Engineer

macOS

Apples operativsystem for personlige datamaskiner

Hva det er

macOS er operativsystemet Apple leverer med hver Mac. Det du ser, er skrivebordet, Finder og appene som følger med; under dem ligger et fullt Unix-system med terminal, skall og de kjente kommandolinjeverktøyene. Apple lager systemet for sin egen maskinvare og selger de to samlet, så den som velger macOS, velger i praksis også en Mac.

Grunnmuren heter Darwin, og kildekoden til den publiserer Apple selv. Darwin hviler på en systemkjerne kalt XNU, en hybrid som forener Mach-kjernen fra Carnegie Mellon University med komponenter fra FreeBSD og et driverrammeverk i C++ som heter IOKit. Navnet står for «X is Not Unix». Også mye av userland, programmene som kjører utenfor kjernen, kommer fra FreeBSD: manualsidene for `sed` og `date` på en Mac beskriver FreeBSD-utgavene av de to verktøyene. Siden macOS 10.15 Catalina har zsh vært standardskallet.

Under systemet har maskinvaren nylig byttet prosessorarkitektur. Mot slutten av 2020 begynte Apple å flytte Macen fra Intel-prosessorer til egne brikker, Apple silicon, og i dag finnes hver Mac-serie fra MacBook Air til Mac Pro med Apple silicon. Brikkene er arm64, og det merkes i samme øyeblikk som programvare bygget på en Mac skal kjøre på en x86-64-server.

På bærbare og stasjonære maskiner er det Windows og Linux macOS blir veid opp mot.

Bruk begrepet i en beslutning

Bruk «Hva er macOS?» i en konkret beslutning. Vi kan knytte begrepet til systemet dere bygger eller kjøper. Den bærbare PC-en, skriveren, påloggingen og e-postkassen, ordnet raskt og på klart og enkelt språk, uten at noen føler seg dum.

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. Etablerte en avdeling for teknisk støtte — 10,000 kunder betjent av en supportavdeling bygd opp fra bunnen av

Etablerte en avdeling for teknisk støtte Ta kontakt →

Hvorfor det er viktig

For et selskap som lager programvare, har macOS to roller som lett flyter over i hverandre: det er maskinen mange utviklere jobber på, og det er en plattform kundene kjører programvare på.

Som arbeidsstasjon skiller en Mac seg på tre punkter fra Linux-serverne koden skal kjøre på i produksjon. Kommandolinjeverktøyene kommer fra FreeBSD og ikke fra GNU, så det samme flagget kan bety to ulike ting i et skript. Prosessoren er arm64, mens serveren kan være x86-64. Og en Mac har ingen Linux-kjerne, mens Linux-containere deler kjernen med verten sin. Verktøy som kjører dem på en Mac, starter derfor først en virtuell Linux-maskin. Apples eget verktøy `container`, som krever Apple silicon og macOS 26, gir hver container sin egen lette virtuelle maskin. Ingenting av dette diskvalifiserer en Mac. Det betyr at dere tester på systemet dere produksjonssetter til, i stedet for å stole på en grønn kjøring på en bærbar.

Som plattform reiser macOS spørsmålet om hvordan dere skal bygge for den. Et skall på tvers av plattformer, altså én kodebase med en innebygd nettlesermotor som leveres til Windows, Linux og Mac, holder flere systemer i takt til en lav pris. På Macen drar det likevel med seg en nettleserkjøretid som en native applikasjon ikke har bruk for, og det tegner sine egne kontroller i stedet for systemets. En native applikasjon i Swift og SwiftUI bruker systemets kontroller og materialer, og den følger brukerens innstillinger for utseende, aksentfarge, kontrast, tekststørrelse og redusert bevegelse uten ekstra kode. Prisen er en egen klient og folk som kan plattformen. Native lønner seg når Mac-brukerne er kundene.

Leveranser til Mac må også gjennom et kontrollpunkt. Programvare som distribueres utenfor Mac App Store, signeres med et Developer ID-sertifikat og sendes til Apples notariseringstjeneste, en automatisk skanning etter skadelig innhold og feil i kodesigneringen. Den er noe annet enn App Review. Siden

macOS 10.15 må all programvare som er bygget etter 1. juni 2019 og distribueres med Developer ID, være notarisert. Apple sender tilbake en notariseringsbillett (ticket) som festes til programvaren, og publiserer den samme billetten på nettet, der Gatekeeper slår den opp første gang brukeren åpner appen. Sertifikatet følger med Apple Developer Program, som koster 99 dollar i året. Ta med medlemskapet og signeringen i planen for den første utgivelsen.

Slik ser det ut i praksis

Et meldingsprodukt hadde en skrivebordsklient på macOS som var et skall på tvers av plattformer, og den verken så ut eller oppførte seg som en Mac-app. Produktets godt testede Rust-kjerne har et C-grensesnitt, og Swift kan kalle C, så en native klient i Swift og SwiftUI ble bygget rett på det grensesnittet. Kjernen er lenket inn som et statisk arkiv fra en fastlåst oppstrømsrevisjon, noe som holder byggingen reproduserbar, og en tynn wrapper eier hver peker som krysser mellom språkene. Klienten starter som en Mac-app, bruker plattformens egne materialer og kontroller og har ingen innebygd nettleser. Siden er ærlig om avstanden: det finnes verken notarisert distribusjon eller oppdateringskanal, så dette er en fungerende app og ikke et produkt andre kjører. Notariseringen beskrevet over er nettopp det som skiller de to.

Plattformens utseende kan håndheves i stedet for å etterlignes. Et designsystem med 48 navngitte tokens i seks grupper bygger farge og materiale på plattformens semantiske farger og glassmaterialet, ikke på faste verdier. Dermed følger grensesnittet systemets utseende, aksentfargen og kontrastinnstillingene uten at noen kode passer på dem, og tekststilene skalerer med brukerens foretrukne størrelse. En linter avviser et rått tall der et token hører hjemme, en hardkodet skriftstørrelse og en animasjon som overser innstillingen for redusert bevegelse. Siden sier også hvor grensen går: tokens gjør et brukergrensesnitt konsistent, og det er noe annet enn å gjøre det godt.

Komprimert minne og swap lar en macOS-app vokse langt forbi maskinens fysiske minne. Dokumentasjonen av én hendelse omtaler diagnoserapporter som viste 111 GB komprimerte sider på en maskin med 36 GB den 2026-09-01, og en separat stikkprøvemåling av minnebruken som viste en vekst på omtrent 41 MB per sekund under en løkke med gjentatte tilkoblingsforsøk. Hendelsesbehandlingen beholdt core-logghendelser i en ubegrenset hovedstrøm og kopierte dem til abonnenter som ikke brukte disse hendelsene. Bufferne fikk grenser, abonnementer ble filtrert etter hendelsestype, og loggingen fikk et frekvensbudsjett. En sammenligning av fullstendige

testkjøringer viste at antallet core-logglinjer falt fra 610 996 til 1 411; dette er målinger fra testkjøringer, ikke en målt verdi for minnebruk etter rettingen eller en ytelsesmåling i produksjon. Avveiningen er uttrykkelig beskrevet: gjentatte diagnosehendelser kan forkastes, og berørt tilstand kan oppdateres i stedet for å beholde et ubegrenset antall hendelser. Separate tidsavbrudd ved avslutning krevde en separat retting.

Hvor det går galt

Aktivitetsmonitor viser nesten ikke noe ledig minne på en Mac som går helt fint. macOS bruker ellers ubrukt minne til caching av nylig brukte filer, og etter hvert som minnet fylles, komprimerer systemet inaktive apper for å gi plass til de aktive. Når brukt minne ligger nær det fysiske og tallet for komprimert minne er høyt, er det altså normalt. Apple skriver selv at ledig eller ubrukt minne ikke nødvendigvis gir bedre ytelse. Les heller Minnebelastning-diagrammet i Minne-panelet: grønt betyr at maskinen bruker all RAM effektivt, gult at den etter hvert kan trenge mer, og rødt at den trenger mer. Swap er det tydeligste signalet:

```
sysctl vm.swapusage
vm_stat | grep -E '^Swap(ins|outs)'
```

Tellerne i `vm_stat` løper fra oppstart, så sammenlign to avlesninger med ett minutts mellomrom. Swapouts som fortsetter å stige, betyr at arbeidslasten har vokst ut av RAM-en.

Et skript som virker på Macen, stopper med en merkelig feil på Linux-serveren, eller omvendt. Den FreeBSD-avledede `sed` på en Mac leser ordet etter `-i` som et suffiks for sikkerhetskopien. GNU-vanen `sed -i 's/a/b/' file` får dermed uttrykket tolket som suffiks og filnavnet som skript. Mac-vanen `sed -i '' …` får på sin side GNU `sed` til å lese uttrykket som et filnavn. Et suffiks skrevet tett inntil flagget virker begge steder:

```
sed -i.bak 's/^port=80$/port=8080/' app.conf && rm app.conf.bak
```

Den varige løsningen er å kjøre skriptene i CI på det systemet de møter i produksjon.

Et bygg som deles utenfor teamet, blir stoppet av Gatekeeper når mottakeren åpner det. Appen er signert for utvikling, eller den er signert, men aldri notarisert. Signer den med et Developer ID-sertifikat, med hardened runtime slått på og et sikkert tidsstempel, send den til notariseringstjenesten, og fest billetten til appen, så Gatekeeper finner den også uten nettforbindelse. Et ZIP-arkiv kan notariseres, men har ikke plass til en billett, så fest den til appen inni arkivet:

```
ditto -c -k --keepParent App.app App.zip
xcrun notarytool submit App.zip --keychain-profile notary --wait
xcrun stapler staple App.app
```

Et containerimage kjører på utviklerens Mac og dør på serveren med `exec format error`. Docker bygger som standard for plattformen til BuildKit-daemonen som utfører byggingen, og på en Mac med Apple silicon kjører den daemonen i en virtuell arm64-maskin. En container deler kjernen med verten, så uten emulering kjører en x86-64-vert bare x86-64-kode. Sjekk serveren med `uname -m`, og bygg deretter for den med `docker build --platform linux/amd64`, eller bygg i CI på serverens prosessorarkitektur.

Les mer

- Mac User Guide — Apples egen brukerveiledning for macOS, oppdatert for hver gjeldende versjon.
- Check if your Mac needs more RAM — Minnebelastning-diagrammet og hva de tre fargene betyr.
- Notarizing macOS software before distribution — Developer ID, hardened runtime og hvordan Gatekeeper bruker billetten.
- Multi-platform builds — hvorfor arm64- og amd64-verter trenger hvert sitt image, og hvordan ett bygg lager begge.
- container — Apples verktøy for å kjøre Linux-containere på en Mac, hver i sin egen lette virtuelle maskin.
- XNU — kildekoden til systemkjernen i macOS, med en kort forklaring på hvordan Mach, FreeBSD og IOKit henger sammen.

Ordliste

Velg ut fra deres faktiske rammer

Beskriv enheten eller tjenesten, observerte symptomer og dataene som må bevares. Vi kan fastslå det tryggeste nyttige undersøkelsessteget.

Se hva det koster Ta kontakt →

Gjør begrepene praktiske

Spør hvordan «Hva er macOS?» 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/macos/