Engineer

Kvalitetsport

en række tjek, kode skal bestå, før den går videre

Hvad det er

En kvalitetsport er en række betingelser, en ændring skal opfylde, før den må gå videre til næste trin, med ét resultat til sidst: bestået eller fejlet. Trinnet kan være et commit, en merge til hovedgrenen, en release eller en udrulning. Betingelserne er alt, hvad en maskine selv kan afgøre: en ren lint-kørsel, tests, der består, en dækningsgrad, et build, der gennemføres.

Det afgørende ord er port. Et tjek rapporterer; en port beslutter, og beslutningen håndhæves af det, der styrer næste trin. En linter, der skriver fyrre advarsler i en log, er et tjek. Den samme linter koblet sådan, at merge-knappen forbliver deaktiveret, indtil outputtet er rent, er en del af en port.

SonarQube Cloud, hvor begrebet er en navngiven funktion, beskriver en kvalitetsport som betingelser, koden måles op imod under analysen. Hver betingelse parrer en metrik med en sammenligningsoperator og en fejlværdi, så »reliability issues greater than 0« er en betingelse, og er blot én af betingelserne opfyldt, fejler porten. En betingelse gælder enten ny kode eller koden som helhed.

Hostingplatformene bygger samme idé ind i deres merge-flow:

- **GitHub** har en indstilling til branch protection, »Require status checks before merging«. Et påkrævet tjek skal have status success, skipped eller neutral, før ændringer når den beskyttede gren.
- **GitLab** har et merge-tjek, der hedder »Pipelines must succeed«. Når det er slået til, kan en merge request ikke merges, mens dens pipeline fejler, og heller ikke, når der slet ingen pipeline er.

- **På udviklerens egen maskine** installerer pre-commit tjek som Git-hooks, der kører ved commit eller push, og en task runner som Task giver hele sættet ét navn.

Brug begrebet i en beslutning

Brug »Hvad er en kvalitetsport?« i en konkret beslutning. Vi kan knytte begrebet til det system, I bygger eller køber. Testet, udgivet og til at rulle tilbage, så ændringer når kunderne en tirsdag eftermiddag i stedet for sent en fredag aften.

Se vores ydelser Kontakt os →

Undersøg antagelserne bag ordet

Brug forklaringen og det tilknyttede arbejde til at formulere spørgsmål til jeres projekt. Vi kan undersøge antagelserne sammen, før et værktøj vælges. Har automatiseret GIS-SaaS-udrulning med CI/CD — udrulning, data og rapportering, alt sammen uden manuelt arbejde

Har automatiseret GIS-SaaS-udrulning med CI/CD Kontakt os →

Hvorfor det betyder noget

En standard, der kun findes i et dokument, bliver forhandlet væk én undtagelse ad gangen. En standard, der får buildet til at fejle, bliver fulgt af alle, også af den, der skrev den, den eftermiddag, hvor vedkommende helst ville springe den over. En port gør aftalte regler eksekverbare, så det at overholde dem holder op med at afhænge af hukommelse og humør.

Den ændrer også, hvad review mellem mennesker er til for. Når en maskine allerede har fanget formatering, ubrugte imports, døde links og lækkede hemmeligheder, kan revieweren bruge sin opmærksomhed på hensigt og design, som en maskine ikke kan vurdere.

Idéen om ny kode gør en port mulig at indføre på en gammel kodebase. Sonar way, standardporten i SonarQube Cloud, fokuserer på ny kode frem for at udbedre gammel kode, med betingelser som mindst 80,0 % testdækning og højst 3,0 % duplikering i ny kode. På en pull request bruges kun betingelserne om ny kode. Et projekt med tusindvis af gamle fund kan derfor slå porten til uden først at afbetale efterslæbet.

Hastighed afgør, om folk lader porten være slået til. Task kører en opgaves afhængigheder samtidigt, så uafhængige tjek kan køre side om side. Det hjælper også at dele tjekkene efter pris: den billige halvdel ved hvert commit, hele sættet, før koden forlader maskinen.

Dette site har sin egen port: `task check`, kørt på den maskine, sitet skrives på, og ikke hos en hostet byggetjeneste. Hver diagnostik tæller som en fejl, sitegeneratoren kører med `--panicOnWarning`, så selv en udfasningsmeddelelse standser buildet, og inline-undertrykkelser som `# noqa` eller `stylelint-disable` er forbudt, så en fejl bliver rettet ved årsagen i stedet for at blive gjort tavs. En pre-commit-hook kører de fil-nære tjek ved hvert commit, og en pre-push-hook kører hele `task check`, som tager minutter og ikke sekunder. En grøn kørsel fra et rent træ stempler træet, og udrulningsscriptet afviser et træ uden stempel. Den eneste tilsidesættelse er en miljøvariabel, der skriver en tydelig advarsel, så den, der springer porten over i en nødsituation, må sige det højt.

Sådan ser det ud i praksis

Sitets kolofon oplister værktøjerne i porten: vnu til HTML, Stylelint, axe-core på én side pr. skabelon, Lighthouse, lychee til links, Prettier, markdownlint, ESLint, Ruff, basedpyright, yamllint, ShellCheck, codespell og gitleaks, og Task kører det hele under ét navn pr. opgave. Siden viser også, hvad der står uden for porten: af browsertjekkene, som Puppeteer styrer, er det overløb og den visuelle regression, og dem kører et menneske efter behov.

Et enmandsprojekt har slet ingen reviewer, og det er den situation, en side i porteføljen tager udgangspunkt i. Standarderne skulle være eksekverbare, så et brud får et commit til at fejle i stedet for at vente på, at nogen opdager det. I den dokumenterede implementeringsfase fra august 2026 blev svaret en gate med 22 lintere uden noget advarselsniveau, plus projektspecifikke tjek, som intet standardværktøj har en holdning til, for eksempel at det udrullede træ ikke indeholder private stier eller private værtsnavne, og at teksten overholdt tonereglerne på alle tre sprog. En fejl, der fandtes i forvejen, rettes i samme omgang. Siden nævner prisen ligeud: det er langsomt at committe, og porten er den eneste reviewer, projektet har.

En port kan også være uenig med sig selv. En anden side beskriver et projekt med en commit-hook og en fuld kvalitetsport, et dokument, der sagde, at hooken kørte porten, og to lister over tjek, vedligeholdt i hånden i to filer. En test, der udleder, hvilke tjek de to lister reelt kalder, viste fem tjek, der kun kørte i porten, og det var netop dem, der læser prosaen. Hvert tjek, der beskytter kode, kørte automatisk; hvert tjek, der beskytter teksten, kørte kun, når nogen huskede det. De fem blev flyttet ind i hooken, to af dem blev gjort hurtige nok til at holde sig inden for dens tidsbudget, og sammenligningstesten blev stående for at fange forskelle mellem listerne.

Hvor det går galt

De fleste kvalitetsporte svigter ved at lukke ting igennem og samtidig vise grønt.

Advarsler hober sig op bag et bestået. En port, der kun fejler på fejl, lader advarsler samle sig, til loggen er for lang til at læse, og så kommer et reelt problem som advarsel nummer to hundrede og en. Fjern advarselsniveauet: enten er en regel værd at fejle på, eller også er den slået fra.

Et oversprunget tjek tæller som bestået. På GitHub opfylder status skipped et påkrævet tjek, og et job, der springes over af sin `if:` -betingelse, melder success. Et job, der afhænger af et fejlet job, springes også over og blokerer måske ikke merge. GitHubs råd er at bruge `always()` sammen med `needs` : kræv ét job, der altid kører og kun består, når alt, hvad det afhænger af, lykkedes.

```
# .github/workflows/check.yml (uddrag)
jobs:
    gate:
        if: always()
        needs: [lint, test, build]
        runs-on: ubuntu-latest
        steps:
            - run: |
                  for result in ${{ join(needs.*.result, ' ') }}; do
                      test "$result" = success || exit 1
                  done
```

Den modsatte fælde findes også: et påkrævet workflow, der springes over af et sti- eller grenfilter, efterlader sine tjek som pending og blokerer hver merge. Undgå at kræve workflows, der kan springes over.

To lister glider fra hinanden. Hooken kører ét sæt, serveren et andet, og et dokument påstår, at de er ens. Giv tjekkene én definition, og kald den fra hvert sted, eller hold en test, der sammenligner listerne og fejler, når de afviger.

Porten bliver langsom, og folk går udenom. Et flag, der springer hooks over, er lige ved hånden, og på GitHub gælder branch protection som standard ikke for administratorer af repositoriet; »Do not allow bypassing the above settings« ændrer det. Mål, hvor tiden går, hold commit-halvdelen lille, og flyt et reelt langsomt tjek til et dokumenteret manuelt trin.

Et tjek består uden at have tjekket noget. En linter rettet mod en glob, der rammer nul filer, eller et ignore-mønster, der sluger en hel mappe, afslutter rent, og en ren afslutning ser ud som ren kode. Test selve porten: giv hvert tjek en fil, der vides at være forkert, og forvent, at tjekket fejler.

Læs mere

- Introduction to quality gates — betingelser, standardporten Sonar way og fokus på ny kode.
- About protected branches — påkrævede statustjek, og hvem der kan omgå dem.
- Troubleshooting required status checks — hvilke statusser der tæller som bestået, og hvad et oversprunget workflow eller job gør.
- Auto-merge — GitLabs tjek »Pipelines must succeed«, og hvordan det behandler oversprungne pipelines.
- pre-commit — tjek erklæret én gang og installeret som Git-hooks til commit og push.
- Task dependencies — hvordan Task kører en opgaves afhængigheder samtidigt og dens kommandoer i rækkefølge.

Ordliste

Vælg ud fra jeres faktiske vilkår

Medbring udrulningsforløbet, driftsrisiciene og dem, der har vagten. Vi kan afgrænse en ændring med kontroller, en vej tilbage og tydeligt ansvar.

Se, hvad det koster Kontakt os →

Gør begreberne praktiske

Spørg, hvordan »Hvad er en kvalitetsport?« gælder i jeres organisation. Et konkret eksempel fra jeres nuværende system er nok til at begynde samtalen.

Se vores ydelser

welcome@engineer.company +45 50 10 61 84 København, Danmark

Tag denne side med dig

Visitkort

Lavet med kærlighed af D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Copyright © 2025 – nu · Engineer ApS · Alle rettigheder forbeholdes.

Udskrevet fra https://engineer.company/da/glossary/quality-gate/