Engineer

macOS

Apples Desktopbetriebssystem

Was es ist

macOS ist das Betriebssystem, das Apple auf jedem Mac ausliefert: der Schreibtisch und der Finder, die mitgelieferten Apps und darunter ein Unix-System mit Terminal, Shell und den vertrauten Befehlszeilenwerkzeugen. Apple entwickelt es für die eigene Hardware und verkauft beides zusammen. In der Praxis bedeutet die Wahl von macOS deshalb die Wahl eines Macs.

Der Kern ist Darwin, dessen Quellcode Apple veröffentlicht. Sein Kernel XNU ist ein Hybrid, der den Mach-Kernel der Carnegie Mellon University mit Code aus FreeBSD und mit IOKit, einem C++-Framework für Treiber, verbindet. Der Name steht für „X is Not Unix“, also „X ist kein Unix“. Auch große Teile des Benutzerraums stammen aus FreeBSD. Die Handbuchseiten für `sed` und `date` auf einem Mac beschreiben die FreeBSD-Versionen dieser Werkzeuge. Seit macOS 10.15 Catalina ist zsh die Standardshell.

Die darunterliegende Hardware hat sich kürzlich verändert. Ende 2020 begann Apple, den Mac von Intel-Prozessoren auf eigene Chips, Apple silicon, umzustellen. Jede Mac-Reihe vom MacBook Air bis zum Mac Pro hat inzwischen Modelle mit Apple silicon. Diese Chips verwenden arm64. Das wird wichtig, sobald auf einem Mac gebaute Software auf einem x86-64-Server laufen soll.

Auf Laptops und Desktoprechnern wird macOS mit Windows und Linux verglichen.

Das Konzept für eine Entscheidung nutzen

Nutzen Sie „Was ist macOS?“ für eine konkrete Entscheidung. Wir können helfen, das Konzept mit dem System in Verbindung zu bringen, das Sie entwickeln oder einkaufen. Laptop, Drucker, Anmeldung und Postfach werden schnell und verständlich in Ordnung gebracht, ohne dass sich jemand unkundig fühlen muss.

Unsere Leistungen entdecken Kontakt aufnehmen →

Die Annahmen hinter dem Begriff prüfen

Nutzen Sie die Erklärung und die verlinkten Arbeiten, um Fragen für Ihr Projekt zu formulieren. Wir können die Annahmen gemeinsam mit Ihnen prüfen, bevor ein Werkzeug ausgewählt wird. Eine Abteilung für technischen Support aufgebaut — 10,000 Kunden betreut von einer neu aufgebauten Supportabteilung

Eine Abteilung für technischen Support aufgebaut Kontakt aufnehmen →

Warum es wichtig ist

Für ein Unternehmen, das Software schreibt, spielt macOS zwei leicht zu vermischende Rollen: Es ist der Rechner, an dem viele Entwickler arbeiten, und eine Plattform, auf der Kunden Software ausführen.

Als Entwicklungsrechner unterscheidet sich ein Mac in drei Punkten von den Linux-Servern, auf die von ihm aus Software ausgeliefert wird. Seine Befehlszeilenwerkzeuge stammen aus FreeBSD statt aus GNU, sodass derselbe Schalter in einem Skript etwas anderes bedeuten kann. Sein Prozessor verwendet arm64, während der Server x86-64 verwenden kann. Und er hat keinen Linux-Kernel, während Linux-Container den Kernel ihres Hosts teilen. Werkzeuge, die solche Container auf einem Mac ausführen, starten deshalb zuerst eine virtuelle Linux-Maschine. Apples eigenes Werkzeug `container`, das Apple silicon und macOS 26 benötigt, gibt jedem Container eine eigene leichtgewichtige virtuelle Maschine. Das schließt einen Mac nicht aus. Es bedeutet, auf dem Zielsystem der Auslieferung zu testen, statt einem grünen Lauf auf einem Laptop zu vertrauen.

Als Plattform wirft macOS die Frage auf, wie dafür entwickelt werden soll. Eine plattformübergreifende Hülle mit einer Codebasis und eingebetteter Browserengine, die für Windows, Linux und den Mac ausgeliefert wird, hält mehrere Systeme kostengünstig auf demselben Stand. Auf dem Mac bringt sie aber eine Browserlaufzeit mit, die eine native Anwendung nicht braucht, und zeichnet eigene Bedienelemente statt der des Systems. Eine native Anwendung in Swift und SwiftUI verwendet die Bedienelemente und Materialien des Systems. Sie folgt ohne zusätzlichen Code den Nutzereinstellungen für Erscheinungsbild, Akzentfarbe, Kontrast, Textgröße und reduzierte Bewegung. Der Preis sind ein separater Client und Menschen mit Plattformkenntnissen. Eine native Anwendung lohnt sich, wenn Mac-Nutzer die Kunden sind.

Auch die Auslieferung auf Macs durchläuft eine Prüfstufe. Außerhalb des Mac App Store verteilte Software wird mit einem Developer-ID-Zertifikat signiert und an Apples Notarisierungsdienst geschickt. Dieser automatische Scan nach schädlichen Inhalten und Problemen mit der Codesignatur ist vom App Review getrennt. Seit macOS 10.15 muss jede nach dem 1. Juni 2019 gebaute und mit Developer ID verteilte Software notarisiert werden. Apple liefert ein Ticket zum Anheften an die Software zurück und veröffentlicht dasselbe Ticket online. Dort sucht Gatekeeper danach, wenn der Nutzer die App erstmals öffnet. Das Zertifikat ist im Apple Developer Program für $99 pro Jahr enthalten. Planen Sie Mitgliedschaft und Signierschritt in die erste Veröffentlichung ein.

Wie es in der Praxis aussieht

Die Desktopoberfläche eines Messagingprodukts auf macOS war eine plattformübergreifende Hülle, die weder wie eine Mac-Anwendung aussah noch sich so verhielt. Sein gut getesteter Rust-Kern stellt eine C-Schnittstelle bereit, und Swift kann C aufrufen. Deshalb wurde ein nativer Client in Swift und SwiftUI direkt auf dieser Schnittstelle aufgebaut. Der Kern wird als statisches Archiv aus einer festgelegten Revision des Upstream-Projekts eingebunden, was den Build reproduzierbar hält. Ein dünner Wrapper übernimmt die Verantwortung für jeden Zeiger, der die Sprachgrenze überquert. Der Client startet als Mac-Anwendung, verwendet die eigenen Materialien und Bedienelemente der Plattform und enthält keinen eingebetteten Browser. Die Seite benennt die Lücke offen: Es gibt weder eine notarisierte Distribution noch einen Aktualisierungskanal. Es ist deshalb eine funktionierende Anwendung statt eines Produkts, das andere Menschen nutzen. Zwischen beidem steht der oben beschriebene Notarisierungsschritt.

Das Erscheinungsbild der Plattform lässt sich erzwingen statt nachahmen. Ein Designsystem aus 48 benannten Tokens in sechs Gruppen baut Farben und Materialien auf den semantischen Farben und dem Glasmaterial der Plattform statt auf festen Werten auf. Die Oberfläche folgt dadurch dem Erscheinungsbild des Systems, der Akzentfarbe und den Kontrasteinstellungen ohne beobachtenden Zusatzcode; ihre Textstile skalieren mit der vom Nutzer bevorzugten Größe. Ein Linter weist eine rohe Zahl an einer für ein Token vorgesehenen Stelle, eine fest codierte Schriftgröße und eine Animation zurück, die die Einstellung für reduzierte Bewegung ignoriert. Die Seite nennt auch ihre Grenze: Tokens machen eine Oberfläche konsistent. Das ist etwas anderes, als sie gut zu machen.

Komprimierter Speicher und Swap lassen eine macOS-App weit über den physischen Speicher des Rechners hinauswachsen. Ein Vorfallbericht dokumentiert Diagnoseberichte vom 2026-09-01 über 111 GB komprimierter Speicherseiten auf einem Rechner mit 36 GB und eine getrennte Stichprobe des Speicherbedarfs mit einem Wachstum von etwa 41 MB pro Sekunde während einer Schleife erneuter Verbindungsversuche. Der Ereignispfad hielt Logereignisse des Anwendungskerns in einem unbegrenzten Hauptstrom und kopierte sie zu Abonnenten, die diese Ereignisse nicht nutzten. Puffer wurden begrenzt, Abonnements nach Ereignistyp gefiltert und die Protokollierung mit einem Ratenbudget versehen. Ein Vergleich vollständiger Testläufe dokumentierte einen Rückgang der Logzeilen des Anwendungskerns von 610.996 auf 1.411. Das sind Testlaufmessungen, weder eine gemessene Speichergröße nach der Korrektur noch ein Produktionsbenchmark. Der Preis ist ausdrücklich benannt: Wiederholte Diagnoseereignisse dürfen entfallen, und betroffener Zustand kann neu eingelesen werden, statt unbegrenzt Ereignisse aufzubewahren. Separate Zeitüberschreitungen beim Beenden brauchten eine eigene Korrektur.

Wo es schiefgeht

Activity Monitor zeigt fast keinen übrigen Speicher auf einem einwandfrei laufenden Mac. macOS hält kürzlich verwendete Dateien in sonst ungenutztem Speicher zwischen. Füllt sich der Speicher, komprimiert es inaktive Apps, um Platz für aktive zu schaffen. Ein Wert Memory Used (belegter Speicher) nahe Physical Memory (physischer Speicher) und ein großer Wert Compressed (komprimierter Speicher) sind deshalb normal. Apples eigene Hinweise sagen, dass freier Speicher einen Mac nicht zwangsläufig schneller macht. Lesen Sie stattdessen den Graphen Memory Pressure (Speicherdruck): Grün bedeutet effiziente RAM-Nutzung, Gelb, dass der Mac irgendwann mehr brauchen könnte, Rot, dass er mehr braucht. Swap ist das deutlichste Signal:

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

Die Zähler von `vm_stat` laufen seit dem Systemstart. Vergleichen Sie deshalb zwei Messungen im Abstand einer Minute. Ständig zunehmende Auslagerungen bedeuten, dass der RAM für die Arbeitslast nicht mehr ausreicht.

Ein Skript funktioniert auf dem Mac und stoppt mit einem seltsamen Fehler auf dem Linux-Server oder umgekehrt. Das aus FreeBSD stammende `sed` auf einem Mac behandelt das Wort nach `-i` als Sicherungssuffix. Die GNU-Gewohnheit `sed -i 's/a/b/' file` verwendet dadurch den Ausdruck als Suffix und den Dateinamen als Skript. Die Mac-Gewohnheit `sed -i '' …` lässt GNU-`sed` den Ausdruck als Dateinamen lesen. Ein direkt an den Schalter angehängtes Suffix funktioniert auf beiden:

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

Die dauerhafte Lösung ist, Skripte in CI auf dem System auszuführen, auf das sie in Produktion treffen werden.

Gatekeeper blockiert einen außerhalb des Teams geteilten Build, wenn der Empfänger ihn öffnet. Die App wurde für die Entwicklung signiert oder zwar signiert, aber nie notarisiert. Signieren Sie mit einem Developer-ID-Zertifikat, aktivierter gehärteter Laufzeit und sicherem Zeitstempel, schicken Sie die App an den Notarisierungsdienst und heften Sie das Ticket an. Gatekeeper findet es dann auch offline. Ein ZIP-Archiv kann notarisiert werden, aber an ihm lässt sich kein Ticket anheften. Heften Sie das Ticket deshalb an die App darin:

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

Ein Containerimage läuft auf dem Entwickler-Mac und bricht auf dem Server mit `exec format error` ab. Docker baut standardmäßig für die Plattform des BuildKit-Daemons. Auf einem Mac mit Apple silicon läuft dieser Daemon in einer virtuellen arm64-Maschine. Ein Container teilt den Kernel des Hosts. Ohne Emulation führt ein x86-64-Host deshalb nur x86-64-Code aus. Prüfen Sie den Server mit `uname -m` und bauen Sie dann mit `docker build --platform linux/amd64` dafür oder bauen Sie in CI auf einem Rechner mit dem CPU-Typ des Servers.

Weiterführende Lektüre

- Mac-Benutzerhandbuch — Apples eigener, für jede aktuelle Version gepflegter Leitfaden zu macOS.

- Prüfen, ob Ihr Mac mehr RAM braucht — der Graph Memory Pressure und die Bedeutung seiner drei Farben.
- macOS-Software vor der Verteilung notarisieren — Developer ID, die gehärtete Laufzeit und Gatekeepers Verwendung des Tickets.
- Builds für mehrere Plattformen — warum arm64- und amd64-Hosts verschiedene Images brauchen und wie ein Build beide erzeugt.
- container — Apples Werkzeug, um Linux-Container auf einem Mac auszuführen, jeden in einer leichtgewichtigen virtuellen Maschine.
- XNU — der Quellcode des macOS-Kernels mit einer kurzen Erklärung des Zusammenspiels von Mach, FreeBSD und IOKit.

Glossar

Für Ihre tatsächlichen Rahmenbedingungen auswählen

Beschreiben Sie das Gerät oder den Dienst, die beobachteten Symptome und die Daten, die erhalten bleiben müssen. Wir können den sichersten nützlichen Diagnoseschritt bestimmen.

Preise ansehen Kontakt aufnehmen →

Die Begriffe praktisch nutzbar machen

Fragen Sie, wie „Was ist macOS?“ auf Ihre Organisation zutrifft. Ein konkretes Beispiel aus Ihrem bestehenden System reicht aus, um das Gespräch zu beginnen.

Unsere Leistungen entdecken

welcome@engineer.company +45 50 10 61 84 Kopenhagen, Dänemark

Nehmen Sie diese Seite mit

Kontaktkarte

Mit Sorgfalt und Liebe gestaltet von D.

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

Copyright © 2025 – heute · Engineer ApS · Alle Rechte vorbehalten.

Gedruckt von https://engineer.company/de/glossary/macos/