**Engineer**

# Програмний пакет

код, зібраний з описом і готовий до встановлення

## Що це таке

Програмний пакет — це форма, у якій програмне забезпечення потрапляє від тих, хто його пише, на машини, де воно працює: сам код у вигляді архіву чи каталогу разом із маніфестом, що вказує назву, версію і все, без чого цей код не запрацює. Коротко кажуть просто пакет, у родовому відмінку — пакета. Кожна екосистема проводить межі по-своєму. Для npm пакет — це файл або каталог, який описує файл package.json, і документація npm відрізняє його від модуля, тобто будь-чого, що Node.js може завантажити через require() чи import; пакетами є лише ті модулі, які мають package.json. Python уживає це слово у двох значеннях, і Python Packaging User Guide їх розводить: пакет імпорту (import package) — це каталог модулів, які імпортує код, а пакет розповсюдження (distribution package) — версіонований архів, який користувач завантажує й установлює. У Rust пакет об’єднує один або кілька крейтів (crate) і файл Cargo.toml, що описує, як їх зібрати. У Debian одиницею є бінарний пакет — файл .deb, який система встановлює й відстежує.

Пакет — це одиниця розповсюдження, і найпростіше пояснити його через сусідні поняття. Бібліотека — це код; пакет — це спосіб доставити цей код, і одна бібліотека може водночас доходити до користувачів як пакет npm, як wheel для Python і як пакет Debian. Менеджер пакетів — програма, яка читає маніфести, завантажує все, що в них перелічено, і встановлює це в порядку, який справді працює: npm, pip, Cargo, apt. Реєстр або сховище — місце, де пакети чекають, поки їх заберуть: реєстр npm, PyPI, crates.io чи дзеркало Debian. Мовні менеджери пакетів установлюють код для одного проєкту, зазвичай у каталог поруч із ним. Системні менеджери пакетів установлюють програми для всієї машини й записують, який пакет поклав кожен файл і куди.

Застосуйте поняття до рішення

Застосуйте «Що таке програмний пакет?» до конкретного рішення. Ми можемо пов'язати поняття із системою, яку ви створюєте або купуєте. Структура системи, спершу продумана на папері, щоб частини пасували одна до одної, а вартість була відома до першого рядка коду.

**Переглянути наші послуги** Зв'язатися з нами →

Перевірте припущення за терміном

Використайте пояснення та пов'язані роботи, щоб сформулювати запитання для проєкту. Ми можемо перевірити припущення разом до вибору інструмента. Один інфраструктурний план для 14 підрозділів — 14 дослідницьких підрозділів, охоплених спільним інфраструктурним планом

**Один інфраструктурний план для 14 підрозділів** Зв'язатися з нами →

# Чому це важливо

Чужий код потрапляє в проєкт одним із трьох шляхів: через менеджер пакетів, копією в репозиторії разом із ліцензією та контрольною сумою, що називають вендорингом (vendoring), або збиранням із вихідного коду. Менеджер пакетів — звичний вибір, і недарма. Він розв’язує все дерево залежностей, оновлює його однією командою і записує, що встановив. Ціна — довіра: відтепер проєкт покладається на кожен маніфест у цьому дереві й на кожен реєстр, з якого воно прийшло. Вендоринг міняє цю довіру на роботу, бо скопійований код змінюється лише тоді, коли хтось свідомо копіює нову версію. Збирання з вихідного коду дає найбільше контролю й вимагає найбільше догляду.

Найбільше компромісів видно в маніфесті. Зазвичай він називає не одну версію, а діапазон; у npm діапазон із кареткою, як-от ^10.0.1, приймає будь-який пізніший випуск із тією самою мажорною версією. Діапазони пропускають виправлення без жодних правок у файлі, але через них дві інсталяції з різницею в тиждень можуть дати два різні дерева. Цю щілину закриває lock-файл. npm записує в package-lock.json точне дерево, яке побудував, щоб наступні інсталяції його повторили, Cargo записує в Cargo.lock точну версію кожної залежності, і обидва розраховують, що файл зберігатимуть у репозиторії. Маніфест каже, що проєкт готовий прийняти; lock-файл — що він отримав насправді.

Маніфести також кажуть, наскільки сильно одному пакету потрібен інший. Debian Policy Manual описує цілу драбину. Depends — абсолютна вимога: пакет налаштовують лише після того, як налаштовано все, від чого він залежить. Recommends — сильна, але не абсолютна залежність, для пакетів, які стоять поруч із цим у всіх інсталяціях, крім незвичних. Suggests називає

пакети, з якими цей стає кориснішим, але й без них інсталяція цілком розумна. Інсталяція без рекомендованого пакета коректна, а програмі всередині він усе одно може знадобитися для якоїсь із її задач.

Кожен інструмент, що міркує про пакети, довіряє їхнім метаданим. Перевірка ліцензій читає задекларовану ліцензію, часто у вигляді виразу SPDX, як-от MIT OR Apache-2.0, де OR дає вибір, а AND вимагає всіх ліцензій одночасно. Сканер вразливостей зіставляє задекларовані назви й версії з опублікованими бюлетенями безпеки. Якщо декларація хибна, кожен інструмент, що її читає, помиляється однаково, тому файли ліцензій усередині пакета, який іде до користувачів разом із продуктом, варто прочитати окремо.

## Як це виглядає на практиці

Під час перевірки захисту робочого сервера з'ясувалося, що jail, який мав блокувати зловмисників, за весь час не заблокував жодного, і одна з трьох окремих причин крилася у зв'язку між пакетами. Для самого блокування jail звертався до зовнішньої програми брандмауера, яку пакетні декларації вважають рекомендованою, а не обов'язковою, і сервер зібрали без неї. Jail запускався, рахував невдалі входи й щоразу доходив до рішення, а ламався лише на останньому кроці, коли треба було власне заблокувати. Перевірка трьох засобів захисту SSH виявила це, розглядаючи систему під час роботи, а не файли налаштувань, і залишила по собі правило: кожен засіб безпеки підкріплює автоматична перевірка, а не пояснення в коментарі.

Ядро на Rust, скомпоноване із застосунком, який отримують користувачі, приносить у продукт увесь свій граф залежностей — тут 644 крейти, кожен із власною декларацією ліцензії. Залежність кількома рівнями нижче може між двома версіями перейти на копілефтну ліцензію, і єдиним слідом стане змінений рядок у lock-файлі, який мало хто читає. Аудит ліцензій 644 крейтів запускається під час кожного збирання: він звіряє ліцензійний вираз кожного крейта зі списком дозволених, пропускає вибір між ліцензіями, якщо дозволена хоч одна, а поєднання — лише якщо дозволені всі, і зупиняє збирання на всьому іншому. Щоб показати, що перевірка здатна спрацювати, її двічі зламали навмисно: раз — замінивши в одному крейті вираз ліцензії на неприйнятний, раз — перемістивши закріплену ревізію ядра й не згенерувавши аудит наново. Межу сторінка називає прямо: аудит читає задекларовані метадані, які самі можуть бути хибними, і юридичною порадою він не є.

Власний git-сервер компанії ставили пакетом зі сховища пакунків, яке веде сам постачальник, і працює він як одна програма, яку підтримує в робочому стані операційна система. Додати репозиторій постачальника на сервер — окреме рішення, бо відтоді звичайні оновлення встановлюють усе, що цей постачальник публікує й підписує.

Сам сайт не потребує Node для роботи. Його package.json позначено як приватний, тож npm відмовиться його публікувати; у ньому лише інструменти розробки з діапазонами з кареткою, а поруч лежить package-lock.json, внесений у репозиторій. Його pyrgoject.toml налаштовує лінтер, перевірку правопису й перевірку типів і не має таблиць [project] чи [build-system], бо пакета для збирання цей файл не описує. Іконки вендорені: їх скопійовано разом з оригінальною ліцензією й контрольною сумою SHA-256 для кожного файлу.

# Де виникають проблеми

## Функція звітує, що все гаразд, і нічого не робить

Сервіс запускається, пише в журнал, що працює, і щоразу мовчки пропускає одну зі своїх задач. Програма, яку він викликає, належить до рекомендованого чи запропонованого пакета, і інсталяція мала повне право його пропустити. Коли функція залежить від зовнішньої програми, перевіряйте її наявність під час запуску й зупиняйтеся з чіткою помилкою, якщо її бракує, оголосіть її жорсткою вимогою у власному розгортанні й тестуйте функцію від початку до кінця на сервері, зібраному так само, як робочий.

## Дві машини встановлюють різний код з одного маніфесту

Збирання проходить на ноутбуці й падає в безперервній інтеграції або навпаки, хоча власний код проєкту той самий. Маніфест задає діапазони версій, lock-файл так і не внесли в репозиторій, а між двома інсталяціями одна із залежностей випустила нову версію. Внесіть lock-файл у репозиторій, в автоматизації встановлюйте з нього командами, які відмовляються його змінювати, як-от npm ci чи cargo build –locked, і оновлюйте його окремою зміною, яку хтось переглядає.

### Ліцензія змінюється всередині знайомої залежності

Продукт, що торік пройшов перевірку ліцензій, тепер постачає код на умовах, на які ніхто не погоджувався, хоча перелік залежностей виглядає так само. Одна з наявних залежностей змінила ліцензію між версіями, а звичайне оновлення підтягнуло нову версію через lock-файл. Перевіряйте задекларовані ліцензії за явним списком дозволених під час кожного збирання, щоб така зміна зупиняла збирання того ж дня, коли з'являється.

## Що почитати далі

- npm Docs: About packages and modules — власне визначення пакета від npm і пояснення, чому модуль не завжди є пакетом.
- Debian Policy Manual: Declaring relationships between packages — Depends, Recommends, Suggests та інші поля і що з кожним із них робить менеджер пакетів.
- Python Packaging User Guide: Writing your pyproject.toml — три таблиці pyrgoject.toml і метадані, які декларує пакет розповсюдження Python.
- The Cargo Book: Cargo.toml vs Cargo.lock — чому маніфест описує, що проєкт приймає, lock-файл фіксує точні версії, і коли його вносити в репозиторій.
- SPDX Specification 2.3.1: License expressions — синтаксис, яким пакети декларують одну ліцензію, вибір із кількох або набір, що діє разом.

Словник

Обирайте за фактичними умовами

Опишіть робочий процес, наявні інтерфейси та користувачів, яким потрібна зміна. Ми можемо визначити перевірюваний результат і спосіб передачі, який дасть вашій команді змогу підтримувати його.

**Дивитися, скільки це коштує** Зв'язатися з нами →

# Зробіть термінологію практичною

Запитайте, як «Що таке програмний пакет?» стосується вашої організації. Конкретного прикладу з поточної системи достатньо для початку розмови.

**Переглянути наші послуги**

welcome@engineer.company +45 50 10 61 84 Копенгаген, Данія

Візьміть цю сторінку із собою

Візитівка

**Дбайливо зроблено з любов'ю — D.**

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

**Авторське право © 2025 – дотепер · Engineer ApS · Усі права захищено.**

**Надруковано з https://engineer.company/uk/glossary/package/**