

Corso di Laurea Magistrale in Ingegneria Informatica e
Robotica



DIPARTIMENTO DI INGEGNERIA

Corso di laurea magistrale in
INGEGNERIA INFORMATICA E ROBOTICA

Progetto del Virtual Network and Cloud Computing

Candidato Valerio Brunacci

Matricola 310675

Sommario

Obiettivo	3
Strumenti Utilizzati	5
Requisiti	6
Installazioni iniziali	6
Svolgimento	7
Vagrantfile	7
Ansible files	9
Master-playbook	9
Node-playbook	13
Inizializzazione	16
Test	17
Descrizione dettagliata dei File	18
Struttura Vagrantfile	18
Struttura Ansible Files	20
Riferimenti	22

Provisioning a kubernetes cluster using Ansible and Vagrant

Obiettivo

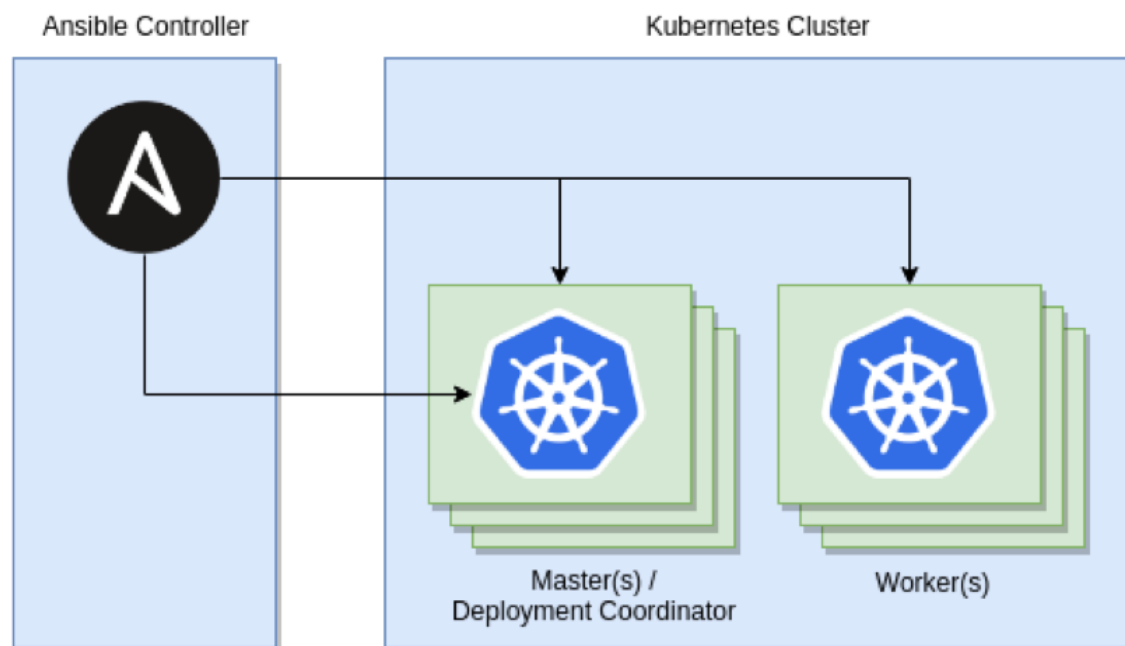
In questo elaborato viene spiegato come istanzianziare un cluster kubernetes di macchine virtuali usando Ansible.

In particolare l'architettura del sistema prevede 3 macchine virtuali :

- 1 macchina master
- 2 macchine slave

Questo cluster è controllato attraverso Kubernetes.

L'inizializzazione delle macchine virtuali avviene sfruttando le API di VirtualBox con Vagrant ed inoltre è stato sfruttato Ansible per automatizzare l'installazione di kubernetes.



Nei nodi del cluster è stato utilizzato Ubuntu debian nella versione “bionic” sia per i client che per il master; non è stato inserito altro Sw nelle VM oltre kubernetes e docker.

Questo task ha necessitato la scrittura di 3 file e l’installazione nella macchina host di Vagrant e Ansible.

Strumenti Utilizzati

come già accennato in precedenza sono 4 i software sfruttati per svolgere il task:

- Virtualbox: è un noto strumento open-source per l'esecuzione di VM; in questo lavoro ne sono state sfruttate le API pubbliche per istanziare il master e i due client.
- Vagrant: gestore di macchine virtuale, anch'esso open-source, fa il paio con l'altro diffusissimo e più noto software di containerizzazione "Docker"; il ruolo che ha svolto in questa tesina è quello di sfruttare le API di virtualbox per costruire macchine virtuali.
- Kubernetes: anche chiamato K8s è un sistema open-source per l'orchestrazione di VM e container; in questo elaborato è servito per la creazione di cluster di macchine virtuali.
- Ansible: software libero, altamente flessibile, agentless, richiede soltanto una connessione ssh nelle macchine target, sfrutta il linguaggio YAML e si occupa di automatizzare le procedure di installazione di quasi qualunque SW; come già accennato sopra, il suo compito è stato quello di automatizzare l'installazione di kubernetes in tutte le macchine create con Vagrant.

Requisiti

Per eseguire correttamente il provisioning di un sistema si è sfruttata una macchina host con SO debian (Ubuntu LTS 20.04) nella quale è stato necessario installare anche Vagrant 2.2.9, Ansible 2.9.6 e VirtualBox 6.1 .

Innanzitutto è bene creare una cartella dedicata al progetto:

```
mkdir kubernetes  
cd kubernetes
```

Installazioni iniziali

installazione VirtualBox :

```
sudo apt update  
sudo apt install virtualbox
```

Installazione Vagrant:

```
curl -O  
https://releases.hashicorp.com/vagrant/2.2.9/vagrant_2.2.9_x86_64.deb  
sudo apt install ./vagrant_2.2.9_x86_64.deb
```

Installazione Ansible:

```
sudo apt install ansible
```

Svolgimento

Il sistema prevede la scrittura di 3 file:

- Vagrantfile
- master-playbook.yaml
- node-playbook.yaml

Nota: una descrizione dettagliata dei file è presente nella sezione successiva

I file possono essere editati con qualunque editor, tuttavia è bene prestare attenzione alla spaziatura dei caratteri specialmente nei due file “.yaml”.

I file vanno inseriti nella cartella “kubernetes” appena creata; una volta completata la loro stesura la cartella avrà il seguente contenuto:

```
valiokei@valiokei-G5-5590:~/Scrivania/kubernetes$ tree
.
├── master-playbook.yaml
├── node-playbook.yaml
└── Vagrantfile

0 directories, 3 files
```

Vagrantfile

Il primo file, come suggerito dal nome, serve per istruire Vagrant su come, quante e con che SW e risorse, macchine virtuali istanziare.

Di seguito è riportato il Vagrantfile:

```
1 IMAGE_NAME = "ubuntu/bionic64"
2 N = 2
3
4 Vagrant.configure("2") do |config|
5     config.ssh.insert_key = false
6
7     config.vm.provider "virtualbox" do |v|
8         v.memory = 1024
9         v.cpus = 2
10    end
11
12    config.vm.define "master" do |master|
13        master.vm.box = IMAGE_NAME
14        master.vm.network "private_network", ip: "192.168.50.10"
15        master.vm.hostname = "master"
16        master.vm.provision "ansible" do |ansible|
17            ansible.playbook = "master-playbook.yml"
18            ansible.extra_vars = {
19                node_ip: "192.168.50.10",
20            }
21        end
22    end
23
24    (1..N).each do |i|
25        config.vm.define "node-#{i}" do |node|
26            node.vm.box = IMAGE_NAME
27            node.vm.network "private_network", ip: "192.168.50.#{i + 10}"
28            node.vm.hostname = "node-#{i}"
29            node.vm.provision "ansible" do |ansible|
30                ansible.playbook = "node-playbook.yml"
31                ansible.extra_vars = {
32                    node_ip: "192.168.50.#{i + 10}",
```


33	}
34	end
35	end
36	end
37	end

Ansible files

Di seguito si riportano i file di configurazioni dei nodi scritti in yaml necessari al funzionamento di Ansible.

Master-playbook

Il seguente file “master-playbook.yaml” specifica le operazioni da eseguire nel nodo master alfine di installare docker e kubernetes.

1	---
2	- hosts: all
3	become: true
4	tasks:
5	- name: Install packages that allow apt to be used over HTTPS
6	apt:
7	name: "{{ packages }}"
8	state: present
9	update_cache: yes
10	vars:
11	packages:
12	- apt-transport-https
13	- ca-certificates
14	- curl
15	- gnupg-agent
16	- software-properties-common

```
17
18 - name: Add an apt signing key for Docker
19   apt_key:
20     url: https://download.docker.com/linux/ubuntu/gpg
21     state: present
22
23 - name: Add apt repository for stable version
24   apt_repository:
25     repo: deb [arch=amd64] https://download.docker.com/linux/ubuntu xenial stable
26     state: present
27
28 - name: Install docker and its dependencies
29   apt:
30     name: "{{ packages }}"
31     state: present
32     update_cache: yes
33     vars:
34       packages:
35         - docker-ce
36         - docker-ce-cli
37         - containerd.io
38     notify:
39       - docker status
40
41 - name: Add vagrant user to docker group
42   user:
43     name: vagrant
44     group: docker
45
46 - name: Disable swap
47   command: swapoff -a
48
49 - name: Add an apt signing key for Kubernetes
50   apt_key:
51     url: https://packages.cloud.google.com/apt/doc/apt-key.gpg
```

```
52     state: present
53
54 - name: Adding apt repository for Kubernetes
55     apt_repository:
56     repo: deb https://apt.kubernetes.io/ kubernetes-xenial main
57     state: present
58     filename: kubernetes.list
59
60 - name: Install Kubernetes binaries
61     apt:
62     name: "{{ packages }}"
63     state: present
64     update_cache: yes
65     vars:
66     packages:
67     - kubelet
68     - kubeadm
69     - kubectl
70     register: installed
71
72 - name: Restart kubelet
73     service:
74     name: kubelet
75     daemon_reload: yes
76     state: restarted
77
78 - name: Initialize the Kubernetes cluster using kubeadm
79     command: kubeadm init --apiserver-advertise-address="192.168.50.10"
--apiserver-cert-extra-sans="192.168.50.10" --node-name master
--pod-network-cidr=192.168.0.0/16
80     when: installed is changed
81
82 - name: Create .kube folder
83     become: false
84     file:
```

```
85     path: /home/vagrant/.kube
86     state: directory
87
88     - name: Copy admin.conf file
89       copy: remote_src=True src=/etc/kubernetes/admin.conf
90       dest=/home/vagrant/.kube/config
91
92     - name: Change admin.conf owner
93       file:
94         path: /home/vagrant/.kube/config
95         owner: vagrant
96         group: vagrant
97
98     - name: Install calico pod network
99       become: false
100       command: kubectl apply -f
101       https://docs.projectcalico.org/v3.10/manifests/calico.yaml
102
103     - name: Generate join command
104       command: kubeadm token create --print-join-command
105       register: join_command
106
107     - name: Copy join command to local file
108       copy:
109         dest: "join"
110         content: "{{ join_command.stdout_lines[0] }}"
111         become: false
112         delegate_to: localhost
113
114     handlers:
115       - name: docker status
116         service: name=docker state=started
```

Node-playbook

Il seguente file “node.yaml” specifica le operazioni da eseguire nei nodi client:

```
1  ---
2  - hosts: all
3    become: true
4    tasks:
5      - name: Install packages that allow apt to be used over HTTPS
6        apt:
7          name: "{{ packages }}"
8          state: present
9          update_cache: yes
10         vars:
11           packages:
12             - apt-transport-https
13             - ca-certificates
14             - curl
15             - gnupg-agent
16             - software-properties-common
17
18      - name: Add an apt signing key for Docker
19        apt_key:
20          url: https://download.docker.com/linux/ubuntu/gpg
21          state: present
22
23      - name: Add apt repository for stable version
24        apt_repository:
25          repo: deb [arch=amd64] https://download.docker.com/linux/ubuntu xenial stable
26          state: present
27
28      - name: Install docker and its dependencies
29        apt:
```

```
30     name: "{{ packages }}"
31     state: present
32     update_cache: yes
33     vars:
34     packages:
35     - docker-ce
36     - docker-ce-cli
37     - containerd.io
38     notify:
39     - docker status
40
41 - name: Add vagrant user to docker group
42     user:
43     name: vagrant
44     group: docker
45
46 - name: Disable swap
47     command: swapoff -a
48
49 - name: Add an apt signing key for Kubernetes
50     apt_key:
51     url: https://packages.cloud.google.com/apt/doc/apt-key.gpg
52     state: present
53
54 - name: Adding apt repository for Kubernetes
55     apt_repository:
56     repo: deb https://apt.kubernetes.io/ kubernetes-xenial main
57     state: present
58     filename: kubernetes.list
59
60 - name: Install Kubernetes binaries
61     apt:
62     name: "{{ packages }}"
63     state: present
64     update_cache: yes
```

```
65     vars:
66     packages:
67         - kubelet
68         - kubeadm
69         - kubectl
70
71     - name: Restart kubelet
72       service:
73         name: kubelet
74         daemon_reload: yes
75         state: restarted
76
77     - name: Copy the join command to server location
78       copy:
79         src: "join"
80         dest: /tmp/join-command.sh
81         mode: 0777
82         become: false
83
84     - name: Join the node to cluster
85       command: sh /tmp/join-command.sh
86
87     handlers:
88         - name: docker status
89           service: name=docker state=started
90
```

Inizializzazione

È necessario seguire il seguente comando per far sì che vagrant inizializzi le VM e le renda accessibili attraverso SSH:

```
vagrant up
```

Se il comando non riporta errori si è conclusa la fase di inizializzazione e le macchine sono accessibili dall'esterno.

Test

Per accertarsi che tutto il procedimento sia andato a buon fine e che effettivamente il cluster sia stato creato è sufficiente richiamare la macchina master

```
valiokei@valiokei-G5-5590:~/Scrivania/kubernetes$ vagrant ssh master
Welcome to Ubuntu 18.04.4 LTS (GNU/Linux 4.15.0-108-generic x86_64)

* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:       https://ubuntu.com/advantage

System information as of Sun Jun 28 14:15:26 UTC 2020

System load:  0.9               Users logged in:      0
Usage of /:   35.0% of 9.63GB   IP address for enp0s3: 10.0.2.15
Memory usage: 25%              IP address for enp0s8: 192.168.50.10
Swap usage:   0%               IP address for docker0: 172.17.0.1
Processes:    112

* "If you've been waiting for the perfect Kubernetes dev solution for
  macOS, the wait is over. Learn how to install Microk8s on macOS."

https://www.techrepublic.com/article/how-to-install-microk8s-on-macos/

7 packages can be updated.
0 updates are security updates.

Last login: Sun Jun 28 14:17:59 2020 from 10.0.2.2
```

e successivamente controllare il corretto funzionamento di kubernetes attraverso il comando:

```
vagrant@master:~$ kubectl get nodes
```

NAME	STATUS	ROLES	AGE	VERSION
master	Ready	master	6d21h	v1.18.4
node-1	Ready	<none>	6d21h	v1.18.4
node-2	Ready	<none>	6d21h	v1.18.4

Come si può vedere ogni nodo è up e partecipa al cluster

Descrizione dettagliata dei File

Struttura Vagrantfile

Il vagrantfile, scritto in ruby, esposto sopra ha il compito di istanziare le 3 virtual machine del cluster.

Le prime 2 righe del file settano il valore di 2 variabili condivise, "IMAGE_NAME" e N, che indicano rispettivamente il SO che condividono le 3 VM e il numero di nodi client da istanziare.

La riga 4 setta la versione del vagrantfile a 2, ciò non compromette comunque la retrocompatibilità con i vecchi formati.

La riga 5 indica che non sarà necessario inserire una chiave ssh per la connessione alle VM.

Successivamente, dalla riga 7 alla 10 si va a specificare il provider di virtualizzazione, la memoria e il numero di core da assegnare a ciascuna macchina.

Conclusa la sezione generale di settaggio si passa alle righe dedicate esplicitamente alla creazione delle macchine virtuali.

Dalla riga 12 alla 37 sono presenti 2 blocchi di codice distinti:

1. il primo definisce le specifiche del nodo master:
 - a. si sfrutta la variabile "IMAGE_NAME" per settare il SO
 - b. si assegna un'ip privato nella subnet 192.168.178.0
 - c. si setta l'hostname della macchina con il quale la si richiamerà attraverso SSH

- d. si specifica che il provision, il settaggio interno della macchina, verrà svolto attraverso ansible, in particolare sfruttando il file “master-playbook.yml”; viene anche settata la variabile di ansible che punta all’ip del nodo con l’indirizzo sopra specificato.

2. il secondo blocco ricalca il precedente:

- a. aggiunge un ciclo che scorre una variabile da 0 a N, questa variabile identifica quale macchina si sta istanziando.
- b. esegue tutte le operazione svolte sopra, ossia dalla a. alla d. andando però a parametrizzare il tutto sfruttando il valore dell’iteratore del ciclo.

Struttura Ansible Files

In questa sezione verrà analizzata la composizione dei 2 file di provisioning che ansible processa. Entrambi sono scritti sfruttando il linguaggio YAML e provvedono a installare tutto il software necessario all'interno della/delle macchine target.

Sono necessari due file in quanto le macchine virtuali, come già detto sono di due tipologie: una server e due client.

Di seguito un'analisi dei comandi sfruttati in entrambi i file.

I file iniziano con la chiave "host" settata con "all" che specifica che tutte le macchine a cui sarà applicato questo file eseguiranno i comandi sottostanti espressi nella sezione "task".

Come appena accennato il seguito del file è una lista di task, ossia comandi e moduli shell, che sono usualmente inviati attraverso una shell locale o remota in sequenza.

Si andrà ora a illustrare quali comandi sono stati eseguiti senza entrare nel dettaglio della scrittura degli stessi o della struttura del file stesso, per queste informazioni si possono confrontare le relative documentazioni nei [Riferimenti](#).

Comandi in comune tra la macchina master e i due client:

- Installazione dipendenze preliminari
 - Installazione dei pacchetti che rendono possibile sfruttare apt attraverso HTTPS (righe 5-16)
- Installazione di Docker
 - Aggiunta della chiave apt per il download di Docker (righe 18-21)
 - Aggiunta di una repository stabile per il download di Docker (righe 23-26)
 - Installazione di Docker e delle sue dipendenze (righe 28-39)
- Aggiunta dell'utente vagrant nel gruppo docker (righe 41-44)
- Disabilitazione del volume swap (righe 46-47)
- Installazione di Kubernetes
 - Aggiunta della chiave apt per il download di Kubernetes (righe 49-52)
 - Aggiunta di una repository stabile per il download di Kubernetes (righe 54-58)
 - Installazione di Kubernetes (righe 60-70)
 - Riavvio del servizio kubelet (righe 72-76)

Comandi caratteristici della macchina master:

- Settaggio di Kubernetes
 - Inizializzazione del cluster (righe 78-80)
 - Creazione della cartella .kube (righe 82-86)
 - Copia del file admin.conf nella cartella appena creata (righe 88-89)
 - Settaggio dei permessi del file admin.conf per permetterne l'accesso all'utente vagrant e all'omonimo gruppo (righe 91-95)
 - Aggiunta del plugin Calico per fornire capacità di rete L3 ai kubernetes pods (righe 97-99)
- Aggiunta del nodo master al cluster Kubernetes
 - Generazione del comando di Join che sarà poi sfruttato dai client per unirsi al cluster (righe 101-103)
 - Copia di tale comando all'interno di un file locale, chiamato "join-command.sh", posto nella stessa cartella dove sono presenti tutti i file .yaml (righe 105-110)
- Riavvio del servizio di docker (righe 112-114)

Comandi caratteristici delle macchine client:

- Join al dominio della macchina client
 - Copia del file "join-command.sh" all'interno della VM (righe 77-82)
 - Esecuzione del comando contenuto nel file appena copiato, ossia del comando atto ad effettuare il join della macchina su cui viene eseguito al cluster istanziato nella macchina master (righe 84-85)
- Riavvio del servizio di docker (righe 112-114)

Riferimenti

- [ToMyGitHubProject](#)
- [Ansible Documentation](#)
- [Ansible Playbooks](#)
- [Kubernetes Documentation](#)
- [Kubernetes Components](#)
- [Docker Documentation](#)
- [Vagrant Documentation](#)
- [Vagrantfile](#)