Cav Analisi - Report

Luca Pitzoi

2023-06-26

Premessa

Lo scopo di questo documento non e' criticare il lavoro di nessuno, ma prendere in considerazione lo stato dell'arte del processo e delle metodologie utilizzate sul progetto, individuare le criticita' e proporre delle soluzioni considerando paradigmi e pattern moderni del mondo DevOps come IaC (Infrastructure as Code) e gitOps.

Questo report non pretende di essere esaustivo, dato il recente approdo nel gruppo da parte del sottoscritto, potrebbe includere quindi imprecisioni, tuttavia si ritiene di aver portato alla luce delle evidenze oggettive e, che i principi tratti dalla letteratura, siano non solo utili alla causa, ma giusti e necessari.

Metodo

Benche' ci si trovi spesso a parlare di problem solving, e' fondamentale fare un passo indietro e parlare di problem setting.

Qual'é il nostro problema?

Una volta calato il problema sul contesto si e' proceduto con l'analisi delle possibili soluzioni.

Il metodo utilizzato lo si puo' riassumere cosi':

- Fase teorica: analisi e raccolta informazioni comunicando con i colleghi della struttura.
- Fase sperimentale: porting della soluzione ingegnerizzata e implementata su Moova utilizzando alcuni componenti campioni.

Problema

Nel momento della stesura di questo documento il problema evidenziato dal punto di vista operativo si riassume nei seguenti punti:

- Oggettiva incapacita' di mantere gli ambienti allineati e consistenti.
- Scarsa automazione e mal ingegnerizzata.
- Processo non chiaro e definito su tutte le fasi della gestione del ciclo di vita del software.
- Alto overload sulle risorse dedicate alla gestione dei rilasci e su processi manuali.

Processo

In questa sezione viene rappresentata una fotografia della situazione attuale e del processo in uso.

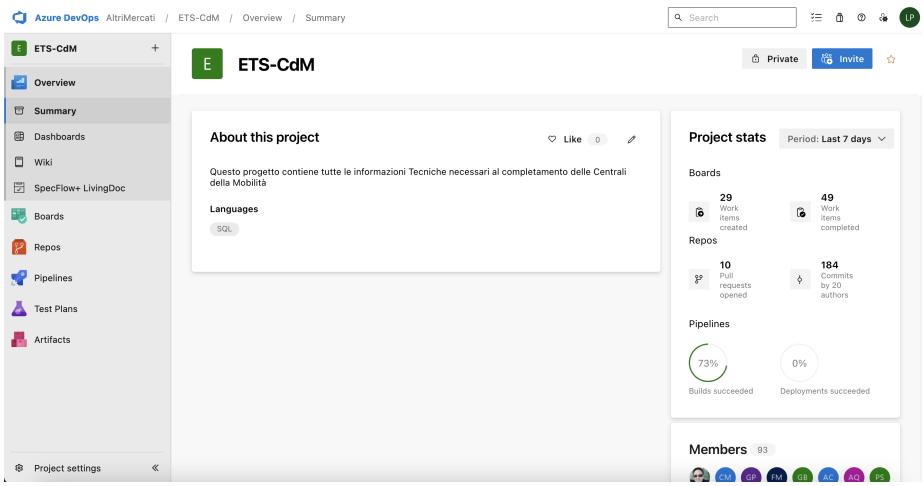

Figure 1: Strumento Azure DevOps

—

—

—

—

—

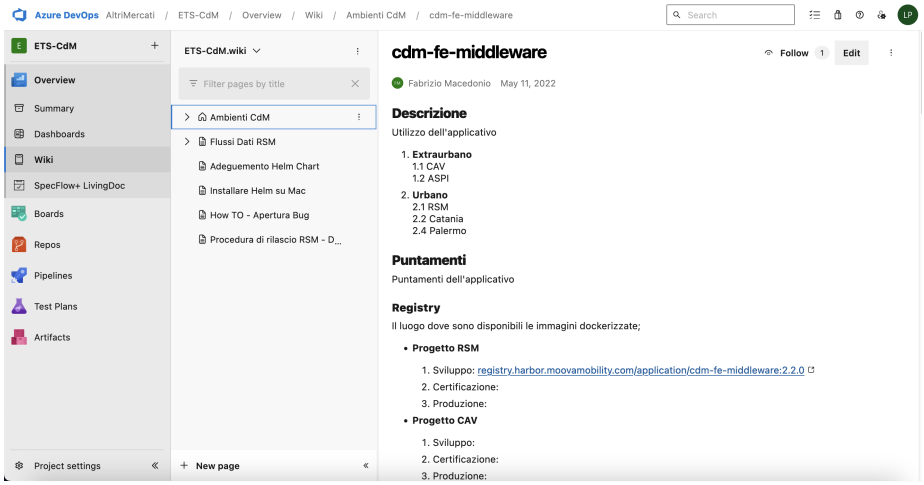

Figure 2: Ambienti Cdm

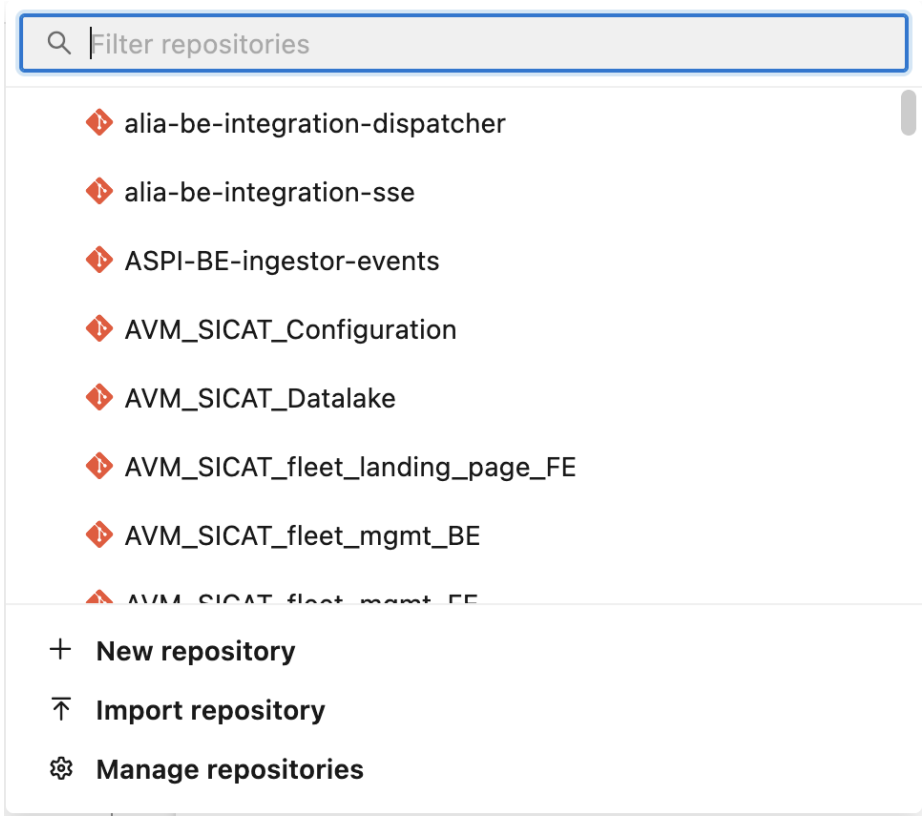

Figure 3: Nomenclature

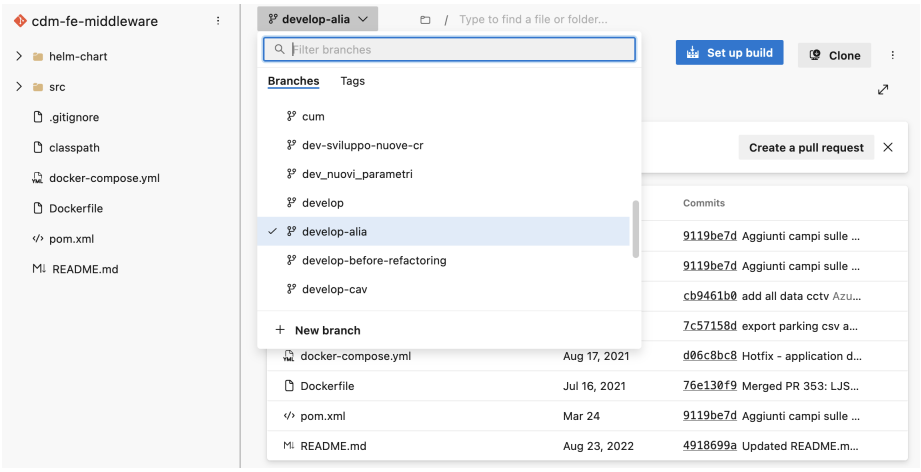

Figure 4: Branching Model

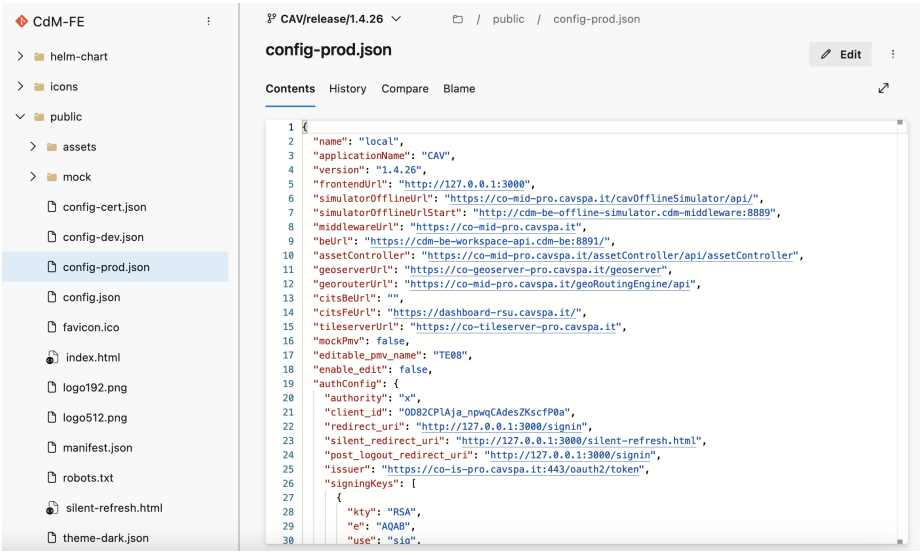

Figure 5: Configurazioni hardcoded

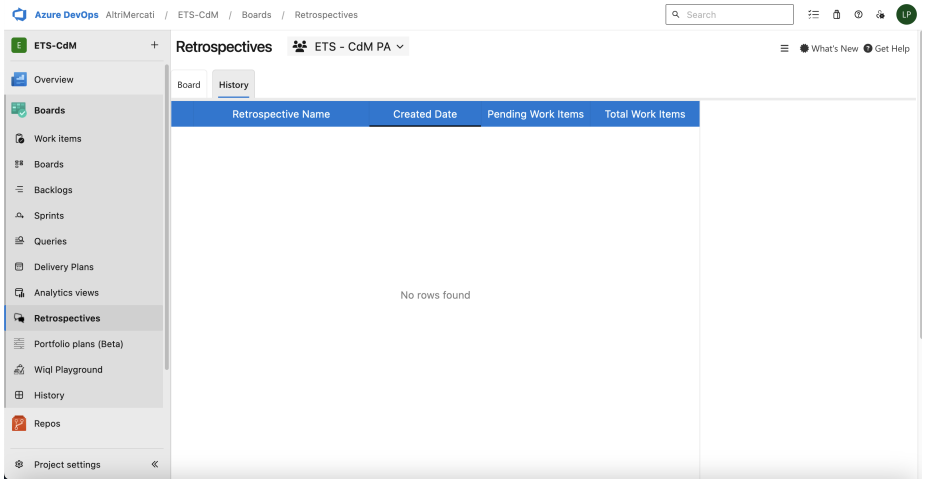

Figure 6: Retrospettiva

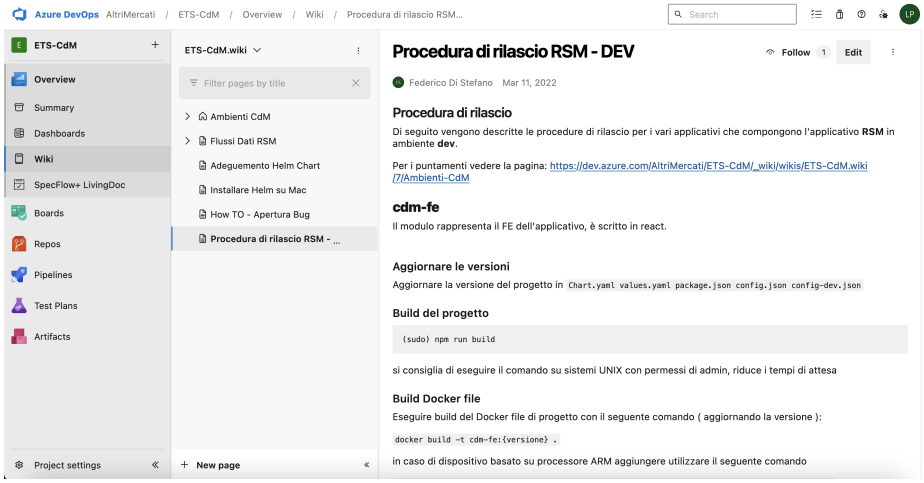

Figure 7: Procedura di rilascio parte 1

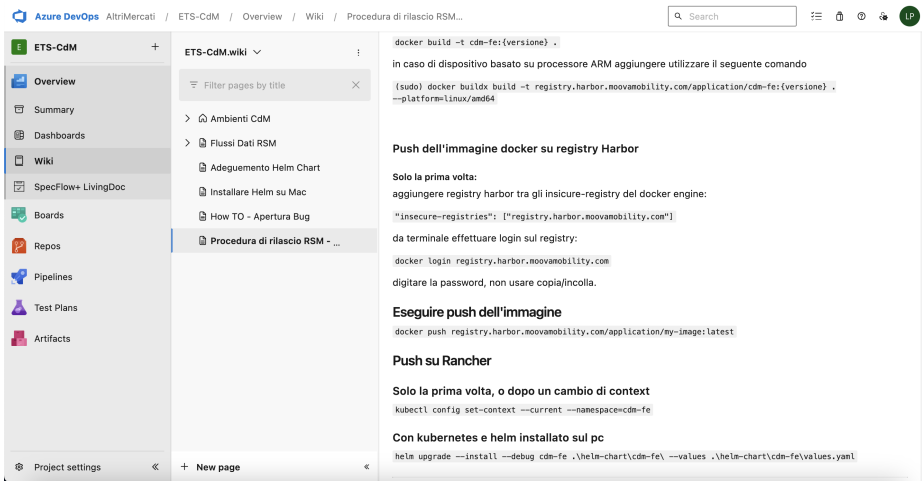

Figure 8: Procedura di rilascio parte 2

Deployment

- Namespace di riferimento in K8S è **XXXXXX**

Istruzioni per deployare

```
1.  docker build -t almaviva.jfrog.io/artifactory/moova-ets/mobility/XXXXXXXX:1.X.X .
2.  docker push almaviva.jfrog.io/artifactory/moova-ets/mobility/XXXXXXXX:1.X.X

Ambiente RSM
1. docker pull registry.harbor.moovamobility.com/application/XXXXXXXX:x.x.x
2. docker tag registry.harbor.moovamobility.com/application/XXXXXXXX:x.x.x  registry.romamobilita.it/cc
3. docker push registry.romamobilita.it/ccr-iot-col-k8s/XXXXXXXX:x.x.x


Ambiente CAV
1. docker pull registry.harbor.moovamobility.com/application/XXXXXXXX:x.x.x
2. docker tag registry.harbor.moovamobility.com/application/XXXXXXXX:x.x.x  registry.romamobilita.it/cc
3. docker push registry.romamobilita.it/ccr-iot-col-k8s/XXXXXXXX:x.x.x
```

Contenerizzazione

```
4. kubectl create namespace situation
5. kubectl config set-context --current --namespace=XXXXXXXX
6. kubectl create secret docker-registry rsm-registry --docker-server=harborcollaudo.romamobilita.it --
7. helm upgrade --install --debug XXXXXXXX .\XXXXXXXX --values .\XXXXXXXX\values.yaml
```

Figure 9: Deploy parte 1

Build and Test

1. npm run start
2. npm run build

Deployment

- Namespace di riferimento in K8S è **cum-fe**

Istruzioni per deployare

1. docker login almaviva.jfrog.io/artifactory/moova-ets/ --username=moova-ets --password=Moova-ets-01
2. docker build . -t almaviva.jfrog.io/moova-ets/mobility/cdm-fe/cum-fe:x.x.x
3. docker push almaviva.jfrog.io/moova-ets/mobility/cdm-fe/cum-fe:x.x.x
4. helm upgrade --install --debug cdm-fe .\helm-chart\cdm-fe\ --values .\helm-chart\cdm-fe\values.yaml -n cdm-fe

Andare in rancher:

5. Nella configMap del pod CUM-FE cambiare la versione con quella appena deployata.
6. Nel pod CUM-FE cambiare la versione con quella appena deployata.

Contenerizzazione

Figure 10: Deploy parte 2

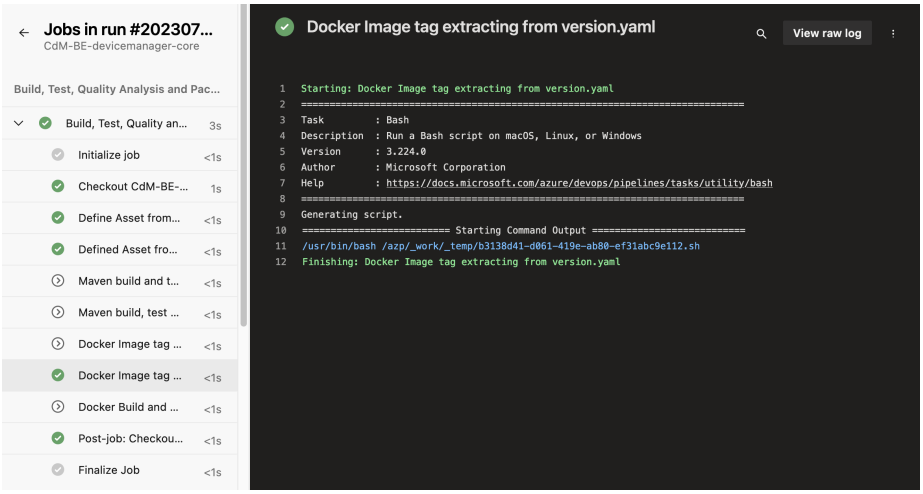

Figure 11: Pipeline parte 1

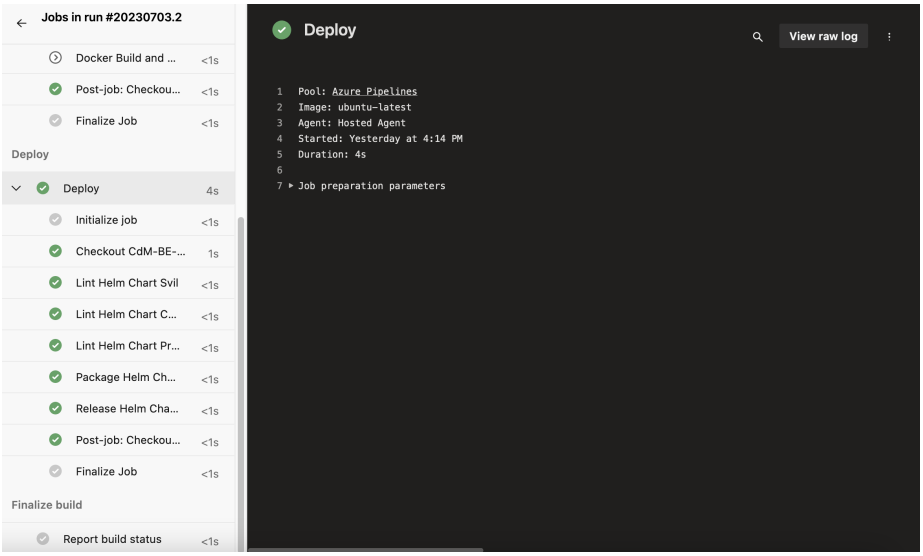

Figure 12: Pipeline parte 2

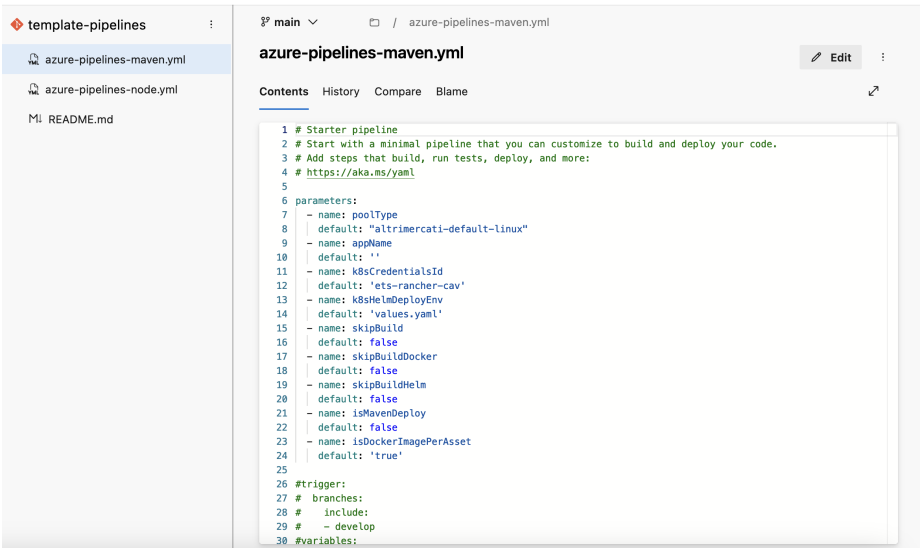

Figure 13: Pipeline template

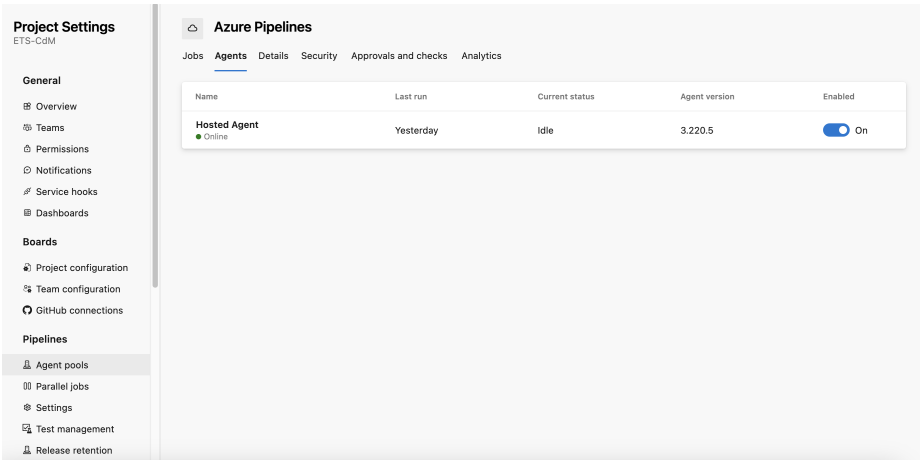

Figure 14: Agents Pool

Criticita'

- Implementazione metodologia Agile incompleta e con antipattern
- Sprint planning? Roadmap? Restrospettive? Daily Standup?
- Risorse junior lasciate senza guida non ben formate.
- Branching model e versionamento semantico confusionario e inefficace
- Codice duplicato, fork continuo e discrepanza tra i vari componenti - nessun riutilizzo del codice
- Standardizzazione compomenti architetturali e alberatura codici sorgenti mancante e/o parziale
- Automazione non igegnerizzata, incompleta e molti step manuali
- Pipeline incomplete, manca encapsulamento e step disabilitati (settati come skippabili)
- Template delle pipeline non igegnerizzati (starter template di esempio con delle pezze)
- Codice nelle pipeline hadcoded dentro yaml - Non mantenibile
- Configurazione cluster k8s, namespace, infrastruttura e helmizzazione da rivedere
- Ambienti di sviluppo locali e containerizzazione non standard
- Standardizzazione, censimento assets e nomenclature inesistente

- Valori hardcoded - configuration management e policy non definiti
- Gestione secrets mancante
- Utenze nominali e di servizio da definire
- Database automation - ora tutto gestito tramite script DDL e dump manuali

Soluzione

La soluzione proposta si basa sul background e sulle esperienze pregresse del sottoscritto in altri contesti simili. Inoltre si basa su una soluzione gia' ingegnerizzata e implementata in ambito Moova dove la situazione di partenza era analoga se non identica.

Di seguito si proporrano le varie soluzioni per tutti i vari punti critici individuati ed in seguito vedremo anche una proposta di pianificazione per tali attivita'.

Per la maggior parte dei punti citati nella sezione "Criticita'" vengono risolti in automatico, tuttavia occorre precisare che ci sono alcuni punti critici che vanno smarcati per primi in quanto prerequisiti, come ad esempio le nomenclature dei compomenti che devono assolumentante essere uguali in tutte le fasi (repository, artefatto, immagine, pod)

Per una visione di dettaglio della soluzione Moova si rimanda al pptx allegato insieme a questo documento.

Pianificazione

- Milestone 0: Discovery and Analysis
- Milestone 1: Revisione e standardizzazione (nomenclature, branching model, metodologia agile, ambienti, configurazioni)
- Milestone 2: Migrazione componenti standardizzati ai template PaC
- Milestone 3: Revisione helm su nuovo stack e generalizzazione values
- Milestone 4: Implementazione soluzione database automation basata sui pattern migrations and seeders
- Milestone 5: Implementazione integration tests tramite kind sulle pipeline

Conclusioni

Soluzioni implementate orientate al risultato immediato senza visione della big picture che portano debito tecnico

Fonti

- https://azure.microsoft.com/it-it/overview/what-is-devops/#devops-overview
- https://www.redhat.com/it/topics/devops
- https://it.wikipedia.org/wiki/Pipeline_software
- https://www.redhat.com/it/topics/devops/what-cicd-pipeline
- https://docs.microsoft.com/it-it/azure/architecture/example-scenario/apps/devops-dotnet-webapp

Bibliografia

- F. Mora, "DevOps: Guida per integrare Development e Operations e produrre software di qualita'", 2019
- Robert C. Martin: "Clean architecture. Guida per diventare abili progettisti di architetture software", 2018
- A. Stellman, J. Greene: "Learning Agile: Understanding Scrum, XP, Lean, and Kanban", 2014
- E. Gamma, R. Helm, R. Johnson, J. Vlissides: "Design Patterns: Elements of Reusable Object-Oriented Software", 1994

Note

- Cultura DevOps (Guidelines e standardizzazione)
- Agile
- Pipeline as Code
- Automated Tests (lint, unit tests, QA, integration tests, security analysis, security scanning, penetration tests, load tests, end-to-end)
- Infrastructure as Code (IaC)
- Mutable IaC
- Immutable IaC
- Configuration Management (CM)
- gitOps
- noOps Pipeline
- Monitoring e Alerting
- Backup and Restore
- Security Assessment
- Comunicazione