Engineer

Kodebase

all kildekoden til én app, holdt i ett repo

Hva det er

En kodebase er hele mengden kildekode som et team redigerer, gjennomgår og leverer som én ting. Når utviklere sier «kodebasen», mener de stedet der hver endring i en applikasjon havner: selve koden, testene, skriptene som bygger og sjekker den, endringene i databasen og dokumentene som forklarer hvordan bitene henger sammen.

The Twelve-Factor App, en metode for å bygge webtjenester, gir den mest nyttige arbeidsdefinisjonen, fordi den gjør grensen til noe du kan teste. I et sentralisert versjonskontrollsystem som Subversion er en kodebase ett repo. I et distribuert system som Git er den ethvert sett med repoer som deler en rotcommit, den første commiten i historikken, den uten forelder. To repoer som vokste fra den samme første commiten, er én kodebase, uansett hvor langt fra hverandre de har glidd. To som startet hver for seg, er to kodebaser, selv om de inneholder nesten identiske filer.

Det samme dokumentet setter en regel nummer to ved siden av definisjonen: én kodebase, mange utrullinger. En utrulling (*deploy*) er en kjørende instans av applikasjonen, og som regel finnes det flere på en gang. Produksjon er én, testmiljøet (*staging*) en annen, og hver utviklers lokale kopi teller også. Alle kjører den samme kodebasen, ofte ved ulike commits: en utvikler har arbeid testmiljøet ikke har sett, og testmiljøet har arbeid produksjon ikke har sett.

Hva som hører hjemme inni, følger av dette. Kildekoden og alt som trengs for å gjøre den om til et fungerende program, hører hjemme der, også et manifest som navngir hver avhengighet og versjonen av den. Selve avhengighetene ligger som regel utenfor og hentes av en pakkebehandler. Verdier som endres fra én utrulling til den neste, ligger også utenfor, og fremfor alt påloggingsdata: et passord som er committet én gang, reiser med hver kopi av historikken etterpå.

En kodebase kan romme flere programmer. Et monorepo holder en frontend, et API og databasekoden side om side i ett repo, så én commit kan endre alle tre. Etter regelen om rotcommiten er det fortsatt én kodebase. Det motsatte valget sprer en applikasjon over repoer med adskilte historikker, og da er hvert repo sin egen kodebase. Twelve-factor sier det rett ut: finnes det flere kodebaser, er det et distribuert system, og hver komponent er en app i seg selv.

Bruk begrepet i en beslutning

Bruk «Hva er en kodebase?» i en konkret beslutning. Vi kan knytte begrepet til systemet dere bygger eller kjøper. Formen på et system tegnet opp på papiret først, slik at delene passer sammen og kostnaden er kjent før noen skriver kode.

Se tjenestene våre Ta kontakt →

Undersøk antakelsene bak ordet

Bruk forklaringen og det tilknyttede arbeidet til å formulere spørsmål for prosjektet. Vi kan undersøke antakelsene sammen før et verktøy velges. Én infrastrukturplan for 14 avdelinger — 14 forskningsavdelinger på én felles infrastrukturplan

Én infrastrukturplan for 14 avdelinger Ta kontakt →

Hvorfor det er viktig

Grensen avgjør hva som kan endres sammen. Inne i én kodebase kan en endring i et API og skjermen som kaller det, skrives, gjennomgås og slås sammen som én enhet. På tvers av to kodebaser blir den samme endringen to pull requests, to utgivelser og en periode der den ene siden ligger foran den andre. Begge deler fungerer, og prisen bestemmes av hvor streken ble trukket.

Kodebasen er også enheten standarder kjøres over. En testsuite, en linter, en formaterer, en regel for commit-meldinger og et verktøy for oppgaver tar alle «kodebasen» som sitt omfang, og det som ligger utenfor den, blir sjekket av ingen av dem.

Tidlige beslutninger stivner raskere i en kodebase enn noe annet sted. Hver linje som skrives etter et valg, forutsetter det valget, så en regel for lagdeling eller et kvalitetskrav som settes de første ukene, blir formen på alt som følger, og å snu det senere betyr å skrive om koden som vokste rundt det.

For mennesker er en kodebase det en nykommer får utdelt. Rommer den alt som trengs for å bygge, teste og kjøre applikasjonen, kommer nykommeren i gang med én kloning og én kommando. Bor kunnskapen delvis i koden og delvis i noens hukommelse, går den første uka med til å finne ut hva som er hva.

Slik ser det ut i praksis

På én plattform er kodebasen et flerspråklig monorepo i Go, TypeScript, SQL, Python og shell, og hvert språk kommer med sin egen måte å bygge, teste og linte på. Svaret ble én inngangsdør: et lag med go-task som vokste til 578 navngitte mål, 50 i rotfilen og 528 i filer med egne navnerom under den, og som dekker utviklingsmodusene native, Docker og HTTPS, kodekvalitet, databasehåndtering og produksjonssetting. Hvem som helst kan kjøre `task --list` og se hele verktøykjeden. Siden leser også tallet som en advarsel: 578 mål er mer enn noen kan holde i hodet, så det er navngivingen og navnerommene som holder det brukbart.

Et annet prosjekt begynte med et tomt repo 25. november 2025, og tre beslutninger tatt i de første ukene sto seg fortsatt ni måneder og flere tusen commits senere: en lagdelt stack, datatilgang bak lagrede funksjoner og et krav om null advarsler, innført før det fantes mye kode å holde opp mot det. Begrunnelsen på siden er kjernen i denne artikkelen: en standard som innføres ved commit 5 000, er et opprydningsprosjekt, mens den samme standarden ved commit 50 rett og slett er slik repoet fungerer.

Veggene inne i en kodebase kan være selve utformingen. I grunnleggerens arkitekturbidrag hos NextMariner, en maritim plattform bygd i tre lag, tar Next.js-frontenden seg av presentasjon og ingenting annet. Et bevisst tynt Go-API bygd med Huma over Fiber ruter, validerer og orkestrerer, og forretningslogikken ligger i PostgreSQL-funksjoner. Når en regel endres, endres den i ett lag, i SQL. Grensene ble skrevet ned og håndhevet i gjennomganger, fordi en konvensjon slutter å være en konvensjon når ingen passer på at den følges.

Dette nettstedets egen kodebase trekker en annen type strek. Prosjektsidene skrives i et eget repo, eksporteres inn i dette av et skript og committes, så nettstedet bygges på egen hånd. Inne i denne kodebasen er de sidene genererte filer, og instruksene for arbeid her sier at de aldri redigeres for hånd.

Hvor det går galt

Kopiert kode glir fra hverandre

To applikasjoner trenger den samme hjelpefunksjonen, og det raskeste er å kopiere den over. Et år senere finnes det to versjoner med ulike rettelser, og en feil som er rettet i den ene, lever fortsatt i den andre. Twelve-factor kaller flere

apper som deler den samme koden, et brudd og navngir løsningen: flytt den delte delen inn i et bibliotek, hent den inn gjennom avhengighetsbehandleren, og la hver kodebase oppgi versjonen den bruker.

En hemmelighet havner i historikken

En tilkoblingsstreng eller en API-nøkkel blir committet i en konfigurasjonsfil, og fra da av ligger den i historikken hver kloning bærer med seg, også etter at en senere commit sletter linjen. Twelve-factor-testen er verdt å kjøre før hver commit: kunne denne kodebasen blitt publisert som åpen kildekode i dag uten å avsløre en eneste driftshemmelighet? Alt som ville stryke på den testen, hører hjemme i miljøet til hver utrulling.

En generert fil blir redigert for hånd

En fil som er skrevet av et skript, ser ut som enhver annen kildefil, så noen retter en skrivefeil direkte i den. Neste gang skriptet kjører, skriver det den gamle teksten tilbake, og rettelsen forsvinner uten et ord. Merk genererte stier tydelig, la generatoren være den eneste som skriver dem, og send hver rettelse til det generatoren leser.

De indre veggene slites tynne

Under en frist går en spørring rett inn i en handler for forespørsler fordi det er den korteste veien, og lagdelingen prosjektet startet med, slutter å beskrive koden. Hver snarvei er liten. Sammen gjør de en endring på to linjer om til en som rører hvert lag. En skrevet regel hjelper, kodegjennomgang hjelper mer, og en sjekk som stopper byggingen når ett lag griper inn i et annet, hjelper mest.

En standard dekker halve koden

En linter satt opp for frontenden lar shell-skriptene og SQL-en være i fred, og nettopp de filene samler opp problemene linteren skulle stoppe. Et krav for hele kodebasen må navngi hvert språk i den, og det er den skjulte prisen for et flerspråklig repo.

En fork starter et nytt produkt

Et team kopierer et repo for å starte et beslektet produkt. Begge kopiene beholder den samme rotcommiten, så etter twelve-factor-regelen er de fortsatt én kodebase, nå for to apper, og avdriften beskrevet ovenfor begynner i et helt repos målestokk. Å sammenligne rotcommitene viser hvilken situasjon du er i:

```
git -C app-a rev-list --max-parents=0 HEAD
git -C app-b rev-list --max-parents=0 HEAD
```

Lik utskrift betyr en felles rot og en felles kodebase, og ulik utskrift betyr to adskilte historikker.

Les mer

- I. Codebase — regelen om én kodebase og mange utrullinger, og testen med felles rotcommit bak den.
- II. Dependencies — hver avhengighet erklært i et manifest og isolert, så en kodebase sier nøyaktig hva den henter inn.
- III. Config — testen med åpen kildekode for hva som må holdes utenfor en kodebase.
- gitsubmodules — hvordan Git bygger ett repo inn i et annet mens hvert av dem beholder sin egen historikk.
- git-rev-list — referansen for å liste commits, også rotcommitene i en historikk.

Ordliste

Velg ut fra deres faktiske rammer

Beskriv arbeidsflyten, eksisterende grensesnitt og brukerne som trenger endringen. Vi kan definere en testbar leveranse og en overlevering som gjør teamet i stand til å vedlikeholde løsningen.

Se hva det koster Ta kontakt →

Gjør begrepene praktiske

Spør hvordan «Hva er en kodebase?» gjelder i organisasjonen deres. Et konkret eksempel fra dagens system er nok til å starte samtalen.

Se tjenestene våre

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

Ta med deg denne siden

Visittkort

Laget med kjærlighet av D.

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

Opphavsrett © 2025 – nå · Engineer ApS · Alle rettigheter forbeholdt.

Skrevet ut fra https://engineer.company/nb/glossary/codebase/