Sistemi Distribuiti

OpenStack Setup: best practices and procedures for optimal installation and configuration

Gloria Semprini
gloria.semprini@studio.unibo.it

January 2024

Abstract

Il progetto è incentrato sulla distribuzione e gestione di un'infrastruttura cloud privata utilizzando OpenStack. Ciò prevede l'installazione e la configurazione di vari servizi OpenStack tra cui Keystone, Nova, Neutron, Glance ecc., all'interno di un cluster di server. L'architettura è stata progettata per fornire risorse di elaborazione, archiviazione e rete virtualizzate per supportare le esigenze di un'organizzazione.

I temi che copre questo report sono la progettazione e implementazione di OpenStack, inclusa l'architettura, i requisiti hardware e software e una guida finale sull'interazione con i suoi servizi: Dashboard Horizon e OpenStack Client.

Il progetto ha presentato diverse sfide, tra cui problemi di compatibilità software, che hanno richiesto un'attenta risoluzione dei problemi e configurazione adeguata. Nonostante ciò, è stato implementato con successo e ha fornito servizi cloud affidabili, scalabili e flessibili.

Il progetto ha dimostrato la fattibilità e il valore dell'utilizzo di OpenStack come piattaforma per l'implementazione di cloud privati.

Indice

1 Introduzione **1**

2 Tecnologie utilizzate **3**

2.1 Python 3

2.2 Ansible 3

2.3 Kolla 4

2.4 OpenStack 5

2.4.1 Architettura 5

2.4.2 Tools per la gestione del Cloud 8

3 Creazione del Cluster **10**

3.1 Requisiti minimi 10

3.2 Python 3 e PIP 11

3.3 Creazione e configurazione delle chiavi pubbliche e private 11

3.4 Ansible e Kolla-Ansible 11

3.4.1 Installazione Ansible 12

3.4.2 Configurazione della sicurezza all'interno di ansible 12

3.4.3 Installazione e configurazione di Kolla-Ansible 12

3.4.4 Configurazione della sicurezza all'interno di kolla-ansible . . 13

3.4.5 File di configurazione di Ansible 18

3.4.6 Creazione del Virtual IP 19

3.4.7 File di configurazione di Kolla 19

3.5 Configurazione IP Openstack 23

3.6 Installazione della Dashboard e PlayBook 25

4 User experience **32**
4.1 Horizon 32
4.1.1 Panoramica della Dashboard 32
4.1.2 Creazione di una nuova istanza 33
4.1.3 Attaccare il volume ad un'istanza 33
4.1.4 Gestione delle reti 34
4.1.5 Creare e gestire volumi di Storage 34
4.1.6 Creare e gestire immagini 35
4.1.7 Orchestrazione (Heat) 35
4.1.8 Configurazione Avanzata 35
4.2 CLI 36
4.2.1 Panoramica della Dashboard 36
4.2.2 Creazione di una nuova istanza 36
4.2.3 Creare e gestire volumi di Storage 37
4.2.4 Attaccare il volume ad un'istanza 37
4.2.5 Gestione delle reti 38
4.2.6 Creare e gestire immagini 39
4.2.7 Orchestrazione (Heat) 39
4.2.8 Configurazione Avanzata 40

Conclusioni **41**

Bibliografia **42**

Elenco delle figure

2.1 Architettura e configurazione dei nodi di OpenStack 6
2.2 Architettura dei servizi di OpenStack 8

3.1 Schermata di login di OpenStack Horizon 31

4.1 Panoramica della Dashboard Horizon 33

Capitolo 1

Introduzione

Openstack è tra le più innovative ed avanzate piattaforme open-source per cloud-computing ed è orientata alla gestione di configurazioni di cloud pubblico e privato.

OpenStack offre un'ampia gamma di funzionalità e servizi che possono essere personalizzati per soddisfare le proprie esigenze specifiche. Consente, a chi ne necessita, di implementare e gestire risorse, macchine virtuali, storage e infrastrutture di rete in modo scalabile e flessibile.

Uno scenario tipico per OpenStack coinvolge un'organizzazione che desidera creare la propria infrastruttura di cloud privato per ospitare e gestire le proprie applicazioni e servizi. Questa organizzazione può essere una grande impresa, un istituto di ricerca o un'agenzia governativa che deve gestire il proprio data center e le proprie risorse. Viene utilizzato OpenStack per fornire servizi come macchine virtuali, storage di oggetti, storage a blocchi e networking ai propri utenti o clienti, oppure per gestire e orchestrare applicazioni e servizi in esecuzione nel cloud.

Per poter trarne benefici, l'organizzazione deve installarlo e configurare i vari componenti OpenStack, che includono Nova per il calcolo, Cinder per l'archiviazione a blocchi, Glance per l'archiviazione delle immagini e Neutron per il networking, Keystone per la gestione delle identità, Swift per l'object storage e Horizon per

la gestione basata sul web. Queste configurazioni sono fondamentali per poter usufruire appieno della piattaforma.

Per poter rendere il servizio ottimale è possibile integrare OpenStack con altri sistemi e servizi, come un DNS o uno o più loadbalancer per gestire il carico di lavoro.

Una volta che l'infrastruttura OpenStack è attiva e funzionante, l'organizzazione può creare e gestire macchine virtuali, storage e risorse di rete tramite la dashboard OpenStack, tramite le API, o attraverso la shell del sistema operativo interagendo con un Client dedicato. E' possibile anche interagire con la piattaforma utilizzando proprie applicazioni, API o altri strumenti appositi come Heat o Murano.

Capitolo 2

Tecnologie utilizzate

Sono state impiegate una serie di tecnologie per garantire un'efficace automazione, gestione e implementazione delle risorse. Ci sono servizi e componenti che dovranno essere installati e configurati con adeguati script o file di configurazione. Le sezioni seguenti forniranno una spiegazione dettagliata di ciascuna tecnologia e del loro ruolo all'interno del progetto.

2.1 Python

Interpreter

Python Interpreter è un software che comprende la sintassi e la struttura del codice Python ed è in grado di eseguire il codice direttamente senza compilazione e in tempo reale. Legge il codice riga per riga e lo converte nelle istruzioni richieste dallo script che gli viene dato per eseguire le operazioni previste.
In questo caso, esegue gli script che gli vengono forniti dai pacchetti di installazione, configurazione ed esecuzione dei tool Ansible e Kolla.

2.2 Ansible

Ansible è uno strumento di automazione open source che semplifica la gestione e l'implementazione di applicazioni, infrastrutture e sistemi IT. Utilizza un linguag-

gio semplice e flessibile chiamato YAML per descrivere e configurare i componenti del sistema, consentendo agli amministratori di automatizzare le attività ripetitive e semplificare i flussi di lavoro complessi.
Ansible adotta un'architettura senza agenti, detta "Agentless", connettendosi agli host da remoto tramite SSH. Questa soluzione evita l'installazione di software agenti su ogni nodo, semplificando la configurazione e riducendo il carico operativo complessivo. Aumenta, così, la scalabilità e agevola la manutenzione anche in ambienti distribuiti di grandi dimensioni.
Il suo design modulare facilita anche l'uso di moduli, plug-in e playbook personalizzati, consentendo agli amministratori di automatizzare qualsiasi processo di cui hanno bisogno. Ansible aiuta, infatti, i team a ridurre gli errori, aumentare la produttività e creare ambienti IT più affidabili e sicuri.

2.3 Kolla

Kolla è un software open-source che fornisce strumenti e immagini di container per la distribuzione e la gestione dei servizi OpenStack. Mira a semplificare l'implementazione di OpenStack fornendo immagini di container per ciascun servizio della piattaforma in ambienti containerizzati.
I container utilizzati sono creati da Docker e servono per impacchettare ogni servizio OpenStack e le sue dipendenze in un'immagine autonoma.
Kolla fornisce anche strumenti per la configurazione e la personalizzazione della distribuzione di OpenStack, tra cui un'interfaccia a riga di comando (OpenStack Client) e una dashboard basata sul web (Horizon).

Kolla-Ansible

Kolla-ansible è un progetto Kolla, che fornisce immagini per container pronti all'uso per i servizi OpenStack. Per poterlo sfruttare, abbiamo bisogno di Ansible, poichè Kolla-Ansible dipende da Ansible, che automatizza l'implementazione e la configurazione di questi servizi.
Per ottimizzare i processi al meglio, offre quindi degli strumenti per semplificare

e automatizzare il processo di distribuzione dei servizi OpenStack containerizzati su un cluster di nodi. Sfrutta i componenti di OpenStack (Nova, Neutron, Cinder ecc.) per migliorare la manutenibilità e la scalabilità dell'intera infrastruttura.

2.4 OpenStack

OpenStack è una piattaforma di cloud computing open source progettata per fornire Infrastructure-as-a-Service (IaaS).
OpenStack viene rilasciato regolarmente con nuove versioni, ognuna con miglioramenti, nuove funzionalità e talvolta rimozioni di componenti obsoleti. Le differenze tra le versioni sono graduali, con la maggior parte dei componenti principali che rimangono stabili e vengono migliorati nel tempo.
A tal proposito si è scelto di utilizzare la versione "Xena" perchè, rispetto alle altre, offre funzionalità importanti per il networking, una migliore sicurezza e una maggiore stabilità.

2.4.1 Architettura

L'architettura è modulare e distribuita ed è composta da vari componenti:

- **Nova (Compute)**: servizio che alloca e gestisce le risorse di calcolo. É responsabile del provisioning e della gestione delle macchine virtuali.
- **Neutron (Networking)**: gestisce le reti tra le interfacce dei dispositivi controllati da OpenStack. Fornisce funzionalità di rete avanzate come reti virtuali, bilanciamento del carico e firewall, garantendo l'isolamento e la sicurezza del traffico della rete.
- **Cinder (Block Storage)**: servizio di storage a blocchi persistenti. Lo storage persistente è essenziale per conservare dati a lungo termine e per garantire che le applicazioni possano accedere ai dati anche dopo il riavvio o il termine delle istanze. Permette il provisioning, la gestione e l'allocazione dinamica dei volumi di storage alle macchine virtuali.
- **Swift (Object Storage)**: responsabile dell'archiviazione di una grande quantità di dati non strutturati (es. file di backup, immagini, video e dati statici

di applicazioni web). Fornisce uno storage di oggetti scalabile e altamente disponibile.

- **Keystone (Identity Service)**: servizio per la sicurezza e la gestione delle identità. Gestisce l'autenticazione e l'autorizzazione degli utenti, fornendo un catalogo di servizi per il controllo degli accessi e la gestione delle identità.
- **Glance (Image Service)**: componente per la gestione delle immagini delle macchine virtuali (creazione, archiviazione, recupero delle immagini di dischi e server).

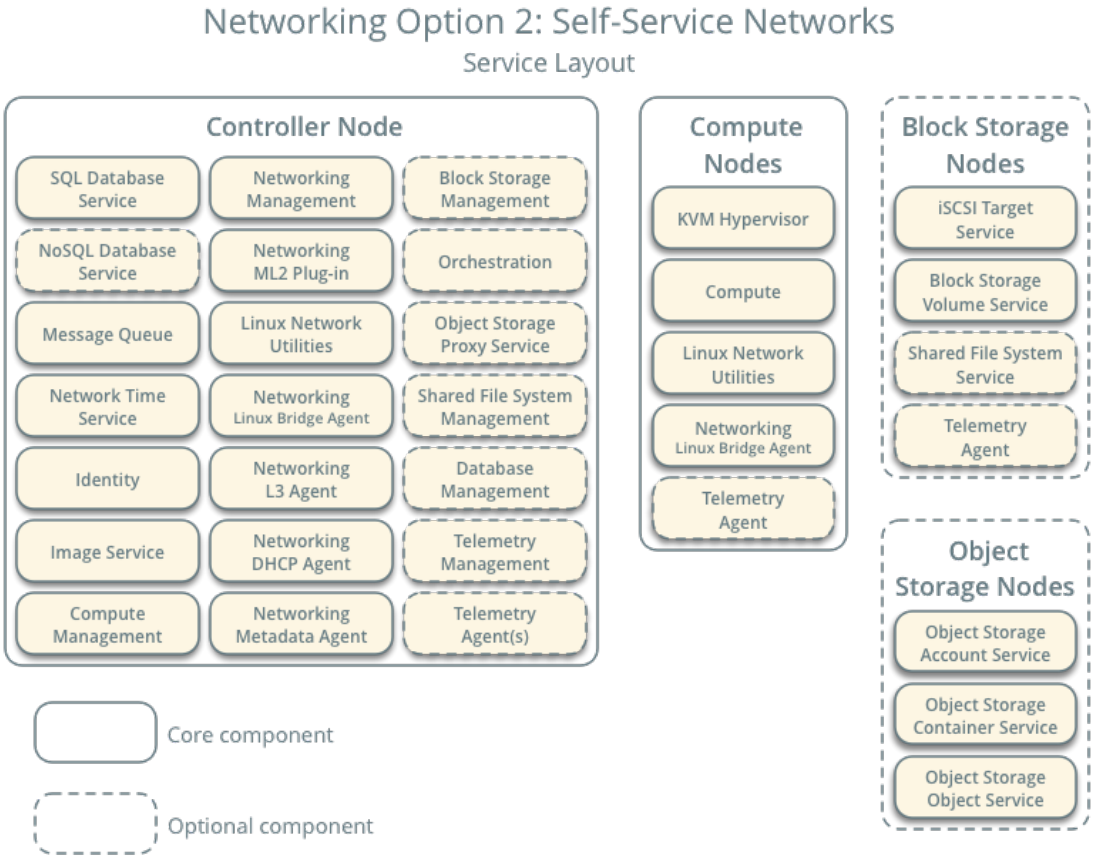

Figura 2.1: Architettura e configurazione dei nodi di OpenStack

Mentre queste componenti sono le fondamenta di OpenStack, dobbiamo considerare anche le unità che supportano e migliorano l'operatività della piattaforma.

Questi moduli aggiuntivi giocano un ruolo cruciale nel garantire la scalabilità, la sicurezza e l'affidabilità di OpenStack. Alcuni di questi elementi includono:

- **MariaDB**: un database relazionale di OpenStack utilizzato per archiviare la configurazione e i dati operativi dei suoi servizi. La maggior parte dei servizi core di OpenStack (come Keystone, Nova, Neutron, Cinder) richiedono un database per memorizzare informazioni critiche di configurazione e stato.
- **RabbitMQ**: un message broker utilizzato per la comunicazione asincrona tra i vari servizi OpenStack. Garantisce che i messaggi vengano consegnati tra i componenti distribuiti di OpenStack, facilitando il coordinamento delle operazioni e il decoupling dei servizi.
- **Memcached**: un sistema distribuito di caching in memoria, utilizzato per migliorare le performance delle applicazioni OpenStack. Riduce il carico sui database e velocizza l'accesso ai dati temporanei spesso richiesti dai servizi di OpenStack.
- **HAProxy**: un load balancer utilizzato per distribuire il traffico di rete tra diverse istanze dei servizi OpenStack. Garantisce alta disponibilità e bilanciamento del carico, migliorando la scalabilità e la resilienza dei servizi.
- **Cron**: è un servizio per la pianificazione di job ricorrenti. Viene utilizzato per eseguire script e attività di manutenzione automatizzate nei sistemi OpenStack.
- **Fluentd**: un aggregatore di log open source. Raccoglie, aggrega e trasmette i log dei vari servizi di OpenStack, aiutando nella gestione e nel monitoraggio delle attività del sistema.
- **Heat**: servizio di orchestrazione di OpenStack. Permette di definire e gestire l'infrastruttura cloud usando modelli dichiarativi, facilitando il provisioning e la gestione automatizzata delle risorse.
- **Open vSwitch (OVS)**: uno switch virtuale utilizzato per la gestione della rete in ambienti virtualizzati. Fornisce funzionalità avanzate di rete necessarie per Neutron, gestendo il traffico di rete tra le macchine virtuali e i servizi di rete virtuale.
- **Placement**: servizio a supporto di Nova, traccia e alloca risorse. Gestisce

l'inventario delle risorse (CPU, memoria, disco) e assicura che siano allocate efficientemente ai servizi di computing.

- **Keepalived**: fornisce servizi di alta disponibilità utilizzando VRRP (Virtual Router Redundancy Protocol). Assicura la continuità operativa dei servizi critici di OpenStack, riducendo i tempi di inattività in caso di guasti.

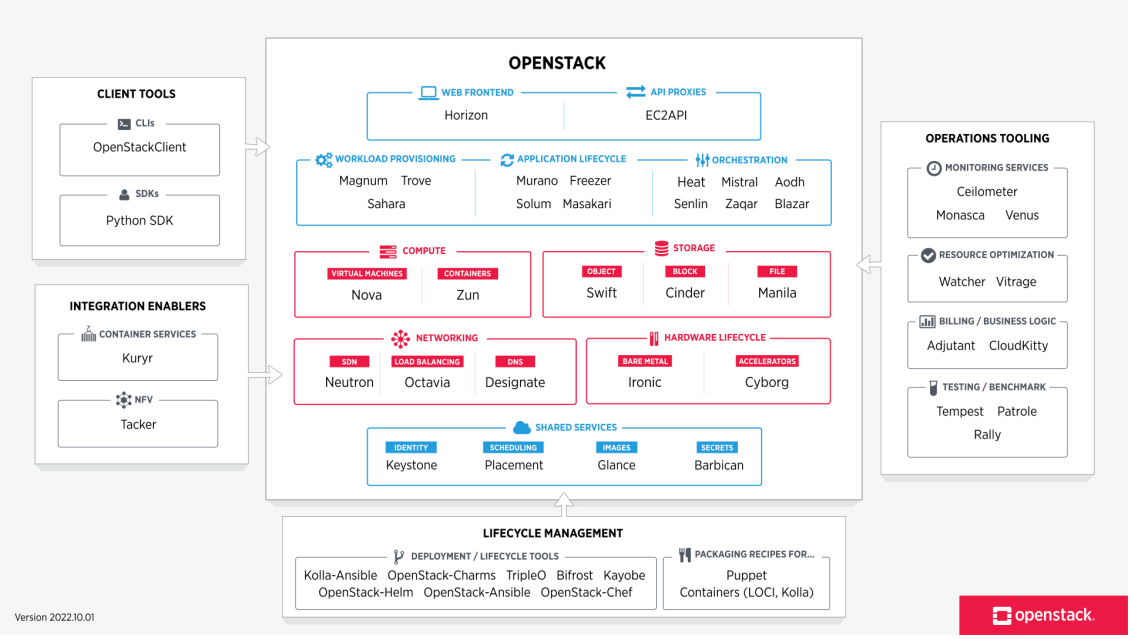

Figura 2.2: Architettura dei servizi di OpenStack

2.4.2 Tools per la gestione del Cloud

Tra i vari vantaggi che emergono sulla piattaforma, Openstack offre la possibilità di gestire le risorse attraverso una Dashboard (interfaccia grafica), chiamata Horizon, oppure il client di OpenStack a riga di comando utilizzando OpenStack CLI.

Horizon Dashboard vs OpenStack CLI

Openstack Horizon è una GUI web-based usata per interagire con la piattaforma in modo intuitivo ed immediato, senza doverlo obbligatoriamente usare attraverso

una shell, semplificando così la gestione e il provisioning delle risorse.

A differenza della dashboard, l'utilizzo di OpenStack CLI è fondamentale per gli utenti più esperti che possono accedere ai comandi più rapidamente o automatizzare operazioni ripetitive tramite script.

Capitolo 3

Creazione del Cluster

In questo capitolo verrà illustrato come impostare l'ambiente virtuale, installare e configurare OpenStack. Per automatizzare la piattaforma verrà utilizzato Ansible attraverso un playbook specifico chiamato Kolla-Ansible.

3.1 Requisiti minimi

Per iniziare, è necessario determinare quali e quante risorse sono disponibili. Questa guida prevede la configurazione di due nodi: Controller e Compute. Di seguito sono elencati i requisiti preliminari per l'installazione:

- CPU: 4 core @ 2.4 GHz
- RAM: 8 GB
- HDD 2x500 GB
- Software: Ubuntu 22.04+

Il sistema operativo a servizio della piattaforma è Ubuntu 22.04 poichè è compatibile con Openstack. Altre versioni di Ubuntu o Linux non sono completamente adatti ad eseguirlo al loro interno.

3.2 Python 3 e PIP

Per installare e utilizzare OpenStack, sono necessari Python e PIP. Di seguito i comandi per installare e aggiornare i relativi pacchetti:

```
#aggiornamento del pacchetto
sudo apt update

#installazione delle dipendenze delle build di Python
sudo apt install -y python3 python3-dev pip libffi-dev gcc
↪  libssl-dev
```

3.3 Creazione e configurazione delle chiavi pubbliche e private

Per usufruire al meglio e in sicurezza della piattaforma, è importante configurare una chiave pubblica e una privata all'interno della macchina virtuale. Per farlo, è utile il seguente comando:

```
ssh-keygen
```

Durante l'esecuzione del comando, il terminale chiederà una serie di caratteri che andranno a determinare la coppia di chiavi privata-pubblica in un determinato percorso `'path/to/id_rsa'` e `'path/to/id_rsa.pub'`. Nel corso della creazione delle chiavi pubblica e privata, verrà richiesta anche una *passphrase* per un ulteriore livello di sicurezza che verrà inserita ogni volta che verrà utilizzata la chiave privata.

3.4 Ansible e Kolla-Ansible

All'interno di questa sezione verranno descritte l'installazione e configurazione di Ansible e Kolla-Ansible.

3.4.1 Installazione Ansible

L'installazione di Ansible è strettamente collegata a Kolla-Ansible e richiede una versione inferiore alla 4.11 per essere compatibile.

```
pip3 install 'ansible<4.11' 'ansible-core<2.12'
```

3.4.2 Configurazione della sicurezza all'interno di ansible

Durante la configurazione di Ansible, è importante garantire la sicurezza contro potenziali attacchi informatici. Ansible offre uno strumento per crittografare le chiavi generate nella sezione precedente.

3.4.3 Installazione e configurazione di Kolla-Ansible

Come spiegato nel capitolo precedente, la scelta della distribuzione dipende dalle esigenze e dai requisiti. La versione `Xena`, in questo caso, viene direttamente scaricata e installata attraveso Python.
Il branch che viene utilizzato è `'stable/xena'`.

```
#Installazione Kolla-Ansible su branch stable/xena
pip3 install
↪ git+https://github.com/openstack/kolla-ansible@stable/xena
```

Durante la configurazione di Kolla-Ansible, è necessario:

- creare una cartella `kolla` nella directory principale `'/etc/'`
- assegnare l'utente corrente come proprietario e il gruppo proprietario della directory `'/etc/kolla'`
- copiare i file di configurazione `globals.yml` e `passwords.yml` all'interno della cartella
- copiare i file di inventario nella directory corrente.

Quest'ultimo punto serve perchè Kolla-Ansible contiene i file di inventario (all-in-one e multinode) nella directory `'ansible/inventory'`.

```
#Creazione della cartella
mkdir -p /etc/kolla

#Assegnazione utente corrente come proprietario e gruppo
↪ proprietario della directory
sudo chown $USER:$USER /etc/kolla

#Copia dei file nella cartella di Kolla-Ansible
cp -r /usr/local/share/kolla-ansible/etc_examples/kolla/*
↪ /etc/kolla

#Copiare i file di inventario nella directory corrente
cp /usr/local/share/kolla-ansible/ansible/inventory/* .
```

3.4.4 Configurazione della sicurezza all'interno di kolla-ansible

Uno step importante da eseguire è la generazione e la sicurezza delle password di Kolla-Ansible. Tale comando è il seguente:

```
kolla-genpwd
```

Questo consente di utilizzare la piattaforma e i dati in modo sicuro.
É possibile personalizzare la password tramite le opzioni del comando, a seconda delle preferenze per una password facile da memorizzare, che può includere simboli, numeri, ecc.
I flag sono i seguenti:

- **`-c, --complexity`**: Specifica il livello di complessità della password.
  - low: password di facile memorizzazione
  - medium: password di media complessità
  - high: password di alta complessità

- **-s, --symbols**: Specifica se includere simboli nella password.
  - yes: includere simboli
  - no: non includere simboli
- **-l, --lowercase**: Specifica se includere caratteri minuscoli nella password.
  - yes: includere caratteri minuscoli
  - no: non includere caratteri minuscoli
- **-u, --uppercase**: Specifica se includere caratteri maiuscoli nella password.
  - yes: includere caratteri maiuscoli
  - no: non includere caratteri maiuscoli
- **-n, --numbers**: Specifica se includere numeri nella password.
  - yes: includere numeri
  - no: non includere numeri

Una volta definita la password, sarà utile avviare il comando `ssh-agent` in modo che le chiavi private possano essere memorizzate in sicurezza e fornire un'interfaccia per le applicazioni client SSH in modo da non dover richiedere ogni volta la passphrase.

Il comando è composto da:

- **eval**: valuta o esegue dinamicamente il contenuto della stringa come se fosse una serie di comandi validi nella shell.
- **ssh-agent -s**: avvia il comando ssh-agent e imposta le variabili d'ambiente necessarie.

... ed è il seguente:

```
echo 'eval $(ssh-agent -s)' >> .bashrc
```

Dopo aver avviato `ssh-agent`, è possibile aggiungere le chiavi private utilizzando il comando `ssh-add`. Tuttavia, all'interno di Ansible, è necessario eseguire `ssh-add` tramite un comando specifico per garantire che le chiavi siano accessibili durante l'esecuzione dei playbook.

Il comando si suddivide in:

- **`ansible localhost`**: specifica l'hostname o il gruppo degli host su cui eseguire il comando Ansible. In questo caso indica che il comando ‘`localhost`‘ verrà eseguito sul computer in cui Ansible è in esecuzione, piuttosto che su un host remoto.
- **`-m command`**: specifica il modulo Ansible da utilizzare. In questo caso, viene utilizzato il modulo `command`, che consente di eseguire comandi arbitrari sulla macchina locale.
- **`-a`**: specifica gli argomenti da passare al modulo command.
- **`"sshpass -P passphrase -p {{ passphrase }} ssh-add .ssh/id_rsa"`**: questo è il comando effettivo passato come argomento al modulo.
  - **`sshpass`**: è un'utility che permette di fornire automaticamente una passphrase per l'autenticazione SSH.
  - **`-P passphrase`**: indica che `sshpass` deve cercare una passphrase da inserire.
  - **`-p {{ passphrase }}`**: specifica la passphrase effettiva da usare, che poi verrà recuperata dalla variabile Ansible `passphrase`.
  - **`ssh-add .ssh/id_rsa`**: il comando aggiunge la chiave privata specificata (`.ssh/id_rsa`) all'agent SSH, rendendo la chiave disponibile per le connessioni successive.
- **`-e "@secrets.yml"`**: specifica le variabili extra che devono essere caricate. In questo caso, si sta utilizzando il file `secrets.yml` come fonte di variabili extra. Quindi, il valore di ‘`{{ passphrase }}`‘ viene recuperato dal file fornito in input tramite l'opzione `-e` nel comando..

Riepilogo del comando:

```
echo 'ansible localhost -m command -a "sshpass -P
↪  passphrase -p {{ passphrase }} ssh-add .ssh/id_rsa" -e
↪  "@secrets.yml"' >> .bashrc
```

Per poter archiviare e riutilizzare i dati durante l'automazione delle attività di gestione di Ansible, è consigliabile criptarle utilizzando **Ansible-Vault**, un framework di Ansible che gestisce la sicurezza e la riservatezza delle informazioni sensibili.
I comandi utili a ovviare a questa situazione sono `encrypt` e `encrypt_string` di ansible-vault.

- **‘ansible-vault encrypt‘**: un'opzione per crittografare un file dato in input.
  - **‘--vault-pass-file <path/to/password_file> <path/to/key_file>‘**: è un flag che specifica il file della password necessaria per accedere e gestire la cifratura (<path/to/password_file>), mentre il secondo argomento è il percorso del file che si desidera cifrare (<path/to/key_file>).
- **‘ansible-vault encrypt_string‘**: un'opzione che consente di crittografare una stringa data in input.
  - **‘-n [label]‘ <variable_name>**: il flag assegna un'etichetta alla stringa cifrata per identificarla. In questo caso un'etichetta può essere "user", "password" o "passphrase", mentre ‘<variable_name>‘ è una variabile contenente una stringa.

...e questi sono i comandi completi:

- **‘ansible-vault encrypt_string‘**:

```
ansible-vault encrypt_string -n passphrase
↪ $PASSPHRASE --vault-pass-file $PASSWORD_FILE >>
↪ $SECRETS_FILE
```

- **‘ansible-vault encrypt‘**:

```
ansible-vault encrypt --vault-pass-file $PASSWORD_FILE
↪ $PRIVATE_KEY_FILE
```

Ovviamente per rendere il procedimento automatizzato e più accessibile, è consigliabile inserire tutto all'interno di file.
Di seguito viene illustrato lo script completo per il settaggio e la cifratura di username, password e passphrase:

```
# Reading data from shell input
read -sp "Insert password for encryption: " PASSWORD; echo
read -p "Insert username [root]: " USER_NAME
USER_NAME=${USER_NAME:-root}
read -sp "Insert user password: " USER_PASSWORD; echo
read -p "Insert private key path [$HOME/.ssh/id_rsa]: "
↪ PRIVATE_KEY
PRIVATE_KEY=${PRIVATE_KEY:-$HOME/.ssh/id_rsa}
PUBLIC_KEY=$PRIVATE_KEY.pub
read -sp "Insert passphrase: " PASSPHRASE; echo

# Setting file paths
PASSWORD_FILE=home/.password/value
SECRETS_FILE=home/secrets.yml
PRIVATE_KEY_FILE=home/.ssh/id_rsa
PUBLIC_KEY_FILE=home/.ssh/id_rsa.pub

# Encrypting and setting up the files
echo $PASSWORD > $PASSWORD_FILE
ansible-vault encrypt_string -n user $USER_NAME
↪ --vault-pass-file $PASSWORD_FILE > $SECRETS_FILE
ansible-vault encrypt_string -n password $USER_PASSWORD
↪ --vault-pass-file $PASSWORD_FILE >> $SECRETS_FILE
ansible-vault encrypt_string -n passphrase $PASSPHRASE
↪ --vault-pass-file $PASSWORD_FILE >> $SECRETS_FILE
```

```
cp $PRIVATE_KEY $PRIVATE_KEY_FILE
cp $PUBLIC_KEY $PUBLIC_KEY_FILE
ansible-vault encrypt --vault-pass-file $PASSWORD_FILE
↪ $PRIVATE_KEY_FILE
ansible-vault encrypt --vault-pass-file $PASSWORD_FILE
↪ $PUBLIC_KEY_FILE
```

3.4.5 File di configurazione di Ansible

Si configura l'environment di ansible modificando `‘ansible inventory file‘ (‘*.cfg‘)`, ovvero il file di configurazione utilizzato da Ansible per definire gli host su cui eseguire le operazioni. Le opzioni che devono essere modificate sono le seguenti:

- **`‘host_key_checking‘`**: questa impostazione abilita/disabilita la verifica delle chiavi degli host SSH. Quando Ansible si connette a un host via SSH per eseguire le operazioni, verifica la chiave SSH dell'host di destinazione. Disabilitando questa opzione, si evita l'interazione dell'utente per la verifica delle chiavi e si accettano le connessioni SSH senza domande.
- **`‘pipelining‘`**: è una tecnica che consente di eseguire più passaggi senza dover copiare i file su disco, riducendo così la latenza nelle operazioni di esecuzione di Ansible.
- **`‘forks‘`**: questa opzione imposta il numero massimo di host su cui Ansible può eseguire operazioni in parallelo. Inserendogli un valore N, andiamo a comunicare ad Ansible di poter eseguire operazioni su N host. Questo valore è regolabile in base alle risorse del proprio sistema e dai requisiti delle operazioni che vengono svolte all'interno.
- **`‘vault_password_file‘`**: specifica il percorso del file contenente la password necessaria per decifrare i dati cifrati con Ansible Vault.
- **`‘private_key_file‘`**: specifica il percorso della chiave privata SSH da utilizzare quando Ansible si connette agli host target tramite SSH. La chiave

privata viene utilizzata per l'autenticazione e per stabilire una connessione sicura agli host.

```
[defaults]
host_key_checking=False
pipelining=True
forks=100
vault_password_file = /root/.password/value
private_key_file = /root/.ssh/id_rsa
```

3.4.6 Creazione del Virtual IP

Prima di poter lanciare la piattaforma e lavorare con tutte le funzionalità, è importante creare un Virtual IP (VIP) per indirizzare il traffico di rete verso un gruppo di nodi che eseguono determinati servizi. Questo passaggio è importante perchè migliora l'affidabilità e la disponibilità del servizio dando Keepalived e High Availability al servizio.

- **High Availability (Affidabilità)**: assicura che i servizi siano sempre disponibili e accessibili anche in presenza di guasti hardware, errori del sistema ecc. per questo parliamo di affidabilità.
- **Keepalived (Disponibilità)**: consente di assegnare un VIP a uno dei nodi del cluster alla volta. In caso di guasto di un nodo, il VIP può essere trasferito in modo trasparente al nodo successivo, garantendo che il servizio rimanga disponibile.

Per creare un Virtual IP viene utilizzato un comando nella shell di Linux:

```
sudo ip addr add 192.168.11.104/24 dev eth0
```

3.4.7 File di configurazione di Kolla

Ora è possibile inserire il Virtual IP all'interno del file `globals.yml` con tutte le impostazioni adeguate ad avviare correttamente il servizio.

1. **VIP Interface/DNS:**
   - **‘kolla_internal_vip_interface‘**: VIP che vada ad indirizzare il traffico di rete fornendo Keepalived e High Availability all'interno dell'infrastruttura. Può quindi essere associato a servizi interni alla piattaforma (Keystone, Nova, Neutron ecc.).
   - **‘kolla_external_vip_interface‘**: VIP utilizzato per il traffico dall'esterno dell'infrastruttura (es. IP pubblico). Un punto di ingresso per i servizi esterni all'infrastruttura. (es. API, dashboard o altri servizi che devono essere raggiunti dagli utenti o applicazioni esterne). Come per l'internal VIP, anche l'external VIP fornisce Keepalived garantendo disponibilità continua dei servizi per gli utenti esterni.
   - **‘kolla_internal_fqdn‘**: è il nome del DNS che mappa il Virtual IP interno. FQDN (Fully Qualified Domain Name) è una rappresentazione completa del DNS (domini, sottodomini ecc.) che identifica univocamente un nodo alla rete. In questo caso il valore è direttamente quello dell'VIP.
   - **‘kolla_external_fqdn‘**: è il nome del DNS che mappa il Virtual IP esterno.

   N.B. In questo caso, internal/external VIP/FQDN coincidono di default, ma sarebbe più opportuno invece dividerli poichè uno gestisce il carico di lavoro e i servizi interni, mentre l'altro quelli esterni.

2. **Network Options (Neutron):**
   - **‘network_interface‘**: opzione che specifica l'interfaccia di rete utilizzata da Kolla-Ansible per la comunicazione interna tra i nodi. (es. eth0, ens160 ecc.)
   - **‘kolla_external_vip_interface‘**: è l'interfaccia di rete utilizzata da Neutron per associare il VIP esterno di `kolla_external_vip_address`, per consentire l'accesso interno ai servizi OpenStack.
   - **‘neutron_external_interface‘**: interfaccia di rete che sarà utilizzata da Neutron per ascoltare e instradare il traffico di rete esterno di

OpenStack per i carichi di lavoro che richiedono connettività esterna.

3. **OpenStack Options:**
   - **‘kolla_base_distro‘**: è l’opzione che specifica la distribuzione Linux utilizzata per Kolla-Ansible, che può essere ”centos”, ”ubuntu”, ”debian” ecc.
   - **‘kolla_base_distro_version_default_map‘**: è la mappatura delle versioni predefinite dei sistemi operativi di base supportati da Kolla-Ansible. (es. ”ubuntu”: ”jammy”)
   - **‘openstack_release‘**: è la versione di OpenStack che verrà installata per Kolla-Ansible. (es. Xena, Rocky, Queens, Diablo, Austin ecc.)
   - **‘enable_haproxy‘**: controlla se il servizio HAProxy deve essere abilitato o disabilitato nell’environment OpenStack gestito da Kolla-Ansible.
   - **‘enable_mariadb‘**: controlla se il servizio MariaDB deve essere abilitato o disabilitato nell’ambiente OpenStack gestito da Kolla-Ansible.

```
###############
# Kolla options
###############

# Distribuzione di Linux
kolla_base_distro: "ubuntu"

# Versione di Linux
kolla_base_distro_version_default_map: {
  "ubuntu": "jammy",
}

# Versione di OpenStack
openstack_release: "xena"
```

```
# Configurazione del nodo corrente interno
kolla_internal_vip_address: "192.168.11.104"
kolla_internal_fqdn: "{{ kolla_internal_vip_address }}"

# Configurazione del nodo corrente esterno
kolla_external_vip_address: "{{ kolla_internal_vip_address
↪  }}"
kolla_external_fqdn: "{{ kolla_external_vip_address }}"
```

```
################################
# Neutron - Networking Options
################################

#interfaccia di rete interna dei nodi di OpenStack
network_interface: "ens160"

#interfaccia di rete VIP esterno
kolla_external_vip_interface: "{{ network_interface }}"

#interfaccia di rete esterna dei nodi di OpenStack
neutron_external_interface: "lo"
```

```
###################
# OpenStack options
###################

#Abilitazione/Disabilitazione HA Proxy
enable_haproxy: "yes"

#Abilitazione/Disabilitazione MariaDB
enable_mariadb: "yes"
```

3.5 Configurazione IP Openstack

Per testare che tutto sia configurato al meglio, abbiamo bisogno di avviare un file (`multinode.ini`) che vada a comunicare con le macchine sulle quali gira OpenStack. Il file necessita di:

- **‘[compute] - [control]‘**: sono i diversi ruoli dei nodi dell'infrastruttura. `[control]` rappresenta i nodi di controllo, mentre `[compute]` quelli di calcolo. Ogni nodo ha un nome associato e in questo caso vengono dati convenzionalmente i nomi ‘`opcompute1`‘ e ‘`opcontrol1`‘.
- **‘ansible_host‘**: è una variabile che assegna ad IP la sua relativa configurazione associata. Abbiamo quindi `192.168.11.102` per il nodo di controllo, mentre `192.168.11.103` è quello di calcolo.
- **‘ansible_become‘**: può avere valori "`true/false`". Se è settato a "`true`" permette ad Ansible di eseguire comandi con privilegi di amministratore. E' l'equivalente del comando "`sudo`".
- **‘ansble_user‘**: è il nome utente che Ansible utilizzerà per connettersi ai nodi. Viene utilizzata la stessa variabile impostata durante la creazione del file contenenti le variabili per l'accesso ai servizi di Ansible.
- **‘ansible_password‘**: è il parametro che memorizza la password necessaria per l'accesso al nodo ed in questo caso come password è fissata la passphrase.

- **‘ansible_become_pass‘**: per poter ottenere i privilegi di amministratore e poter processare alcune azioni, viene utilizzata questa variabile. Differente da ‘ansible_password‘ perchè serve per l’accesso remoto al nodo.

```
[control]
# These hostname must be resolvable from your deployment host
opcontrol1 ansible_host=192.168.11.102 ansible_become=true
↪  ansible_user='{{ user }}' ansible_password='{{ passphrase
↪  }}' ansible_become_pass='{{ password }}'

[compute]
opcompute1 ansible_host=192.168.11.103 ansible_become=true
↪  ansible_user='{{ user }}' ansible_password='{{ passphrase
↪  }}' ansible_become_pass='{{ password }}'
# The network nodes are where your l3-agent and loadbalancers
↪  will run
# This can be the same as a host in the control group
```

Serve ora un comando per eseguire un test di raggiungibilità (ping) su tutti gli host definiti nel file di inventario `test.ini` Per fare ciò abbiamo bisogno anche delle variabili definite nel file `secrets.yml`. Il comando è composto da:

- **`ansible`**: comando per eseguire ansible
- **`-i "test.ini" all`**: specifica il file di inventario (inventory file) da utilizzare e che deve essere eseguito su tutti gli host elencati all’interno.
- **`-m ping`**: il modulo ping viene utilizzato per testare la raggiungibilità degli host che verifica se Ansible può connettersi a ciascun host remoto usando SSH e restituisce pong se la connessione è riuscita.
- **`-e "@secrets.yml"`**: permette di passare variabili extra al comando Ansible e caricarle usando il file YAML.

Il comando è il seguente:

```
ansible -i test.ini all -m ping -e "@secrets.yml"
```

3.6 Installazione della Dashboard e PlayBook

Utilizzare OpenStack Dashboard ha dei vantaggi, tra cui una GUI che permette di sfruttare appieno e comodamente le sue funzionalità.
Installare e configurare un servizio o un insieme di servizi, con l'utilizzo di playbook e file di inventario, favorisce una configurazione centralizzata e sicura, riducendo il rischio di errori manuali e migliorando la ripetibilità del processo di distribuzione.
Il primo comando è utilizzato per preparare i server per l'installazione, eseguendo una serie di operazioni preliminari sui nodi definiti nel file di inventario.

- **`kolla-ansible`**: comando per eseguire kolla-ansible
- **`-i ./multinode.ini`**: specifica il file di inventario (inventory file) da utilizzare e che deve essere eseguito su tutti gli host elencati all'interno.
- **`bootstrap-servers`**: esegue i playbook di Ansible necessari per preparare i server all'installazione.
- **`-e "@secrets.yml"`**: permette di passare variabili extra al comando Ansible e caricarle usando il file YAML.

```
kolla-ansible -i ./multinode.ini bootstrap-servers -e
↪ "@secrets.yml"
```

Le attività preliminari di bootstrap includono:

- **Configurazione delle impostazioni di base del sistema operativo.**
- **Installazione delle dipendenze necessarie.**
- **Configurazione della rete.**
- **Creazione di utenti e gruppi necessari.**
- **Configurazione di SSH e altre impostazioni di sicurezza.**

Nella fase dei prechecks, Kolla-Ansible viene utilizzato per orchestrare il deployment di OpenStack verificando che tutti i nodi soddisfino i requisiti necessari e controllando configurazioni di rete, risorse e altri parametri critici.

- **`kolla-ansible`**: comando per eseguire kolla-ansible
- **`-i ./multinode.ini`**: specifica il file di inventario (inventory file) da utilizzare.
- **`prechecks`**: esegue una serie di verifiche preliminari sui nodi definiti nel file di inventario. Le verifiche prechecks assicurano che tutte le configurazioni e i prerequisiti siano soddisfatti prima di procedere con l'installazione di OpenStack.
- **`-e "@secrets.yml"`**: permette di passare variabili extra al comando Ansible e caricarle usando il file YAML.

```
kolla-ansible -i ./multinode.ini prechecks -e "@secrets.yml"
```

Le attività del prechecks sono:

- **Verifica delle risorse del sistema.**
- **Verifica delle dipendenze software.**
- **Verifica della rete.**
- **Verifica della configurazione ansible.**
- **Verifica delle variabili di configurazione.**

Dopo aver eseguito i prechecks, la fase di deploy in kolla-ansible è responsabile della distribuzione effettiva dei servizi OpenStack nell'infrastruttura. In questa fase, Kolla-Ansible utilizza i file di configurazione e le credenziali, come quelle definite in secrets.yml, per distribuire e configurare automaticamente i container Docker che gestiscono i diversi componenti di OpenStack.

- **`kolla-ansible`**: comando per eseguire kolla-ansible
- **`-i ./multinode.ini`**: specifica il file di inventario (inventory file) da utilizzare.
- **`deploy`**: esegue una distribuzione sui nodi indicati nel file di inventario.
- **`-e "@secrets.yml"`**: permette di passare variabili extra al comando Ansible e caricarle usando il file YAML.

```
kolla-ansible -i ./multinode.ini deploy -e "@secrets.yml"
```

Le attività del deploy sono:

- **Creazione e configurazione dei container Docker.**
- **Installazione dei componenti core di OpenStack.**
- **Configurazione dei volumi di storage.**
- **Creazione di reti virtuali.**
- **Avvio dei servizi OpenStack.**

La quarta fase riguarda l'installazione effettiva del Client di OpenStack (CLI), un'interfaccia a riga di comando unificata che permette agli utenti di interagire e gestire una distribuzione OpenStack. Consente di eseguire operazioni su vari servizi di OpenStack, come la gestione delle istanze di calcolo (Nova), le reti (Neutron), le risorse di storage (Cinder) e molto altro.

```
pip install python-openstackclient -c
↪ https://releases.openstack.org/constraints/upper/xena
```

La fase di post-deploy è parte del processo di distribuzione ed è essenziale per rifinire la configurazione dell'ambiente e garantire che tutto funzioni correttamente.

- **`kolla-ansible`**: comando per eseguire kolla-ansible
- **`-i ./multinode.ini`**: specifica il file di inventario (inventory file) da utilizzare.

- **deploy**: esegue una distribuzione sui nodi indicati nel file di inventario.
- **-e "@secrets.yml"**: permette di passare variabili extra al comando Ansible e caricarle usando il file YAML.

```
kolla-ansible post-deploy -e "@secrets.yml"
```

Le attività del deploy sono:

- **Generazione del file admin-openrc.sh.**
- **Configurazione del client OpenStack.**
- **Verifica della funzionalità dei servizi**
- **Abilitazione del monitoraggio e logging**
- **Backup delle configurazioni**
- **Abilitazione di servizi complementari**

Dopo la fase di post-deploy, una delle prime operazioni necessarie per interagire con i servizi OpenStack è caricare le credenziali generate durante la distribuzione. Per farlo deve essere avviato lo script generato dal post-deploy.

```
. /etc/kolla/admin-openrc.sh
```

All'interno dello script troviamo le seguenti variabili:

- **OS_AUTH_URL**: Specifica l'URL dell'endpoint del servizio di autenticazione e autorizzazione Keystone di OpenStack. I client si possono autenticare e possono Wottenere i token di autenticazione necessari per accedere agli altri servizi di OpenStack.
- **OS_PROJECT_ID / OS_TENANT_ID**: Specifica il progetto (o tenant) nel quale l'utente sta lavorando usando un ID numerico. OpenStack definisce i progetti per isolare le risorse e le operazioni tra diversi gruppi di utenti assicuradosi che le operazioni vengano eseguite nel contesto progetto/tenant corretto e garantendo l'isolamento e la sicurezza delle risorse.

- **OS_PROJECT_NAME / OS_TENANT_NAME**: nome del progetto/tenant per identificarlo in alternativa ad un ID numerico.
- **OS_USER_DOMAIN_NAME**: nome del dominio a cui appartiene l'utente. Nei deployment multi-dominio, rende più semplice distinguere utenti con lo stesso nome, ma in domini differenti, oppure garantire autenticazione e autorizzazione dell'utente identificandolo univocamente.
- **OS_PROJECT_DOMAIN_ID**: Specifica il dominio a cui appartiene il progetto. Anche il progetto deve essere identificato in modo univoco per supportare il multi-tenancy separando e gestendo finemente le risorse.
- **OS_USERNAME**: il nome utente che si autentica nei servizi OpenStack.
- **OS_PASSWORD_INPUT**: Utilizzato per leggere in input e in sicurezza da linea di comando la password inserita dall'utente.
- **OS_PASSWORD**: contiene la password per l'autenticazione dell'utente impostata da OS_PASSWORD_INPUT.
- **OS_REGION_NAME**: Specifica la regione geografica o logica in cui operare. Viene utilizzato per identificare un insieme di servizi OpenStack in un'area specifica.
- **OS_INTERFACE**: Specifica quale interfaccia di rete utilizzare per le operazioni. Determina attraverso quale interfaccia i servizi di OpenStack sono accessibili.
  - public: permette l'accesso esterno.
  - internal: limitato alla rete interna
- **OS_IDENTITY_API_VERSION**: Specifica la versione dell'API di Keystone da utilizzare per l'autenticazione e la gestione dell'identità. La versione in questione, la 3, introduce supporto per i domini e migliora la sicurezza.

Il file è il medesimo:

```
export OS_AUTH_URL=http://192.168.11.104:5000
export OS_PROJECT_ID=ffdb4a5d9ee74091abba532f48534abb
export OS_PROJECT_NAME="admin"
export OS_USER_DOMAIN_NAME="Default"
if [ -z "$OS_USER_DOMAIN_NAME" ]; then unset
 ↪  OS_USER_DOMAIN_NAME; fi
export OS_PROJECT_DOMAIN_ID="default"
if [ -z "$OS_PROJECT_DOMAIN_ID" ]; then unset
 ↪  OS_PROJECT_DOMAIN_ID; fi
unset OS_TENANT_ID
unset OS_TENANT_NAME
export OS_USERNAME="admin"
echo "Please enter your OpenStack Password for project
 ↪  $OS_PROJECT_NAME as user $OS_USERNAME: "
read -sr OS_PASSWORD_INPUT
export OS_PASSWORD=$OS_PASSWORD_INPUT
export OS_REGION_NAME="RegionOne"
if [ -z "$OS_REGION_NAME" ]; then unset OS_REGION_NAME; fi
export OS_INTERFACE=public
export OS_IDENTITY_API_VERSION=3
```

L'indirizzo a cui accedere per sfruttare la piattaforma è l'indirizzo IP che era stato utilizzato come VIP, mentre il nome utente sarà "admin" mentre la password si trova all'interno della variabile "`keystone_admin_password`" del file `passwords.yml`, come mostrato in Figura 3.1.

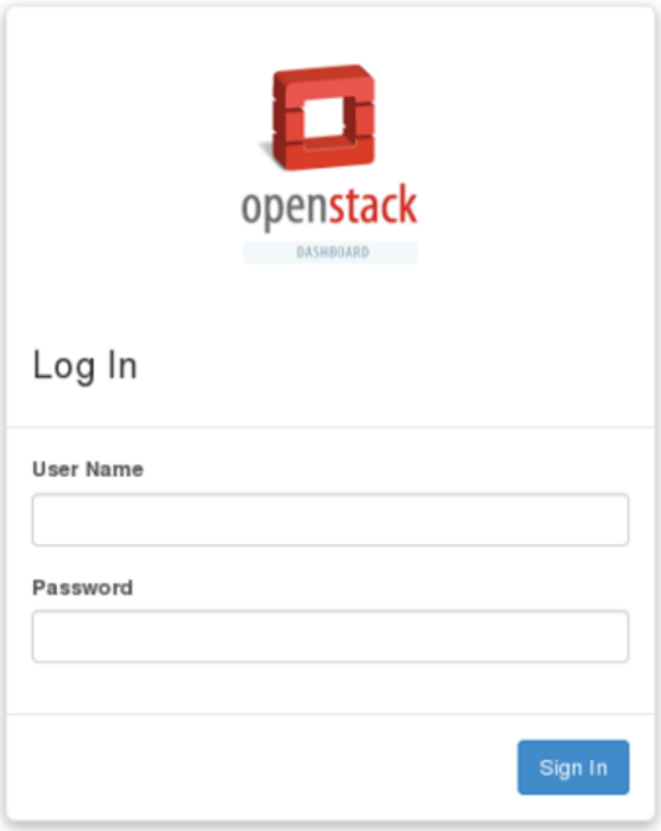

Figura 3.1: Schermata di login di OpenStack Horizon

Se si volesse poi avviare una demo per provare la piattaforma, c'è un file script `init-runonce.sh` che configura l'ambiente di test OpenStack con pochi comandi. Comandi che verranno spiegati nel capitolo successivo. É utile per testare rapidamente le funzionalità di base senza dover configurare manualmente ogni risorsa. Ora è possibile sfruttare tutto il potenziale di OpenStack.

Capitolo 4

User experience

4.1 Horizon

4.1.1 Panoramica della Dashboard

Horizon è l'interfaccia web di OpenStack che fornisce un modo facile e intuitivo per gestire le risorse cloud. La dashboard offre agli utenti e agli amministratori una vista centralizzata delle risorse disponibili, come istanze, volumi, reti e immagini. Dal menu laterale, è possibile accedere alle diverse funzionalità, come la gestione dei progetti, la configurazione di utenti e la visualizzazione dei log.
La dashboard è divisa in schede per facilitare la navigazione e consentire una gestione efficiente di ogni componente di OpenStack, come mostrato in Figura 4.1.

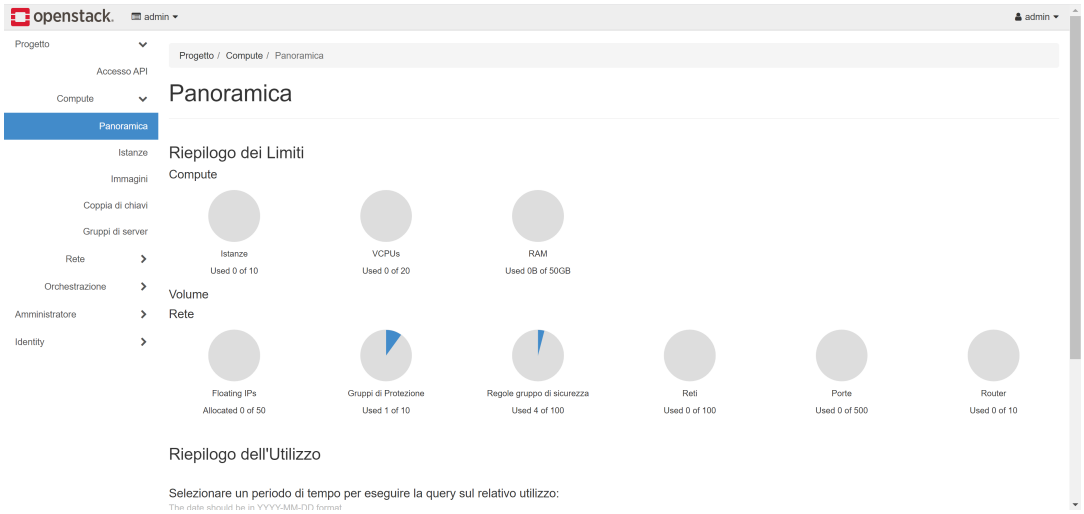

Figura 4.1: Panoramica della Dashboard Horizon

4.1.2 Creazione di una nuova istanza

Per creare una nuova istanza (macchina virtuale) tramite Horizon, è necessario seguire alcuni passaggi:

1. Navigare nella sezione Istanze.
2. Cliccare su Avvia Istanze.
3. Selezionare un'immagine disponibile (ad esempio, una distribuzione Linux o Windows).
4. Configurare il flavor, che determina le risorse hardware (CPU, RAM, disco).
5. Configurare le opzioni di rete, collegando l'istanza alla rete desiderata.
6. Cliccare su Avvia per lanciare l'istanza.

4.1.3 Attaccare il volume ad un'istanza

Horizon consente di attaccare volumi di storage aggiuntivi a istanze già in esecuzione:

1. Navigare nella sezione Volumi e creare un nuovo volume se necessario.
2. Una volta creato, tornare alla pagina delle istanze.
3. Cliccare sul menu a discesa accanto all'istanza e selezionare Attacca Volume.
4. Selezionare il volume desiderato e cliccare su Attacca.

4.1.4 Gestione delle reti

Le reti virtuali in OpenStack sono fondamentali per connettere le istanze e fornire accesso ai servizi. Tramite Horizon:

1. Navigare nella sezione Reti e creare una nuova rete.
2. Definire una subnet associata alla rete.
3. Configurare i router e le regole di sicurezza per controllare il traffico.
4. Assegnare un floating IP alle istanze per consentire l'accesso esterno.

4.1.5 Creare e gestire volumi di Storage

I volumi di storage possono essere creati e gestiti direttamente dalla dashboard di Horizon:

1. Andare nella sezione Volumi e cliccare su Crea Volume.
2. Specificare la dimensione del volume e un nome.
3. Una volta creato, il volume può essere associato a una o più istanze, o utilizzato per il backup di dati.
4. Horizon permette anche la gestione dei snapshot per creare punti di ripristino del volume.

4.1.6 Creare e gestire immagini

Le immagini in OpenStack rappresentano i template di sistema operativo utilizzati per creare nuove istanze. In Horizon, è possibile:

1. Caricare nuove immagini nella sezione Immagini.
2. Specificare i dettagli dell'immagine, come il formato (QCOW2, RAW) e l'architettura.
3. Le immagini caricate possono essere utilizzate come base per nuove istanze.
4. È anche possibile gestire immagini personalizzate create da snapshot di istanze esistenti.

4.1.7 Orchestrazione (Heat)

Heat è lo strumento di orchestrazione di OpenStack, che consente agli utenti di definire infrastrutture complesse come codice. In Horizon:

- È possibile caricare template di orchestrazione YAML nella sezione Stacks.
- Un template Heat definisce risorse come istanze, volumi, reti e regole di sicurezza.
- Avviando uno stack, Heat si occupa di creare tutte le risorse definite nel template in maniera automatica e orchestrata.

4.1.8 Configurazione Avanzata

La configurazione avanzata tramite Horizon include:

- Gestione delle chiavi di crittografia per la sicurezza delle risorse.
- Configurazione delle regole firewall e dei gruppi di sicurezza per proteggere le istanze.
- Gestione delle quote di progetto, che controllano l'allocazione delle risorse (CPU, RAM, storage) tra gli utenti.

4.2 CLI

4.2.1 Panoramica della Dashboard

La Command Line Interface (CLI) di OpenStack offre una modalità interattiva per amministrare le risorse cloud direttamente dal terminale. A differenza della dashboard Horizon, la CLI permette di eseguire operazioni tramite comandi, offrendo una maggiore flessibilità per gli utenti che desiderano automatizzare processi o gestire ambienti complessi. Ogni componente di OpenStack ha una serie di comandi CLI specifici (ad esempio, nova per le istanze, neutron per la rete, e così via), ma molti di questi sono stati consolidati nel comando unificato openstack, che facilita l'interazione con tutte le risorse del cloud.
Per utilizzare la CLI, è necessario caricare le credenziali contenute nel file admin-openrc.sh, che include variabili come il nome utente e la password per l'autenticazione, oltre a informazioni sul progetto e il dominio.

4.2.2 Creazione di una nuova istanza

La creazione di una nuova istanza tramite la CLI può essere eseguita con il comando `openstack server create`. Per avviare un'istanza, è necessario specificare alcuni parametri, tra cui:

- **`--image <image_name>`**: nome dell'immagine da utilizzare per l'istanza.
- **`--flavor <flavor_name>`**: specifica il flavor che determina le risorse hardware dell'istanza.
- **`--network <network_name>`**: Nome della rete alla quale l'istanza sarà collegata.
- **`<server_name>`**: Nome da dare all'istanza.

```
openstack server create --image <image_name> --flavor
↪  <flavor_name> --network <network_name> <server_name>
```

Opzionalmente, è possibile aggiungere chiavi SSH, specificare volumi di storage e altre configurazioni tramite parametri aggiuntivi.

4.2.3 Creare e gestire volumi di Storage

La gestione dei volumi di storage avviene principalmente tramite il comando `openstack volume`. É possibile creare nuovi volumi, cancellarli o attaccarli a diverse istanze. Questi volumi rappresentano unità di storage persistente, che rimangono disponibili anche se l'istanza viene eliminata. E i parametri sono i seguenti:

- `--size <size_in_GB>`: dimensione del volume in gigabyte (GB).
- `<volume_name>`: nome da assegnare al nuovo volume.

Comando per creare un volume:

```
openstack volume create --size <size_in_GB> <volume_name>
```

Comando per cancellare un volume:

```
openstack volume delete <volume_name>
```

É possibile controllare i volumi esistenti utilizzando il comando `openstack volume list`.

4.2.4 Attaccare il volume ad un'istanza

Per attaccare un volume di storage a un'istanza, si utilizza il comando `openstack server add volume`. Prima di farlo, il volume deve essere stato precedentemente creato. Questo permette di espandere lo spazio di archiviazione dell'istanza, utile per applicazioni che richiedono un maggior utilizzo di dischi. I parametri sono:

- `<server_name>`: nome dell'istanza (server) a cui si desidera aggiungere il volume.
- `<volume_id>`: ID del volume che si desidera attaccare all'istanza.

Comando per aggiungere un volume:

```
openstack server add volume <server_name> <volume_id>
```

Comando per rimuovere un volume:

```
openstack server remove volume <server_name> <volume_id>
```

4.2.5 Gestione delle reti

La gestione delle reti tramite la CLI include la creazione di reti, subnet e router per connettere le istanze. Utilizzando comandi come `openstack network` e `openstack subnet`, è possibile creare reti private e associare indirizzi IP alle istanze semplicemente assegnando un nome ad una rete con il parametro `<network_name>`.

```
openstack network create <network_name>
```

É possibile anche creare una subnet associata, usando i seguenti parametri:

- **`--network <network_name>`**: nome della rete alla quale associare la subnet.
- **`--subnet-range <CIDR>`**: intervallo di indirizzi IP da utilizzare per la subnet, specificato nel formato CIDR (Classless Inter-Domain Routing).
- **`<subnet_name>`**: nome da assegnare alla nuova subnet.

```
openstack subnet create --network <network_name>
↪  --subnet-range <CIDR> <subnet_name>
```

Le reti create possono essere successivamente collegate alle istanze con il comando `openstack server add network`.

4.2.6 Creare e gestire immagini

Le immagini sono template preconfigurati di sistemi operativi o software che vengono utilizzati per avviare nuove istanze. Con la CLI, le immagini possono essere gestite con il comando openstack image. Per avviare uno stack Heat tramite la CLI servono questi parametri:

- **`--file <file_path>`**: percorso del file da cui si desidera creare l'immagine. Questo file può essere un'immagine ISO, una virtual disk image (VDI) o altri formati supportati.
- **`--disk-format <format>`**: formato del disco dell'immagine. I formati comuni includono qcow2, raw, vmdk, e iso.
- **`<image_name>`**: nome da assegnare alla nuova immagine.

```
openstack image create --file <file_path> --disk-format
↪  <format> <image_name>
```

Le immagini possono essere aggiornate, listate e cancellate con comandi specifici.

4.2.7 Orchestrazione (Heat)

L'orchestrazione in OpenStack viene gestita tramite Heat, che permette di definire infrastrutture complesse come codice. Attraverso file YAML chiamati template, è possibile specificare un insieme di risorse (istanze, volumi, reti, etc.) e far sì che vengano create e gestite in maniera automatica. I parametri da utilizzare sono:

- –template **`<template_file>`**: Percorso del file template da utilizzare per la creazione dello stack. Il template definisce le risorse da creare, come istanze, reti e volumi.
- **`<stack_name>`**: Nome da assegnare al nuovo stack.

```
openstack stack create --template <template_file> <stack_name>
```

Dopo aver creato lo stack, è possibile monitorare lo stato e le risorse gestite tramite openstack stack show e openstack stack update per modificarne il comportamento.

4.2.8 Configurazione Avanzata

La configurazione avanzata tramite CLI include operazioni più complesse come:

- **Gestione delle chiavi di sicurezza**: le chiavi SSH possono essere aggiunte e gestite con `openstack keypair`.
- **Gestione delle regole di sicurezza**: regole di firewall e sicurezza possono essere configurate tramite `openstack security group` per limitare il traffico verso le istanze.
- **Quota di progetto**: tramite CLI, è possibile gestire le risorse assegnate a ciascun progetto con `openstack quota set`.

Conclusioni

Risultati ottenuti

L'obiettivo principale di questo lavoro è stato l'installazione, la configurazione e la gestione di OpenStack tramite Horizon e CLI, per garantire una gestione efficiente delle risorse cloud distribuite su più nodi. Inizialmente, si è proceduto con l'analisi del contesto e l'identificazione delle tecnologie più adeguate per il deployment di OpenStack. Successivamente, sono state condotte diverse attività chiave, tra cui l'automatizzazione del processo di installazione con strumenti come Kolla Ansible e la configurazione dei principali servizi di OpenStack, tra cui Nova, Neutron e Cinder.
La gestione delle risorse cloud tramite Horizon ha permesso una visione chiara e intuitiva delle operazioni amministrative, facilitando la creazione di istanze, volumi e reti in modo semplice ed efficiente. Parallelamente, l'utilizzo della CLI ha garantito maggiore flessibilità e controllo per configurazioni avanzate e task automatizzati. Entrambi gli approcci hanno dimostrato l'importanza di bilanciare accessibilità e potenza nella gestione di ambienti cloud complessi.
Per quanto riguarda la distribuzione del carico sui nodi, il sistema ha dimostrato ottime prestazioni, con un'elevata scalabilità e flessibilità nell'allocazione delle risorse. L'orchestrazione tramite Heat ha semplificato la distribuzione e gestione automatizzata delle risorse, riducendo il carico amministrativo.
Nel complesso, il progetto ha raggiunto gli obiettivi prefissati, garantendo una piattaforma stabile e scalabile per la gestione delle risorse cloud.

Sviluppi futuri

Nonostante i risultati positivi, ci sono diversi margini di miglioramento per futuri sviluppi. Ad esempio, è possibile esplorare soluzioni più avanzate per la gestione della sicurezza, implementando politiche di autenticazione e autorizzazione più raffinate tramite Keystone. Inoltre, sarebbe interessante migliorare l'integrazione con piattaforme di containerizzazione come Kubernetes, per gestire workload più dinamici.
Un altro aspetto che merita attenzione è l'ottimizzazione della distribuzione multinodo, esplorando tecnologie come Docker Swarm o Kubernetes per distribuire i componenti di OpenStack su infrastrutture distribuite. Ciò migliorerebbe la resilienza e scalabilità del sistema, consentendo una migliore gestione delle risorse su più macchine fisiche e datacenter.
Infine, un futuro studio potrebbe concentrarsi sull'introduzione di soluzioni serverless integrate, offrendo nuove possibilità di ottimizzazione delle risorse e una maggiore flessibilità nella gestione delle applicazioni distribuite su OpenStack.

Apprendimento sul progetto sviluppato

In conclusione, questa esperienza ha permesso di approfondire la conoscenza dei sistemi cloud distribuiti, migliorando la comprensione delle tecnologie chiave alla base di OpenStack, come Horizon, CLI e l'orchestrazione tramite Heat. Gli strumenti utilizzati hanno dimostrato il loro valore nell'automatizzazione delle operazioni e nella gestione efficace delle risorse, offrendo un punto di partenza solido per futuri progetti cloud.

Bibliografia

[1] Kolla-Ansible. Quick start for deployment/evaluation. URL https://docs.openstack.org/kolla-ansible/latest/user/quickstart.html.

[2] OpenStack. Openstack services, . URL https://www.openstack.org/software/project-navigator/openstack-components#openstack-services.

[3] OpenStack. Administrator guides for openstack services, . URL https://docs.openstack.org/2024.2/admin/.

[4] OpenStack. User guides for openstack services, . URL https://docs.openstack.org/2024.2/user/.