

Teststrategi - Feature Branching

Syfte: Målet med denna teststrategi är att säkerställa applikationens funktion och kvalitet. Strategin är utformad för att minimera risken för fel och problem inför en lansering i en produktionsmiljö. All utveckling sker i feature branches som måste genom PR merge in i main.

Strategi-översikt:

1. Enhetstester
2. Integrationstester
3. Systemtester

Detaljerad Strategi:

1. Enhetstester

Verktyg: JUnit, JaCoCo, Checkstyle

Frekvens: Vid push, build samt (PR).

Mål: 90% kodtäckning och en kodstruktur med maximalt 150 varningar.

2. Integrationstester

Verktyg: JUnit, JaCoCo, Checkstyle

Frekvens: Vid push, build samt PR.

Mål: 90% kodtäckning och en kodstruktur med maximalt 150 varningar.

3. Systemtester

Verktyg: Healthcheck - Gör healthcheck av container.


Frekvens: Vid push och PR.

Mål: Säkerställa att applikationen kan starta i en Docker-testmiljö.

CI/CD Pipeline

- **Snabb felhantering (Fail Fast):** Om ett test misslyckas, avbryts pipeline omedelbart och teamet notifieras via e-post.
- **Automatiserad distribution:** Efter framgångsrik testning och när en Docker-image har pushats till GitHub, initieras en automatisk distribution.

Hantering av merge-konflikt.



This branch has conflicts that must be resolved

Use the [web editor](#) or the [command line](#) to resolve conflicts.

Resolve conflicts

Conflicting files

README.md

Merge pull request

You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

README.md1 conflict

1# DevOps-ExaminationProject

2<<<<<< feature-fixes

3Instruktionsbok för Utvecklare

4För att underlätta för utvecklare som ska releasa programmet, inkluderas en instruktionsbok med värdefulla kommandon. Exempel på dessa är:

5./gradlew checkstyleMain

6./gradlew checkstyleIntegrationTest

7./gradlew build

8./gradlew test

9./gradlew integrationTest

10git log

11git log -S

12git show <git-sha>

13|=====

14here i will create a conflict

15>>>>>> main

16

Resolved conflict

1 conflicting file

README.md

README.md

1# DevOps-ExaminationProject

2Instruktionsbok för Utvecklare

3För att underlätta för utvecklare som ska releasa programmet, inkluderas en instruktionsbok med värdefulla kommandon. Exempel på dessa är:

4./gradlew checkstyleMain

5./gradlew checkstyleIntegrationTest

6./gradlew build

7./gradlew test

8./gradlew integrationTest

9git log

10git log -S


11git show <git-sha>

12

13

Github package

devops-examinationproject

 Install from the command line

[Learn more about packages](#)

```
$ docker pull ghcr.io/rterborch/devops-examinationproject:development-latest
```

GitHub Actions

Testing JUnit och integrationtester körs i simultanska jobb, ifall någon av dessa misslyckas kommer körningen inte att köra build. Detta för att inte slösa på onödiga resurser och så snabbt som möjligt upptäcka fel.

Triggered via push 3 hours ago



RterBorch pushed

→ 87c2dc9 [feature-fixes](#)

Status

Success

Total duration

4m 57s

Artifacts

1

feature.yml

on: push

✓ testing-JUnit

58s

✓ testing-IntegrationTest

58s

✓ building

1m 16s

✓ publish

24s

✓ system-testing

1m 26s

✓ deploy-test

⌚ deploy-to-production

GitHub Pull-Requests (PR)

main branch är låst och kan inte pushas till. All merge med databasen måste ske från en PR av en branch som benämning som börjar med feature.

- Minst 1 person måste kontrollera och godkänna commiten.
- Kan endast merga genom en pull request från feature branch.
- Måste köra testerna system-testing, testing JUnit, IntegrationTest.
- main är en “locked Branch” och går endast merga med en pull request.

✖

Review required

At least 1 approving review is required by reviewers with write access. [Learn more about pull request reviews.](#)

✔

All checks have passed

12 successful and 2 skipped checks

Hide all checks

| | | | | | |
|---|---|---|-------------------|----------|---------|
| ✔ | 🤖 | Test-run on push / testing-JUnit (pull_request) | Successful in 59s | Required | Details |
| ✔ | 🤖 | Test-run on push / testing-JUnit (push) | Successful in 1m | Required | Details |
| ✔ | 🤖 | Test-run on push / testing-IntegrationTest (pull_request) | Successful in 1m | Required | Details |
| ✔ | 🤖 | Test-run on push / testing-IntegrationTest (push) | Successful in 1m | Required | Details |
| ✔ | 🤖 | Test-run on push / building (pull_request) | Successful in 59s | | Details |
| ✔ | 🤖 | Test-run on push / building (push) | Successful in 1m | | Details |

✖

Merging is blocked

Merging can be performed automatically with 1 approving review.

☐ Merge without waiting for requirements to be met (bypass branch protections)

Merge pull request

▼

You can also [open this in GitHub Desktop](#) or view [command line instructions](#).


Test som misslyckats:

Triggered via push 3 minutes ago

Status

Total duration

Artifacts

 RterBorch pushed • 05bd6d0 [feature-fixes](#)

Failure

1m 18s

–

feature.yml

on: push

✔ testing-JUnit 1m 5s

✖ testing-IntegrationTest 1m 0s

building 0s

publish 0s

system-testing 0s

deploy-test 0s

deploy-to-production 0s

```
PS C:\Users\robin\githubProjects\School\DevOps-ExaminationProject> git show 05bd6d0f1ea53501bdc6d6879d5766e42bc265f16
commit 05bd6d0f1ea53501bdc6d6879d5766e42bc265f16 (HEAD -> feature-fixes, origin/feature-fixes)
Author: RterBorch <robin.ter.borch@gmail.com>
Date: Thu Oct 5 19:15:55 2023 +0200

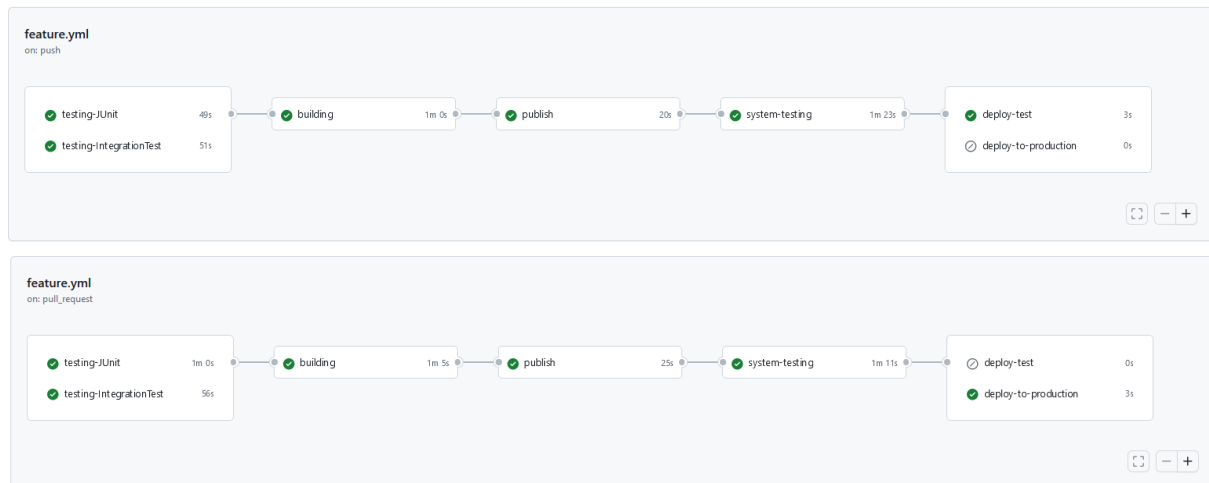
    fail test

diff --git a/src/integrationTest/java/com/example/DevOpsExaminationProject/CarControllerTest.java b/src/integrationTest/java/com/example/DevOpsExaminationProject/CarControllerTest.java
index 721b66e..d55b420 100644
--- a/src/integrationTest/java/com/example/DevOpsExaminationProject/CarControllerTest.java
+++ b/src/integrationTest/java/com/example/DevOpsExaminationProject/CarControllerTest.java
@@ -29,7 +29,7 @@ class CarControllerTest {
     void greet_shouldReturnHelloWorld() throws Exception {
         mockMvc.perform(get("/greet"))
             .andExpect(status().isOk())
             .andExpect(content().string("Hello, World!FAIL"));
     }
 }
```

Vid en PR till main körs både en pull och en push pipeline igång. Vilket ser till att vi först ser till att testerna går gröna innan vi mergar i main, och sedan att de går gröna efter feature-branchen mergad med main.

Detta projekt kör fail-fast. Om inte både testingUnit och integrationTests fortsätter inte projektet. Om båda dessa passerar går pipelinen vidare. Systemtest sker i en container miljö vid systemTesting.

I första varvet deployas projektet till en testmiljö, nästa går den ut i produktion.



Handbok (README) för utvecklare i projektet.

DevOps-ExaminationProject

Instruktionsbok för Utvecklare

För att göra det enklare att releasa programmet, här är en lista på värdefulla kommandon och exempel på kod.

Gradle-kommandon

- `./gradlew checkstyleMain` : Kör checkstyle för huvudkod.
- `./gradlew checkstyleIntegrationTest` : Kör checkstyle för integrationstester.
- `./gradlew build` : Bygger hela projektet.
- `./gradlew test` : Kör enhetstester.
- `./gradlew integrationTest` : Kör integrationstester.

Git-kommandon

- `git log` : Visar en lista över alla tidigare commits.
- `git log -S` : Söker i commit-historiken med en specifik sträng.
- `git show <git-sha>` : Visar detaljer om en specifik commit, ersätt `<git-sha>` med commitens SHA-1 hash.
- `git commit -m "Meddelande"` : Committar ändringar med ett specifikt meddelande.
- `git push` : Skickar dina commits till ett fjärrlager.
- `git pull` : Hämtar den senaste versionen från ett fjärrlager.
- `git merge <branch>` : Sammanfogar en annan gren med den aktuella grenen, ersätt `<branch>` med grenens namn.
- `git rebase` : Används för att flytta eller kombinera en sekvens av commits till en ny baskommit.

Checkstyle:

C:\Users\robin\GithubProjects\School\DevOps-ExaminationProject\build\checkstyle\main.html

Jacoco:

C:\Users\robin\GithubProjects\School\DevOps-ExaminationProject\build\jacocoHtml\index.html