



# Laborator 09

Scopul acestui laborator este crearea unui mini-cluster Kubernetes și rularea unui site web pe mai multe mașini cu suport load balance și auto-scale.

**Acest laborator va folosi mașini virtuale destul de scumpe, vă recomandăm să rezolvați laboratorul fix cu o seară înainte de oră.**

**E mai important ca niciodată să vă asigurați că imediat după laborator ștergeți toate resursele.**

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

[Documentație Kubernetes](#)

[Documentație comenzi Kubernetes](#)

**Comenzi extrem de utile în caz de greșeli:**

```
kubect1 get all  
kubect1 delete TIP NUME
```

## Exerciții

1. Treceți prin primele 5 scenarii despre Kubernetes de pe:  
<https://katacoda.com/courses/kubernetes>
2. Se vor porni 2 mașini virtuale Linux în Azure cu următoarele caracteristici:
  - Resource group: `kubernetes` (Extrem de important să fie ambele mașini în același group. Pentru a crea grupul apăsați [Create new](#).)
  - Virtual machine name: `kube1` și `kube2`
  - Region: (Europe) Germany West Central
  - Availability options: No infrastructure redundancy required
  - Azure Spot instance: nu este selectat
  - Image Ubuntu Server 18.04 LTS - Gen1
  - Size: D2s\_v3 (Una cu mai puține resurse nu va putea rula Kubernetes)
  - Authentication type: Password (la alegerea voastră dar **notați-le**)
  - Restul se lasă nemodificat.
  - Se apasă Next:Disks
  - Se apasă Next: Networking
  - NIC network security group: None
  - Se va merge la Review+Create și se va apăsa Create.
3. Prin putty conectați-vă la cele 2 mașini (un terminal putty pentru fiecare).
  - IP-ul îl puteți găsi în pagina de informații a mașinii.



- Veți folosi `username@IP_public` dar notați și IP-ul privat al celor 2 mașini. **E important ca toate să se afle în aceeași locație, și în același grup.**
- 4. Testați conexiunea între cele 2 mașini.
  - Pe una se va porni `nc -l -p 5000`
  - Pe cealaltă 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).
  - Dacă aveți nevoie de root puteți scrie `sudo su`
- 5. Instalați pe toate mașinile [Microk8s](#) (o versiune de Kubernetes de la Canonical).
  - Se vor face [toate cele 7 puncte](#) pe ambele mașini.
  - **Verificare:** `microk8s status --wait-ready`
- 6. [Uniți cele două noduri](#) într-un cluster:
  - Pe **kube1** se va da comanda: `microk8s add-node`
  - Copiați comanda care vă apare după rularea pe cealaltă mașină. Se poate să dureze.
  - **Verificare:** Pe **kube1** rulați `kubect1 get node` . Ar trebui să vedeți toate mașinile.
- 7. Porniți add-on-uri (pot fi date doar pe **kube1**):
  - DNS: `microk8s enable dns`
  - Load Balancer: `microk8s enable metallb:IP_PRIVAT_KUBE1-IP_PRIVAT_KUBE2`
  - Metrice/Log-uri: `microk8s enable metrics-server prometheus`
  - **Verificare:** `kubect1 port-forward -n monitoring service/prometheus-k8s --address IP_PRIVAT_KUBE1 9090:9090` apoi intrat din browser pe `IP_PUBLIC_KUBE1:9090` apoi Ctrl+C
  - GUI: `microk8s enable dashboard`
  - Obținere token autentificare dashboard (apar în terminal după activare):  
`token=$(microk8s kubect1 -n kube-system get secret | grep default-token | cut -d " " -f1)`  
`microk8s kubect1 -n kube-system describe secret $token`
  - **Verificare:** `kubect1 port-forward -n kube-system service/kubernetes-dashboard --address IP_PRIVAT_KUBE1 10443:443` apoi intrat din browser pe `IP_PUBLIC_KUBE1:10443` apoi Ctrl+C
  - Storage: `microk8s enable storage`
- 8. Instalați un registry privat. Un fel de Docker Hub privat pe portul 32000. Îl vom folosi pentru a putea crea imagini Docker și a le putea distribui tuturor nodurilor din cluster.
  - Instalare: `microk8s enable registry`
  - **Verificare:** `curl localhost:32000/v2/_catalog` de pe ambele mașini
  - Instalare docker:  
`sudo apt-get update`  
`sudo apt-get install docker.io`  
`sudo usermod -aG docker ${USER}`  
`su - ${USER}`
  - Se poate să fie necesar să refaceți alias-ul pentru kubect1
  - Pentru a putea folosi acest registry pe http din k8s e necesar să creați fișierul `/etc/docker/daemon.json` CU  

```
{  
  "insecure-registries" : ["localhost:32000"]  
}
```




- o }
    - o Apoi să restartați docker: `sudo systemctl restart docker`
    - o Descărcați o imagine: `docker image pull nginx`
    - o Dați un nou tag imaginii: `docker image tag nginx localhost:32000/mynginx`
    - o Puneți imaginea în registry: `docker image push localhost:32000/mynginx`
    - o **Verificare:** `curl localhost:32000/v2/_catalog` de pe ambele mașini
    - o Creați un deployment cu 3 pod-uri: `kubectl create deployment mynginx -- replicas=3 --image localhost:32000/mynginx`
    - o **Verificare:** `kubectl get all`
- 9. Porniți un deployment cu un pod **redis** (cel de pe dockerhub) și un deployment cu **3 pod-uri de server web** construite din Dockerfile:
  - o Construiți imaginea de docker din dockerfile (docker build)
  - o Puneți imaginea în registry (docker push). Va fi necesar să aibă o anumită formă la tag (vedeți exerciții anterioare, lab anterior).
  - o Porniți un deployment de redis. Începeți cu `kubectl create deployment` . Va trebui să îi dați numele *myredis*.
  - o Dorim ca acest deployment de redis să aibă o adresă de DNS internă, aceasta se atașează serviciului, astfel va trebui să expunem acest deployment folosind `kubectl expose` . Acestei comenzi trebuie să îi specificați portul 6379 și același target-port.
  - o Creați un deployment pentru containerele de servere web create. Vom dori ca acest deployment să aibă 3 replicas.
  - o Dorim ca serverele web să fie expuse către internet. Va trebui să folosim `kubectl expose` pentru deployment-ul creat cu serverele web cu external-ip IP\_PRIVAT\_KUBE1 cu port 80 și target-port 80.
  - o **Verificare:**
    1. Pe server1: `kubectl get all`
    2. Din browser intrați pe IP\_PUBLIC\_KUBE1.
    3. Se vor completa câmpurile și se va da refresh cu SHIFT apăsat.
    4. Ar trebui să vedeți:
 

Hello World

Shared Value:

Submit

Private Value:

Submit
- 10. Modificați containerul anterior (Dockerfile) astfel încât să nu mai primiți eroare de fișier lipsă.
  - o Adăugați noua imagine în registru, sub un alt nume.
  - o Rulați un nou set de 2 containere folosind portul 88. De data aceasta din dashboard. Se apasă pe  . Se apasă Create from form. Tipul de **Service** este **External**. Apăsăți **Show Advanced Options** și selectați **Namespace Default**.



11. Vrem să facem un al 3-lea deployment identic cu cel precedent folosind un fișier .yaml.
- Pentru a descoperi cum arată un fișier .yaml puteți da comenzi kubect1 de forma:  
`kubect1 create deployment nginx --image=nginx --dry-run --output='yaml'`
  - Puteți să definiți mai multe obiecte (deployment, servicii, șamd) într-un singur fișier .yaml separând acestea printr-un rând cu 3 liniițe (---).
  - O dată creat acest fișier poate fi personalizat (schimbat nume, port-uri, șamd).
  - Dorim acestui deployment să adăugăm [scalare automată](#). Vom face acest lucru din linia de comandă urmărind ghidul. Când prezentați va trebui să demonstrați că acest deployment scalează automat. Fișierului .yaml trebuie să îi adăugați partea de **resources** din exemplul de pe link.
  - O dată creat un fișier .yaml poate fi aplicat complet folosind:  
`kubect1 apply -f myyaml.yaml`
  - Infrastructura creată pe baza unui fișier .yaml poate fi oprită și ștearsă folosind:  
`kubect1 delete -f myyaml.yaml`
  - Este posibil ca în momentul afișării `kubect1 get all` auto scaler-ului să îi apară TARGET <unknown>, dacă se întâmplă asta trebuie disable și enable la metrics-server și prometheus.
  - În momentul în care prezentați va trebui să demonstrați că merge serverul web, de preferință și auto scaler-ul pentru acesta.
  - În final regula de autoscale va fi și ea adăugată în fișierul .yaml.

**Exercițiile de la 1 la 11 sunt obligatorii.** Conceptele explorate sunt esențiale pentru obținerea notei **minime** de promovare.

**Vă recomandăm, pentru a crește șansele de a obține o notă cât mai mare să explorați și următoarele exerciții:**

12. Scrieți fișierul Dockerfile și construiți containerul care să conțină proiectul vostru de la tehnologii web.
- Containerul ar trebui să aibă minim un server apache, dar poate fi necesar și php/mysql, în funcție de stadiul proiectului.
  - Portul 80 va fi deschis pe mașinile fizice pe portul 8080, pentru a permite funcționarea simultană a acestor servere cu cele de la exercițiile anterioare.
13. Porniți acest container peste clusterul Kubernetes folosind 4 replici ale sale și auto scaling.

Laboratorul va fi prezentat. Veți intra pe rând pe teams. Va trebui să aveți terminal putty deschis la mașina principală și să dați `kubect1 get all`. Deasemenea va trebui să aveți deschis site-ul Azure și site-urile din kube1. **Toate setările ar trebui să le faceți dinainte.**

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