Engineer

Softwarepakke

kode, samlet med en beskrivelse, klar til at installere

Hvad det er

En softwarepakke er den form, software rejser i fra dem, der skriver den, til de maskiner, der kører den: selve koden som et arkiv eller en mappe sammen med et manifest, der oplyser navn, version og hvad pakken ellers skal bruge for at virke. I daglig tale siger man bare pakke, i bestemt form pakken og i flertal pakker og pakkerne. Hvert økosystem trækker grænsen sit eget sted. I npm er en pakke en fil eller mappe, som en package.json beskriver, og npm's dokumentation skelner den fra et modul, som er alt, hvad Node.js kan indlæse med require() eller import; kun de moduler, der har en package.json, er pakker. Python bruger ordet om to ting, som Python Packaging User Guide holder adskilt: en importpakke er en mappe med moduler, som koden importerer, mens en distributionspakke er det versionerede arkiv, brugeren henter og installerer. I Rust samler en pakke en eller flere crates med en Cargo.toml, der fortæller, hvordan de bygges. På Debian er enheden den binære pakke, .deb-filen, som systemet installerer og holder styr på.

En pakke er en distributionsenhed, og den er lettest at forklare ud fra naboerne. Et bibliotek er kode; en pakke er måden, koden sendes ud på, og det samme bibliotek kan nå sine brugere som npm-pakke, Python-wheel og Debian-pakke på samme tid. En pakkehåndtering er det program, der læser manifesterne, henter det, de nævner, og installerer det hele i en rækkefølge, der går op: npm, pip, Cargo og apt. Et register eller pakkearkiv er stedet, hvor pakkerne ligger klar, som npm-registret, PyPI, crates.io eller et Debian-spejl. Sprogenes egne pakkehåndteringer installerer kode til ét projekt, som regel i en mappe ved siden af det. Systemets pakkehåndtering installerer software til hele maskinen og noterer, hvilken pakke der har lagt hver fil hvor.

Brug begrebet i en beslutning

Brug »Hvad er en softwarepakke?« i en konkret beslutning. Vi kan knytte begrebet til det system, I bygger eller køber. Systemets form lagt fast på papir først, så delene passer sammen, og prisen er kendt, før nogen skriver kode.

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. Én infrastrukturplan til 14 afdelinger — 14 forskningsafdelinger på én infrastrukturplan

Én infrastrukturplan til 14 afdelinger Kontakt os →

Hvorfor det betyder noget

Kode fra andre når et projekt ad en af tre veje: gennem en pakkehåndtering, kopieret ind i repositoriet med sin licens og en kontrolsum, hvilket kaldes vendoring, eller bygget fra kildekoden. Pakkehåndteringen er det sædvanlige valg, og med god grund. Den løser hele træet af afhængigheder, opdaterer det med én kommando og noterer, hvad den installerede. Prisen er tillid: projektet stoler nu på hvert manifest i træet og på hvert register, træet kom fra. Vendoring bytter den tillid for arbejde, for den kopierede kode ændrer sig kun, når nogen bevidst kopierer en ny version ind. At bygge fra kildekoden giver mest kontrol og kræver mest vedligehold.

Det er i manifestet, de fleste afvejninger viser sig. Det angiver som regel et interval af versioner frem for én bestemt; i npm godtager et caret-interval som ^10.0.1 enhver senere udgivelse med samme hovedversion. Intervaller lader rettelser komme ind, uden at nogen redigerer filen, men de betyder også, at to installationer med en uges mellemrum kan give to forskellige træer. En lockfil lukker det hul. npm skriver package-lock.json med det præcise træ, den byggede, så senere installationer gentager det, Cargo skriver Cargo.lock med den præcise version af hver afhængighed, og begge regner med, at filen bliver committet. Manifestet siger, hvad projektet vil acceptere; lockfilen siger, hvad det faktisk fik.

Manifester siger også, hvor meget én pakke har brug for en anden. Debian Policy Manual opstiller en rangstige. Depends er absolut: pakken bliver først konfigureret, når alt, hvad den afhænger af, er konfigureret. Recommends er stærk, men ikke absolut, og gælder pakker, der hører med i alle andre end usædvanlige installationer. Suggests nævner pakker, der gør den mere nyttig, mens en installation uden dem er helt rimelig. En installation uden en anbefalet pakke er gyldig, og softwaren i den kan stadig have brug for pakken til en af sine opgaver.

Alle værktøjer, der ræsonnerer om pakker, stoler på deres metadata. En licenskontrol læser den erklærede licens, ofte et SPDX-udtryk som MIT OR Apache-2.0, hvor OR giver et valg, og AND kræver alle licenserne på én gang. En sårbarhedsscanner sammenholder erklærede navne og versioner med offentliggjorte sikkerhedsadvarsler. Er en erklæring forkert, tager hvert værktøj, der læser den, fejl på samme måde, og derfor fortjener licensfilerne i en pakke, der sendes ud med et produkt, at blive læst for sig.

Sådan ser det ud i praksis

En hærdningsgennemgang af en produktionsserver viste, at dens jail, der skulle udelukke angribere, aldrig havde udelukket nogen, og en af tre separate årsager var en relation mellem pakker. Til selve udelukkelsen brugte jailen et eksternt firewallværktøj, som pakkernes erklæringer kun regner for anbefalet og ikke for påkrævet, og serveren var bygget uden det. Jailen startede, talte mislykkede logins og nåede frem til sin beslutning hver gang, og først ved det sidste skridt, selve blokeringen, gik den i stå. Gennemgangen af tre SSH-beskyttelser opdagede det ved at se på det kørende system i stedet for konfigurationsfilerne og efterlod en regel om, at enhver sikkerhedskontrol bakkes op af en automatisk assertion og ikke af en kommentar.

En Rust-kerne, der linkes ind i en udgivet app, tager hele sin afhængighedsgraf med ind i produktet, her 644 crates, hver med sin egen licenserklæring. En afhængighed flere lag nede kan skifte til en copyleft-licens fra én version til den næste, og det eneste spor ville være en ændret linje i en lockfil, som få læser. Licensauditten af 644 crates kører ved hvert build: den holder hver crates licensudtryk op mod en godkendelsesliste, lader et valg mellem licenser passere, når én af dem er godkendt, og en kombination kun, når de alle er, og stopper buildet ved alt andet. For at vise, at tjekket kan slå til, blev det med vilje brudt to gange, én gang ved at omskrive en crates licensudtryk og én gang ved at flytte den fastlåste kernerevision uden at generere auditten igen. Siden er klar om grænsen: auditten læser erklærede metadata, som selv kan være forkerte, og den er ikke juridisk rådgivning.

Virksomhedens egen git-forge blev hentet som pakke fra leverandørens eget repository og kører som ét program, styresystemet holder i gang. At føje en leverandørs repository til en server er en beslutning for sig, for fra da af installerer de almindelige opdateringer alt, hvad den leverandør udgiver og signerer.

Selve sitet behøver ikke Node for at køre. Dets package.json er markeret som privat, hvilket får npm til at nægte at udgive den, rummer kun udviklingsværktøjer med caret-intervaller og ligger ved siden af en committet package-lock.json. Dets pyproject.toml indstiller en linter, en stavekontrol og en typekontrol og har hverken en [project]- eller en [build-system]-tabel, fordi filen ikke beskriver en pakke, der skal bygges. Ikonerne er vendoret: kopieret ind sammen med den oprindelige licensfil og en SHA-256-kontrolsum for hver fil.

Hvor det går galt

En funktion melder alt vel og udfører intet

En tjeneste starter, logger, at den kører, og springer stille en af sine opgaver over hver gang. Et program, den kalder, hører til en anbefalet eller foreslået pakke, som installationen havde lov til at udelade. Når en funktion afhænger af et eksternt program, så tjek ved opstart, at programmet findes, og stop med en tydelig fejl, hvis det mangler, erklær det som et hårdt krav i jeres egen udrulning, og test funktionen hele vejen igennem på en server, der er bygget som produktionen.

To maskiner installerer forskellig kode fra samme manifest

Buildet går igennem på en bærbar og fejler i den kontinuerlige integration, eller omvendt, uden at projektets egen kode er ændret. Manifestet angiver versionsintervaller, lockfilen blev aldrig committet, og en afhængighed udgav en ny version mellem de to installationer. Commit lockfilen, installér fra den i automatiseringen med kommandoer, der nægter at ændre den, som npm ci eller cargo build –locked, og opdatér den som en selvstændig ændring, som nogen gennemgår.

En licens skifter inde i en velkendt afhængighed

Et produkt, der bestod en licensgennemgang sidste år, sender nu kode ud under vilkår, ingen har sagt ja til, mens listen over afhængigheder ser ud som før. En eksisterende afhængighed skiftede licens mellem to versioner, og en rutineopdatering trak den nye version ind via lockfilen. Tjek de erklærede licenser mod en eksplicit godkendelsesliste ved hvert build, så skiftet stopper buildet samme dag, det dukker op.

Læs mere

- npm Docs: About packages and modules — npm's egen definition af en pakke, og hvorfor et modul ikke altid er en.

- Debian Policy Manual: Declaring relationships between packages — Depends, Recommends, Suggests og de øvrige felter, og hvad en pakkehåndtering gør med hvert af dem.
- Python Packaging User Guide: Writing your pyproject.toml — de tre tabeller i pyproject.toml og de metadata, en Python-distributionspakke erklærer.
- The Cargo Book: Cargo.toml vs Cargo.lock — hvorfor manifestet beskriver, hvad et projekt accepterer, mens lockfilen noterer præcise versioner, og hvornår den skal committes.
- SPDX Specification 2.3.1: License expressions — syntaksen, pakker bruger til at erklære én licens, et valg mellem flere eller et sæt, der gælder samtidig.

Ordliste

Vælg ud fra jeres faktiske vilkår

Beskriv arbejdsgangen, de eksisterende grænseflader og de brugere, der har brug for ændringen. Vi kan definere en testbar leverance og en overdragelse, der gør teamet i stand til at vedligeholde løsningen.

Se, hvad det koster Kontakt os →

Gør begreberne praktiske

Spørg, hvordan »Hvad er en softwarepakke?« 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/package/