Vagrant

Esercizio 4 – Multi machine

Introduzione

Nel contesto delle operazioni IT, Vagrant fornisce una soluzione per la creazione di ambienti monouso che mantengono la coerenza tra piattaforme e ambienti virtuali. Una delle grandi caratteristiche di Vagrant è la possibilità di configurare interi ambienti in codice all'interno di un unico file di configurazione (Vagrantfile). Ciò significa che con un solo comando, "vagrant up", è possibile avviare più macchine virtuali in una sola volta, e anche con la loro rete privata.

Questo esercizio vi guiderà attraverso il processo di utilizzo delle configurazioni in Vagrant per creare e distribuire un piccolo ambiente di test

Procedimento

La configurazione per questo esercizio verrà trattata in varie fasi.

File base

Creare un Vagrantfile con questo contenuto:

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

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

  config.vm.define "web" do |subconfig|
    subconfig.vm.box = "hashicorp/precise64"
  end

  config.vm.define "db" do |subconfig|
    subconfig.vm.box = "hashicorp/precise64"
  end

end
```

Si tratta di una configurazione minimale e poco efficiente ma che ci permette già di provare alcuni comandi. Infatti il precedente comando “vagrant up” usato in questo contesto avvia entrambe le macchine virtuali, ma se volessimo avviarne una sola delle due dovremmo specificarne il nome

Es:

```
vagrant up web
```

```
vagrant up db
```

Ma aspettiamo un momento prima di dare il via a queste vm.

Variabili

Il Vagrantfile è un file RUBY e quindi un file di scripting completo di variabili e istruzioni.
Siccome entrambe le macchine usano la medesima box possiamo specificarne il nome in una variabile da usare poi nei subconfig.

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

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

      BOX_IMAGE = "hashicorp/precise64"

      config.vm.define "web" do |subconfig|
            subconfig.vm.box = BOX_IMAGE
      end

      config.vm.define "db" do |subconfig|
            subconfig.vm.box = BOX_IMAGE
      end

end
```

Sempre con le variabili possiamo specificare il proxy (aggiornare l'ip se caso).

```
PROXY_ENABLE       = false
PROXY_HTTP         = "http://10.0.2.2:5865"
PROXY_HTTPS        = "http://10.0.2.2:5865"
PROXY_EXCLUDE      = "localhost,127.0.0.1"
```

Da usare nei subconfig

```
subconfig.proxy.http     = PROXY_HTTP
subconfig.proxy.https    = PROXY_HTTPS
subconfig.proxy.no_proxy = PROXY_EXCLUDE
```

Attività

Imposta il proxy nelle due box

Rete

La configurazione di rete base per queste box è una scheda in NAT.
Per facilitare la comunicazione tra le due VM aggiungiamo una scheda configurata come private network interna.

```
config.vm.define "web" do |subconfig|
  subconfig.vm.network "private_network", ip: "10.10.20.10", virtualbox__intnet: true
  …
end

config.vm.define "db" do |subconfig|
  subconfig.vm.network "private_network", ip: "10.10.20.15", virtualbox__intnet: true
  …
end
```

Configurazione che possiamo rendere più flessibile con l'uso di variabili.

```
BASE_NETWORK = "10.10.20"
subconfig.vm.network "private_network", ip: "#{BASE_NETWORK}.10", virtualbox__intnet:
true
```

```
subconfig.vm.network "private_network", ip: "#{BASE_NETWORK}.15", virtualbox__intnet:
true
```

Ecco quindi il risultato finale con anche l'aggiunta di un port forwarding sulla macchina web

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

Vagrant.configure("2") do |config|
  BOX_IMAGE        = "hashicorp/precise64"
  BASE_NETWORK     = "10.10.20"

  PROXY_HTTP       = "http://10.0.2.2:5865"
  PROXY_HTTPS      = "http://10.0.2.2:5865"
  PROXY_EXCLUDE    = "localhost,127.0.0.1"

  config.vm.define "web" do |subconfig|
    subconfig.vm.box = BOX_IMAGE
      subconfig.vm.network "private_network", ip: "#{BASE_NETWORK}.10",
virtualbox__intnet: true
      subconfig.vm.network "forwarded_port", guest: 80, host: 9080
      subconfig.proxy.http     = PROXY_HTTP
      subconfig.proxy.https    = PROXY_HTTPS
      subconfig.proxy.no_proxy = PROXY_EXCLUDE
  end

  config.vm.define "db" do |subconfig|
    subconfig.vm.box = BOX_IMAGE
      subconfig.vm.network "private_network", ip: "#{BASE_NETWORK}.15",
virtualbox__intnet: true

      subconfig.proxy.http     = PROXY_HTTP
      subconfig.proxy.https    = PROXY_HTTPS
      subconfig.proxy.no_proxy = PROXY_EXCLUDE
  end

end
```

Attività

Aggiorna il Vagrantfile

Hostname e VirtualBox

Anche se non c'è un dns nella rete interna, ai fini di simulare un ambiente reale impostiamo l'hostname delle due macchine:

```
subconfig.vm.hostname = "mmweb.cpt.local"
subconfig.vm.hostname = "mmdb.cpt.local"
```

All'esecuzione di "vagrant up" vengono create due nuove VM visibili in VirtualBox con nomi generati da Vagrant.

Possiamo cambiare nomi ed altri parametri sempre da Vagrantfile:

```
subconfig.vm.provider "virtualbox" do |vb|
      vb.name = "MMWeb"
      vb.memory = "1024"
      vb.cpus = 1
```

```
      #alter connection to the vm for more reliable boot
      vb.customize ["modifyvm", :id, "--uart1", "0x3F8", "4"]
      vb.customize ["modifyvm", :id, "--uartmode1", "file", File::NULL]
      #vb.gui = true     #enable only for debugging
end


subconfig.vm.provider "virtualbox" do |vb|
      vb.name = "MMDb"
      vb.memory = "1024"
      vb.cpus = 1
      #alter connection to the vm for more reliable boot
      vb.customize ["modifyvm", :id, "--uart1", "0x3F8", "4"]
      vb.customize ["modifyvm", :id, "--uartmode1", "file", File::NULL]
      #vb.gui = true     #enable only for debugging
end
```

Attività

Aggiorna la configurazione del Vagrantfile con questi nuovi dati

Flags

Per attivare o disattivare alcune funzionalità o parti della configurazione possiamo istanziare delle variabili aggiuntive.

```
  BOX_CHK_UPDATE    = false
  SSH_INSERT_KEY    = false
  PROXY_ENABLE      = false
  VB_CHK_UPDATE     = false
```

Da richiamare direttamente nei subconfig per modificare il comportamento della VM

```
subconfig.vm.box_check_update = BOX_CHK_UPDATE
subconfig.ssh.insert_key = SSH_INSERT_KEY
```

O per cambiare la configurazione dell'ambiente

```
      if PROXY_ENABLE == true
            #only use with plugin vagrant-proxy
            #print "setting proxy config"
            subconfig.proxy.http     = PROXY_HTTP
            subconfig.proxy.https    = PROXY_HTTPS
            subconfig.proxy.no_proxy = PROXY_EXCLUDE
      end

      if Vagrant.has_plugin?("vagrant-vbguest")
            subconfig.vbguest.auto_update = VB_CHK_UPDATE
      end
```

Attività

Imposta le variabili e usale nel Vagrantfile.

Prova a modificare l'attivazione del proxy in modo che venga controllato se il plug-in corrispondente è presente in Vagrant.

Installazione del sw

Le VM sono ora pronte per ospitare i servizi che ci servono.

Nella cartella di progetto creiamo una cartella “scripts”

Con all’interno i seguenti files

`provision_update.sh`

```
#!/bin/bash

echo "APT provisioning - begin"
sudo apt-get update -y
echo "APT provisioning - end"
```

Aggiungere questo script alla fase di provisioning di entrambe le vm.

```
#provisioning
subconfig.vm.provision "shell", path: "./scripts/provision_update.sh"
```

VM web

Creare un nuovo file tra gli script di provisioning.

`provision_apache.sh`

```
#!/bin/bash

echo "Apache2 provisioning - begin"
sudo apt-get install apache2 -y       #install apache
sudo a2enmod rewrite                  #enable mod_rewrite
sudo service apache2 restart    #start apache
echo "Apache2 provisioning - end"
```

Entrambi i files andiamo a configurarli nel Vagrantfile in modo che la macchina web li esegua durante la fase di provisioning.

```
subconfig.vm.provision "shell", path: "./scripts/provision_apache.sh"
```

Attività

Aggiornare il Vagrantfile.

Testare che apache risponda (http://localhost:9080) con una pagina di test.

Creare un file di provisioning per l’installazione di php (>5) comprensivo di modulo mysql per la macchina web

In questa fase si consiglia di tenere la VM “db” spenta e di usare “vagrant up web”, “vagrant destroy web”, “vagrant halt web”.

Un buon modo per simulare l’installazione è quella di usare apt-get nella vm “vagrant ssh web” e poi copiare i comandi nello script. Poi con vagrant destroy / up verificarne il contenuto.

VM db

Creare un nuovo file di provisioning

provision_mysql.sh

Usare questo template per la prossima attività

```
#!/bin/bash

echo "MySql provisioning - begin"

#installare mysql-server
sudo apt-get install [nome moduli] -y

#impostare la pw di root


#abilitare le connessioni da altri server
echo "Updating bind address"
sudo sed -i "s/.*bind-address.*/bind-address = 0.0.0.0/"
/etc/mysql/mysql.conf.d/mysqld.cnf

#riavviare mysql in modo da applicare le modifiche
echo "Restarting mysql service"
sudo service mysql restart

echo "MySql provisioning - end"
```

Attività

Completare file di provisioning per mysql e aggiungerlo alle istruzioni di provisioning nel Vagrantfile.

Impostare l'ambiente di sviluppo

A questo punto entrambe le macchine sono pronte per lo sviluppatore.

Servono solo due cose:

- dei files sorgente per il lato web
- un db applicativo sulla VM db

Nella cartella di progetto creare una nuova cartella "www".

Attività

Cercare nella documentazione di Vagrant come aggiungere una nuova cartella condivisa in modo che la cartella "www" (e il suo contenuto) venga montata direttamente nella cartella "/var/www" della VM.

Creare un nuovo index.html con contenuto personalizzato e verificare tramite browser che le modifiche siano presenti.

Lato DB possiamo creare un nuovo schema tramite script

```
#!/bin/bash

ROOTPASSWD=[password usata precedentemente in provision_mysql.sh]

DBNAME=vagrant
DBUSER=vagrant
DBPASSWD=vagrantpass
```

```
echo "Creating new db $DBNAME"
#INFO if $ROOTPASSWD is not set during setup skip -p parameter
mysql -uroot -p$ROOTPASSWD -e "CREATE DATABASE $DBNAME"
#TODO create user with $DBUSER identified by $DBPASSWD
mysql -uroot -p$ROOTPASSWD -e "grant all privileges on $DBNAME.* to '$DBUSER'@'%'
identified by '$DBPASSWD'"
```

Attività

Aggiornare il provisioning e verificare tramite mysql client che il db esista.

Verificare inoltre che il nuovo schema sia raggiungibile anche da macchina web.

Versioning

Avendo ora un ambiente di sviluppo completo possiamo salvare tutto quanto in un sistema di versioning (come git).

Da notare che la cartella ".vagrant" non va versionata perché contiene i dischi e le configurazioni attive delle VM. Dati che non servono allo sviluppo e che non vogliamo vadano ad inquinare il repository.

La cartella va quindi impostata come da ignore nel corrispondente file di configurazione gitignore.

Attività

Creare un repository su github contenente il Vagrantfile, gli scritp di provisioning e una pagina php che esegua un test di connessione al nuovo database.