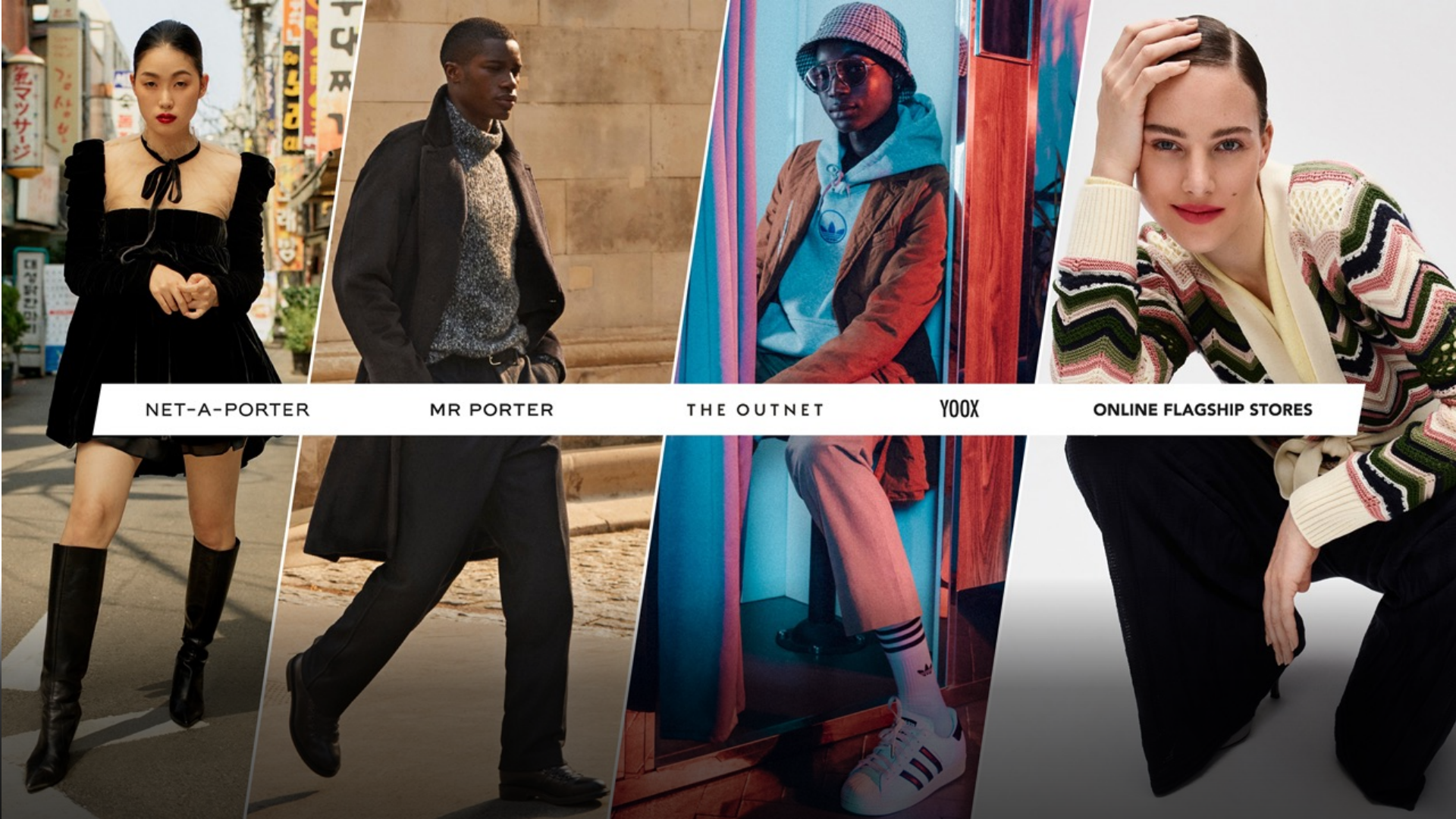
Evoluzione della gestione dell'infrastruttura cloud: il nostro percorso verso il self-service (con AWS)

Platform Engineering Milano Meet-Up – 13 Luglio 2023

Giacomo Grangia – Cloud Automation Engineer

Platform Engineering


NET-A-PORTER
MR PORTER
THE OUTNET
YOOX
ONLINE FLAGSHIP STORES

Chi sono

Giacomo Grangia

Cloud Engineer @ Yoox Net-A-Porter

giacomo.grangia@ynap.com

https://www.linkedin.com/in/giacomo-grangia/

Agenda

- Di cosa mi occupo?
- Le origini
- ADF in breve
- ADF in Ynap
- Verso il self-service
- Conclusione: cosa abbiamo imparato e next steps

Di cosa mi occupo?

- Faccio parte del team **Cloud Foundation**
- Fornire le risorse basilari e necessarie (spesso critiche) che permettono di «costruire» su AWS
- Creazione di account AWS
- Gestione Organization
- Layer di networking (VPC, TGW, peerings, R53, etc...)

Le origini

- Team centralizzato
- 1 repository per VPC (nello stesso progetto)
- Codice non uniforme tra i vari repo
- Poca modularità
- Impossibilità di eseguire cambiamenti su tutte le VPC
- Copy & Paste di snippet di codice
- Basato sui ticket creati manualmente
- Automazione custom per creare accounts
- Durata processo creazione (account + VPC): settimane (3+)

Developers

Infra Team

I nostri obiettivi

- Controllo sul codice
- Automazione per creare account
- Supporto multi-account (Landing Zone)
- Possibilità di deployare risorse su TUTTA la nostra Organizzazione
- Facilità di estensione ed utilizzo
- Sicurezza
- Supporto per il codice già in nostro possesso (principalmente terraform)
- Facile onboarding con Organizations già esistenti

La scelta

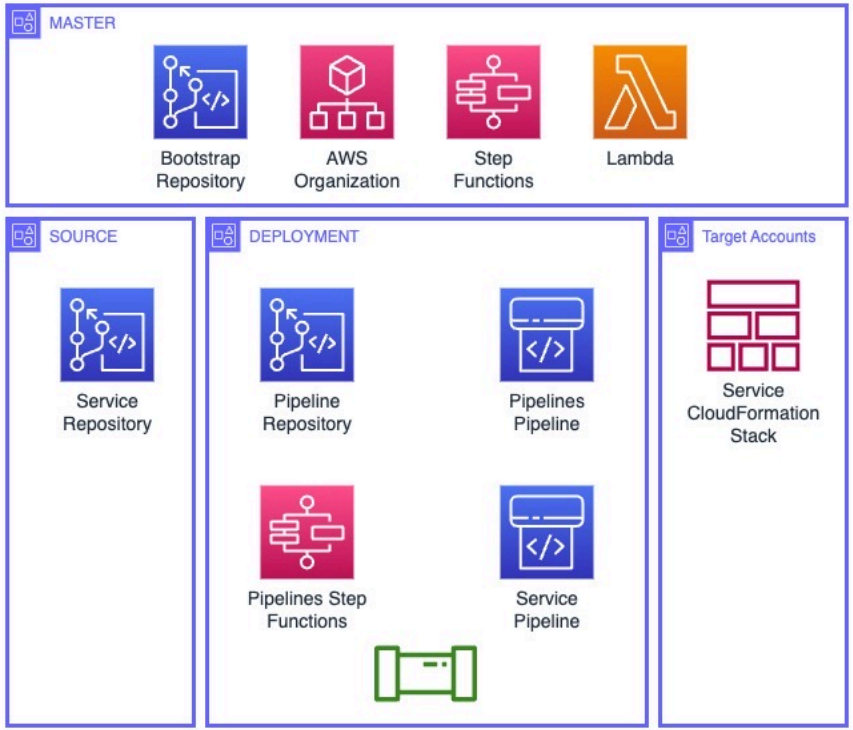
- Alcuni prodotti erano troppo immaturi (AWS control Tower)
- Altri sono stati deprecati poco dopo (AWS Landing Zone)

Amazon Deployment Framework

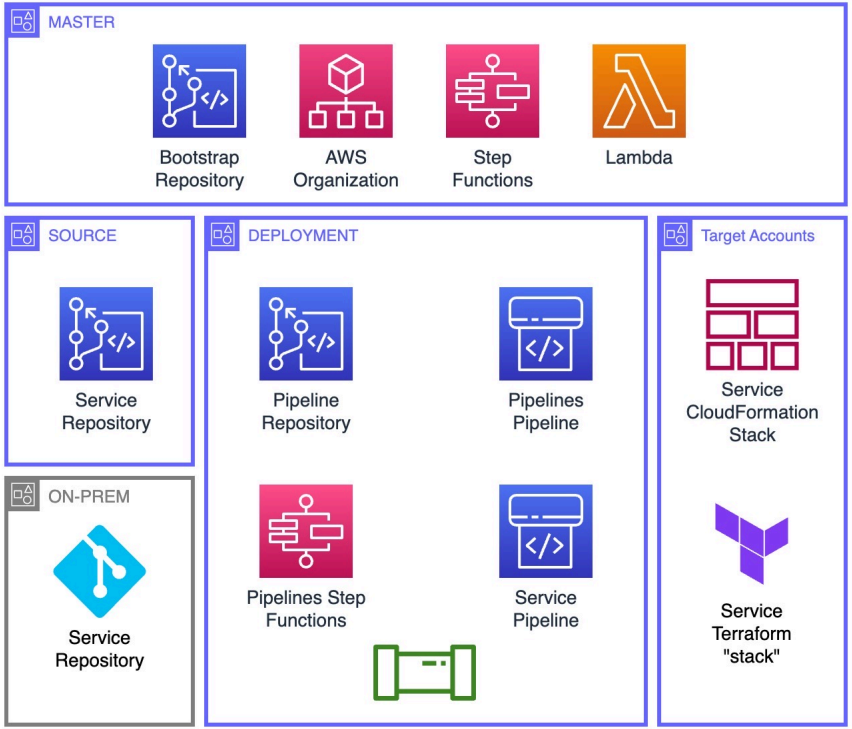
ADF in breve

- Framework opinionato per gestire e creare risorse tra più account e regioni nella stessa Org
- Permette definizione di pipeline cross-account con OUs (o accounts) come target in maniera centralizzata
- Facile creazione di nuovi account
- Basato sui prodotti CICD di AWS (CodePipeline, CodeBuild, CodeDeploy, etc...)
- Integrazioni con altri prodotti (e.g Github come Source e Jenkins come Build)
- Supporto CloudFormation out-of-the-box
- Supporto a pipeline con relazione uno-a-molti (più Pipeline basate su un repository)
- Facile condivisione nelle pipeline di script custom

ADF in breve – Landing Zone



ADF in breve – Landing Zone in YNAP



ADF in YNAP – integrazioni

BitBucket (self-hosted on-prem)

- Integrazione in Cloudformation rilasciata con ADF
- Basata su webhook (push, merge, etc)
- Archivia il repository e lo carica su S3
- S3 «source» delle pipelines

Terraform

- Custom script usato negli stage di Build
- Complessità nel gestire il parallelismo
- Gestione statefiles (S3) e locktable (DynamoDB)

Primi successi con ADF

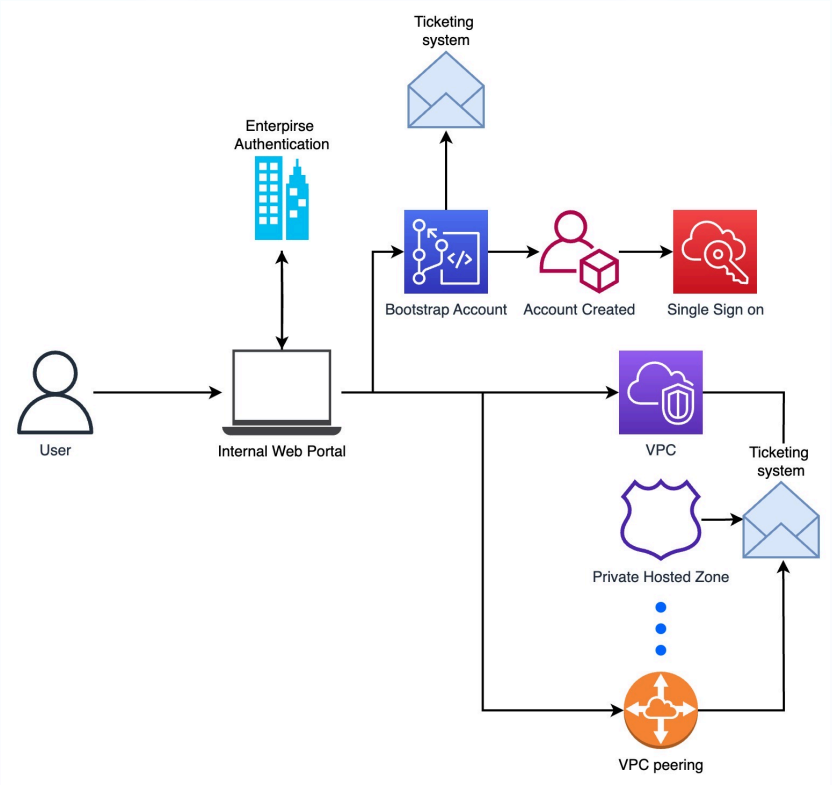
- Obiettivo comune >- sforzo cross-team per maggiore documentazione, collaborazione e processo
- Rapidità nel creare account -> ADF impiega pochi minuti (la soluzione custom falliva spesso)
- Separazione logica degli account grazie alle OU -> facile cambio di OU
- Buone performance nel deploy dello stesso codice su tutti gli account (e.g. IAM role)
- Self-service gitOps con PR approvate dal nostro Team (siamo owner del repository)

- Durata processo creazione account: settimane (1-2)
- Processo ancora basato su molti ticket
- Alcuni malumori per il metodo gitOps

Sfide di ADF

- Riduzione di performance nella pipeline di bootstrap
- Riduzione di performance nella pipelines di «deployment maps» (versioni precedenti)
- Difficoltà nel debug di «deployment maps» (tante Step functions)
- Limiti di CodePipeline (10 stages, 50 actions)
- Processo di update manuale con PR enormi
- Possibili incompatibilità tra update e customizzazioni
- Molte step-functions e script per tenere assieme il framework
- CDK v1.0 su SAR
- Pipeline utilizzano polling (source S3)

Portale self-service



Successo del portale self-service

- Processo quasi interamente automatizzato
- Ticket «automatici» per gestione centri di costo e reporter
- Alcune dipendenze rimangono manuali (gestite da altri team)

- Durata processo creazione account: giorni (1-4)
- Aggiunta di un altro «entrypoint» per il developer

E le altre risorse?

- Procedura gestita da DevOps team
- Bootstrap di pipelines e repo con codice di esempio
- Gli sviluppatori sono i proprietari del repo
- Libertà di personalizzazione del codice
- Pensato per update individuali

Conclusione

- Processo manuale ticket-based -> self-service gitOps -> portale self-service
- Tutta l’azienda deve sforzarsi per supportare una piattaforma self-service
- Evitare di creare «self-service silos»

Grazie!

giacomo.grangia@ynap.com

https://www.linkedin.com/in/giacomo-grangia/