Deployment all'interno di una Landing Zone con AWS Deployment Framework (ADF)

6 Gennaio 2023 - 8 min. read

AWS Deployment Framework (ADF) | Cloud adoption | governance and compliance | Landing Zone | multi-account strategy

Introduzione

Negli anni il mondo del Cloud si è spostato verso un approccio sempre più orientato all'*Infrastructure-as-Code* arrivando ad enormi vantaggi come replicabilità e standardizzazione.

L'Infrastrcuture-as-Code è notoriamente associato al deploy di infrastrutture atte a ospitare web application o, più in generale, workload aziendali.

Questa tematica cambia quando ci troviamo davanti a scenari in cui dobbiamo configurare il nostro ambiente Cloud per ospitare queste infrastrutture. Una preparazione adeguata richiede il deploy di una **Landing Zone** strutturata.

Ma come possiamo trarre i vantaggi dal paradigma dell'**Infrastructure-as-Code quando parliamo di Landing Zone**?

Abbiamo già trattato il concetto di Landing Zone in precedenza in questi articoli:

- Perché hai bisogno di una Landing Zone... (ma ancora non lo sai!)
- Progettare una Landing Zone su AWS: i pilastri fondamentali
- Esempi di implementazione di Landing Zone

Nell'articolo di oggi parleremo di come si possa gestire una Landing Zone tramite AWS Deployment Framework (ADF), uno strumento sviluppato dal team dei Proserv di AWS. Questo framework punta a semplificare e automatizzare il deployment di servizi e infrastrutture all'interno di una Landing Zone.

Perchè ADF

ADF possiamo vederlo come un'alternativa di AWS CloudFormation StackSet. Di fatto lo scopo di entrambe le soluzioni è **effettuare deployment di template CloudFormation per gestire uno scenario multi-account e multi-region**.

Ma quindi, perchè dovrei utilizzare ADF al posto di StackSet?

Per rispondere, diamo fondo alla nostra esperienza per ragionare sui pro e sui contro che speriamo possano tornarvi utili.

PRO

- Con ADF possiamo creare Account e Organizational Unit programmaticamente;
- Fornisce una flessibilità maggiore rispetto a StackSet;
- L'installazione di ADF prevede la creazione di una pipeline di CI/CD che ci permette di automatizzare la generazione di ulteriori pipeline di deployment al netto di un semplice git push;
- Perfetto per manutenere e creare automatismi per aziende di medie e grosse dimensioni (enterprise).

CONTRO

- La volontà di tenere questo framework il più flessibile possibile, ci porta a dover creare template ad hoc che fungano da plug-in; si possono trovare template di esempio, ma che non arrivano a coprire tutte le esigenze;
- Curva di apprendimento ripida;
- Overkilling per start-up e piccole aziende.

Come installare ADF

Prerequisiti

I presequisiti per installare ADF all'interno della nostra organization sono l'**accesso amministrativo all'account master**, o l'accesso ad un account stand alone. Nell'ultimo caso

sarà ADF a creare l'organization a partire da qui.

L'altro prerequisito, molto importante, perchè è essenziale per il funzionamento della creazione degli account, è quello di avere sempre all'interno dell'account master **un trail CloudTrail non default**, dato che verrà utilizzato da ADF per ispezionare gli eventi.

Setup

Per installare ADF basta seguire la guida all'interno del repository di ADF.

Potrebbe succedere che per via del costante aggiornamento delle versioni del codice sorgente all'interno del Serverless Application Repository ci si possa imbattere in release poco stabili. Ad esempio, il bootstrap degli account potrebbe fallire a causa di mancate dipendenze. Per risolvere questa problematica, abbiamo seguito una procedura alternativa utilizzando direttamente il repository su GitHub e i seguenti passi:

```
git clone https://github.com/awslabs/aws-deployment-framework.git # clone del codice
git checkout tags/v3.1.2 # revert alla versione 3.1.2
git cherry-pick -X theirs 0f971387e9a756a62719cc3be63a41fb8f370912 # cherry pick a commit con dipendenze corrette
git cherry-pick –continue # finalizzazione cherry pick
```

Come Funziona

Componenti

Per gestire i deployment sull'organization ADF sfrutta principalmente 2 account con mansioni specifiche, ovvero:

- Master Account
- Deployment Account

Master account

Il Master Account è l'unico account abilitato a contattare le API di AWS Organization, quindi è l'unico che puó creare account e organizational unit.

Il processo di creazione di un account parte dal repository che troviamo all'interno del Master account.

Navigando il servizio **AWS CodeCommit** nella region **us-east-1** troveremo il repository **aws-deployment-framework-bootstrap**. A questo punto, dovremo scrivere il codice necessario per la creazione dello scenario multi-account. Per il deployment verrà invocata una pipeline del servizio **AWS CodePipeline** che si occuperà di chiamare le API di Organization per creare gli account precedentemente definiti nel repository.

I passi eseguiti dalla pipeline prevedono la creazione dell'account e lo spostamento dello stesso dalla root organization alla Organzational unit che abbiamo indicato nel repository. Questo passaggio di spostamento verrà intercettato da una **AWS CloudWatch Event Rule** in ascolto sull'evento di tipo **MoveAccount**. La regola invocherà una **AWS Step Function** che si occuperà di creare o aggiornare gli stack CloudFormation di base dell'account appena creato e di quelli che creeremo.

Infine verrà invocata un'altra Step Function all'interno dell'account di Deployment che analizzeremo nel prossimo paragrafo.

Deployment account

Il deployment account viene creato da ADF durante il setup ed è colui che si occuperà del deployment di tutti i servizi e di tutte le risose all'interno della nostra Organization.

Affinché l'account di deployment possa gestire gli account figli, necessiterà di ruoli IAM che siano federati con esso permettendo l'azione di assume role. Per abilitare gli accessi cross-account, alla creazione di un nuovo account verrà invocata una **AWS Step Function** che aggiornerà/creerà i ruoli utilizzati da ADF per accedere all'account in questione.
Essendo un framework di deployment, avrete già capito che questo account è centrale, dato che da questo singolo punto avremo modo di creare innumerevoli pipeline in pochi click; queste pipeline verranno a loro volta create da ADF tramite l'utilizzo di un **AWS CodeCommit** repository e una **AWS CodePipeline.**
All'interno del repository troveremo una cartella *deployment_map* con al suo interno le definizioni delle pipeline configurate e messe in opera nell'account di Deployment. Una volta effettuato il push dei commit contententi le deployment_map, verrà fatta partire una pipeline che avrà come scopo la creazione di altre pipeline che rispecchino le deployment map nel repository: in sostanza una pipeline di pipeline :)

Esempi di utilizzo

Creazione Account

Come abbiamo detto in precedenza, per creare un account tramite ADF dovremo lavorare sull'account master dell'Organization, in quanto è l'unico che è abilitato a chiamare le API di Organization.

Una volta sull'account master della Organization, nella region us-east-1 e sul servizio CodeCommit, dovremo clonare il repository di codice chiamato *aws-deployment-framework-bootstrap* e immettere all'interno della cartella *adf-accounts* un file simile a questo:

```
accounts:
  - account_full_name: deployment
    organizational_unit_path: /deployment
    email: adf@proud2becloud.com
    alias: deployment
    tags:
      - created_by: adf
```

Una volta creato questo template, la pipeline effettuerà il pull del codice, farà i test, e chiamerà le API di Organization per:

- Creare l'account con nome e alias deployment;
- Metterlo all'interno dell'organization unit deployment;
- E dargli come mail di notifica adf@proud2becloud.com.

Non appena completata questa procedura, partirá la Step Function che creerà al suo interno gli stack CloudFormation di base per poi invocare la Step Function all'interno dell'account di deployment per abilitare gli accessi cross-account dagli account figli. Questo meccanismo permette di fare in modo che meno persone/servizi possibili abbiano accesso all'account master, seguendo quindi i canoni del **least privilege principle** e di **zero trust**.

Una volta finita questa ultima step function verranno lanciate tutte le pipeline legate all'Organizational unit **deployment**. Cosi facendo, ADF automatizzerà non solo la creazione di un account, ma anche la configurazione dei servizi al suo interno.

Creazione di una pipeline

Mettiamo caso che si voglia una pipeline che crei una VPC in tutti gli account all'interno della organizational unit deployment.

Per farlo, dovremo andare nell'account di deployment, sul repository *aws-deployment-framework-pipeline*, entrare nella sotto cartella *deployment_maps* e creare un file simile a questo:

```
pipelines:
  - name: vpc
    default_providers:
      source:
        provider: codecommit
        properties:
          account_id: 111111111111 # account con repository di codice
      build:
        provider: codebuild
        properties:
          environment_variables:
            CONTAINS_TRANSFORM: True
      deploy:
        provider: cloudformation
        properties:
          action: replace_on_failure
    params:
      notification_endpoint: adf+deployment@proud2becloud.com
    targets:
      - path: /deploymnet
        regions: eu-west-1
```

Questa è la definizione di una pipeline per ADF in cui basterà specificare su che account troverà il repository (source), che tipo di deploy utilizzare (deploy) e su che account andare a fare il deploy (target). Al push del file di configurazione partirà la pipeline di ADF che, a sua volta, creerà la pipeline sull'account di deployment a partire da questo file di configurazione.

Se il repository vpc non si troverà all'interno dell'account identificato dal proprio id (es. 111111111111) allora sarà la pipeline a provvedere alla sua creazione. Una volta fatto, andrà effettuato il push all'interno del repository appena creato con il codice, ovvero, il template CloudFormation e i parametri.

Il template CloudFormation è standard, mentre i parametri, a differenza del normale utilizzo di CloudFormation, possono essere scritti usando un linguaggio YAML.

```
Parameters:
  VpcCidr: 10.0.0.0/16
  DeployNat: true
```

Una cosa importante riguardo i parametri é il nome del file, dato che **ADF utilizza il nome dell'account e la region** in cui vogliamo effettuare il deploy per comporre il nome del file di parametri.

Mettiamo caso che uno degli account dentro alla organization unit si chiami adf-dev e vogliamo effettuare il deploy in Irlanda; il file di parametri dovrà chiamarsi adf-dev_eu-west-1.yml.

Una volta che il deploy sarà completo, ogni account che verrà messo all'interno della organizational unit /deployment avrà lo stack della VPC, anche successivamente al primo deploy.

Conclusioni

L'AWS Deployment Framework è uno strumento relativamente nuovo che, pur dimostrando di avere ancora margini di miglioramento, sta già rivoluzionando il modo di gestire e deployare risorse, servizi e infrastrutture all'interno di una Landing Zone su AWS.

In questi mesi abbiamo testato il servizio con diverse complessità, da infrastrutture con pochi account, fino a Landing Zone con centinaia di account distribuiti su più region, arrivando a identificare quelli che, secondo noi, sono i pro e i contro da tenere in considerazione quando si deve decidere quale strumento utilizzare per il nostro use case.

È bene ricordare che non esiste la soluzione migliore in assoluto: occorre sempre fare un'analisi approfondita di ciascun caso specifico. Speriamo che questo nostro punto di vista possa semplificarvi in questo!

E voi avete già utilizzato ADF all'interno della vostra Landing Zone? Raccontatecelo nei commenti!
Ci vediamo tra 14 giorni su Proud2beCloud per un nuovo articolo

---

About Proud2beCloud

Proud2beCloud è il blog di beSharp, APN Premier Consulting Partner italiano esperto nella progettazione, implementazione e gestione di infrastrutture Cloud complesse e servizi AWS avanzati. Prima di essere scrittori, siamo Solutions Architect che, dal 2007, lavorano quotidianamente con i servizi AWS. Siamo innovatori alla costante ricerca della soluzione più all'avanguardia per noi e per i nostri clienti. Su Proud2beCloud condividiamo regolarmente i nostri migliori spunti con chi come noi, per lavoro o per passione, lavora con il Cloud di AWS. Partecipa alla discussione!

Manuel Petrungaro

Solution Architect @ beSharp. Progetto e implemento infrastrutture complesse sul Cloud.Amo gli scacchi e faccio parte del Club "Computer Knowers"
