apoi start la containerul de server web.
    - Verificați din browser. Ce valori au rămas? De ce? (De citit cod index.php)
    - Opriți, ștergeți și reporniți containerul de server web.
    - Verificați din browser. Ce valori au rămas? De ce? (De citit cod index.php)
11. Modificare container server web:
    - Creați un fișier numit MY_NAME care să conțină numele vostru.

- Pentru a înțelege cum funcționează tot **citiți** codul din index.php.
- Modificați Dockerfile pentru a copia acest fișier în container în aceeași locație ca fișierul index.php.
- Reconstruiți imaginea.
- De ce acum docker build a mers mult mai repede?
- **Verificare**: Din browser verificați că pe ultima linie apare numele vostru.

Fiind testată funcționalitatea containerelor dorim scalabilitate. De acum vom folosi toate 3 serverele:

12. Opriți și ștergeți containerele de la punctele precedente.
13. Porniți și setați docker swarm:
    - Pe server1 se va da comanda: `docker swarm init`
    - Copiați comanda care vă apare după rularea docker swarm init pe celelalte doua servere.
    - **Verificare**: Pe server1 rulați `docker node ls` . Ar trebui să vedeți toate 3 mașinile.
14. Creați o rețea de tip overlay care va rula peste swarm, aceasta va oferi funcționalitatea de DNS dar și load balancing:
    - `docker network create -d overlay my-swarm-network`
15. Creați un registry și puneți imaginea creată din Dockerfile acolo:
    - Celelalte servere nu au acces la imaginile primului. Astfel trebuie să creăm un spațiu unde să punem imaginea care să fie accesibil de peste tot, acesta este un registry (e asemănător cu ce oferă în spate dockerhub).
      ```
      docker service create --network my-swarm-network --name registry --publish published=5000,target=5000 registry:2
      ```
    - Adăugăm imaginea la registry:
      `docker build -t IMAGE_NAME .` E important să **nu uitați de punct**.
      ```
      docker tag IMAGE_NAME 127.0.0.1:5000/IMAGE_NAME
      docker push 127.0.0.1:5000/IMAGE_NAME
      ```
    - **Verificare**:
      - `docker container ls` pe toate 3 serverele. Unul din ele va avea containerul registry.
      - `wget http://127.0.0.1:5000/v2/_catalog/` urmat de `cat index.html` Registry comunică peste http deci putem să îl interogăm așa. Ar trebui să vedem imaginea adăugată de noi.
16. Porniți un container redis și 3 containere de server web, ca servicii peste swarm:
    - Porniți serviciu pentru redis:
      ```
      docker service create --name myredis --network my-swarm-network redis
      ```
    - Porniți serviciu pentru serverul web cu 3 containerele, una pe server:
      ```
      docker service create --name SERVICE_NAME --network my-swarm-network -p 80:80 --replicas 3 --replicas-max-per-node 1 127.0.0.1:5000/IMAGE_NAME
      ```
    - **Verificare**:
      - Pe server1: `docker service ls`
      - Pe toate serverele: `docker container ls`
      - Din browser intrați pe IP_SERVER_2 și IP_SERVER_3.
      - Se vor completa câmpurile și se va da refresh cu SHIFT apăsat.

Laboratorul va fi prezentat. Veți intra pe rând pe teams. Va trebui să aveți terminal putty deschis la mașina principală și să dați docker serverice ls. Deasemenea va trebui să aveți deschis site-ul Azure și site-ul din server2. **Toate setările ar trebui să le faceți dinainte.**

După prezentarea laboratorului mergeți pe Azure în tab-ul Resources și ștergeți toate resursele create.