



Implementazione di un servizio dimostrativo (PoC) di container-image

Documentazione

Candidato: Niccolò Zuccolo
Classe: Informatica MI4
Anno scolastico: 2019/2020
Formatore: Alessandro Prato
Azienda: CSCS

Sommario

1.	Analisi preliminare	3
1.1	Introduzione.....	3
1.2	Obiettivi	3
2.	Analisi / Design.....	4
2.1	Concetto e prototipazione	4
2.2	Rischi tecnici	9
2.3	Schema di rete	10
2.4	Pianificazione	11
2.5	Progettazione	14
3.	Realizzazione	16
3.1	Preparazione e inizializzazione di Terraform.....	16
3.2	Scrittura del file Terraform (IaC).....	21
3.3	Esecuzione del file Terraform.....	26
3.4	Configurazione del dominio.....	30
3.5	Installazione e configurazione di GitLab (Registry)	34
3.6	Utilizzo di GitLab e comandi per operare con Registry.....	37
3.7	Installazione del server mail	42
3.8	Installazione di Ossec (HIDS).....	44
3.9	Installazione di M/Monit (monitoraggio dei processi).....	48
3.10	Implementazione di M/Monit.....	49
3.11	Implementazione delle Snapshot in Google Cloud.....	53
4.	Protocollo di test.....	61
4.1	Verifica dell'integrità del progetto	61
4.2	Test credenziali e accesso a GitLab.....	62
4.3	Simulazione di avaria per riscontro da Ossec	64
4.4	Verifica del funzionamento dei trigger M/Monit.....	65
4.5	Test di accesso a porte bloccate con firewall	67
5.	Analisi dei costi.....	70
6.	Glossario	72
7.	Conclusioni e ringraziamenti	74
8.	Allegati.....	76
8.1	Approfondimenti	76
8.2	Documenti allegati.....	77

1. Analisi preliminare

1.1 Introduzione

Il progetto ha lo scopo di analizzare e cercare la migliore soluzione per poter creare in Google Cloud Platform un'infrastruttura virtuale automatizzata che permetta di gestire un Docker Image Registry. Per quanto riguarda il Docker Image Registry sarà necessario implementarlo in modo che abbia le funzionalità di autenticazione e autorizzazione degli accessi.

Per completare ciò la macchina virtuale verrà monitorata da software che puntualmente dovranno avvisare gli amministratori in casi di allarme: se ad esempio il servizio di container Registry dovesse smettere di funzionare, puntualmente gli amministratori verranno informati.

Un elemento importante del progetto è che la maggior parte delle operazioni da effettuare nelle impostazioni e configurazioni di Google Cloud saranno effettuate mediante l'esecuzione di un "infrastructure as Code". Inoltre per le installazioni degli ulteriori software verranno creati degli script in modo da automatizzarne e facilitarne la gestione.

Grazie all'automatizzazione dei processi di creazione di tutta l'infrastruttura in Google Cloud e agli script, l'installazione di tutti i software nel sistema operativo dell'infrastruttura saranno notevolmente agevolati. Si potrà trarne un enorme vantaggio in termini di tempo e i costi saranno sensibilmente ridotti.

Il tempo per lo svolgimento del lavoro è di 80 ore ed è previsto venga ultimato entro 10 giorni (quindi 8 ore al giorno).

1.2 Obiettivi

Gli obiettivi di questo progetto sono i seguenti:

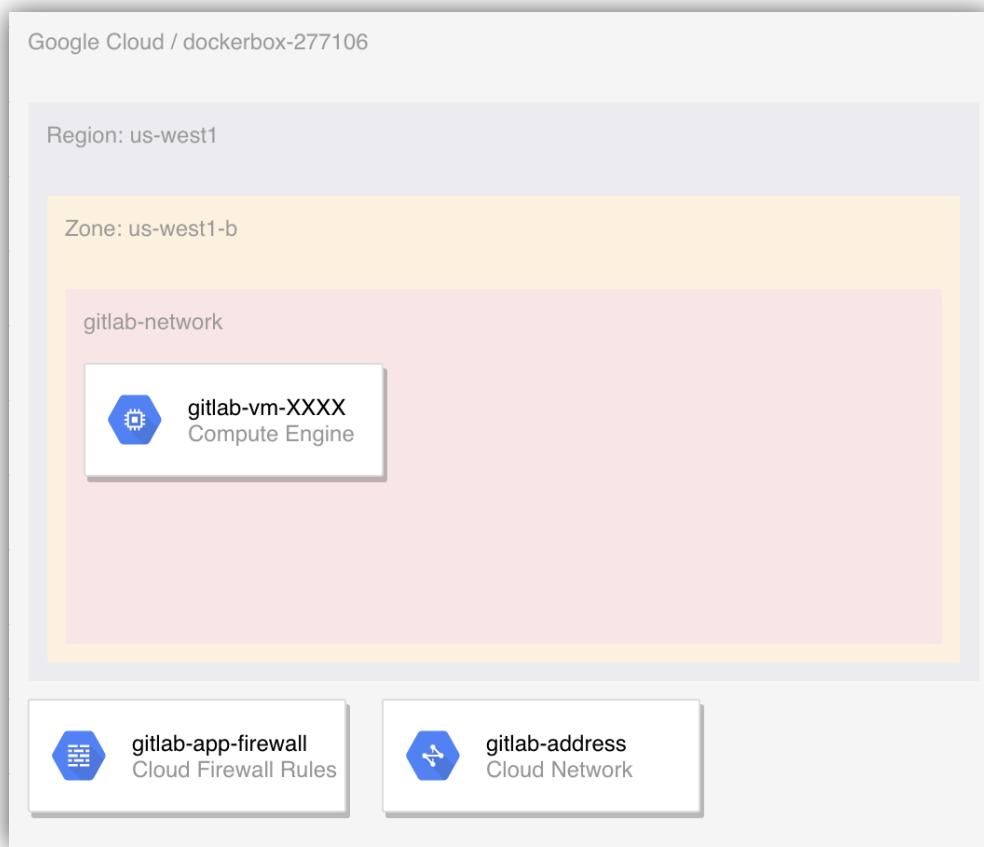
- il servizio Registry deve essere perfettamente compatibile con Docker;
- deve esserci un controllo di autenticazione e autorizzazione per gli utenti;
- l'infrastruttura deve essere gestita mediante un modello descrittivo come l'"Infrastructure as Code" (IaC);
- occorre rendere conto al committente in modo puntuale ed esaustivo di tutte le operazioni eseguite nel progetto;
- dev'esserci un sistema di backup, in modo che in caso di malfunzionamento si possa ripristinare a uno stato precedente tutto il sistema (il sistema deve garantire la riproducibilità dell'infrastruttura);
- l'accesso ai servizi dev'essere protetto da un firewall di rete;
- dev'esserci un monitoraggio attivo di tutte le attività.

2. Analisi / Design

2.1 Concetto e prototipazione

Hardware

Come hardware ci affideremo a un'infrastruttura virtuale basata su Google Cloud: qui di seguito sono elencati gli elementi indispensabili con i relativi dettagli.



Schema concetto hardware

Dockerbox-227106

Dockerbox è il nome del progetto in Google Cloud: all'interno di questo progetto verrà creata tutta l'infrastruttura virtuale.

Gitlab-vm-XXXX

Questa è la macchina virtuale nella quale verranno installati tutti i software necessari per un buon funzionamento.

Gitlab-network

Gitlab network è un “contenitore”, una rete virtuale, nella quale verrà inserita la nostra macchina virtuale.

Gitlab-address

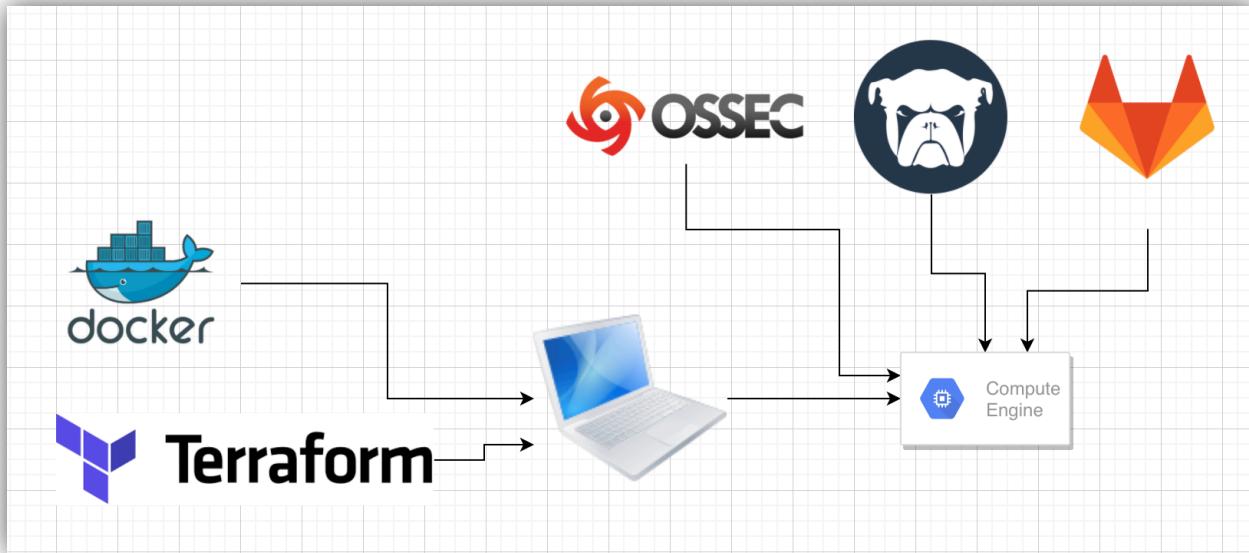
Verrà riservato un indirizzo IP pubblico con il quale potremo connetterci al network virtuale e, grazie a un Redirect/Forward, potremo accedere alla macchina all'interno della rete virtuale.

Gitlab-app-firewall

È il firewall all'interno del quale inseriremo tutte le regole necessarie per il funzionamento della nostra infrastruttura e per proteggerci dalle minacce provenienti dall'esterno.

Software utilizzato

Qui di seguito riportiamo una panoramica di tutti i software che comporranno questo progetto.

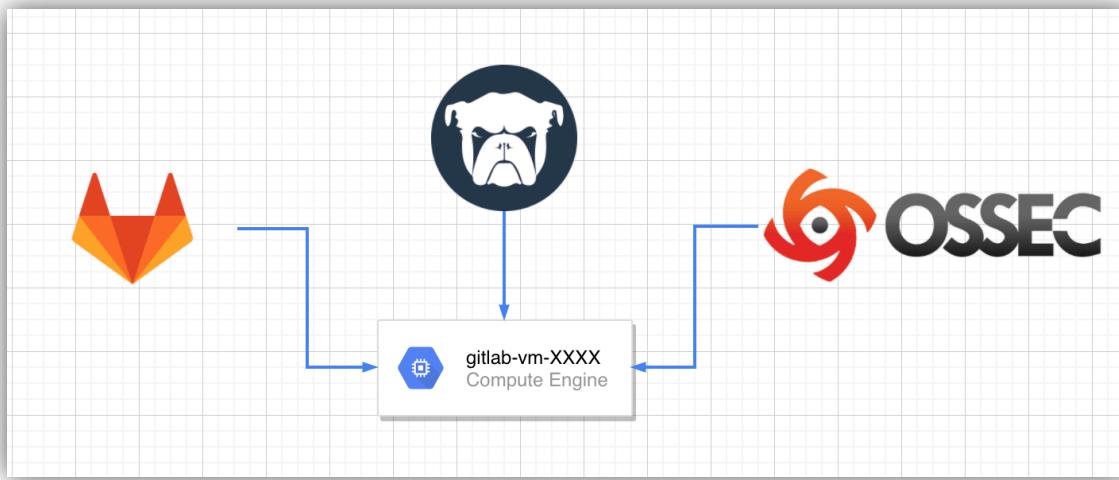


Schema di insieme dei software

Come si può vedere nello schema Docker e Terraform sono installati nel computer dal quale si gestirà tutta la infrastruttura Google Cloud. Nella piattaforma Google Cloud verrà creata un'istanza con la quale verrà installato Ossec GitLab e M/Monit.

Software sul Server

Per ottenere un ambiente operativo funzionale sarà necessario selezionare e utilizzare dei software: per ulteriori informazioni nel capitolo progettazione vengono spiegate tutte le motivazioni e le scelte per il loro utilizzo. Questi software verranno installati nella macchina virtuale in Google Cloud



Schema concetto dei software

GitLab

GitLab sarà il software utilizzato come Docker Image Registry, nello specifico utilizzeremo la funzione “Container Registry” per poter effettuare il push delle immagini Docker dal computer locale a esso.

M/Monit

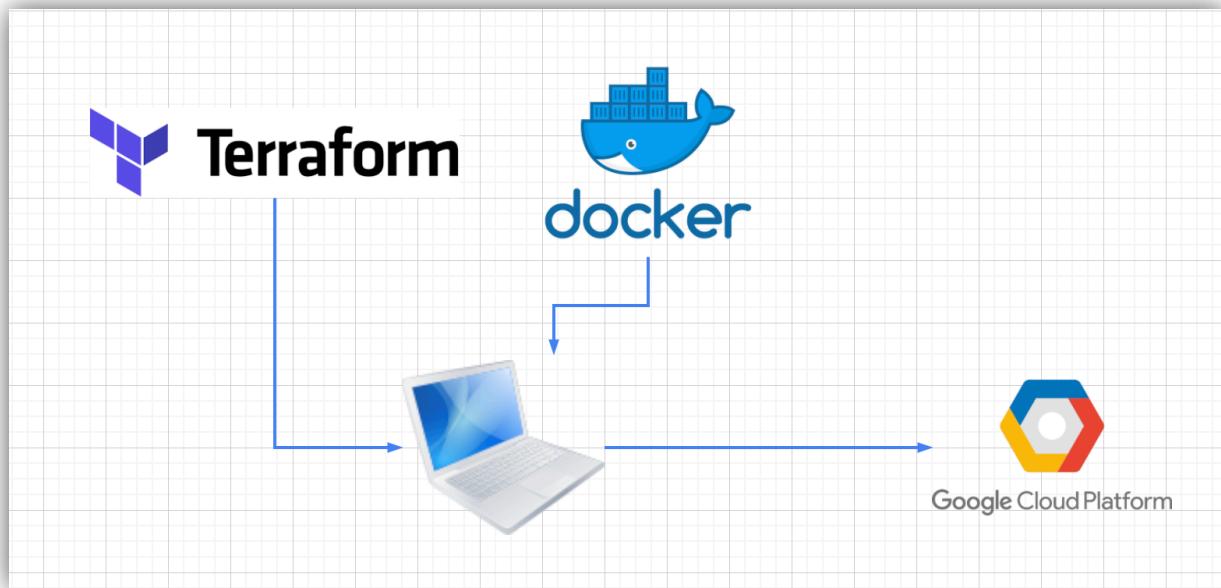
Per monitorare invece il buon funzionamento dei servizi web di GitLab e verificare che il servizio SSH sia acceso e funzionante, utilizzeremo M/Monit seguendo regole specifiche. Questo software dovrà anche monitorare l'utilizzo delle risorse del sistema e, in caso di avaria, dovrà notificarla prontamente agli amministratori.

Ossec

Sarà il nostro monitoraggio HIDS a supervisionare tutto ciò che avviene nel sistema operativo, come ad esempio accessi SSH, danneggiamenti di file e eventuali attacchi. In caso si verificassero imprevisti, gli amministratori saranno prontamente informati via email.

Software sul computer locale

Per chiarire meglio la posizione e l'utilizzo di altri software, che verranno illustrati nella documentazione, riporto un piccolo schema dei software che andranno utilizzati localmente per poter operare con il "Cloud".



Schema dei software utilizzati sul computer locale.

Terraform

Terraform è il software che andremo a installare nel nostro computer dal quale "dirigeremo" la piattaforma Google Cloud. Con esso potremo eseguire uno script della nostra infrastruttura e applicarla al Cloud o distruggerla in modo rapido e semplice.

Docker

Docker è un software che permette di creare container nel nostro computer locale. In questo caso noi lo utilizzeremo per accedere all'infrastruttura istanziata e lo utilizzeremo per potere operare con il "Container Registry" offerto da "GitLab". Docker viene installato solo per effettuare i test finali.

2.2 Rischi tecnici

In questo capitolo sono elencati i principali rischi tecnici che potrebbero verificarsi prima dell'inizio del progetto e l'eventuale soluzione per eludere o limitare questi rischi.

Notifica di avaria agli amministratori

Per poter permettere ai programmi di comunicare che il sistema è in avaria è necessario un server mail di posta in uscita. Finora non ho mai configurato localmente un mail server su una macchina virtuale, pertanto il lavoro di ricerca per poter effettuare questa operazione potrebbe richiedere tempo.

Per risolvere questo problema devo effettuare puntuale ricerche in internet che mi permettano di capire il concetto del server mail di posta in uscita. Inoltre sicuramente in internet posso trovare una documentazione completa per poter creare il mio server mail nell'istanza virtuale in Google Cloud.

Impostazione delle regole di Ossec

Ossec è un software particolarmente complesso che permette di monitorare il sistema operativo. Pertanto devo comprenderne le regole e approfondire quali siano tutti gli eventuali parametri.

Carenza di tempo

Considerata la complessità del lavoro, è possibile che non sia disponibile sufficiente tempo per poter effettuare al meglio tutti i passaggi. Per ridurre il rischio che si presenti questo problema, occorre controllare costantemente la pianificazione del progetto: è infatti necessario riflettere su tutte le fasi previste e definire in modo ottimale le tempistiche.

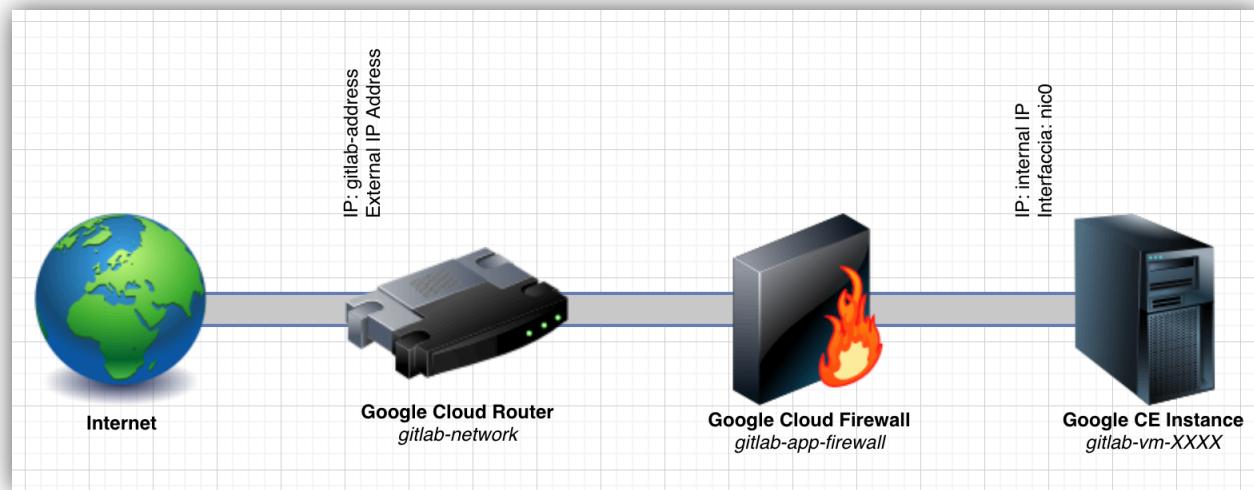
Attacco hacker

Internet è un luogo pieno di malintenzionati che mediante script e robot cercano puntualmente macchine virtuali esposte in rete e di conseguenza provano a entrarvi.

Per ridurre il rischio di accesso non autorizzato alla macchina virtuale posso ad esempio utilizzare il login mediante chiave pubblica e chiave privata.

2.3 Schema di rete

Il funzionamento dell'infrastruttura e della rete che andremo ad allestire è il seguente:



Schema di rete

Il principio di funzionamento dell'infrastruttura virtualizzata nel Cloud è molto semplice: come possiamo osservare, per poter entrare e uscire dall'istanza Google in internet bisogna per forza passare dal firewall.

Google Cloud Firewall

In esso verranno istanziate tutte le regole per poter garantire il funzionamento dei servizi che andremo a installare nella macchina virtuale, ma al contempo bloccheremo tutte le porte non utilizzate e non necessarie per la macchina virtuale.

Google Cloud Router

Il Google Cloud Router in realtà non esiste, ma nello schema serve per poter rappresentare al meglio il ruolo che ha la futura rete virtuale. Se creassimo più istanze virtuali simultaneamente nel nostro network, esse avranno indirizzi IP locali raggiungibili tra di loro. Un altro ruolo importante del router è quello di generare una NAT per le nostre istanze.

Siccome gli indirizzi IP vengono generati soltanto al momento della creazione di questa infrastruttura, a sua volta generata dall'orchestratore, non posso riportarli nello schema di rete.

Google CE Instance

È l'istanza virtuale nella quale risiedono il sistema operativo e tutti i software che andremo a installarvi: si possono trattare come se fossero un normale server fisico con all'interno un sistema operativo.

2.4 Pianificazione

La pianificazione del progetto è prevista su un lasso di tempo di 80 ore distribuite su 10 giorni (8 ore al giorno). Quale metodo di pianificazione scelgo il diagramma di Gantt, poiché permette di avere in una singola schermata un piano che rappresenta tutte le attività da svolgere e le tempistiche previste. Qui di seguito sono esposte le tappe per l'attuazione del lavoro e una piccola descrizione di come intendo procedere.

Analisi dell'incarico

In questa fase vengono analizzati l'incarico e i prerequisiti per poter attuare il progetto.

Allestimento dell'ambiente di lavoro

In questa fase viene allestito l'ambiente di lavoro, vengono creati il file del diario, lo scheletro della documentazione e un Repository GitHub nel quale viene caricato giornalmente lo sviluppo del progetto.

Scelta del software IaC

Qui viene scelto il software che si desidera utilizzare per definire l'infrastruttura come codice.

Scelta del software Registry

In questa fase viene scelto il software che si andrà a utilizzare per contenere le immagini Docker.

Scrittura file Terraform [IaC] + Health Check

In questa fase viene scritto il file della infrastruttura e vengono definiti gli Health Check.

Installazione di GitLab [REGISTRY]

In questa fase viene installato il software Registry scelto in precedenza.

Installazione di Ossec [HIDS]

Qui viene installato Ossec nell'istanza, ossia un software avanzato per il monitoraggio e l'integrità dei file di sistema.

Installazione di M/Monit [monitor dei processi]

In questa fase si installa e si configura il software M/Monit per monitorare e tenere traccia dei processi del sistema.

Implementare GCP CE Snapshots

In questo punto del progetto si provano e si documentano le snapshots in Google Cloud Platform.

Implementazione GCP Firewall

In questa fase viene implementato e testato il firewall in Google Cloud.

Test rollback dei backup

Dopo aver creato alcuni snapshot dell'ambiente, si effettuano dei rollback per verificare il funzionamento della piattaforma GCP.

Test di sicurezza Network

In questa fase si verifica e si testa il buon funzionamento del firewall in modo più specifico, si prova a accedere a porte bloccate e si controlla che le porte definite siano funzionanti.

Revisione del prodotto

In questa fase si cerca di caricare immagini docker nel Docker Registry.

Esecuzione del programma di test

In questa fase si effettua un protocollo di test per verificare che M/Monit e Ossec rispondano correttamente.

Breve controllo con il committente (ogni due giorni)

Nella seconda settimana di lavoro si controlla con il committente che tutti i parametri siano soddisfatti.

Scrittura della documentazione operativa

La scrittura della documentazione operativa avviene durante la realizzazione di tutto il progetto e nella seconda settimana ci si concentra maggiormente sulla parte dei test e degli schemi.

Redazione del diario di lavoro

La redazione del diario di lavoro avviene giornalmente; nella parte finale della giornata lavorativa vi si riassumono gli obiettivi raggiunti, i problemi sorti e le soluzioni adottate.

Redazione Abstract

Una volta che il progetto è terminato vengono fatti trasparire in un Abstract i punti fondamentali del progetto.

Controllo dei documenti generati dal progetto

Negli ultimi giorni si effettua un controllo di tutto il materiale che si intende consegnare.

Preparazione della consegna del progetto

Alla fine dei 10 giorni di elaborazione del progetto si prepara tutto il materiale per la consegna.

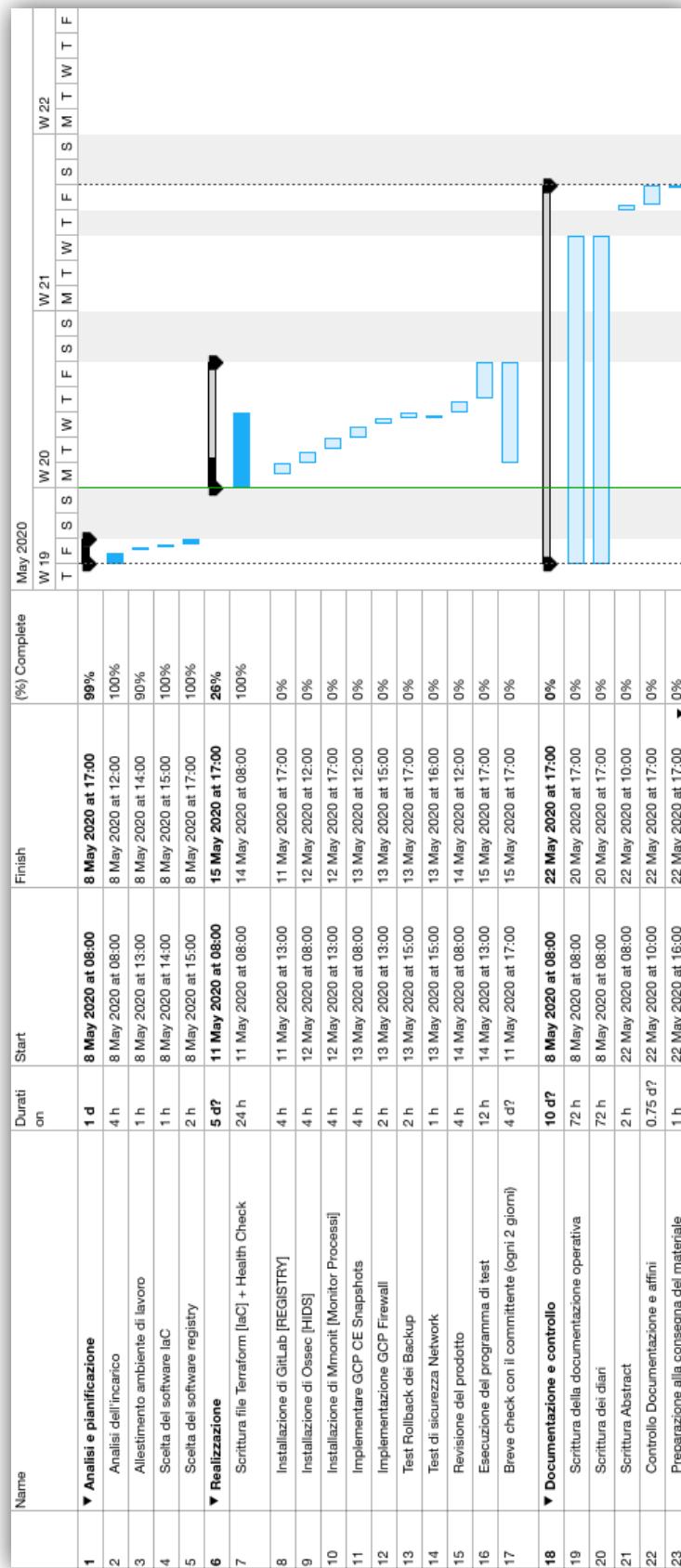


Diagramma di Gantt

Responsabile dopo la consegna del progetto

Una volta che il progetto sarà ultimato e consegnato, il responsabile che lo implementerà sarà Alessandro Prato: si occuperà di manutenere l'istanza virtuale e di gestire tutti gli accessi, nonché di osservare il buon funzionamento del prodotto.

2.5 Progettazione

Scelta del software GitLab (software Registry)

Nella scelta del software per il docker Registry ho avuto modo di provare tre diverse configurazioni.

Registry by Docker

Docker fornisce già un suo Registry che è possibile mettere in funzionamento mediante le istruzioni riportate in questa pagina: https://hub.docker.com/_/registry.

Di default Registry by Docker non viene fornito con un sistema di autenticazione già integrato.

Jfrog Artifactory

Jfrog Artifactory è un software commerciale prodotto dalla Jfrog. Per poter accedere e utilizzare la funzione registry è necessario essere in possesso di una licenza. Nella scheda dei requisiti del progetto è stato indicato preferibile far capo a software non commerciali.

GitLab

GitLab è un noto software di versioning dei file opensource, tuttavia dalla versione 8.8 di GitLab è stato introdotto il supporto al Docker registry di default, mentre prima bisognava abilitarlo manualmente dalle impostazioni del progetto.

Grazie a GitLab possiamo gestire gli utenti, i progetti, i permessi e gli accessi in un modo usuale e comodo.

Il CSCS attualmente utilizza GitLab come repository per dei files, ma non come Image Registry.

Scelta di Terraform (software Infrastructure as Code)

Nella scelta dei software per orchestrare e che permettono di scrivere un'infrastruttura come se fosse un codice, mi sono imbattuto principalmente in tre prodotti: Terraform, Ansible e Google Cloud Console SDK.

Google Cloud Console SDK è un software che si può installare sul proprio computer, che offre l'opportunità di interfacciarsi con l'infrastruttura e da cui si può lanciare un comando alla volta per mostrare o creare l'infrastruttura stessa. Sicuramente è un'ottima soluzione per poter eseguire singole operazioni, ma personalmente non mi ha molto convinto, poiché il linguaggio per comporre uno script sembra più complesso.

Ansible è un software molto flessibile che permette di definire una “Infrastructure as Code”, Ansible prevede di creare del codice procedurale e è un “configuration management”; Terraform invece è un orchestratore di infrastruttura, il tipo di codice che si scrive è di tipo dichiarativo. Pertanto preferisco utilizzare Terraform per questo progetto.

Terraform è un prodotto della HashiCorp ben documentato dalla stessa casa produttrice e anche ben approfondito da parte della Google: tutti i file di lavoro si trovano in un'unica cartella e, in caso di cambiamento di postazione di lavoro, si può riprendere l'infrastruttura in corso in modo semplice e dinamico. Terraform inoltre supporta ampiamente i diversi cloud presenti sul mercato. In un'ottica futura è possibile migrare la propria infrastruttura da un cloud all'altro.

Terraform non è solo potente, è il coltellino svizzero multi-piattaforma degli strumenti IAC. Sviluppato da HashiCorp, la stessa azienda che sta dietro a Vault e Nomad-Terraform, è completamente cloud-agnostico e aiuta ad affrontare grandi infrastrutture per applicazioni distribuite complesse, ad esempio, con più facilità rispetto al lavoro su una piattaforma specifica per il cloud.

Top 7 IaC Tools - <https://www.ibexlabs.com/top-7-infrastructure-as-code-tools/>

3. Realizzazione

3.1 Preparazione e inizializzazione di Terraform

In questo capitolo viene mostrato come si installa e configura Terraform per un conseguente utilizzo in un progetto sulla piattaforma Google Cloud. I prerequisiti sono un computer nel quale andremo a installare Terraform e un account attivo di Google Cloud.

Installazione di Terraform su Mac

Per installare Terraform nel modo più veloce e corretto è consigliabile utilizzare lo strumento HomeBrew.

1. Inizialmente aprire l'applicazione Terminal e lanciare la stringa di installazione di HomeBrew:

```
/bin/bash -c "$(curl -fsSL  
https://raw.githubusercontent.com/Homebrew/install/master/install.sh)"
```

2. Poi procedere con l'installazione effettiva del software Terraform:

```
brew install terraform
```

3. Per verificare che Terraform sia installato correttamente eseguire il seguente comando:

```
terraform -version
```

4. Questo è l'output che ci si deve aspettare:



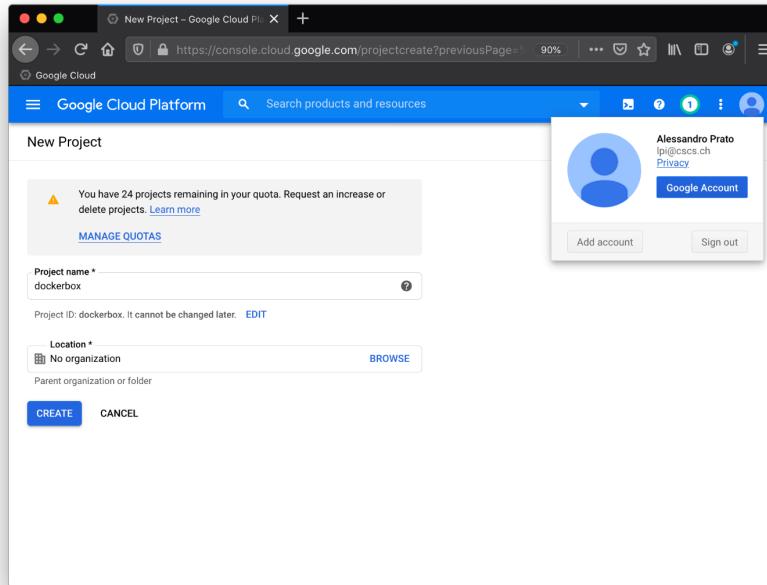
The screenshot shows a dark-themed macOS terminal window. At the top, there are three colored window control buttons (red, yellow, green) and the title bar which includes the user name 'niccolo', the host name 'root@gitlab-vm-8111', the current directory ' ~ - bash', and the terminal size '52x5'. Below the title bar, the command 'terraform -version' is typed in, followed by its output: 'Terraform v0.12.24'. The prompt '\$niccolo@macbookpro: ~ >' is visible at the bottom of the terminal window.

Terminale con l'esecuzione del comando

Creazione del progetto nella piattaforma Google Cloud

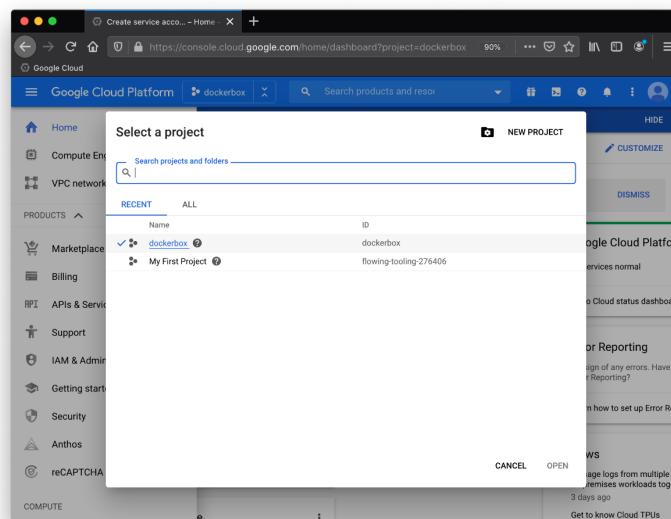
Per poter procedere alla creazione dell'ambiente di lavoro locale Terraform/Google è necessario creare un progetto nella piattaforma Google Cloud: qui di seguito la procedura.

1. Recarsi sul sito <https://console.cloud.google.com/> e creare un nuovo progetto.



Pagina di creazione del progetto “dockerbox” in Google Cloud

2. Selezionare il progetto “dockerbox” e attendere il caricamento dei servizi (come ad esempio “Compute Engine”).

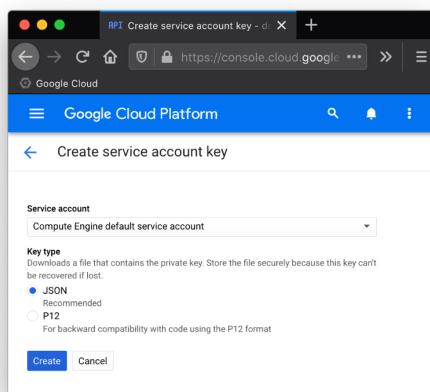


Schermata di selezione del progetto in Google Cloud

Ottenimento della chiave di accesso al progetto per Terraform

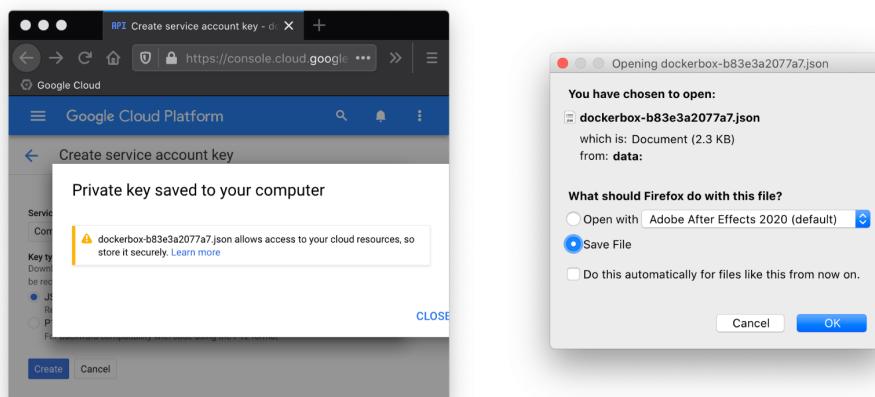
Per poter inizializzare il progetto bisogna creare una cartella nella quale inseriremo un file di tipo .json contenente i permessi per accedere al progetto nella GCP e un file .tf (Terraform) contenente il codice dell'infrastruttura. Per l'attuazione è opportuno seguire la procedura qui sotto elencata.

1. Recarsi sul sito internet
<https://console.cloud.google.com/apis/credentials/serviceaccountkey>.
2. Selezionare le impostazioni raffigurate nella seguente immagine e premere "Create".



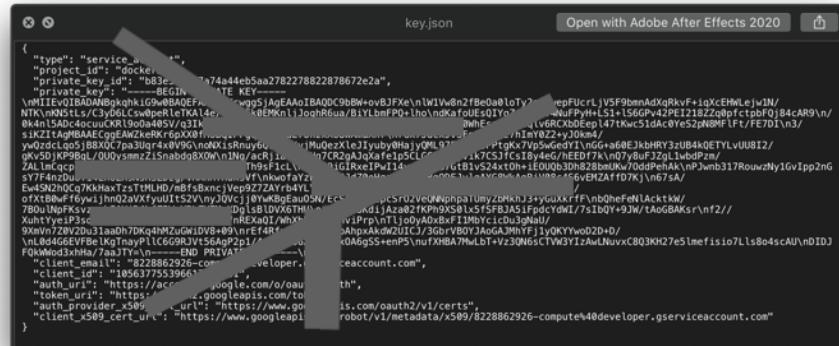
Pagina di creazione della “Service Account Key”

3. Dopo averla creata, la chiave verrà scaricata: salvarla nella cartella del progetto Terraform, per semplicità rinominare il file in “key.json”:



Schermata di scaricamento e salvataggio del file “Service Account Key”

4. Il contenuto del file “Service Account Key” auspicato si presenta nel seguente modo.

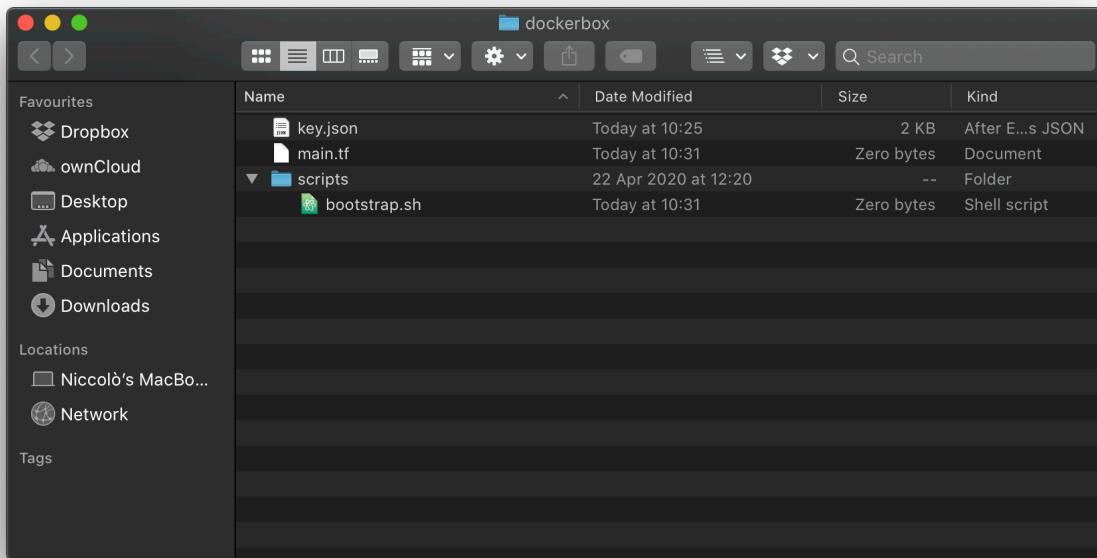


```
{ "type": "service_account", "project_id": "docke-74a4eb5a2782278822878672e2a", "private_key": "-----BEGIN KEY-----\nMIIEv0IBADANBgkqhkiG9w0BAQEFAQIExgQSjAqEAoIBAQDC9bBW+ovBjFXe/nlw1Vw8n2f8e0a0loTy2...repFUcrLjV5F9bnAdXqRkvF+1oXcEHwle]wIN/NTKInN5tsLsC3y6Glcsw0pekkleTKAl4e...k0EMKm1J0ohR6us/B1YLbmP0-lho\\ndKa0uf0s0Y?>\n-----END PRIVATE KEY-----", "client_email": "docke@docke-74a4eb5a2782278822878672e2a.iam.gcp.id", "client_id": "10788620849314147144", "auth_uri": "https://accounts.google.com/o/oauth2/auth", "token_uri": "https://oauth2.googleapis.com/token", "auth_provider_x509_cert_url": "https://www.googleapis.com/oauth2/v1/certs", "client_x509_cert_url": "https://www.googleapis.com/serviceaccount/v1/metadata/s589/8228862926-compute%40developer.gserviceaccount.com"}\n\n
```

Contenuto del file “key.json”

Creazione dei file di progetto

Per finalizzare la configurazione dell'ambiente di lavoro, procedere con la creazione di una cartella nella quale verranno inseriti tutti i file di progetto. Ecco come si presenta la cartella con il file key.json e i file vuoti su cui lavorare.



Contenuto della cartella “dockerbox”

Nel contenuto della cartella vi sono i seguenti file:

key.json

Questo file creato in precedenza contiene la “Service Account Key”. Si tratta di una chiave che permette a Terraform di interfacciarsi con Google Cloud e accedere alle risorse del progetto.

main.tf

Main.tf è il file nel quale va definita tutta l'infrastruttura da applicare a Google Cloud. Qui principalmente viene descritto tutto l'ambiente operativo, come per esempio la macchina virtuale, il network e il firewall.

scripts/bootstrap.sh

Questo è il file dello script del primo avvio: qui va definito tutto ciò che verrà eseguito al primo avvio della macchina virtuale istanziata nel servizio Compute Engine.

3.2 Scrittura del file Terraform (IaC)

Nel file Terraform attualmente vuoto “main.tf” inserisco aggiungendo blocco dopo blocco quanto segue.

Configurazione del provider

In questa prima parte si collega il file “key.json” e si inseriscono sostanzialmente il provider, il progetto e l’ubicazione in cui vogliamo creare la nostra intera infrastruttura.

```
// CONFIGURAZIONE PROVIDER
provider "google" {
  credentials = file("key.json")
  project     = "dockerbox-277106"
  region      = "us-west1"
}
```

Documentazione ufficiale: <https://www.terraform.io/docs/providers/google/index.html>

Creazione del network

Qui viene definita una nuova rete di nome “gitlab-network”.

```
// CREAZIONE DEL NETWORK
resource "google_compute_network" "vpc_network" {
  name = "gitlab-network"
}
```

Documentazione ufficiale: https://www.terraform.io/docs/providers/google/r/compute_network.html

Prenotazione dell’indirizzo IPv4 pubblico statico

Prenotiamo un indirizzo IPv4 di nome “gitlab-address” che collegheremo alla macchina virtuale.

```
// CREAZIONE IP PUBBLICO STATICO
resource "google_compute_address" "ip_address" {
  name = "gitlab-address"
}
```

Documentazione ufficiale: https://www.terraform.io/docs/providers/google/r/compute_address.html

Impostazione del firewall

Con questa parte di codice definiamo il nome del firewall “gitlab-app-firewall”. Lo assegniamo al network virtuale creato in precedenza e inseriamo alcune regole: in questo caso permettiamo il protocollo TCP sulle porte 80,22,5050,443.

Le porte in questo progetto vengono utilizzate per gli utilizzi qui sotto elencati.

- **Porta 80 (HTTP)**

Questa porta viene utilizzata per fare passare il traffico HTTP: l'utilizzo principale di questa porta è effettuare un reindirizzamento HTTPS del browser nel caso in cui un utente provi ad accedere attraverso HTTP (non sicuro).

- **Porta 443 (HTTPS)**

Questa è la porta HTTPS per poter accedere a GitLab dall'interfaccia web in modo sicuro e criptato.

- **Porta 22 (SSH)**

Mediante questa porta potremo collegarci mediante SSH per poter lanciare dei comandi alla nostra istanza Compute Engine.

- **Porta 5050 (Registry)**

Da questa porta passerà il traffico per poter caricare e scaricare le immagini dal Container Registry di GitLab.

- **Porta 8080 (M/Monit)**

Questa porta verrà utilizzata per il traffico HTTP di M/Monit e la sua interfaccia web per poter monitorare i processi in corso nel nostro sistema operativo.

```
// IMPOSTAZIONE DEL FIREWALL
resource "google_compute_firewall" "default" {
  name      = "gitlab-app-firewall"
  network   = "${google_compute_network.vpc_network.id}"
  allow {
    protocol = "tcp"
    ports    = ["80", "443", "22", "5050", "8080"]
  }
}
```

Documentazione ufficiale: https://www.terraform.io/docs/providers/google/r/compute_firewall.html

Estensione per generare numeri casuali

Per facilitare e dare un nome a tutte le istanze VM che vengono create ho scelto di generare un codice univoco.

```
// GENERATORE DI NUMERI RANDOM
resource "random_id" "instance_id" {
    byte_length = 2
}
```

Documentazione ufficiale: <https://www.terraform.io/docs/providers/random/r/id.html>

Creazione dell'istanza VM

Questa è l'istanza VM che andremo a creare nel Compute Engine. Ha un nome composto da “gitlab-vm-” al quale viene generato e aggiunto automaticamente un codice univoco che serve per poter ottenere un immediato riscontro, se la macchina è stata distrutta e ricreata dai rispettivi comandi Terraform (che analizzeremo più tardi).

Ho assegnato il tipo di macchina desiderato, qui si può trovare la lista completa: https://en.wikipedia.org/wiki/Google_Compute_Engine#Machine_type_comparison.

Ho selezionato la zona us-west-1-b. Da sottolineare che anche in Svizzera sono disponibili server. A scopo dimostrativo preferisco mantenere la mia infrastruttura in America, poiché Google Cloud non ha un'elevata disponibilità di indirizzi IPv4 da allocare agli account demo in Svizzera.

Qui di seguito la lista completa di tutte le zone:

<https://cloud.google.com/compute/docs/regions-zones>.

Ho creato un disco di 64GB (che bastano e avanzano per questo scopo dimostrativo), nel quale automaticamente verrà installato Ubuntu Server 18.04 LTS. Come si può osservare qui sotto, vado ad aggiungere al server la mia chiave pubblica dal mio computer locale per potermi collegare mediante SSH con chiave pubblica/chiave privata; niccolo.zuccolo è l'utente con il quale mi collegherò.

Si può osservare anche il collegamento al file “scripts/bootstrap.sh” che contiene lo script del primo avvio.

Nella sezione di network assegno all'istanza l'indirizzo IP pubblico riservato in precedenza e imposto che l'IP “privato” venga nattato sotto quello pubblico, ossia che l'indirizzo di rete venga tradotto.

```
// CREAZIONE DELLA ISTANZA O VM
resource "google_compute_instance" "default" {
  name          = "gitlab-vm-${random_id.instance_id.hex}"
  machine_type = "n1-standard-4"
  zone          = "us-west1-b"
  boot_disk {
    initialize_params {
      image = "ubuntu-1804-lts"
      size  = 64
    }
  }
  metadata = {
    ssh-keys = "niccolo.zuccolo:${file("~/ssh/id_ed25519.pub")}"
    startup-script = "${file("scripts/bootstrap.sh")}"
  }
  network_interface {
    network = "${google_compute_network.vpc_network.id}"
    access_config {
```

```
    nat_ip = "${google_compute_address.ip_address.address}"
}
}
}
```

Documentazione ufficiale: https://www.terraform.io/docs/providers/google/r/compute_instance.html

Stampa dell'indirizzo IP

Questa porzione di codice stampa l'indirizzo IP interno al momento dell'esecuzione del codice.

```
// STAMPA DELL'INDIRIZZO IP
output "instance_details" {
  value = "${google_compute_instance.default}"
}
```

Health check di rete

Aggiungo un “Health Check” alla mia infrastruttura che controlla che la VM risponda correttamente alle chiamate sulla porta 80.

```
// HEALTH CHECK TCP
resource "google_compute_health_check" "tcp-health-check" {
  name        = "tcp-health-check"
  description = "Health check via tcp"

  timeout_sec      = 1
  check_interval_sec = 1
  healthy_threshold = 4
  unhealthy_threshold = 5

  tcp_health_check {
    port_name        = "health-check-port"
    port_specification = "USE_NAMED_PORT"
    request          = "ARE YOU HEALTHY?"
    proxy_header     = "NONE"
    response         = "I AM HEALTHY"
  }
}
```

Documentazione ufficiale: https://www.terraform.io/docs/providers/google/r/compute_health_check.html

3.3 Esecuzione del file Terraform

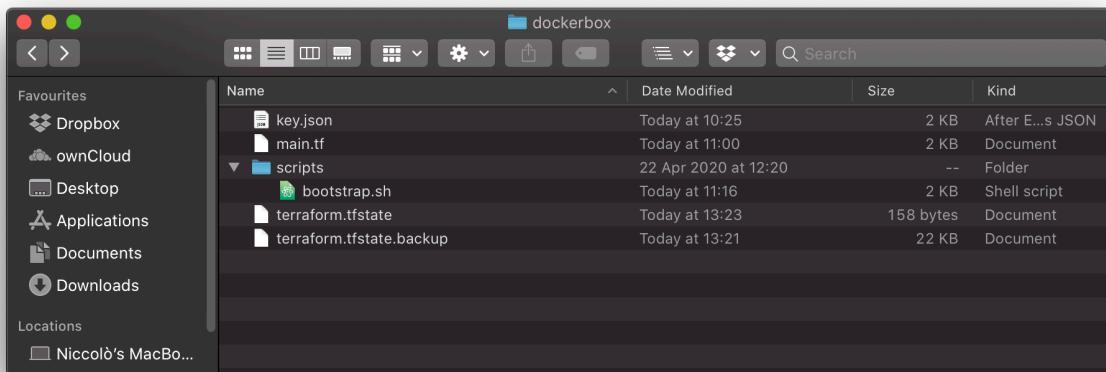
Terraform è un software molto versatile: una volta che viene definita l'infrastruttura come effettuata in precedenza nel file “main.tf”, si può procedere ad applicarla.

Prima di operare con i **successivi** bisogna aprire un terminale e accedere con il comando “cd” alla cartella “dockerbox” precedentemente creata.

Inizializzazione del progetto

```
terraform init
```

Una volta lanciato questo comando, Terraform inizializza la cartella creata in precedenza con dei file di lavoro e di backup per poter effettuare ad esempio la cancellazione del progetto in base a quanto applicato in precedenza.



Cartella del progetto dopo aver lanciato il comando di inizializzazione

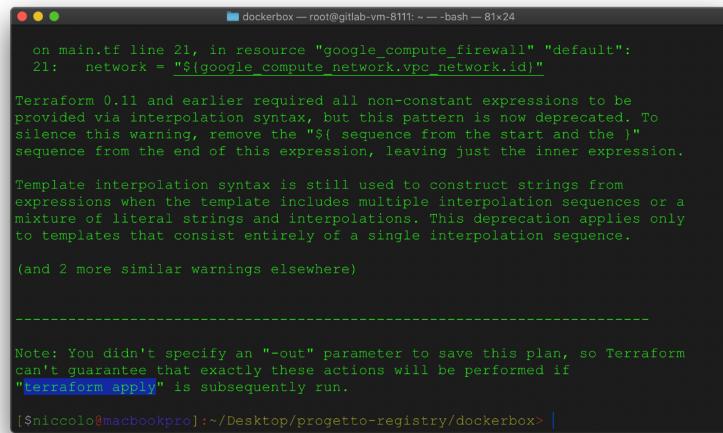
Come si può osservare vengono creati questi due file:

terraform.tfstate
terraform.tfstate.backup

Pianificazione del progetto

```
terraform plan
```

Questo comando permette di pianificare la nostra infrastruttura: quando viene lanciato, viene controllata la presenza di errori nei file configurati in precedenza.



The screenshot shows a terminal window titled "dockerbox" with the command "terraform plan" executed. The output indicates several deprecation warnings about template interpolation syntax, specifically regarding the use of \${...} sequences. It also notes that Terraform 0.11 and earlier required all non-constant expressions to be provided via interpolation syntax, but this pattern is now deprecated. To silence these warnings, users are advised to remove the "\${" sequence from the start and the ")" sequence from the end of the expression, leaving just the inner expression. The output also mentions that template interpolation syntax is still used to construct strings from expressions when the template includes multiple interpolation sequences or a mixture of literal strings and interpolations. This deprecation applies only to templates that consist entirely of a single interpolation sequence. There are also notes about specifying an "-out" parameter to save the plan and running "terraform apply" subsequently. The terminal prompt at the bottom is "\$niccolò@macbookpro:~/Desktop/progetto-registry/dockerbox>".

Esempio di esecuzione del comando di pianificazione

Applicazione del progetto

```
terraform apply
```

Questo comando applica l'infrastruttura precedentemente pianificata.

```

dockerbox — root@gitlab-vm-8111: ~ — terraform-provider-google_v3.20.0_x5 • terraform apply — 81x24
Terraform 0.11 and earlier required all non-constant expressions to be
provided via interpolation syntax, but this pattern is now deprecated. To
silence this warning, remove the "${" sequence from the start and the ")"
sequence from the end of this expression, leaving just the inner expression.

Template interpolation syntax is still used to construct strings from
expressions when the template includes multiple interpolation sequences or a
mixiture of literal strings and interpolations. This deprecation applies only
to templates that consist entirely of a single interpolation sequence.

(and 2 more similar warnings elsewhere)

Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: yes

random_id.instance_id: Creating...
random_id.instance_id: Creation complete after 0s [id=gno]
google_compute_address.ip_address: Creating...
google_compute_network.vpc_network: Creating...
google_compute_health_check.tcp-health-check: Creating...

```

Esempio di esecuzione del comando di applicazione del progetto al cloud reale

Una volta lanciato il comando, nella Google Cloud Console è possibile riscontrare quanto applicato. Qui un esempio per quanto riguarda la macchina virtuale:

Name	Zone	Recommendation	In use by	Internal IP	External IP	Connect
gitlab-vm-827a	us-west1-b			10.138.0.2 (nic0)	34.82.27.49	SSH

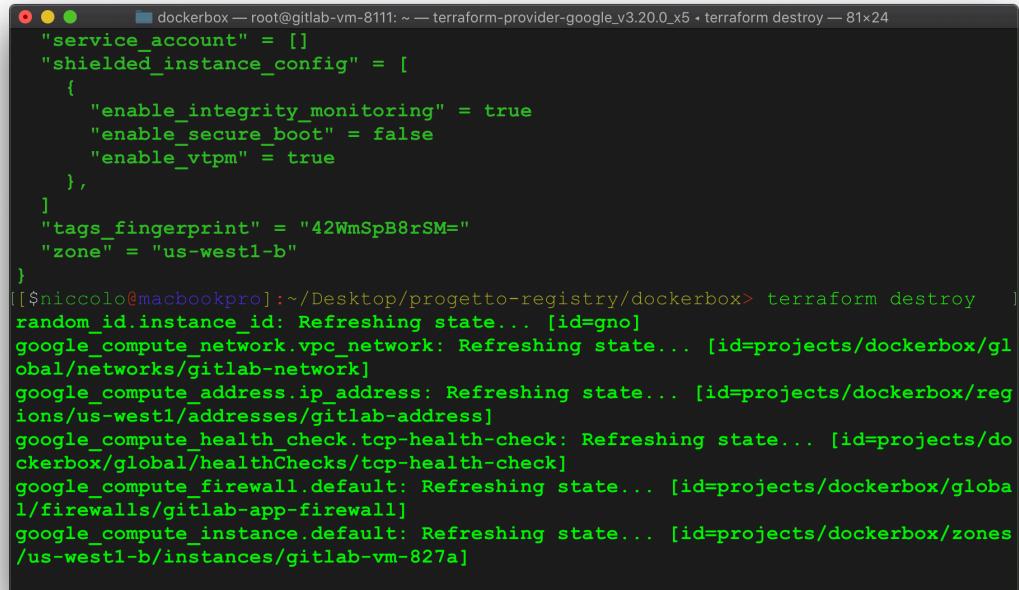
Visualizzazione delle macchine virtuali su GCP

Si prega di osservare che l'indirizzo IPv4 pubblico viene inizializzato alla creazione dell'istanza, pertanto ricreando l'istanza potrebbe variare e di conseguenza bisogna puntualmente aggiornare il DNS "dockerbox.ch".

Distruzione del progetto

```
terraform destroy
```

Per rimuovere completamente tutta l'infrastruttura precedentemente applicata su Google Cloud, basta lanciare il comando per distruggere permanentemente il tutto.



The screenshot shows a terminal window with a dark background and light-colored text. The command 'terraform destroy' is being run, and the output shows the destruction of various Google Cloud resources. The resources listed include a random ID instance, a VPC network, a compute address, a health check, a firewall, and a compute instance. Each resource is shown in a state of 'Refreshing state...' with its corresponding ID.

```
root@dockerbox: ~ - terraform destroy
"service_account" = []
"shielded_instance_config" = [
  {
    "enable_integrity_monitoring" = true
    "enable_secure_boot" = false
    "enable_vtpm" = true
  },
]
"tags_fingerprint" = "42WmSpB8rSM="
"zone" = "us-west1-b"
}
[[${niccolo@macbookpro}:~/Desktop/progetto-registry/dockerbox> terraform destroy
random_id.instance_id: Refreshing state... [id=gno]
google_compute_network.vpc_network: Refreshing state... [id=projects/dockerbox/global/networks/gitlab-network]
google_compute_address.ip_address: Refreshing state... [id=projects/dockerbox/regions/us-west1/addresses/gitlab-address]
google_compute_health_check.tcp-health-check: Refreshing state... [id=projects/dockerbox/global/healthChecks/tcp-health-check]
google_compute_firewall.default: Refreshing state... [id=projects/dockerbox/global/firewalls/gitlab-app-firewall]
google_compute_instance.default: Refreshing state... [id=projects/dockerbox/zones/us-west1-b/instances/gitlab-vm-827a]]]
```

Esempio di esecuzione del comando per distruggere l'infrastruttura

Anche qui aprendo la Console WEB di Google Cloud si può immediatamente riscontrare l'avvenuta rimozione dell'infrastruttura.

3.4 Configurazione del dominio

In questa parte della documentazione viene mostrato come configurare propriamente il dominio, in modo di poter accedere alle risorse mediante SSH o dall'interfaccia web da un dominio e non da un indirizzo IP.

Ho scelto di utilizzare il servizio Cloudflare poiché oltre a essere più reattivo nel cambiamento delle impostazioni DNS, è notevolmente più avanzato, ha interessanti funzioni come ad esempio il monitoraggio del traffico e la reportistica di statistiche.

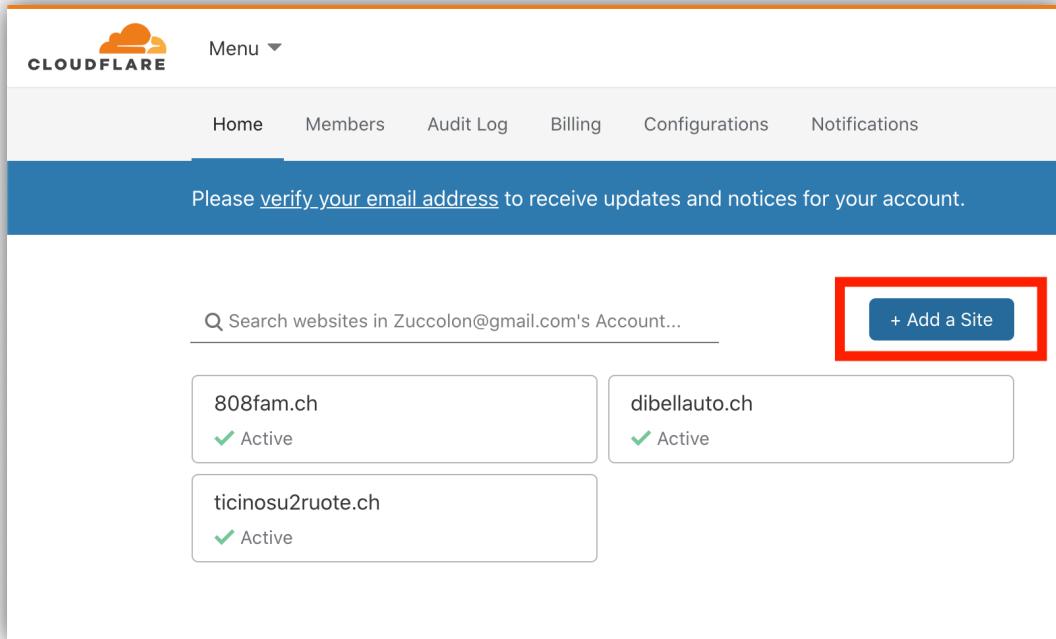
La premessa per lo svolgimento è di essere già possessori di un dominio. In questa guida vedremo inoltre come è facile configurare e gestire i domini con il servizio “Cloudflare”. In questo caso ho acquistato il dominio “dockerbox.ch” da “Hostpoint”.

1. Per prima cosa bisogna aprire il pannello amministrativo di Hostpoint e impostare il Nameserver come indicato: così facendo il Nameserver verrà gestito esclusivamente da Cloudflare e non dallo stesso Hostpoint.

The screenshot shows the 'Domain' page for 'dockerbox.ch'. At the top, there are three buttons: 'ZURÜCK ZUR DOMAIN ÜBERSICHT', 'WHOIS', and 'DNS EDITOR ÖFFNEN' (which is highlighted). Below these, there are two main sections: 'Domain Status' and 'Nameserver'. The 'Nameserver' section contains three input fields, each with a red 'X' icon and a placeholder: 'art.ns.cloudflare.com', 'rachel.ns.cloudflare.com', and 'Nameserver'. A large red rectangle highlights these three fields. At the bottom of the 'Nameserver' section are three buttons: 'SPEICHERN' (highlighted), 'ZURÜCKSETZEN', and 'HOSTPOINT NAMESERVER VERWENDEN'.

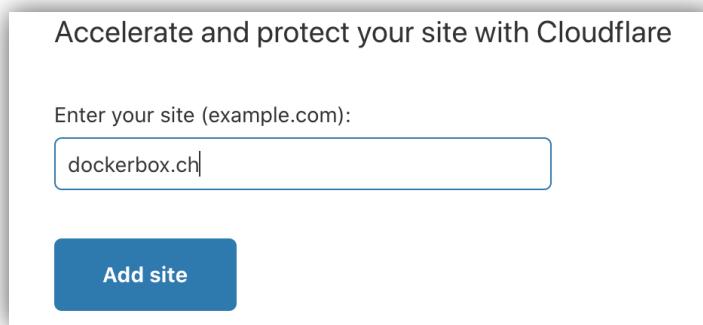
Pannello amministrativo Hostpoint

2. Recarsi quindi su [cloudflare.com](https://www.cloudflare.com), effettuare l'accesso con il proprio account e nella schermata principale aggiungere un nuovo sito internet.



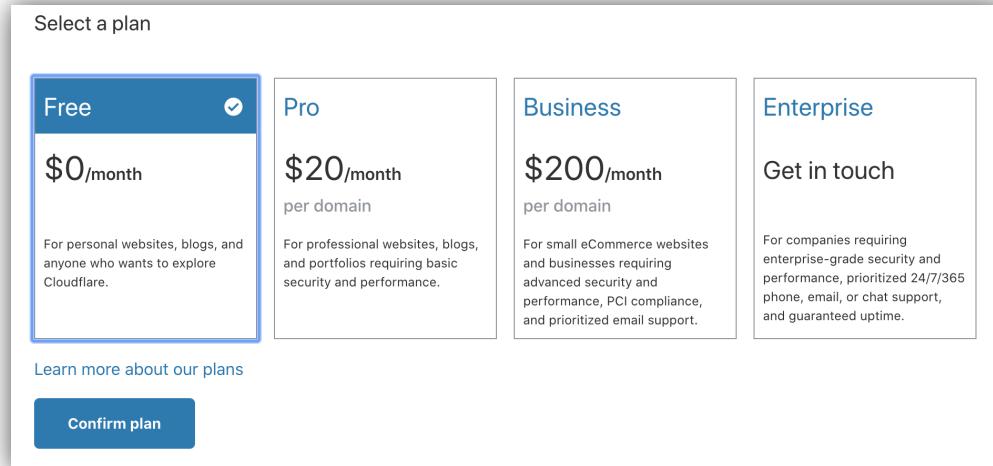
Schermata di "Menu" di Cloudflare

3. Una volta fatto ciò comparirà un processo di configurazione con diverse schermate, quindi va inserito il proprio dominio.



Schermata di configurazione del dominio

4. Selezionare il piano desiderato: in questo caso il piano gratuito, poiché vi si trova tutto il necessario per poter proseguire.



Piani Cloudflare

5. In questa ultima schermata inserire il record DNS come segue: tipo A (IPv4), @ (senza sottodominio), IP del server, TTL automatico e “DNS Only” e in seguito cliccare su “Save”.

DNS management for dockerbox.ch

+ Add record Search DNS Records Advanced

dockerbox.ch points to 34.82.83.152.

Type	Name	IPv4 address	TTL	Proxy status
A	@	34.82.83.152	Auto	DNS only

Cancel Save

Type	Name	Content	TTL	Proxy status

Step di configurazione del DNS su cloudflare.com

Se in futuro si dovessero cambiare le impostazioni DNS, lo si può fare dalla schermata di configurazione del DNS su cloudflare.com.

A few more steps are required to complete your setup.

- ✓ Add an A, AAAA, or CNAME record for **www** so that [www.dockerbox.ch](#) will resolve.
- ✓ Add an MX record for your **root domain** so that mail can reach [@dockerbox.ch](#) addresses.

DNS management for dockerbox.ch				
Type	Name	Content	TTL	Proxy status
A	dockerbox.ch	34.82.27.49	Auto	DNS only Delete

Schermata di configurazione DNS in Cloudflare

3.5 Installazione e configurazione di GitLab (Registry)

In questa fase ho scritto nello script di avvio l'installazione GitLab: tutti i blocchi di codice sono stati appesi al file scripts/bootstrap.sh.

Inizialmente questo codice esegue l'aggiornamento del repository e del pacchetto

```
#!/bin/bash

# AGGIORNAMENTO DEI SOFTWARE
sudo apt update -y
sudo apt dist-upgrade -y
```

Qui avviene l'installazione effettiva di GitLab con i suoi software specifici e le dipendenze.

```
# INSTALLAZIONE DI GITLAB
sudo apt install curl openssh-server ca-certificates -y
curl -sS https://packages.gitlab.com/install/repositories/gitlab/gitlab-
ce/script.deb.sh | sudo bash
sudo apt install gitlab-ce -y
```

In questa parte viene modificato il file di configurazione “/etc/gitlab/gitlab.rb”, viene definito un dominio che verrà utilizzato “dockerbox.ch” e viene abilitato il certificato gratuito di letsencrypt. Inoltre viene impostato un comando che rinnova automaticamente i certificati SSL per l'HTTPS. Questi certificati di letsencrypt sono validi due settimane, ma verranno automaticamente rinnovati.

```
# MODIFICA E CONFIGURAZIONE DEL FILE gitlab.rb
sudo sed -i "s/http:\\\\gitlab.example.com/https:\\\\dockerbox.ch/g"
/etc/gitlab/gitlab.rb
sudo sed -i "s/# letsencrypt['enable']= nil/letsencrypt['enable']= true/g"
/etc/gitlab/gitlab.rb
sudo sed -i "s/# letsencrypt['contact_emails']=
\\[]/letsencrypt['contact_emails']= \\['zuccolon@cscs.ch']/g" /etc/gitlab/gitlab.rb
sudo sed -i "s/# letsencrypt['group']= 'root'/letsencrypt['group']= 'root'/g"
/etc/gitlab/gitlab.rb
sudo sed -i "s/# letsencrypt['key_size']= 2048/letsencrypt['key_size']= 2048/g"
/etc/gitlab/gitlab.rb
sudo sed -i "s/# letsencrypt['owner']= 'root'/letsencrypt['owner']= 'root'/g"
/etc/gitlab/gitlab.rb
sudo sed -i "s/# letsencrypt['wwwroot']=
'\\\\var\\\\opt\\\\gitlab\\\\nginx\\\\www'/letsencrypt['wwwroot']=
'\\\\var\\\\opt\\\\gitlab\\\\nginx\\\\www'/g" /etc/gitlab/gitlab.rb
sudo sed -i "s/# letsencrypt['auto_renew']= true/letsencrypt['auto_renew']=
true/g" /etc/gitlab/gitlab.rb
sudo sed -i "s/# letsencrypt['auto_renew_hour']=
0/letsencrypt['auto_renew_hour']= 2/g" /etc/gitlab/gitlab.rb
```

```
sudo sed -i "s/\# letsencrypt\[\'auto_renew_day_of_month'\] =\n\"*\v4\"/letsencrypt\[\'auto_renew_day_of_month'\] = \"*\v4\"/g"\n/etc/gitlab/gitlab.rb
```

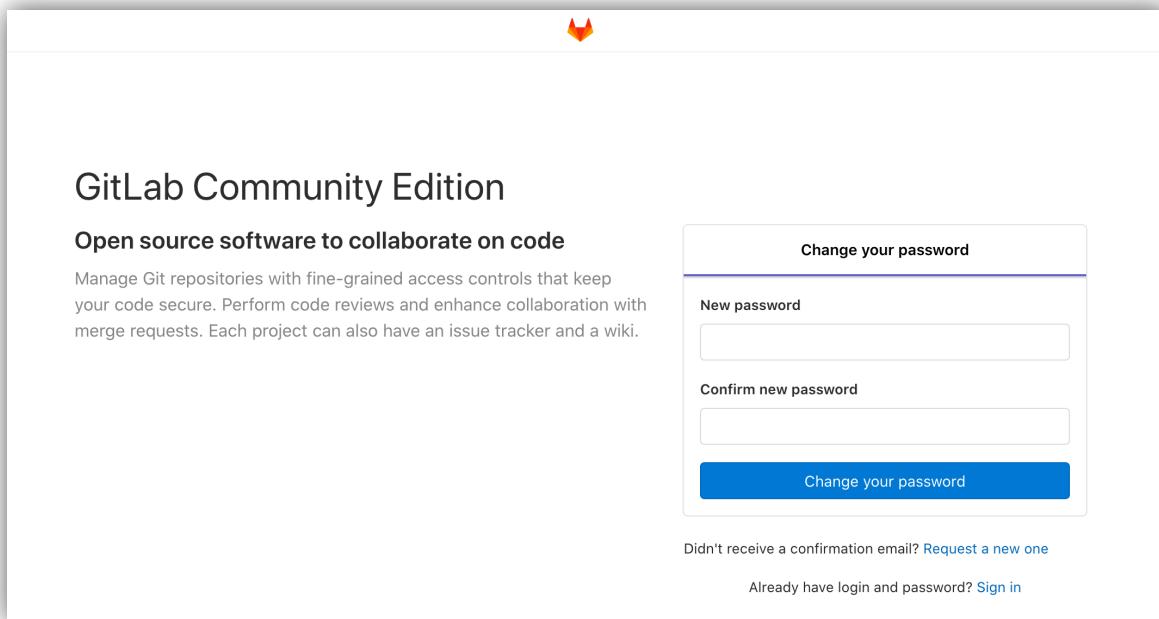
A ogni modifica del file “gitlab.rb” è necessario lanciare questo comando per riconfigurare il servizio.

```
# RICONFIGURAZIONE DI GITLAB\nsudo gitlab-ctl reconfigure
```

Per applicare effettivamente lo script e ultimare l’installazione si prega di distruggere l’infrastruttura, ripianificarla e riapplicarla. Le istruzioni sono disponibili al punto 3.3 della guida.

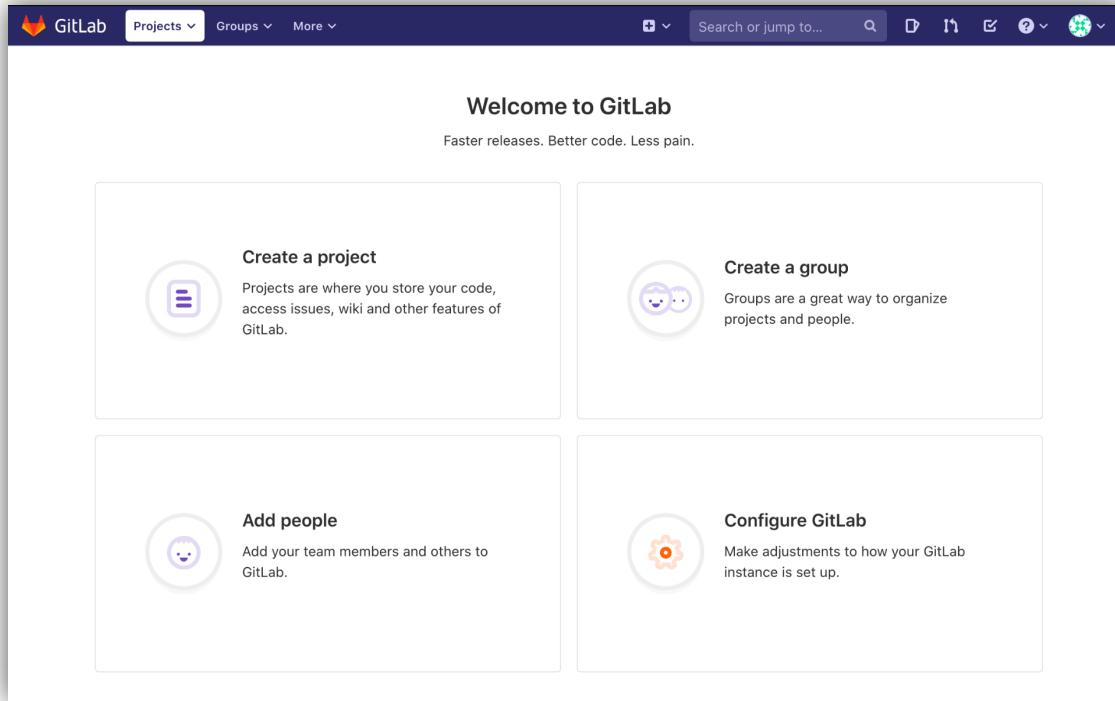
Questo script viene eseguito automaticamente subito dopo aver lanciato il comando “terraform apply”. Siccome contiene del codice che imposta automaticamente il dominio e va a richiedere i certificati SSL, è necessario impostare prima il DNS (vedi punto 3.4).

Una volta configurato il DNS e riconfigurato il tutto, possiamo recarci su “dockerbox.ch” mediante il browser che preferiamo; nella prima schermata sarà necessario impostare la password dell’utente “root”.



Schermata di primo accesso a GitLab su dockerbox.ch

Dopo aver impostato la password di “root”, possiamo effettuare il login.



Schermata principale GitLab

3.6 Utilizzo di GitLab e comandi per operare con Registry

In questa parte della guida viene mostrato come operare con GitLab e utilizzare il servizio di Registry integrato in esso.

Si assume che sul computer client Docker, per poter effettuare i comandi “push” e “pull” mostrati in seguito, sia già installato.

Creare un progetto

Per prima cosa bisogna concepire un progetto su GitLab: a questo scopo andiamo nella schermata di creazione, nel caso specifico creo un progetto privato chiamato “Moon”:

New project

A project is where you house your files (repository), plan your work (issues), and publish your documentation (wiki), [among other things](#).

All features are enabled for blank projects, from templates, or when importing, but you can disable them afterward in the project settings.

Information about additional Pages templates and how to install them can be found in our [Pages getting started guide](#).

Tip: You can also create a project from the command line. [Show command](#)

Blank project Create from template Import project

Project name: moon

Project URL: https://dockerbox.ch/ root Project slug: moon

Want to house several dependent projects under the same namespace? [Create a group](#).

Project description (optional)

Description format

Visibility Level

Private Project access must be granted explicitly to each user. If this project is part of a group, access will be granted to members of the group.

Internal The project can be accessed by any logged in user.

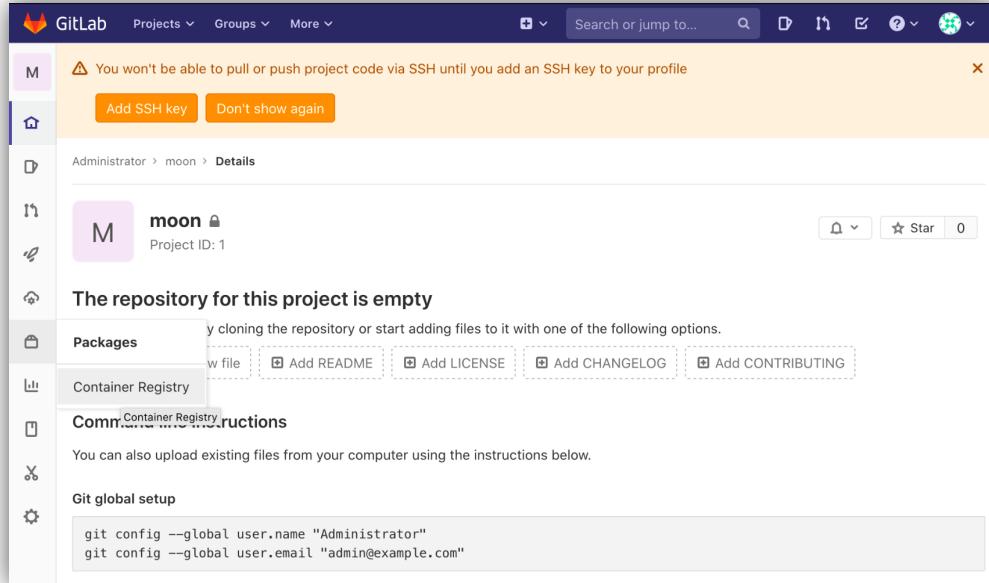
Public The project can be accessed without any authentication.

Initialize repository with a README Allows you to immediately clone this project's repository. Skip this if you plan to push up an existing repository.

Create project Cancel

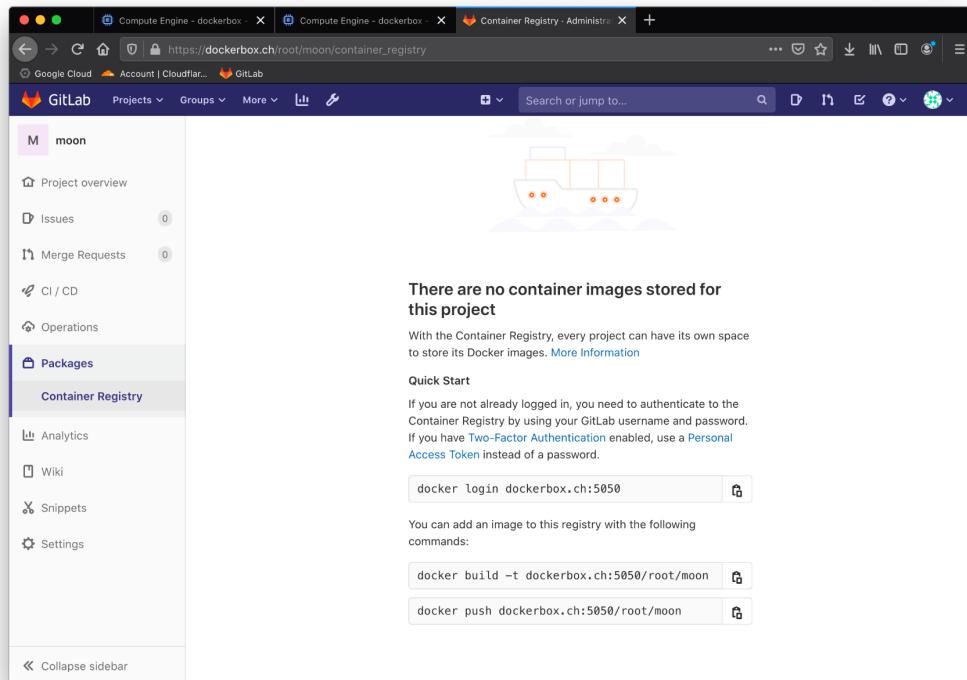
Schermata di creazione di un progetto

Dopo che è stato creato il progetto, possiamo recarci nella sezione adibita al Registry, quindi nel menu laterale vado sotto “Packages” > “Container Registry”.



Schermata principale del progetto

Come si può osservare nella schermata “Container Registry” vi sono utili informazioni che serviranno al punto successivo.



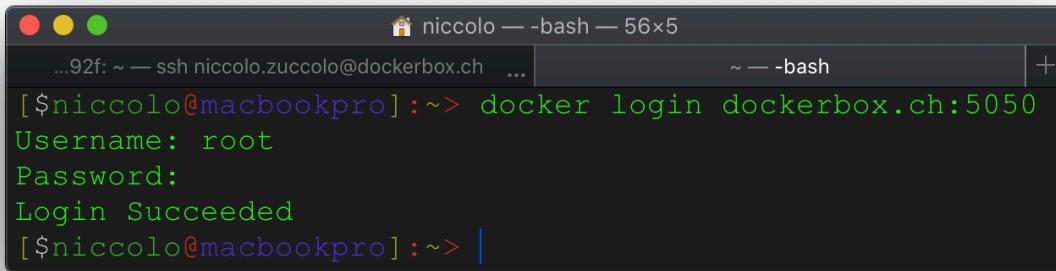
Sezione “Container Registry” del progetto “Moon”

Effettuare il login con Docker

Per poter utilizzare il Registry dobbiamo effettuare il login, quindi con il terminale del nostro client (in questo caso Mac), lanciare il seguente comando:

```
docker login dockerbox.ch:5050
```

Inserire quindi l'utente che ha accesso al progetto. A scopo dimostrativo utilizziamo l'utente root.



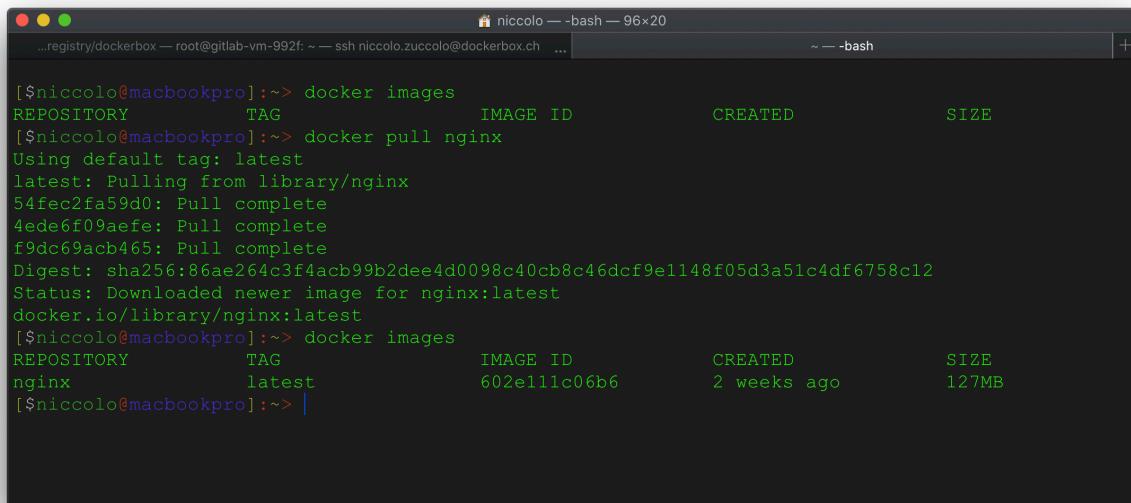
```
niccolo — -bash — 56x5
...92f: ~ — ssh niccolozuccolo@dockerbox.ch ...
[ $niccolozuccolo@macbookpro ] :~> docker login dockerbox.ch:5050
Username: root
Password:
Login Succeeded
[ $niccolozuccolo@macbookpro ] :~>
```

Esecuzione del comando docker login in terminale Mac

Scaricare un'immagine

In questo caso a scopo dimostrativo preleviamo e copiamo nel computer locale un'immagine dal DockerHub ufficiale mediante il seguente comando:

```
docker pull nginx
```



```
niccolo — -bash — 96x20
...registry/dockerbox — root@gitlab-vm-992f: ~ — ssh niccolozuccolo@dockerbox.ch ...
[ $niccolozuccolo@macbookpro ] :~> docker images
REPOSITORY          TAG           IMAGE ID      CREATED        SIZE
[ $niccolozuccolo@macbookpro ] :~> docker pull nginx
Using default tag: latest
latest: Pulling from library/nginx
54fec2fa59d0: Pull complete
4ede6f09aefe: Pull complete
f9dc69acb465: Pull complete
Digest: sha256:86ae264c3f4acb99b2dee4d0098c40cb8c46dcf9e1148f05d3a51c4df6758c12
Status: Downloaded newer image for nginx:latest
docker.io/library/nginx:latest
[ $niccolozuccolo@macbookpro ] :~> docker images
REPOSITORY          TAG           IMAGE ID      CREATED        SIZE
nginx              latest        602e111c06b6    2 weeks ago   127MB
[ $niccolozuccolo@macbookpro ] :~>
```

Esecuzione del comando docker pull

Taggare un'immagine

Per poter caricare la nostra immagine nel Docker Registry appena allestito abbiamo bisogno di taggarla, in sostanza dobbiamo rinominarla in modo che essa contenga l'URL del Docker Registry. In questo caso andremo a rinominare l'immagine da “nginx” a “moon:alpha”, ma inserendo prima le informazioni del nostro Registry. “Moon” è il nome dell'immagine e “alpha” è il tag che vi aggiungiamo. Per effettuare questa operazione lanciare il seguente comando:

```
docker tag nginx dockerbox.ch:5050/root/moon:alpha
```

Come si può osservare con il comando “docker images” l'immagine è stata rinominata con successo.

```
niccolo — -bash — 97x13
...-registry/dockerbox — root@gitlab-vm-992f: ~ — ssh niccolo.zuccolo@dockerbox.ch ...
[$niccolo@macbookpro]:~> docker tag nginx dockerbox.ch:5050/root/moon:alpha
[$niccolo@macbookpro]:~> docker images
REPOSITORY          TAG      IMAGE ID      CREATED        SIZE
nginx              latest    602e111c06b6   2 weeks ago   127MB
dockerbox.ch:5050/root/moon  alpha     602e111c06b6   2 weeks ago   127MB
[$niccolo@macbookpro]:~>
```

Esecuzione del comando “docker tag”

Caricare nel Registry l'immagine

Per caricare l'immagine con i relativi tag nel docker-hub bisogna eseguire il comando “docker push” seguito dal nome completo dell'immagine, così come segue:

```
docker push dockerbox.ch:5050/root/moon:alpha
```

Se l'operazione è andata a buon fine viene mostrato un “digest” (o ash) e sul computer client viene mostrato quanto segue:

```
niccolo — -bash — 97x7
...-registry/dockerbox — root@gitlab-vm-992f: ~ — ssh niccolo.zuccolo@dockerbox.ch ...
[$niccolo@macbookpro]:~> docker push dockerbox.ch:5050/root/moon:alpha
The push refers to repository [dockerbox.ch:5050/root/moon]
b3003aac411c: Pushed
216cf33c0a28: Pushed
c2adabaaedb: Pushed
alpha: digest: sha256:cccef6d6bdea671c394956e24b0d0c44cd82dbe83f543a47fdc790fad4a48422 size: 948
[$niccolo@macbookpro]:~>
```

Esecuzione del comando “docker push”

Come si può osservare nel Container Registry di GitLab nel progetto “Moon” su dockerbox.ch è comparsa l’immagine con il relativo tag.

The screenshot shows the GitLab interface for the 'moon' project. The sidebar on the left lists various project sections: Project overview, Issues (0), Merge Requests (0), CI / CD, Operations, Packages, Container Registry (selected), Analytics, Wiki, Snippets, and Settings. The main content area is titled 'Container Registry'. It displays a message: 'Retention policy has been Enabled' with a note about the retention and expiration policy running in 6 days. Below this is a 'Quick Start' button. A dark callout box highlights the URL 'dockerbox.ch:5050/root/moon'. Underneath, there is a list entry for 'root/moon' with a trash icon to its right.

Container Registry dopo il primo push delle immagini Docker

Per eliminare l’immagine basta premere sull’icona del cestino sulla destra.

3.7 Installazione del server mail

Prima di poter implementare i sistemi di monitoraggio del nostro sistema è molto importante avere e implementare il server mail in modo che di conseguenza possano venir spedite le mail di notifica dai relativi software di monitoraggio.

Quello che ci occorre è un server per la posta in uscita, non in entrata; quindi per il server mail ho scelto di utilizzare postfix come tramite per poter spedire la posta mediante i server di Gmail.

Per semplificare l'installazione ho creato un apposito script che è reperibile nella sezione allegati della guida. Più avanti in questo stesso capitolo commento e spiego il funzionamento di questo script.

Inizialmente lo script inizializza già i parametri da passare negli step di installazione di postfix in modo che avvenga in modo silenzioso senza mostrare niente all'utente. Come si può osservare vengono installate anche le mailutils, ovvero utility che ci permetteranno di spedire le mail mediante comandi da terminale a scopo di test.

```
# INSTALLO POSTFIX
echo "postfix postfix/mailname string dockerbox.ch" | debconf-set-selections
echo "postfix postfix/main_mailer_type string 'Internet Site'" | debconf-set-
selections
sudo apt install -y postfix
sudo apt install libsasl2-modules mailutils -y
```

In questa fase viene creato un file di configurazione chiamato “sasl_passwd”: in questo file dobbiamo definire l'username e password dell'account Google. Per facilitare l'utente e per questioni di privacy ho preferito che l'utente inserisca il suo username e la sua password in modo da evitare di condividere informazioni riservate.

```
touch /etc/postfix/sasl/sasl_passwd
echo "[smtp.gmail.com]:587 $GMAILUSER@gmail.com:$GMAILPASS" >
/etc/postfix/sasl/sasl_passwd
cat /etc/postfix/sasl/sasl_passwd
```

In questi comandi diamo l'ordine per formattare in un database le informazioni precedentemente configurate nel file “sasl_password”. Per questioni di sicurezza imposto e restringo gli accessi a questi file contenenti informazioni riservate al solo utente root.

```
sudo postmap /etc/postfix/sasl/sasl_passwd
sudo chown root:root /etc/postfix/sasl/sasl_passwd /etc/postfix/sasl/sasl_passwd.db
sudo chmod 0600 /etc/postfix/sasl/sasl_passwd /etc/postfix/sasl/sasl_passwd.db
```

In questa fase vi è l'impostazione di postfix per potersi collegare ai server di Gmail.

```
sed -i '/myhostname/d' /etc/postfix/main.cf
sed -i '/relayhost/d' /etc/postfix/main.cf
echo "myhostname = $CURRENTDOMAIN" >> /etc/postfix/main.cf
echo "relayhost = [smtp.gmail.com]:587" >> /etc/postfix/main.cf
echo "smtp_sasl_auth_enable = yes" >> /etc/postfix/main.cf
echo "smtp_sasl_security_options = noanonymous" >> /etc/postfix/main.cf
echo "smtp_sasl_password_maps = hash:/etc/postfix/sasl/sasl_passwd" >>
/etc/postfix/main.cf
echo "smtp_tls_security_level = encrypt" >> /etc/postfix/main.cf
echo "smtp_tls_CAfile = /etc/ssl/certs/ca-certificates.crt" >> /etc/postfix/main.cf
cat /etc/postfix/main.cf
```

Utilizzare questo comando per riavviare e abilitare il nostro server di posta elettronica postfix.

```
sudo systemctl restart postfix && sudo systemctl enable postfix
```

Prima di inviare la mail assicurarsi di aver abilitato il permesso alle applicazioni di terze parti (o meno sicure) di poter accedere ai servizi Google e di aver disabilitato la doppia autenticazione con CAPTCHA all'account Google.

Per abilitare l'accesso alle app meno sicure cliccare qui:

<https://www.google.com/settings/security/lesssecureapps>

Per consentire l'accesso al proprio account Google cliccare qui:

<https://accounts.google.com/DisplayUnlockCaptcha>

Per spedire una mail velocemente e con un singolo comando lanciare quanto segue:

corpo email: "Ciao, questa mail proviene dalla istanza postfix appena configurata e funzionante";

soggetto email: Postfix configurato;

destinatario: zuccolon@gmail.com.

```
echo "Ciao, questa mail proviene dalla istanza postfix appena configurata e
funzionante" | mail -s "Postfix configurato" zuccolon@gmail.com
```

Ecco qui di seguito quello che possiamo aspettarci nella nostra casella di posta elettronica:



Messaggio di test nella casella di posta elettronica

3.8 Installazione di Ossec (HIDS)

Per poter monitorare il corretto funzionamento del sistema scelgo di implementare il software “Ossec”, un software molto completo che permette di monitorare in modo specifico l’integrità dei file presenti nel sistema operativo.

Ossec principalmente si occupa del nostro sistema operativo ai seguenti livelli:

- **File Integrity checking**

Questa funzione di Ossec permette di controllare costantemente l’integrità dei file mediante una comparazione dei checksum.

- **Log Monitoring**

Permette di monitorare e controlla tutto ciò che avviene nel sistema operativo, come le connessioni SSH, la promozione e l’accesso dell’utente root ecc.

- **Rootkit detection**

Monitora il comportamento e gli accessi dei software installati nel sistema operativo.

Per installare Ossec ho scritto il seguente script per automatizzare tutto. Si prega di notare che in questa documentazione ci sono blocchi di codice commentati; per un corretto funzionamento appendere allo script tutti i blocchi. Per ulteriori informazioni è possibile ottenere lo script osservando il capitolo “Allegati”.

Per prima cosa installo tutti i software di cui Ossec necessita. Ossec è un software che va compilato e pertanto è necessario avere un compilatore (make).

```
# INSTALLAZIONE DELLE DIPENDENZE OSSEC
sudo apt install -y \
make \
build-essential \
libz-dev \
libssl-dev \
libpcre2-dev \
libevent-dev \
inotify-tools
```

Per installare Ossec bisogna scaricare un file tar.gz, estrarlo, accedervi e lanciare lo script di installazione presente in esso.

```
# INSTALLAZIONE DI OSSEC
wget -P /root https://github.com/ossec/ossec-hids/archive/3.6.0.tar.gz
tar -xvzf /root/3.6.0.tar.gz
cd /root/ossec-hids-3.6.0
sh /root/ossec-hids-3.6.0/install.sh
```

Verrà lanciata una configurazione guidata all'installer di Ossec; dopo aver selezionato una lingua, selezionare l'installazione di tipo "local".

Abilitare le notifiche mediante email, scegliere un'email alla quale spedire le notifiche e gli avvisi e scegliere localhost come server mail.

```
3.1- Do you want e-mail notification? (y/n) [y]: y
- What's your e-mail address? zuccolon@cscs.ch

- We found your SMTP server as: mc3.ethz.ch.
- Do you want to use it? (y/n) [y]: n

- What's your SMTP server ip/host? localhost
```

Configurazione dell'email nel wizard di Ossec

Per verificare che Ossec è stato installato correttamente possiamo controllare la versione lanciando il seguente comando.

```
# CONTROLLO VERSIONE INSTALLATA
cat /etc/ossec-init.conf
```

Avviamo il servizio Ossec con il seguente comando:

```
# AVVIO SERVIZIO
sudo /var/ossec/bin/ossec-control start
```

Le impostazioni di Ossec si trovano nel file di configurazione situato a "/var/ossec/etc/ossec.conf". Ciò che vado a modificare è il tempo espresso in secondi dell'intervallo desiderato fra un check di sistema e l'altro; a questo scopo eseguo il seguente comando, in tal modo passeremo dall'avere un check ogni 79200 secondi ad averne uno ogni 600 secondi (10 minuti).

```
sudo sed -i "s/79200/600/g" /var/ossec/etc/ossec.conf
```

Con il seguente comando vado a modificare il file local_rules.xml locato in "/var/ossec/rules/local_rules.xml". Questo file contiene già molte regole: noi andiamo a modificarlo nel seguente modo aggiungendo ulteriori regole:

```
# AGGIUNGO ULTERIORI REGOLE
sed -i "/<\!-- EOF -->/d" /var/ossec/rules/local_rules.xml

cat <<EOT >> /var/ossec/rules/local_rules.xml
<group name="syslog,sshd,authentication_success,>
  <rule id="105715" level="3">
    <if_sid>5715</if_sid>
    <options>alert_by_email</options>
    <description>ssh login</description>
  </rule>
</group>
```

```
<group name="sys log,authentication_success,">
  <rule id="105303" level="3">
    <if_sid>5303</if_sid>
    <options>alert_by_email</options>
    <description>root activity</description>
  </rule>
</group>
<group name="snap_loop_devices_alerts_suppression,">
  <rule id="105305" level="0">
    <if_sid>531</if_sid>
    <regex>'df -P':\s+/dev/loop\d+\s+\d+\s+\d+\s+0\s+100%\s+/snap/\w+/\d+</regex>
    <description>Ignore full snap loop devices in Ubuntu 18.04</description>
  </rule>
</group>

<!-- EOF -->
EOT
```

- Quando avverrà una connessione SSH al nostro server riceveremo una notifica via email.
- Se un utente si autentica come root, riceveremo una notifica.
- L'ultima regola serve a correggere l'invio involontario di email di un falso allarme generato dall'utilizzazione del disco (“Snap load devices”).

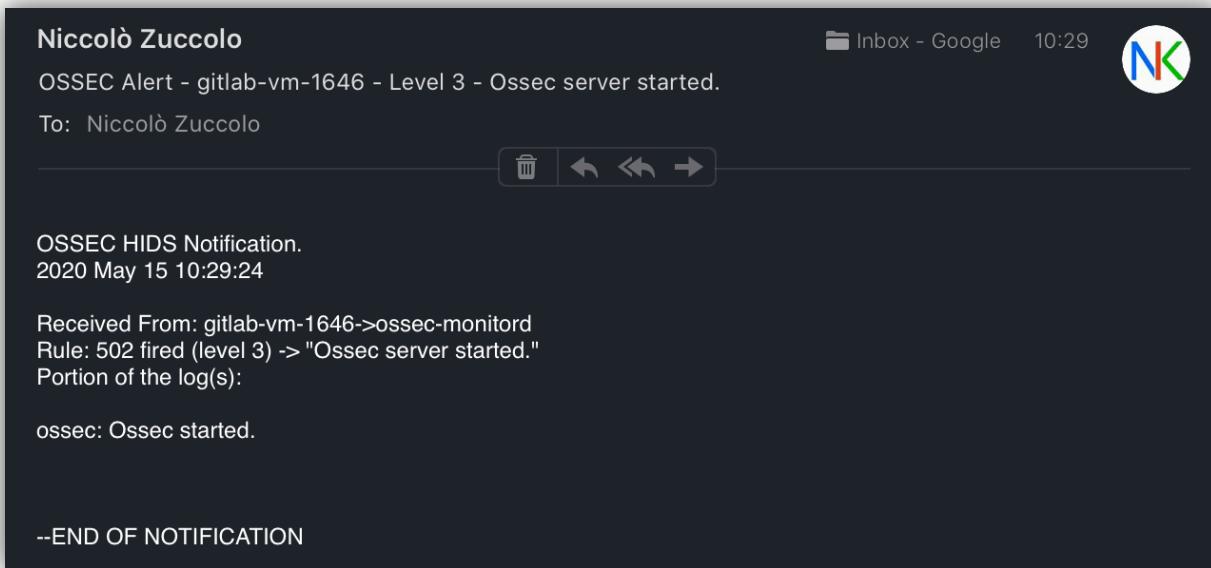
Ossec utilizza un suo file per risolvere i domini, pertanto per ovviare a problemi dovuti all'errata risoluzione dei domini andiamo a copiare il file resolv.conf di sistema nella cartella di Ossec.

```
# COPIA RESOLV CONF
cp /etc/resolv.conf /var/ossec/etc/resolv.conf
```

Dopo aver copiato questo file e a ogni modifica delle regole che apponiamo al file dobbiamo effettuare un riavvio del servizio.

```
systemctl restart ossec
```

Una volta riavviato correttamente il servizio riceviamo la seguente email di conferma.



Email di conferma dell'avvenuto riavvio del servizio Ossec.

Ogni regola ha un livello di importanza definito da un numero: 1 è il livello più basso di importanza dal quale si va poi a salire.

Andando a modificare il file “/var/ossec/etc/ossec.conf” possiamo definire nell'apposita sezione quanto deve essere importante la regola adottata per ricevere una notifica via email.

```
<alerts>
  <log_alert_level>1</log_alert_level>
  <email_alert_level>7</email_alert_level>
</alerts>
```

Sezione “allerte” nel file di configurazione di Ossec

3.9 Installazione di M/Monit (monitoraggio dei processi)

M/Monit è uno strumento che, mediante parametri inseriti nel file di configurazione, permette di andare ad analizzare in modo mirato alcuni processi. In questo caso andremo a controllare se il nostro sito internet risponde correttamente sulla porta 80 e 443; inoltre vogliamo sapere se il processo SSHD dovesse presentare malfunzionamenti. Sia per quanto riguarda il check HTTP, HTTPS e SSHD desideriamo essere puntualmente informati via email.

Prima di installare M/Monit dobbiamo avere un mail server per l'invio delle email, quello precedentemente installato mediante postfix va benissimo.

Per installare M/Monit basta lanciare i seguenti comandi:

```
apt-get update -y  
apt-get install monit -y
```

Una volta che abbiamo installato M/Monit è necessario avviare il servizio e abilitarlo in modo che si avvii automaticamente.

```
systemctl start monit  
systemctl enable monit
```

3.10 Implementazione di M/Monit

M/Monit è un software molto avanzato, ma al contempo semplificato: in questa parte della guida spiego come imposto il file di configurazione e come funziona.

Principi e basi del file di configurazione

Il principio di M/Monit è abbastanza semplice: per impostarlo di fatto basta modificare il file di configurazione che si trova in /etc/monit/monitrc. Tutte le configurazioni desiderate vanno appese in questo file, così come le regole che vi impostiamo.

Ogni volta che modifichiamo questo file di configurazione dobbiamo applicarlo, ma prima è sempre preferibile controllarlo con il seguente comando in modo da verificare che è stato scritto nel modo corretto:

```
sudo monit -t
```

Dopo aver controllato che quello che abbiamo scritto nel file di configurazione è corretto, possiamo procedere a riavviare il servizio in modo da applicare le modifiche apportate.

```
systemctl restart monit
```

Di seguito ecco come ho impostato il mio file di configurazione: prima di appendere questi blocchi ho cancellato l'intero file e l'ho riscritto nel seguente modo.

Impostazione delle directory

Nel file di configurazione inizio con il definire le varie directory di base, inoltre definisco ogni quanto tempo (espresso in secondi) devo effettuare un check.

```
set daemon 60 # check services at 1-minute intervals
set log /var/log/monit.log
set idfile /var/lib/monit/id
set statefile /var/lib/monit/state
set eventqueue
  basedir /var/lib/monit/events # set the base directory where events will be stored
  slots 100                      # optionally limit the queue size
include /etc/monit/conf.d/*
include /etc/monit/conf-enabled/*
```

In base al file di configurazione di default

Impostazione dell'interfaccia web

In questa parte del file di configurazione abilito il servizio web che sarà fruibile dalla porta 8081. Per potervi accedere sarà richiesta la seguente autenticazione: admin:monit.

```
set httpd port 8081 and
  allow admin:monit
```

In base al file di configurazione di default

Impostazione delle notifiche via email

Di seguito imposto il server email e l'indirizzo email al quale desidero ricevere puntualmente le notifiche in caso di malfunzionamento di uno dei servizi controllati dalle regole. In questo caso imposto il server email locale e la mia email di lavoro come casella di ricezione delle notifiche.

```
set mailserver 127.0.0.1
set alert zuccolon@cscs.com
```

In base al file di configurazione di default

Aggiunta di regole

Le regole da adottare sono le quattro che seguono:

1. Questa regola mi permette di monitorare il sistema operativo che viene principalmente controllato se si è in presenza di un sovrautilizzo della CPU e della RAM.

```
check system $HOST
if memory usage > 80% for 4 cycles then alert
if swap usage > 20% for 4 cycles then alert
# Test the user part of CPU usage
if cpu usage (user) > 80% for 2 cycles then alert
# Test the system part of CPU usage
if cpu usage (system) > 20% for 2 cycles then alert
# Test the i/o wait part of CPU usage
if cpu usage (wait) > 80% for 2 cycles then alert
```

Documentazione di Alibaba Cloud - https://www.alibabacloud.com/blog/how-to-install-monit-monitoring-tool-on-ubuntu-16-04_594339

2. Con questa regola monitoreremo il processo e la connessione a SSH: se di fatto qualcosa dovesse andare storto per cinque cicli, gli amministratori predefiniti riceveranno una notifica.

```
check process sshd with pidfile /var/run/sshd.pid
start program "/etc/init.d/ssh start"
stop program "/etc/init.d/ssh stop"
if failed port 22 protocol ssh then restart
if 5 restarts within 5 cycles then timeout
```

Documentazione ufficiale - <https://mmonit.com/monit/documentation/monit.html#PID-TEST>

3. Questa regola controlla invece che ci sia risposta al sito internet nella quale è istanziata l'infrastruttura GitLab: è un semplice check HTTP sulla porta 80.

```
check host dockerbox.ch with address dockerbox.ch
if failed port 80 protocol http for 2 cycles then alert
```

4. Questa regola va a richiedere la schermata di accesso di GitLab con un protocollo HTTPS attraverso la porta 443.

```
check host dockerbox
with address dockerbox.ch
```

```
if failed
  host dockerbox.ch
  port 443
  type TCPSSL
  protocol https
then alert
```

Documentazione ufficiale - <https://mmonit.com/monit/documentation/monit.html#SSL-OPTIONS>

Controllo del file e riavvio del servizio

Per applicare le modifiche apportate controllo che non vi siano errori di sintassi nel file con il seguente comando:

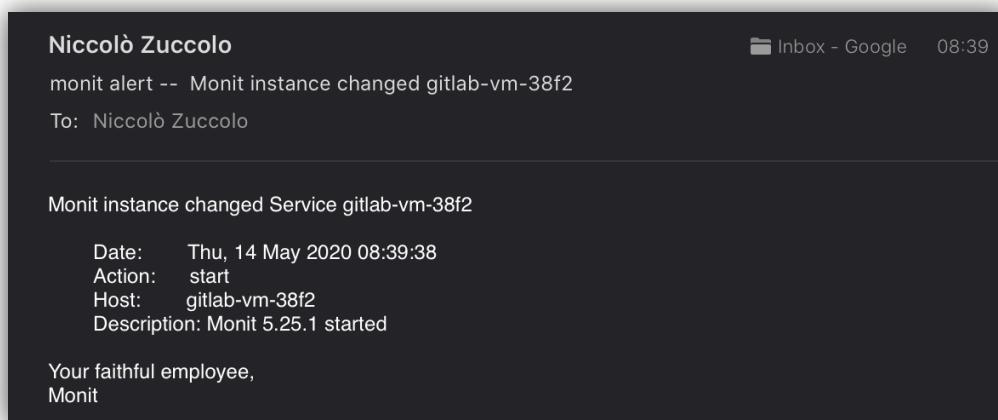
```
sudo monit -t
```

Riavvio il servizio in questo modo:

```
systemctl restart monit
```

Notifica email

Dopo aver lanciato il comando di riavvio del servizio M/Monit ricevo un'email di conferma.



Email di notifica dopo un riavvio del servizio M/Monit

Interfaccia web

Per verificare la situazione attuale dello stato dei nostri check mi reco sul sito <http://dockerbox.ch:8081> e effettuo l'accesso con le credenziali impostate in precedenza. Si prega di osservare che al momento dello screenshot i servizi web segnalano un errore, in quanto non ancora installati al momento del test.

The screenshot shows the Monit Service Manager interface. At the top, it says "Monit is running on gitlab-vm-38f2 and monitoring:". Below this, there are three main sections: System, Process, and Host.

System		Status	Load		CPU	Memory	Swap
gitlab-vm-38f2		OK	[0.00]	[0.00]	0.0%us, 0.0%sy, 0.0%wa	1.7% [253.3 MB]	0.0% [0 B]

Process	Status	Uptime	CPU Total	Memory Total	Read	Write
sshd	OK	1h 11m	0.0%	0.2% [33.2 MB]	0 B/s	0 B/s

Host	Status	Protocol(s)
dockerbox.ssl	Connection failed	[HTTP] at port 443
dockerbox.ch	Connection failed	[HTTP] at port 80

Copyright © 2001-2017 [Tideslash](#). All rights reserved. [Monit web site](#) | [Monit Wiki](#) | [M/Monit](#)

Interfaccia Web di M/Monit

Un altro esempio di email di notifica di errore da connessione ai servizi web definiti nelle regole in precedenza è il seguente:

Connection failed Service [dockerbox.ch](#)

Date: Thu, 14 May 2020 09:27:37
Action: alert
Host: gitlab-vm-38f2
Description: failed protocol test [HTTP] at [[dockerbox.ch](#)]:80 [TCP/IP] -- Connection refused

Your faithful employee,
Monit

Email di notifica M/Monit

3.11 Implementazione delle Snapshot in Google Cloud

In questo capitolo viene spiegato come creare un programma di Snapshot e i salvataggi della VM periodici e automatici.

Prima di ciò effettuiamo manualmente uno snapshot e un restore in modo da comprenderne il funzionamento.

Creazione di un singolo snapshot

Innanzitutto andiamo nella Google Cloud Console dal sito

<https://console.cloud.google.com>, dopodiché selezioniamo il progetto di GCP, apriamo lo strumento “Compute Engine” e nella barra laterale premiamo su “Snapshots”.

Name	Location	Snapshot size	Creation time
snapshot-1	us	799.26 MB	May 14, 2020,

Snapshots di Compute Engine su GCP

Per creare uno snapshot manualmente clicchiamo su “Create Snapshot”: comparirà una schermata. Inseriamo il nome desiderato e selezioniamo il disco di sorgente che intendiamo salvare. Così facendo creiamo una copia istantanea del disco, il sistema rimane acceso e quindi non è necessario spegnere la nostra istanza. Clicchiamo su ‘Create’.

The screenshot shows the 'Create a snapshot' dialog in the Google Cloud Compute Engine interface. On the left, a sidebar lists various Compute Engine services: VM instances, Instance groups, Instance templates, Sole-tenant nodes, Machine images, Disks, **Snapshots** (which is selected), Images, TPUs, Migrate for Compute Engine, Committed use discounts, Metadata, Health checks, Zones, Network endpoint groups, Operations, Security scans, and Marketplace.

The main form has the following fields:

- Name ***: la-mia-snapshot
- Description**: Snapshot della istanza pulita con mmonit
- Source disk ***: gitlab-vm-38f2
- Location**:
 - Multi-regional
 - Regional Select location: us (United States)
- Labels**: + ADD LABEL
- Encryption type**:
 - Google managed
 - Your free trial credit will be used for this snapshot. [GCP Free Tier](#)

At the bottom are two buttons: **CREATE** (highlighted in blue) and **CANCEL**. Below the buttons is the note: Equivalent [REST](#) or [command line](#).

Schermata di creazione di uno snapshot

Come possiamo osservare lo snapshot è presente nella nostra libreria.

The screenshot shows a table with the following data:

	Name	Location	Snapshot size	Creation time
<input type="checkbox"/>	la-mia-snapshot	us	12.89 MB	May 14, 2020, 10:04:24 AM
<input type="checkbox"/>	snapshot-1	us	799.26 MB	May 14, 2020, 9:41:18 AM

Snapshots in GCP CE

Dopo aver creato questo snapshot, a scopo di test preparo una cartella nella home dell'utente root e creo dei file.

```
[root@gitlab-vm-38f2:~# pwd
/root
[root@gitlab-vm-38f2:~# mkdir aftersnapshot
[root@gitlab-vm-38f2:~# cd aftersnapshot/
[root@gitlab-vm-38f2:~/aftersnapshot# touch ciaosnap.txt
[root@gitlab-vm-38f2:~/aftersnapshot# ls
ciaosnap.txt
root@gitlab-vm-38f2:~/aftersnapshot#
```

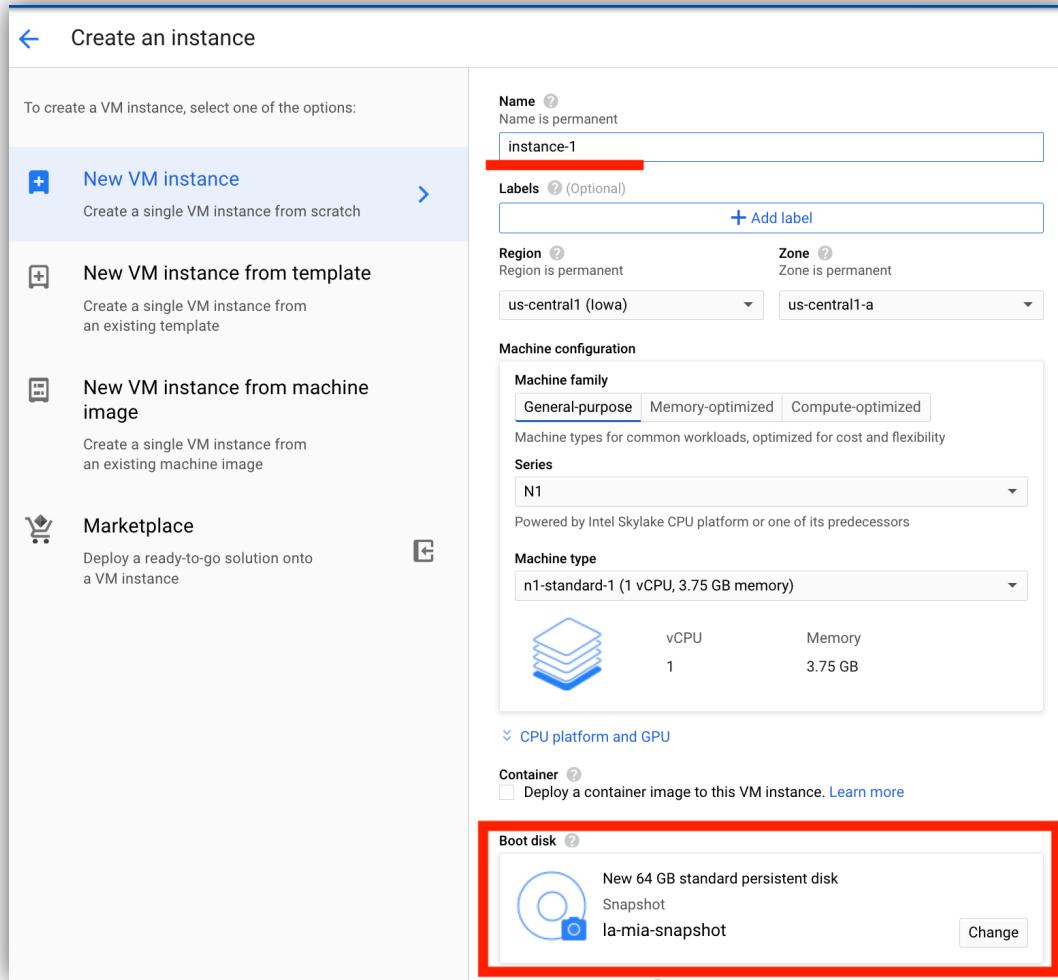
Creazione di cartelle nell'istanza dopo aver effettuato uno snapshot

Per effettuare il restore bisogna aprire il dettaglio dello snapshot in questo modo:

The screenshot shows the 'Snapshot details' page for 'la-mia-snapshot'. The 'CREATE INSTANCE' button is highlighted with a red box.

Schermata di dettaglio di uno snapshot

Dopodiché inizierà un processo di creazione di un'istanza in Compute Engine, con la differenza che il disco di boot sarà generato e ripreso dallo snapshot come segue:



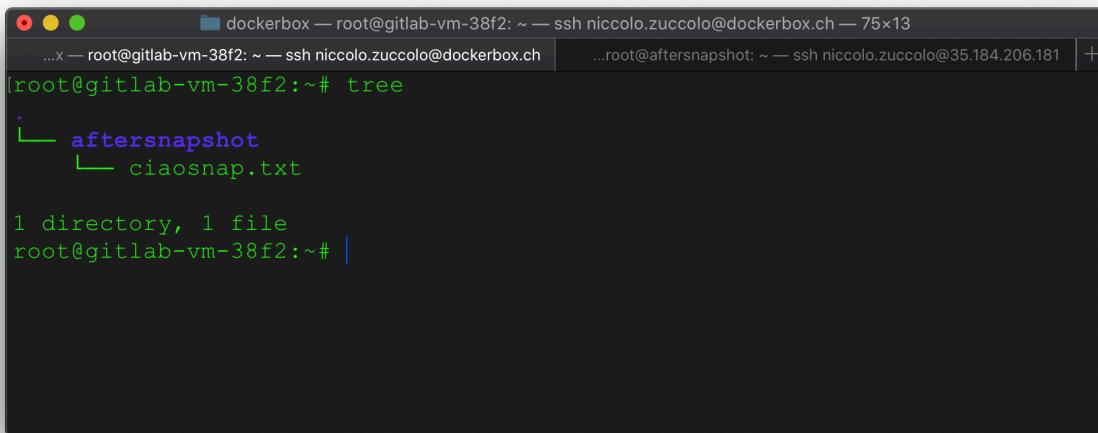
Creazione di una Istanza Google Cloud da Snapshot

Impostare l'istanza nel modo desiderato e avviarla.

Name	Zone	Recommendation	In use by	Internal IP	External IP	Connect
<input checked="" type="checkbox"/> aftersnapshot	us-central1-a			10.128.0.2 (nic0)	35.184.206.181	SSH <input type="button" value="⋮"/>
<input checked="" type="checkbox"/> gitlab-vm-38f2	us-west1-b			10.138.0.2 (nic0)	34.83.39.165	SSH <input type="button" value="⋮"/>

Per verificare che non ci siano file creati dopo lo snapshot mi collego in SSH all'istanza appena creata e in seguito mi collego e osservo la differenza dei contenuti delle cartelle create.

Immediatamente dopo aver creato lo snapshot con la cartella /root pulita abbiamo creato i seguenti file:

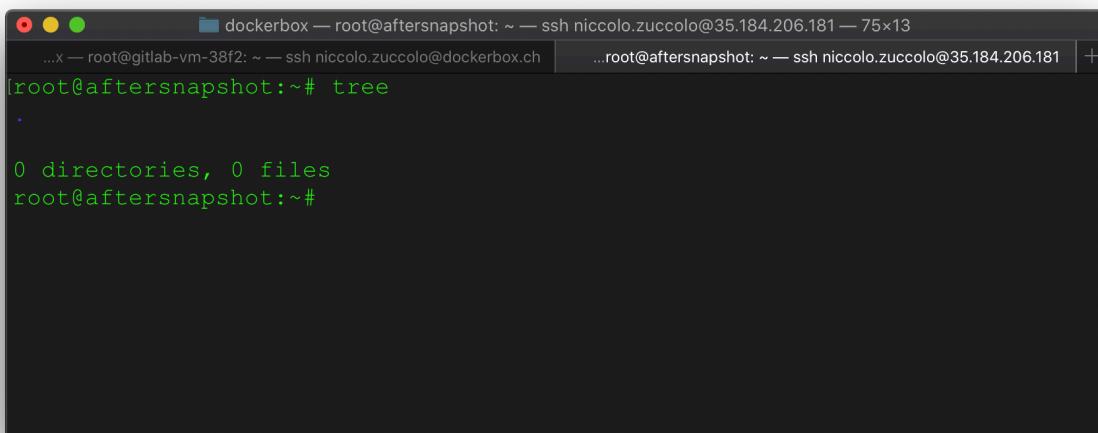


```
[root@gitlab-vm-38f2:~# tree
.
└─ aftersnapshot
    └─ ciaosnap.txt

1 directory, 1 file
root@gitlab-vm-38f2:~# ]
```

VM Originale

Dopo aver effettuato il restore della macchina virtuale possiamo osservare come lo snapshot abbia funzionato: i file creati dopo lo snapshot non sono più presenti.



```
[root@aftersnapshot:~# tree
.

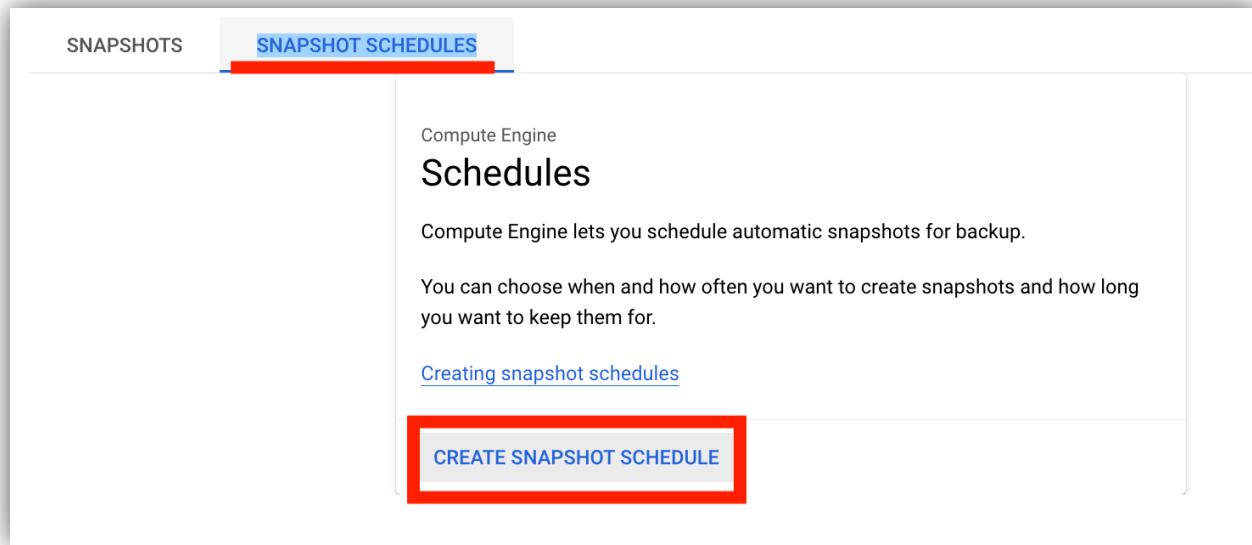
0 directories, 0 files
root@aftersnapshot:~# ]
```

VM dopo aver eseguito il restore

Pianificazione degli snapshot

Per non dover effettuare uno snapshot manualmente ogni volta, Google permette con uno strumento la pianificazione degli snapshot stessi in modo che vengano eseguiti autonomamente snapshot per ogni intervallo di tempo.

Per pianificarli apriamo quindi lo strumento apposito, sempre reperibile presso la Google Cloud Console, andiamo sotto la tab “Snapshot Schedules” e clicchiamo “Create Snapshot Schedule”.



Pannello amministrativo degli snapshot GCP CE

Dopo aver cliccato su “Create Snapshot Schedule” si aprirà una pagina di configurazione nella quale inseriremo i seguenti parametri:

Name *
backup-giornaliero
Lowercase letters, numbers, hyphens allowed

Description
Questo snapshot schedule prevede di effettuare un backup giornaliero al disco di boot della istanza git

Region
us-central1

Snapshot location ?
 Multi-regional
 Regional
Select location
us-central1 (Iowa)

There may be a network transfer fee if you choose to store this snapshot in a location different than the source disk. [Learn more](#)

Schedule frequency
Daily

Start time (UTC)
1:00 AM - 2:00 AM

Autodelete snapshots after *
30 days

Deletion rule ?
After you delete the disk that uses this schedule:
 Keep snapshots
 Delete snapshots older than 30 days

Integrate volume shadow copy service ?
 Enable VSS

Snapshot labels ?
+ ADD LABEL

You can't edit a schedule after you create it

CREATE **CANCEL**

Equivalent [REST](#) or [command line](#)

Nome

In questo campo scriviamo il nome della nostra regola.

Descrizione

Per essere facilmente individuata in futuro in questo campo descriviamo come funziona la regola.

Schedule Frequency

Qui definiamo quanto frequentemente desideriamo che avvengano gli snapshot automatici, siccome sono snapshot incrementali possiamo inserire periodi brevi.

Start Time

Qui inseriamo l'orario di inizio dello snapshot; è consigliabile effettuare questa operazione di notte in modo che nessuno possa percepire la mancanza di risposta da parte dell'istanza.

Autodelete Snapshots After

Siccome non vogliamo accumulare gli snapshot all'infinito aumentando sensibilmente i costi di gestione, possiamo impostarne la cancellazione automatica. Qui sono impostati 30 giorni.

Schermata di configurazione alla creazione di uno “snapshot schedule”

Una volta creato lo scheduling dello snapshot esso si presenterà in questo modo.

		SNAPSHOT SCHEDULES				
		Name	Region	Schedule frequency (UTC)	Autodelete snapshots after	In use by
Filter table						
<input type="checkbox"/>	giornaliero	us-central1	Every day between 1:00 AM and 2:00 AM		30 days	

Snapshots schedulati

4. Protocollo di test

4.1 Verifica dell'integrità del progetto

Lo scopo di questa parte della documentazione è volto a verificare che tutti i componenti del progetto (GitLab, Postfix, Ossec, M/Monit) siano stati installati e in perfetto funzionamento.

GitLab

Per verificare che GitLab sia installato possiamo eseguire il seguente comando:

```
sudo apt-cache policy gitlab-ce | grep Installed
```

<https://docs.gitlab.com/omnibus/update/#updating-community-edition-to-enterprise-edition>

File di configurazione: /etc/gitlab/gitlab.rb

Cartella dei log: /var/log/gitlab

Postfix

Per verificare che Postfix sia installato possiamo eseguire il seguente comando:

```
service postfix status
```

File di configurazione account: /etc/postfix/sasl/sasl_passwd

File di configurazione server email: /etc/postfix/main.cf

File di log: /var/log/mail.log

Ossec

Per verificare che Ossec sia installato possiamo eseguire il seguente comando:

```
cat /etc/ossec-init.conf
```

File di configurazione: /var/ossec/etc/ossec.conf

File delle regole: /var/ossec/rules/local_rules.xml

File di log: /var/ossec/logs/ossec.log

M/Monit

Per verificare che M/Monit sia installato possiamo eseguire il seguente comando:

```
service monit status
```

File di configurazione e regole: /etc/monit/monitrc

4.2 Test credenziali e accesso a GitLab

Questo test è pensato per controllare i permessi degli utenti che a loro volta andranno a utilizzare GitLab come Container Registry.

In questo test la situazione iniziale sarà la seguente:

Progetto: “mouse”.

Utente “root”: “Maintainer”.

Utente “zuccolon”: non ha accesso.

Come primo test proviamo a effettuare un “docker login” come root sul computer locale e a pushare delle immagini nel Container Registry.

Ci aspettiamo che questa operazione si possa eseguire.

Attuazione

Come si può osservare effettuo il comando docker login e accedo al Docker Container Registry mediante l’utente root.

```
[[ $niccolo@macbookpro] :~> docker login dockerbox.ch:5050
Username: root
[Password:
Login Succeeded
```

In questa fase scarico un’immagine nginx, la taggo e la carico nel Container Registry.

```
[[ $niccolo@macbookpro] :~> docker pull nginx
Using default tag: latest
latest: Pulling from library/nginx
Digest: sha256:f85c2305909e5881feec31efcf2a3449e5abd9b55a34522343c4b55ca2c947bb
Status: Image is up to date for nginx:latest
docker.io/library/nginx:latest
[[ $niccolo@macbookpro] :~> docker tag nginx dockerbox.ch:5050/root/mouse:alpha
[[ $niccolo@macbookpro] :~> docker push dockerbox.ch:5050/root/mouse:alpha
The push refers to repository [dockerbox.ch:5050/root/mouse]
b3003aac411c: Pushed
216cf33c0a28: Pushed
c2adabaecedb: Pushed
alpha: digest: sha256:cccef6d6bdea671c394956e24b0d0c44cd82dbe83f543a47fdc790fadea4842
2 size: 948
[$niccolo@macbookpro] :~> ]]
```

Risultati

L'interfaccia web di GitLab permette ora di osservare l'immagine caricata con successo. I risultati riscontrati sono in linea con quelli attesi.

The screenshot shows the GitLab interface for the 'mouse' project. On the left, there's a sidebar with links: Project overview, Issues (0), Merge Requests (0), CI / CD, Operations, Packages (selected), Container Registry, and Analytics. The main area shows the 'Container Registry' settings for the 'root/mouse' repository. A message at the top states: 'Retention policy has been Enabled. The retention and expiration policy for this Container Registry has been enabled and will run in 6 days. For more information visit the documentation'. Below this is a 'Container Registry' section with a 'Quick Start' button. The repository 'root/mouse' is listed with a trash icon next to it.

Container Registry in GitLab

Un ulteriore test consiste nel provare a eseguire la stessa operazione, ma con un utente che non ha diritto a caricare le risorse del progetto “mouse” e a usufruirne. In questo caso ho creato un utente “zuccolon”, ma non gli ho dato i permessi di accesso al progetto.

Dopo aver effettuato il login, scaricato e taggato l'immagine, proviamo ora a caricarla nel nostro Registry. Come auspicato ciò che si può osservare è che il caricamento dell'immagine non avviene e in risposta otteniamo un messaggio specifico che ne spiega il motivo.

```
[[ $niccolò@macbookpro ] : ~ > docker push dockerbox.ch:5050/root/mouse:alpha
The push refers to repository [dockerbox.ch:5050/root/mouse]
b3003aac411c: Preparing
216cf33c0a28: Preparing
c2adabaecedb: Preparing
denied: requested access to the resource is denied
```

4.3 Simulazione di avaria per riscontro da Ossec

Questo semplice test mira a controllare che, in caso di eventi involontari come ad esempio un riavvio del sistema operativo o l'avvento di una attività sospetta, l'imprevisto venga puntualmente segnalato via email all'amministratore del sistema.

Per effettuare questo test utilizzeremo la nostra infrastruttura e proveremo a fare un semplice riavvio dell'istanza. Il risultato atteso è un'email di notifica di questo evento nella nostra casella di posta elettronica.

Attuazione

In questo caso mi reco nella Google Cloud Console, seleziono la macchina che ospita la mia infrastruttura ed effettuo un reset (ovvero riavvio forzato) della macchina virtuale.

Name	Zone	Recommendation	In use by	Internal IP	External IP	Connect
gitlab-vm-061b	us-west1-b			10.138.0.2 (nic0)	34.83.39.165	SSH

Istanze VM in Google Cloud Console

Risultati

Come si può osservare dalla seguente immagine (in questo caso dopo l'avvenuto riavvio) le notifiche email da parte del software Ossec sono funzionanti.

OSSEC Alert - gitlab-vm-061b - Level 2 - Unknown problem somewhere

OSSEC HIDS <zuccolon@gmail.com>
Zuccolo Niccolo Ernesto
Friday, 15 May 2020 at 15:33
[Show Details](#)

OSSEC HIDS Notification.
2020 May 15 15:32:35

Received From: gitlab-vm-061b->/var/log/syslog
Rule: 1002 fired (level 2) -> "Unknown problem somewhere in the system."
Portion of the log(s):

```
May 15 15:32:35 gitlab-vm-061b startup-script: INFO startup-script: *  
templatesymlink[Create a rack_attack.rb and create a symlink to Rails root] action  
create
```

Email generata da Ossec

4.4 Verifica del funzionamento dei trigger M/Monit

Questo test è molto simile al test effettuato per Ossec: serve per verificare il funzionamento delle allerte di M/Monit. Per effettuare questo test utilizzeremo l'infrastruttura creata in precedenza.

Uno stato “sano” dell’infrastruttura come ci viene riportato dall’interfaccia web M/Monit è il seguente:

The screenshot shows the Monit Service Manager web interface. At the top, it displays the URL "Monit: gitlab-vm-061b" and the version "Monit 5.25.1". Below this, the main title is "Monit Service Manager" with the sub-instruction "Monit is running on gitlab-vm-061b and monitoring:". The interface is divided into several sections:

- System:** Shows the host "gitlab-vm-061b" with a status of "OK". It provides metrics for Load (0.03, 0.15, 0.28), CPU (2.7%us, 0.4%sy, 0.0%wa), Memory (15.5% [2.3 GB]), and Swap (0.0% [0 B]).
- Process:** Shows the process "sshd" with a status of "OK". It provides Uptime (16m) and CPU Total (0.0%).
- Host:** Shows two hosts: "dockerbox.ch" and "dockerbox", both with a status of "OK". It indicates monitoring protocols: "[HTTP] at port 80" for dockerbox.ch and "[HTTP] at port 443" for dockerbox.

At the bottom, there is a copyright notice: "Copyright © 2001-2017 [Tildeslash](#). All rights reserved. [Monit web site](#) | [Monit Wiki](#) | [M/Monit](#)".

Interfaccia web di M/Monit

Attuazione

Il trigger che desideriamo far scattare è quello del check HTTP e HTTPS: per provarli ci basterà cambiare le impostazioni del DNS server, quindi apro Cloudflare e nella sezione DNS inserisco un indirizzo IP volutamente sbagliato. M/Monit proverà a fare un check HTTP al dominio dockerbox.ch. Il risultato atteso è che non trovi nulla e che mandi una email di avviso nel tempo stabilito dell’intervallo tra i check.

Apro quindi il mio gestore DNS e imposto un indirizzo IP “errato”.

The screenshot shows a DNS management interface with a toolbar at the top containing icons for Overview, Analytics, DNS, SSL/TLS, Firewall, Access, Speed, Caching, Workers, Page Rules, Network, Traffic, Stream, Custom P..., Apps, and Scrape S... . Below the toolbar is a message box stating: "A few more steps are required to complete your setup." with a "Hide" link. It lists two items: "✓ Add an A, AAAA, or CNAME record for www so that [www.dockerbox.ch](#) will resolve." and "✓ Add an MX record for your **root domain** so that mail can reach [@dockerbox.ch](#) addresses." Below this is a section titled "DNS management for **dockerbox.ch**" with a table of records:

Type	Name	Content	TTL	Proxy status
A	dockerbox.ch	8.8.8.8	Auto	DNS only Delete

Schermata di gestione del DNS

Risultati

Qualche minuto dopo aver inserito un indirizzo IP sbagliato nella configurazione DNS, riscontreremo le seguenti email di notifica da parte di M/Monit.

The screenshot shows an email from Monit with the subject "monit alert -- Connection failed dockerbox". The email is addressed to "zuccolon@gmail.com <zuccolon@gmail.com>" and "Zuccolo Niccolo Ernesto". It was sent on "Friday, 15 May 2020 at 15:52" and includes a "Show Details" link. The body of the email contains the following text:

```

Connection failed Service dockerbox

Date: Fri, 15 May 2020 15:51:57
Action: alert
Host: gitlab-vm-061b
Description: failed protocol test [HTTP] at [dockerbox.ch]:443 [TCP/IP TLS] --
Connection timed out

Your faithful employee,
Monit

```

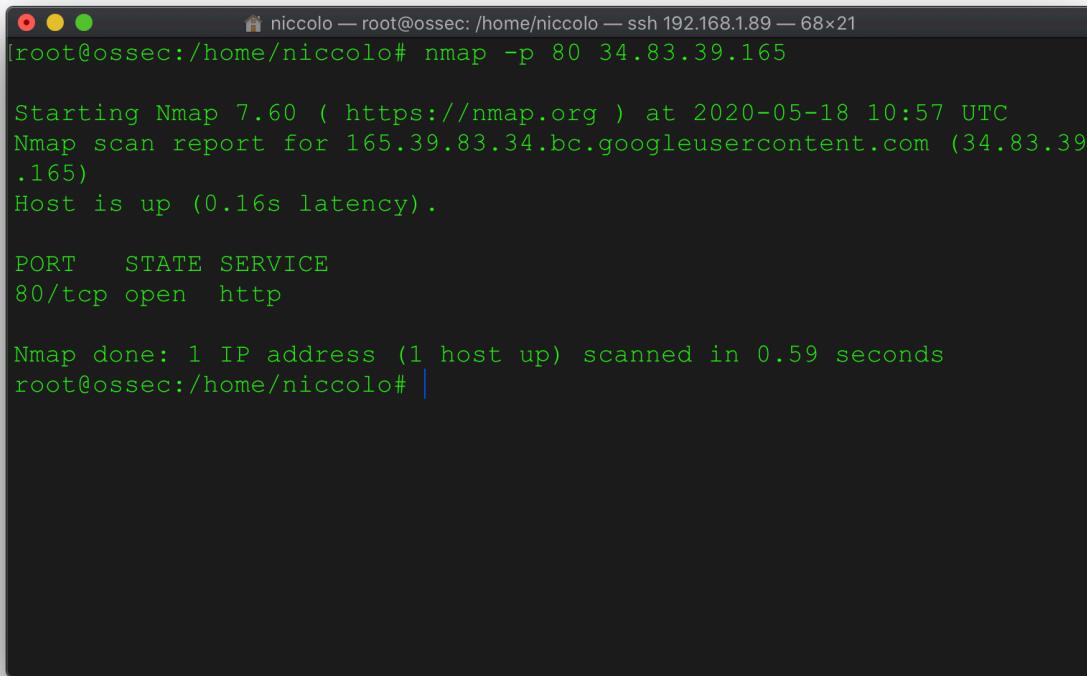
Email di notifica da parte di M/Monit

4.5 Test di accesso a porte bloccate con firewall

Questo test mira a verificare il corretto funzionamento del firewall nella piattaforma Google Cloud. Oltre che a verificare il corretto funzionamento dello stesso firewall messo a disposizione da Google, deve verificare che il firewall stia effettivamente operando nel network che abbiamo in uso nella nostra infrastruttura e non in un altro.

Test con le porte del FW aperte

Il test procederà come segue: innanzitutto effettuiamo mediante il comando nmap una chiamata all'indirizzo IP pubblico della nostra istanza virtuale sulla porta 80. Ciò che ci si aspetta è che lo stato di questa porta sia aperto, poiché lo abbiamo definito nel file "Infrastructure as Code" precedentemente.



```
niccolo — root@ossec: /home/niccolo — ssh 192.168.1.89 — 68x21
[root@ossec:/home/niccolo# nmap -p 80 34.83.39.165 ]]

Starting Nmap 7.60 ( https://nmap.org ) at 2020-05-18 10:57 UTC
Nmap scan report for 165.39.83.34.bc.googleusercontent.com (34.83.39
.165)
Host is up (0.16s latency).

PORT      STATE SERVICE
80/tcp    open  http

Nmap done: 1 IP address (1 host up) scanned in 0.59 seconds
root@ossec:/home/niccolo# |
```

Comando NMAP con le porte aperte del firewall

Possiamo osservare che il riscontro è positivo e la porta 80 risulta aperta.

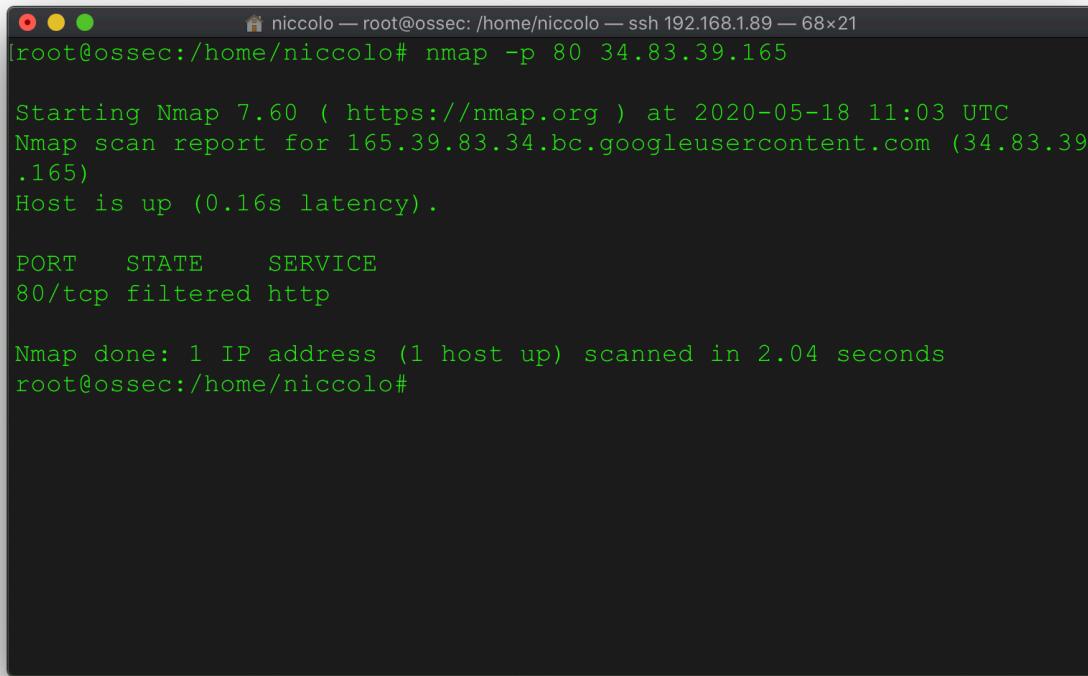
Chiusura delle porte del firewall

Prima di riprovare a effettuare la connessione mediante nmap, chiudo la porta 80 del firewall: per farlo elimino la porta 80 dalle impostazioni del firewall nella Google Cloud Console.

The screenshot shows the Google Cloud Platform interface for managing VPC network firewall rules. On the left, a sidebar lists various VPC-related options like VPC networks, External IP addresses, Firewall rules, Routes, VPC network peering, Shared VPC, Serverless VPC access, and Packet mirroring. The 'Firewall rules' option is currently selected. The main panel displays a form for creating a new firewall rule. The 'Targets' section is set to 'All instances in the network'. The 'Source filter' section shows 'IP ranges'. The 'Source IP ranges' field contains '0.0.0.0/0' with a note 'for example, 0.0.0.0/0, 192.168.2.0/24'. The 'Second source filter' is set to 'None'. Under 'Protocols and ports', the 'Specified protocols and ports' radio button is selected, and the 'tcp' field contains '80,443,22,5050,8080,8081'. The 'udp' field has 'all' selected. There is also a field for 'Other protocols' with the placeholder 'protocols, comma separated, e.g. ah, sctp'. At the bottom, there are 'SAVE' and 'CANCEL' buttons, and a note 'Equivalent REST'.

Impostazioni del firewall in VPC network

Una volta modificate e salvate le impostazioni rieffettuo un test di connessione con nmap: questa volta a differenza di prima mi aspetto di ottenere un risultato negativo.

A screenshot of a terminal window titled "niccolo — root@ossec: /home/niccolo — ssh 192.168.1.89 — 68x21". The window contains the following text:

```
[root@ossec:/home/niccolo# nmap -p 80 34.83.39.165

Starting Nmap 7.60 ( https://nmap.org ) at 2020-05-18 11:03 UTC
Nmap scan report for 165.39.83.34.bc.googleusercontent.com (34.83.39
.165)
Host is up (0.16s latency).

PORT      STATE      SERVICE
80/tcp    filtered  http

Nmap done: 1 IP address (1 host up) scanned in 2.04 seconds
root@ossec:/home/niccolo#]
```

Comando NMAP con porte chiuse del firewall

Come possiamo osservare nella schermata qui sopra, nella sezione “STATE” questa volta nmap rileva la presenza del firewall, pertanto non si può accedere alla risorsa.

5. Analisi dei costi

Sebbene abbiamo prevalentemente utilizzato tecnologie gratuite messe a disposizione da aziende come GitLab, Terraform, Ossec ecc. In questo capitolo riportiamo i costi per 100 utenti.

I costi reali per l'infrastruttura sono i seguenti:

Google Cloud Platform

Il costo di utilizzo dei servizi in Google Cloud è molto vario e difficile da stimare. Google mette a disposizione una pagina <https://cloud.google.com/pricing/list> nella quale riporta tutti i costi dei vari servizi: come possiamo osservare i costi per le macchine virtuali dipendono dal tipo, dalla locazione, dal server e dall'uso che ne facciamo e da molti altri fattori. Per facilitare l'operazione di calcolo ho utilizzato un calcolatore trovato su internet: <https://cloud.google.com/products/calculator> in cui vanno inserite le risorse di cui necessitiamo con i vari dettagli. Sulla base delle tabelle ufficiali riportate da Google, il calcolatore fornirà i relativi costi.

The screenshot shows the Google Cloud Pricing Calculator. At the top, there's a navigation bar with icons for Compute Engine, App Engine, Kubernetes Engine, Cloud Run, Cloud Storage, Networking Egress, and Cloud Load Balancing. Below the bar, a search bar says "Search for a product you are interested in.". On the left, there's a sidebar titled "Instances" with dropdown menus for Number of instances (set to 1), What are these instances for? (Operating System / Software, Free: Debian, CentOS, CoreOS, Ubuntu, or other User Provided OS), Machine Class (Regular), Machine Family (General purpose), Series (N1), and Machine type (N1). On the right, under the "Estimate" tab, it shows "Compute Engine" with 1 x gitlab selected. It lists "730 total hours per month", "VM class: regular", "Instance type: n1-standard-4", "Region: Iowa", "Sustained Use Discount: 30%", "Effective Hourly Rate: USD 0.133", and "Estimated Component Cost: USD 97.09 per 1 month". At the bottom, it shows "Total Estimated Cost: USD 97.09 per 1 month", "Estimate Currency: USD - US Dollar", and two buttons: "EMAIL ESTIMATE" and "SAVE ESTIMATE".

Calcolatore dei servizi Google Cloud

Abbiamo inserito nel calcolatore le risorse necessarie per poter far sì che l'infrastruttura sia in grado di supportare un carico di 1000 progetti GitLab e 50 immagini Docker in totale, con 8 CPU, 8GB RAM, 200GB SSD: i costi stimati dal calcolatore corrispondono complessivamente a 190 dollari al mese, ossia due dollari per utente.

Per quanto riguarda la manutenzione del software GitLab rilascia ogni mese un aggiornamento, pertanto il costo per tenere aggiornata la nostra macchina virtuale è stimato attorno alle 2 ore di lavoro al mese.

Grazie al fatto che l'infrastruttura è totalmente virtuale vengono abbattuti i costi di gestione e manutenzione fisica.

Inoltre l'interessante vantaggio dell'utilizzo di un orchestratore è che permette di creare tutta l'infrastruttura in modo riproducibile.

Dominio dockerbox.ch

Il dominio dockerbox.ch è stato acquistato presso hostpoint.ch e ha un costo di CHF 15 all'anno. Ho acquistato questo dominio in promozione: di fatto durante il primo anno mi costerà solamente CHF 5 .

6. Glossario

Questo piccolo glossario spiega i principali termini utilizzati nella presente documentazione.

Cloud Computing

Il cloud computing indica in informatica un paradigma di erogazione di servizi offerti su richiesta da un fornitore a un cliente finale attraverso la rete internet, a partire da un insieme di risorse preesistenti, configurabili e disponibili in remoto sotto forma di architettura distribuita.

~ Wikipedia / Cloud Computing - https://it.wikipedia.org/wiki/Cloud_computing

Compute Engine (CE)

Google Compute è un prodotto presente in Google Cloud: mediante questo prodotto si possono creare istanze virtuali e allocare risorse al sistema.

Domain Name Server (DNS)

Il DNS è il sistema che collega il dominio del progetto “dockerbox.ch” all’indirizzo IP della macchina virtuale. Sostanzialmente quando ci colleghiamo alla VM, anziché attraverso il suo indirizzo IP esterno, è possibile collegarsi mediante il dominio dockerbox.ch.

Firewall

Un firewall è un elemento che sta tra internet e la nostra macchina virtuale o server; permette con regole di bloccare certi tipi di traffico in modo da circoscrivere gli accessi fraudolenti alla macchina virtuale.

GitLab

GitLab è un software sviluppato dalla GitLab Inc. Il suo scopo principale è la gestione e il versioning dei file o di codice. Come funzionamento è molto simile al sito internet github.com ma la sostanziale differenza è che GitLab si può hostare in casa. A partire dalla versione 8.8 è disponibile un Container Registry e pertanto lo si può utilizzare per hostare le immagini Docker.

Google Cloud Platform (GCP)

Google Cloud è la piattaforma offerta da Google per il Cloud Computing. In sostanza è un ambiente di lavoro incui si può usufruire di servizi principalmente virtualizzati e da remoto, come ad esempio macchine virtuali, reti virtuali, server SQL e certi applicativi.

Host based intrusion detection system (HIDS)

Un HIDS è un software di “monitoraggio” che viene solitamente installato nel server che si desidera controllare. Una volta installato e configurato questo software, avviene un

monitoraggio proattivo di tutto quello ciò che succede nel sistema. L'HIDS controlla quali file vengono cambiati ed eventualmente danneggiati, o tiene traccia degli accessi ed eventi che avvengono nel sistema operativo.

Image Registry

Un Image Registry è un registro di immagini dei container. È un servizio che memorizza le nostre immagini Docker. Esistono servizi pubblici come il DockerHub, ma per ragioni di sicurezza e privacy molte aziende possono preferirne uno privato.

Infrastructure as Code (IaC)

L'infrastruttura come codice è un processo che prevede di descrivere in un file di configurazione l'infrastruttura virtuale che dobbiamo istanziare, ad esempio mediante software realizzati appositamente possiamo definire un ambiente virtuale di un Cloud anziché dover sempre creare manualmente la VM, impostare le chiavi di accesso e le risorse della VM, creare la rete ecc.

Ossec

Ossec è uno dei più noti HIDS, è completo sia della componente di monitoraggio sia di quella di notifica via email istantanea all'amministratore in caso di avaria del sistema.

Postfix

Postfix è il software che gestisce la posta in uscita nel nostro progetto "Registry". Principalmente viene utilizzato dai software di monitoraggio per inviare le email di notifica all'amministratore del progetto.

Terraform

Terraform è un prodotto sviluppato dalla HashiCorp ed è un programma che permette di applicare mediante un driver specifico a un cloud a scelta la nostra infrastruttura come codice.

VPC Network

VPC Network sta per Virtual Private Cloud Network: sostanzialmente è il prodotto che Google mette a disposizione per poter creare reti virtuali nella piattaforma Google Cloud.

7. Conclusioni e ringraziamenti

Il progetto è stato sviluppato secondo le tempistiche definite inizialmente nel diagramma di Gantt. Considerato l'importante carico di lavoro, inizialmente si è creato un piccolo ritardo delle attività di circa un giorno. Tuttavia le attività svolte e gli eventuali ritardi del progetto sono stati documentati e puntualmente riportati nel diario di lavoro.

Durante questi giorni ho avuto occasione di ampliare le mie competenze per quanto concerne l'implementazione e la configurazione di un'infrastruttura GCP definita in un file Terraform. A differenza dell'usuale lavoro svolto in quattro anni presso il CSCS ho avuto modo di mettere in pratica l'allestimento di piccoli script per automatizzare molte operazioni di installazione.

Con questo progetto ho potuto rendermi conto dell'elevato vantaggio pratico di avere un'infrastruttura inizializzata con Terraform. Di fatto, sono rimasto colpito dalla facilità e dalla rapidità con cui è possibile distruggere l'intera infrastruttura per ricrearne immediatamente un'altra; è un po' come premere due bottoni (crea e distruggi) anziché dover sempre ricreare manualmente, mediante l'interfaccia web di GCP, la macchina virtuale e reimpostarla.

Sono molto soddisfatto dell'esperienza di lavoro con le macchine virtuali in cloud. Fra una macchina virtuale in cloud e una macchina fisica non ho tuttavia riscontrato grandi differenze di configurazione e utilizzo. Il cloud è sicuramente vantaggioso per il fatto che si possono effettuare tutte le operazioni come se ci fosse un computer; inoltre non bisogna pensare all'infrastruttura fisica e ci si può concentrare in modo ottimale sulle attività da svolgere e non soltanto sulla loro attuazione.

Oltre ad aver appreso utili stratagemmi per poter istanziare in modo rapido e efficace delle risorse, con questo progetto ho potuto confrontarmi più da vicino con gli HIDS e con la configurazione di un server email Postfix.

La durata del progetto e l'attuazione sono state in linea con la pianificazione iniziale. Se avessi avuto più tempo, avrei potuto approfondire maggiormente il monitoraggio e implementare gli unit test, ad esempio sarebbe stato interessante creare un test automatico quotidiano per caricare e scaricare un'immagine Docker dal Registry.

Attualmente ogni macchina virtuale in GCP che creiamo genera ogni volta un indirizzo IP. Sarebbe stato interessante approfondire le API che Cloudflare mette a disposizione agli utenti: utilizzandole si sarebbe potuto automatizzare con certezza l'aggiornamento del server DNS nel record "dockerbox.ch" che cambia l'IP in modo automatico quando viene istanziata una nuova macchina virtuale con Terraform.

Come futuri sviluppi del progetto si potrebbe implementare un sistema di “Load balancing” in modo che tutte le risorse siano sfruttate al meglio, inoltre si potrebbe integrare un sistema “High Availability” in modo da rendere le risorse altamente disponibili.

Attualmente quando eseguiamo il comando “terraform destroy” perdiamo tutta l’infrastruttura e i dati in essa, pertanto si potrebbe implementare un “Object Storage” per poter immagazzinare le informazioni in modo permanente.

Giunto al termine di questo lavoro desidero ringraziare il Centro Svizzero di Calcolo Scientifico (CSCS) di Lugano, ovvero il mio datore di lavoro con cui ho avuto il piacere di svolgere il mio apprendistato, in un clima di scambio formativo e culturale di indubbio interesse e valore.

Oltre quindi ai colleghi della sede luganese, con i quali ho avuto modo di confrontarmi quotidianamente sempre in maniera propositiva, tengo qui a ringraziare in particolare i miei formatori - signor Alessandro Prato ed il suo predecessore, signor Nicola Bianchi - per la pazienza e la competenza con le quali hanno saputo accompagnarmi in questi quattro anni di formazione professionale.

8. Allegati

8.1 Approfondimenti

Documentazione ufficiale di Terraform

Link: <https://www.terraform.io/docs/providers/google/>

Nella documentazione ufficiale della casa produttrice sono presenti tutti i dettagli per poter configurare il file Terraform. Se in futuro qualcosa dovesse cambiare o qualche funzione non dovesse più essere supportata, attingendo alla documentazione ufficiale è possibile aggiornare il file Terraform.

Documentazione delle regole di Ossec

Link: https://ossec-docs.readthedocs.io/en/latest/docs/syntax/head_rules.html

In questa pagina è riportata in modo esaustivo tutta la sintassi per la creazione delle regole di Ossec.

Documentazione ufficiale di M/Monit

Link: <https://mmonit.com/monit/documentation/monit.html>

In questa documentazione è spiegato in modo molto specifico il funzionamento di M/Monit.

8.2 Documenti allegati

Questa sezione di documentazione spiega in dettaglio il contenuto dei vari allegati di questo progetto. Gli allegati verranno consegnati a parte.

File dell'infrastruttura Terraform [main.tf]

Il file contiene prevalentemente codice e alcuni piccoli commenti per potersi orientare al meglio. In questo file è definita tutta l'infrastruttura da assemblare in Google Cloud; per poterlo eseguire è necessario aver installato il software Terraform, un account GCP e la chiave per poter accedere con Terraform alla piattaforma.

Script di primo avvio dell'infrastruttura [bootstrap.sh]

Questo file contiene codice “bash” che viene eseguito immediatamente dopo aver eseguito lo script Terraform visto in precedenza. L’output generato da questo file è “headless”, ovvero tutti i messaggi generati alla console non verranno mai visti dall’utente. Pertanto è consigliabile evitare che avvengano processi di richiesta di input da parte dell’utente e inserire anticipatamente i parametri di un comando in modo da raggirare queste richieste.

Script di installazione di M/Monit [monit.sh]

Questo file contiene lo script di installazione di M/Monit; nella parte finale del file va a modificare il file di configurazione cancellandone il contenuto e aggiungendone uno proprio. A differenza del file di primo avvio questo script mostra un output a console, considerato che viene eseguito da un terminale SSH.

Script di installazione di Ossec [ossec.sh]

Questo è lo script di installazione di Ossec; a differenza dello script di M/Monit questo script necessita di alcune interazioni con l’utente come ad esempio l’inserimento della email dove si desidera ricevere le notifiche.

Script di installazione di Postfix [postfix.sh]

Questo script di installazione installa Postfix e va a configurare i vari file di configurazione per poter interfacciarsi con i server di Gmail.

Diario di lavoro [Diario di lavoro.docx]

In questo documento sono riportati tutti i rapporti dettagliati relativi alle giornate lavorative, le attività svolte, i problemi riscontrati e le soluzioni adottate. Inoltre è anche disponibile un personale commento sullo stato delle tempistiche.

La documentazione è stata consegnata nella seguente data:

.....

Luogo:

Data:

Firma candidato:

Luogo:

Data:

Firma formatore:

Luogo:

Data:

Firma perito: