 Softwareudvikling bachelor rapport   
  
  
Automatiseret End-to-End Test System

Team:  
Anders Stræde Bjerregaard Nielsen asbn26877@edu.ucl.dk

Afleveringsdato:  
3. Juni 2024

Uddannelse:  
PBA i Softwareudvikling

Institution:  
UCL, Erhvervsakademi & Professionshøjskole, Odense Seebladsgade

vejleder:  
Kenneth Jepsen Clausen kjcl@ucl.dk

anslag: tbd

Source Code: <https://github.com/AndersBjerregaard/BetterBoard-BA>

# Indholdsfortegnelse

[Indholdsfortegnelse 2](#_Toc165374248)

[Indledning 4](#_Toc165374249)

[Problemstilling 5](#_Toc165374250)

[Problemformulering 5](#_Toc165374251)

[Afgrænsning 6](#_Toc165374252)

[End-to-End Test 6](#_Toc165374253)

[Cloud Provider 6](#_Toc165374254)

[Tech Stack 6](#_Toc165374255)

[Container Runtime 6](#_Toc165374256)

[Metode 7](#_Toc165374257)

[CI / CD Pipelines 7](#_Toc165374258)

[Secret / Key Management 7](#_Toc165374259)

[Virtualisering / Containerisering 7](#_Toc165374260)

[Systemdokumentation 7](#_Toc165374261)

[System Infrastruktur Diagrammer 7](#_Toc165374262)

[Sekvensdiagrammer 7](#_Toc165374263)

[Azure Cloud & Tooling 7](#_Toc165374264)

[Testniveauer 7](#_Toc165374265)

[End-to-End 7](#_Toc165374266)

[Integration 7](#_Toc165374267)

[Unit 7](#_Toc165374268)

[Kommunikationsprotokoller 7](#_Toc165374269)

[HTTP / S 7](#_Toc165374270)

[AMQP 7](#_Toc165374271)

[Analyse 8](#_Toc165374272)

[Source Control 8](#_Toc165374273)

[GitHub 8](#_Toc165374274)

[Azure DevOps 8](#_Toc165374275)

[Automatiseret Web Interaktion 8](#_Toc165374276)

[Selenium 8](#_Toc165374277)

[Cypress 8](#_Toc165374278)

[Playwright 8](#_Toc165374279)

[Nightwatch 8](#_Toc165374280)

[Virtualisering 8](#_Toc165374281)

[Genskabelige Miljøer 8](#_Toc165374282)

[Lokal & Cloud Miljø 8](#_Toc165374283)

[End-to-End Tests 8](#_Toc165374284)

[Use Cases 8](#_Toc165374285)

[Validering af Webdriver Resultater 8](#_Toc165374286)

[Test Eksekverings Miljø 8](#_Toc165374287)

[Test Miljø 8](#_Toc165374288)

[CI / CD 8](#_Toc165374289)

[GitHub Actions 8](#_Toc165374290)

[Azure Pipelines 9](#_Toc165374291)

[Secret / Key Management 9](#_Toc165374292)

[Self-Hosted Runners / Agents vs. Cloud Provided Runners / Agents 9](#_Toc165374293)

[Udrulning til Azure 9](#_Toc165374294)

[Azure Web GUI 9](#_Toc165374295)

[Infrastructure-as-Code 9](#_Toc165374296)

[.NET Aspire 9](#_Toc165374297)

[Konklusion 10](#_Toc165374298)

[Lokalt System Infrastruktur Diagram 10](#_Toc165374299)

[Distribueret System Infrastruktur Diagram 10](#_Toc165374300)

[Monolit Endpoints Sekvensdiagrammer 10](#_Toc165374301)

[Problemformulering Besvarelse 10](#_Toc165374302)

[Litteraturliste 11](#_Toc165374303)

[Bilag 12](#_Toc165374304)

# Indledning

## TODO: Introducer Anti-Pattern

# Problemstilling

BetterBoard har en ambition om at kunne *deploy* ændringer, lavet i monolittens kodebase, som minimum 8 gange om dagen.

Udviklingsafdelingen får skabt nok ændringer i løbet af en dag, til ovenstående mål. Men den eksisterende CI / CD pipeline udfører kun *continuous delivery*, og ikke *continuous deployment*. Dette er et bevidst valg, da der ikke er høj nok tillid til den eksisterende automatiske kvalitetssikring for softwareproduktet.

Derudover står majoriteten af BB’s distribueret softwareinfrastruktur udokumenteret. Da det primært er blevet kreeret gennem brugergrænsefladen på Azure. Det er der interesse i at afvige fra: Ved at diktere at al fremtidig distribueret software infrastruktur skal dokumenteres, som minimum med et passende *infrastructure-as-code* værktøj.

## Problemformulering

Denne opgave ønsker at opsætte kvantificerbare accept kriterier for en forbedret automatisk software kvalitetssikringsproces. Der kan garantere firmaets tillid til en automatisk *continuous deployment*.

Herunder implementere tilstrækkelig tests til at opfylde nævnte kriterier.

Opgaven ønsker ydermere at ekspandere på eksekveringsplanen for det nuværende CI / CD flow, med den nyligt implementeret kvalitetssikring.

Derudover vil al ny software, der skal udrulles til *cloud provideren* i denne opgave, dokumenteres med passende *IaC*.

Opsummeret set undersøger denne opgave følgende:

Hvordan kan BetterBoard sikre en høj kvalitet i det leverede produkt, når der er en ambition om at opdatere det mindst 8 gange om dagen – og ikke har et team af testere siddende.

Ud fra ovenstående udliciteres følgende underspørgsmål:

Hvordan kan BB kvalitetssikre dets softwareprodukter, løbende og kontinuerligt, når softwaren opdateres eller ændres, før det når ud til slutbrugeren?

Hvad skal der til for at implementere en automatiseret kvalitetssikring, der som minimum dækker de manuelle tests der bliver foretaget før hver udrulning?

Hvad skal der til at dokumentere den automatiseret drift?

# Afgrænsning

Denne opgave går i dybden med en række emner. Nogle af disse emner – som kan ses i underoverskrifterne forneden – vil være afgrænset til en mere snævret specifikation. Til formål for at give læseren indblik i den specifikke kontekst, baseret på terminologien.

Dette er et afkom af at visse ord kan have en bred betydning.

## Automatiseret Acceptance Test

Automatiseret acceptance tests er et emne der bliver berørt ofte i denne opgave. Ordet ’acceptance test’ kan have en bred betydning. Men i denne rapport, går det ud fra den definition at det er forretningsorienterede tests der understøtter udviklingsteamet (Crispin & Gregory, 2009).

Og som problemformuleringen beskriver. Så er det specifikt acceptance tests der menes der kan automatiseres, der vil blive bearbejdet.

## Cloud Provider

På baggrund af at denne opgave er i samarbejde med et firma, som distribuerer sin software gennem Microsoft Azure. Skal det forstås at Azure er cloud provideren, når der tales om udrulning eller distribuering af software.

## Tech Stack

I samme anordning som foroven: Førnævnte firma har en del eksisterende software, skrevet i en bestemt tech stack. Vil denne opgave, tilnærmelsesvis, forsøge at holde sig tæt på C# .NET, i de systemer hvor det er passende. Af den årsag at produkterne beskrevet i denne rapport har til hensigt at kunne driftes af eksisterende udviklere i virksomheden.

## Container Runtime

Docker er den valgte container runtime i produkterne beskrevet i denne opgave. På baggrund af at det efter seneste statistik, er den mest udbredte teknologi inden for dette emne (Statista, 2023). Samt er det – på tidspunktet at denne opgave blev udført – den bedst supporteret container runtime på Microsoft Azure og dets dokumentation.

# Metode

## CI / CD Pipelines

Continuous integration, delivery & deployment eller CI / CD pipelines. Er et værktøj der bruges til at integrere kildekode ændringer, kompilere system artefakter eller udrulle applikationer. I takt med disse primære mål, inkorporeres der ofte automatiseret og kontinuerlig database integration, testing, inspicering, udrulning og feedback i en CI / CD pipeline. Til gavn for at reducere risici og skabe en mere tillidsfuld distribuering (Duvall, Matyas & Glover, 2007).

Pipelines er ofte defineret som en række ’trin’ eller ’jobs’ i en yaml fil. Udtrykt efter en specifik skema reference, som for eksempel Azure Pipelines eller GitHub Action Workflows. Trinnene består typisk af ovennævnte primære mål og inkorporeringer. Derudover har en pipeline typisk en ’trigger’, der beskriver hvornår den skal eksekveres. Eksempelvis ved en pull request mod en branch i source control.

Pipelines bliver eksekveret på en

## Secret / Key Management

## Virtualisering / Containerisering

## Systemdokumentation

### System Infrastruktur Diagrammer

### Sekvensdiagrammer

## Azure Cloud & Tooling

## Testniveauer

### End-to-End

### Integration

### Unit

## Kommunikationsprotokoller

### HTTP / S

### AMQP

# Analyse

## Source Control

### GitHub

### Azure DevOps

## Automatiseret Web Interaktion

### Selenium

### Cypress

### Playwright

### Nightwatch

## Virtualisering

### Genskabelige Miljøer

### Lokal & Cloud Miljø

## End-to-End Tests

### Use Cases

### Validering af Webdriver Resultater

### Test Eksekverings Miljø

### Test Miljø

## CI / CD

### GitHub Actions

### Azure Pipelines

### Secret / Key Management

### Self-Hosted Runners / Agents vs. Cloud Provided Runners / Agents

## Udrulning til Azure

### Azure Web GUI

### Infrastructure-as-Code

### .NET Aspire

# Konklusion

## Lokalt System Infrastruktur Diagram

## Distribueret System Infrastruktur Diagram

## Monolit Endpoints Sekvensdiagrammer

## Problemformulering Besvarelse

# Litteraturliste

Crispin, L. & Gregory, J. (2009). *Agile testing: a practical guide for testers and agile teams*. Upper Saddle River, Nj: Addison-Wesley.

Duvall, P.M., Matyas, S. & Glover, A. (2007). *Continuous Integration*. Pearson Education.

Statista. (2013). *Container platform runtimes share worldwide 2021*. [online] Tilgængelig her: https://www.statista.com/statistics/1224618/container-platforms-deployed-runtime/ [Tilgået 3. Maj 2024].

# Bilag