Chas Academy
DOE21
Examensarbete 2023

Examinationsarbete

Av Erik Olsson och Sara Petré

Handledarens namn: Per Sundqvist

Abstract

In this report we have worked on developing a continuous delivery system for our Jira_Test_Management_Migrator (Also known as JTMM). We have written Terraform code, Ansible code, and a GitHub Actions pipeline in order to perform a migration automatically. We successfully managed to develop this continuous delivery system and get it up and running.

We have been working in an agile environment throughout the project with support from Solidify AB. Working agile has given us more training in agile methods which gives us good tools to use in future work.

Innehållsförteckning

1. Inledning

Vi har under vår LIA period på Solidify AB utvecklat ett migreringsverktyg (Jira_Test_Management_Migrator, eller JTMM) för att migrera testfall samt testdata från olika extensioner i Jira (Qmetry, Zephyr, Xray, etc) till Azure DevOps. Vårt examinationsarbete är ett experiment för att se hur verktyget potentiellt kan användas i produktion.

1.1 Bakgrund

Vi har under en längre tid spekulerat över hur man kan använda vårt migrations verktyg i en större produktion och bestämt oss för att testa att bygga en produktionsmiljö för verktyget. Vår kravspecifikation är följande:

- Migrera test data från Zephyrs Jira Extension till Azure DevOps
- Skapa en VM i Azure med hjälp av Terraform
- Sätta upp en migreringsmiljö i VM;en på Azure med hjälp av Ansible (Installera docker, ladda hem container image;n av migreringsverktyget, starta containern, kör migreringsverktyget, stäng av VM när migreringen är klar (optional))
- Bygga en pipeline i valfritt verktyg som testar att migreringsverktyget fungerar och sedan startar terraform scriptet som sätter igång allt

1.2 Syfte

Syftet för projektet är att se hur lämpligt det är att distribuera vårt migrations verktyg i en automatiserad miljö med hjälp av Terraform och Ansible. Detta för att kunna hitta olika affärsområden man kan applicera verktyget till.

1.3 Avgränsning

Det var viktigt för oss att skriva en kravspecifikation som vi kände att det fanns goda möjligheter till att lyckas med, inom den tidsramen vi hade till vårt förfogande.
Det resulterade i ett relativt kort projekt men som har många delar i sig för att täcka de olika kunskapsmålen. Under projektets gång har vi ändå fått stryka ett krav: Att testa docker imagen i pipelinen. Detta då det är oerhört svårt och inte skulle hinnas med inom tidsramen för examensarbetet. Vi använder oss även av Azure i projektet då detta är den resurs som vi har fri tillgång till genom vår LIA på Solidify.

2. Metod och genomförande

Det agila arbetssättet:

Projektet har drivits med hjälp av ett agilt arbetssätt. En board har skapats med backlog som har gåtts igenom med produktägare. En sprintplanering har gjorts för arbetet. Agila delar som vi har anammat är bland annat dagliga möten samt veckomöte med produktägare då vi har haft ett retrospektiv på veckan som har varit och sett över vägen framåt.

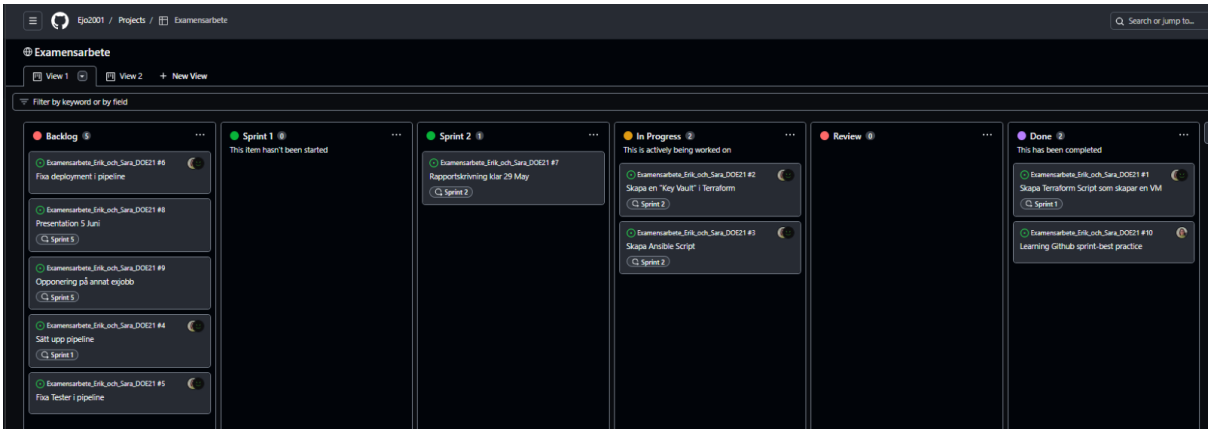

Bild 1. Board som använts. Boarden finns tillgänglig i GitHub-repot under "projects", se bild nedan.

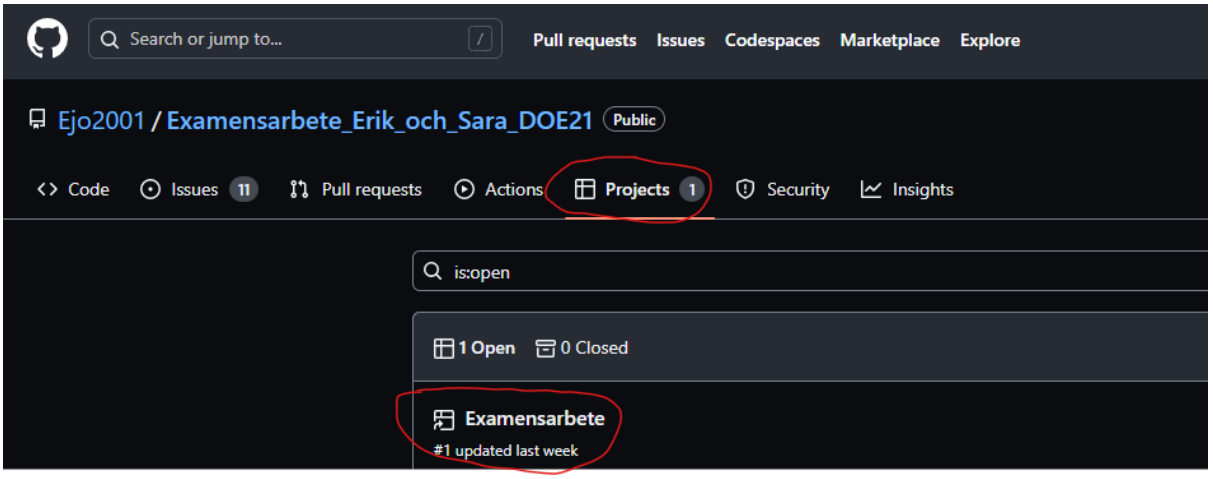

Bild 2. Vägen in till boarden.

Strukturering av vårt projekt:

Projektet har arbetats på i en viss följdordning. Vi började med att göra en mindre pipeline för att undersöka hur GitHub Actions fungerar och kunna se hur vi använder det till vårt projekt.

När vi bekantat oss ett tag med GitHub Actions så började vi skriva på Terraform filen. Terraform scriptet har i uppgift att:

1. Skapa en resource group i Azure
2. Skapa upp en Virtuell Maskin tillsammans med det som den behöver för att användas (Publik IPv4 adress, VLAN, Network Interface)
3. Skapa en key vault som används för att hämta och lagra åtkomst och autentiserings nycklar

Efter att vi fått Terraform koden att fungera så började vi skriva på våran Ansible kod. Terraform sparar ner IPv4 adressen som sedan skickas till Ansible. Ansible filen har i uppgift att:

1. Logga in och exekvera kod på den Virtuella Maskin som skapats i Azure av Terraform
2. Uppdatera systemet så att de senaste paketen kan hämtas till maskinen
3. Installera Docker så att vi kan köra våran Docker Image som byggts för migreringsverktyget, samt installera “jq” för att kunna sortera ut våra nycklar från key vault
4. Installera och Logga in på Azure CLI
5. Hämta nycklarna från key vault och spara ner dem i systemvariabler samt .env filen som krävs för att köra en Docker Container med vår image
6. Logga in på Github Docker Registry för att kunna hämta vår Docker Image
7. Starta igång en migrering med hjälp av “Docker run” kommando.

Sist av allt så skrev vi pipelinen i Github Actions. Den utför kommandon i följande ordning:

1. Logga in på Azure CLI med hjälp av en service principal
2. Uppdatera systemet samt installera Terraform
3. Installera Ansible så att vi kan använda Ansible-Playbook
4. Hämta och spara ner filerna från projektet ner till pipelinen
5. Starta och förbered Terraform
6. Kör Terraform och börja skapa Azure resurser
7. Ladda upp information till Azure Key Vault
8. Ställ in rättigheterna till SSH nyckeln innan Ansible körs
9. Starta Ansible och köra migrerings koden
10. Stäng ner Azure resurserna

Under projektets gång så har vi behövt olika typer av åtkomstnycklar och SSH nycklar. Vi har genererat en publik och en privat SSH nyckel som ligger i projektmappen. Detta är inte optimalt då det säkerhetsmässigt inte är acceptabelt att lagra hemliga filer i ett publikt projekt. Dock motiveras det med att dessa endast är nycklar som används under exekveringen och inte utsätter projektet för någon större risk. För att få till key vault i Azure så har vi fått använda oss av RBAC-autentisering. Vi har skapat en service principal med RBAC som sedan använts till att logga in och komma åt våran Azure Key Vault.

Svårigheter och begränsningar:

Några svårigheter som kommit upp har varit med autentisering och överföring av nycklar. Vi har fått lösa detta på lite olika icke önskvärda sätt i vissa fall, men vi har till slut nått fram till slutdestinationen i arbetet.

Vi har även fått stryka tester av Docker Imagen som stod i vår kravspecifikation. Detta då Docker inte ger utslag på och migreringen fungerar eller ej, samt att vi inte lyckas få Github Actions att använda vår image (Då många av de funktioner vi velat använda har blivit utfasade). Det kanske går att utveckla något test i form av att kolla det som ligger i Azure DevOps och jämföra före och efter migreringen, alternativt läsa loggen på docker och försöka hitta ett mönster, men inget av alternativen passar inom tidsramen angiven för detta projekt.

4. Resultat

Resultatet är positivt i projektet. Vi lyckades sätta upp majoriteten av det vi ville och fick det att fungera. Migreringen slutfördes och projektet städade upp resurserna i Azure så att vi inte behövde spendera tid på att stänga ner dyrbara resurser.

Nedan är några bilder från migreringen:

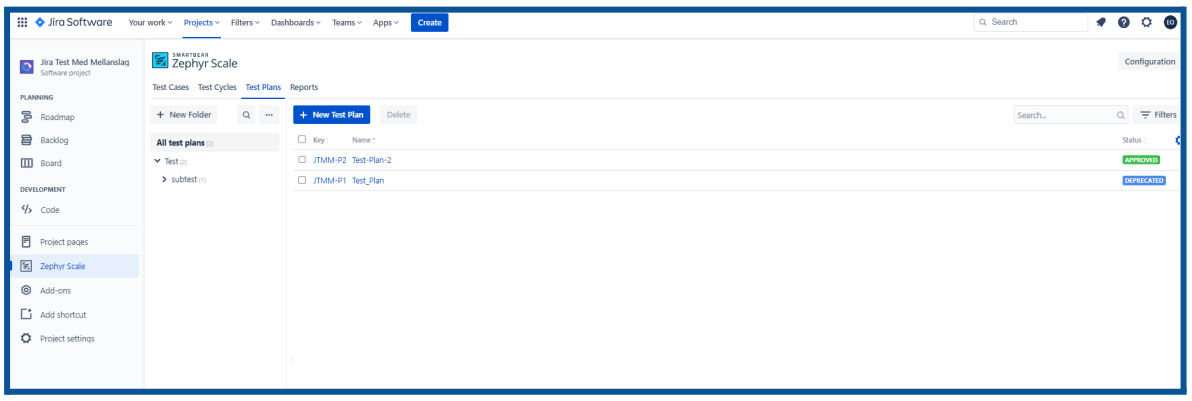

Bild 3. Zephyr projektet i Jira

Bild 4. Tomt projekt i Azure DevOps. Det är till det här projektet objekten från Jira ska migreras.

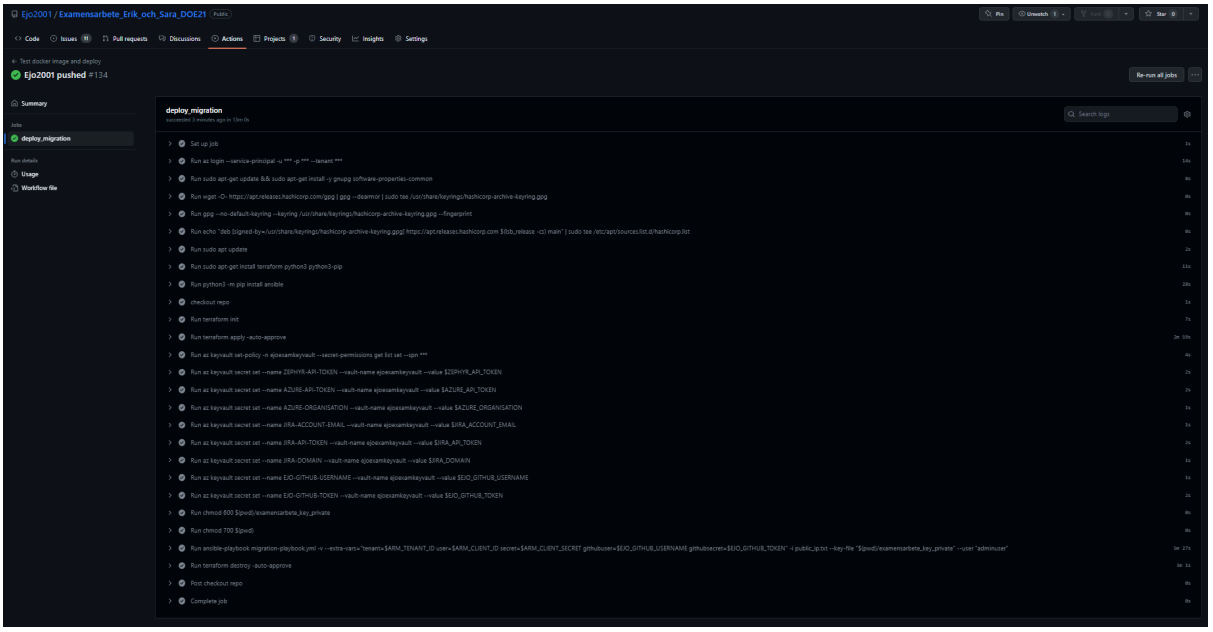

Bild 5. Migrerings pipelinens utförande

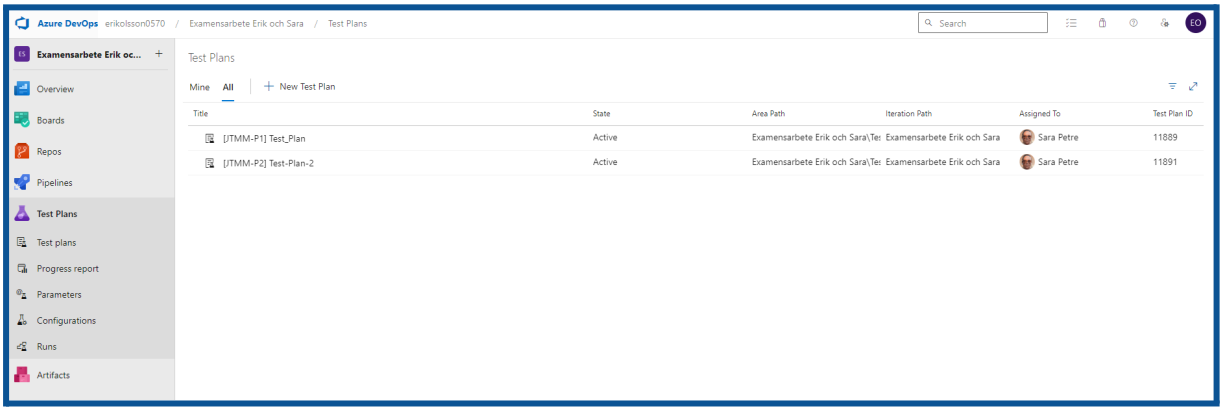

Bild 6. Migreringen har lyckats. Nu återfinns det som fanns i Jira projektet i vårt Azure DevOps -projekt.

6. Diskussion

Arbetet gick som planerat, vi lyckades arbeta agilt på ett bra sätt och nådde fram till slutmålet. Vi kanske inte hade möjlighet att testa Docker Imagen som vi tänkt i vår kravspecifikation, men vi lyckades få till de viktigare delarna i vår kravspecifikation. Något att tänka på till nästa gång kan vara att göra lite mer research innan vi skriver kravspecifikationen för att se att alla delar är möjliga. Projektet hade potentiellt haft möjlighet att täcka alla delar om vi haft mer tid, men med den korta tidsplanen som vi fick så tycker vi att vi har kommit väldigt långt.

7. Källförteckning

None

8. Bilagor

Vårt GitHub Repository: Ejo2001/Examensarbete_Erik_och_Sara_DOE21: Examensarbete av Erik Olsson och Sara Petré. (github.com)
