Ottimizzazione delle Infrastrutture Cloud su AWS: Terraform, Terragrunt e la Gestione Multi-Ambiente

Relatore
Prof. Arcangelo Castiglione

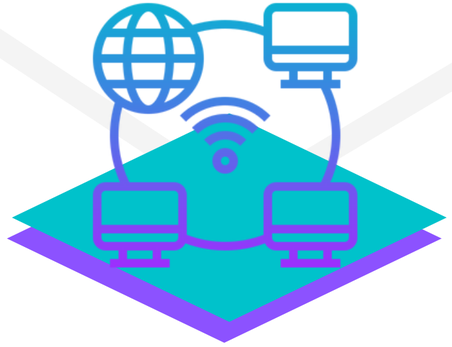
Relatore esterno
Gabriele Previtera (Epsilon SRL)

Candidato
Junhuang Chen
Mat. 0512112650

Contesto e Importanza

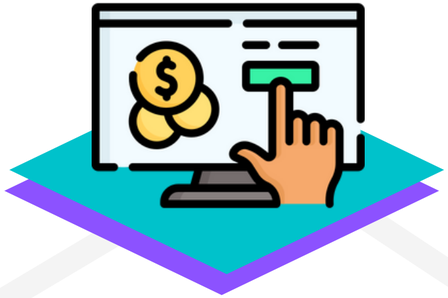
Cluster Computing

Un gruppo di computer collegati tra loro per lavorare come un'unica unità.



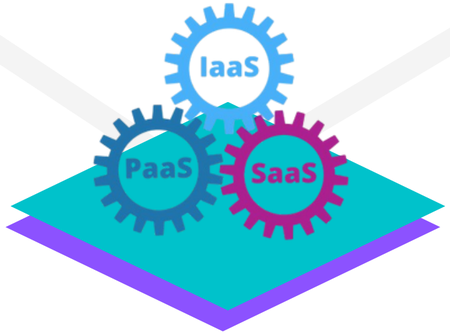
Grid Computing

Una rete distribuita di computer eterogenei che collaborano per risolvere problemi complessi.



Utility Computing

Un modello in cui le risorse computazionali sono fornite su richiesta, come un servizio pubblico.



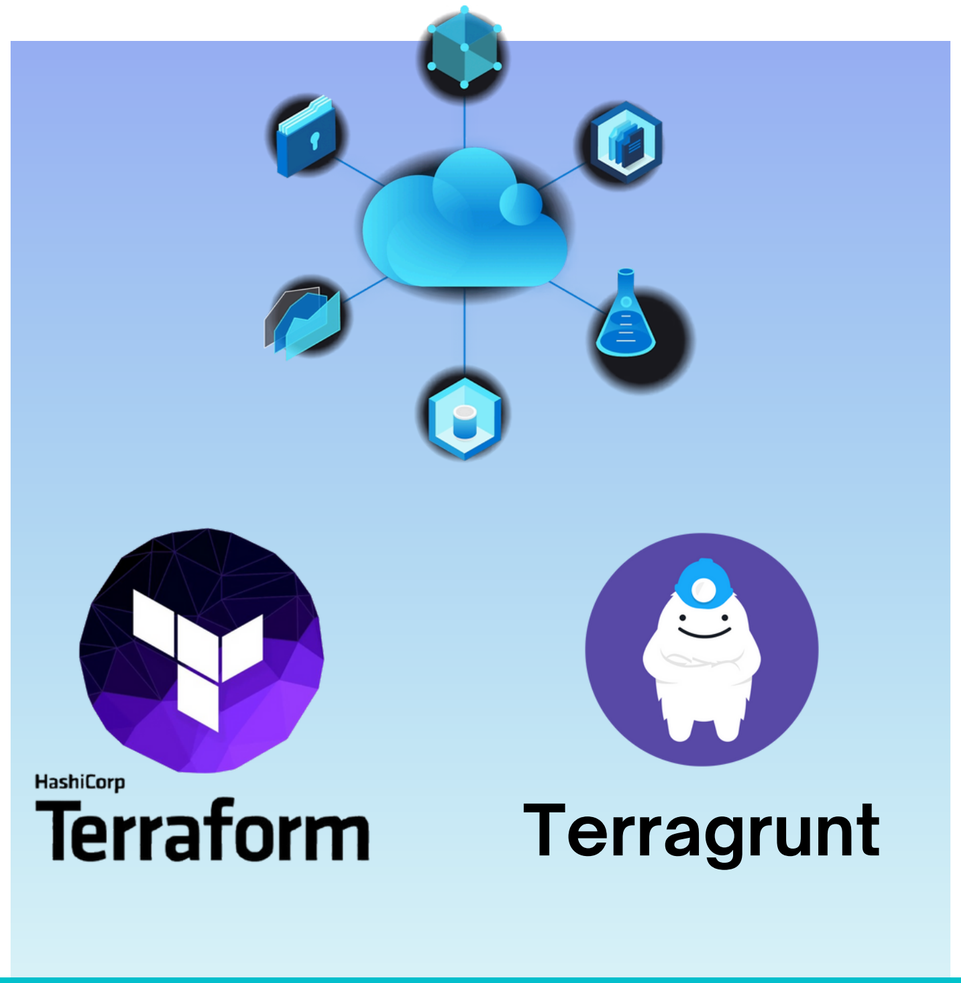
SaaS, PaaS, IaaS

Infrastructure-as-a-Service
Platforms-as-a-Service
Software-as-a-Service.

Cloud Computing

Un modello che fornisce risorse IT tramite Internet.

Obiettivi della Ricerca



- Analizzare i limiti di **Terraform** nella gestione multi-ambiente.
- Sperimentare l’efficacia di **Terragrunt** come soluzione.
- Valutare i vantaggi in termini di scalabilità e automazione.

Introduzione a Terraform

Terraform è un tool di Infrastructure as Code (IaC) che permette di definire e gestire l'infrastruttura cloud attraverso codice dichiarativo.

Caratteristiche

Dichiaratività
L'utente descrive lo stato desiderato dell'infrastruttura e Terraform applica le modifiche necessarie.

Modularità
Possibilità di riutilizzare componenti grazie ai moduli Terraform.

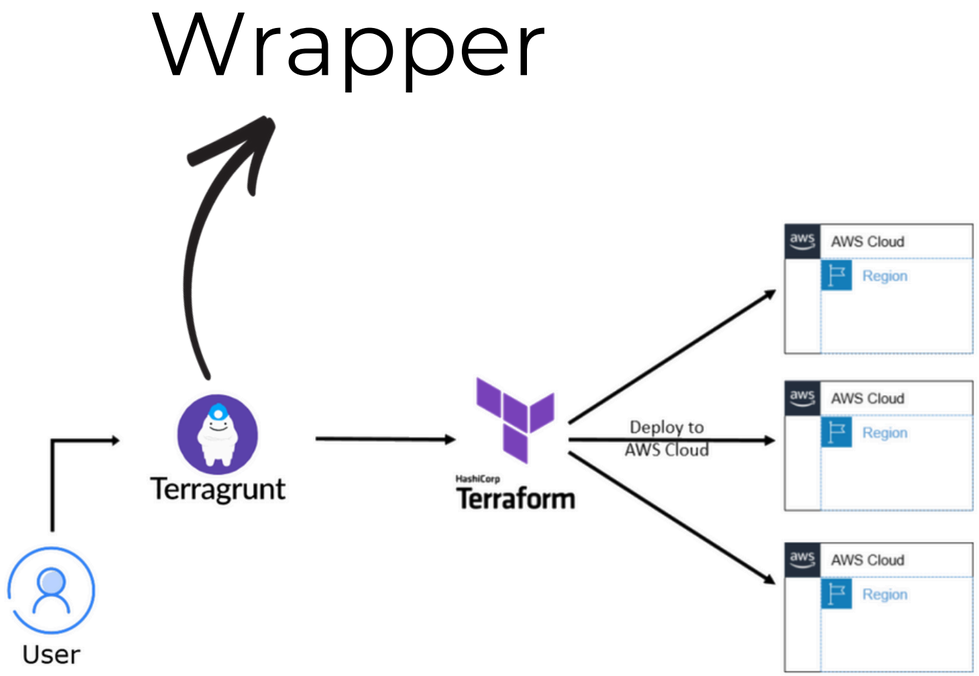
Multi-cloud
Compatibilità con vari provider cloud come AWS, Azure e Google Cloud.

Gestione Multi-Ambiente

Problemi di HashiCorp Terraform

- **Difficoltà nella gestione di configurazioni multi-ambiente.**
- **Configurazioni ripetitive e duplicazione del codice.**
- **Difficoltà nel riutilizzo della configurazione tra ambienti.**
- **Gestione dello stato complessa.**

Introduzione a Terragrunt



Astrazione sopra Terraform

Consente di evitare la duplicazione del codice, gestendo configurazioni ripetitive in modo più efficiente.

Centralizzazione della configurazione

Permette di mantenere le impostazioni condivise in un unico punto, migliorando la coerenza tra gli ambienti.

Automazione delle dipendenze

Facilita il passaggio di variabili tra moduli e la gestione della sequenza di deployment tra più risorse.

Soluzione con Terragrunt

HashiCorp Terraform

```
Middleware
└── environment
    ├── AWS Shared
    │   ├── main.tf
    │   ├── middleware.a
    │   ├── variables.tf
    │   └── outputs.tf
    ├── AWS Customer 1
    │   ├── main.tf
    │   ├── middleware.a
    │   ├── variables.tf
    │   └── outputs.tf
    ├── AWS Develop AI
    │   ├── main.tf
    │   ├── middleware.a
    │   ├── variables.tf
    │   └── outputs.tf
    └── application
        ├── main.tf
        ├── security-grc
        ├── load-balance
        └── ...
```

```
Application
├── environment
│   ├── AWS Shared
│   │   ├── main.tf
│   │   ├── application.auto.tfvars
│   │   ├── variables.tf
│   │   └── outputs.tf
│   ├── AWS Customer 1
│   │   ├── main.tf
│   │   ├── application.auto.tfvars
│   │   ├── variables.tf
│   │   └── outputs.tf
│   └── AWS Develop AI
│       ├── main.tf
│       ├── application.auto.tfvars
│       ├── variables.tf
│       └── outputs.tf
└── application
    ├── main.tf
    └── ...
```

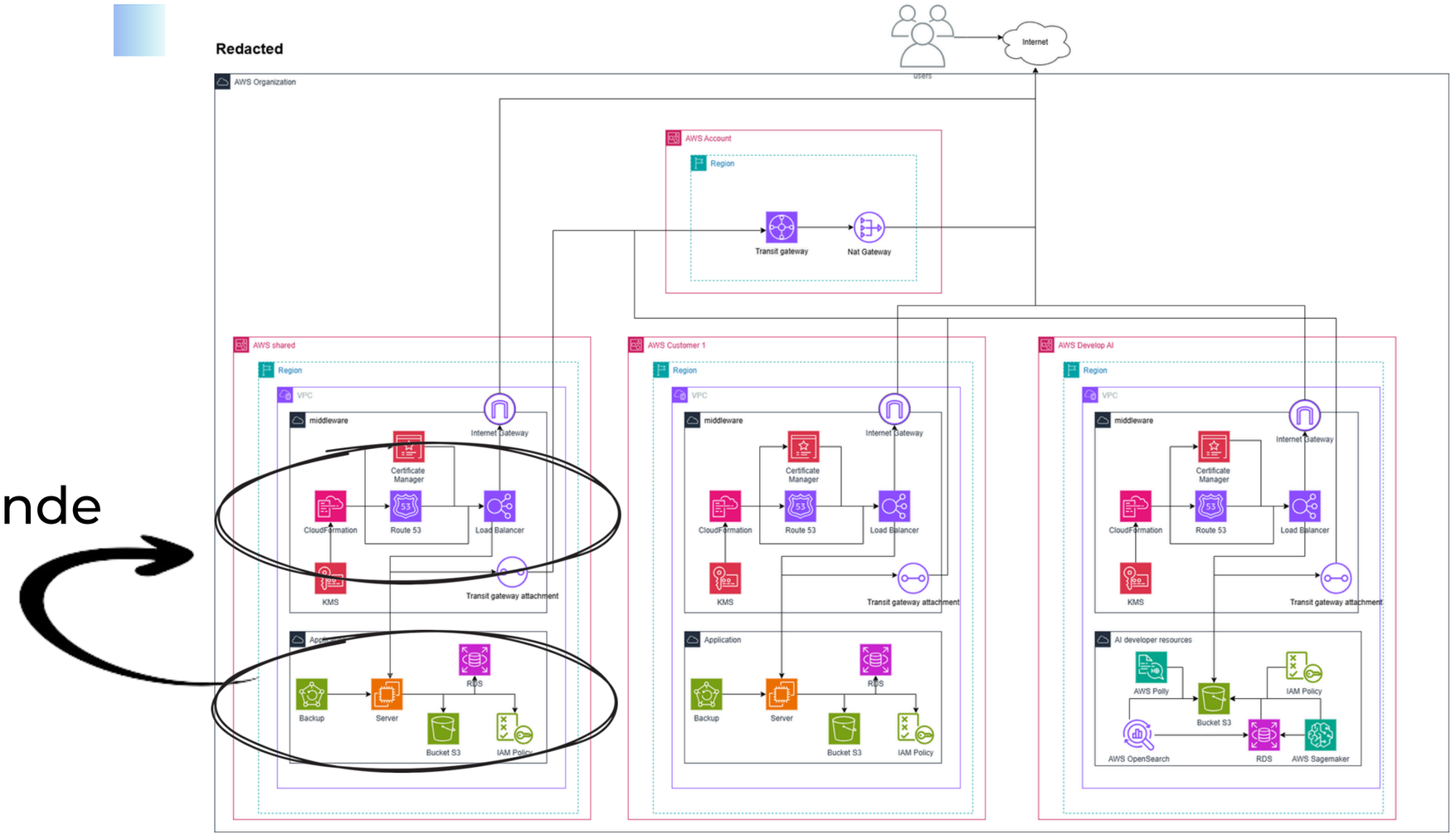
Terragrunt

```
middleware
├── terragrunt.hcl
├── environment
│   ├── AWS Customer 1 - ambiente 1
└── m
```

```
application
├── terragrunt.hcl
├── environment
│   ├── AWS Customer 1 - ambiente 1
│   │   └── terragrunt.hcl
│   ├── AWS Customer 1 - ambiente 2
│   │   └── terragrunt.hcl
│   ├── AWS Develop AI
│   │   └── terragrunt.hcl
│   └── ...
└── modules
    └── ...
```

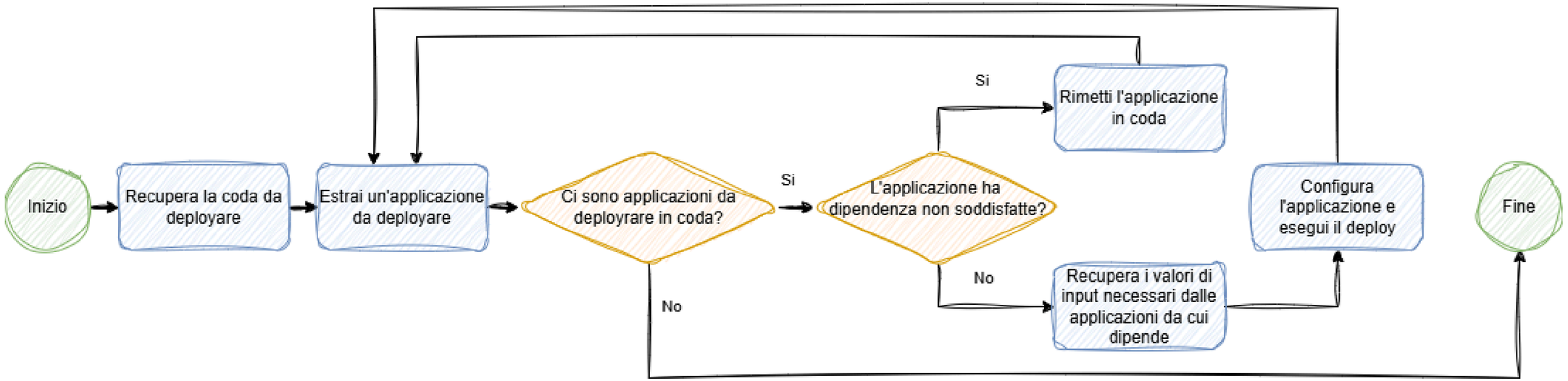
Architettura Implementata

Dipende



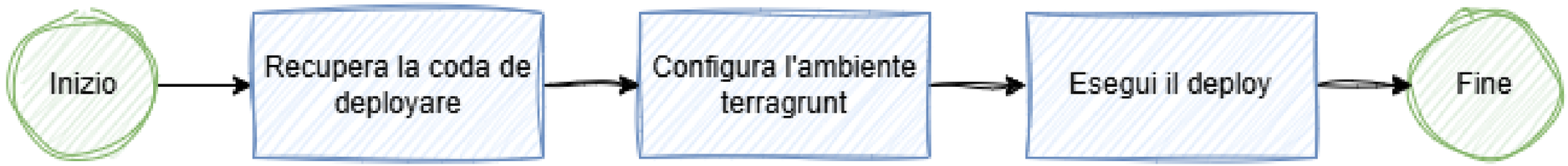
Processo di Deployment

HashiCorp Terraform



Processo di Deployment

Terragrunt

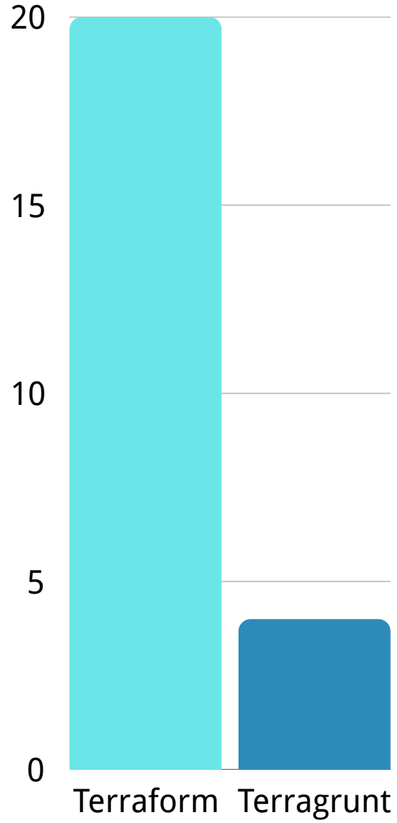


- **Passaggi migliorati rispetto a Terraform.**
- **Automazione delle configurazioni e gestione centralizzata.**

Confronto

HashiCorp Terraform

Flessibile ma con gestione multi-ambiente complicata.

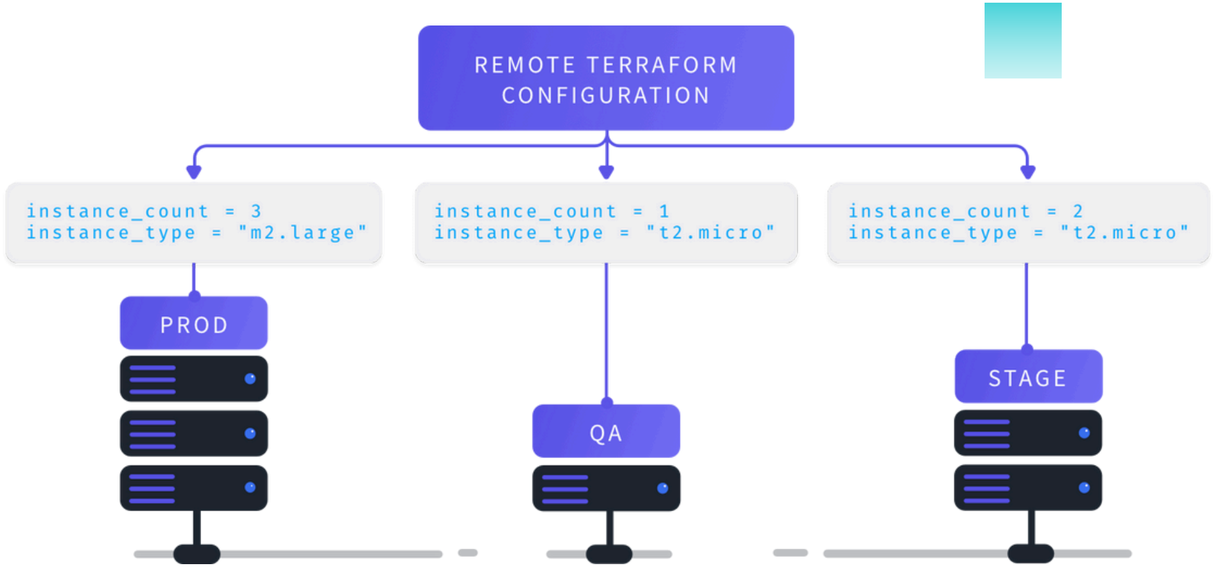


Terragrunt

Migliora la scalabilità, riduce errori e duplicazione del codice.

Una riduzione dell’80% !!

Benefici della Soluzione



Terraform & Terragrunt

Riduzione del tempo di gestione dell’infrastruttura.

Maggiore manutenibilità e scalabilità.

Integrazione più semplice con CI/CD.

Limiti e Possibili Miglioramenti

Limitation

Limitazioni del linguaggio dichiarativo

Non supportano funzioni avanzate come cicli o strutture dati complesse.

Gestione del parallelismo di risorse

L’esecuzione richiede ulteriori risorse del CPU e memoria.

Improvement

Automazione avanzata con hooks

Eseguire script personalizzati prima o dopo determinate operazioni di Terraform.

Integrazione con altri strumenti DevOps

Combinazione con strumenti come Ansible, Kubernetes e sistemi di CI/CD.

Grazie per l’attenzione!

Relatore
Prof. Arcangelo Castiglione

Relatore esterno
Gabriele Previtera
(Epsilon SRL)

Candidato
Junhuang Chen
Mat. 0512112650