DEVOPS

Best Practices per un prodotto migliore

Dario Pasquali

REPLY DATA

AGENDA
DAY 1 - MATTINA

1. Introduzione
2. Dev & Ops
3. Best Practicies
4. Configuration Management
5. Infrastrucutre As a Code
6. Caso d’uso
7. Pratica

INTRODUZIONE

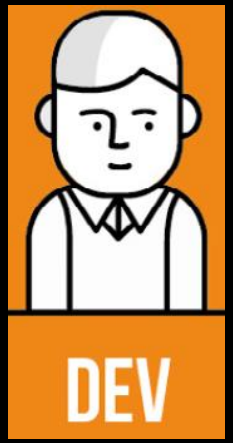
DEVOPS
OBIETTIVI

- Automatizzare il processo di deployment (idealmente ONE-CLICK-DEPLOY) in modo da aumentare notevolmente la frequenza dei rilasci.

- Migliorare la qualità del prodotto sviluppato

- Aumentare la coesione e collaborazione all'interno del team, migliorando la qualità del lavoro.

DEV & OPS
FIGURE IN CONFLITTO



DEVELOPERS

CHI: Developers, Testers, Product Owner, QAs

SCOPO: implementare le features richieste del cliente rapidamente

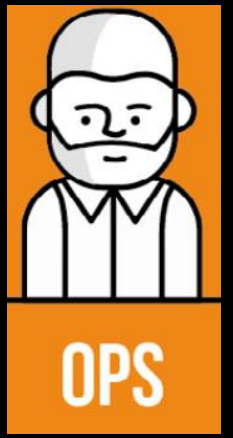
VALUTAZIONE: # features funzionati / tempo

OPERATIONS

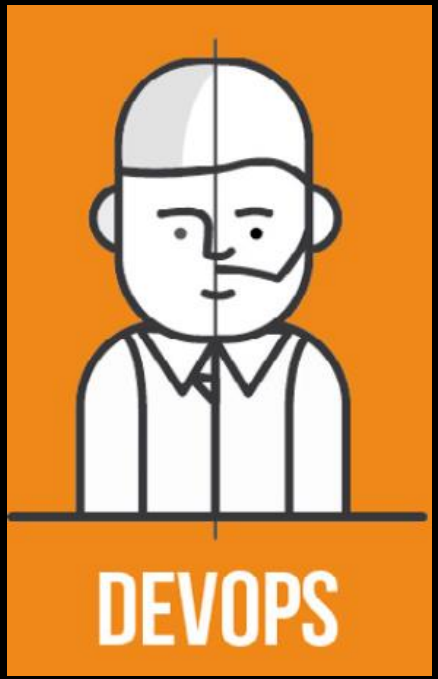
CHI: System Engineer, System Admin, DB Admin, Security

SCOPO: rilasciare il prodotto e assicurarsi che il sistema funzioni in modo sicuro ed efficente

VALUTAZIONE: stato del sistema, soddisfazione del cliente



DEV & OPS
FIGURE IN CONFLITTO



Creare un unico team **multifunzionale,** che si prenda cura del prodotto durante tutto il suo ciclo di vita.

Scopo e metodi di valutazione Unificati.

Il DevOps abbatte le bariere tra le persone, Developer, Data Scientist, System Engineers, Security, Testers, QAs e Product Owner, collaborano per il bene del prodotto.


«YOU BUILD IT, YOU RUN IT»

DEVOPS

CULTURA + TOOLS

DevOps = 7 Best Practices + Tools di supporto unificati

PRATICHE = Cambiamento Culturale, verso la Trasparenza e la Collaborazione nel team.

TOOLS = Guida al cambimento, unificando Strumenti e Responsabilità

7 BEST PRACTICES

1. Continuous Management
2. Infrastructure As a Code
3. Continuous Integration
4. Continuous Testing
5. Continuous Delivery
6. Continuous Deployment
7. Continuous Management

CONTINUOUS LEARNING

7 BEST PRACTICES

1 **CONTINUOUS MANAGEMENT**

2 **INFRASTRUCTURE AS A CODE**

3 Continuous Integration

4 Continuous Testing

5 Continuous Delivery

6 Continuous Deployment

7 Continuous Management

CONTINUOUS LEARNING

CONFIGURATION MANAGEMENT

«ESEGUIRE IL PROVISIONING DELLE ISTANZE DEL SISTEMA USANDO COMPONENTI SOFTWARE CONVERGENTI E IDEMPOTENTI, GESTIBILI CON GLI STESSI STRUMENTI DEL PRODOTTO SVILUPPATO»

CONFIGURATION MANAGMENT

PROVISIONING VIA SCRIPT

- Difficoltà di Comprensione
- Difficoltà di Condivisione tra i membri del team
- Dipendenza dalle Abitudini del singolo
- Complessa automazione della configurazione
- Nessuna garanzia di Idempotenza e Convergenza

CONFIGURATION MANAGMENT

IDEMPOTENZA & CONVERGENZA

IDEMPOTENZA = L’applicazione multipla della stessa operazione, dopo la prima esecuzione, non ha effetti sul sistema target

CONVERGENZA = un’operazione è eseguita se e soltanto se è strettamente necessario.

MODELLI DI CM

IMPERATIVO

Configurazione come SEQUENZA DI AZIONI da eseguire per raggiungere lo stato desiderato.

Modello MASTER-SLAVE necessita di un agente installato sul target

MASSIMO CONTROLLO ma necessità di conoscere il processo

MODELLI DI CM

DICHIARATIVO

Configurazione come STATO DA RAGGIUNGERE, specificato in modo Descrittivo

Modello MASTER-SLAVE necessita di un agente installato sul target

MASSIMA ASTRAZIONE ma perdita del controllo nel dettaglio

MODELLI DI CM

IBRIDO

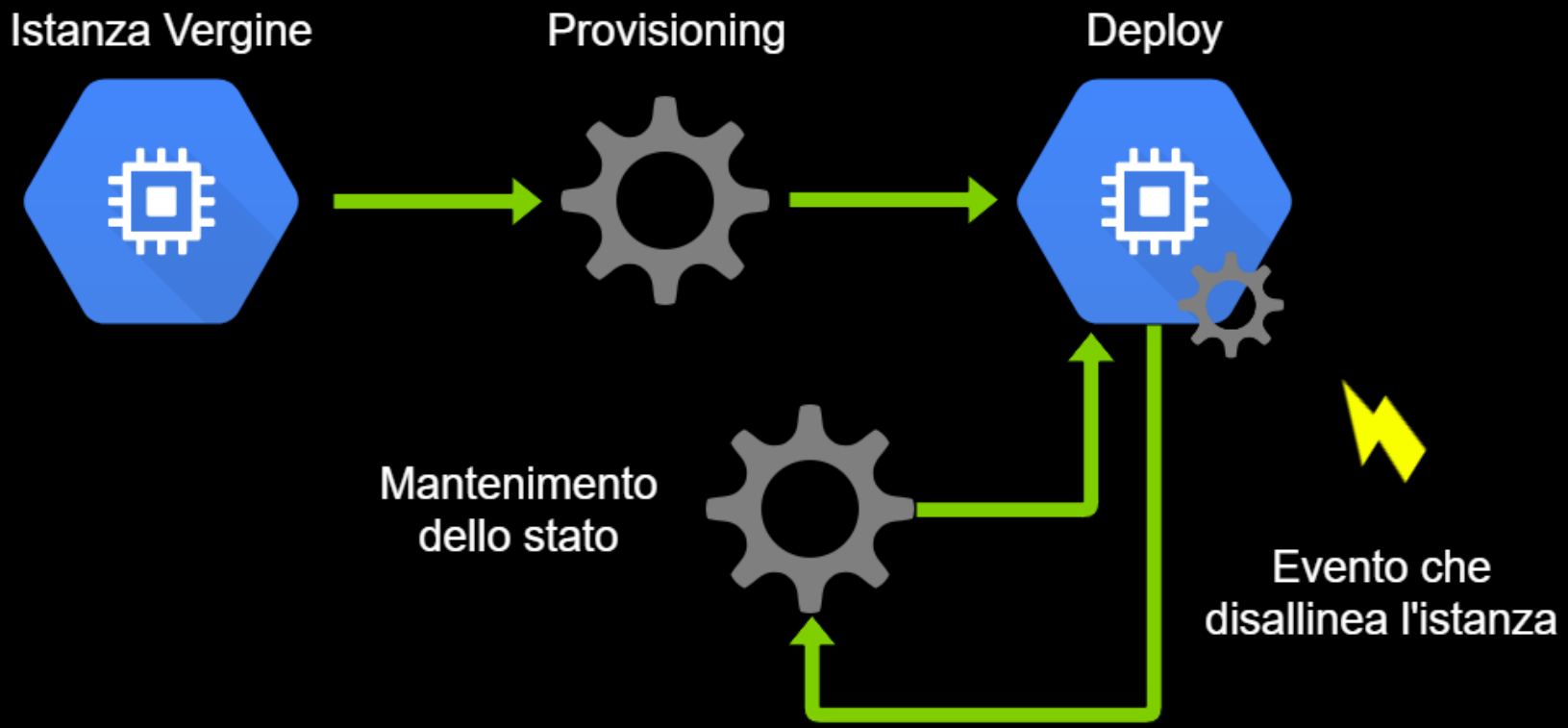
Configurazione come sequeza di azioni astratte in forma di ROLES

Modello AGENT-LESS basato su SSH e Python

MASSIMA MODULARITA' e customizzazione al livello di dettaglio desiderato

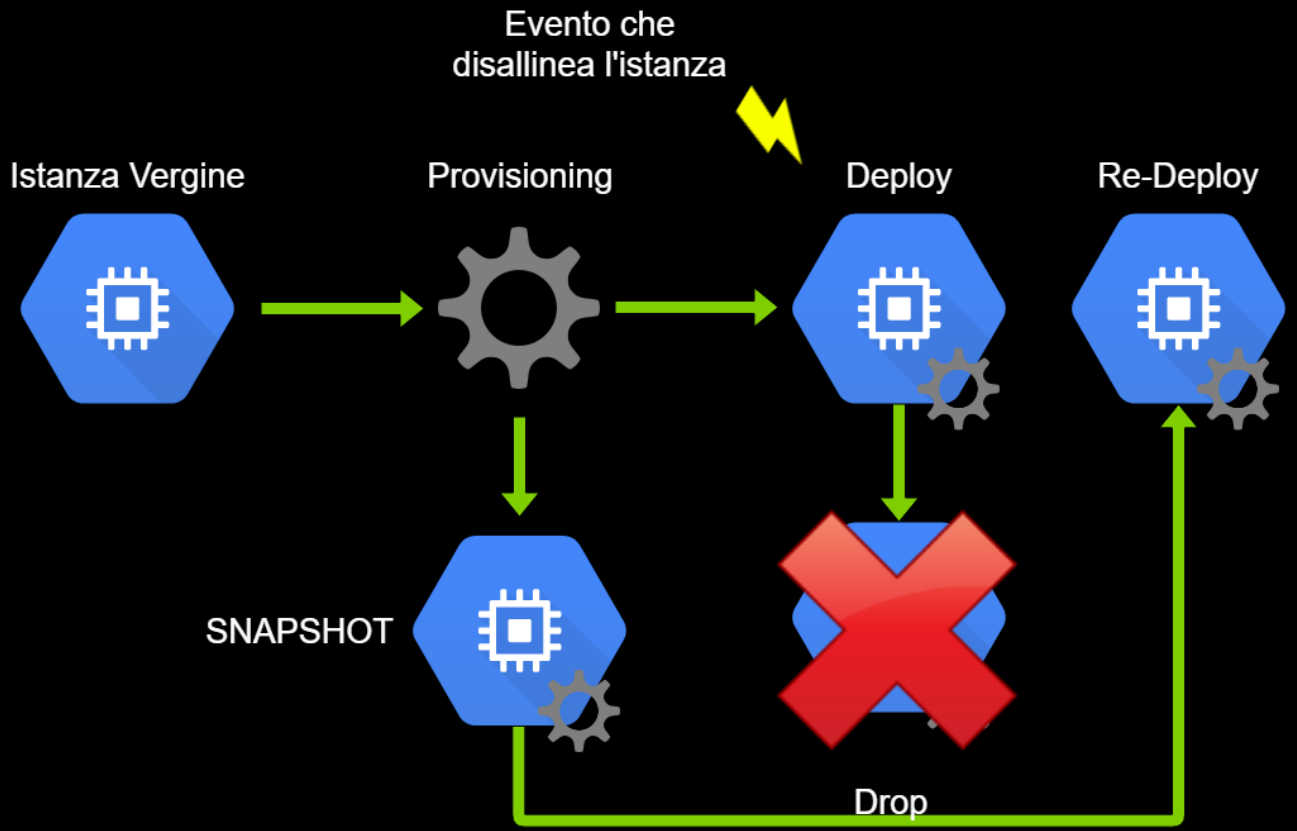
MODELLI DI GESTION

ARCHITETTURA DINAMICA – FRY MODEL



MODELLI DI GESTIONE

IMMUTABLE INFRASTRUCTURE – BAKED MODEL



ANSIBLE
CM IBRIDO

- Tool di CM ibrido, unisce i vantaggi degli approcci Imperativo e Dichiarativo
- Possibilità di specificare la configurazione come sequenza di azioni, o come astrazione di **ROLE** nel sistema
- Provisioning Agent – less basato su SSH

ANSIBLE

CM IBRIDO

PLAYBOOK = modulo auto – contenuto, scritto in YAML che descrive la configurazione da applicare ad un’istanza

INVENTORY = modulo auto – contenuto, scritto in YAML, che definisce la configurazione e l’istanza target

ROLE = Astrazione standalone di una configurazione del sistema

ANSIBLE
PLAYBOOK

PLAYBOOK

Definisce il target del Provisioning e la sequenza di task da applicare.

Yum e get_url sono moduli che racchiudono un comportamento

Target definiti tramite labels

```yaml
- hosts: cluster-nodes
  become: yes
  become_user: root
  tasks:
    - name: Install JDK
      yum:
        name: java-1.8.0-openjdk
        state: present

    - name: Donwload SBT
      get_url:
        url: http://dl.bintray.com/sbt/rpm/sbt-0.13.12.rpm
        dest: /opt/sbt-0.13.12.rpm
        mode: 0440

    - name: Install SBT
      yum:
        name: /opt/sbt-0.13.12.rpm
        state: present
```

ANSIBLE
INVENTORY

INVENTORY

Definisce le istanze target

Possibilità di:

- Creare gruppi di istanze identificate da label
- Usare regex nei nomi
- Configurare parametri specifici di istanza (porta, utente, chiave, ...)

```
[local]
127.0.0.1

[cluster-nodes]
gcp-cluster-node-[01:50]
```

ANSIBLE
ROLE

ROLE

Componente STANDALONE che racchiude e definisce una responsabilità nel sistema.

Arricchito da tasks, variabili, tests, hadlers, documentazione, metadati.

Condivisione e distribuzione su Ansible Galaxy

```
- hosts: cluster-nodes
  become: yes
  become_user: root
  roles:
    - sbt
```

INFRASTRUCTURE AS A CODE

«ORCHESTRATION DELL'INFRASTRUTTURA CON COMPONENTI ESEGUIBILI, IDEMPOTENTI E CONVERGENTI, GESTIBILI CON GLI STESSI STRUMENTI DEL PRODOTTO SVILUPPATO»

INFRASTRUCTURE AS A CODE

IAC

INFRASTRUTTURA

Modello eseguibile, Convergente, Idempotente e REPLICABILE in grado di essere compreso ed utilizzato da tutti i membri del team.

Nessuna dipendenza da un team di sistemisti o da complessi script bash.

TERRAFORM

ORCHESTRATION PROVIDER CLOUD

Orchestration di infrastrutture Cloud sui più comuni Cloud Providers (GCP, AWS, Docker, OpenStack, ...)

Oltre a numerosi servizi standard (GitHub, Slack, DNS, ...)

Esecuzione AGENT–LESS via SSH, idempotente e convergente.

TERRAFORM

ORCHESTRATION PROVIDER CLOUD

Configurazione Dichiarativa, specifica lo STATO DESIDERATO.

Terraform si occupa di trovare il modo migliore per raggiungerlo.

Esecuzione idempotente basata sullo stato attuale e sulle azioni passate di Terraform.

PLAN

Componente eseguibile che rappresenta lo snapshot di un'infrastruttura cloud. Replicabile a piacimento.

```
provider "google" {
  credentials = "${file("terraform-admin.json")}"
  project     = "endless-upgrade"
  region      = "us-east1-b"
}
resource "google_compute_instance" "worker" {
 project = "endless-upgrade"
 zone = "us-east1-b"
 count = "${var.server_count}"
 name = "gcp-cluster-node-${count.index}"
 machine_type = "n1-standard-2"
 tags = ["jenkins", "monitoring", "http", "https"]

 boot_disk {
    initialize_params {
      image = "centos-7-v20171213"
      size = "30"
    }
 }
}
output "worker" {
 value = "${google_compute_instance.worker.self_link}"
}
```

TERRAFORM
PROVISIONING

Terraform permette di eseguire il Provisioning con Tool esterni...

...come ANSIBLE !!

```
...

provisioner "local-exec" {
  command = "sleep 90;
    ansible-playbook -i '${google_compute_instance.worker.name},'
    --private-key=~/.ssh/ansible_rsa /opt/ansible/cluster-nodes.yml
    -e 'ansible_ssh_user=dario_pasquali93' -e 'host_key_checking=False'"
 }
}

...
```


CM + IAC
=
CREAZIONE E PROVISIONING
AUTOMATICO BY-NEED
Possibilità di Scalare il Sistema a piacimento, in modo
Idempotente, Convergente e Replicabile

CASO D’USO

MOVIE ADVICER

REQUISITI POC

- RACCOMANDATORE BINARIO basato sul Dataset Movielens 100k
- I dati di interesse devono essere memorizzati su un DATA LAKE, implementato in Kudu
- Si vuole presentare una semplice INTERFACCIA per interagire con il sistema

MOVIE ADVICER

ARCHITETTURA

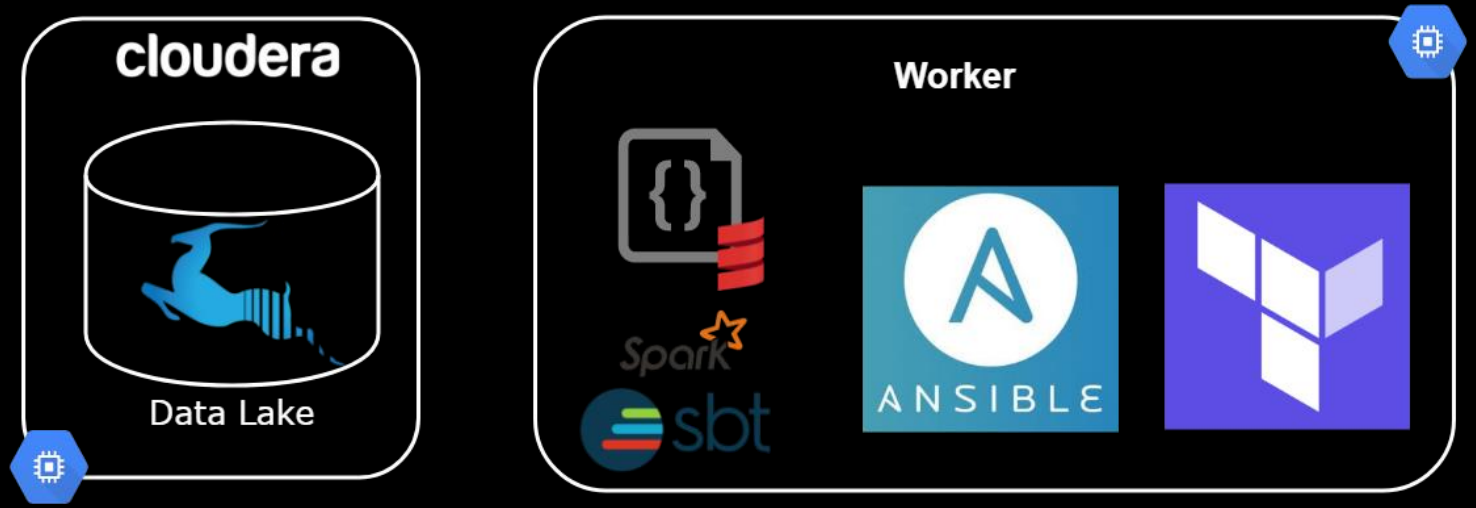
Sistema basato su due nodi:

- CLOUDERA: nodo storage, ospita il Data Lake
- WORKER: nodo di elaborazione, ospita il sistema di raccomandazione.

Come creare queste istanze RAPIDAMENTE e in modo AUTOMATICO ??

CM + IAC

ARCHITETTURA
RESPONSABILITÀ NEL SISTEMA



ARCHITETTURA

IDENTIFICARE I RUOLI

Ogni RESPONSABILITÀ nel sistema deve essere convertita in RUOLO.

Elementi indipendenti e semplici !!

- Cloudera CDH
- Oracle JDK 1.7
- Spark 2.2.0
- SBT
- Ansible
- Terraform
- Directory processo

ARCHITETTURA

RUOLI ANSIBLE – ESEMPI

```yaml
---
# tasks file for spark
- name: Donwload Spark
  get_url:
      url: http://mirror.nohup.it/apache/spark/
        spark-2.2.0/spark-2.2.0-bin-hadoop2.7.tgz
      dest: /opt/spark-2.2.0-bin-hadoop2.7.tgz
      mode: 0440

- name: Extract Spark tgz
  unarchive:
      src: /opt/spark-2.2.0-bin-hadoop2.7.tgz
      dest: /opt/

- name: Ensure Export file exists
  copy:
      content: ""
      dest: /etc/profile.d/exports.sh
      force: no
      group: root
      owner: root
      mode: 0555
```

```yaml
- name: Export Spark home
  lineinfile:
      path: /etc/profile.d/exports.sh
      state: present
      insertafter: EOF
      line: 'export SPARK_HOME=/opt/spark-2.2.0-bin-hadoop2.7'

- name: Export Path
  lineinfile:
      path: /etc/profile.d/exports.sh
      state: present
      insertafter: EOF
      line: 'export PATH=$PATH:$SPARK_HOME/bin'

- name: Source evn variables
  shell: /etc/profile.d/exports.sh
```

ARCHITETTURA

RUOLI ANSIBLE - ESEMPI

```yaml
# tasks file for cloudera
- name: Config Environment
  include_role:
    name: cm-env-setup

- name: Setup repository
  include_role:
    name: cm-repo

- name: Install Oracle Java
  include_role:
    name: oracle-java

- name: Install Cloudera Manager with Path B
  include_role:
    name: cm-install-path-b

- name: Wait for cloudera to start up before proceeding.
  shell: "curl -D - --silent --max-time 5 http://localhost:7180/cmf"
  register: result
  until: (result.stdout.find("403 Forbidden") != -1) or
    (result.stdout.find("200 OK") != -1) and
    (result.stdout.find("Please wait while") == -1)
  retries: "60"
  delay: "10"
  changed_when: false
  check_mode: no
```

```yaml
- name: Config users
  include_tasks: users.yml

- name: Config network
  include_tasks: network.yml

- name: Config services
  include_tasks: services.yml
```

```yaml
- name: disable SELINUX
  lineinfile:
    path: /etc/selinux/config
    regexp: '^SELINUX='
    line: 'SELINUX=disabled'
  register: selinux_disabled

- include_tasks: reboot.yml
  when: selinux_disabled.changed

- name: Disable useless services
  service:
    name: '{{ item }}'
    state: stopped
    enabled: no
  ignore_errors: yes
  with_items:
    - postfix

- name: Disable paging and memory swapping
  lineinfile:
    path: /etc/rc.local
    line: '{{ item }}'
  with_items:
    - echo never > /sys/kernel/mm/redhat_transparent_hugepage/defrag
    - echo never > /sys/kernel/mm/redhat_transparent_hugepage/enabled
```

```yaml
- name: Manual stop instance
  local_action: shell gcloud compute
    instances stop {{ inventory_hostname }} --zone=us-east1-b

- name: Manual start instance
  local_action: shell gcloud compute
    instances start {{ inventory_hostname }} --zone=us-east1-b

- name: Wait for machine to restart
  local_action:
    module: wait_for
      host={{ inventory_hostname }}
      port=22
      delay=15
      timeout=300
      state=started
  become: false
```

ANSIBLE

SETUP

```
# ------------------------------
# Installare Ansible in Centos 7
# ------------------------------
sudo yum install python-devel python-pip -y
pip install ansible
```

ANSIBLE
CREARE UN ROLE

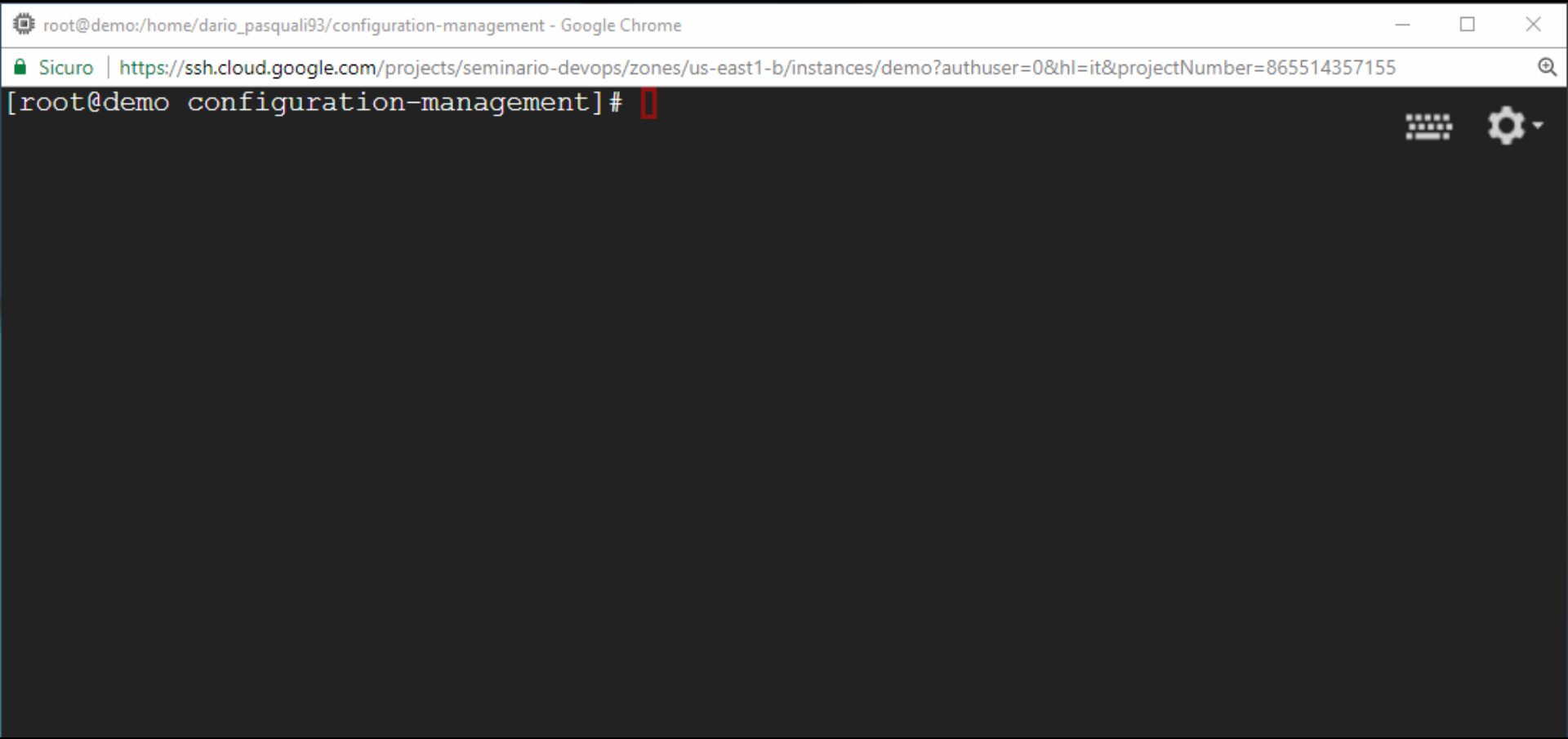
```
# ------------------------------
# Struttura directory
# ------------------------------
./
|- worker-pb.yml
|- .hosts
|- roles
    |- spark
    |- sbt
    |- ...

# ------------------------------
# Creare Role
# ------------------------------
ansible-galaxy *nome-ruolo*
```

ANSIBLE
ESECUZIONE

```
# ------------------------------
# Eseguire comando direttamente
# ------------------------------
ansible -i ./hosts [--connection=local local] | *label* -m ping
ansible -i 'machineAddress,' all -m ping

# ------------------------------
# Eseguire playbook
# ------------------------------
ansible-playbook -i ./hosts [--connection=local local] | *label* worker-pb.yml
ansible-playbook -i 'machineAddress,' all worker-pb.yml
```


root@demo:/home/dario_pasquali93/configuration-management - Google Chrome
Sicuro
https://ssh.cloud.google.com/projects/seminario-devops/zones/us-east1-b/instances/demo?authuser=0&hl=it&projectNumber=865514357155
[root@demo configuration-management]#

TERRAFORM
CONFIGURAZIONE HARDWARE

CLOUDERA

n1-standard-4 (4 vCPU, 15 GB di memoria), us-east1-b, disco 40 GB

tags: «cloudera, http-server, https-server»

WORKER

n1-highmem-2 (2 vCPU, 13 GB di memoria), us-east1-b, disco 40 GB

tags: «worker, support, http-server, https-server»

TERRAFORM

ESEMPIO

```
provider "google" {
  credentials = "${file("/path/to/private/gcp/project/key.json")}"
  project     = "seminario-devops"
  region      = "us-east1-b"
}

resource "google_compute_instance" "worker" {
 project = "seminario-devops"
 zone = "us-east1-b"
 name = "terraform-test"
 machine_type = "n1-standard-1"

 tags = ["worker", "support", "http", "https"]

 boot_disk {
   initialize_params {
     image = "centos-7-v20171213"
     size = "10"
   }
 }
```

```
 network_interface {
   network = "default"
   access_config {
   }
 }

 provisioner "local-exec" {
   #command = "sleep 90; ansible-playbook -i '${google_compute_instance.worke
   command = "sleep 90; ansible -i '${google_compute_instance.worker.name},'
   all --private-key=ssh/private/key/path
   -m ping
   -e 'ansible_ssh_user=dario_pasquali93'
   -e 'host_key_checking=False'"
 }
}

output "worker" {
 value = "${google_compute_instance.worker.self_link}"
}
```

TERRAFORM

SETUP CHIAVI

IAM GCP

1. IAM e Amministrazione – Account di Servizio – Crea Account di Servizio
2. Ruolo = Proprietario – Fornisci nuova chiave privata = SI
3. Scaricare il JSON e metterlo nell’istanza

SSH RSA

1.
```
ssh-keygen -t rsa -f ~/.ssh/[KEY_FILENAME] -C [USERNAME]
```
2. Copia chiave pubblica in Compute Engine – Metadati – Chiavi SSH

```
[root@demo dario_pasquali93]# ll
total 4
drwxr-xr-x. 4 root             root             157 Feb 23 09:15 configuration-management
drwxrwxr-x. 4 dario_pasquali93 dario_pasquali93  32 Feb 23 09:51 infrastructure-as-a-code
-rw-rw-r--. 1 dario_pasquali93 dario_pasquali93 2328 Feb 23 09:42 seminario-devops-335ca62a75
e7.json
[root@demo dario_pasquali93]#
```

PRATICA

PRATICA
ESPERIMENTI

1. Creare un’istanza Compute Engine su GCP Centos 7
2. Clonare i repo configuration-management e infrastructure-as-a-code da GitHub
3. Installare Ansible
4. Analizzare i playbook forniti (fate ESPERIMENTI, se vi va male basta creare una nuova istanza)
5. Installare Terraform con Ansible
6. Configurare le Chiavi e provare a creare istanze

PRATICA

CREA LA TUA ARCHITETTURA

1. Immagina un’architettura a te familiare (Spark + Python + Anaconda + Jupyter + SBT + ...)
2. Convertine la configurazione in un insieme di ruoli, ben strutturati e modulari, in modo da poter generare istanze a piacimento.
3. Prova a creare N istanze di quel tipo con **ANSIBLE + TERRAFORM**

PRATICA

LINK UTILI

- Materiale del seminario: https://github.com/endless-upgrade
- Ansible Docs: http://docs.ansible.com/ansible/latest/index.html
- Ansible Modules: http://docs.ansible.com/ansible/latest/modules_by_category.html
- Terraform Docs: https://www.terraform.io/docs/index.html
- Terraform GCP provider: https://www.terraform.io/docs/providers/google/index.html