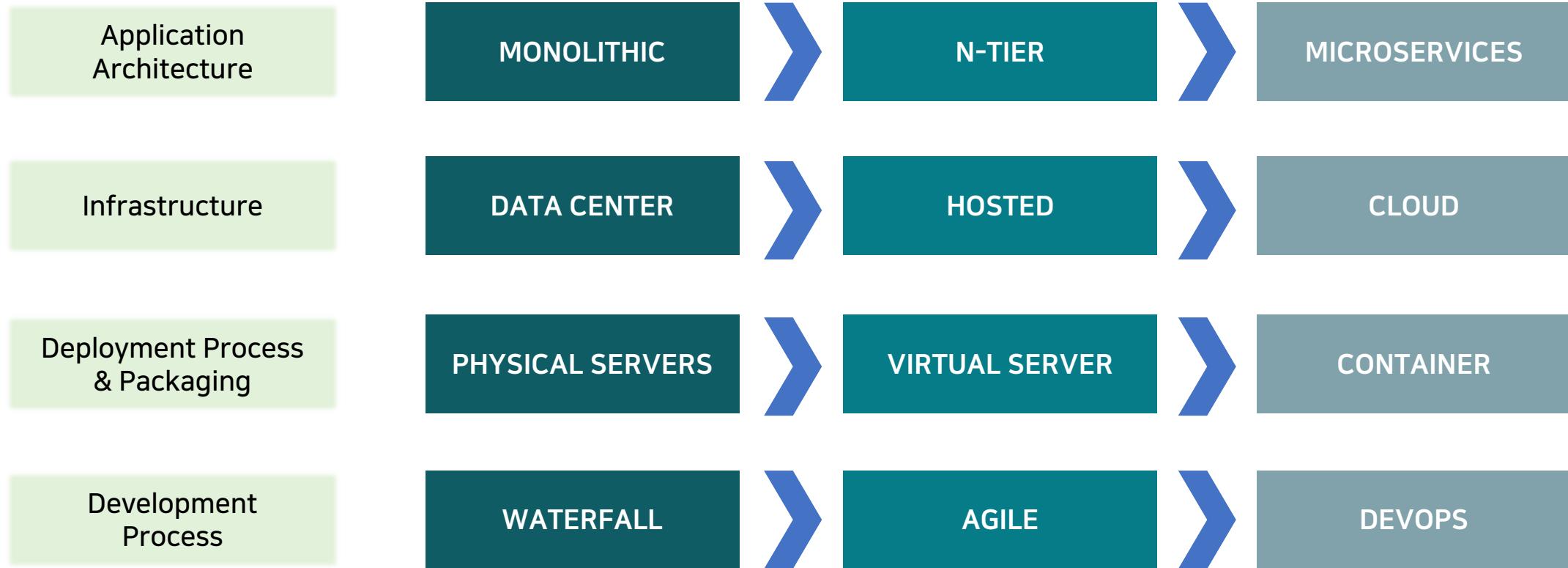


**NAVER Cloud**

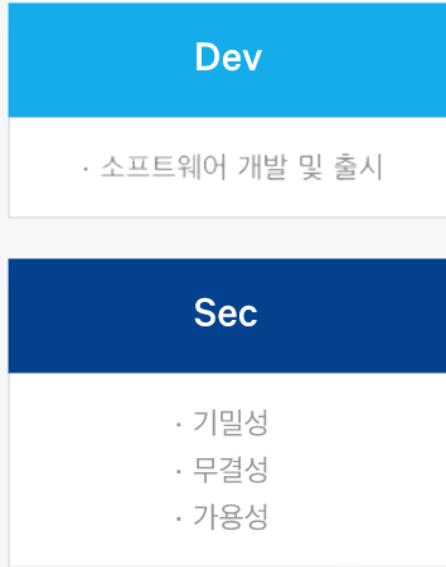
하루안에 클라우드 인프라 구축과  
DevOps 운영 정복하기

# DevOps on NAVER Cloud Platform

# 개발 환경의 변화



# CI/CD를 위한 DevTools

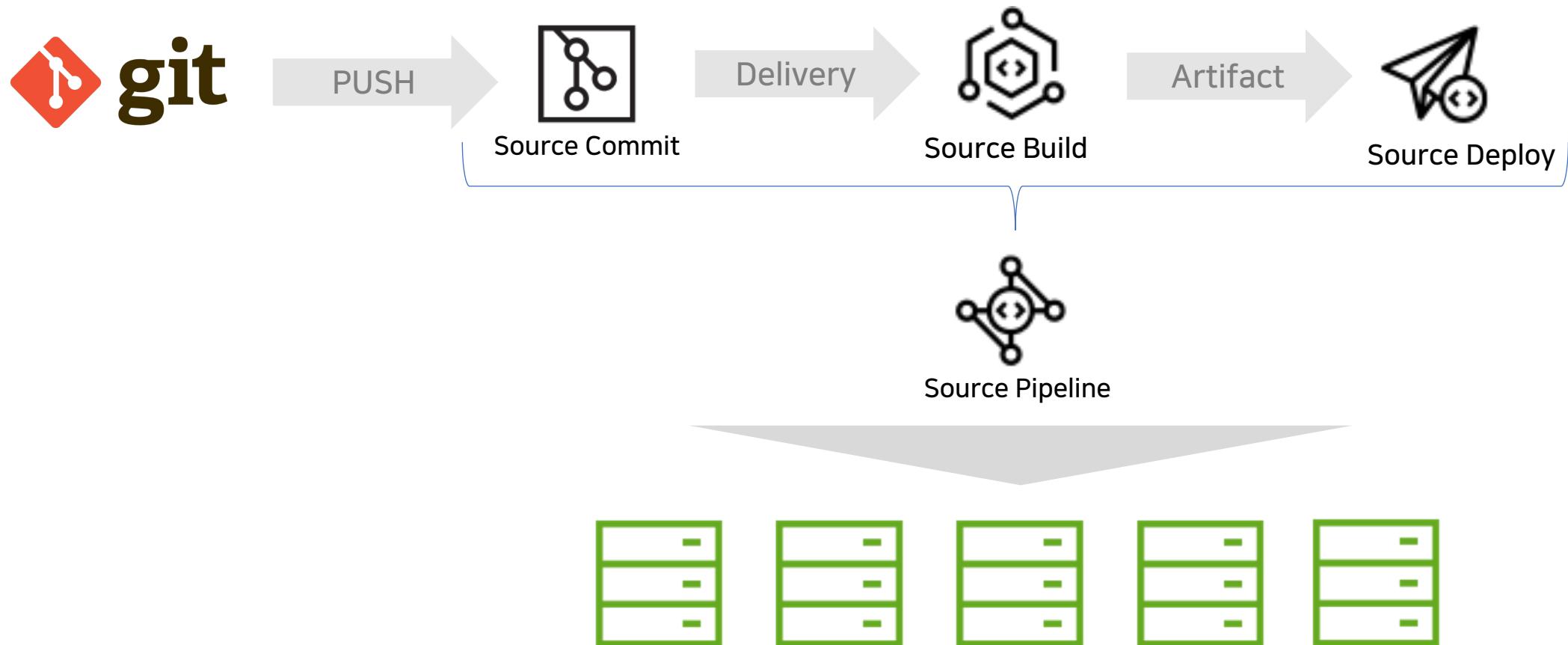


설계부터 개발, 테스트, 생산 및 운영까지의 어플리케이션 라이프 사이클 전반에 보안을 통합

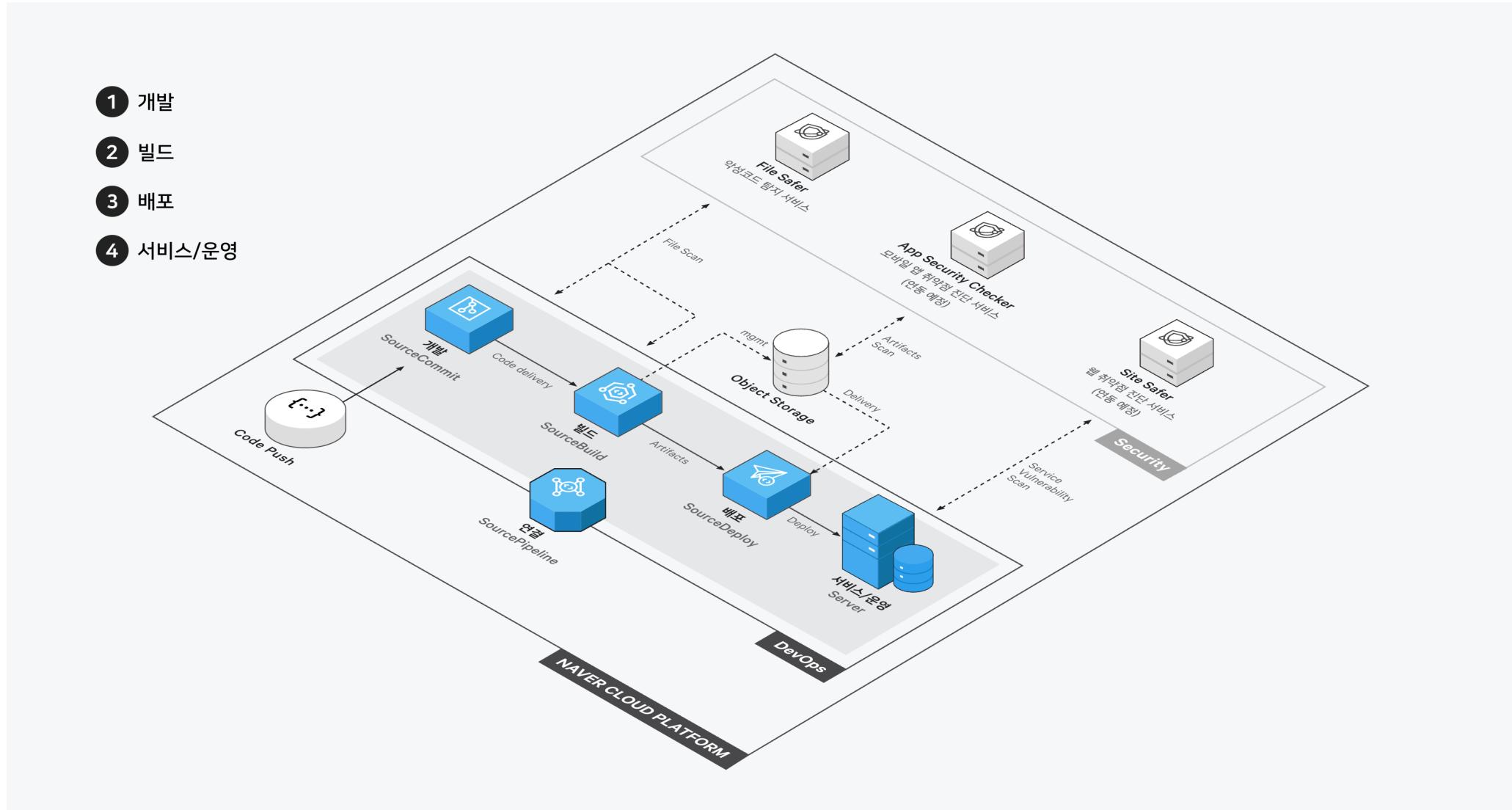
# Continuous Integration / Deployment



# 네이버 클라우드 플랫폼 CI/CD Pipeline



# 네이버 클라우드 플랫폼의 CI/CD Pipeline



# SourceCommit

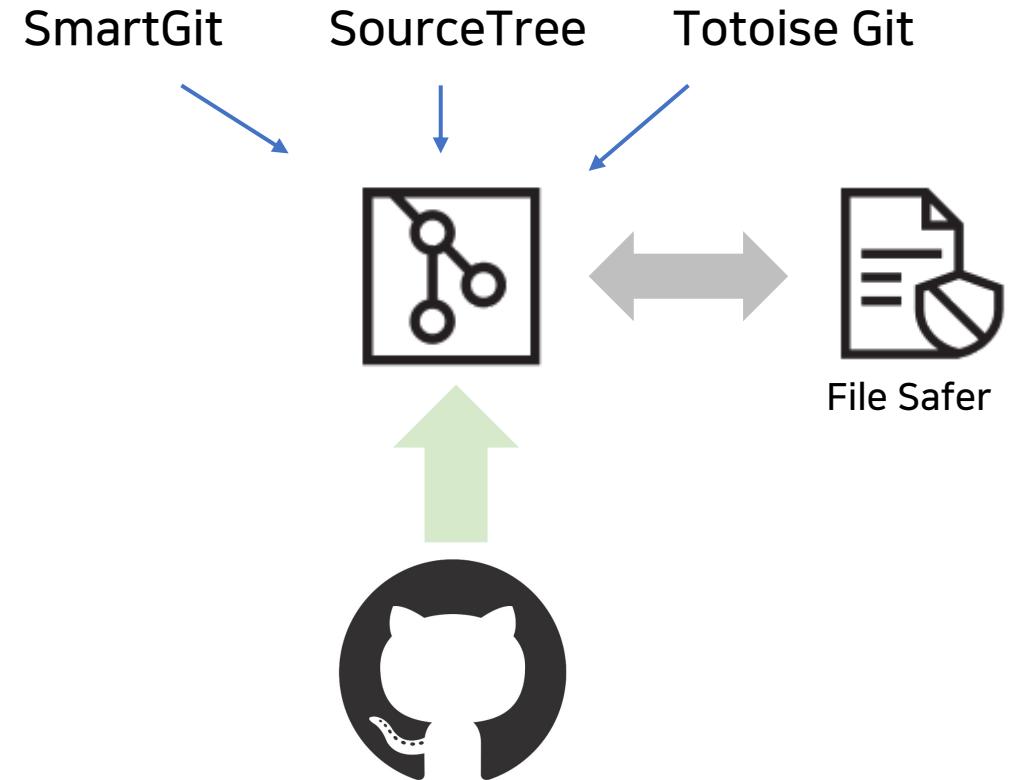
소스 리파지토리 제공

모든 Git 명령어 사용

다양한 Git Client 와 호환

외부의 Git Repository 와 연동(GitHub)

악성 코드 탐지를 위한 File Safer 연동



# SourceCommit – 보안상품 연동

The screenshot shows the SourceCommit interface for configuring external repository backups. The current step is '보안상품 연동' (Step 2). The interface includes a navigation bar with tabs: '기본 설정' (Step 1), '보안상품 연동' (Step 2, highlighted), '사용자 공유' (Step 3), and '최종 확인' (Step 4). A progress bar at the top indicates the completion of Step 1 and the start of Step 2.

**보안상품 연동**

리파지토리와 FILE SAFER 상품을 연동하여 보다 안전하게 리파지토리를 사용할 수 있습니다. 필수입력 사항입니다.

File Safer는 네이버에서 오랜기간 사용되고 검증된 악성코드 필터링 시스템입니다.  
개발자 포털에서 리파지토리에 업로드된 바이너리/스크립트 파일들의 악성 여부를 File Safer를 통해 검사 할 수 있습니다.

**보안상품 연동**

1  FILE SAFER (FILE FILTER)상품 연동하기

보안상품 연동 설정은 이후 리파지토리 환경 설정에서 변경 가능합니다.  
FILE SAFER 상품을 연동하실 경우, 사용량에 따라 요금이 발생할 수 있습니다. (FILE  
SAFER 요금 안내 바로가기)

< 이전      다음 >

# SourceCommit - 계정 권한 분리

네이버클라우드플랫폼 콘솔

개발자 포털

개인 PC

고객 계정(네이버클라우드플랫폼 계정)

리파지토리 생성

리파지토리 사용자 공유

리파지토리 삭제

Sub Account

리파지토리 생성

리파지토리 사용자 공유

리파지토리 사용자 삭제

리파지토리 기본 정보 확인

소스 코드 확인

커밋 히스토리 확인

브랜치 그래프 확인

Git 계정 설정

Push

Clone

Pull

Fetch

Admin 권한

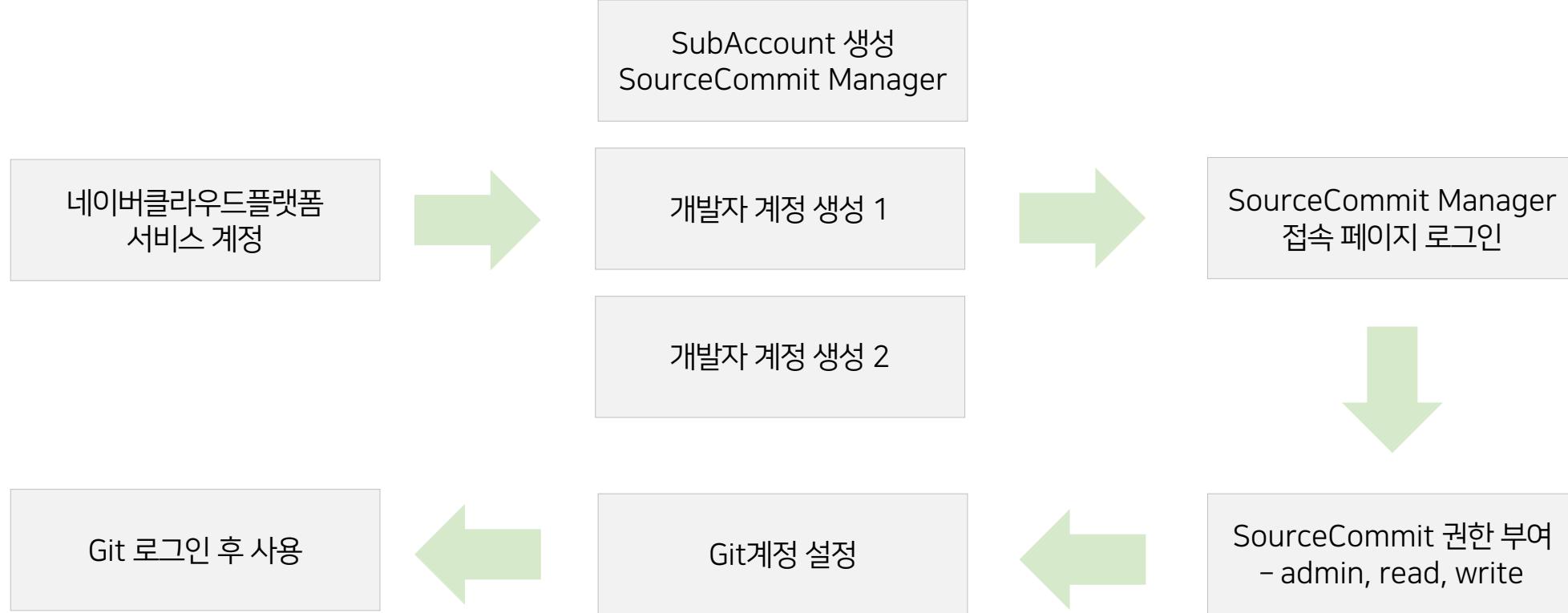
Write, Read 권한

Admin, Write, Read 권한

Admin, Write 권한

Read 권한

# SourceCommit – 계정 발급 및 권한 부여 절차



# SourceCommit – Git command 지원

명령어	설명
git init	현재 디렉터리를 git 저장소 설정
git add	새로운 파일 추가
git commit	변경사항 커밋
git clone	저장소 복제
git branch	브랜치 생성
git checkout	브랜치 이동
git merge	브랜치 병합
git push	원격 저장소에 Push

# SourceCommit Hands-on

# SourceBuild

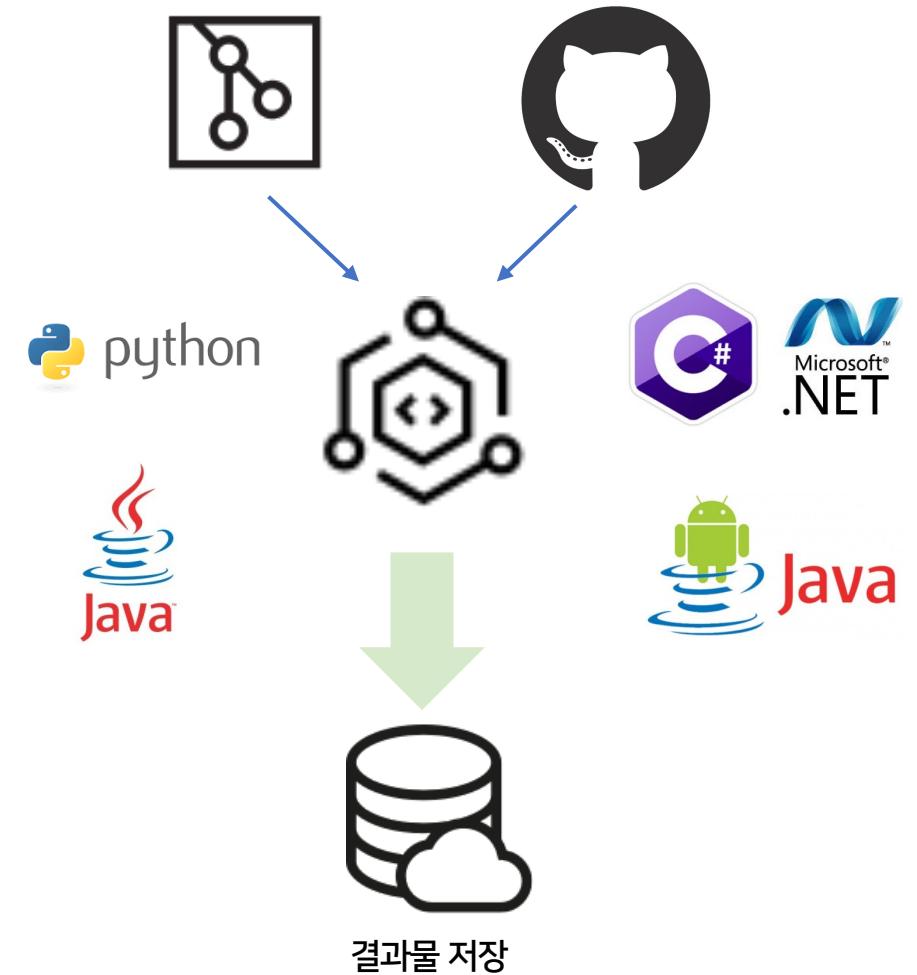
다양한 빌드 환경 제공

고성능의 전용 인프라 제공

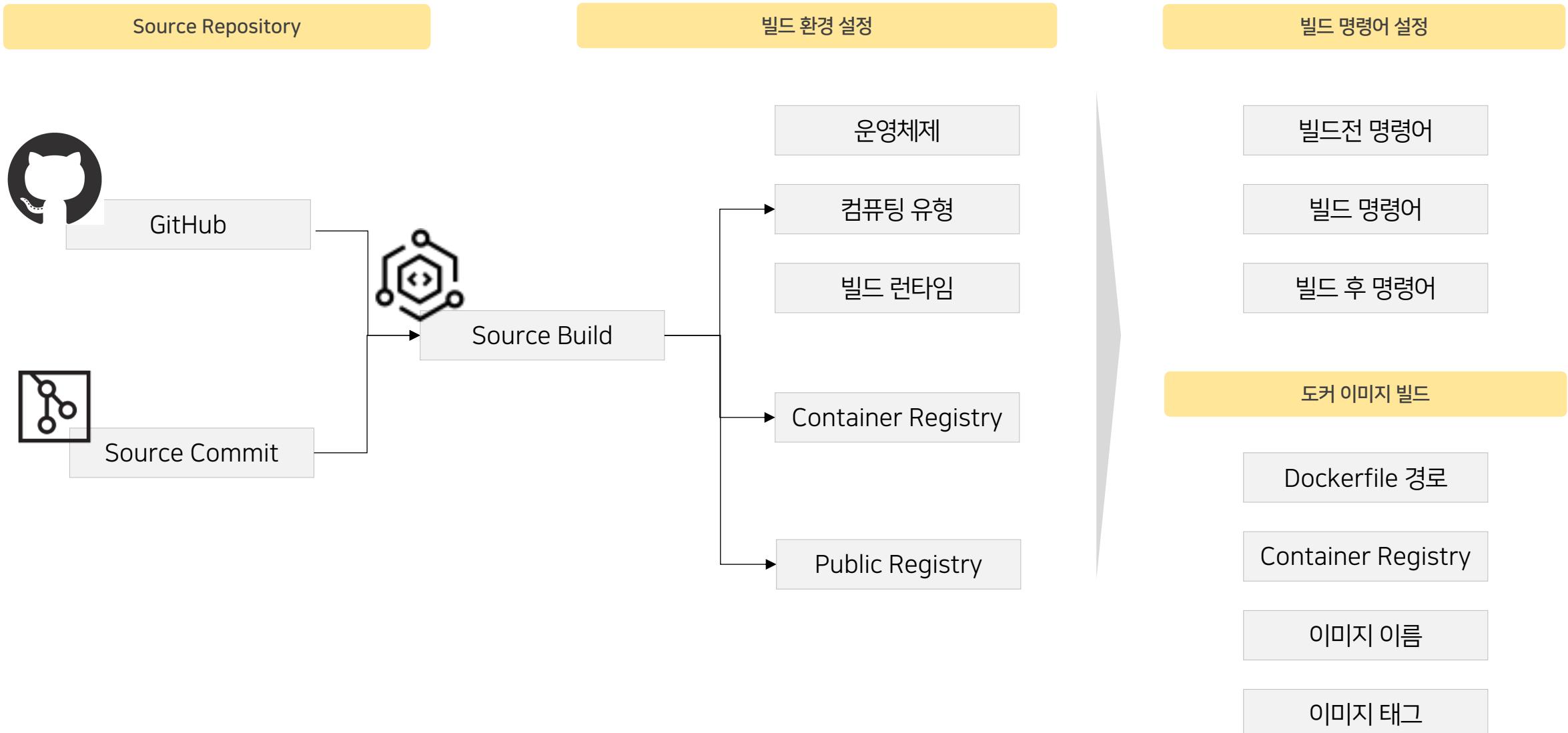
ObjectStorage에 빌드 결과물 저장

빌드 전후 명령어를 통한 작업

다양한 언어 제공



# SourceBuild 구성



# SourceBuild – 빌드 환경

**빌드 환경 설정**

빌드를 진행할 환경을 설정합니다. (• 필수 입력 사항입니다.)

운영 체제 •

컴퓨팅 유형 •

빌드 런타임 •

빌드 런타임 버전 •

타임 아웃 •  분

(최소 5분, 최대 540분)



# SourceBuild – Build Command

구분	설명	샘플
빌드 전 명령어	Build 시작 전 실행할 명령어 Build에 필요한 라이브러리 설치 파일 복사 작업 권한 변경	chmod +x ./gradlew apt-get install curl curl -sL <a href="https://deb.nodesource.com/setup_8.x">https://deb.nodesource.com/setup_8.x</a>   bash - apt-get install -y nodejs npm install vue
빌드 명령어	빌드를 수행할 스크립트 실행 라이브러리 실행	./gradlew build mvn package mvn clean install
빌드 후 명령어	빌드 결과물 파일 복사 압축 작업	cp ./target/web-* .jar ./deploy/

# SourceBuild Hands-on

# SourceDeploy 구성

## 배포 타겟



Server



AutoScalingGroup



Kubernetes Service

## 배포 파일 위치



SourceBuild



ObjectStorage

## 배포 전략

순차 배포

동시 배포

순차 배포

동시 배포

Rolling

Blue / Green

Canary

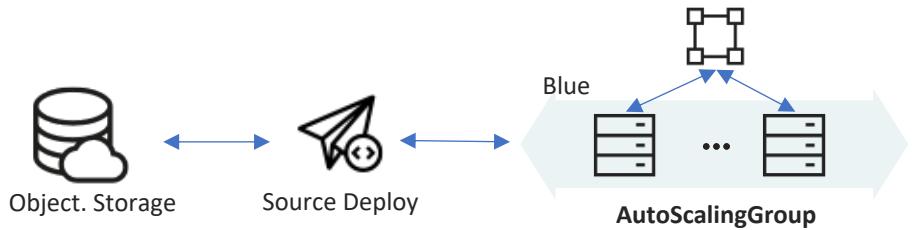
Server

Auto Scaling

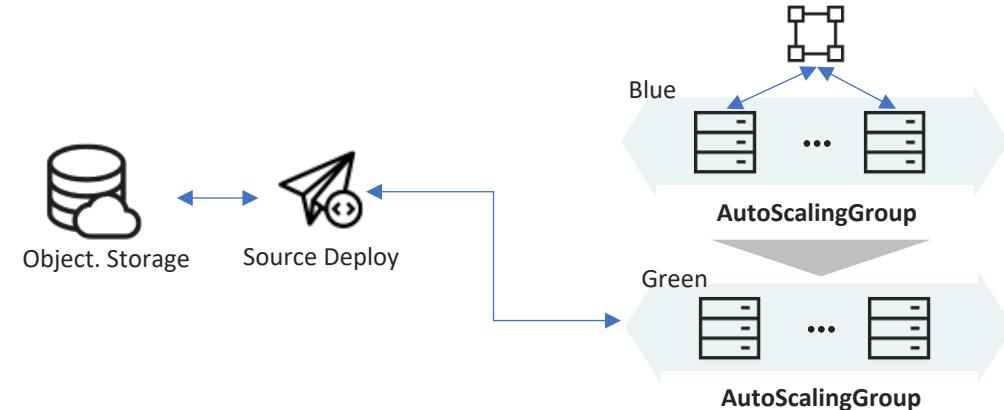
Kubernetes Service

# AutoScaling Blue/Green Deploy

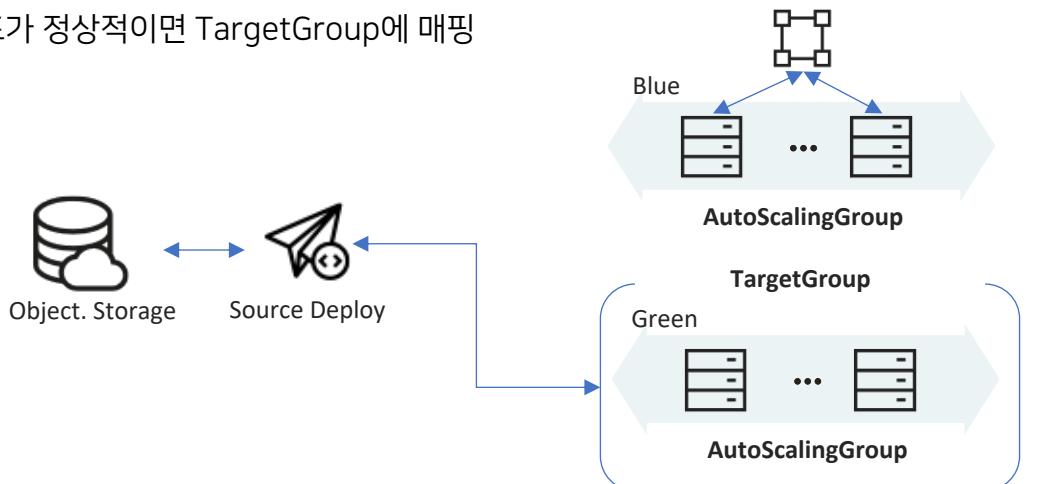
ObjectStorage에 빌드 결과물 저장  
SourceDeploy 를 통해 AutoScalingGroup에 배포



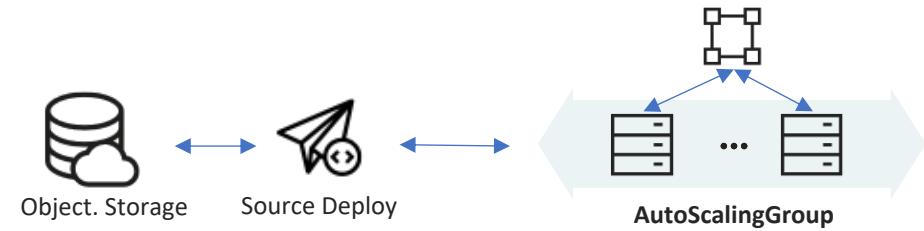
Blue/Green 배포시 AutoScalingGroup 생성후 배포



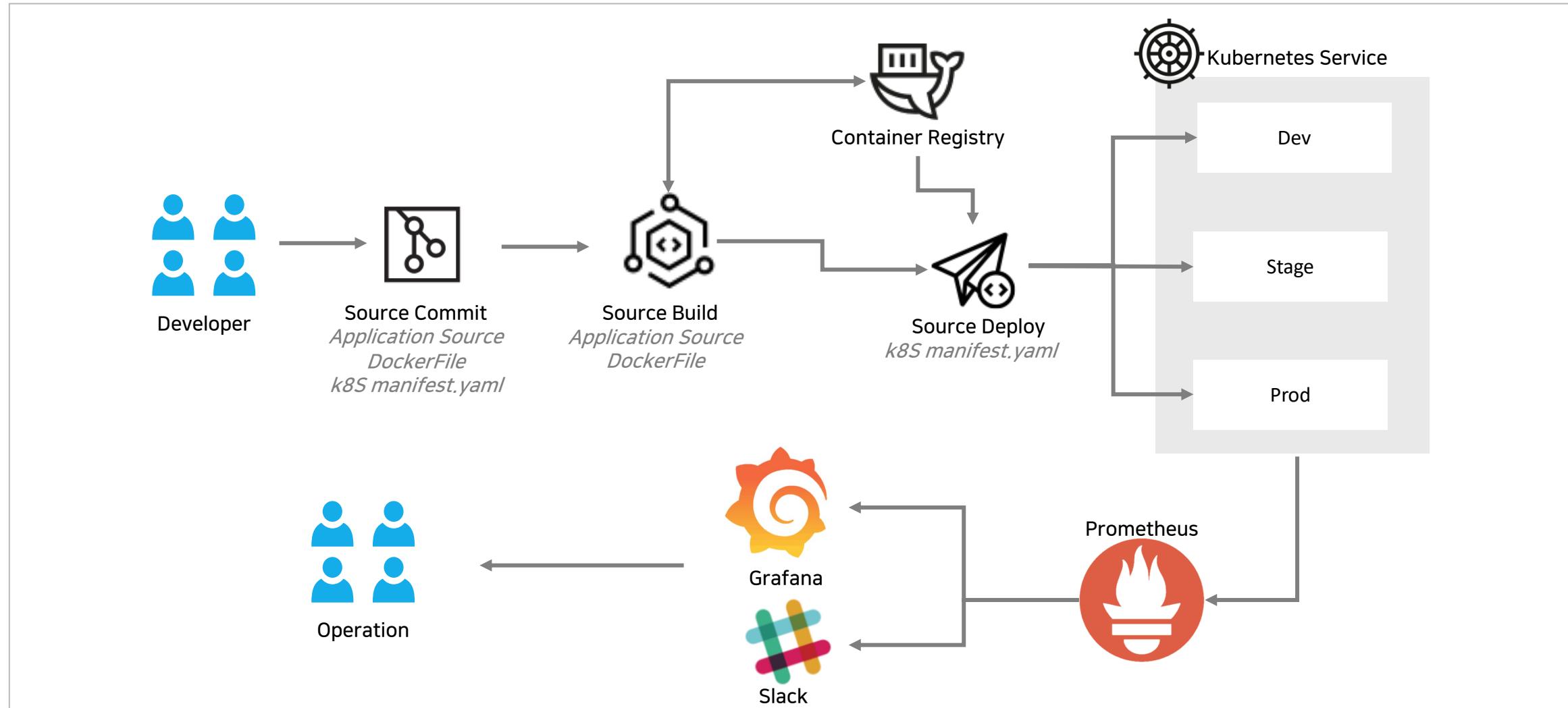
배포가 정상적이면 TargetGroup에 매팅



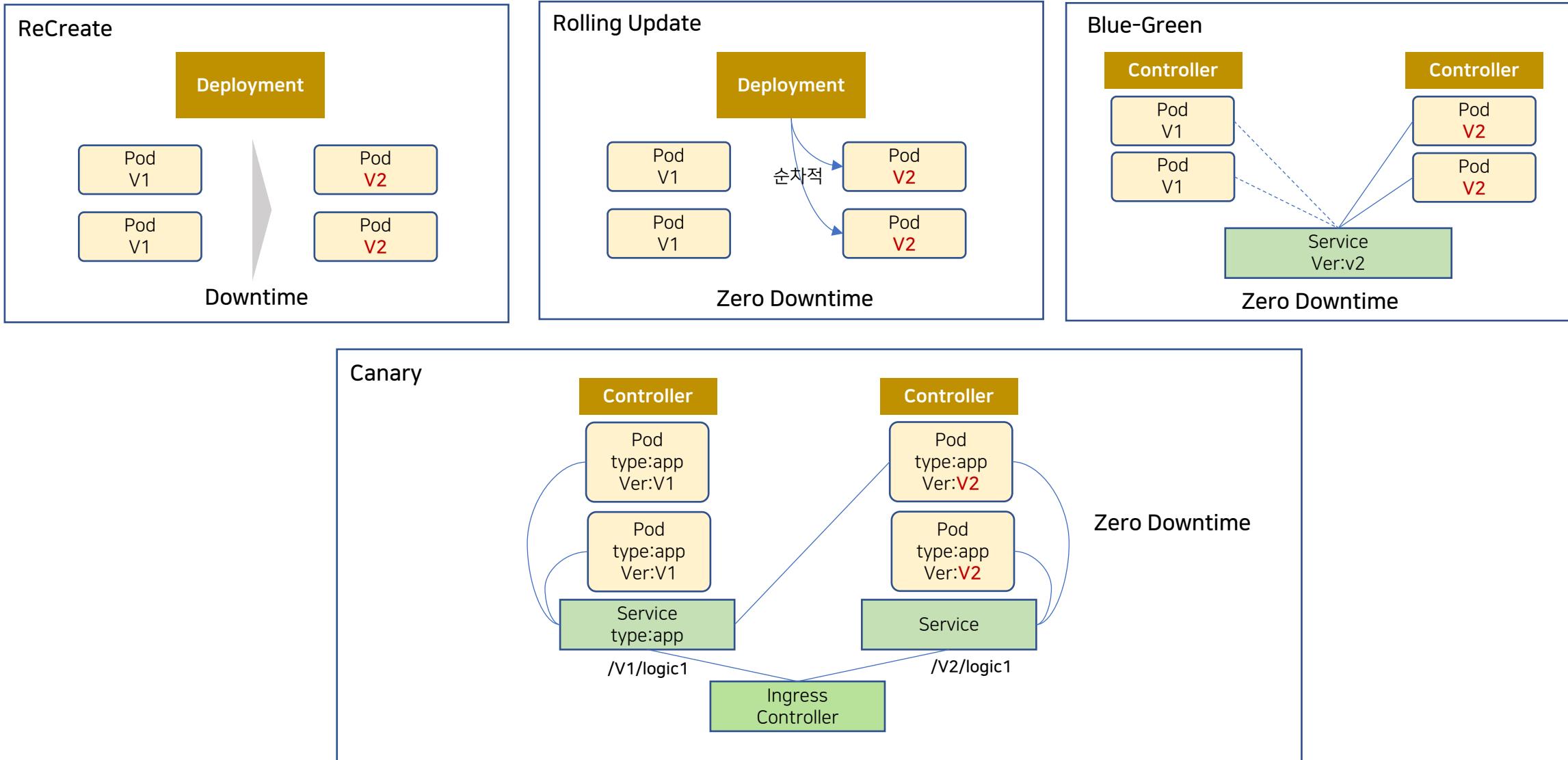
LoadBalancer를 통해 서비스



# SourceDeploy Kubernetes CI/CD



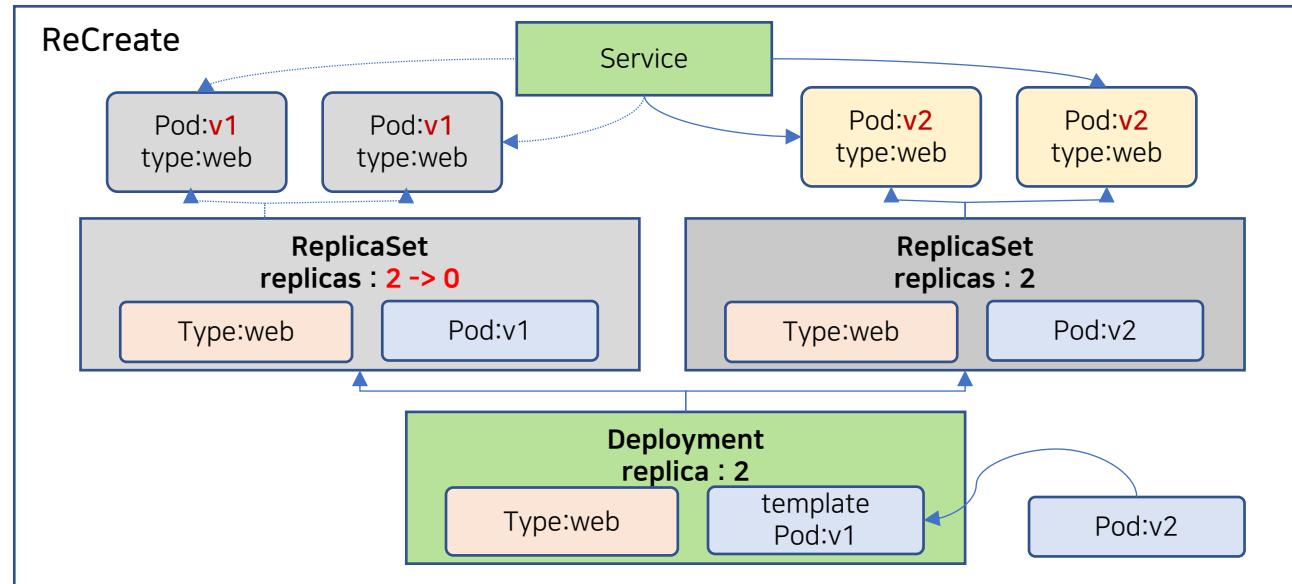
# Kubernetes Deploy



# Kubernetes Deploy

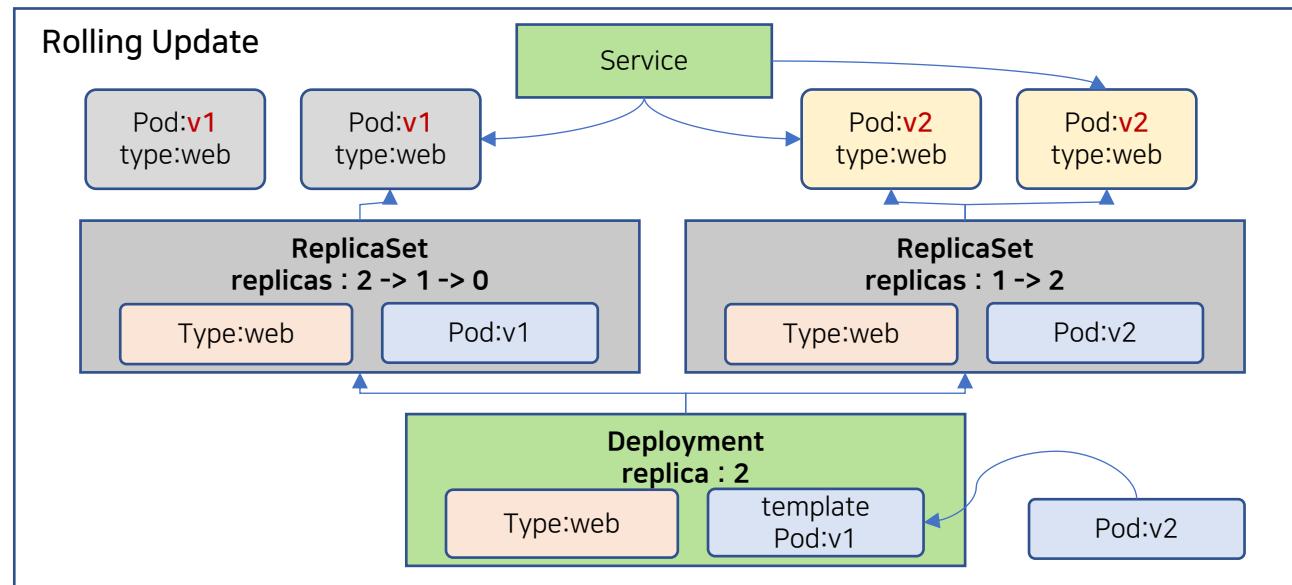
## ReCreate

- 이전 버전의 Pod를 모두 삭제 후 신규 버전의 Pod 생성
- 서비스 중단 발생
- 개발 및 테스트 환경에 적합



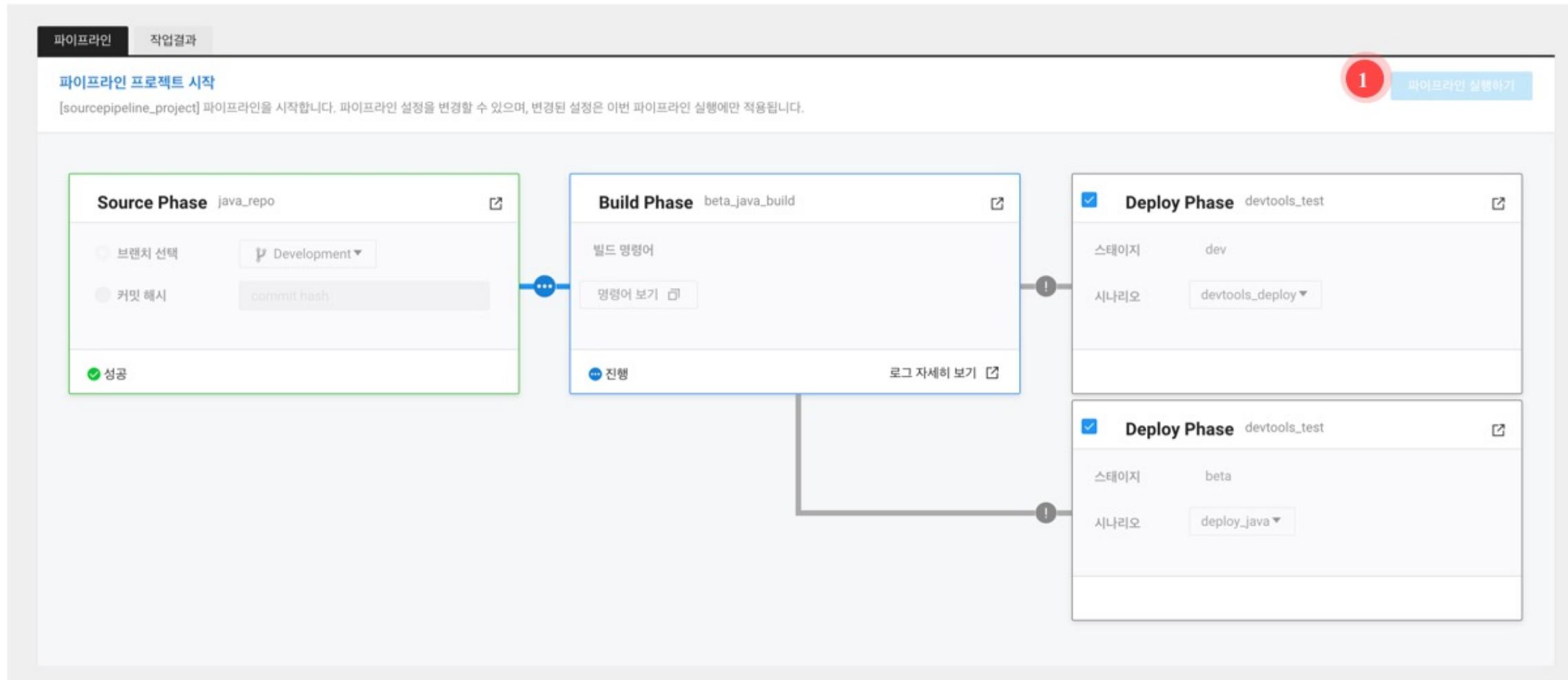
## Rolling Update

- Pod를 점진적으로 신규 버전으로 업데이트
- 이전 버전으로 룰백 가능
- 서비스 중단 없음

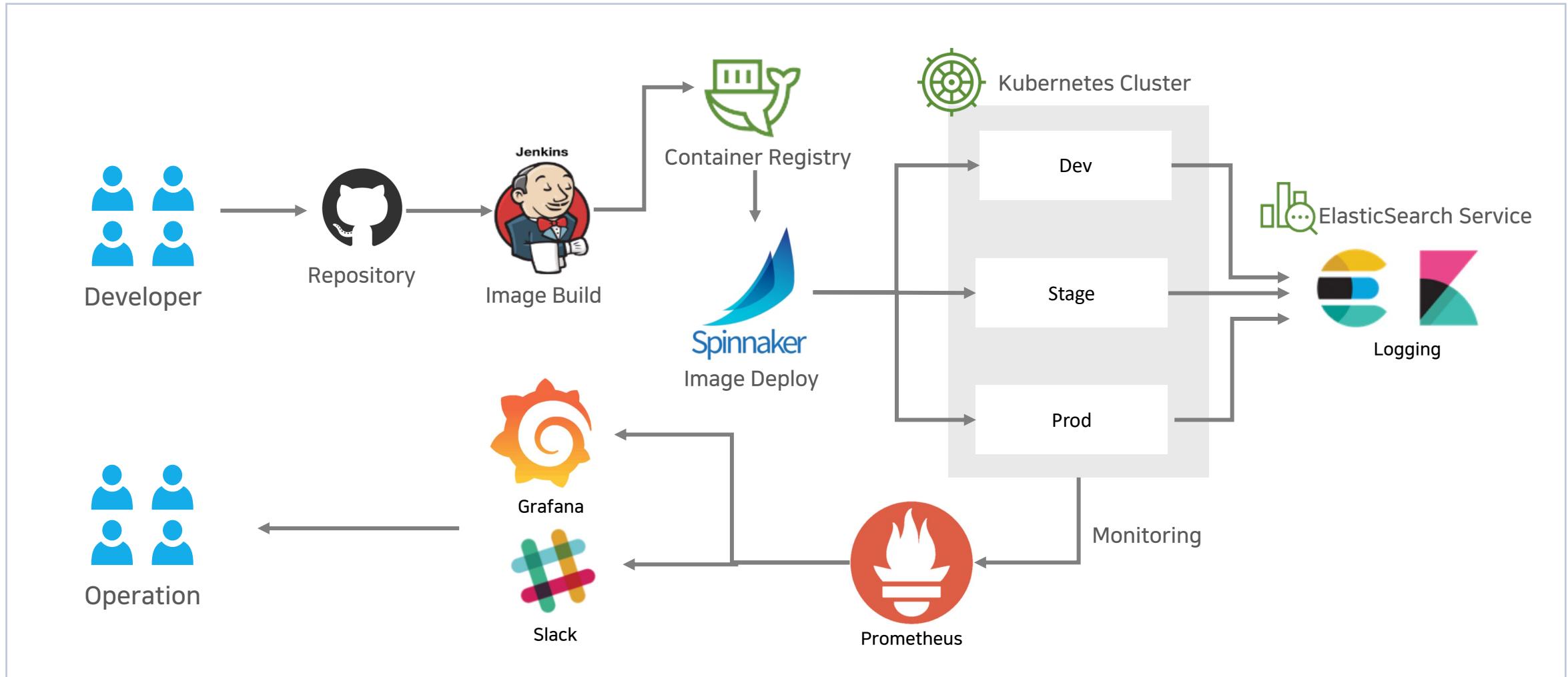


# SourceDeploy Hands-on

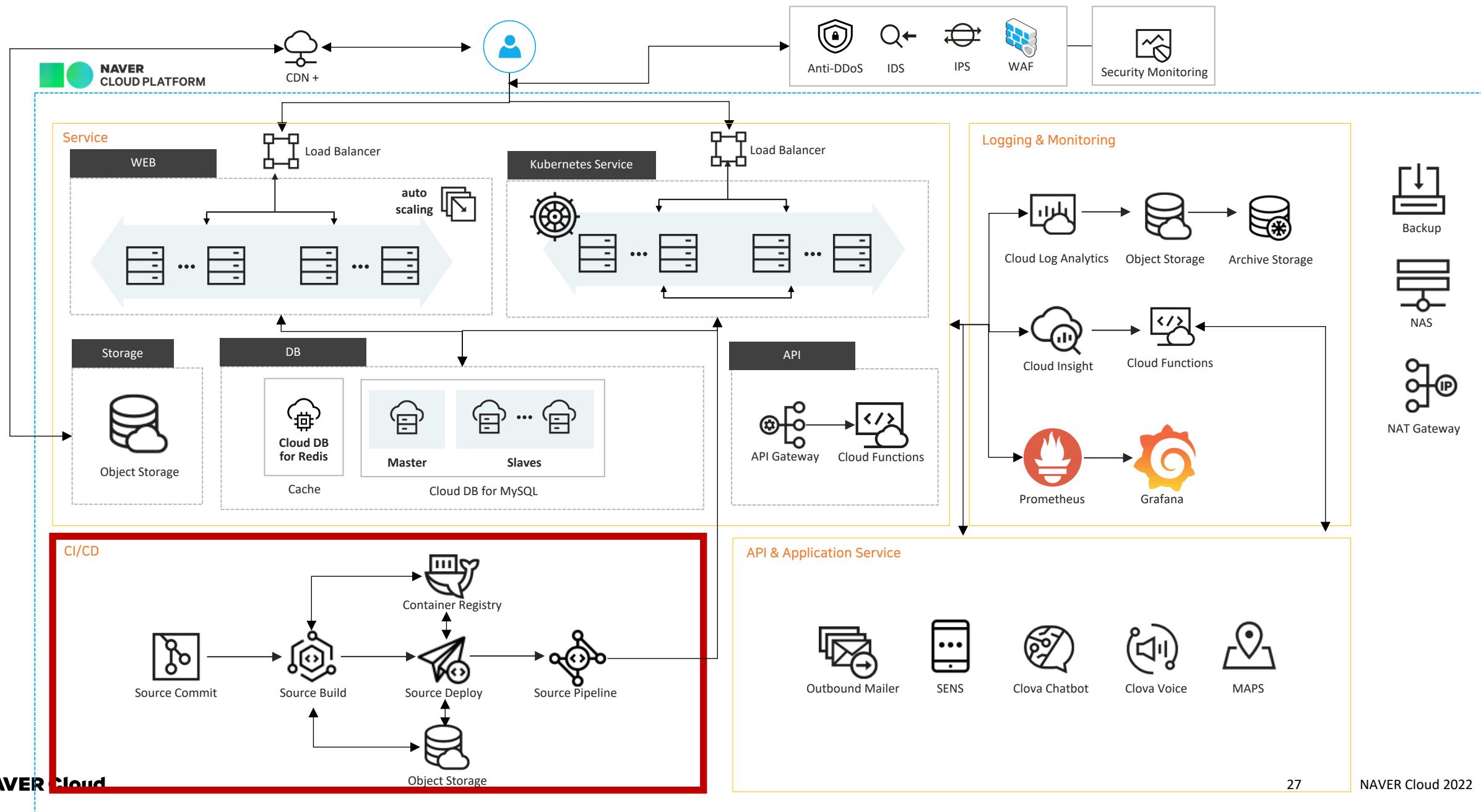
# SourcePipeline



# CI / CD / Monitoring



# Reference Architecture



# DevOps와 모니터링

# Monitoring이란?



시스템에서 문제 발생시

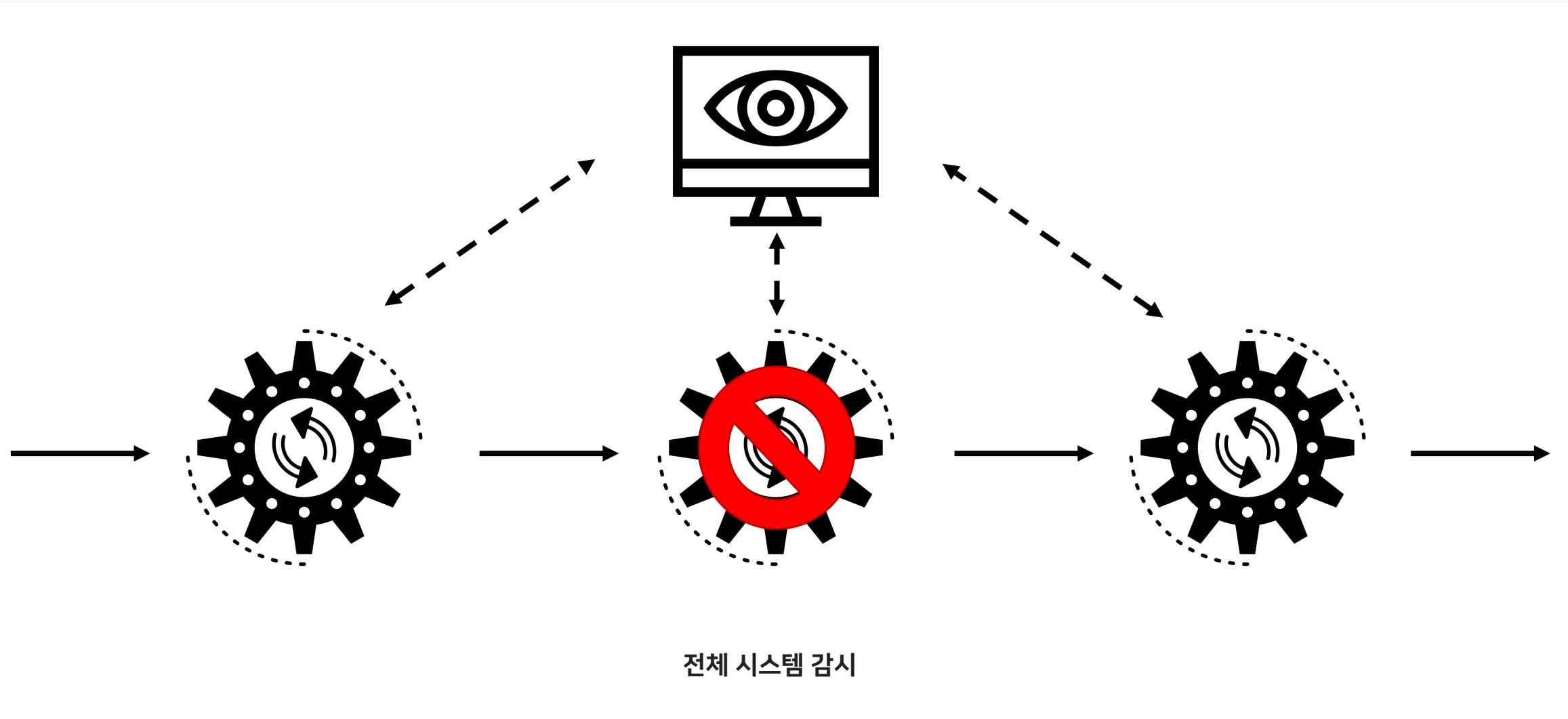
# Monitoring이란?

문제 발생시 경험적인 방법으로 처리 가능

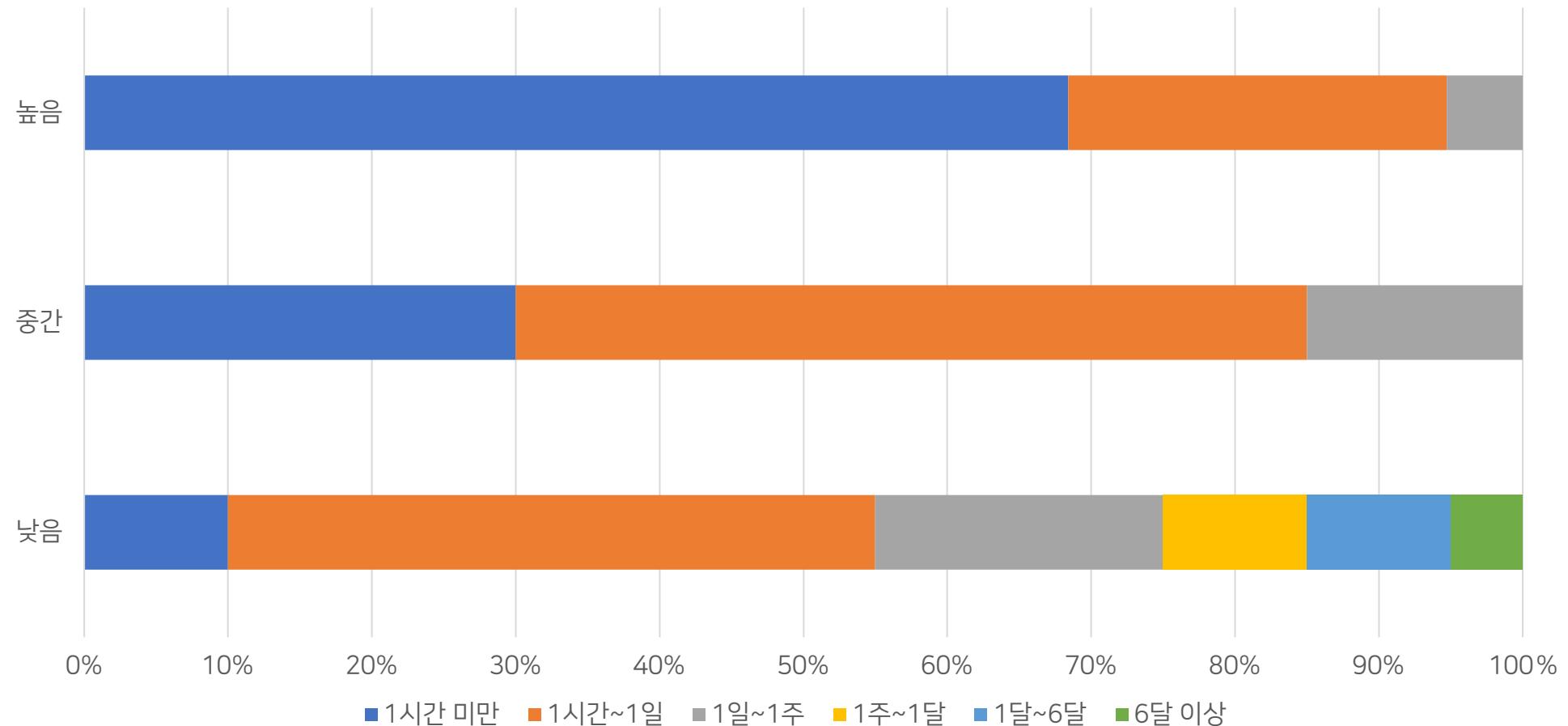
## 서버 접속이 안되면

1. 서버를 재부팅한다.
2. 그래도 정상 동작하지 않으면, 그 옆에 있는 서버를 재부팅한다.
3. 그래도 안되면 모든 서버를 재부팅한다.
4. 이렇게 해도 작동하지 않으면, 개발자를 비난한다.

# Monitoring이란?



# 높은, 중간, 낮은 성과를 내는 사람들의 사고 처리 시간



출처 : Puppet Labs, State of DevOps Report

# 모니터링 플랫폼 종류

## Application Performance Monitoring

Zabbix  
Splunk  
Prometheus  
Nagios  
Sysdig  
Sensu  
Sematext

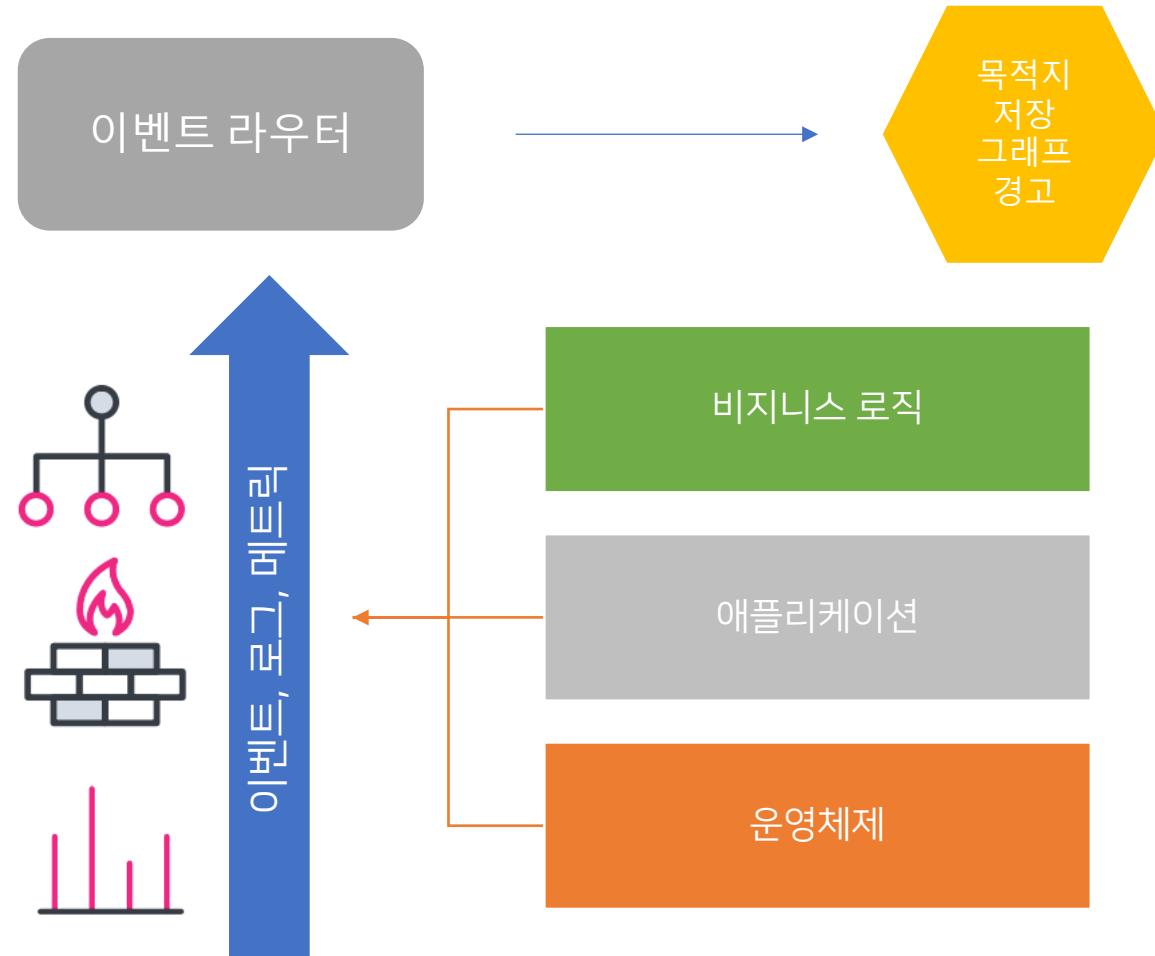
## Server Monitoring

SignalFx  
AppDynamics  
Raygun  
New Relic

## Network Monitoring

Catchpoint

# 모니터링 프레임워크 구조



출처 : Turnbull, The Art of Monitoring

# 수집되는 정보의 종류



## Metrics

- System Metrics(CPU, memory, disk)
- Infrastructure Metrics
- Web Tracking script(Google Analytics 등)



## Event(Logs)

- 일반적으로 3가지 종류(Plain Text, Structured, Binary)
- System과 Server log(syslog 등)
- Application, platform logs(log4j, apache, MySQL)

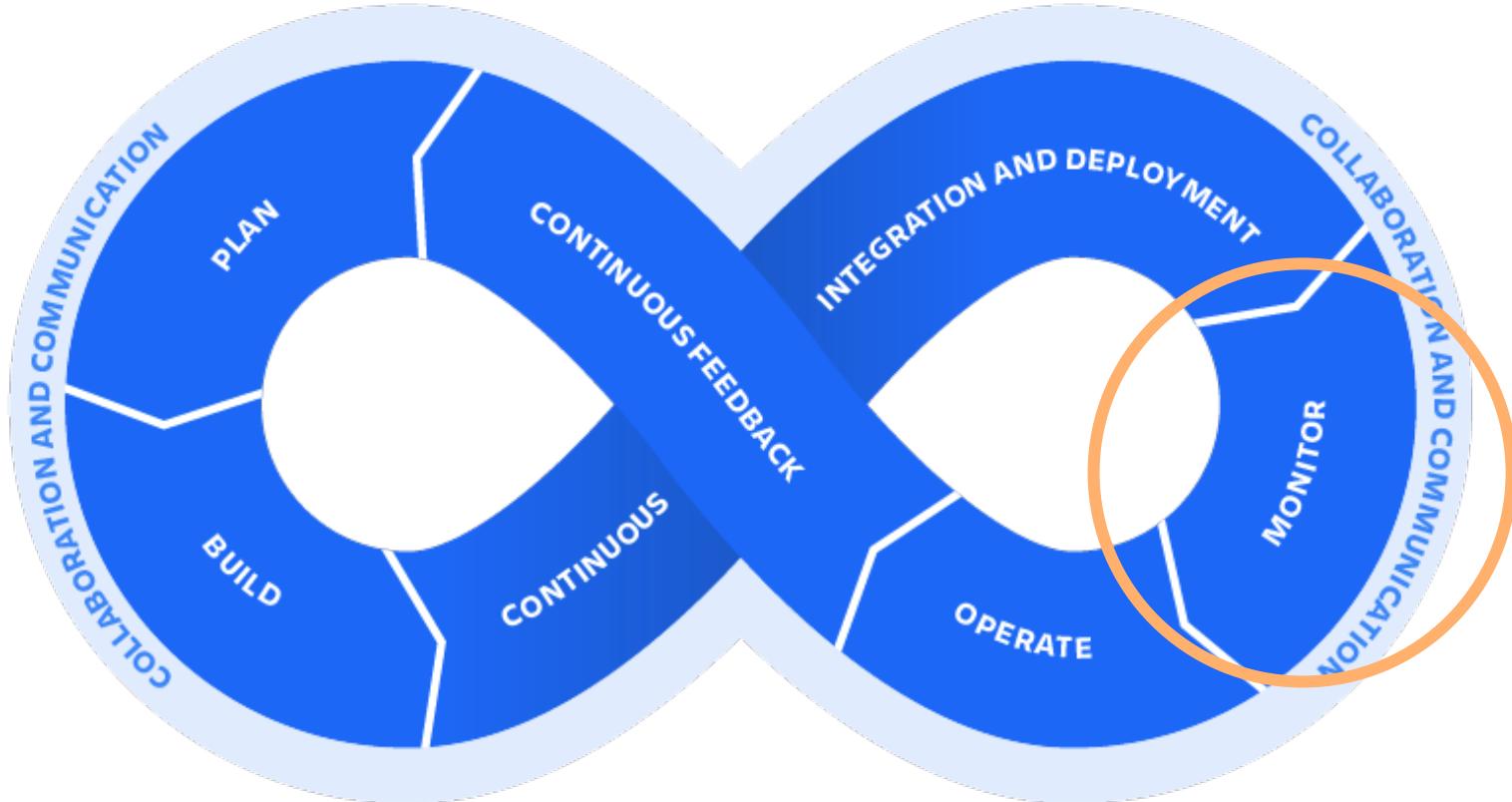


## Traces

- 호출된 서비스 경로
- Container, host, instance 등
- APM

출처 : Splunk, Monitoring Observability

# DevOps Monitoring이란?



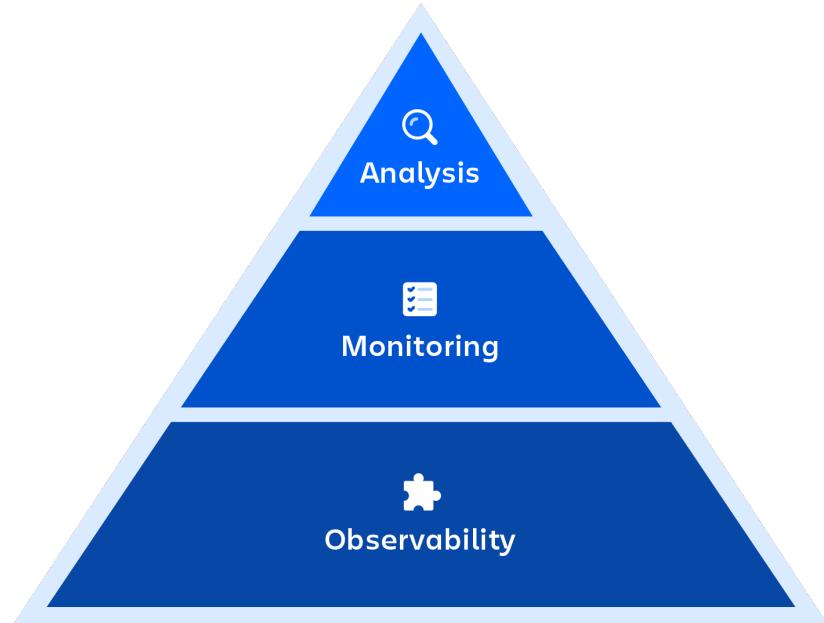
DevOps 사이클

출처 : Atlassian, DevOps Monitoring

## 주요 역할

- 빈번한 코드 변경 가시성
- 자동화된 협업
- 최적화를 위한 테스팅 지원
- 변경 사항 관리
- 파편화된 시스템 관리

# DevOps monitoring vs. observability



	Monitoring	Observability
목적	<ul style="list-style-type: none"><li>• 시스템이 동작하는지 확인</li></ul>	<ul style="list-style-type: none"><li>• 왜 동작이 안되는지 확인</li></ul>
방법	<ul style="list-style-type: none"><li>• 시스템에서 메트릭 및 로그 수집</li></ul>	<ul style="list-style-type: none"><li>• Data로 유용한 정보 수집</li></ul>
범위	<ul style="list-style-type: none"><li>• 실패 중심</li></ul>	<ul style="list-style-type: none"><li>• 시스템 전반 동작 정보</li></ul>

- Observability는 Monitoring을 포함하는 더 넓은 개념

출처 : Atlassian, DevOps Monitoring

# NAVER Cloud Platform 모니터링 서비스

# Cloud Insight

## 클라우드 환경의 가시성/통찰력 확보

### 지표 조회 및 시각화

네이버 클라우드 플랫폼 상품의 성능/운영 지표를 다양한 형태로 시각화  
사용자 설정에 따른 Custom Metric 지원  
현재 Compute, Networking, Analytics 지원

### 사용자 대시보드 구성

성능 / 운영 지표를 위젯 형식으로 만들어 나만의 대시보드 생성

### Event Rule 및 Event 관리

장애 조짐이 의심되는 상황 및 실제 장애 상황을 식별할 수 있도록 Event Rule 생성  
각 Event마다 담당자 지정 가능

### 유지 보수 일정 관리

유지 보수 일정을 등록하여 해당 시간에 알림

# Cloud Insight

## 클라우드 환경의 성능/운영 지표를 통합 관리

VPC / Cloud Insight(Monitoring) / Dashboard

Dashboard

대시보드에 다양한 위젯을 추가해 원하는 정보를 한눈에 확인할 수 있고, 다수의 대시보드를 생성 관리할 수 있습니다.

+ 대시보드 생성      상품 더 알아보기      새로고침      자동 새로고침(60s)

Service Dashboard / VPC Server

+ 위젯 추가      대시보드 관리      위젯 데이터 다운로드      위젯 데이터 변경

TOP 10      그룹      전체 리소스

Target      검색어를 입력하세요.

Network In Average      CPU Utilization Average      File System Utilization Average      Memory Utilization      Network Out Average

Target	Network In Average	CPU Utilization Average	File System Utilization Average	Memory Utilization	Network Out Average
s17748647696	1912	7.575377	31.754055	52.88314	3232
smzcs	13270.8	3.1069677	6.825034	6.312352	13023.733
s1773d55df9e	678.4	1.6775162	5.8898726	14.472701	1570.8
s177ec5ad9a9	719.3333	1.5368361	5.877675	6.8728557	1586.4
s176aced3705	520.5333	1.5024819	4.6047487	7.340762	1410.9333

GMT+9(South Korea)      1H      6H      12H      1D      1W      Custom      2021-03-09 15:07 ~ 2021-03-09 16:07      현재

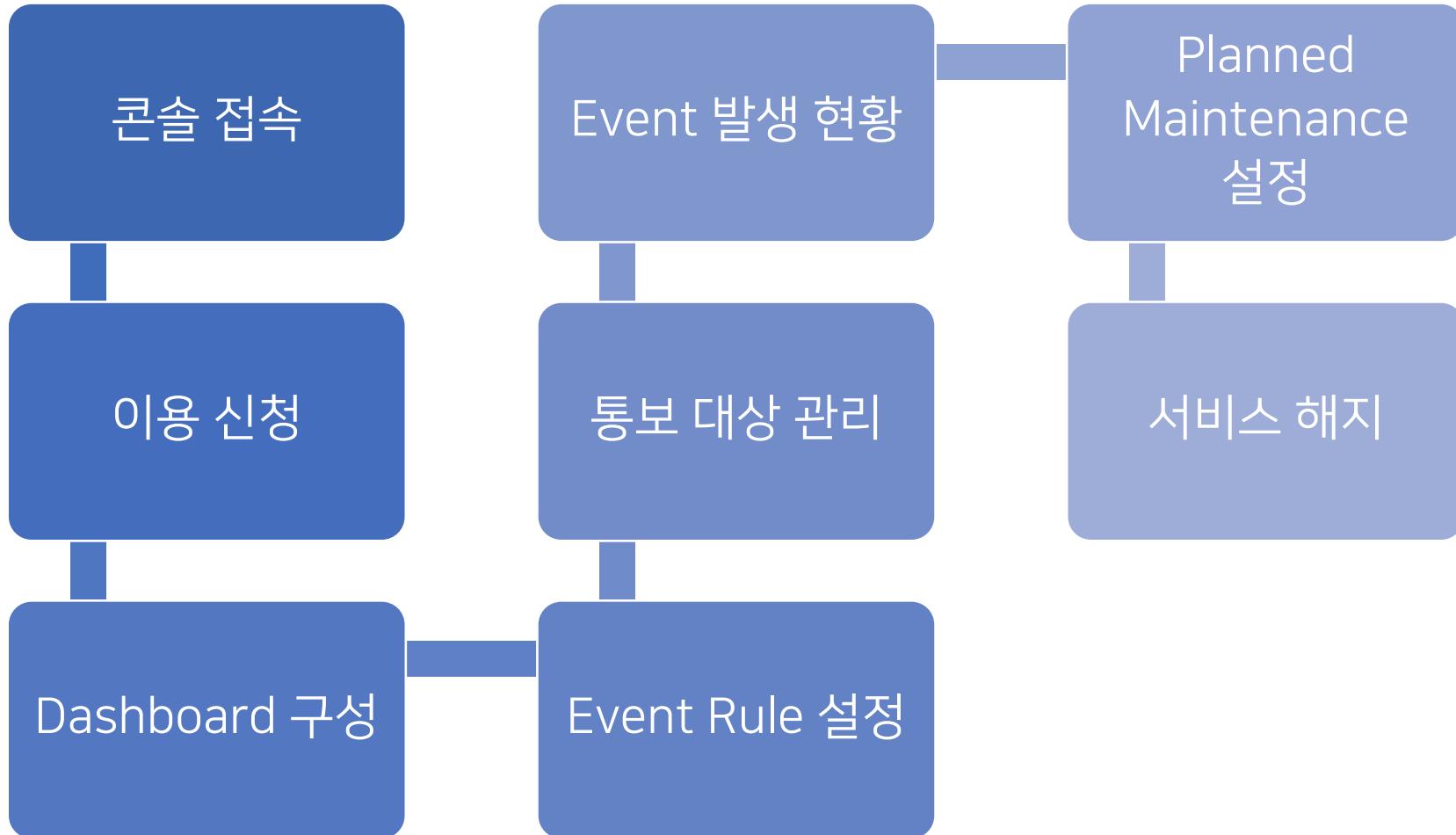
Network In Average      CPU Utilization Average      File System Utilization Average

Memory Utilization      Network Out Average

The screenshot shows the Cloud Insight Monitoring Dashboard for a VPC server. At the top, there's a navigation bar with 'VPC / Cloud Insight(Monitoring) / Dashboard'. Below it is a 'Dashboard' section with a note about creating and managing dashboards. A table lists the top 10 targets with their respective metrics. Below the table are three line charts showing Network In Average, CPU Utilization Average, and File System Utilization Average over a one-hour period. The charts include multiple data series for different servers. At the bottom, there are additional tabs for Memory Utilization and Network Out Average.

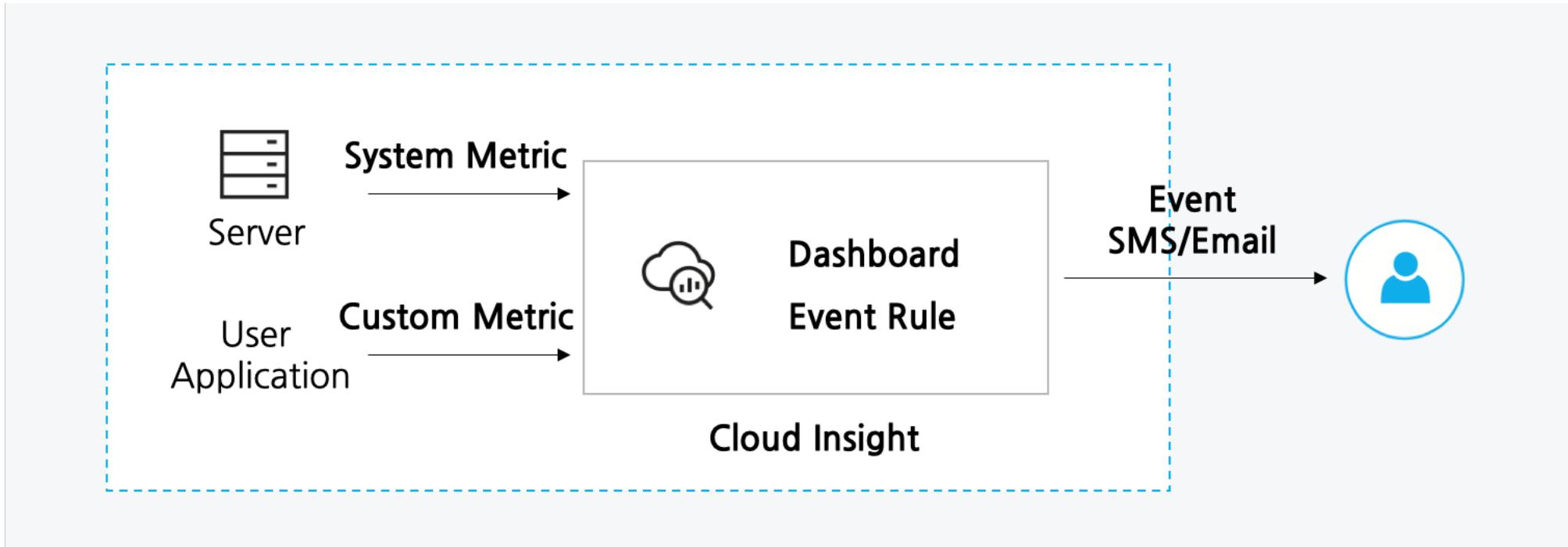
# Cloud Insight

## 전체 프로세스



# Cloud Insight

## 모니터링 시나리오



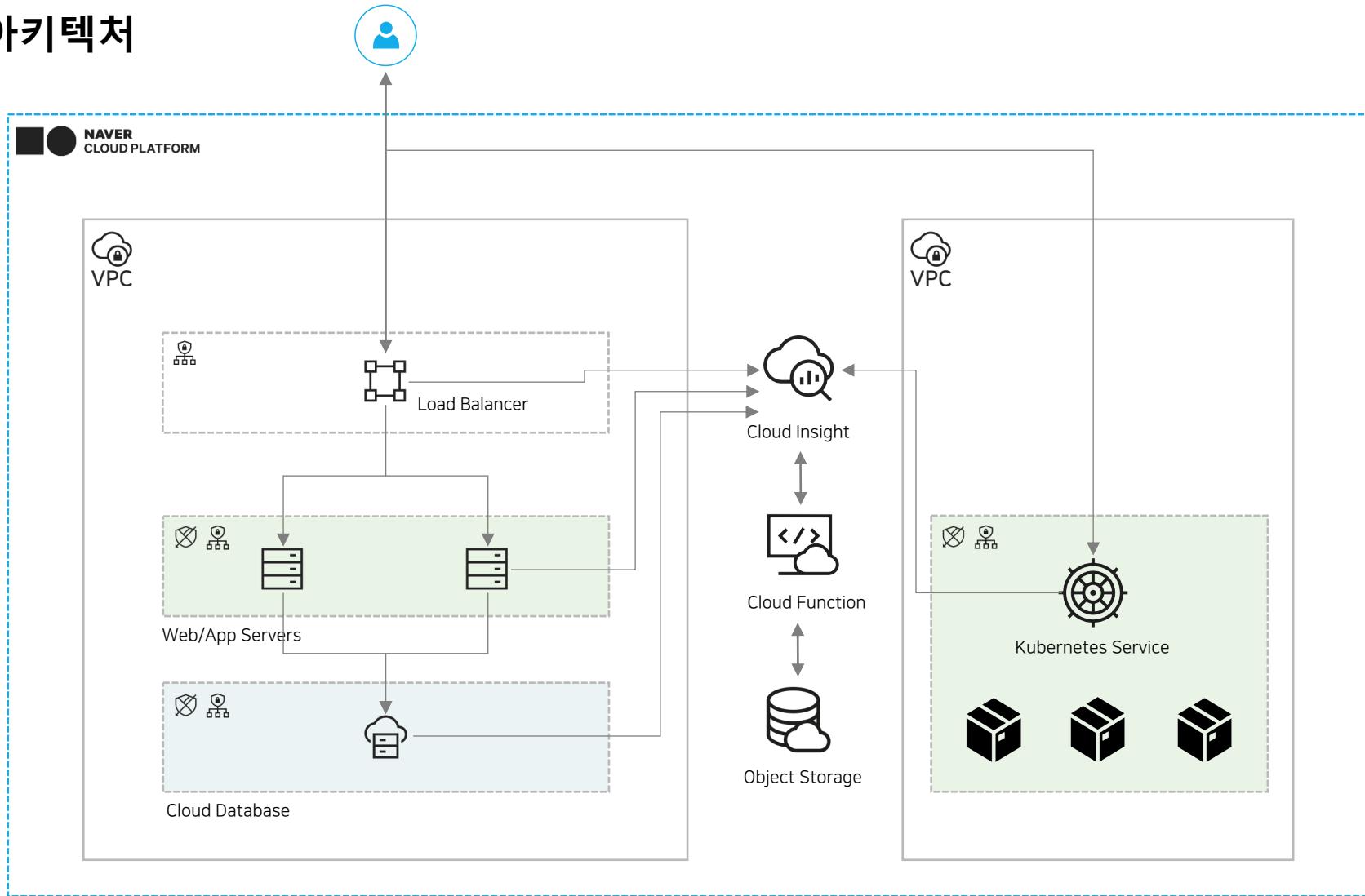
# Cloud Insight

## 지원 Metric

구분	설명
Basic Metric	<ul style="list-style-type: none"><li>네이버 클라우드 플랫폼의 각 상품에서 제공하는 기본 성능/운영 지표</li></ul>
Extended Metric	<ul style="list-style-type: none"><li>기본 성능/운영 지표인 Basic Metric 기능보다 더 상세한 모니터링이 필요한 사용자를 위해 네이버 클라우드 플랫폼의 각 상품이 추가 제공하는 성능/운영 지표</li><li>Extended Metric 사용을 위해서는 별도 수집 설정이 필요</li></ul>
Custom Metric	<ul style="list-style-type: none"><li>Cloud Insight가 제공하는 API를 통해 사용자가 직접 수집한 성능/운영 지표</li><li>사용자 애플리케이션 등의 성능/운영 지표를 Cloud Insight를 통해 수집</li></ul>

# Cloud Insight

## Cloud Insight 아키텍처



# Cloud Insight Hands-on

# Web Service Monitoring System

## 간단한 URL 등록만으로 사용 가능한 웹사이트 모니터링

### 실시간 모니터링을 통한 서비스 안정성 항상

- URL만 등록하면 웹페이지의 응답 속도 및 정상 동작 여부 실시간 확인 가능
- 모니터링은 분단위 확인 가능, 오류가 발생하면 SMS와 메일로 알림

### 시나리오 기반 모니터링 수행

- 사용자가 웹서비스를 이용하는 행동 패턴을 고려하여 시나리오 작성하면 해당 기능들 잘 동작하는지 확인 수행

### 글로벌 응답 속도 측정

- 미국, 싱가폴, 일본, 홍콩에서도 측정 가능

# Web Service Monitoring System

## 실시간 상태 분석

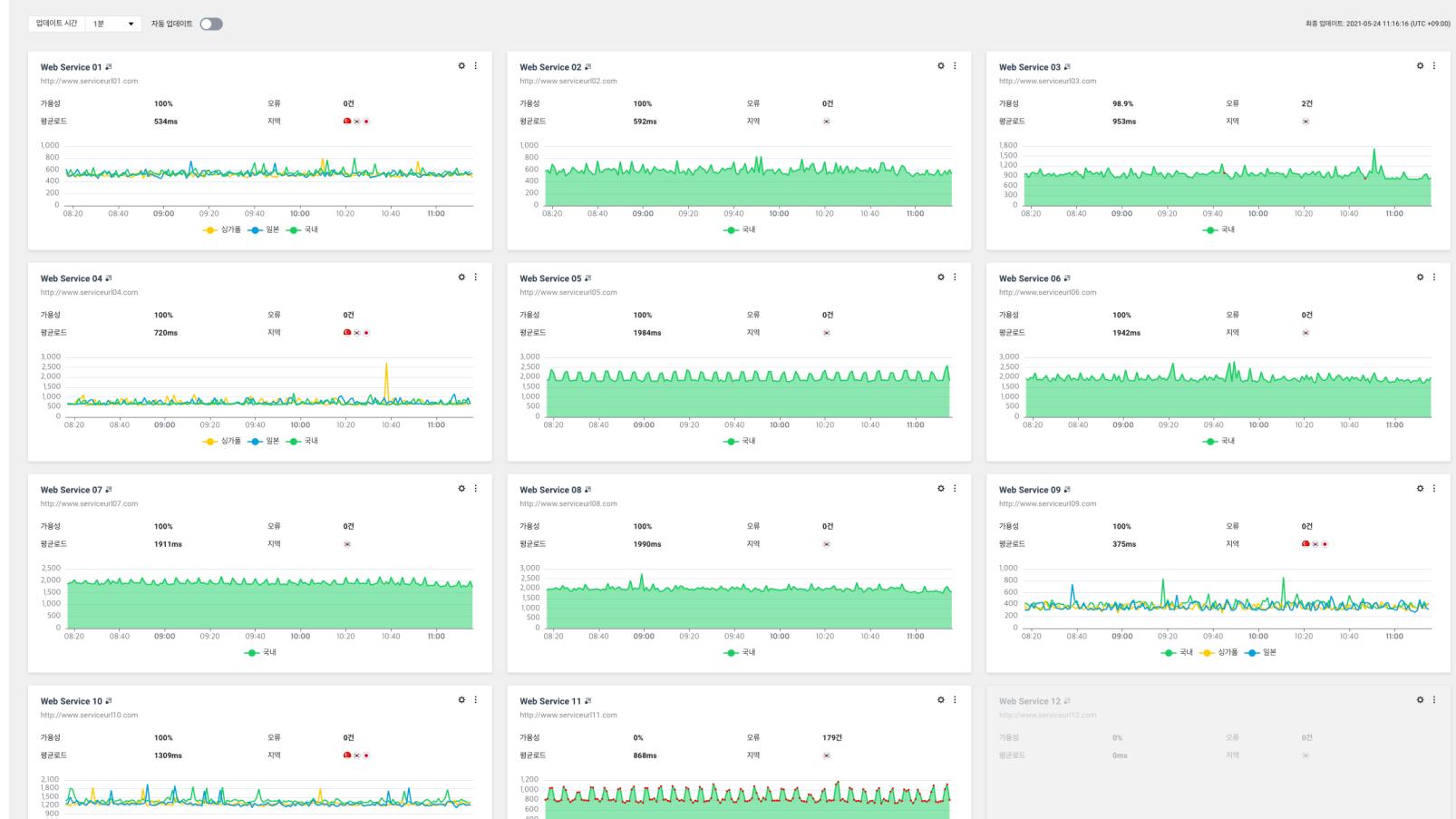
The screenshot shows the NAVER Cloud Platform interface with the 'Web Monitoring' section selected. On the left, there's a sidebar with 'Region 한국 / KR 한국어', 'All Products', 'Dashboard', 'My Products (1)', 'EDIT', 'WMS', 'Web Monitoring', 'Notification Setting', and 'Subscription'. Below that is a 'Recently Viewed' section with 'Server', 'Data Reporter', and 'Sub Account'. The main content area is titled 'WMS / Web Monitoring' and 'URL 등록'. It shows a form for setting up a monitoring task, including '모니터링 유형' (Virtual), '서비스 유형' (PC), '지역 선택' (국내), and '모니터링 URL' (http://www.naver.com). A '테스트 시작' (Start Test) button is present. To the right, under '모니터링 결과' (Monitoring Results), there's a summary card with metrics: Load Time (435ms), Page size (2.60MB), Request (92), and Error (0). Below this is a preview of the Naver homepage and a table of file requests:

File Requests	Method	Status	Type	Size	Time
1 http://www.naver.com/	GET	200	text/html	0.13KB	9ms
2 https://www.naver.com/	GET	200	text/html	28.61KB	52ms
3 https://pm.pstatic.net/css/main_v190926.css	GET	200	text/css	19.65KB	63ms
4 https://pm.pstatic.net/css/webfont_v170623.css	GET	200	text/css	0.22KB	70ms

로드시간, 페이지 크기,  
Request, Error 등 표시

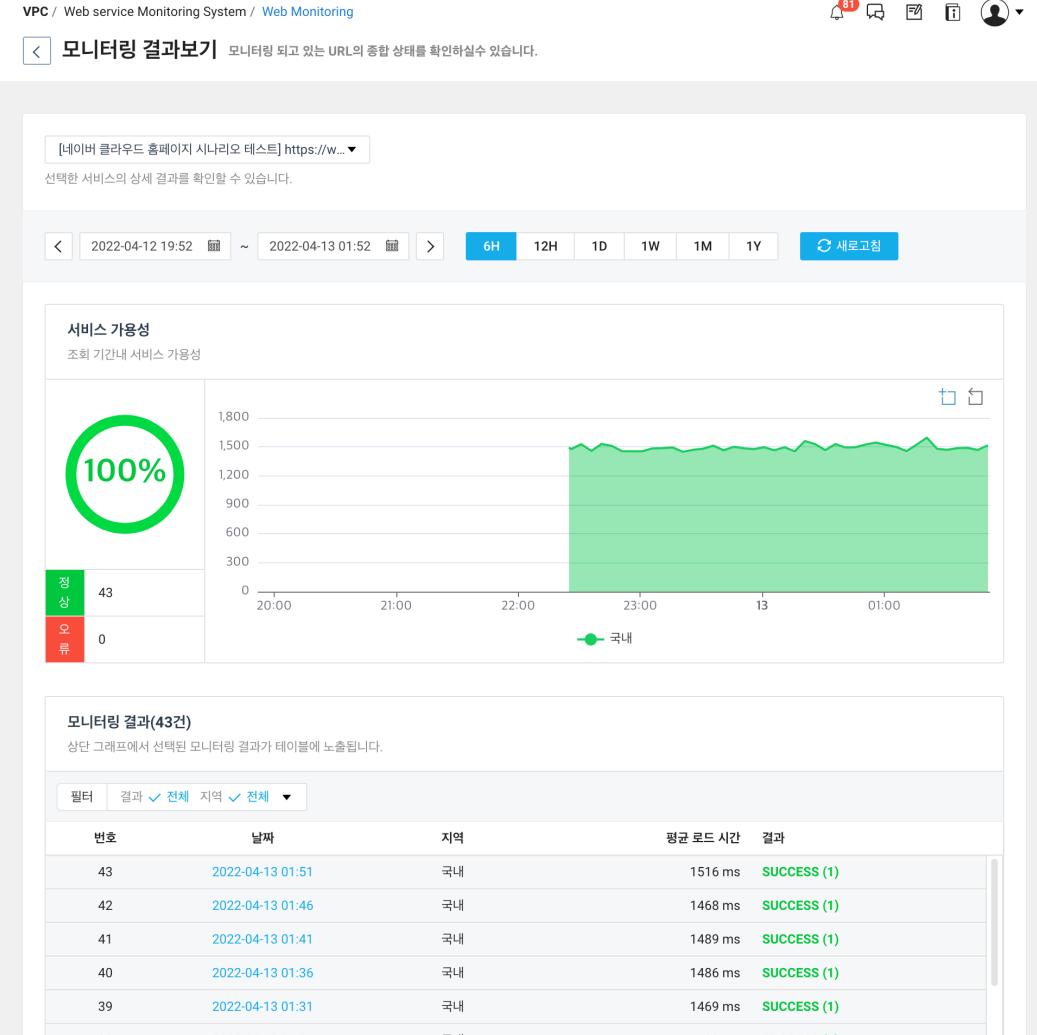
# Web Service Monitoring System

웹 서비스의 정상 동작 여부를 모니터링하고 글로벌 환경에서 응답속도 측정



# Web Service Monitoring System

## 서비스 상세 Metric 및 에러 로그



## 페이지 설명

1. 서비스 상세 페이지
2. 기간 조회
3. 서비스 가용성(오류 건수 / 정상 건수) \* 100
4. 차트 확대 / 되돌리기
5. 측정 지역
6. 모니터링 결과 조회 필터
7. 모니터링 상세 페이지

# Web Service Monitoring System

## 시나리오 기반 테스트 기능

VPC / Web service Monitoring System / Web Monitoring

서비스 등록 URL의 상태를 테스트하고 주기적인 모니터링이 가능하도록 등록할 수 있습니다.

1. 스텝 작성 2. 서비스 설정 3. 서비스 등록

테스트 환경 선택  
스텝을 테스트할 환경을 선택 합니다.

모니터링 유형 • URL SCENARIO 모니터링 하실 유형을 선택하세요.

서비스 유형 • PC MOBILE 모니터링 하실 서비스 유형을 선택하세요.

지역 선택 • 국내 홍콩 일본 싱가폴 미국(서부) 독일

스텝 작성  
스텝을 작성하고 테스트를 진행합니다.

URL 접속 / 입력한 URL에 접속 합니다. 음선 설정을 통해 Request Header, Body를 변경할 수 있습니다.

URL GET https://www.example.com/

텍스트 입력 / 대상을 찾아 텍스트를 입력합니다.

대상 \* input[id="box"] 텍스트 CSA

대기 시간 / 입력한 시간 만큼 대기 합니다.

시간 \* 1초

마우스 클릭 / 대상을 찾아 클릭 합니다.

대상 \* input[id=button]

마우스 클릭 결과 / 입력한 시간 만큼 대기 합니다.

시간 \* 1초

## 페이지 설명

1. URL 접속 후 수행할 작업 선택
2. 텍스트 입력, 대기시간, 마우스 클릭 등
3. 스텝을 자유롭게 추가하거나 삭제 가능

① URL 접속 → ② 아이디 입력 → ③ 대기  
→ ④ 로그인 버튼 클릭 → ⑤ 로그인 성공 확인

# Web Service Monitoring System Hands-on

# Network Traffic Monitoring

고객 서버에서 발생하는 트래픽을 수집하고 분석하여 정보를 제공

## Network 안정적 운용

- 고객의 서버에서 사용되는 네트워크 트래픽을 쉽게 확인하여 장애 분석

## 시나리오 기반 모니터링 수행

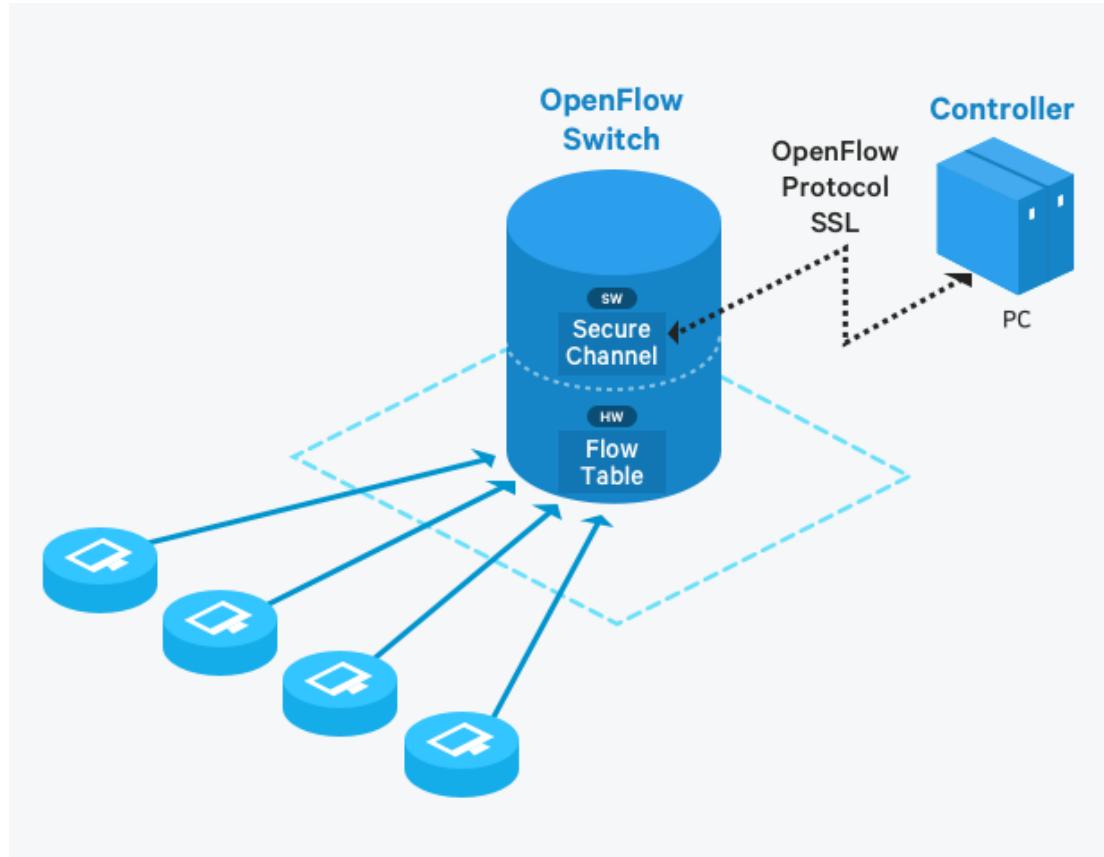
- 고객이 원하는 형태의 차트와 대시보드 생성 및 운영
- 서버 별로 분류도 가능하고 서버 그룹별로 상태 확인 가능

## 고객 서비스의 사용자 국가 분석

- 사용자의 국가 정보 기준으로 네트워크 모니터링 가능

# Network Traffic Monitoring

## 아키텍처 및 핵심 용어

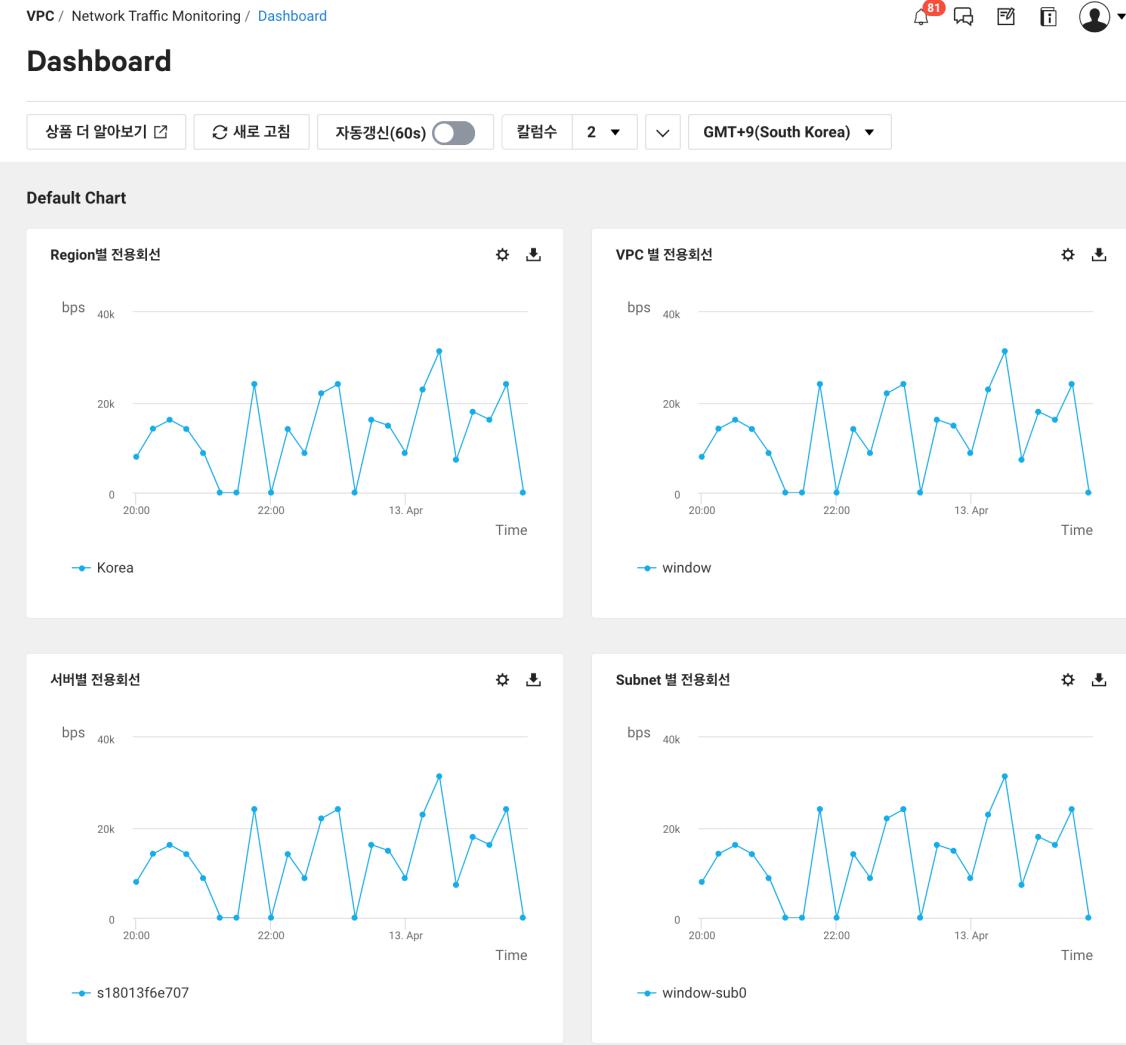


### 핵심 용어

- OpenFlow
  - OpenFlow Switch, OpenFlow Controller로 구성
  - Flow 정보를 제어해서 패킷의 전달 경로 결정
  - 내부에 FlowTable 존재
- Traffic
  - 네트워크를 통해 움직이는 데이터의 양
- Flow
  - 특정시간동안 네트워크상의 지정된 관찰지점을 지나가는 패킷의 집합
- Packet
  - 통신망을 통해 전송하기 쉽게 자른 데이터 전송단위

# Network Traffic Monitoring

## Dashboard에서 다양한 지표 차트로 제공



## 제공 되는 지표

- 리전별 Internet Outbound
- 리전별 전용회선 Outbound
- 서버별 Internet
- 서버별 전용회선
- 서버 그룹별 Internet
- 서버 그룹별 전용회선
- 국가별 Internet

# Network Traffic Monitoring Hands-on

# NKS 환경에서 Monitoring

## Naver Kubernetes Service Monitoring

네이버 쿠버네티스 서비스에서는 2가지 방식으로 모니터링 지원



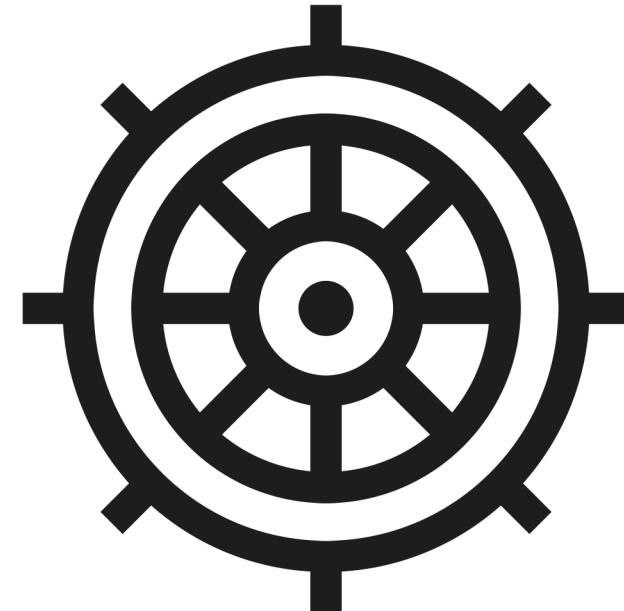
### Cloud Insight

- Naver Cloud Platform의 핵심 모니터링 서비스  
인 Cloud Insight와 연동



### Grafana

- 오픈소스인 Grafana 서비스를 통해 모니터링  
시각화 가능



# NKS 환경에서 Monitoring

## Metrics Export 설정

YAML	Copy
<pre>apiVersion: v1 kind: ConfigMap metadata:   name: nks-metrics-exporter data:   baseurl: https://cw.apigw.ntruss.com   basepath: /cw_collector/real/data   dmn_cd: PUB   prodkey: "526115048926613504"   nrn: nrn:PUB:VPCKubernetesService:KR:NUM:Cluster/cluster-id-fix-me   accesskey: 2M1ACCESSKEYFIXME63rX7   secretkey: z3feSECRETKEYFIXMEhR6mTyl   ignore_namespaces: kube-system</pre>	

Metrics Exporter에서 사용할 설정값을 Configmap으로 생성

### nrn:

- Resource Manager 상품에서 Kubernetes Service(VPC) 클러스터의 nrn 입력

### accesskey:

- 인증키에서 얻은 Access Key 입력

### secretkey:

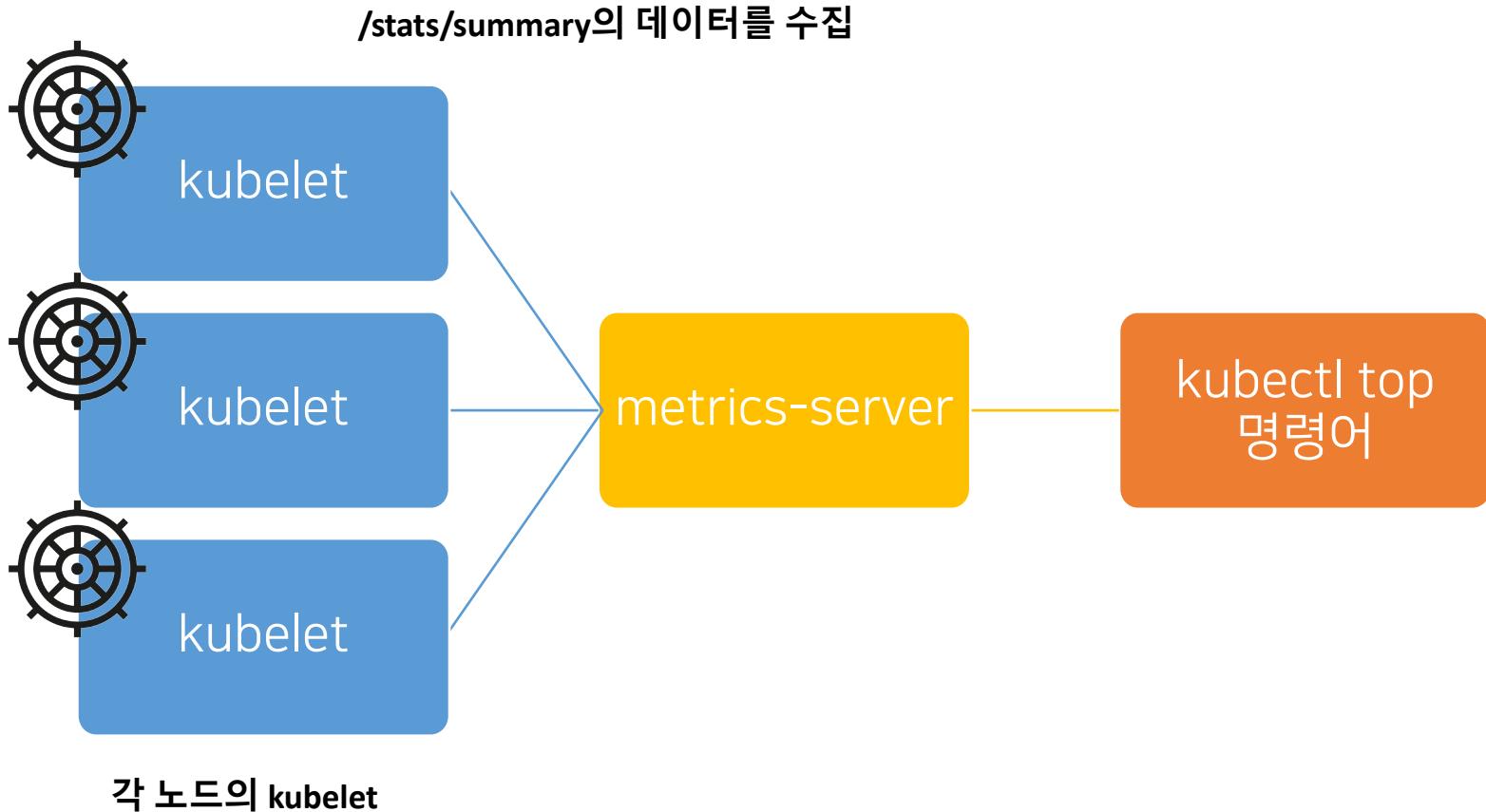
- 인증키에서 얻은 Secret Key 입력

### ignore\_namespaces:

- 특정 네임스페이스 내에 pod 자원 사용량을 집계하지 않으면, 해당 필드에 네임스페이스 이름 입력

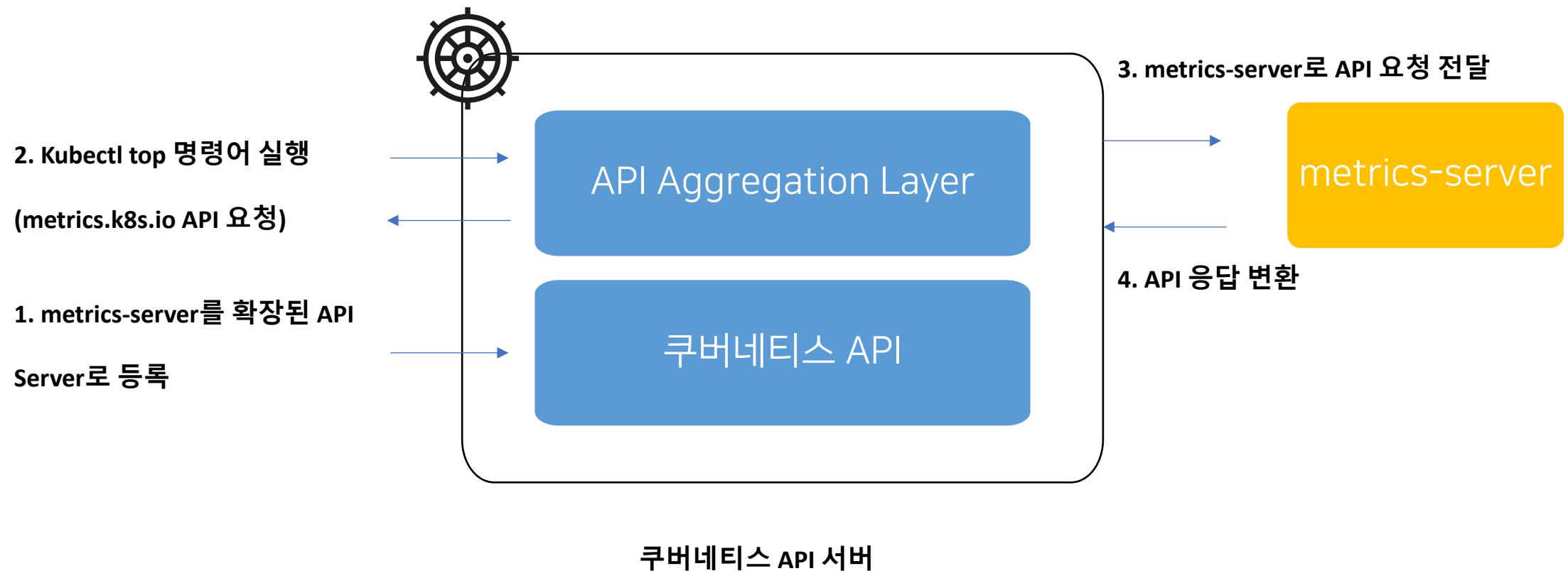
# NKS 환경에서 Monitoring

## metrics-server의 메트릭 수집 구조



# NKS 환경에서 Monitoring

## metrics-server의 동작 원리



# NKS 환경에서 Monitoring

## Cloud Insight를 통한 NKS 모니터링의 한계

VPC / Cloud Insight(Monitoring) / Dashboard

Widget 생성 데이터 소스에 따라 다양한 타입의 위젯을 생성할 수 있습니다.

81

Widget 생성

기본 정보 설정 데이터 설정 최종 확인

데이터 설정 Target을 선택하면 해당 Target의 Metric을 선택할 수 있습니다. 2개 이상의 Target을 설정하려면 '감시 대상 그룹'을 활용하여 선택할 수 있습니다.

Product Type: Kubernetes Service(VPC)

Target: 그룹 보유 리소스 전체  
nks-test

Metric: 템플릿 전체 메트릭

ALL

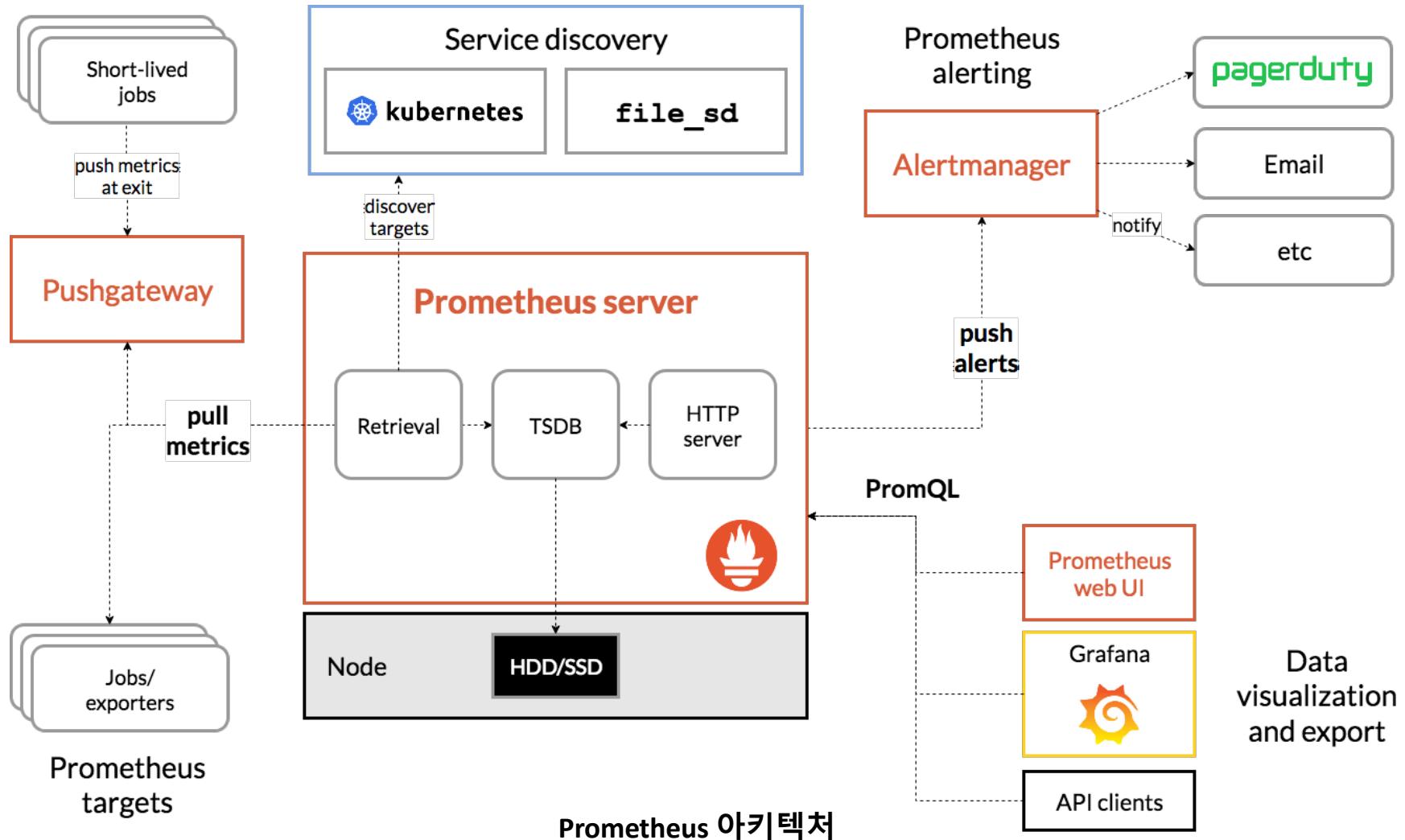
Metric ID: 검색어를 입력하세요. Q

Metric	Description
real_cpu	resource usage
real_mem	resource usage
rqst_cpu	resource request
rqst_mem	resource request

NKS 지원 Metric

# NKS 환경에서 Monitoring

## 오픈소스를 사용한 NKS 모니터링



# NKS Monitoring Hands-on

# Infrastructure as Code (IaC)

# IaC (Infrastructure as Code)란

## 코드로 관리하는 인프라, 프로그래밍 하듯 인프라를 관리

```
$vpcConfigs = Get-Content -Path 'config.json' -Raw | ConvertFrom-Json  
$grafanaVpcConfig = $vpcConfigs.subnets.grafana  
  
##Creating VPC  
$labVpc = ncloud vpc createVpc --regionCode KR --vpcName $vpcConfigs.vpcName --ipv4CidrBlock  
$vpcConfigs.vpcNetwork  
$labVpc = ($labVpc | ConvertFrom-Json).createVpcResponse.vpcList  
$labVpc | Export-Csv "vpcInfo.csv" -NoTypeInformation -Encoding oem  
  
##GET ACL  
$labVpcACL = ncloud vpc getNetworkAclList --vpcNo $labVpc[0].vpcNo  
$labVpcACL = ($labVpcACL | ConvertFrom-Json).getNetworkAclListResponse.networkAclList  
  
##Creating Subnet  
$labVpcSubnet = ncloud vpc createSubnet `  
    --regionCode KR `  
    --zoneCode KR-1 `  
    --vpcNo $labVpc[0].vpcNo `  
    --subnetName $($labVpc[0].vpcName + $grafanaVpcConfig.postfix) `  
    --subnet $grafanaVpcConfig.subnet `  
    --networkAclNo $($labVpcACL[0].networkAclNo) `  
    --subnetTypeCode $grafanaVpcConfig.typeCode `  
    --usageTypeCode $grafanaVpcConfig.usageCode  
  
$labVpcSubnet = ($labVpcSubnet | ConvertFrom-Json).createSubnetResponse.subnetList  
$labVpcSubnet | Export-Csv "subnetInfo.csv" -NoTypeInformation -Encoding oem
```

```
GET {API_URL}/createServerInstances  
?regionCode=KR  
&serverImageProductCode=SW/SVR.OS.LNX64.CNTOS.0703.B050  
&vpcNo=***04  
&subnetNo=***43  
&serverProductCode=SVR.VSR.STAND.C002.M004.NETSSD.B050.G001  
&feeSystemTypeCode=MTRAT  
&serverCreateCount=1  
&serverName=test-***  
&networkInterfaceList.1.networkInterfaceOrder=0  
&networkInterfaceList.1.accessControlGroupNoList.1=***63  
&placementGroupNo=***61  
&isProtectServerTermination=false  
&initScriptNo=***44  
&loginKeyName=test-***  
&associateWithPublicIp=true
```

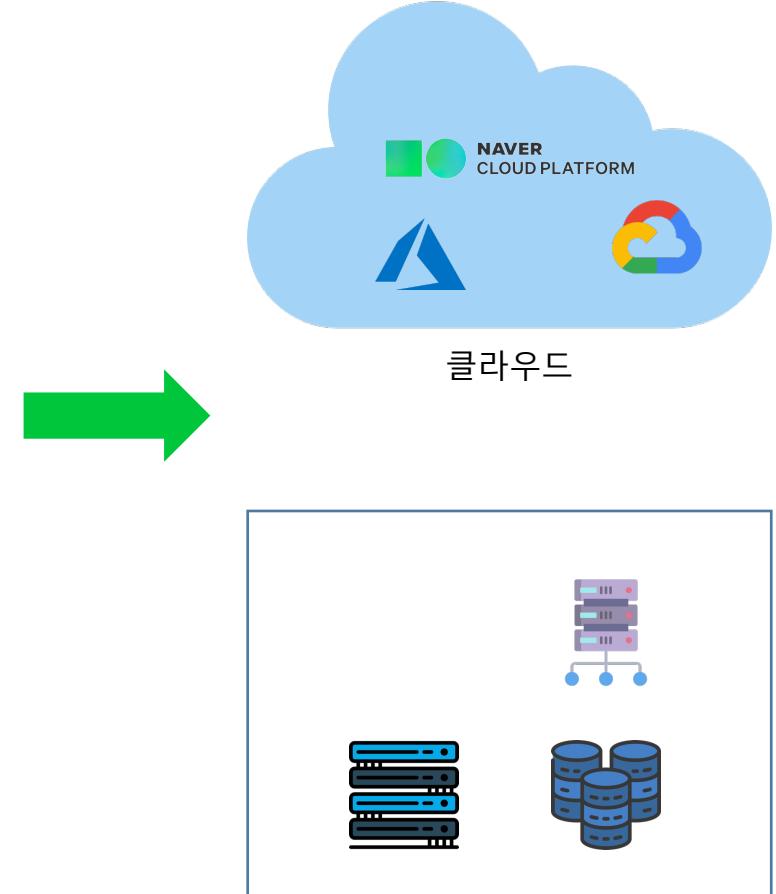
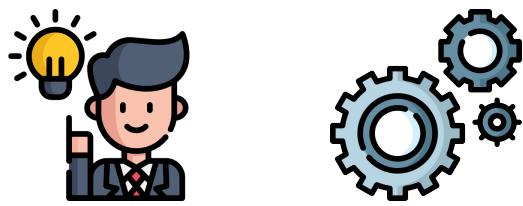


Ncloud CLI



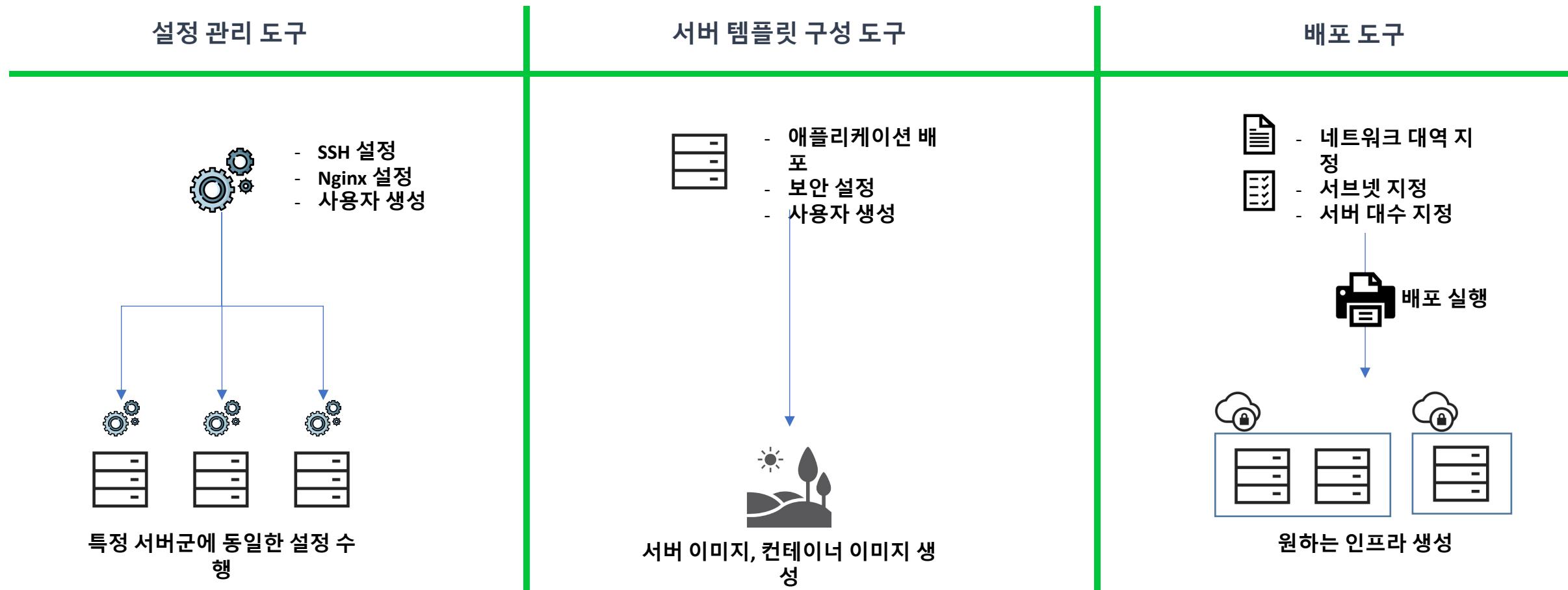
Ncloud API

# IaC 가 필요한 이유



# IaC 도구의 종류

IaC 도구는 세 가지로 분류 가능



# IaC 도구의 종류

다양한 IaC 관리 도구가 존재

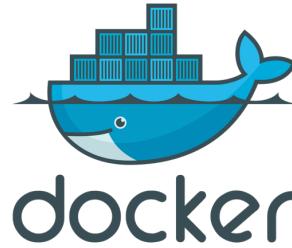
설정 관리 도구



ANSIBLE



서버 템플릿 구성 도구

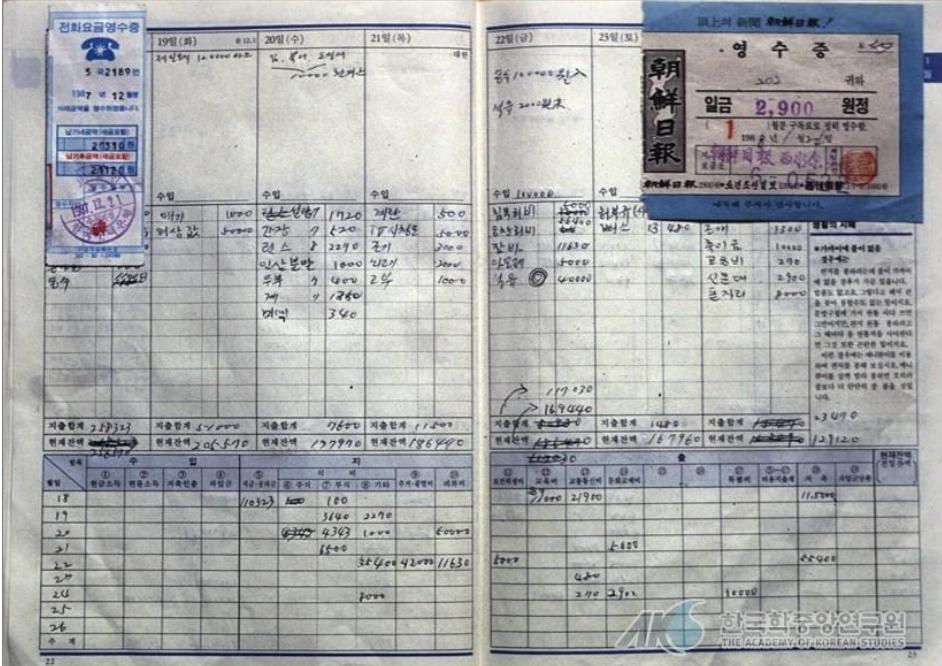


배포 도구



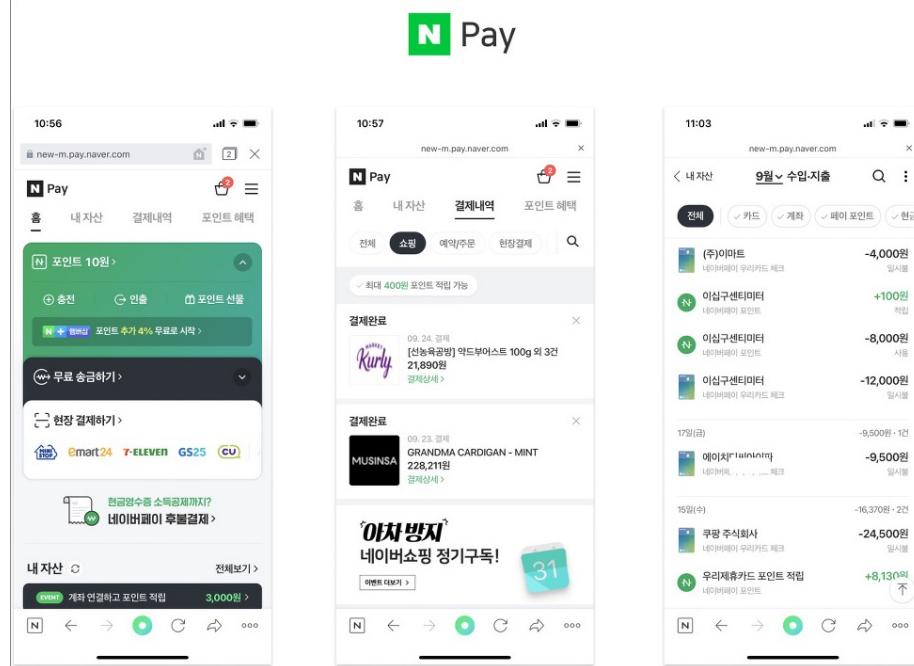
# IaC 도구를 써야하는 이유

현재 상태를 기록하고, 지난 상태를 확인할 수 있어야함



1980년대의 가계부

한국학중앙연구원 제공



현대의 가계부

네이버페이 제공

# Ncloud CLI

# 네이버클라우드 플랫폼 CLI (Command Line Interface)

## 네이버클라우드 플랫폼의 리소스를 명령어로 관리하기 위한 통합 도구

- bash, CMD, PowerShell등 환경에서 명령어 실행
- 숙련된 사용자에겐 콘솔보다 빠른 작업 시간 보장
- 빠른 작업을 위한 스크립트 개발에 사용 가능
  - 특정 환경을 빠르게 배포하는 스크립트
  - 동시에 VM 을 종료하는 등 특정 상황을 만드는 스크립트
  - 대량의 서버를 Load Balancer에 배치하기 위한 스크립트

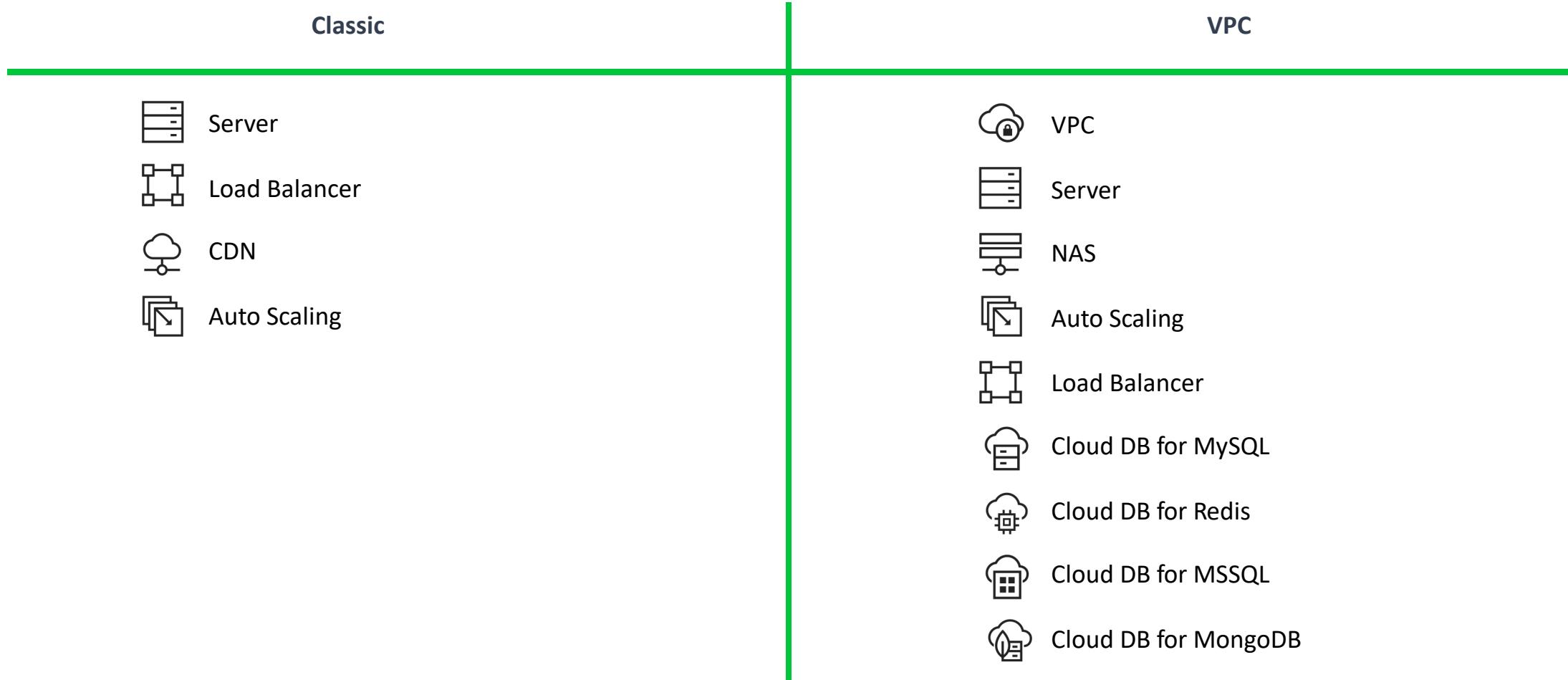
```
PS C:\Users\USER> ncloud help
=====설명=====
Ncloud Command Line Interface(CLI)는 Ncloud 서비스를 관리하기 위한 통합툴입니다.

=====개요=====
- Ncloud Command Line 인터페이스는 아래와 같은 방식으로 명령을 수행합니다.
  ncloud [options] <command> <subcommand> [parameters]
- 구체적인 명령어에 대해서 더 알고 싶으시면 아래와 같은 방식으로 명령어를 입력하세요
  ncloud help
  ncloud <command> help
  ncloud <command> <subcommand> help
  help 명령어에 관련된 토큰들은 개요에서 매개변수와 사용예를 보아줍니다.

=====주요명령어=====
- server
- loadbalancer
- cdn
- autoscaling
- vserver
- vpc
- vnas
- vloadbalancer
- vautoscaling
- vmysql
- vmssql
- vredis
- vmongod
=====기타 명령어=====
- configure
=====옵션=====
- 디버그옵션
  옵션 : --debug (Boolean)
  --debug 전언시 debug모드로 로그가 출력됩니다.
  사용예 : ncloud --debug <command> <subcommand>
```

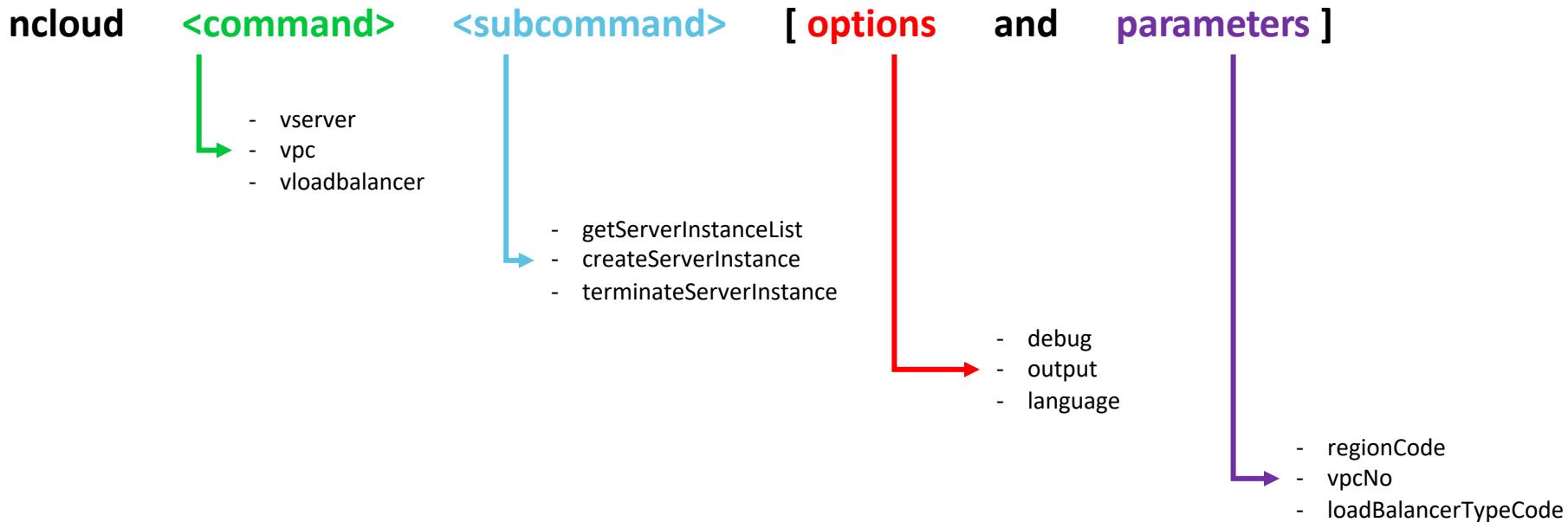
# 네이버클라우드 플랫폼 CLI (Command Line Interface)

네이버클라우드 플랫폼의 리소스를 명령어로 관리하기 위한 통합 도구



# 네이버클라우드 플랫폼 CLI 명령어 구조

```
ncloud vserver getServerInstanceList --output json --regionCode KR
```



# 네이버 클라우드 플랫폼 CLI 설명서

The screenshot shows the '개요' (Overview) page of the Naver Cloud Platform CLI Guide. The left sidebar contains a navigation menu with sections like 'HOME', 'CLI 가이드' (selected), 'Ncloud CLI 명령어', and various service-specific sections. The main content area has a title '개요' and a section '소개' (Introduction) which describes the CLI as a tool for managing resources across Linux and Windows environments. It lists three types of CLIs: Ncloud CLI, Object Storage CLI, and Archive Storage CLI. Below this is a '시작하기' (Getting Started) section for the Ncloud CLI, which provides instructions for download and installation. A note at the bottom states that the CLI files are available for both Linux and Windows environments.

NAVER CLOUD PLATFORM | CLI 가이드

필터

검색

개요

인쇄 | 공유 | PDF

## 소개

네이버 클라우드 플랫폼 CLI(Command Line Interface, 명령줄 인터페이스)는 네이버 클라우드 플랫폼의 리소스를 관리하기 위한 통합 도구입니다. Linux와 Windows에서 개발 도구를 설치하고 구성하면 다양한 네이버 클라우드 플랫폼 서비스들을 명령줄에서 제어하고 스크립트를 이용해 자동화할 수 있습니다.

네이버 클라우드 플랫폼에서는 아래와 같이 3가지 방식의 CLI를 제공하고 있습니다.

- Ncloud CLI
- Object Storage CLI
- Archive Storage CLI

네이버 클라우드 플랫폼에서 제공하는 CLI를 이용하여 각 상품들의 기능을 확인하고 리소스를 관리할 수 있는 스크립트를 개발할 수 있습니다.

본 CLI 참조서에서는 CLI의 구조, 형태 및 실행 방법과 상품별로 제공하는 CLI의 요청 파라미터, 응답 데이터 타입 및 예시 등을 제공합니다.

## 시작하기

### Ncloud CLI

네이버 클라우드 플랫폼에서 제공하는 **Ncloud CLI**는 Server, Load Balancer, CDN, Auto Scaling, VPC, Server(VPC), NAS(VPC), Auto Scaling(VPC), Load Balancer(VPC), Cloud DB for MySQL(VPC), Cloud DB for REDIS(VPC), Cloud DB for MSSQL(VPC) 서비스의 CLI 환경을 제공합니다.

Ncloud CLI를 사용하려면 아래 설치 프로그램 파일을 다운로드받고, 로컬에서 명령줄을 실행하세요. 설치 및 실행 방법에 대한 자세한 사항은 [Ncloud CLI 사용 가이드](#)를 참조하세요.

[Linux/Windows용 CLI 파일 다운로드](#)

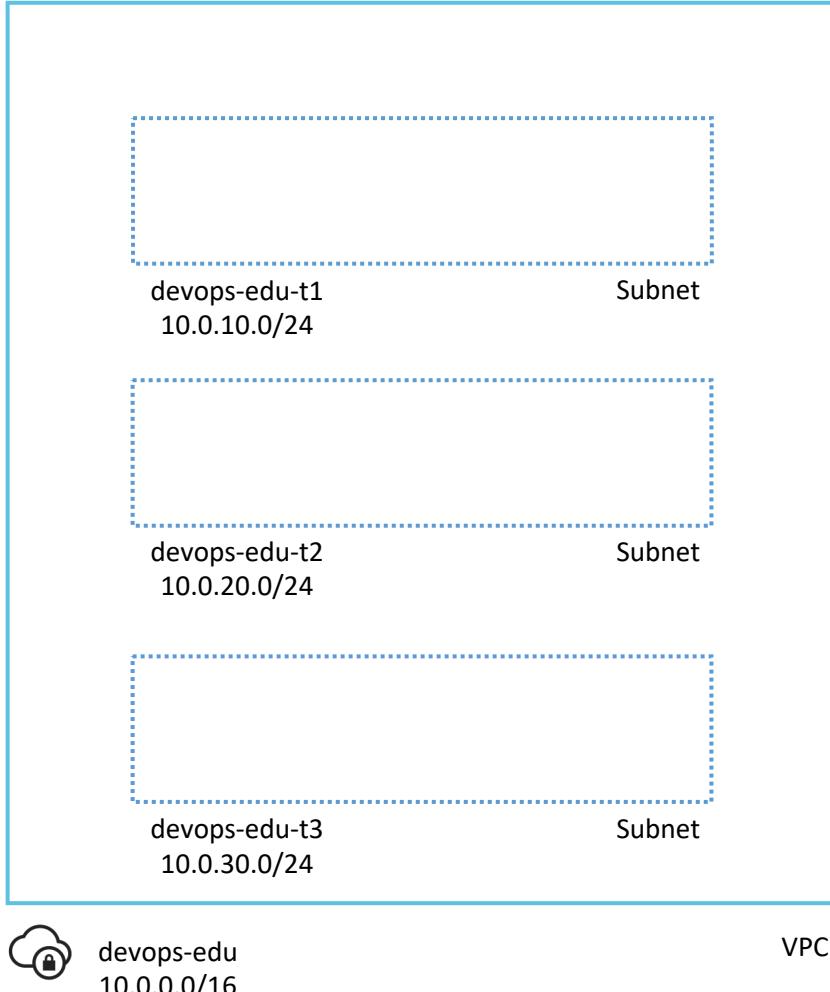
- 해당 CLI 다운로드 파일은 Linux 개발 환경과 Windows 환경에 따라 설치할 수 있도록 `cli_linux` 폴더와 `cli_window` 폴더로 구성되어 있습니다.

<https://cli.ncloud-docs.com/docs/guide>

# Ncloud CLI Hands-on

# 3Tier 네트워크 구성

## 네트워크 아키텍처



1. devops-edu VPC 생성
2. 각 서브넷 생성
3. 서브넷 목록 확인

# Ncloud API

# API는 무엇인가?

1 ~ N 까지의 수를 더하는 프로그램을 만들어 주세요

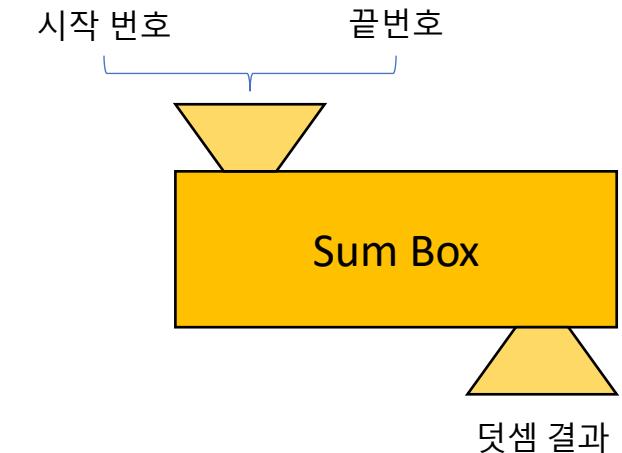
철수

```
$start = 1  
$end = 4  
$sum = 0  
  
for ($i = $start; $i -le $end; $i++) {  
  
    $sum += $i  
  
}  
  
$sum
```

영희

```
$start = 1  
$end = 4  
$sum = 0  
  
$sum = ($end * ($end + 1)) / 2  
  
$sum
```

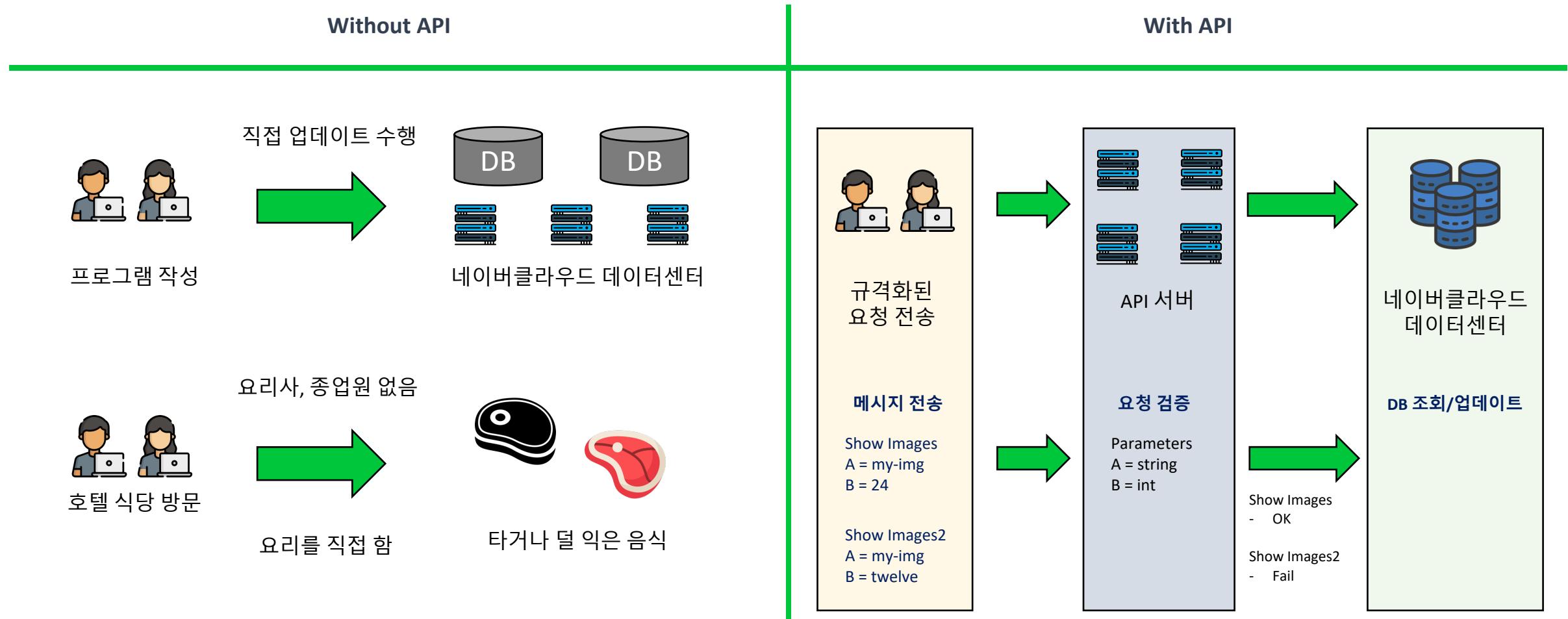
크로바



<http://asdf.clova.xyz/sum/1/10>

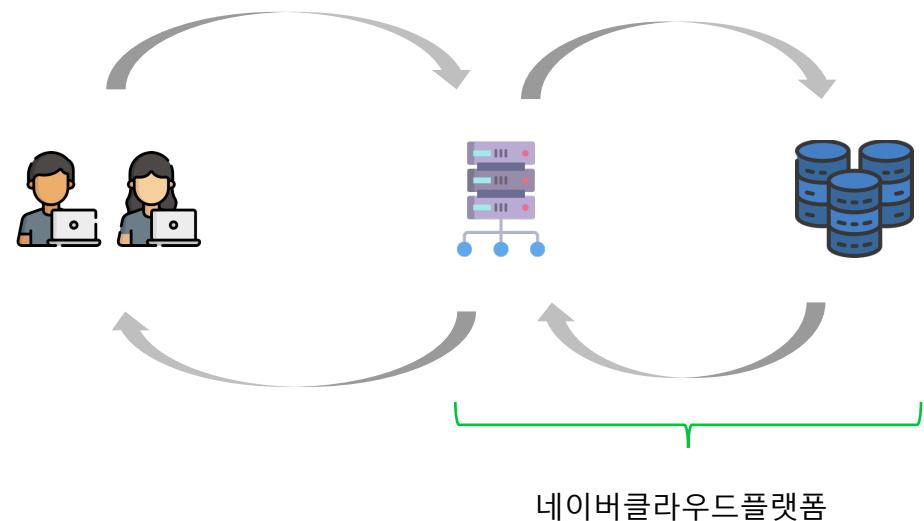
# API는 무엇인가?

사용자는 정해진 규격으로 요청을 전송하고, 서버는 요청을 안전하게 처리할 수 있음



# 네이버 클라우드 플랫폼 REST API

## 서버 생성 API 샘플



### HTTP

```
GET {API_URL}/createServerInstances  
?regionCode=KR  
&serverImageProductCode=SW.VSVR.OS.LNX64.CNTOS.0703.B050  
&vpcNo=****04  
&subnetNo=****43  
&serverProductCode=SVR.VSVR.STAND.C002.M004.NET.SSD.B050.G001  
&feeSystemTypeCode=MTRAT  
&serverCreateCount=1  
&serverName=test-***  
&networkInterfaceList.1.networkInterfaceOrder=0  
&networkInterfaceList.1.accessControlGroupNoList.1=****63  
&placementGroupNo=****61  
&isProtectServerTermination=false  
&initScriptNo=****44  
&loginKeyName=test-***  
&associateWithPublicIp=true
```

# 네이버 클라우드 플랫폼 REST API

## REST API 내용 확인하기

HTTP
GET or POST <a href="https://ncloud.apigw.ntruss.com/vserver/v2/">https://ncloud.apigw.ntruss.com/vserver/v2/</a>

HTTP
GET {API_URL}/createServerInstances ?regionCode=KR &serverImageProductCode=SW.VSVR.OS.LNX64.CNTOS.0703.B050 &vpcNo=***04 &subnetNo=***43 &serverProductCode=SVR.VSVR.STAND.C002.M004.NET.SSD.B050.G001 &feeSystemTypeCode=MTRAT &serverCreateCount=1 &serverName=test-*** &networkInterfaceList.1.networkInterfaceOrder=0 &networkInterfaceList.1.accessControlGroupNoList.1=***63 &placementGroupNo=***61 &isProtectServerTermination=false &initScriptNo=***44 &loginKeyName=test-*** &associateWithPublicIp=true

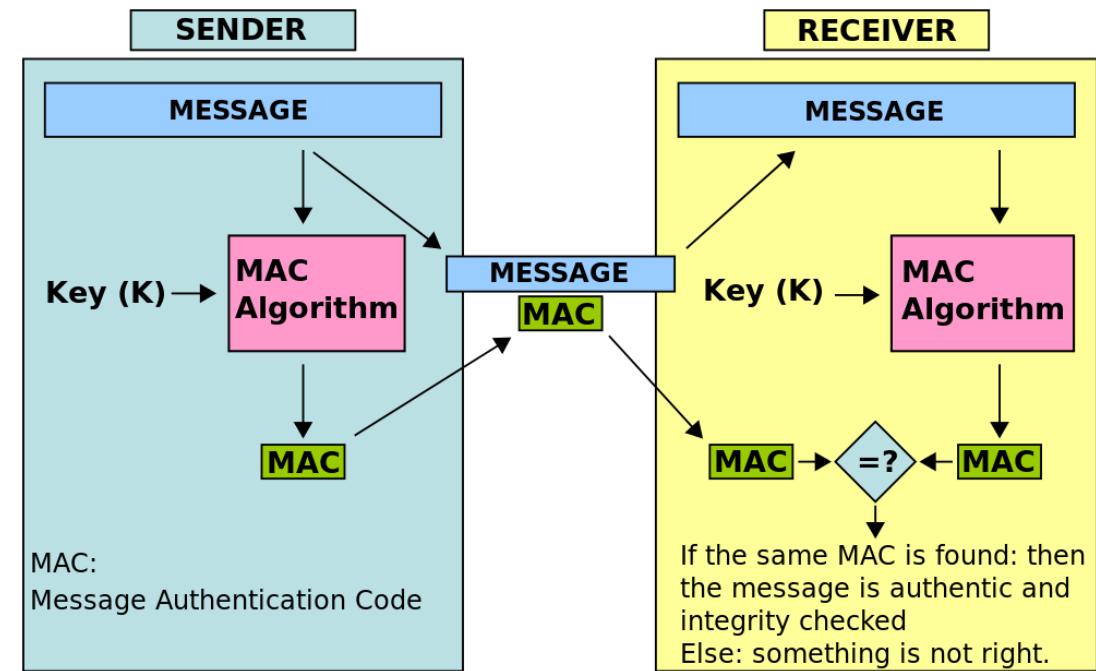
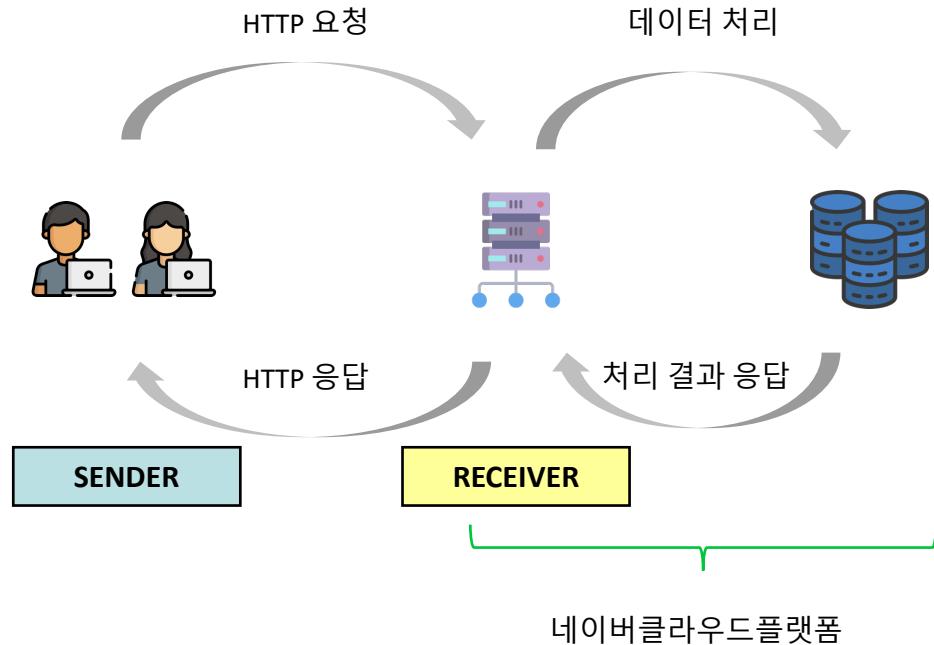
메시지	내용
Method	GET / POST / PUT / DELETE ...
URL	<a href="https://ncloud.apigw.ntruss.com">https://ncloud.apigw.ntruss.com</a>
Headers	Platform Headers
Resources	vserver/v2
Act	createServerInstances?regionCode=KR.....
Body	Data

# 네이버클라우드플랫폼 REST API 헤더

헤더	설명
<b>x-ncp-apigw-timestamp</b>	<ul style="list-style-type: none"><li>- Unix Time / POSIX Time / Epoch Time</li><li>- API Gateway 서버와 5분 이상 시간차가 나는 경우 유효하지 않은 요청으로 간주</li></ul>
<b>x-ncp-iam-access-key</b>	<ul style="list-style-type: none"><li>- 네이버클라우드 플랫폼 주 계정 또는 Sub Account에서 발급받은 Access Key ID</li></ul>
<b>x-ncp-apigw-signature-v2</b>	<ul style="list-style-type: none"><li>- HTTP 요청 메시지를 Access Key ID와 매핑된 Secret Key로 암호화한 서명</li><li>- HMACSHA256 알고리즘 사용</li></ul>

# 사용자 요청 보호 HMAC (Hash Message Authentication Code)

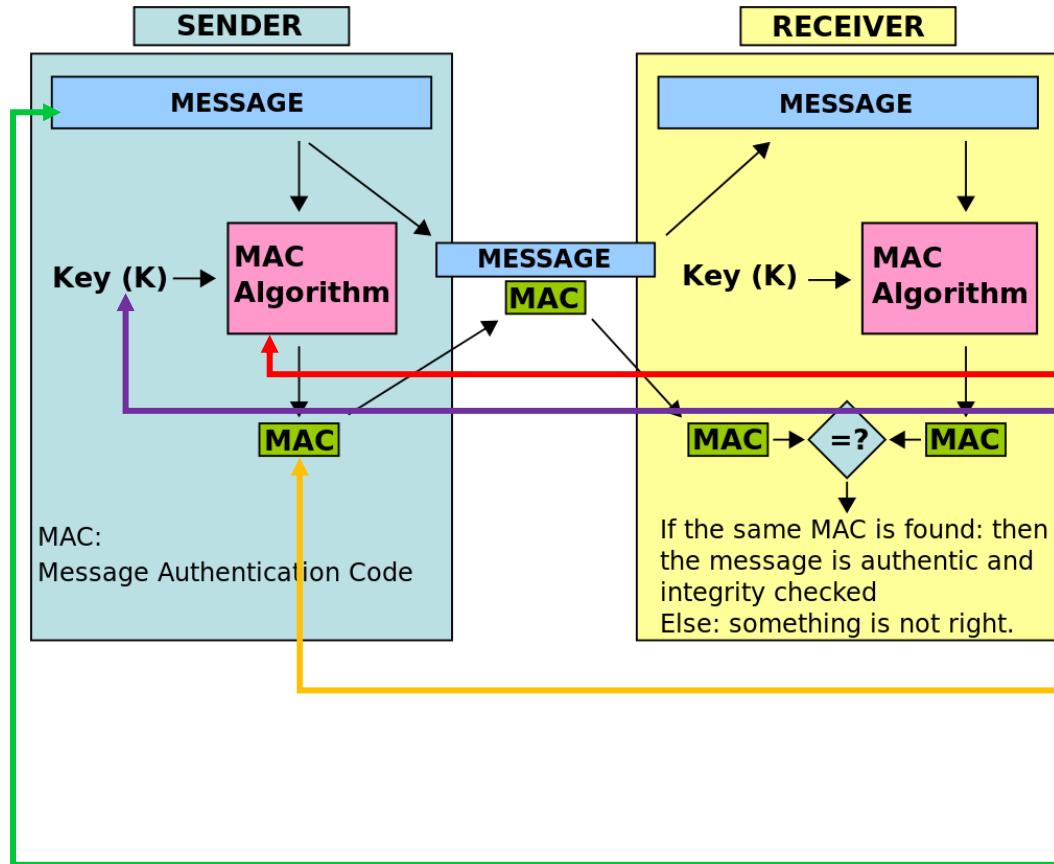
메시지 변조 여부를 확인하기 위해, Sender 요청을 Receiver가 검증



출처 : [https://ko.wikipedia.org/wiki/메시지\\_인증\\_코드](https://ko.wikipedia.org/wiki/메시지_인증_코드)

# 사용자 요청 보호 HMAC (Hash Message Authentication Code)

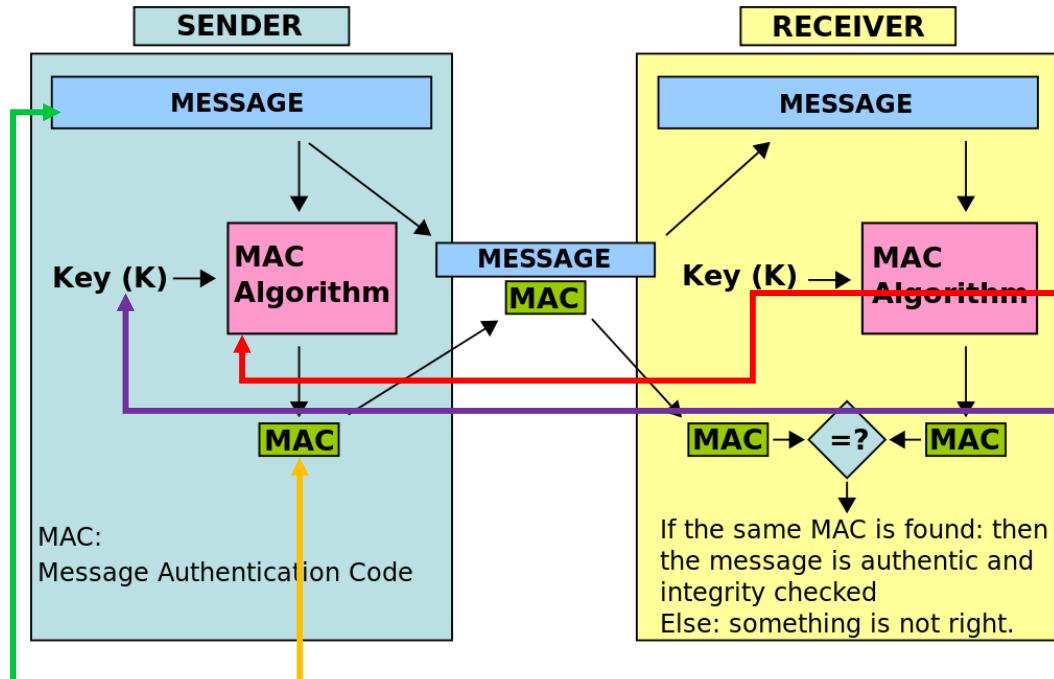
메시지 변조 여부를 확인하기 위해, Sender 요청을 Receiver가 검증



```
1 reference
1 function createSignedSignature(
2     [string] $method,          # GET, PUT, POST, DELETE
3     [string] $url,             # path+query
4     [string] $accessKey,       # access key id
5     [string] $secretKey        # access key value
6 )
7
8     $verb = $method.ToUpperInvariant()
9     $timeStamp = [int](Get-Date -UFormat %s) * 1000
10
11    $stringToSign = $verb + " " + $url + "`n" +
12        [string]$timeStamp + "`n" +
13        $accessKey
14
15    $hmac = New-Object System.Security.Cryptography.HMACSHA256
16    $hmac.Key = [Text.Encoding]::UTF8.GetBytes($secretKey)
17    $signature = [Convert]::ToBase64String($hmac.ComputeHash([Text.Encoding]::UTF8.GetBytes($stringToSign)))
18
19    return @{
20        "x-ncp-apigw-timestamp" = $timeStamp;
21        "x-ncp-iam-access-key" = $accessKey;
22        "x-ncp-apigw-signature-v2" = $signature;
23    }
24
25
26    $cred = Get-Content -Raw -Path "./credentials/key.json" | ConvertFrom-Json
27    $signedHeader = createSignedSignature `
28        -method "GET" ` 
29        -url "/vserver/v2/getRegionList?responseFormatType=json" ` 
30        -accessKey $cred.accessId ` 
31        -secretKey $cred.secret
32
33    Invoke-WebRequest -Method Get ` 
34        -Uri "https://ncloud.apigw.ntruss.com/vserver/v2/getRegionList?responseFormatType=json" ` 
35        -Headers $signedHeader
36
37
```

# 사용자 요청 보호 HMAC (Hash Message Authentication Code)

메시지 변조 여부를 확인하기 위해, Sender 요청을 Receiver가 검증



MAC:  
Message Authentication Code

```
1 reference
1 function createSignedSignature(
2     [string] $method,          # GET, PUT, POST, DELETE
3     [string] $url,             # path+query
4     [string] $accessKey,       # access key id
5     [string] $secretKey        # access key value
6 )
7
8     $verb = $method.ToUpperInvariant()
9     $timeStamp = [int](Get-Date -UFormat %s) * 1000
10
11    $stringToSign = $verb + " " + $url + "`n" +
12        [string]$timeStamp + "`n" +
13        $accessKey
14
15    $hmac = New-Object System.Security.Cryptography.HMACSHA256
16    $hmac.Key = [Text.Encoding]::UTF8.GetBytes($secretKey)
17    $signature = [Convert]::ToBase64String($hmac.ComputeHash([Text.Encoding]::UTF8.GetBytes($stringToSign)))
18
19    return @{
20        "x-ncp-apigw-timestamp" = $timeStamp;
21        "x-ncp-iam-access-key" = $accessKey;
22        "x-ncp-apigw-signature-v2" = $signature;
23    }
24
25
26    $cred = Get-Content -Raw -Path "./credentials/key.json" | ConvertFrom-Json
27    $signedHeader = createSignedSignature `
28        -method "GET" ` 
29        -url "/vserver/v2/getRegionList?responseFormatType=json" ` 
30        -accessKey $cred.accessId ` 
31        -secretKey $cred.secret
32
33    Invoke-WebRequest -Method Get ` 
34        -Uri "https://ncloud.apigw.ntruss.com/vserver/v2/getRegionList?responseFormatType=json" ` 
35        -Headers $signedHeader
36
37
```

# 네이버 클라우드 플랫폼 API 설명서

The screenshot shows the NAVER Cloud Platform API documentation for the Object Storage API. The left sidebar contains a navigation tree with categories like Object Storage API, Archive Storage API, NAVER 연동 API, API 데이터 탐색, API 데이터 탐색 (VPC), Compute, Server, Server (VPC), and Server. The main content area is titled 'NAVER Cloud Platform API' and has a sub-section titled '개요' (Overview). It explains that the API provides a way to use various services through a programmatic interface and that it supports RESTful API calls with XML and JSON formats. It also mentions that the API can be used with VPC and provides examples of how to use it with HTTP GET/POST methods. A '참고' (Reference) section at the bottom provides instructions for generating API keys.

NAVER Cloud Platform API

개요

네이버 클라우드 플랫폼에서 제공하는 인프라/솔루션 상품을 이용할 수 있도록 지원하는 응용 프로그램 인터페이스(Application Programming Interface, API) 제공하고 있습니다. 본 페이지에서는 NAVER Cloud Platform API 대한 간략한 설명 및 API 호출하는 방법을 제공합니다.

API는 RESTful API 방식으로 제공되며, XML과 JSON 형식으로 응답합니다. 액션에 따라 파라미터 값을 입력하고 등록, 수정, 삭제, 조회할 수 있으며, 서비스 및 운영 도구 자동화에 활용할 수 있습니다. HTTP 방식의 GET/POST 메서드 호출을 통해서 사용됩니다.

만일 호출이 잘못되었을 경우는 오류 코드와 메시지를 리턴합니다.

NAVER Cloud Platform API 호출 절차

NAVER Cloud Platform API 호출은 다음과 같은 단계로 진행되어야 합니다.

인증키 생성하기

NAVER Cloud Platform 계정이 생성되면 기본적으로 NAVER Cloud Platform API 인증키가 함께 발급됩니다. 발급된 인증기는 [네이버 클라우드 플랫폼 홈페이지](#)의 [마이페이지] > [계정관리] > [인증키관리]에서 확인할 수 있습니다. 인증기는 계정 생성 시 자동으로 발급되는 것 외에 사용자가 하나 더 생성할 수 있어서 두 개까지 발급받을 수 있습니다.

참고

인증키를 '사용 중지'로 설정하거나 삭제하면 유효하지 않은 키로 인식됩니다.

API 인증기는 [Access Key](#) 와 Secret Key 한 쌍으로 구성되어 있습니다. 한 쌍의 API 인증기는 API를 인증할 때 파라미터로 직접 전달됩니다.

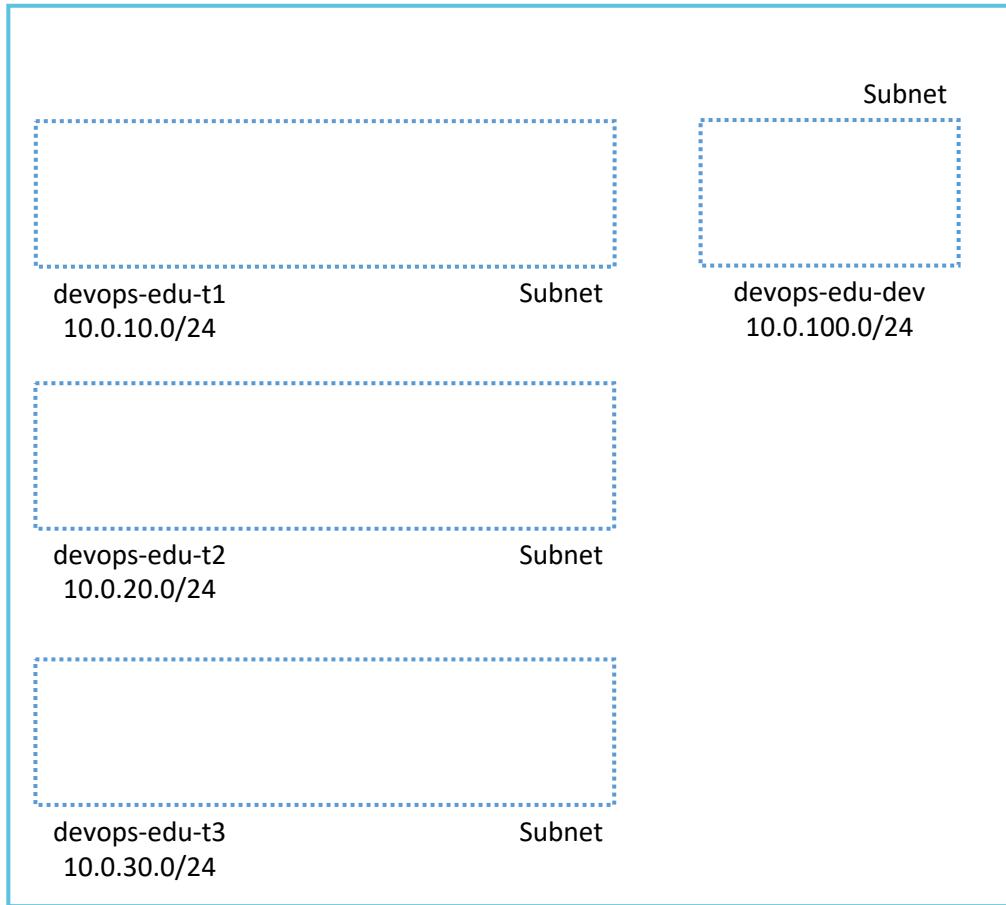
1. [네이버 클라우드 플랫폼 홈페이지](#)에서 로그인을 합니다.
2. [마이페이지] > [계정관리] > [인증키관리] 메뉴로 접속하시면 "신규 API 인증키 생성" 버튼을 클릭합니다.
  - 기존에 생성하신 인증키가 있으실 경우에는 해당 인증키를 사용하실 수 있습니다.

<https://api.ncloud-docs.com/docs/common-ncpapi>

# Ncloud API Hands-on

# VPC 정리하기

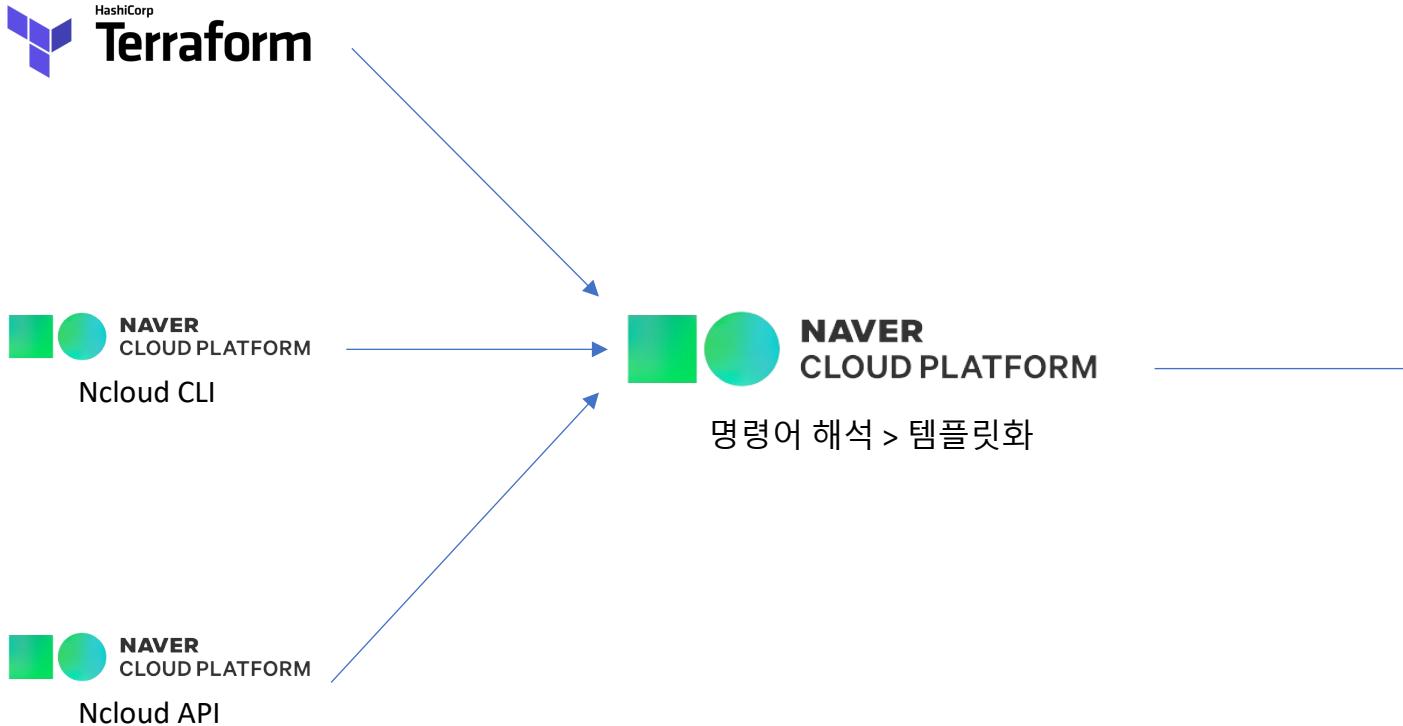
## 네트워크 아키텍처



1. 디렉토리 구조 만들기
2. 인증 함수 만들기
3. VPC 조회하기
4. devops-edu-dev 서브넷 생성
5. devops-edu VPC의 서브넷 조회
6. devops-edu VPC의 모든 서브넷 삭제
7. devops-edu VPC 삭제

# Ncloud Terraform

# 명령어 해석



```
[{"serverInstanceNo": "9816926", "serverName": "ansible-target", "serverDescription": "", "cpuCount": 2, "memorySize": 4294967296, "platformType": {"code": "LNX64", "codeName": "Linux 64 Bit"}, "loginKeyName": "pcw-ansible", "publicIpInstanceNo": "9816992", "publicIp": "223.130.163.20", "serverInstanceState": {"code": "RUN", "codeName": "Server run state"}, "serverInstanceOperation": {"code": "NULL", "codeName": "Server NULL OP"}, "serverInstanceStateName": "running", "createdDate": "2022-01-18T15:06:13+0900", "uptime": "2022-01-18T15:28:20+0900", "serverImageProductCode": "SW.VSVR.OS.LNX64.CNTOS.0708.B050", "serverProductCode": "SVR.VSVR.HICPU.C002.M004.NET.HDD.B050.G002", "isProtectServerTermination": false, "zoneCode": "KR-1", "regionCode": "KR", "vpcNo": "13624", "subnetNo": "25588", "networkInterfaceNoList": ["506072"], "initScriptNo": "", "serverInstanceType": {"code": "HICPU", "codeName": "High CPU"}, "baseBlockStorageDiskType": {"code": "NET", "codeName": "Network Storage"}, "baseBlockStorageDiskDetailType": {"code": "HDD", "codeName": "HDD"}, "placementGroupNo": "", "placementGroupName": "", "memberServerImageInstanceNo": "9816889"}],
```

# 네이버 클라우드 VPC 서버 템플릿

```
{  
    "serverInstanceNo": "9816926",  
    "serverName": "ansible-target", ←  
    "serverDescription": "",  
    "cpuCount": 2,  
    "memorySize": 4294967296, ←  
    "platformType": {  
        "code": "LNX64",  
        "codeName": "Linux 64 Bit"  
    },  
    "loginKeyName": "pcw-ansible", ←  
    "publicIPInstanceNo": "9816992",  
    "publicIP": "223.130.163.20",  
    "serverInstanceState": {  
        "code": "RUN",  
        "codeName": "Server run state"  
    },  
    "serverInstanceOperation": {  
        "code": "NULL",  
        "codeName": "Server NULL OP"  
    },  
    "serverInstanceStateName": "running",  
    "createDate": "2022-01-18T15:06:13+0900",  
    "uptime": "2022-01-18T15:28:20+0900",  
    "serverImageProductCode": "SW.VSVR.OS.LNX64.CNTOS.0708.B050", ←  
    "serverProductCode": "SVR.VSVR.HICPU.C002.M004.NET.HDD.B050.G002",  
    "isProtectServerTermination": false,  
    "zoneCode": "KR-1",  
    "regionCode": "KR",  
    "vpcNo": "13624",  
    "subnetNo": "25588",  
    "networkInterfaceNoList": [  
        "506072"  
    ],  
    "initScriptNo": "",  
    "serverInstanceType": {  
        "code": "HICPU",  
        "codeName": "High CPU"  
    },  
    "baseBlockStorageDiskType": {  
        "code": "NET",  
        "codeName": "Network Storage"  
    },  
    "baseBlockStorageDiskDetailType": {  
        "code": "HDD",  
        "codeName": "HDD"  
    },  
    "placementGroupNo": "",  
    "placementGroupName": "",  
    "memberServerImageInstanceNo": "9816889"  
},
```

서버 이름  
서버 자원 사양  
로그인 키 정보  
이미지 정보  
...

# Template

규격화가 되어있으면 사용자가 쉽게 선택할 수 있고, 공급자도 간편하게 공급 가능

상품명을 검색하세요.  검색

메인보드 상품개수: 1768개

제조사  ASUS  MSI  ASRock  GIGABYTE  ECS

제품 분류  인텔 CPU용  AMD CPU용  임베디드  주변기기

CPU 소켓  인텔(소켓1700)  인텔(소켓1200)  인텔(소켓1151v2)  인텔(소켓2066)  AMD(소켓AM4)

세부 칩셋  인텔 Z690  인텔 H670  인텔 B660  인텔 H610  AMD X570

플랫폼  ATX (30.5x24.4cm)  M-ATX (24.4x24.4cm)  Mini-ITX (17.0x17.0cm)  E-ATX (30.5x33.0cm)  M-STX (14.0x14.7cm)

메모리 종류  DDR5  DDR4  LPDDR4  LPDDR3  DDR3  DDR2

메모리 속도  PC5-56000 (7,000MHz)  PC5-54400 (6,800MHz)  PC5-53300 (6,666MHz)  PC5-52800 (6,600MHz)  PC5-51200 (6,400MHz)

인기상품순 신상품순 낮은 가격순 높은 가격순 상품명순 결과 내 검색

광고상품이며, 검색결과와 다를 수 있습니다.

MSI MAG B560M 박격포 인텔(소켓1200)/인텔 B560/M-ATX (24.4x24.4cm)/전원부:12+2+1페이즈/DDR4/PC4-40500 (5,066MHz)/메모리:PC5-56000 (7,000MHz) 상품의견	161,670원	<span>담기</span>
ASRock B660M Pro RS D4 메인보드 인텔(소켓1200)/인텔 B660/M-ATX (24.4x24.4cm)/전원부:8페이즈/DDR4/PC4-38400 (4,800MHz)/메모리:PC5-54400 (6,800MHz) 상품의견 <span>이벤트</span> 구매 시 게이밍 마우스 충첨 증정!	163,500원	<span>해제</span>
GIGABYTE B560M DS3H PLUS 듀러블에디션 피씨디렉트 인텔(소켓1200)/인텔 B560/M-ATX (24.4x24.4cm)/전원부:6+2페이즈/DDR4/PC4-42600 (5,333MHz)/메모리:PC5-56000 (7,000MHz) MD추천 상품의견	110,670원	<span>담기</span>
MSI PRO H610M-G DDR4 인텔(소켓1700)/인텔 H610/M-ATX (23.6x20.2cm)/DDR4/PC4-25600 (3,200MHz)/메모리 용량:최대 64GB/ MD추천 상품의견 무보정리뷰	124,780원	<span>담기</span>

PC 견적

다나와 제공

현금 최저가 카드 최저가

건적카트 로그인하여 저장한 견적들을 확인해보세요.

PC주요부품

CPU  
인텔 코어i9-12세대 12900K (엠더레이크) (정품) 824,900원

이벤트 구매 인증 시 XBOX 게임패스 3개월 충첨 증정

쿨러/튜닝  
JONASO CR-1000 AUTO RGB (WHITE) 26,490원

메인보드  
ASRock B660M Pro RS D4 메인보드 163,500원

메모리

그래픽카드

SSD

하드디스크

외장HDD/SSD

케이스

파워

분체선택

추가상품

모니터 변경

케이스 변경

쿨러 변경

램 메모리 추가/SSD변경

하드디스크 추가

메인보드 변경

주변기기

공유기

기타

총 상품 금액 ? 0원

구매하기

톡톡문의 찜하기 264 장바구니

① 쇼핑할 때 필독 [안전거래TIP](#)

② 상품정보에 문제가 있나요? [신고하기](#)

PC 견적

네이버 스마트스토어 제공

# 명령어 비교



Ncloud CLI

```
# VPC 정보 확인  
$vpcNo=((ncloud vpc getVpcList --vpcName devops-edu) | ConvertFrom-  
Json).getVpcListResponse.vpcList.vpcNo  
  
# NACL 정보 확인  
$naclNo=((ncloud vpc getNetworkAclList --vpcNo $vpcNo) | ConvertFrom-  
Json).getNetworkAclListResponse.networkAclList.networkAclNo  
  
ncloud vpc createSubnet --regionCode KR --zoneCode KR-1`  
    --vpcNo $vpcNo --subnetName devops-edu-t2`  
    --subnet 10.0.20.0/24 --networkAclNo $naclNo`  
    --subnetTypeCode PRIVATE
```



Ncloud API

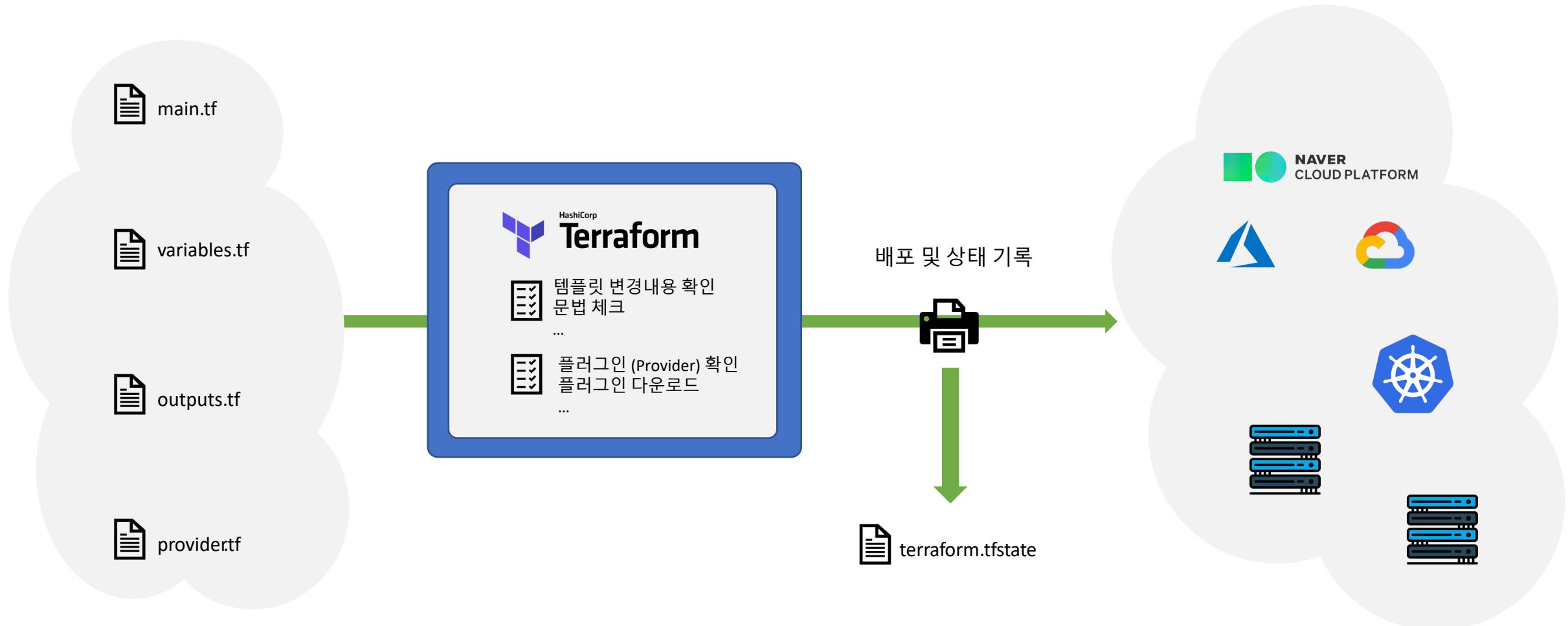
```
$cred = Get-Content -Raw -Path  
"./credentials/key.json" | ConvertFrom-Json  
$resource =  
"/vpc/v2/createSubnet?{0}&{1}&{2}&{3}&{4}&{5}" -f  
"zoneCode=KR-1",  
"vpcNo=<your-VPC>",  
"subnetName=devops-edu-dev",  
"subnet=10.0.100.0/24",  
"networkAclNo=<your-ACL>",  
"subnetTypeCode=PUBLIC"  
  
$url = "{0}{1}" -f  
"https://ncloud.apigw.ntruss.com", $resource  
$signedHeader = Create-SignedSignature  
    -method "GET"  
    -url $resource  
    -accessKey $cred.accessId  
    -secretKey $cred.secret  
  
Invoke-WebRequest -Method Get  
    -Uri $url  
    -Headers $signedHeader
```



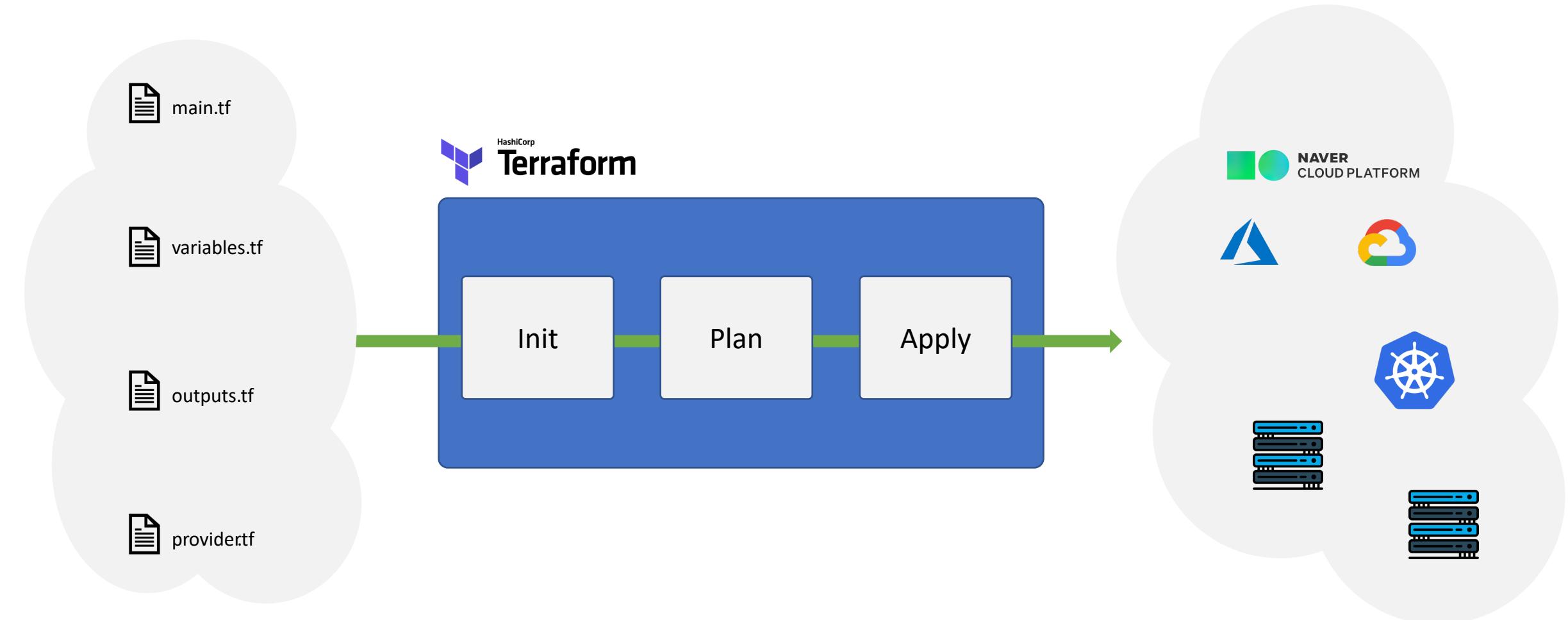
Terraform

```
resource "ncloud_vpc" "vpc" {  
    name = "${var.vpcName}"  
    ipv4_cidr_block = "10.100.0.0/16"  
}  
  
resource "ncloud_network_acl" "nac1" {  
    depends_on = [  
        ncloud_vpc.vpc  
    ]  
    vpc_no = ncloud_vpc.vpc.id  
    name = "${var.vpcName}-nac1"  
}
```

# Terraform 동작방식



# Terraform 배포 절차



# Terraform 코드 살펴보기

main.tf

```
resource "local_file" "pet" {
    filename = "C:/Users/Administrator/Desktop/Lab-Test/test.txt"
    content = "My favorite pet is ${random_pet.my-pet.id}"
}

resource "random_pet" "my-pet" {
    prefix = var.prefix
    separator = var.separator
    length = var.length
}
```

variables.tf

```
variable "filename" {
    default = "C:/Users/Administrator/Desktop/Lab-Test/test.txt"
}

variable "prefix" {
    default = "Mrs"
}

variable "separator" {
    default = "."
}

variable "length" {
    default = "1"
}
```

provider.tf

```
terraform {
    required_providers {
        local = {
            source = "hashicorp/local"
            version = "2.2.2"
        }
    }

    provider "local" {
        # Configuration options
    }
}
```

outputs.tf

```
output pet-name {
    value = random_pet.my-pet.id
    description = "Random_pet resource ID"
}
```

# Terraform 코드 살펴보기

## HCL(HashiCorp Configuration Language)과 Resource 블록



<https://www.terraform.io/language/syntax>

<https://www.terraform.io/language/resources>

# Terraform 주요 명령어

## terraform init

- 구성 파일들이 포함된 Working Directory 초기화
- Terraform Provider 다운로드
- Working Directory를 최신으로 유지하기 위해  
여러 번 실행 하는 것이 안전함

The screenshot shows a terminal window with the following output:

```
File Edit Selection View Go Run
EXPLORER
LAB-TEST
> .terraform
└ .terraform.lock.hcl
main.tf
outputs.tf
provider.tf
variables.tf
...
Initializing the backend...
Initializing provider plugins...
- Finding hashicorp/local versions matching "2.2.2"...
- Finding latest version of hashicorp/random...
- Installing hashicorp/random v3.1.0...
- Installed hashicorp/random v3.1.0 (signed by HashiCorp)
- Installing hashicorp/local v2.2.2...
- Installed hashicorp/local v2.2.2 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider selections it made above. Include this file in your version control repository so that Terraform can guarantee to make the same selections by default when you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see any changes that are required for your infrastructure. All Terraform commands should now work.

If you ever set or change modules or backend configuration for Terraform, rerun this command to reinitialize your working directory. If you forget, other commands will detect it and remind you to do so if necessary.
PS C:\Users\Administrator\Desktop\Lab-Test>
```

# Terraform 주요 명령어

## terraform plan

- 이미 존재하는 원격 개체의 현재 상태를 읽고 State가 최신 상태인지 확인
- 현재 구성을 이전 상태와 비교하고 차이점을 확인
- Apply될 경우, 현재 설정과 원격 개체를 일치시키기 위해 Terraform이 어떤 작업을 수행하는지 보여줌

```
PS C:\Users\Administrator\Desktop\Lab-Test> terraform plan
Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
+ create
Terraform will perform the following actions:
# local_file.pet will be created
+ resource "local_file" "pet" {
  + content      = (known after apply)
  + directory_permission = "0777"
  + file_permission   = "0777"
  + filename       = "C:/Users/Administrator/Desktop/Lab-Test/test.txt"
  + id            = (known after apply)
}

# random_pet.my-pet will be created
+ resource "random_pet" "my-pet" {
  + id          = (known after apply)
  + length     = 1
  + prefix     = "Mrs"
  + separator  = "."
}

Plan: 2 to add, 0 to change, 0 to destroy.

Changes to Outputs:
+ pet-name = (known after apply)

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run "terraform apply" now.
PS C:\Users\Administrator\Desktop\Lab-Test>

PS C:\Users\Administrator\Desktop\Lab-Test> terraform plan
random_pet.my-pet: Refreshing state... [id=Mrs.feline]
local_file.pet: Refreshing state... [id=61149ac1b16f2a62896439e88c52eed6aba943c9]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
-/+ destroy and then create replacement

Terraform will perform the following actions:
# local_file.pet must be replaced
-/+ resource "local_file" "pet" {
  ~ content      = "My favorite pet is Mrs.feline" -> (known after apply) # forces replacement
  ~ id          = "61149ac1b16f2a62896439e88c52eed6aba943c9" -> (known after apply)
  ~ # (3 unchanged attributes hidden)
}

# random_pet.my-pet must be replaced
-/+ resource "random_pet" "my-pet" {
  ~ id          = "Mrs.feline" -> (known after apply)
  ~ prefix     = "Mrs" -> "Hello" # forces replacement
  ~ # (2 unchanged attributes hidden)
}

Plan: 2 to add, 0 to change, 2 to destroy.

Changes to Outputs:
~ pet-name = "Mrs.feline" -> (known after apply)

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run "terraform apply" now.
PS C:\Users\Administrator\Desktop\Lab-Test>
```

# Terraform 주요 명령어

## terraform apply

- terraform plan 단계에서 보여준 작업을 실행
- 각 Resource 의존성에 따라 순차적으로 작업 수행
- 작업 실패시 자동 롤백을 수행하지 않음

```
main.tf > resource "random_pet" "my-pet" > prefix
1   resource "local_file" "pet" {
2     filename = "C:/Users/Administrator/Desktop/Lab-Test/test.txt"
3     content = "My favorite pet is ${random_pet.my-pet.id}"
4   }
5
6   resource "random_pet" "my-pet" {
7     prefix = "var.prefix"
8     separator = var.separator
9     length = var.length
10 }

File Edit Selection View Go Run
EXPLORER LAB-TEST
> .terraform
> main.tf
> outputs.tf
> provider.tf
{>} terraform.tfstate
> test.txt
variables.tf

PS C:\Users\Administrator\Desktop\Lab-Test> terraform apply
Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
+ create

Terraform will perform the following actions:

# local_file.pet will be created
+ resource "local_file" "pet" {
  + content          = (known after apply)
  + directory_permission = "0777"
  + file_permission    = "0777"
  + filename           = "C:/Users/Administrator/Desktop/Lab-Test/test.txt"
  + id                = (known after apply)
}

# random_pet.my-pet will be created
+ resource "random_pet" "my-pet" {
  + id      = (known after apply)
  + length  = 1
  + prefix   = "var.prefix"
  + separator = "."
}

Plan: 2 to add, 0 to change, 0 to destroy.

Changes to Outputs:
+ pet-name = (known after apply)

Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: yes
random_pet.my-pet: Creating...
random_pet.my-pet: Creation complete after 0s [id=var.prefix.clam]
local_file.pet: Creating...
local_file.pet: Creation complete after 0s [id=691d1e02dee07a03be0352ddc0bc3fdaf71d6fd]

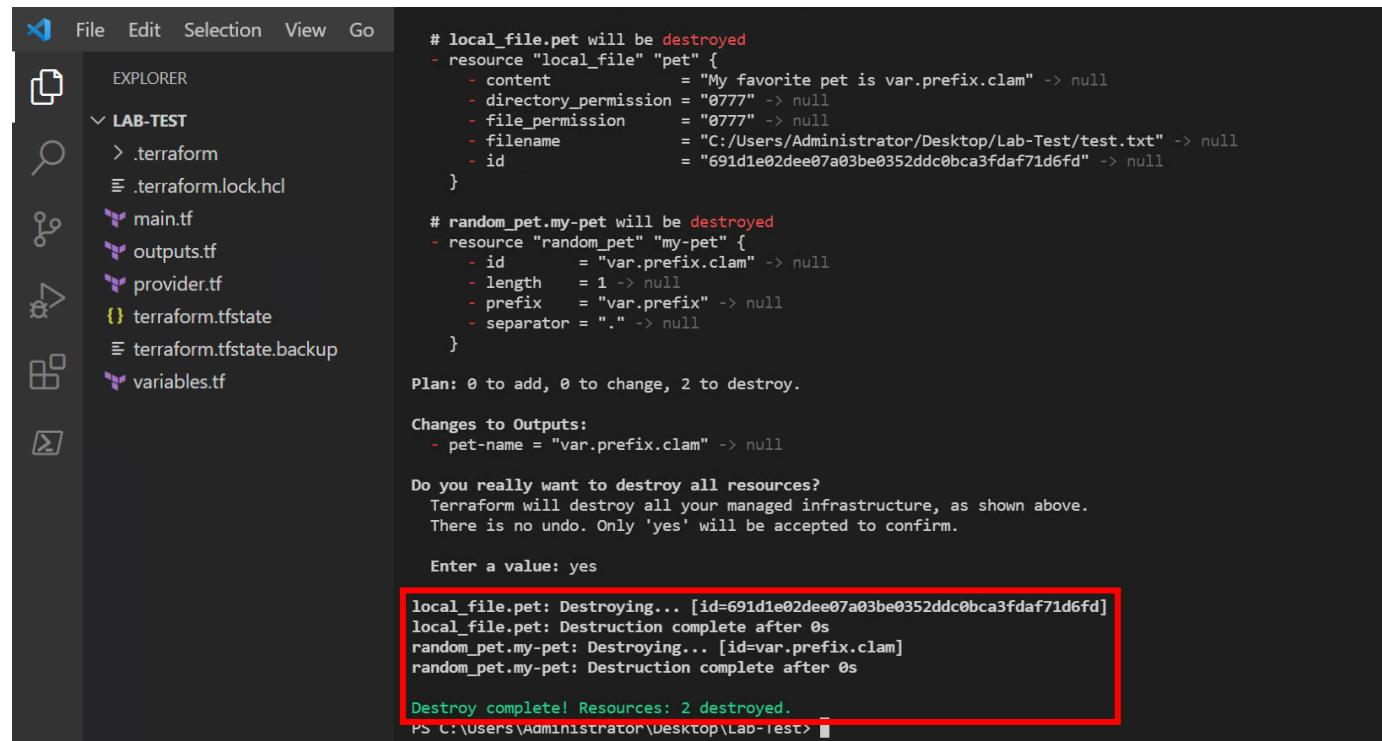
Apply complete! Resources: 2 added, 0 changed, 0 destroyed.

Outputs:
pet-name = "var.prefix.clam"
```

# Terraform 주요 명령어

## terraform destroy

- 특정 Terraform 구성에서 관리하는 모든 원격 개체를 삭제
- 일반적인 운영 환경에선 사용하지 않음
- 주로 테스트, 개발환경 등을 한꺼번에 삭제하기 위해 사용



```
# local_file.pet will be destroyed
- resource "local_file" "pet" {
  - content      = "My favorite pet is var.prefix.clam" -> null
  - directory_permission = "0777" -> null
  - file_permission   = "0777" -> null
  - filename        = "C:/Users/Administrator/Desktop/Lab-Test/test.txt" -> null
  - id             = "691d1e02dee07a03be0352ddc0bca3fdaf71d6fd" -> null
}

# random_pet.my-pet will be destroyed
- resource "random_pet" "my-pet" {
  - id          = "var.prefix.clam" -> null
  - length     = 1 -> null
  - prefix     = "var.prefix" -> null
  - separator  = "." -> null
}

Plan: 0 to add, 0 to change, 2 to destroy.

Changes to Outputs:
- pet-name = "var.prefix.clam" -> null

Do you really want to destroy all resources?
Terraform will destroy all your managed infrastructure, as shown above.
There is no undo. Only 'yes' will be accepted to confirm.

Enter a value: yes

local_file.pet: Destroying... [id=691d1e02dee07a03be0352ddc0bca3fdaf71d6fd]
local_file.pet: Destruction complete after 0s
random_pet.my-pet: Destroying... [id=var.prefix.clam]
random_pet.my-pet: Destruction complete after 0s

Destroy complete! Resources: 2 destroyed.
PS C:\users\Administrator\Desktop\Lab-Test>
```

# Terraform 실행 결과

## 결과물

Explorer view showing the following files:

- .terraform
- .terraform.lock.hcl
- main.tf
- outputs.tf
- provider.tf
- terraform.tfstate
- terraform.tfstate.backup
- test.txt
- variables.tf

The test.txt file contains the content: "My favorite pet is var.prefix.coral".

## 배포 결과 // terraform.tfstate

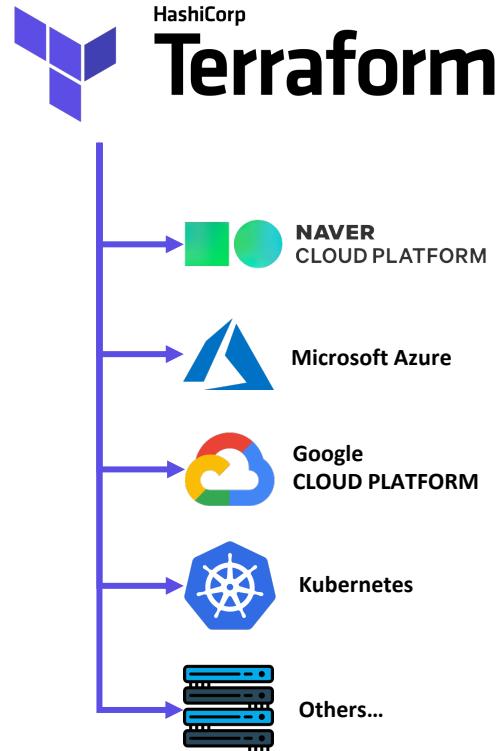
Explorer view showing the following files:

- .terraform.lock.hcl
- main.tf
- outputs.tf
- provider.tf
- terraform.tfstate
- terraform.tfstate.backup
- test.txt
- variables.tf

The terraform.tfstate file content is as follows:

```
1 {
2   "version": 4,
3   "terraform_version": "1.1.7",
4   "serial": 9,
5   "lineage": "20007a48-a39c-be75-058e-7355eddc665e",
6   "outputs": [
7     {
8       "pet-name": {
9         "value": "var.prefix.coral",
10        "type": "string"
11      }
12    },
13    "resources": [
14      {
15        "mode": "managed",
16        "type": "local_file",
17        "name": "pet",
18        "provider": "provider[\"registry.terraform.io/hashicorp/local\"]",
19        "instances": [
20          {
21            "schema_version": 0,
22            "attributes": {
23              "content": "My favorite pet is var.prefix.coral",
24              "content_base64": null,
25              "directory_permission": "0777",
26              "file_permission": "0777",
27              "filename": "C:/Users/Administrator/Desktop/Lab-Test/test.txt",
28              "id": "a58e01a8925d0285a753945beac0253adfed8c0b",
29              "sensitive_content": null,
30              "source": null
31            },
32            "sensitive_attributes": [],
33            "private": "bnVsbA==",
34            "dependencies": [
35              "random_pet.my-pet"
36            ]
37          }
38        ],
39        "mode": "managed"
40      }
41    ]
42 }
```

# Terraform Providers



## Providers

- 다양한 원격 개체의 배포를 지원하기 위한 플러그인



### Official

- HashiCorp에서 직접 관리하는 플러그인
- 관리 책임은 HashiCorp에 있음



### Verified

- 네이버클라우드플랫폼과 같은 공식 서비스 제공자가 관리하는 플러그인
- 관리 책임은 서비스 제공자에게 있음

### Community

- Terraform 사용자들이 개발하고 관리하는 비공식 플러그인
- 사용 문의 버그 제보 등은 커뮤니티를 통해 이루어짐

# Terraform Provider – Naver CLOUD PLATFORM



NAVER  
CLOUD PLATFORM

NCLOUD DOCUMENTATION

Filter

ncloud provider

Resources

- ncloud\_access\_control\_group
- ncloud\_access\_control\_group\_rule
- ncloud\_auto\_scaling\_group
- ncloud\_auto\_scaling\_policy
- ncloud\_auto\_scaling\_schedule
- ncloud\_block\_storage
- ncloud\_block\_storage\_snapshot
- ncloud\_init\_script
- ncloud\_launch\_configuration
- ncloud\_lb
- ncloud\_lb\_listener
- ncloud\_lb\_target\_group
- ncloud\_lb\_target\_group\_attachment
- ncloud\_load\_balancer
- ncloud\_load\_balancer\_ssl\_certificate
- ncloud\_login\_key
- ncloud\_nas\_volume
- ncloud\_nat\_gateway
- ncloud\_network\_acl

## Ncloud Provider

The Ncloud provider is used to interact with [Ncloud](#) (NAVER Cloud Platform) services. The provider needs to be configured with the proper credentials before it can be used.

### Example Usage

```
terraform {  
  required_providers {  
    ncloud = {  
      source = "NaverCloudPlatform/ncloud"  
    }  
  }  
  required_version = ">= 0.13"  
  
  // Configure the ncloud provider  
  provider "ncloud" {  
    access_key  = var.access_key  
    secret_key = var.secret_key  
    region     = var.region  
    site       = var.site  
    support_vpc = var.support_vpc  
  }  
  
  // Create a new server instance  
  resource "ncloud_server" "server" {  
    # ...  
  }  
}
```

<https://registry.terraform.io/providers/NaverCloudPlatform/ncloud/latest/docs>

# Terraform Hands-on

End of Document  
Thank You