

MODUL PRAKTIKUM KOMPUTASI AWAN



Disusun oleh:

I Made Murwantara, S.Si., M.Kom., Ph.D.

Richard David Tedja, S.Kom.

**PROGRAM STUDI INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS PELITA HARAPAN
TANGERANG**

2021

DAFTAR ISI

PRAKTIKUM 1 VAGRANT – SINGLE VIRTUAL MACHINE.....	1
PRAKTIKUM 2 VAGRANT – MULTIPLE VIRTUAL MACHINES	5
PRAKTIKUM 3 SHELL PROVISIONING.....	7
PRAKTIKUM 4 ANSIBLE PROVISIONING.....	8
PRAKTIKUM 5 LOAD BALANCING SYSTEM.....	12
PRAKTIKUM 6 WORKLOAD TESTING.....	15
PRAKTIKUM 7 COMMERCIAL CLOUD.....	18
PRAKTIKUM 8 TERRAFORM PROVISIONING	25
PRAKTIKUM 9 APPLICATIONS ON THE CLOUD	28
PRAKTIKUM 10 DOCKER - SINGLE CONTAINER.....	38
PRAKTIKUM 11 DOCKER - MULTI CONTAINER	41
PRAKTIKUM 12 KUBERNETES CLUSTER.....	46
PRAKTIKUM 13 SCALING KUBERNETES CLUSTER.....	50

PRAKTIKUM 1

VAGRANT – SINGLE VIRTUAL MACHINE

Praktikum ini akan membahas mengenai instalasi Virtualbox dan Vagrant, serta percobaan untuk membuat satu Virtual Machine (VM). Pada bagian akhir dari praktikum ini, akan dibahas pula parameter yang dapat digunakan untuk memodifikasi *resource* yang digunakan oleh VM.

1.1 Instalasi Virtualbox dan Vagrant

- a) Praktikum 1 hingga 6 mempergunakan Virtualbox dan Vagrant, yang dapat diunduh pada tautan berikut:

Jika menggunakan Windows:

Virtualbox: <https://www.virtualbox.org/wiki/Downloads>

Vagrant: <https://www.vagrantup.com/downloads>

Jika menggunakan Linux:

```
sudo apt-get update
sudo apt-get install virtualbox -y
sudo apt-get install vagrant -y
```

- b) Setelah proses instalasi selesai, *restart* perangkat.
- c) Lakukan verifikasi instalasi Virtualbox dan Vagrant dengan mengetik perintah berikut pada terminal:

```
vboxmanage --version
vagrant --version
```

- d) Jika terminal menampilkan nomor versi dari Virtualbox dan Vagrant, maka proses instalasi berhasil.

1.2 Contoh Vagrantfile 1: Single VM

- a) Buat sebuah direktori (folder) baru dengan menggunakan perintah berikut (nama direktori dapat disesuaikan):

```
mkdir Contoh1  
cd Contoh1
```

- b) Jalankan perintah berikut untuk menginisialisasi direktori tersebut menjadi sebuah lingkungan eksekusi Vagrant:

```
vagrant init
```

- c) Vagrant akan secara otomatis membuat sebuah Vagrantfile dalam direktori tersebut. Lakukan modifikasi terhadap Vagrantfile tersebut menggunakan editor teks seperti Notepad++ atau Sublime.
- d) Untuk contoh ini, modifikasi isi dari Vagrantfile hingga menjadi seperti berikut:

```
# Contoh 1  
#  
# Membuat satu VM  
  
Vagrant.configure("2") do |config|  
  config.vm.box = "ubuntu/focal64"  
  config.vm.hostname = "Contoh1"  
  config.vm.network "private_network", ip: "192.168.0.42"  
end
```

- e) Jalankan perintah berikut untuk membuat dan menyalakan Virtual Machine (VM) menggunakan konfigurasi yang terdapat pada Vagrantfile tersebut:

```
vagrant up
```

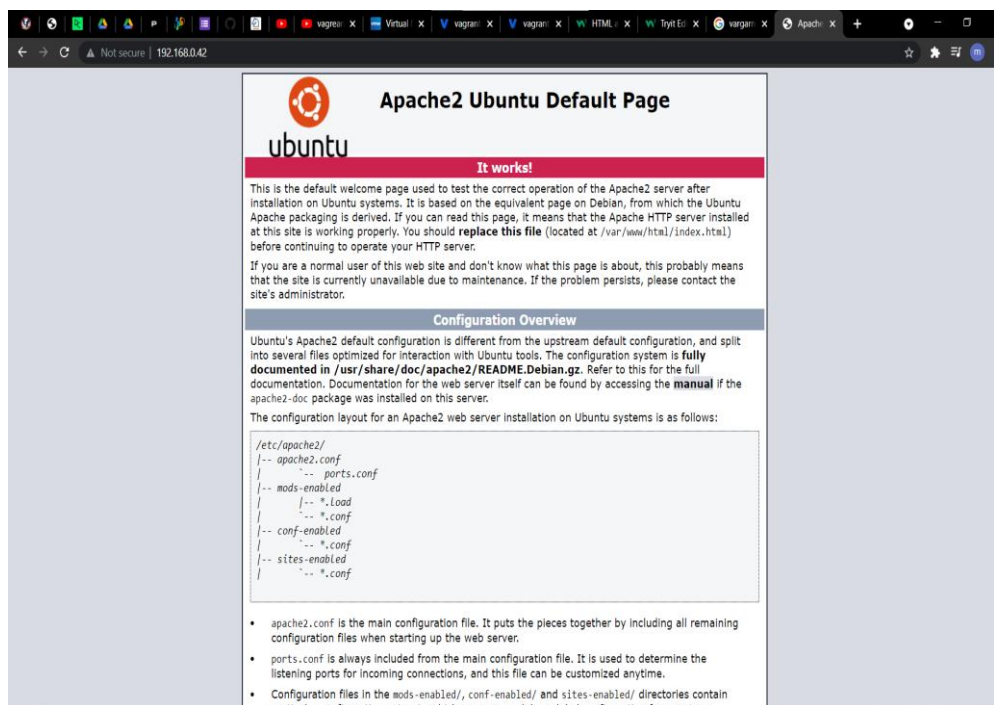
- f) Jika proses *boot* sudah selesai, maka login ke dalam VM dengan menggunakan perintah berikut:

```
vagrant ssh
```

- g) Lakukan instalasi webserver Apache2 dengan menggunakan perintah berikut:

```
sudo apt-get install apache2 -y
```

- h) Jalankan browser, lalu ketik alamat IP dari VM yang telah dibuat
i) Jika muncul tampilan seperti ini, maka konfigurasi VM berhasil.



1.3 Contoh Vagrantfile 2: Single VM dengan Modifikasi

- a) Ulangi langkah **1.1 a) hingga c)** untuk membuat sebuah lingkungan eksekusi Vagrant pada direktori yang berbeda.
- b) Untuk contoh ini, modifikasi isi dari Vagrantfile hingga menjadi seperti berikut:

```
# Contoh 2
#
# Membuat satu VM dengan parameter modifikasi

Vagrant.configure("2") do |config|
  config.vm.box = "ubuntu/focal64"
  config.vm.hostname = "Example2"
  config.vm.network "private_network", ip: "192.168.0.42"

  config.vm.provider :virtualbox do |vb|
    vb.customize [
      "modifyvm", :id,
      "--cpuexecutioncap", "50",
      "--memory", "256",
    ]
  end
end
```

- c) Perbedaan contoh ini dengan contoh Vagrantfile 1 terletak pada parameter `modifyvm`, dimana kita memodifikasi `cpuexecutioncap` (persentase host CPU yang dapat digunakan oleh VM), dan `memory` (ukuran RAM yang dapat dialokasikan untuk VM).
- d) Ulangi langkah **1.1 e) hingga i)**
- e) Bandingkan performa VM pada contoh ini dengan contoh sebelumnya.

PRAKTIKUM 2

VAGRANT – MULTIPLE VIRTUAL MACHINES

Praktikum ini akan melanjutkan bahasan dari Praktikum 1. Untuk praktikum ini, akan dilakukan percobaan membuat tiga Virtual Machine (VM).

2.1 Contoh Vagrantfile 3: Multiple Virtual Machines (VM)

- a) Ulangi langkah **1.1 a) hingga c)** untuk membuat sebuah lingkungan eksekusi Vagrant pada direktori yang berbeda.
- b) Untuk contoh ini, modifikasi isi dari Vagrantfile hingga menjadi seperti berikut:

```
# Contoh 3
# Membuat tiga VM

Vagrant.configure("2") do |config|

  config.vm.define "vm1" do |vm1|
    vm1.vm.box = "ubuntu/focal64"
    vm1.vm.hostname = 'VM1'
    vm1.vm.network :private_network, ip: "192.168.56.101"

    vm1.vm.provider :virtualbox do |vb1|
      vb1.customize ["modifyvm", :id, "--cpuexecutioncap", "50"]
      vb1.customize ["modifyvm", :id, "--memory", "256",]
    end
  end

  config.vm.define "vm2" do |vm2|
    vm2.vm.box = "ubuntu/focal64"
    vm2.vm.hostname = 'VM2'
    vm2.vm.network :private_network, ip: "192.168.56.102"

    vm2.vm.provider :virtualbox do |vb2|
      vb2.customize ["modifyvm", :id, "--cpuexecutioncap", "50"]
      vb2.customize ["modifyvm", :id, "--memory", "512",]
    end
  end

  config.vm.define "vm3" do |vm3|
    vm3.vm.box = "ubuntu/focal64"
    vm3.vm.hostname = 'VM3'
    vm3.vm.network :private_network, ip: "192.168.56.103"

    vm3.vm.provider :virtualbox do |vb3|
      vb3.customize ["modifyvm", :id, "--cpuexecutioncap", "50"]
      vb3.customize ["modifyvm", :id, "--memory", "512",]
    end
  end
end
```

- c) Jalankan perintah berikut untuk membuat dan menyalakan Virtual Machine (VM) menggunakan konfigurasi yang terdapat pada Vagrantfile tersebut:

```
vagrant up
```

- d) Jika terdapat lebih dari satu VM pada sebuah lingkungan eksekusi Vagrant, maka pada saat login perlu disebutkan *hostname* VM yang dituju. Contoh berikut dapat digunakan untuk login ke dalam VM1.

```
vagrant ssh vm1
```

- e) Lakukan instalasi webserver Apache2 seperti pada langkah **1.1 g) hingga i)**

PRAKTIKUM 3

SHELL PROVISIONING

Pada praktikum ini, akan dilakukan percobaan membuat tiga Virtual Machine (VM), dan kemudian melakukan *provisioning* dengan *provisioner* Shell, yaitu *shell script* yang dijalankan pada VM. *Provisioner* akan dijalankan pada saat proses **vagrant up** dan dapat digunakan untuk melakukan instalasi software, modifikasi konfigurasi, dan lain-lain, pada VM yang dibuat. Untuk praktikum ini, *provisioner* Shell akan melakukan instalasi *load balancer* HAProxy pada VM1, *webserver* Apache2 pada VM2, dan MySQL Server pada VM3.

3.1 Contoh Vagrantfile 4: Multiple VM dengan Shell Provisioning

- a) Ulangi langkah **1.1 a) hingga c)** untuk membuat sebuah lingkungan eksekusi Vagrant pada direktori yang berbeda.
- b) Untuk contoh ini, modifikasi isi dari Vagrantfile hingga menjadi seperti berikut:

```
# Contoh 4
# Shell Provisioning

Vagrant.configure("2") do |config|
  if Vagrant.has_plugin?("vagrant-vbguest")
    config.vbguest.auto_update = false
  end

  config.vm.define "vm1" do |vm1|
    vm1.vm.box = "ubuntu/focal64"
    vm1.vm.hostname = 'WEB'
    vm1.vm.network :private_network, ip: "192.168.56.101",
      auto_correct: true
    vm1.vm.network :forwarded_port, guest: 22, host: 2201,
      id: "ssh", auto_correct: true
    vm1.vm.provision "shell", inline: <<-SHELL
      sudo apt-get update
      sudo apt-get install apache2 -y
      SHELL
    end

    config.vm.define "vm2" do |vm2|
      vm2.vm.box = "ubuntu/focal64"
      vm2.vm.hostname = 'LB'
      vm2.vm.network :private_network, ip: "192.168.56.102",
        auto_correct: true
      vm2.vm.network :forwarded_port, guest: 22, host: 2201,
        id: "ssh", auto_correct: true
      vm2.vm.provision "shell", inline: <<-SHELL
```

```

    apt-get update
    apt-get install haproxy -y
    SHELL
end

config.vm.define "vm3" do |vm3|
  vm3.vm.box = "ubuntu/focal64"
  vm3.vm.hostname = 'DB'
  vm3.vm.network :private_network, ip: "192.168.56.103",
  auto_correct: true
  vm3.vm.network :forwarded_port, guest: 22, host: 2201,
  id: "ssh", auto_correct: true
  vm3.vm.provision "shell", inline: <<-SHELL
  apt-get update
  apt-get install mysql-server -y
  SHELL
end
end

```

- c) Ulangi langkah **2.1 c) dan d)** untuk menjalankan VM dan login ke dalam setiap VM yang telah dibuat
- d) Jalankan perintah berikut pada VM1 untuk memastikan bahwa *provisioner* Shell telah melakukan instalasi load balancer HAProxy. Jika *output* tidak menunjukkan *error*, maka *provisioning* berhasil dan HAProxy telah terinstalasi pada VM tersebut.

```
sudo systemctl status haproxy
```

- e) Jalankan perintah berikut pada VM2 untuk memastikan bahwa *provisioner* Shell telah melakukan instalasi webserver Apache2. Jika *output* tidak menunjukkan *error*, maka *provisioning* berhasil dan Apache2 telah terinstalasi pada VM tersebut.

```
sudo systemctl status apache2
```

- f) Jalankan perintah berikut pada VM3 untuk memastikan bahwa *provisioner* Shell telah melakukan instalasi MySQL Server. Jika *output* tidak menunjukkan *error*, maka *provisioning* berhasil dan MySQL telah terinstalasi pada VM tersebut.

```
sudo systemctl status mysql
```

PRAKTIKUM 4

ANSIBLE PROVISIONING

Pada praktikum ini, akan dilakukan percobaan membuat tiga Virtual Machine (VM), dan kemudian melakukan *provisioning* dengan *provisioner* Ansible, yaitu sebuah *automation tool* dan *infrastructure as a code* yang tersedia secara *open source*. Untuk praktikum ini, *provisioner* Ansible akan melakukan instalasi *load balancer* HAProxy pada VM1, dan *webserver* Apache2 pada VM2 dan VM3 secara otomatis pada saat `vagrant up`.

4.1 Contoh Vagrantfile 5: Multiple VM dengan Ansible Provisioning

- a) Ulangi langkah **1.1 a) hingga c)** untuk membuat sebuah lingkungan eksekusi Vagrant pada direktori yang berbeda.
- b) Untuk contoh ini, modifikasi isi dari Vagrantfile hingga menjadi seperti berikut:

```
# Contoh 5
# Ansible Provisioning

Vagrant.configure("2") do |config|
  if Vagrant.has_plugin?("vagrant-vbguest")
    config.vbguest.auto_update = false
  end

  (1..2).each do |i|
    config.vm.define "vm#{i}" do |node|
      node.vm.box = "ubuntu/focal64"
      node.vm.hostname = "Web#{i}"
      node.vm.network "private_network", ip:
        "192.168.56.10#{i}"
      node.vm.network "forwarded_port", guest: 80, host:
        "809#{i}"
      node.vm.provision "ansible_local" do |ansible|
        ansible.verbosity = "v"
        ansible.playbook = "web.yml"
      end
    end
  end

  config.vm.define "vm3" do |vm3|
    vm3.vm.box = "ubuntu/focal64"
    vm3.vm.hostname = 'LB'
    vm3.vm.network :private_network, ip: "192.168.56.103",
      auto_correct: true
```

```

vm3.vm.network :forwarded_port, guest: 80, host: 8092,
  id: "lb", auto_correct: true
vm3.vm.provision "ansible_local" do |ansible|
  ansible.verbose = "v"
  ansible.playbook = "lb.yml"
end
end
end

```

- c) Jalankan editor teks seperti Notepad++ atau Sublime, kemudian buat sebuah file pada direktori tersebut dan beri nama **web.yml**. File ini merupakan Ansible Playbook dan berisi konfigurasi yang akan dijalankan oleh Ansible guna melakukan instalasi *webserver* pada VM1 dan VM2 secara otomatis. Isi dari file tersebut adalah sebagai berikut:

```

- hosts: vm1, vm2
  become: yes
  tasks:
    - name: install apache2
      apt: name=apache2 update_cache=yes state=latest

```

- d) Buat sebuah Ansible Playbook baru pada direktori tersebut dan beri nama **lb.yml**. File ini berisi konfigurasi yang akan dijalankan oleh Ansible guna melakukan instalasi *load balancer* pada VM3 secara otomatis. Isi dari file tersebut adalah sebagai berikut:

```

- hosts: vm3
  become: yes
  tasks:
    - name: install haproxy
      apt: name=haproxy update_cache=yes state=latest
    - name: Enable init script
      replace: dest='/etc/default/haproxy'
               regexp='ENABLED=0'
               replace='ENABLED=1'

```

- e) Ulangi langkah **2.1 c) dan d)** untuk menjalankan VM dan login ke dalam setiap VM yang telah dibuat

- f) Jalankan perintah berikut pada VM1 dan VM2 untuk memastikan bahwa *provisioner* Ansible telah melakukan instalasi webserver Apache2. Jika *output* tidak menunjukkan *error*, maka *provisioning* berhasil dan Apache2 telah terinstalasi pada kedua VM tersebut.

```
sudo systemctl status apache2
```

- g) Jalankan perintah berikut pada VM3 untuk memastikan bahwa *provisioner* Ansible telah melakukan instalasi load balancer HAProxy. Jika *output* tidak menunjukkan *error*, maka *provisioning* berhasil dan HAProxy telah terinstalasi pada VM tersebut.

```
sudo systemctl status haproxy
```

PRAKTIKUM 5

LOAD BALANCING SYSTEM

Pada praktikum ini, akan dilakukan percobaan untuk mengkonfigurasi sistem penyeimbang beban (*load balancer system*) dengan menggunakan lingkungan eksekusi Vagrant yang telah dibuat pada praktikum sebelumnya. Pada akhir praktikum ini, akan dilakukan pengujian sederhana terhadap sistem yang telah dikonfigurasi.

5.1 Konfigurasi Load Balancer

- a) Gunakan lingkungan eksekusi Vagrant yang telah dibuat pada Praktikum 4
- b) Modifikasi file konfigurasi HAProxy dengan menjalankan perintah berikut pada terminal:

```
sudo nano /etc/haproxy/haproxy.cfg
```

- c) Pada bagian akhir file konfigurasi tersebut, tambahkan:

```
frontend http_front
  stats enable
  stats refresh 10s
  bind *:8080
  stats uri /stats
  default_backend http_back

backend http_back
  balance roundrobin
  server web1 192.168.56.101:80 check
  server web2 192.168.56.102:80 check
```

d) Penjelasan dari setiap baris yang ditambahkan:

- **frontend**: mendefinisikan dan memberikan nama untuk bagian *Frontend* dari *load balancing system*, dalam contoh ini nama yang digunakan adalah **http_front**
- **stats enable**: Digunakan untuk dapat mengakses HAProxy *statistics monitoring page*
- **stats refresh 10s**: Menginstruksikan HAProxy untuk memuat ulang *statistics monitoring page* setiap 10 detik
- **bind *:8080**: Beban kerja yang masuk menuju *load balancing system* harus melalui port 8080
- **stats uri /stats**: *Statistics monitoring page* dapat diakses pada uri /stats
- **default_backend**: Menginstruksikan HAProxy untuk menggunakan *Backend* tertentu, dalam contoh ini **http_back**
- **backend**: mendefinisikan dan memberikan nama untuk bagian *Backend* dari *load balancing system*, dalam contoh ini nama yang digunakan adalah **http_back**
- **balance**: Algoritma yang digunakan untuk menyeimbangkan beban kerja yang masuk menuju *load balancing system* (roundrobin, leastconn, static-rr, source, uri, dan sebagainya)
- **server [nama] [alamat ip]**: Mendefinisikan daftar *server* dimana beban kerja dapat dialokasikan oleh *load balancer*.
- **check**: Menginstruksikan *load balancer* untuk melakukan *health check* setiap beberapa waktu untuk memeriksa status (*up/down*) dari setiap server.

e) Jalankan perintah berikut untuk memuat ulang (*reload*) HAProxy setiap melakukan perubahan terhadap file konfigurasi.

```
sudo systemctl reload haproxy
```

5.2 Uji Beban Sederhana dengan Apache Benchmarking Tool

- a) Unduh dan lakukan instalasi Apache Benchmarking Tool

Jika menggunakan Windows:

<https://www.apachelounge.com/download/>

Jika menggunakan Linux:

```
sudo apt-get update
sudo apt-get install apache2-utils
```

- b) Akses HAProxy statistics monitoring page melalui link berikut:

[http:// 192.168.56.103:8080/stats/](http://192.168.56.103:8080/stats/)

- c) Jalankan Apache Benchmarking Tool dengan perintah berikut:

```
ab -n 1000 -c 200 http://192.168.56.103:8080/
```

- d) Penjelasan parameter yang digunakan:

- **-n**: Jumlah total *request* yang dikirimkan kepada load balancer
- **-c**: *Concurrency* atau jumlah *request* yang dikirim secara bersamaan pada satu waktu.

- e) Lakukan observasi pada HAProxy *statistics monitoring page* selama Apache Benchmarking Tool bekerja. Silakan ulangi eksperimen dengan mengubah nilai parameter **-n** atau **-c**, atau mengubah algoritma pada konfigurasi *load balancer*.

PRAKTIKUM 6 WORKLOAD TESTING

Pada praktikum ini, akan dilakukan percobaan untuk melakukan simulasi beban kerja (*workload simulation*). Aplikasi yang digunakan adalah Apache JMeter, yang merupakan sebuah aplikasi *open-source* berbasis Java untuk melakukan simulasi beban kerja dari sebuah layanan web.

6.1 Instalasi Apache JMeter

- a) Gunakan lingkungan eksekusi Vagrant yang telah terkonfigurasi pada Praktikum 5
- b) Unduh Apache JMeter dari link berikut:

https://jmeter.apache.org/download_jmeter.cgi

- c) Ekstrak file JMeter yang telah diunduh, lalu masuk ke direktori `/bin`
- d) Jalankan Apache JMeter (GUI Mode) dengan perintah berikut:

Jika menggunakan Windows:

```
jmeter
```

Jika menggunakan Linux:

```
./jmeter
```

6.2 Membuat Test Plan dengan GUI Mode

- a) Pada Apache JMeter GUI Mode yang telah terbuka, masukkan nama Test Plan yang akan dibuat (contoh: TP1)
- b) Klik kanan pada elemen TP1 > Add > Threads > Thread Group
- c) Masukkan jumlah *thread* yang diinginkan (contoh: 100, maka JMeter akan mengirimkan 100 *thread* ke *load balancer*)
- d) Masukkan durasi *ramp-up period* (contoh: 10, maka JMeter akan membutuhkan waktu 10 detik untuk mengirim semua *thread*)
- e) Masukkan jumlah *loop* yang diinginkan (contoh: 2, maka JMeter akan mengirim *thread* sebanyak 2 x jumlah thread pada langkah a)

- f) Klik kanan pada elemen Thread Group > Add > Sampler > HTTP
- g) Masukkan `http://192.168.56.103/` pada kolom IP Address dan 8080 pada kolom port
- h) Simpan perubahan yang dilakukan terhadap Test Plan
- i) Berikut adalah tampilan akhir dari Test Plan yang telah dibuat:

The image shows two screenshots of the Apache JMeter GUI. The top screenshot displays the 'Thread Group' configuration window. It includes fields for 'Name' (Thread Group), 'Comments', and 'Action to be taken after a Sampler error' (with options: Continue, Start Next Thread Loop, Stop Thread, Stop Test, Stop Test Now). Under 'Thread Properties', it shows 'Number of Threads (users): 100', 'Ramp-up period (seconds): 10', 'Loop Count: 2' (with 'Infinite' unchecked), and checkboxes for 'Same user on each iteration' (checked), 'Delay Thread creation until needed' (unchecked), and 'Specify Thread lifetime' (unchecked). There are also fields for 'Duration (seconds):' and 'Startup delay (seconds):'. The bottom screenshot shows the 'HTTP Request' configuration window. It has fields for 'Name' (HTTP Request), 'Comments', and tabs for 'Basic' and 'Advanced'. Under 'Basic', it shows 'Web Server' settings: 'Protocol (http):', 'Server Name or IP: http://192.168.56.103:8080/', and 'Port Number: 8080'. Below this, it shows 'HTTP Request' details: 'Method: GET', 'Path:', and 'Content-Type:'. There are checkboxes for 'Redirect Automatically', 'Follow Redirects', 'Use KeepAlive', 'Use multipart/form-data', and 'browser-compatible headers'. At the bottom, there is a 'Parameters' section with a table for 'Send Parameters With the Request'.

6.3 Menjalankan Test Plan dengan CLI Mode

- a) Tidak disarankan untuk menjalankan Test Plan menggunakan GUI Mode, karena akan memperlambat kinerja perangkat. Untuk itu, gunakan CLI (Non-GUI) Mode yang dapat dijalankan melalui terminal.
- b) Jalankan Test Plan yang telah disimpan dengan perintah berikut:

```
jmeter -n -t TP1.jmx -l report.csv
```

- c) Penjelasan parameter yang digunakan:
- **-n**: Menginstruksikan JMeter untuk berjalan pada CLI Mode
 - **-t**: Nama Test Plan yang akan dijalankan
 - **-l**: Nama file CSV untuk menyimpan laporan hasil pengujian
- d) Lakukan observasi pada HAProxy *statistics monitoring page* selama Test Plan dijalankan

6.4 Eksperimen Lanjut

- a) Tabel berikut merupakan daftar spesifikasi Test Plan yang digunakan oleh Moodle untuk melakukan pengujian beban kerja pada sistem layanan yang mereka gunakan. Anda dapat mengulangi eksperimen dengan memodifikasi jumlah thread, ramp-up period, dan jumlah loop dengan merujuk pada tabel ini.

Number of Threads	Ramp-up Period	Loop Count
1	1	5
30	6	5
100	40	5
1000	100	6
5000	500	6
10000	800	7

Referensi:

https://docs.moodle.org/dev/Load_testing_Moodle_with_JMeter

PRAKTIKUM 7 COMMERCIAL CLOUD

Pada praktikum ini, akan dilakukan percobaan untuk membuat Virtual Machine pada layanan *commercial cloud*. Praktikum ini menggunakan *platform* Microsoft Azure, namun terdapat berbagai alternatif penyedia *layanan commercial cloud* lainnya yang dapat digunakan, seperti Amazon Web Services (AWS) atau Google Cloud Platform (GCP). Layanan yang tersedia pada *commercial cloud* pada umumnya berbayar, namun terdapat kebijakan khusus dimana pelajar/mahasiswa dapat menggunakan layanan secara cuma-cuma dengan batasan-batasan tertentu.

7.1 Aktivasi Layanan Azure for Students

- a) Kunjungi tautan berikut untuk mengaktifkan layanan Azure for Students dan mendapatkan kredit sebesar USD 100 yang berlaku selama satu tahun. Anda dapat memperpanjang masa berlaku tersebut setiap tahun dengan catatan alamat email yang Anda gunakan tersebut masih aktif.

<https://azure.microsoft.com/en-us/free/students/>

- b) Login dengan alamat email yang terafiliasi dengan institusi pendidikan Anda. Alamat email pribadi tidak dapat digunakan untuk layanan ini.
- c) Jika menggunakan Windows, maka Anda perlu mengunduh dan install aplikasi PuTTY melalui tautan berikut:

<https://www.putty.org/>

Anda dapat mengabaikan langkah ini apabila menggunakan Linux.

- d) Anda dapat memeriksa saldo kredit Anda dengan mengakses tautan berikut: <https://www.microsoftazuresponsorships.com/Balance>

7.2 Membuat Virtual Machine untuk Load Balancer

- a) Setelah Anda login ke dalam Azure, akses Azure Portal atau kunjungi tautan berikut: <https://portal.azure.com/#home>
- b) Pada menu Azure Services, klik pilihan **Virtual Machines**
- c) Ketika *dashboard* Virtual Machines sudah terbuka, klik **Create** lalu pilih **Virtual Machine**

- d) Pada kolom **Resource Group**, beri nama **cloudRG**. Pada Azure, sebuah *resource group* adalah kelompok layanan yang digunakan oleh seorang pengguna, dapat berupa *virtual machine*, *database*, ruang penyimpanan, dan sebagainya. Tujuan pengelompokkan tersebut adalah untuk memudahkan manajemen, otomasi, dan konfigurasi dari semua sumber daya yang digunakan. Pada praktikum ini, semua VM akan dikelompokkan dalam resource group cloudRG tersebut.
- e) Pada kolom **Virtual Machine Name**, beri nama **loadbalancer**
- f) Pada kolom **Region**, pilih **(Asia Pacific) Southeast Asia**. Selalu pilih wilayah terdekat dengan lokasi Anda untuk meminimalisir *latency*.
- g) Pada kolom **Size**, pilih **Standard_D2s_v3**, dimana VM ini akan mempergunakan 2 vCPU dan 8GB memori. Perlu diperhatikan bahwa untuk akun Azure for Students, terdapat batasan 4 vCPU untuk setiap pengguna. Apabila 2 vCPU telah dialokasikan untuk *load balancer*, maka tersisa 2 vCPU yang dapat digunakan untuk kedua *webserver* (masing-masing webserver menggunakan 1 vCPU).
- h) Pada kolom **SSH Public Key Source**, pilih **Generate new key pair**
- i) Pada kolom **Key Pair Name**, beri nama **cloudkey**
- j) Pastikan konfigurasi sudah benar seperti tampilan berikut:

Create a virtual machine ...

Create a virtual machine that runs Linux or Windows. Select an image from Azure marketplace or use your own customized image. Complete the Basics tab then Review + create to provision a virtual machine with default parameters or review each tab for full customization. [Learn more](#)

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * ⓘ Azure for Students ▼

Resource group * ⓘ (New) cloudRG ▼ [Create new](#)

Instance details

Virtual machine name * ⓘ loadbalancer ✓

Region * ⓘ (Asia Pacific) Southeast Asia ▼

Availability options ⓘ No infrastructure redundancy required ▼

Security type ⓘ Standard ▼

Image * ⓘ Ubuntu Server 20.04 LTS - Gen2 ▼ [See all images](#) | [Configure VM generation](#)

Azure Spot instance ⓘ ☐

Create a virtual machine ...

Authentication type ⓘ

☒ SSH public key
☐ Password

i Azure now automatically generates an SSH key pair for you and allows you to store it for future use. It is a fast, simple, and secure way to connect to your virtual machine.

Username * ⓘ ✓

SSH public key source ▼

Key pair name * ✓

Inbound port rules

Select which virtual machine network ports are accessible from the public internet. You can specify more limited or granular network access on the Networking tab.

Public inbound ports * ⓘ ☐ None
☒ Allow selected ports

Select inbound ports * ▼

k) Klik tombol **Review + Create**.

l) Unduh *key pair* yang dapat Anda gunakan untuk melakukan koneksi SSH ke VM. Simpan *key pair* tersebut di sebuah direktori yang aman.

m) Tunggu hingga proses validasi dan pembuatan VM selesai

7.3 Membuat Virtual Machine untuk Webserver

a) Langkah berikutnya adalah membuat VM untuk *webserver*. Kembali ke *dashboard* Virtual Machines, lalu pilih **Create > Virtual Machine**

b) Pada kolom Resource Group, pilih resource group yang telah dibuat sebelumnya yaitu cloudRG

c) Pada kolom Virtual Machine Name, beri nama web1 atau web2

d) Pada kolom Region, pilih (Asia Pacific) Southeast Asia

e) Pada kolom Size, pilih Standard_D2s_v2, dimana setiap VM untuk webserver akan mempergunakan 1 vCPU dan 8GB memori


f) Pada kolom SSH Public Key Source, pilih Use existing key stored in Azure

g) Pada kolom Key pair name, pilih cloudkey


h) Pastikan konfigurasi sudah benar seperti tampilan berikut:

Create a virtual machine ...

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription *	<div>Azure for Students</div>
Resource group *	<div>cloudRG</div> <div>Create new</div>
Instance details	
Virtual machine name *	<div>web1</div>
Region *	<div>(Asia Pacific) Southeast Asia</div>
Availability options	<div>No infrastructure redundancy required</div>
Security type	<div>Standard</div>
Image *	<div> Ubuntu Server 20.04 LTS - Gen2</div> <div>See all images Configure VM generation</div>
Azure Spot instance	<div><input type="checkbox"/></div>
Size *	<div>Standard_DS1_v2 - 1 vcpu, 3.5 GiB memory (IDR 900,817.08/month)</div> <div>See all sizes</div>

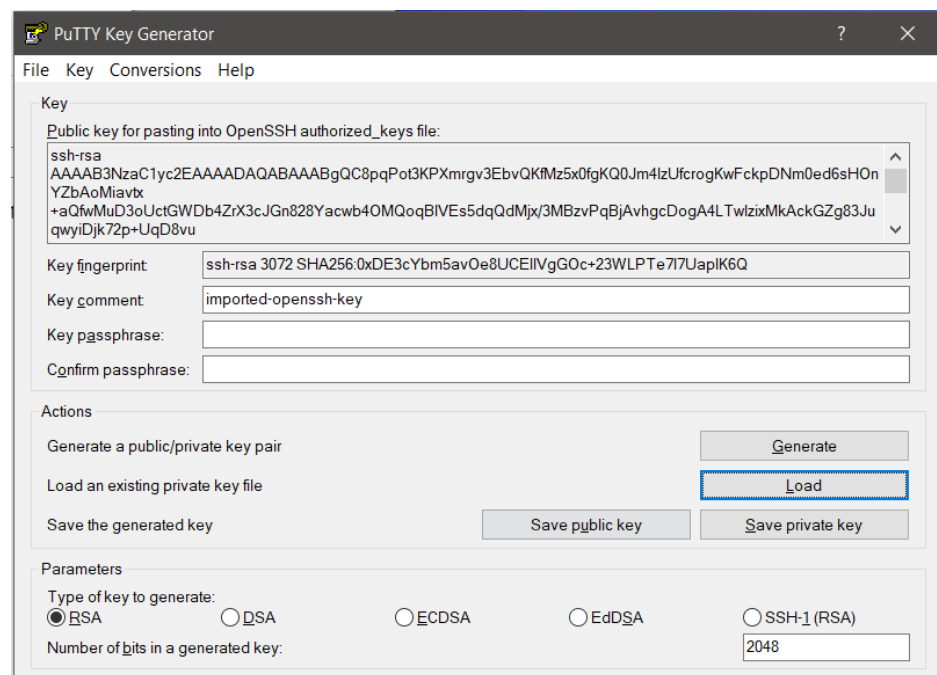
Create a virtual machine ...

Authentication type	<div><input checked="" type="radio"/> SSH public key</div> <div><input type="radio"/> Password</div>
<div> Azure now automatically generates an SSH key pair for you and allows you to store it for future use. It is a fast, simple, and secure way to connect to your virtual machine.</div>	
Username *	<div>azureuser</div>
SSH public key source	<div>Use existing key stored in Azure</div>
Stored Keys	<div>public_key</div>

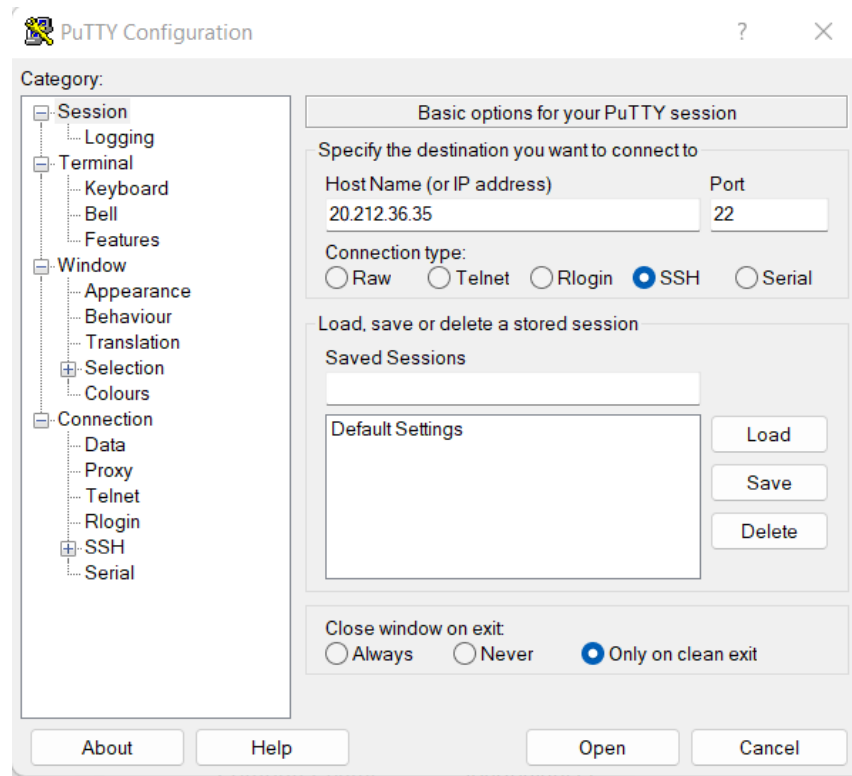
- Tunggu hingga proses validasi dan pembuatan VM selesai
- Ulangi langkah-langkah diatas sehingga akan terdapat dua VM untuk webserver (web1 dan web2)

7.4 Koneksi SSH ke Virtual Machine (Windows)

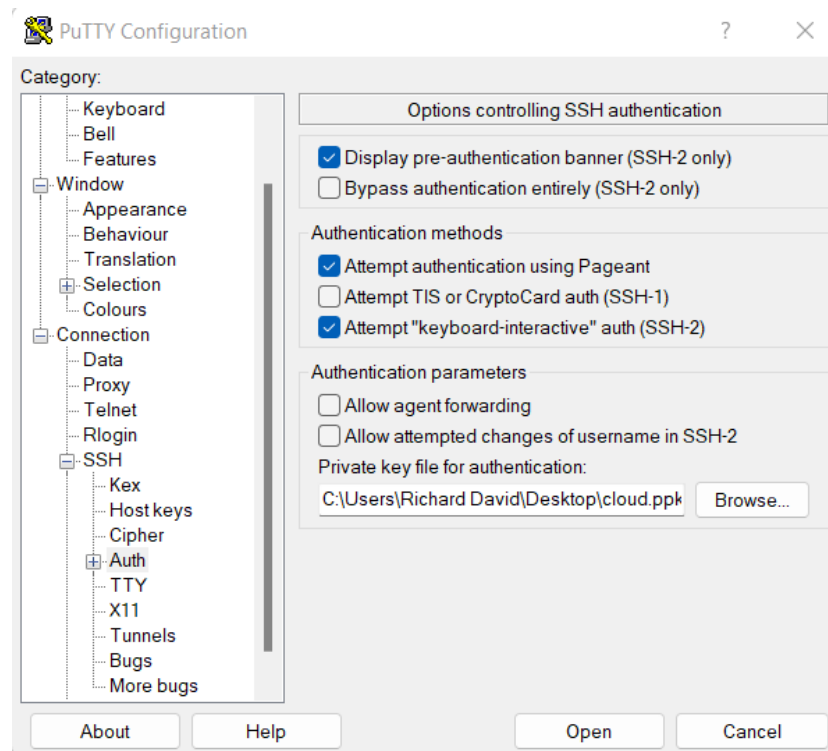
- Sub-bab ini membahas mengenai koneksi SSH ke Azure VM pada sistem operasi Windows. Jika Anda menggunakan Linux, maka Anda dapat melewati bagian ini dan langsung menuju sub-bab 7.4.
- Pada sub-bab 7.1, Anda telah melakukan instalasi PuTTY. Jalankan **PuTTYgen**, yaitu sebuah aplikasi yang dapat melakukan *generate* public dan private SSH key pair. PuTTYgen telah terinstalasi secara otomatis pada saat Anda melakukan instalasi PuTTY.
- Klik tombol **Load**, lalu pilih key pair **cloudkey.pem** yang telah Anda unduh dari Azure
- Klik tombol **Save private key** untuk menyimpan file *private key* dengan ekstensi **.ppk**. File ini akan digunakan untuk melakukan koneksi SSH dengan PuTTY.



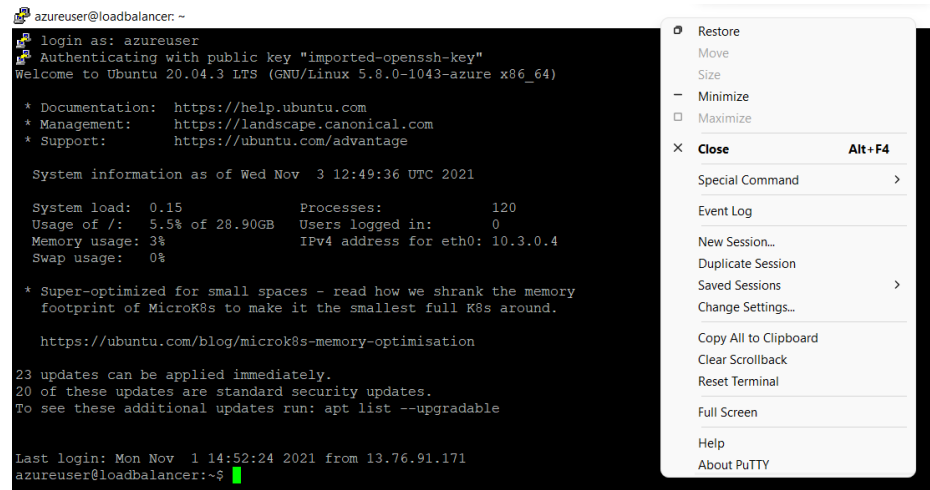
- Tutup PuTTYgen, kemudian jalankan **PuTTY**. Aplikasi ini merupakan *open-source* Telnet/SSH Client, yang dapat digunakan untuk melakukan koneksi SSH ke VM yang terdapat pada *commercial cloud*.
- Masukkan alamat IP dari VM **loadbalancer** pada kolom **Host Name**
- Pada pilihan **Category**, pilih menu **SSH > Auth**



- h) Pilih **file private key** dengan ekstensi **.ppk** yang sudah dilakukan generate menggunakan PuTTYgen



- i) Klik tombol **Open**
- j) Pada terminal yang terbuka, masukkan **azureuser** ketika diminta untuk memasukkan *username*. Kemudian tekan enter untuk membuka koneksi
- k) Klik kanan pada bagian atas terminal, kemudian pilih **new session** untuk membuka sebuah koneksi SSH baru



- l) Ulangi langkah diatas untuk melakukan koneksi SSH terhadap dua VM lainnya (**web1** dan **web2**)

7.5 Koneksi SSH ke Virtual Machine (Linux)

- a) Sub-bab ini membahas mengenai koneksi SSH ke Azure VM pada sistem operasi Linux. Jika Anda menggunakan Windows, maka Anda dapat mengabaikan bagian ini
- b) Ubah working directory Anda menjadi direktori dimana Anda menyimpan key pair dengan ekstensi .pem yang telah Anda unduh dari Azure
- c) Jalankan perintah berikut pada terminal:

```
ssh -i [NAMA KEY PAIR.PEM] azureuser@[ALAMAT IP VM]
```

- d) Ulangi langkah diatas hingga ketiga VM berhasil terkoneksi

PRAKTIKUM 8

TERRAFORM PROVISIONING

Pada praktikum ini, akan dilakukan percobaan untuk melakukan provisioning terhadap Azure VM dengan menggunakan provisioner Terraform. Provisioner tersebut merupakan open-source infrastructure as code serupa dengan Ansible. Secara garis besar, perbedaan Terraform dengan Ansible terletak pada fokus ruang lingkupnya. Terraform berfokus pada manajemen infrastruktur secara keseluruhan, sedangkan Ansible berfokus pada otomasi konfigurasi dari suatu aplikasi atau sistem operasi.

8.1 Import Resource Group

- a) Jalankan Azure Cloud Shell yang dapat diakses pada bagian atas halaman Azure Portal
- b) Jalankan perintah berikut pada terminal Cloud Shell yang terbuka:

```
az group show --name cloudRG --query id --output tsv
```

- c) Perintah tersebut akan menampilkan **Azure Subscription ID** dari *resource group* **cloudRG** yang telah dibuat. Azure Subscription ID tersebut akan digunakan pada saat proses *import* dengan Terraform
- d) Buat sebuah Terraform Configuration File dengan mengetikkan perintah berikut pada Cloud Shell:

```
nano main.tf
```

- e) Pada contoh ini, akan dilakukan percobaan *provisioning* ketiga VM dengan *provisioner* Terraform. Pada VM **loadbalancer**, Terraform akan secara otomatis melakukan instalasi HAProxy, sedangkan pada kedua VM webserver (**web1** dan **web2**), Terraform akan secara otomatis melakukan instalasi Apache2. Modifikasi isi dari **main.tf** hingga menjadi seperti berikut (sesuaikan variabel-variabel yang perlu diubah):

```

provider "azurerm" {
  features{}
}

# create resource group
resource "azurerm_resource_group" "rg"{
  name = "cloud"
  location = "southeastasia"
}
resource "null_resource" "loadbalancer" {

  connection {
    type = "ssh"
    host = "[ALAMAT IP VM LOADBALANCER]"
    user = "azureuser"
    private_key
= file("/home/[USERNAME]/[NAMA KEY PAIR.PEM]")
  }
  provisioner "remote-exec" {
    inline = [
      "sudo apt-get update",
      "sudo apt-get install haproxy -y",
    ]
  }
}
resource "null_resource" "web1" {

  connection {
    type = "ssh"
    host = "[ALAMAT IP VM WEB1]"
    user = "azureuser"
    private_key
= file("/home/[USERNAME]/[NAMA KEY PAIR.PEM]")
  }
  provisioner "remote-exec" {
    inline = [
      "sudo apt-get update",
      "sudo apt-get install apache2 -y",
    ]
  }
}
resource "null_resource" "web2" {

  connection {
    type = "ssh"
    host = "[ALAMAT IP VM WEB2]"
    user = "azureuser"
    private_key
= file("/home/[USERNAME]/[NAMA KEY PAIR.PEM]")
  }
  provisioner "remote-exec" {
    inline = [
      "sudo apt-get update",
      "sudo apt-get install apache2 -y",
    ]
  }
}

```

- f) Jalankan perintah berikut untuk menginstruksikan Terraform agar menginisialisasi *working directory* pada Cloud Shell menjadi sebuah lingkungan eksekusi Terraform. *Plugin* yang diperlukan untuk mendukung integrasi dengan VM Azure yang telah dibuat juga akan dimuat pada tahap ini.

```
terraform init
```

- g) Import *resource group* `cloudRG` yang telah dibuat ke dalam manajemen Terraform, dengan menjalankan perintah berikut pada Cloud Shell:

```
terraform import azurerm_resource_group.rg /[SUBSCRIPTION ID]
```

- h) Upload *key pair* dengan ekstensi `.pem` yang telah diunduh dari Azure ke dalam Cloud Shell (dapat dilakukan dengan cara *drag-and-drop*)
- i) Jalankan perintah berikut untuk memeriksa rencana perubahan-perubahan yang akan dilakukan oleh Terraform terhadap ketiga VM berdasarkan konfigurasi yang didefinisikan dalam `main.tf`

```
terraform plan
```

- j) Apabila rencana perubahan tersebut sudah sesuai dengan konfigurasi yang diinginkan, maka jalankan perintah berikut untuk mengeksekusi perubahan-perubahan tersebut:

```
terraform apply
```

- k) Setelah proses *provisioning* selesai, lakukan koneksi SSH kepada masing-masing VM. Periksa apakah *software* yang diminta sudah terinstalasi dalam VM dengan menjalankan perintah berikut:

Untuk VM `loadbalancer`:

```
sudo systemctl status haproxy
```

Untuk VM `web1` dan `web2`:

```
sudo systemctl status apache2
```

PRAKTIKUM 9

APPLICATIONS ON THE CLOUD

Pada praktikum ini, akan dilakukan percobaan untuk melakukan instalasi aplikasi berbasis web seperti Wordpress dan Moodle, pada kedua VM webserver yang telah dibuat pada praktikum sebelumnya. Praktikum ini juga akan memperlihatkan bagaimana Terraform dapat digunakan untuk re-provisioning VM sesuai dengan kebutuhan.

9.1 Terraform Re-provisioning

- a) Praktikum ini akan mempergunakan ketiga VM yang telah dilakukan *provisioning* oleh Terraform pada praktikum sebelumnya. Terraform telah melakukan provisioning dengan menginstalasi Apache2 pada VM web1 dan web2, namun untuk dapat menjalankan aplikasi berbasis web seperti Wordpress dan Moodle, dibutuhkan perangkat lunak (*dependency*) lainnya seperti PHP dan MySQL. Oleh karena beberapa perangkat lunak yang dibutuhkan belum terinstalasi, maka perlu dilakukan *re-provisioning* (modifikasi konfigurasi Terraform dan menjalankan provisioning kembali) untuk melakukan instalasi perangkat lunak yang dibutuhkan tersebut.
- b) Praktikum ini akan melakukan percobaan untuk melakukan instalasi Wordpress pada VM **web1**, dan Moodle pada VM **web2**.
- c) Jika terdapat perubahan alamat IP VM, sesuaikan pada file konfigurasi Terraform **main.tf**
- d) Ubah file konfigurasi Terraform **main.tf** pada bagian **resource "null_resource" "web1"** menjadi seperti berikut:
Perubahan terdapat pada bagian **provisioner "remote-exec"**, dimana Terraform akan diinstruksikan untuk melakukan instalasi perangkat lunak pendukung Wordpress, seperti MySQL Server, dan beberapa dependency PHP. Kemudian Terraform akan mengunduh *package* instalasi Wordpress secara otomatis.

```

resource "null_resource" "web1" {
  connection {
    type = "ssh"
    host = "20.212.117.95"
    user = "azureuser"
    private_key = file("/home/[NAMA-ANDA]/[NAMA-KEY].pem")
  }
  provisioner "remote-exec" {
    inline = [
      "sudo apt-get update",
      "sudo apt-get install apache2 ghostscript libapache2-
      mod-php mysql-server php php-bcmath php-curl php-imagick
      php-intl php-json php-mbstring php-mysql php-xml php-zip
      -y",
      "sudo mkdir -p /srv/www",
      "sudo chown www-data: /srv/www",
      "curl https://wordpress.org/latest.tar.gz | sudo -u www-
      data tar zx -C /srv/www"
    ]
  }
}

```

- e) Ubah file konfigurasi Terraform `main.tf` pada bagian `resource "null_resource" "web2"` menjadi seperti berikut:

Perubahan terdapat pada bagian `provisioner "remote-exec"`, dimana Terraform akan diinstruksikan untuk melakukan instalasi perangkat lunak pendukung Moodle, seperti MySQL Server, dan beberapa dependency PHP. Kemudian Terraform akan mengunduh *package* instalasi Moodle secara otomatis menggunakan Git, dan mengkonfigurasi direktori instalasi yang diperlukan.

```

resource "null_resource" "web2" {
  connection {
    type = "ssh"
    host = "20.212.117.64"
    user = "azureuser"
    private_key = file("/home/[NAMA-ANDA]/[NAMA-KEY].pem")
  }
  provisioner "remote-exec" {
    inline = [
      "sudo add-apt-repository ppa:ondrej/php -y",
      "sudo apt-get update",
      "sudo apt-get install apache2 mysql-client mysql-
server php7.4 libapache2-mod-php7.4 graphviz aspell
ghostscript clamav php7.4-pspell php7.4-curl
php7.4-gd php7.4-intl php7.4-mysql php7.4-xml
php7.4-xmlrpc php7.4-ldap php7.4-zip php7.4-soap
php7.4-mbstring -y",
      "sudo service apache2 restart",
      "sudo apt install git -y",
      "cd /opt",
      "sudo git clone git://git.moodle.org/moodle.git",
      "cd moodle",
      "sudo git branch -a",
      "sudo git branch
--track MOODLE_39_STABLE origin/MOODLE_39_STABLE",
      "sudo git checkout MOODLE_39_STABLE",
      "sudo cp -R /opt/moodle /var/www/html/",
      "sudo mkdir /var/moodledata",
      "sudo chown -R www-data /var/moodledata",
      "sudo chmod -R 777 /var/moodledata",
      "sudo chmod -R 0755 /var/www/html/moodle",
    ]
  }
}

```

- f) Jalankan perintah berikut untuk menginformasikan kepada Terraform bahwa VM **web1** dan **web2** membutuhkan *re-provisioning*:

```

terraform taint null_resource.web1
terraform taint null_resource.web2

```

- g) Jalankan perintah berikut untuk mengeksekusi proses *re-provisioning*

```

terraform apply

```


9.2 Instalasi Wordpress pada VM **web1**

- a) Lakukan koneksi SSH dengan VM **web1**
- b) Jalankan perintah berikut untuk membuat file konfigurasi Apache Site yang akan digunakan oleh Wordpress

```
sudo nano /etc/apache2/sites-available/wordpress.conf
```

- c) Isi file konfigurasi tersebut adalah sebagai berikut:

```
<VirtualHost *:80>
    DocumentRoot /srv/www/wordpress
    <Directory /srv/www/wordpress>
        Options FollowSymLinks
        AllowOverride Limit Options FileInfo
        DirectoryIndex index.php
        Require all granted
    </Directory>
    <Directory /srv/www/wordpress/wp-content>
        Options FollowSymLinks
        Require all granted
    </Directory>
</VirtualHost>
```

- d) Muat ulang konfigurasi Apache2 dengan perintah berikut:

```
sudo a2ensite wordpress
sudo a2enmod rewrite
sudo a2dissite 000-default
sudo service apache2 reload
```

- e) Sebelum melakukan instalasi Wordpress, Anda perlu melakukan konfigurasi database yang akan digunakan. Masuk ke MySQL CLI dengan menjalankan perintah berikut:

```
sudo mysql -u root
```

- f) Jalankan beberapa SQL *query* berikut untuk melakukan konfigurasi *database* Wordpress. Pada *query* kedua, Anda perlu mendefinisikan sebuah *password* untuk mengamankan *database* yang dibuat.

```
CREATE DATABASE wordpress;
CREATE USER wordpress@localhost IDENTIFIED BY
'[PASSWORD ANDA]';
GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP,ALTER
-> ON wordpress.*
-> TO wordpress@localhost;
FLUSH PRIVILEGES;
QUIT;
```

- g) Jalankan perintah berikut untuk memuat ulang MySQL Server:

```
sudo service mysql start
```

- h) Jalankan perintah berikut untuk mengkonfigurasi Wordpress agar menggunakan *database* yang telah dibuat. Ubah hanya perintah keempat dimana Anda perlu memasukkan *password* yang Anda definisikan pada langkah sebelumnya:

```
sudo -u www-data cp /srv/www/wordpress/wp-config-
sample.php /srv/www/wordpress/wp-config.php
sudo -u www-data sed -i 's/database_name_here/wordpress/'
/srv/www/wordpress/wp-config.php
sudo -u www-data sed -i 's/username_here/wordpress/'
/srv/www/wordpress/wp-config.php
sudo -u www-data sed -i 's/password_here/[PASSWORD ANDA]/'
/srv/www/wordpress/wp-config.php
```

- i) Jalankan perintah berikut untuk membuka file konfigurasi Wordpress:

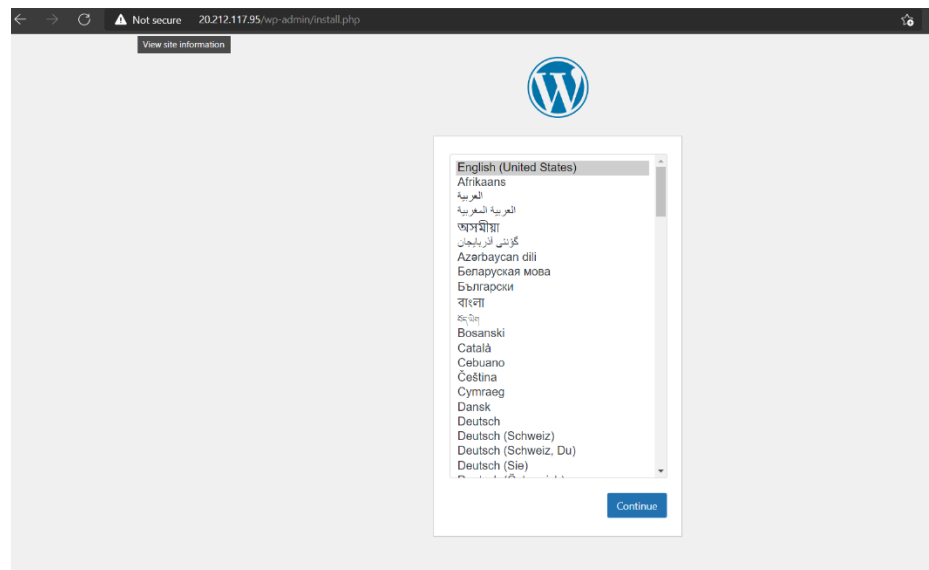
```
sudo -u www-data nano /srv/www/wordpress/wp-config.php
```

- j) Cari baris-baris berikut pada file konfigurasi tersebut:

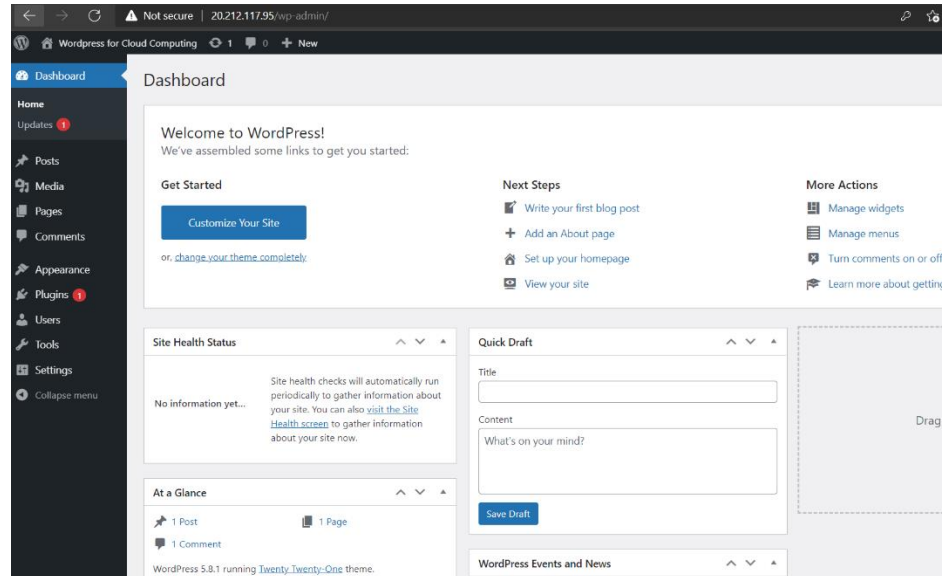
```
define( 'AUTH_KEY', 'put your unique phrase here' );
define( 'SECURE_AUTH_KEY', 'put your unique phrase here' );
define( 'LOGGED_IN_KEY', 'put your unique phrase here' );
define( 'NONCE_KEY', 'put your unique phrase here' );
define( 'AUTH_SALT', 'put your unique phrase here' );
define( 'SECURE_AUTH_SALT', 'put your unique phrase here' );
define( 'LOGGED_IN_SALT', 'put your unique phrase here' );
define( 'NONCE_SALT', 'put your unique phrase here' );
```

Ubah baris-baris tersebut dengan konten yang terdapat pada tautan berikut: <https://api.wordpress.org/secret-key/1.1/salt/>. Tautan tersebut merupakan *randomizer* yang dapat *generate* data secara acak (salt) yang berfungsi untuk meningkatkan keamanan situs Wordpress Anda.

- k) Buka alamat IP dari VM **web1** pada *browser*. Anda akan otomatis diarahkan pada halaman instalasi Wordpress seperti berikut. Ikuti petunjuk instalasi hingga selesai.



- l) Setelah instalasi selesai, situs Wordpress Anda dapat diakses dengan mengetikkan alamat IP dari VM **web1** pada *browser*.



- m) Apabila alamat IP dari VM berubah (misalnya, VM dimatikan lalu dihidupkan kembali), maka Anda perlu mengubah konfigurasi database terlebih dahulu untuk dapat mengakses situs Wordpress Anda. Jalankan perintah berikut untuk masuk ke MySQL CLI:

```
sudo mysql -u root
```

- n) Jalankan beberapa SQL *query* berikut untuk mengubah konfigurasi alamat IP yang tersimpan pada database:

```
USE WORDPRESS;  
SELECT * FROM wp_options WHERE option_id = 1;  
UPDATE wp_options SET option_value = '[ALAMAT IP BARU]'  
WHERE option_id = 1;  
exit;
```

- o) Jalankan perintah berikut untuk memuat ulang MySQL Server:

```
sudo systemctl restart mysql
```

9.3 Instalasi Moodle pada VM **web2**

- a) Lakukan koneksi SSH dengan VM **web2**
- b) Sebelum melakukan instalasi Moodle, Anda perlu melakukan konfigurasi database yang akan digunakan. Masuk ke MySQL CLI dengan menjalankan perintah berikut:

```
sudo mysql -u root
```

- c) Jalankan beberapa SQL *query* berikut untuk melakukan konfigurasi *database* Moodle. Pada *query* kedua, Anda perlu mendefinisikan sebuah *password* untuk mengamankan *database* yang dibuat.

```
CREATE DATABASE moodle DEFAULT CHARACTER SET utf8mb4  
COLLATE utf8mb4_unicode_ci;  
CREATE USER 'moodle'@'localhost'  
IDENTIFIED BY '[PASSWORD ANDA]';  
GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,CREATE  
TEMPORARY TABLES,DROP,INDEX,ALTER  
ON moodle.* TO 'moodle'@'localhost';  
QUIT;
```

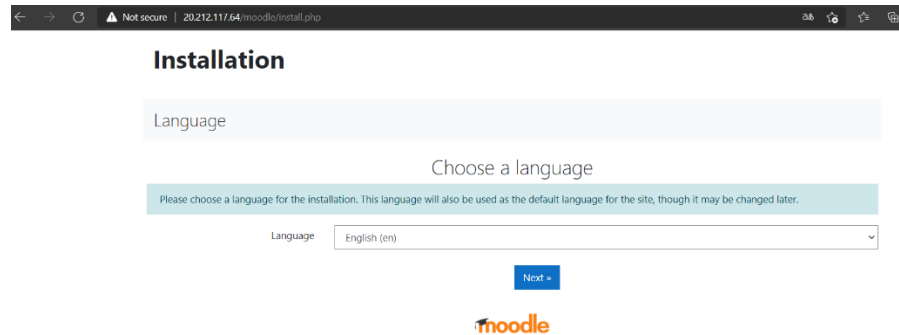
- d) Jalankan perintah berikut untuk memuat ulang MySQL Server:

```
sudo systemctl restart mysql
```

- e) Jalankan perintah berikut untuk mengubah *permission* pada folder instalasi Moodle. Perubahan ini diperlukan untuk membuat folder tersebut *writable* pada saat proses instalasi.

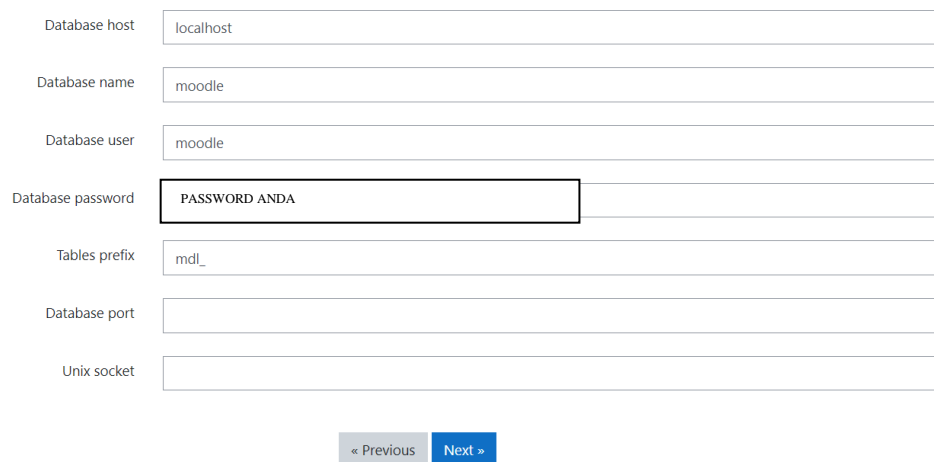
```
sudo chmod -R 777 /var/www  
sudo chmod -R 777 /var/www/html/moodle
```

- f) Buka alamat IP dari VM **web2** pada *browser*, kemudian tambahkan **/moodle** pada akhir alamat IP. Anda akan otomatis diarahkan pada halaman instalasi Moodle seperti berikut. Ikuti petunjuk berikutnya.



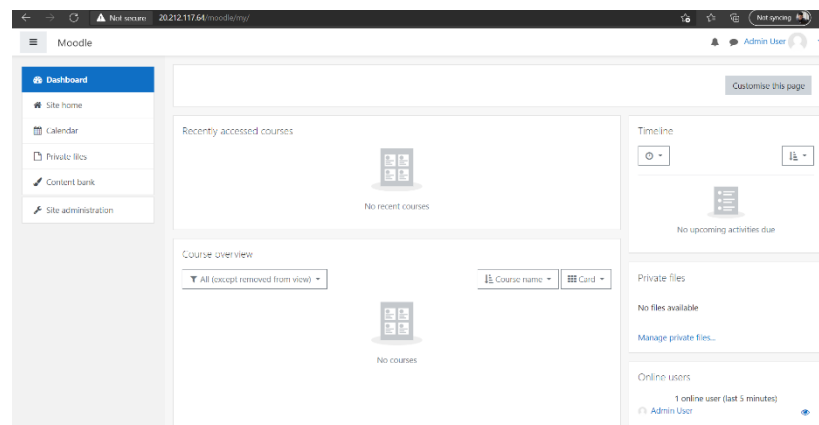
The screenshot shows the Moodle installation 'Installation' page. At the top, it says 'Language'. Below that, a message says 'Choose a language' and 'Please choose a language for the installation. This language will also be used as the default language for the site, though it may be changed later.' There is a dropdown menu for 'Language' with 'English (en)' selected. A 'Next >' button is at the bottom right. The Moodle logo is centered below the button.

- g) Pada halaman konfigurasi database, pastikan Anda mengisi data sesuai dengan format database yang telah dibuat pada **langkah c)**



The screenshot shows the Moodle database configuration page. It has several input fields: 'Database host' (localhost), 'Database name' (moodle), 'Database user' (moodle), 'Database password' (PASSWORD ANDA), 'Tables prefix' (mdl_), 'Database port' (empty), and 'Unix socket' (empty). At the bottom, there are two buttons: '< Previous' and 'Next >'.

- h) Setelah proses instalasi selesai, situs Moodle Anda dapat diakses dengan mengetikkan [**alamat IP dari VM web2/moodle**] pada *browser*.



The screenshot shows the Moodle dashboard after installation. The top bar says 'Moodle' and 'Admin User'. The left sidebar has a 'Dashboard' menu with links to 'Site home', 'Calendar', 'Private files', 'Content bank', and 'Site administration'. The main content area has a 'Recently accessed courses' section with 'No recent courses', a 'Course overview' section with 'No courses', a 'Timeline' section with 'No upcoming activities due', a 'Private files' section with 'No files available', and an 'Online users' section showing '1 online user (last 5 minutes)' and 'Admin User'.

- i) Apabila alamat IP dari VM berubah (misalnya, VM dimatikan lalu dihidupkan kembali), maka Anda perlu mengubah file konfigurasi `config.php` terlebih dahulu untuk dapat mengakses situs Moodle Anda. Jalankan perintah berikut untuk mengubah file tersebut:

```
cd /var/www/html/moodle
sudo nano config.php
```

- j) Ubah nilai variabel `$CFG->wwwroot` dengan alamat IP VM yang baru
k) Jalankan perintah berikut untuk memuat ulang webserver Apache2:

```
sudo systemctl restart apache2
```

9.4 Workload Testing

- a) Lakukan koneksi SSH dengan VM web1, kemudian konfigurasi *load balancer* seperti yang Anda sudah lakukan pada Praktikum 5
b) Buat Test Plan dan lakukan *workload testing* seperti yang Anda sudah lakukan pada Praktikum 6
c) Lakukan observasi pada *latency* ketika *threads* dikirim oleh JMeter dan ketika *threads* diterima oleh HAProxy

PRAKTIKUM 10

DOCKER - SINGLE CONTAINER

Pada praktikum ini, akan dilakukan percobaan untuk melakukan integrasi Docker dengan Azure Container Instances (ACI), yang memungkinkan pengguna untuk melakukan *deployment* aplikasi atau layanan pada lingkungan eksekusi *cloud*. Praktikum ini akan mempergunakan *single container* dan melakukan *deployment* aplikasi yang sudah terkonfigurasi dalam bentuk Docker Image.

10.1 Instalasi Docker

- a) Unduh Docker dari link berikut:

<https://docs.docker.com/get-docker/>

- b) Jalankan terminal lalu ketikkan perintah berikut untuk mengintegrasikan Docker dengan akun Azure Anda. Browser akan terbuka secara otomatis dimana Anda dapat login ke dalam akun Azure for Students Anda.

```
docker login azure
```

10.2 Membuat dan Menjalankan Azure Container Instances (ACI)

- a) Jalankan perintah berikut. Ubah **[NAMA-ACI]** sesuai keinginan Anda. Perintah tersebut akan membuat Azure Container Instances (ACI) pada akun Azure for Students Anda.

```
docker context create aci [NAMA-ACI]
```

- b) Gunakan Resource Group yang telah dibuat pada praktikum sebelumnya
- c) Jalankan perintah berikut untuk menampilkan daftar Docker Context yang telah dibuat. Pastikan daftar tersebut memuat nama ACI yang baru saja dibuat.

```
docker context ls
```


- d) Integrasi Azure ACI dengan Docker terletak pada tingkat Docker Context. Untuk dapat menggunakan ACI yang telah dibuat, maka Anda perlu menginstruksikan Docker agar menggunakan *context* yang terhubung dengan ACI. Jalankan perintah berikut untuk mengubah *context* yang digunakan.

```
docker context use [NAMA-ACI]
```

- e) Jalankan perintah berikut untuk melakukan *deployment* satu *container* ACI terinstalasi dengan *image* Nginx. Anda dapat melakukan *deployment* dengan *image* lainnya, seperti HTTPD atau Wordpress. Perhatikan bahwa dalam perintah tersebut, Anda juga menginstruksikan Docker untuk membuka (*expose*) port 80 pada ACI tersebut.

```
docker run -p 80:80 nginx
```

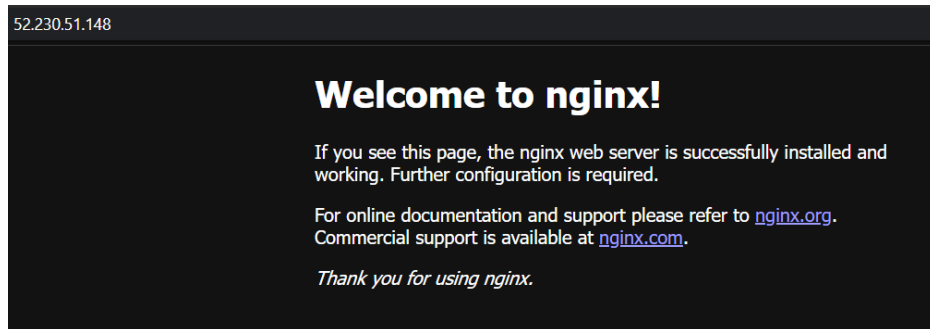
- f) Ketika perintah tersebut dijalankan, maka Docker akan melakukan *generate* Container ID secara acak. Dalam contoh berikut, Container ID dari Azure ACI tersebut adalah **zealous-payne**

```
PS C:\Users\Richard David> docker run -p 80:80 nginx
[+] Running 2/2
 - Group zealous-payne Created
 - zealous-payne Created
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
```

- g) Buka sebuah jendela terminal baru, lalu jalankan perintah berikut untuk menampilkan daftar *container* yang sedang berjalan. Anda dapat melihat alamat IP yang terhubung dengan Container ID dari Azure ACI yang telah dibuat.

```
docker ps
```

- h) Jalankan browser Anda, lalu ketikkan alamat IP dari Azure ACI tersebut. Jika Anda menggunakan *image* Nginx pada **langkah d)**, maka halaman *default* dari Nginx akan tampil.



- i) Jalankan browser Anda, lalu ketikkan alamat IP dari Azure ACI tersebut. Halaman *default* dari Nginx akan tampil.
- j) Jalankan perintah berikut untuk menghentikan Azure ACI

```
docker stop [Container-ID]
```

- k) Jalankan perintah berikut untuk menghapus Azure ACI

```
docker rm [Container-ID]
```

PRAKTIKUM 11

DOCKER - MULTI CONTAINER

Pada praktikum ini, akan dilakukan percobaan untuk melakukan *deployment* sebuah aplikasi pada lebih dari satu ACI (*multi-container*). ACI yang akan digunakan terdiri dari satu ACI untuk *frontend* dan satu ACI untuk backend. Pada praktikum ini tidak akan menggunakan Docker Image seperti pada praktikum sebelumnya, namun dilakukan percobaan untuk melakukan *deployment* aplikasi menggunakan Docker Compose File.

11.1 Instalasi Azure CLI

- a) Unduh Azure CLI dari link berikut:

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

- b) Jalankan terminal lalu ketikkan perintah berikut untuk mengintegrasikan Azure CLI dengan akun Azure Anda. Browser akan terbuka secara otomatis dimana Anda dapat login ke dalam akun Azure for Students Anda.

```
az login
```

11.2 Membuat Azure Container Registry (ACR)

- a) Jalankan perintah berikut untuk membuat sebuah Azure Container Registry (ACR), yang akan digunakan untuk membuat sebuah ACI dan melakukan *deployment* sebuah *custom image* kedalamnya. Anda dapat memberi nama ACR sesuai dengan keinginan Anda, namun perlu diperhatikan bahwa format penamaan ACR harus dalam huruf kecil (*lowercase*).

```
az acr create --resource-group [NAMA-RESOURCE-GROUP]
--name [NAMA-ACR] --sku Basic
```

- b) Jalankan perintah berikut untuk login ke dalam ACR

```
az acr login --name [NAMA-ACR]
```

11.3 Deploy Aplikasi Multi-Container pada Local Machine

- a) Jalankan perintah berikut untuk mengunduh sebuah contoh aplikasi dari Github milik Azure

```
git clone https://github.com/Azure-Samples/azure-voting-app-redis.git

cd azure-voting-app-redis
```

- b) Buka aplikasi teks editor seperti Notepad++ atau Sublime, lalu ubah file `docker-compose.yaml` menjadi seperti berikut. Hal-hal yang perlu diubah terletak pada bagian `azure-vote-front` berikut:

- 1) Ubah variabel `image` menjadi `[NAMA-ACR].azurecr.io/azure-vote-front`
- 2) Ubah pemetaan `port` menjadi `80:80`

```
version: '3'
services:
  azure-vote-back:
    image: mcr.microsoft.com/oss/bitnami/redis:6.0.8
    container_name: azure-vote-back
    environment:
      ALLOW_EMPTY_PASSWORD: "yes"
    ports:
      - "6379:6379"

  azure-vote-front:
    build: ./azure-vote
    image: clouduph.azurecr.io/azure-vote-front
    container_name: azure-vote-front
    environment:
      REDIS: azure-vote-back
    ports:
      - "80:80"
```

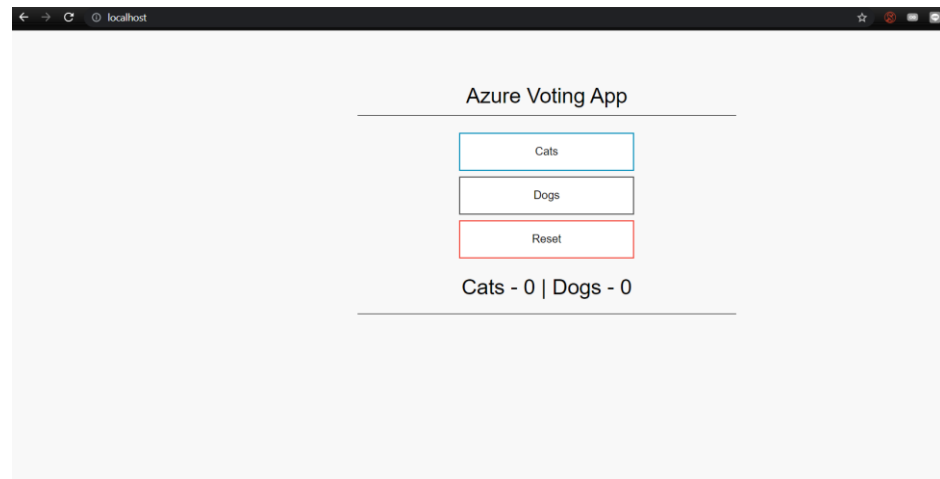
- c) Jalankan perintah berikut untuk menggunakan *context default*

```
docker context use default
```

- d) Jalankan perintah berikut untuk membangun *container image*, mengunduh *image* Redis, dan menjalankan aplikasi

```
docker-compose up --build -d
```

- e) Setelah *container* berhasil terbuat, buka <http://localhost:80> pada browser Anda untuk mengakses aplikasi secara lokal. Tampilannya akan seperti berikut.



- f) Jalankan perintah berikut untuk menghentikan *container*

```
docker-compose down
```

11.4 Deploy Aplikasi Multi-Container pada ACI

- a) Jalankan perintah berikut untuk melakukan mengunggah *image* aplikasi tersebut ke Azure Container Registry (ACR).

```
docker-compose push
```

- b) Jalankan perintah berikut untuk memastikan bahwa *image* berhasil terunggah dengan sempurna

```
az acr repository show --name clouduph --repository azure-vote-front
```

- c) Jika terminal menampilkan JSON seperti berikut, maka *image* berhasil terunggah ke dalam ACR

```
PS D:\UPH\LABS\Cloud\Docker-Multi\azure-voting-app-redis> az acr repository show --name clouduph --repository azure-vote-front
{
  "changeableAttributes": {
    "deleteEnabled": true,
    "listEnabled": true,
    "readEnabled": true,
    "teleportEnabled": false,
    "writeEnabled": true
  },
  "createdTime": "2021-11-07T16:44:07.5362561Z",
  "imageName": "azure-vote-front",
  "lastUpdateTime": "2021-11-07T16:44:07.6603381Z",
  "manifestCount": 1,
  "registry": "clouduph.azurecr.io",
  "tagCount": 1
}
```

- d) Jalankan perintah berikut agar Docker menggunakan *context* yang terintegrasi dengan ACI.

```
docker context use [NAMA-ACI]
```

- e) Gunakan perintah berikut untuk menginstruksikan Docker agar menjalankan aplikasi pada ACI menggunakan *image* yang telah Anda *push* ke ACR

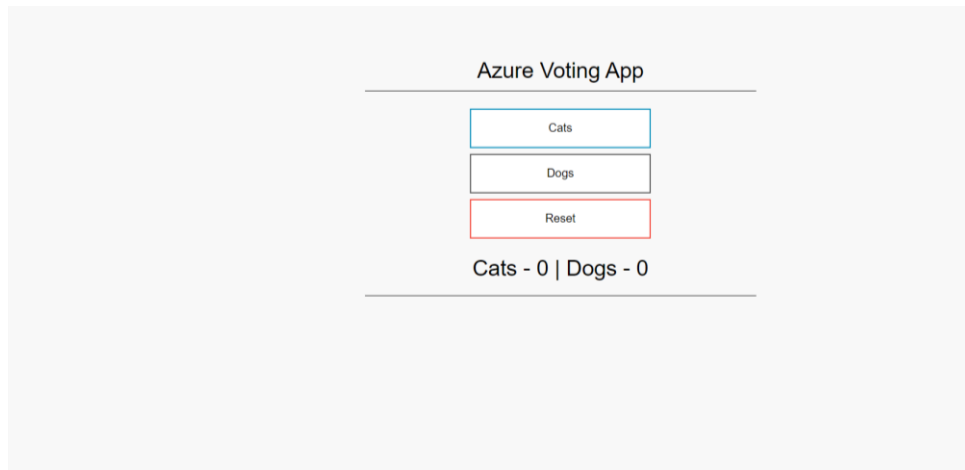
```
docker compose up
```

- f) Setelah proses *deployment* selesai, jalankan perintah berikut untuk mendapatkan alamat IP dari ACI yang berjalan

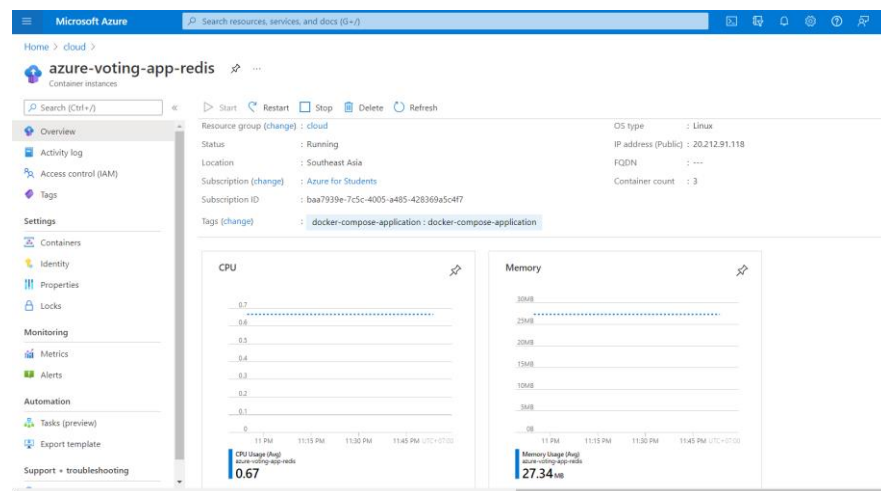
```
docker ps
```

```
PS D:\UPH\LABS\Cloud\Docker-Multi\azure-voting-app-redis> docker ps
CONTAINER ID        IMAGE                                     COMMAND                  STATUS    PORTS
azure-voting-app-redis_azure-vote-back    mcr.microsoft.com/oss/bitnami/redis:6.0.8  redis-server             Running   20.212.91.118:6379->6379/tcp
azure-voting-app-redis_azure-vote-front    clouduph.azurecr.io/azure-vote-front        /start.sh                Running   20.212.91.118:80->80/tcp
PS D:\UPH\LABS\Cloud\Docker-Multi\azure-voting-app-redis>
```

- g) Aplikasi tersebut dapat diakses dengan mengetikkan alamat IP dari ACI pada browser. Tampilan aplikasi tersebut akan sama seperti tampilan ketika berjalan secara lokal.



- h) Anda dapat mengamati statistik dari ACI yang sedang digunakan (penggunaan CPU memori, dan *bandwith*), dengan mengakses menu Container Instances pada Azure Portal



- i) Ketika Anda sudah selesai dengan praktikum ini, jalankan perintah berikut untuk menghentikan ACI

```
docker compose down
```

PRAKTIKUM 12

KUBERNETES CLUSTER

Pada praktikum ini, akan dilakukan percobaan untuk melakukan *deployment* sebuah aplikasi pada Kubernetes Cluster dengan menggunakan Azure Kubernetes Service (AKS). Pada praktikum ini akan dibahas pula mengenai konfigurasi Service Principal, yang diperlukan untuk autentikasi AKS agar dapat mengakses *image* yang terunggah dalam ACR.

12.1 Konfigurasi Service Principal

- a) Praktikum ini akan mempergunakan contoh aplikasi Azure Voting App yang telah diunduh pada praktikum sebelumnya. Agar AKS dapat menarik (*pull*) *image* yang tersimpan pada ACR, maka diperlukan sebuah metode autentikasi yang bernama Service Principal.
- b) Jalankan Azure Cloud Shell, lalu buat sebuah Bash Script dan beri nama `createprincipal`
- c) Isi dari script `createprincipal` adalah sebagai berikut. Anda hanya perlu mengubah nilai variabel `ACR_NAME` dengan nama ACR yang telah dibuat pada praktikum sebelumnya, dan `SERVICE_PRINCIPAL_NAME` sesuai keinginan Anda.

```
#!/bin/bash
# This script requires Azure CLI version 2.25.0
# or later.
# Check version with `az --version`.

ACR_NAME=clouduph
SERVICE_PRINCIPAL_NAME=cloud-principal
ACR_REGISTRY_ID=$(az acr show --name $ACR_NAME --query
"id" --output tsv)

PASSWORD=$(az ad sp create-for-rbac --name
$SERVICE_PRINCIPAL_NAME --scopes $ACR_REGISTRY_ID --
role acrpull --query "password" --output tsv)
USER_NAME=$(az ad sp list --display-name
$SERVICE_PRINCIPAL_NAME --query "[].appId" --output
tsv)

echo "Service principal ID: $USER_NAME"
echo "Service principal password: $PASSWORD"
```


- d) Jalankan perintah berikut untuk mengeksekusi script

```
chmod +x createprincipal  
./createprincipal
```

- e) Terminal akan menampilkan Service Principal ID dan Service Principal Password yang akan digunakan pada langkah berikutnya
- f) Anda perlu memberikan hak akses kepada Service Principal tersebut untuk dapat menarik *image* dari ACR. Pada Azure Cloud Shell, buat sebuah Bash Script dan beri nama `useprincipal`
- g) Isi dari script `useprincipal` tersebut adalah sebagai berikut. Anda hanya perlu mengubah nilai variabel `ACR_NAME` dengan nama ACR yang telah dibuat, dan `SERVICE_PRINCIPAL_ID` yang diperoleh dari langkah sebelumnya

```
#!/bin/bash  
  
ACR_NAME=clouduph  
SERVICE_PRINCIPAL_ID=f5760c35-42f0-4e69-ae7e-6cdf399a350d  
  
ACR_REGISTRY_ID=$(az acr show --name $ACR_NAME --query id  
--output tsv)  
  
az role assignment create --assignee $SERVICE_PRINCIPAL_ID  
--scope $ACR_REGISTRY_ID --role acrpull
```

- h) Jalankan perintah berikut untuk mengeksekusi script. Service Principal kini siap untuk digunakan.

```
chmod +x useprincipal  
./useprincipal
```

12.2 Membuat dan Mengakses Cluster Azure Kubernetes Services (AKS)

- a) Buat sebuah *cluster* Azure Kubernetes Services (AKS) dengan perintah berikut. Anda dapat memberi nama AKS sesuai dengan keinginan Anda.

```
az aks create --resource-group [NAMA-RESOURCE-GROUP]
--name [NAMA-AKS] --node-count 2 --generate-ssh-keys
--attach-acr [NAMA-ACR] --service-principal
[SERVICE-PRINCIPAL-ID] --client-secret
[SERVICE-PRINCIPAL-PASSWORD]
```

- b) Jalankan perintah berikut untuk mengakses AKS Cluster tersebut

```
az aks get-credentials --resource-group
[NAMA-RESOURCE-GROUP] --name [NAMA-AKS]
```

- c) Untuk melihat jumlah *nodes* yang berjalan pada cluster AKS, Anda dapat menggunakan perintah berikut.

```
kubectl get nodes
```

```
PS D:\UPH\LABS\Cloud\Kubernetes\azure-voting-app-redis> az aks get-credentials --resource-group cloud --name cloudAKS
Merged "cloudAKS" as current context in C:\Users\Richard David\.kube\config
PS D:\UPH\LABS\Cloud\Kubernetes\azure-voting-app-redis> kubectl get nodes
NAME                                STATUS    ROLES    AGE   VERSION
aks-nodepool1-23059117-vmss000000  Ready    agent    118s  v1.20.9
aks-nodepool1-23059117-vmss000001  Ready    agent    94s   v1.20.9
PS D:\UPH\LABS\Cloud\Kubernetes\azure-voting-app-redis>
```

12.3 Deploy Aplikasi pada Cluster Azure Kubernetes Services (AKS)

- a) Jalankan aplikasi teks editor dan buka file **azure-vote-all-in-one-redis.yaml** yang terdapat pada direktori **azure-voting-app-redis** yang telah diunduh pada praktikum sebelumnya
- b) Modifikasi variabel **image** pada **azure-vote-front** sehingga bagian tersebut akan menjadi seperti berikut.

```
azure-vote-front:
  build: ./azure-vote
  image: [NAMA-ACR]/azure-vote-front:v1
  container_name: azure-vote-front
  environment:
    REDIS: azure-vote-back
  ports:
    - "8080:80"
```

- c) Jalankan perintah berikut untuk melakukan *deployment* aplikasi ke AKS

```
kubectl apply -f azure-vote-all-in-one-redis.yaml
```

- d) Jalankan perintah berikut untuk mendapatkan alamat IP dari cluster AKS tersebut. Pada umumnya proses alokasi alamat IP memakan waktu beberapa menit.

```
kubectl get service azure-vote-front --watch
```

- e) Ketika Anda mengakses alamat IP cluster AKS pada browser, maka aplikasi Azure Voting App akan tampil.
- f) Ketika Anda sudah selesai dengan praktikum ini, jalankan perintah berikut untuk menghentikan AKS Cluster

```
az aks stop --name [NAMA-AKS]  
--resource-group [NAMA-RESOURCE-GROUP]
```

PRAKTIKUM 13

SCALING KUBERNETES CLUSTER

Pada praktikum ini, akan dilakukan percobaan untuk melakukan *scaling* sebuah Azure Kubernetes Services (AKS) Cluster yang didalamnya terdapat aplikasi yang sedang berjalan. Pada suatu *cluster* Kubernetes, *scaling* dapat dilakukan untuk menambah atau mengurangi jumlah *Pods* untuk menyesuaikan kebutuhan (*demand*). Pada praktikum ini akan diperkenalkan metode *horizontal scaling* dengan dua tipe prosedur, yaitu *manual scaling* dan HPA.

13.1 Manual Scaling

- a) Jalankan perintah berikut untuk menyalakan AKS Cluster

```
az aks start --name [NAMA-AKS]
--resource-group [NAMA-RESOURCE-GROUP]
```

- b) Untuk melihat jumlah *Pods* yang berjalan, gunakan perintah berikut. Pada saat ini terdapat masing-masing satu *pod* untuk *azure-vote-back* dan *azure-vote-front*

```
kubectl get pods
```

- c) Lakukan *manual scaling* dengan perintah berikut untuk mereplikasi *pod* *azure-vote-front* sebanyak lima *Pods*.

```
kubectl scale --replicas=5 deployment/azure-vote-front
```

- d) Ketika Anda menjalankan perintah `kubectl get pods` kembali, maka pada saat ini akan terdapat satu *pod* untuk *azure-vote-back* dan lima *Pods* untuk *azure-vote-front*

13.2 Horizontal Pod Autoscaler (HPA)

- a) Untuk dapat menggunakan fitur *autoscale*, maka Anda perlu mendefinisikan batasan-batasan CPU untuk azure-vote-front dalam file **azure-vote-all-in-one-redis.yaml**.
- b) Ubah bagian azure-vote-front pada file **azure-vote-all-in-one-redis.yaml** menjadi sebagai berikut.

```
azure-vote-front:
  build: ./azure-vote
  image: [NAMA-ACR]/azure-vote-front:v1
  container_name: azure-vote-front
  environment:
    REDIS: azure-vote-back
  ports:
    - "8080:80"
  resources:
    requests:
      cpu: 250m
    limits:
      cpu: 500m
```

- c) Melalui perubahan tersebut, cluster AKS akan melakukan permintaan untuk menggunakan 0.25 CPU pada saat pertama dijalankan dan dapat menggunakan sampai dengan 0.5 CPU. Perlu diperhatikan bahwa angka-angka tersebut adalah satuan CPU, dan berbeda dengan angka persentase penggunaan CPU yang akan digunakan pada langkah berikutnya.
- d) Jalankan perintah berikut agar perubahan tersebut tersimpan pada cluster AKS

```
kubectl apply -f azure-vote-all-in-one-redis.yaml
```

- e) Konfigurasi Horizontal Pod Autoscale (HPA) dengan perintah berikut

```
kubectl autoscale deployment azure-vote-front
--cpu-percent=50 --min=3 --max=10
```

- f) Perintah tersebut mengkonfigurasi HPA untuk menjalankan minimal tiga *Pods* pada saat penggunaan CPU kurang dari 50%, dan menjalankan hingga 10 *Pods* pada saat *workload* meningkat dan penggunaan CPU melebihi 50%.
- g) Anda dapat melihat persentase penggunaan CPU beserta jumlah *Pods* yang sedang berjalan dengan menggunakan perintah berikut

```
kubectl get hpa
```

13.3 Eksperimen Lanjut

- a) Lakukan workload testing menggunakan JMeter pada cluster AKS yang telah terkonfigurasi dengan HPA
- b) Pada saat *workload testing* sedang berlangsung, Anda dapat mengamati kinerja HPA dengan menjalankan perintah `kubectl get hpa`.
- c) Jika Anda sudah selesai dengan praktikum ini, maka Anda dapat menghapus cluster AKS dengan perintah berikut.

```
az aks delete --name [NAMA-AKS]  
--resource-group [NAMA-RESOURCE-GROUP]
```