당근이 파이썬 공급망 공격에 대응하는 방법

김수빈, 나동희 / 당근

발표자 소개

김수빈

당근 Software Engineer
공통 서비스 개발팀

나동희

당근 Software Engineer
ML인프라팀

CPython Core Dev

Python Steering Council 25/26

목차

1. 파이썬 공급망 공격 현황

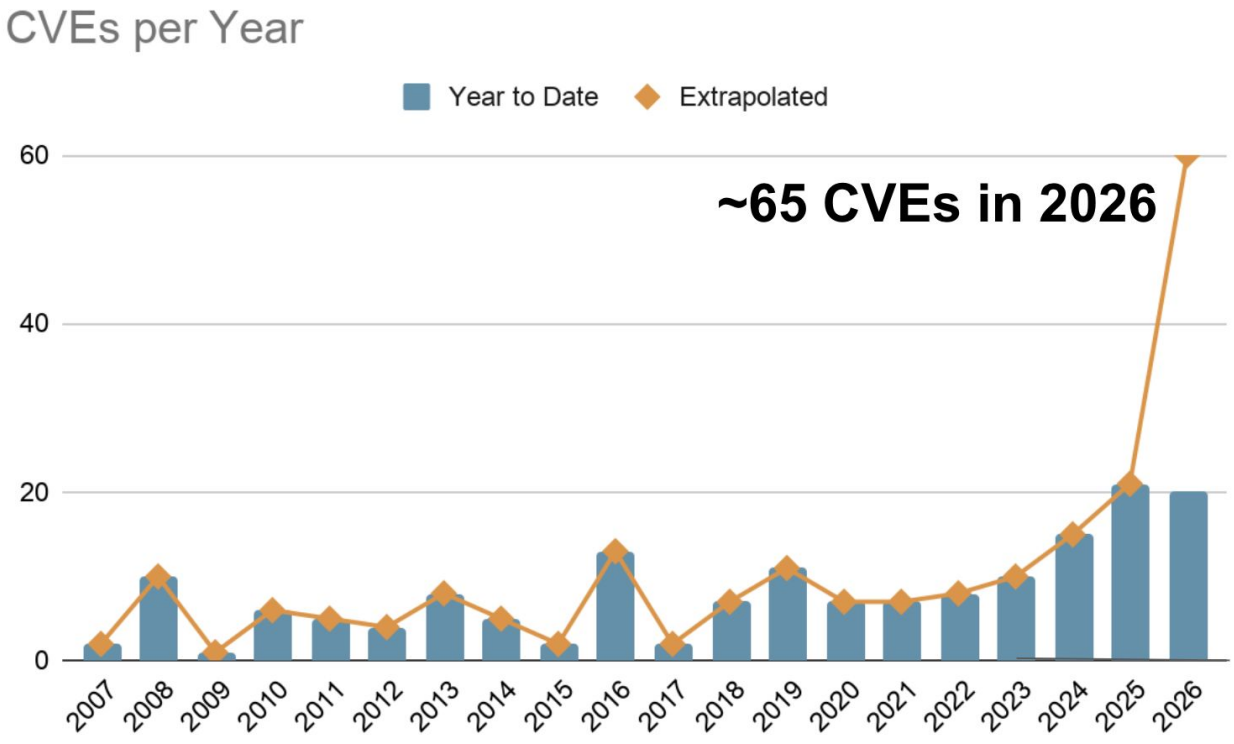
파이썬 공급망 공격 현황

1. 하루에 900개 넘는 새 프로젝트가 계속 등록되고 있음
2. 작년 5월부터 올해 5월까지 2800+ 멀웨어가 리포트되고 있음
3. PyPI -> PSRT에서 보안을 강화하고 있음
    a. Seth Larson
    b. Mike Fiedler

파이썬 공급망 공격 현황


400
300
200
100
0
80
212
298
364
331
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Jan
Feb
Mar
Apr
2025
2026

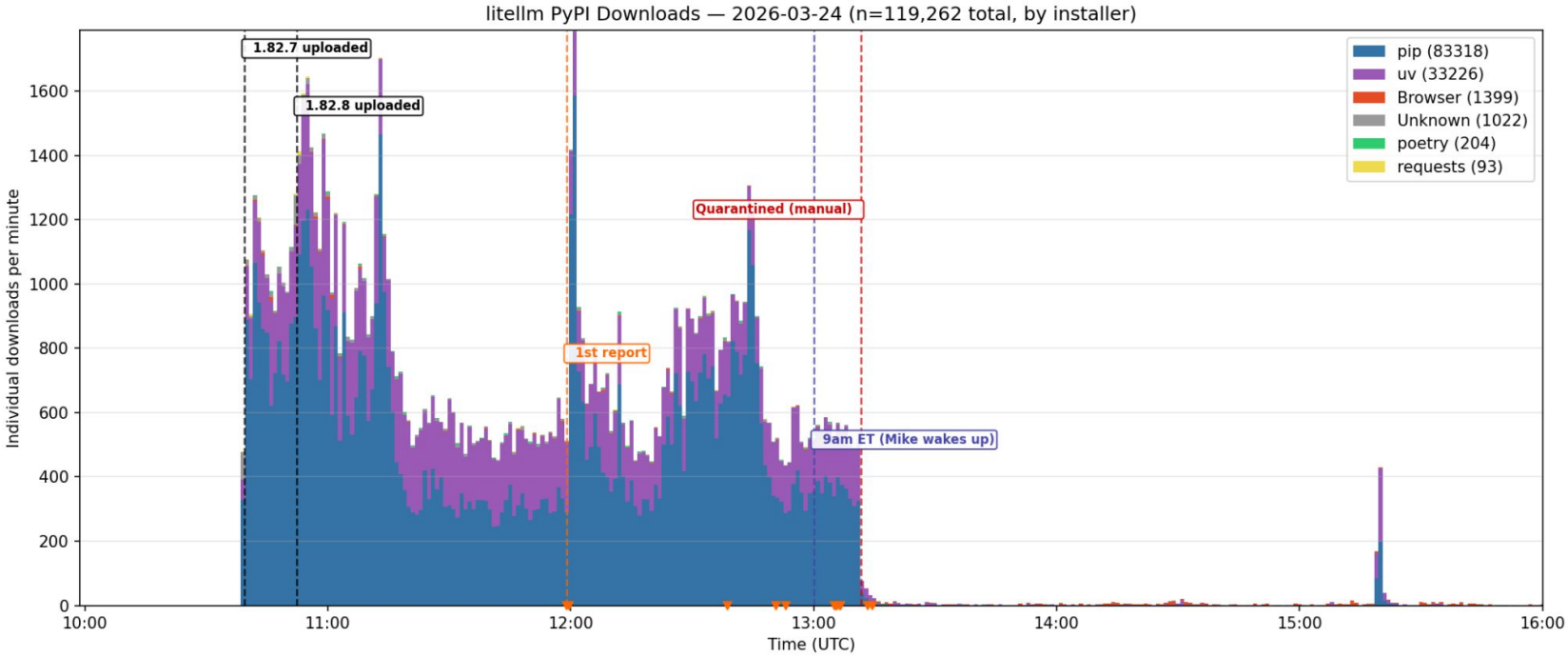
파이썬 공급망 공격 현황


CVEs per Year
Year to Date
Extrapolated
~65 CVEs in 2026
60
40
20
0
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026

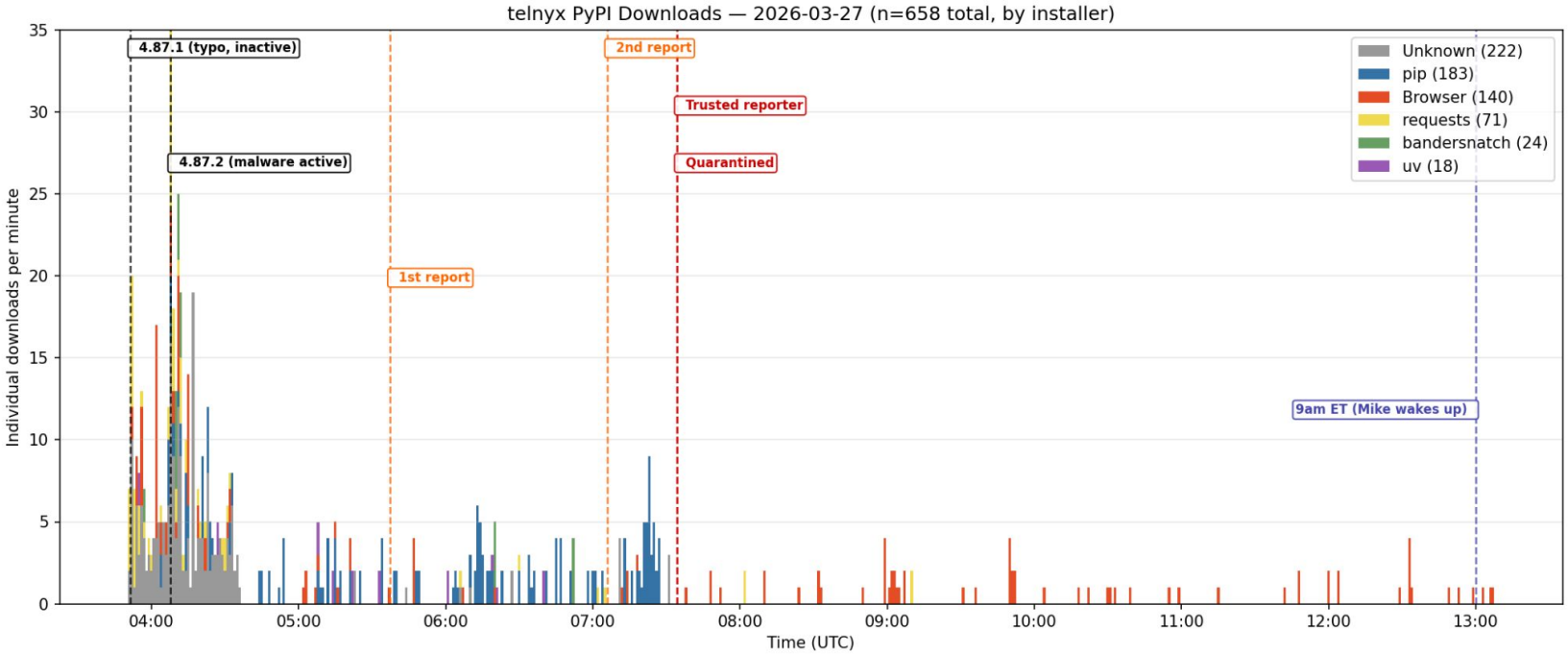
파이썬 공급망 공격 현황

1. Malware
2. Watering Hole Attacks
    a. Shai-Hulud
    b. LiteLLM
    c. Trivy
    d. Phising

litellm 대응 타임라인



telnyx 대응 타임라인



Alpha Omega Project

“Protect society by catalyzing sustainable security improvements to the most critical open source software projects and ecosystems”

PyPI팀 대응 체계

1. Trusted Reporters
2. Trusted Publishing providers
3. Staged Releases
4. 보안감사 강화 w Soverign Tech Funds and Alpha Omega Project
   https://blog.pypi.org/posts/2026-04-16-pypi-completes-second-audit/

대응방안 (라이브러리 개발자 사이드)

1. Zizmor, CodeQL, Semgrep, Fuzzer 혹은 LLM을 통한 라이브러리 검증
2. 보안 취약점 발생시 아래의 핫라인으로 빠르게 전달
    a. security@pypi.org
    b. security@python.org

대응방안 (유저 사이드)

1. 패키지 버전 locking (40-50% 공격 성공의 원인)
2. Cooldown
    a. pip (--uploaded-prior-to) from v26.1
    b. uv (--exclude-newer)
3. 빠른 보안 패치 대응

2. 당근 Python 챕터

당근의 Python 챕터

- 특정 프로그래밍 언어에 관심 있는 구성원들이 자발적으로 모여 논의하는 챕터 그룹
- 사내 Python 사용 사례를 공유하고 생산성과 보안을 함께 논의하고 있어요

당근의 Python 챕터

- 매월 모여서 회사 내에서 발생한 파이썬 관련된 이슈들을 주로 다룸
- 회사 인프라와 관련된 트러블 슈팅
- 회사 인프라와 관련된 노하우 공유
- 기타 파이썬 관련 정보 공유

3. 당근에서의 대응

당근의 사내 pypi 프록시 사례

Release로부터 n일이 되지
않은 패키지는 제외

Python의 수많은 패키지 매니저

pip

conda

poetry

hatch

pipenv

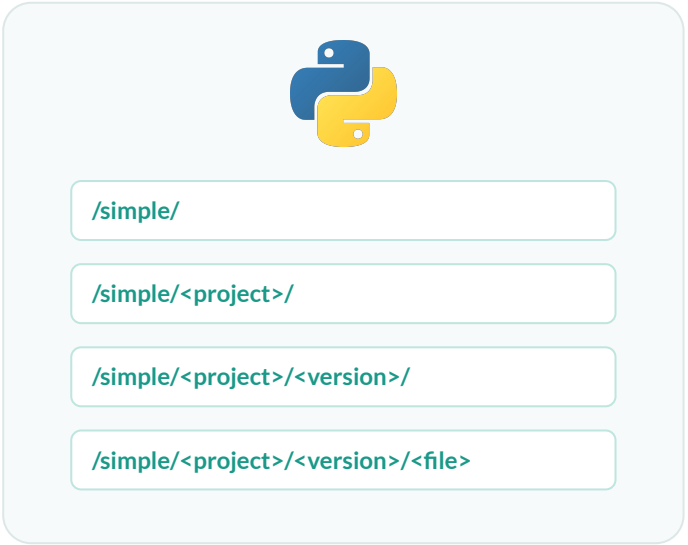
uv

pdm

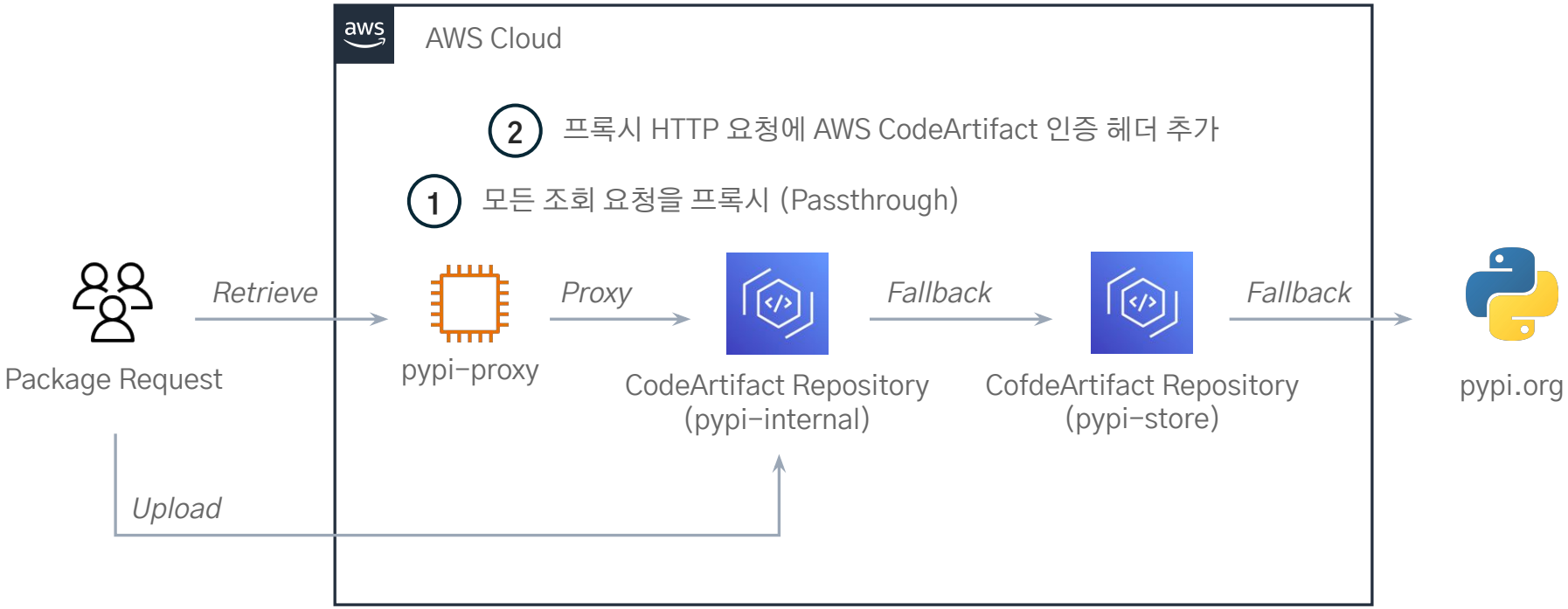
pixi

Python Package Registry

- PyPI Index의 구조
  - /simple
  - /simple/<package>
  - /simple/<package>/<version>
  - /simple/<package>/<version>/<file>
- PEP 503 기반의 Simple Repository API 표준
- 표준만 지키면 모든 환경, 패키지 매니저에서 사용 가능



아키텍처


AWS Cloud
2
프록시 HTTP 요청에 AWS CodeArtifact 인증 헤더 추가
1
모든 조회 요청을 프록시 (Passthrough)
Retrieve
Proxy
Fallback
Fallback
Package Request
pypi-proxy
CodeArtifact Repository
(pypi-internal)
CofdeArtifact Repository
(pypi-store)
pypi.org
Upload

AWS CodeArtifact를 선택한 이유

1. 사내 라이브러리 개발 및 배포 가능
2. 문제가 있는 오픈소스 라이브러리는 패치하여 배포 가능
3. 특정 Python 버전이나 운영체제/CPU에 존재하지 않는 wheel을 채워서 배포 가능
4. 모든 패키지 다운로드가 AWS CodeArtifact를 거쳐가므로 가시성

Region Unsupported

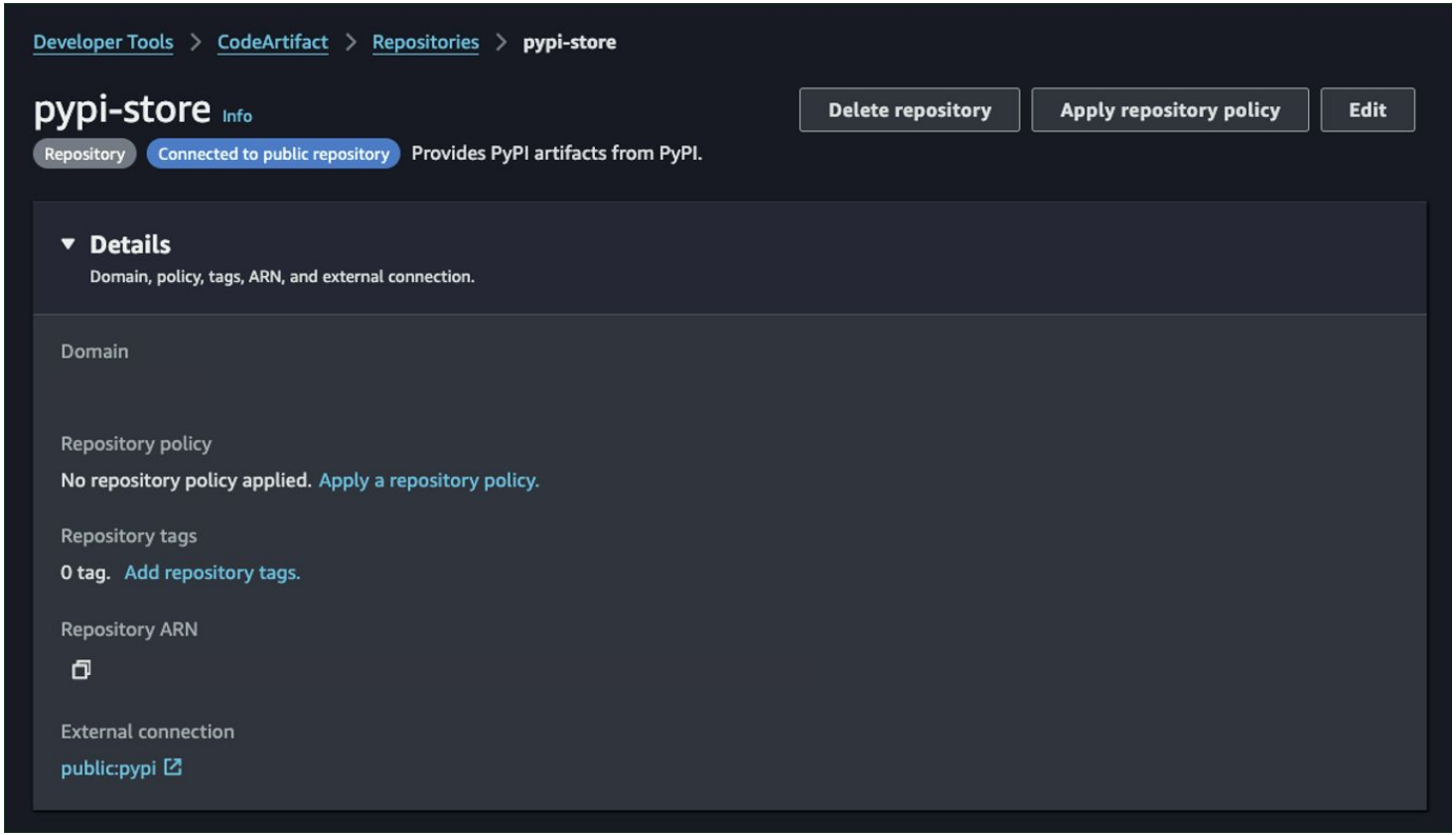
CodeArtifact is not available in **Asia Pacific (Seoul)**. Please select another region.

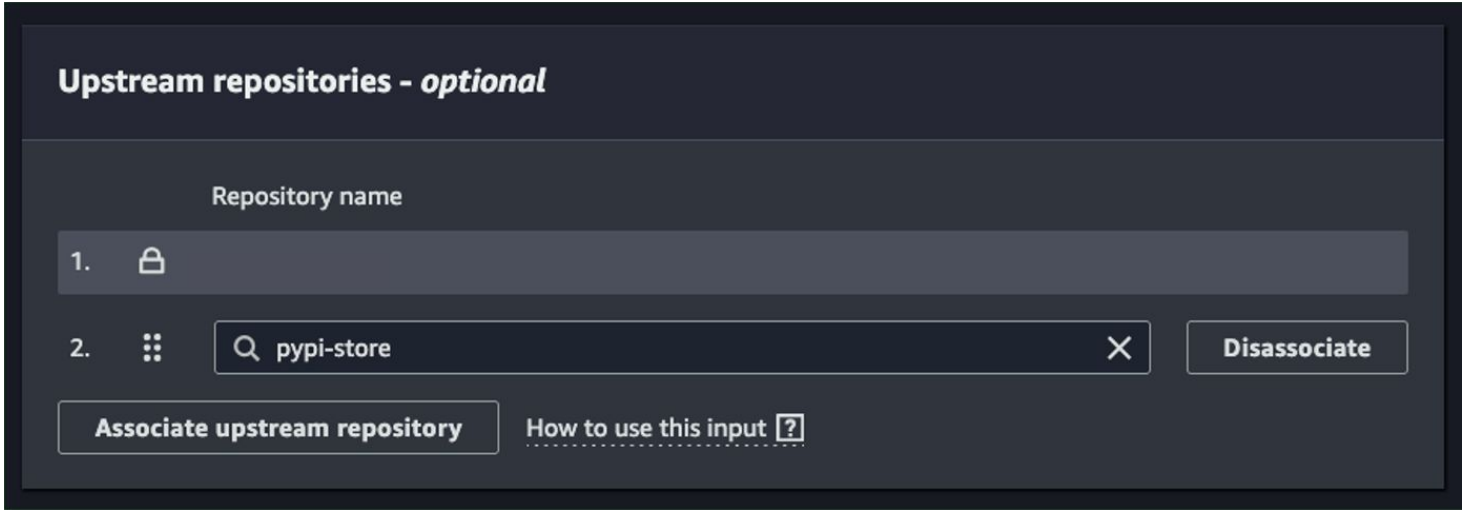
Supported Regions

United States (N. Virginia)
United States (Ohio)
United States (Oregon)

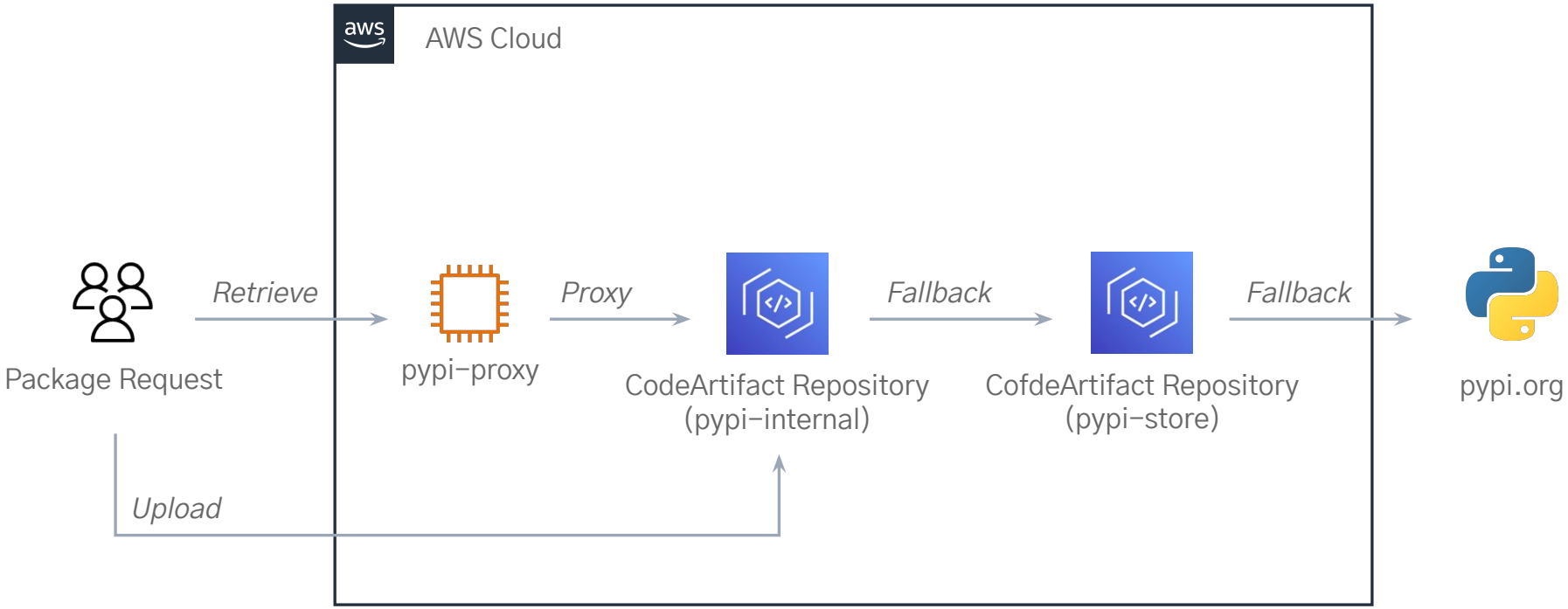
Asia Pacific (Mumbai)
Asia Pacific (Singapore)
Asia Pacific (Sydney)
Asia Pacific (Tokyo)

Europe (Frankfurt)
Europe (Ireland)
Europe (London)
Europe (Paris)
Europe (Stockholm)

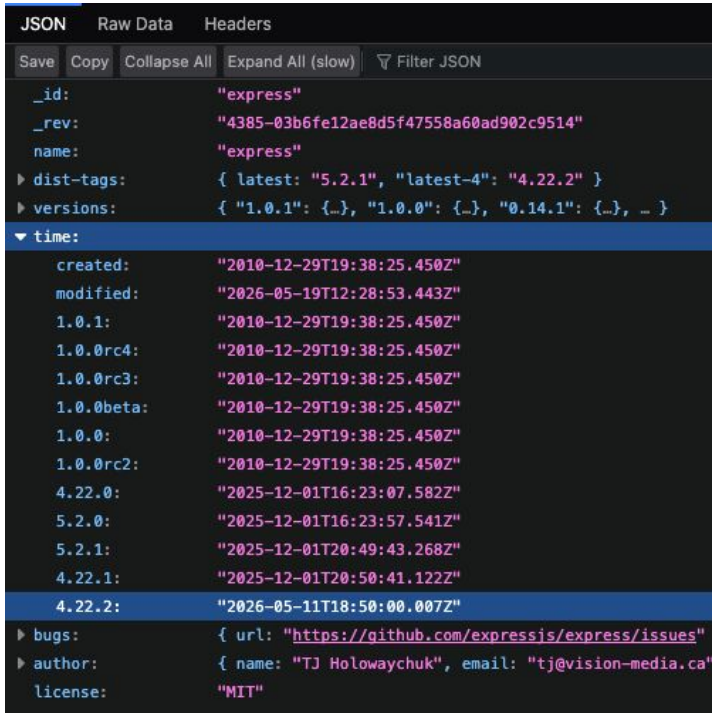

Developer Tools > CodeArtifact > Repositories > pypi-store
pypi-store Info
Delete repository
Apply repository policy
Edit
Repository
Connected to public repository
Provides PyPI artifacts from PyPI.
Details
Domain, policy, tags, ARN, and external connection.
Domain
Repository policy
No repository policy applied. Apply a repository policy.
Repository tags
0 tag. Add repository tags.
Repository ARN
External connection
public:pypi


Upstream repositories - optional
Repository name
1.
2.
pypi-store
Disassociate
Associate upstream repository
How to use this input

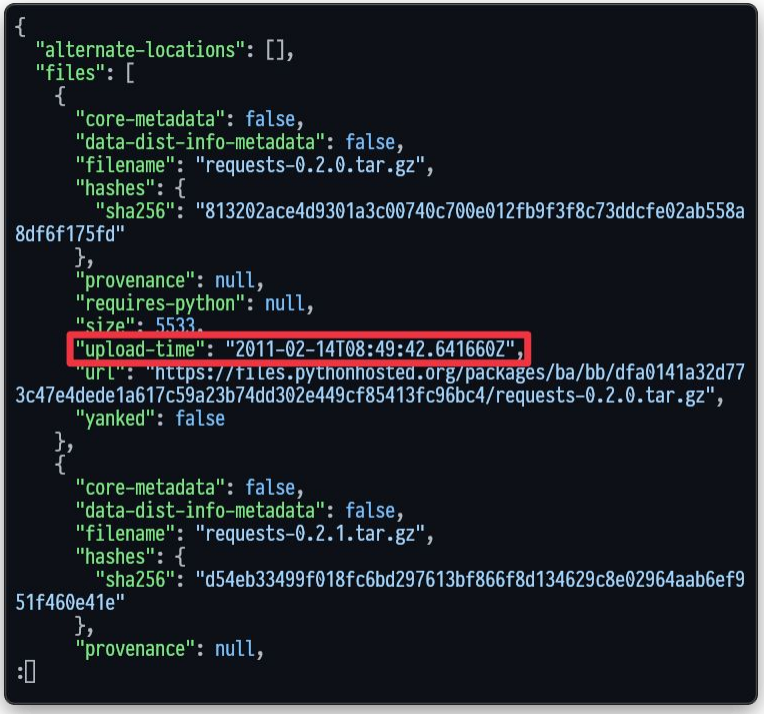
아키텍처 (한 번 더)


aws
AWS Cloud
Package Request
Retrieve
pypi-proxy
Proxy
CodeArtifact Repository
(pypi-internal)
Fallback
CofdeArtifact Repository
(pypi-store)
Fallback
pypi.org
Upload

npm과 PyPI



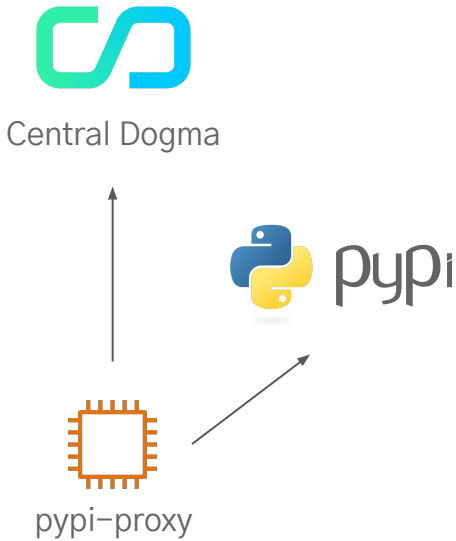
https://registry.npmjs.org/express



https://pypi.org/simple/requests/

Cooldown 정책 적용

- Central Dogma에 저장된 Cooldown 기간 (현재 n일)
  - 필요 시 동적으로 늘리고 줄일 수 있음
- Cooldown의 예외 패키지 목록 관리
  - ex. 사내 패키지
- 패키지의 배포일은 PyPI API를 호출하여 확인하고 필터링

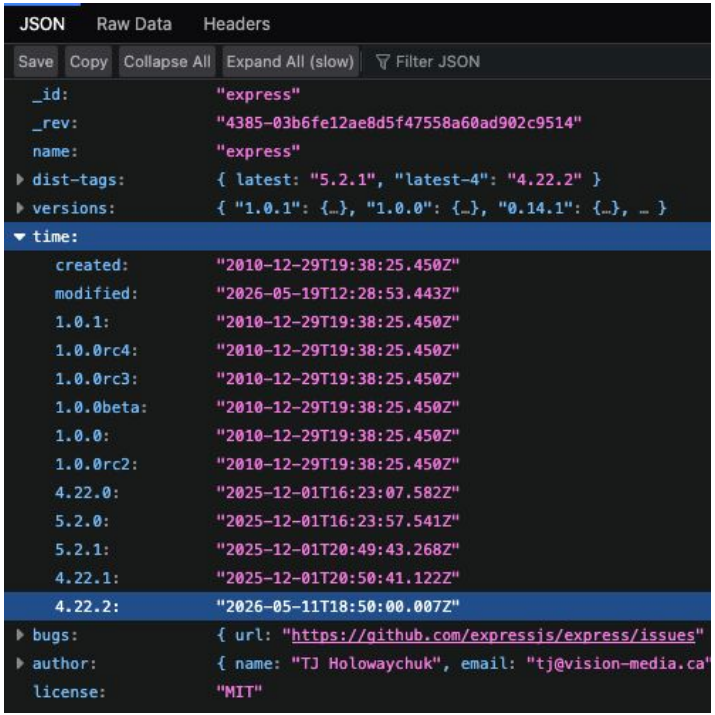


결론

- 매우 간편하게 사용할 수 있게 된 Package Registry
- 높은 개발자 만족도와 구성 편의성
- 최근 다양한 라이브러리들의 보안 문제도 잘 회피함

당근의 npm registry 사례

- Front-end 챕터
- NPM의 경우, 자체 패키지 레지스트리를 운영 중
- 퍼블릭 패키지를 **n일 Cooldown** 해서 제공 중



https://registry.npmjs.org/express

당근의 npm registry 사례

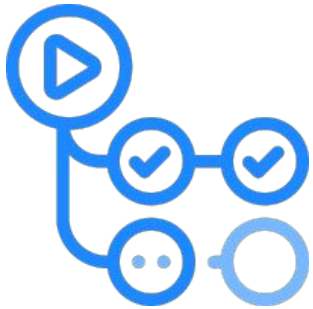
Release로부터 n일이 되지
않은 패키지는 제외

4. 향후 계획

전사 기본 적용 계획

- 현재는 각 프로젝트에서 Registry Proxy를 설정하여 적용
- 이를 전사에 공통으로 일괄 적용할 계획 중

전사 기본 적용 계획



GitHub Actions
Runner Image

Windows 11
macOS

MDM
Device Management

전사 기본 적용 계획

- pip
  - Linux: `/etc/pip.conf`
  - macOS: `/Library/Application Support/pip/pip.conf`
- uv
  - `/etc/uv/uv.toml`

pip.conf — Plaintext

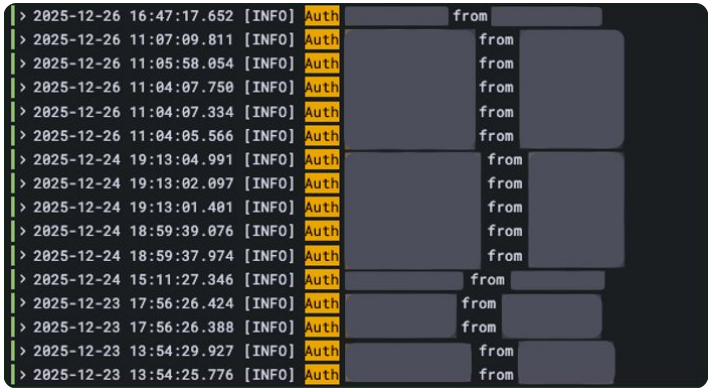
```
[global]
index-url = https://package-registry-proxy.com/simple
trusted-host = package-registry-proxy.com
```

uv.toml — TOML

```
[[index]]
url = "https://package-registry-proxy.com/simple"
default = true
```

Audit / 이상 탐지 등

- 패키지 요청에 대한 가시성
- 패키지 Vulnerability 관측 레이어



Audit / 이상 탐지 등

- 패키지 요청을 한 곳에서 로깅하고 사용 현황 파악
  - 패키지 종류, 버전 파악
  - 이상 탐지
- Cooldown으로 놓치는 악성 패키지의 경우
  Audit, Allow/Deny 정책으로 보완 가능

향후 계획

- AI 시대, Python과 uvx로 실행하는
  임의의 MCP, 스크립트 늘어나는 추세
- 라이브러리 사용 현황을 상시 파악하고,
  악성 패키지가 발견되면 플랫폼에서 즉시 조치
- 개발자들이 주의를 기울이지 않더라도 더 안전한 라이브러리 사용

마치며

- 당근 SRE팀
- 당근 보안팀
- □ 아이디어부터 첫 사용까지 1주일
- □ free-threaded Python 실사용 테스트도 함께 진행

마치며

4개의 공고가 열려있어요

Lead Security Engineer - 인프라 (보안, Offensive Security)

Security Engineer - 당근페이

Security Engineer - 인프라 (보안, AI Security)

Security Engineer - 인프라 (보안, Detection & Response)

https://careers.daangn.com/jobs/?division=tech:security

감사합니다