NHN Cloud OpenStack의 Multi Region 배포 전략

NHN Cloud 조성수

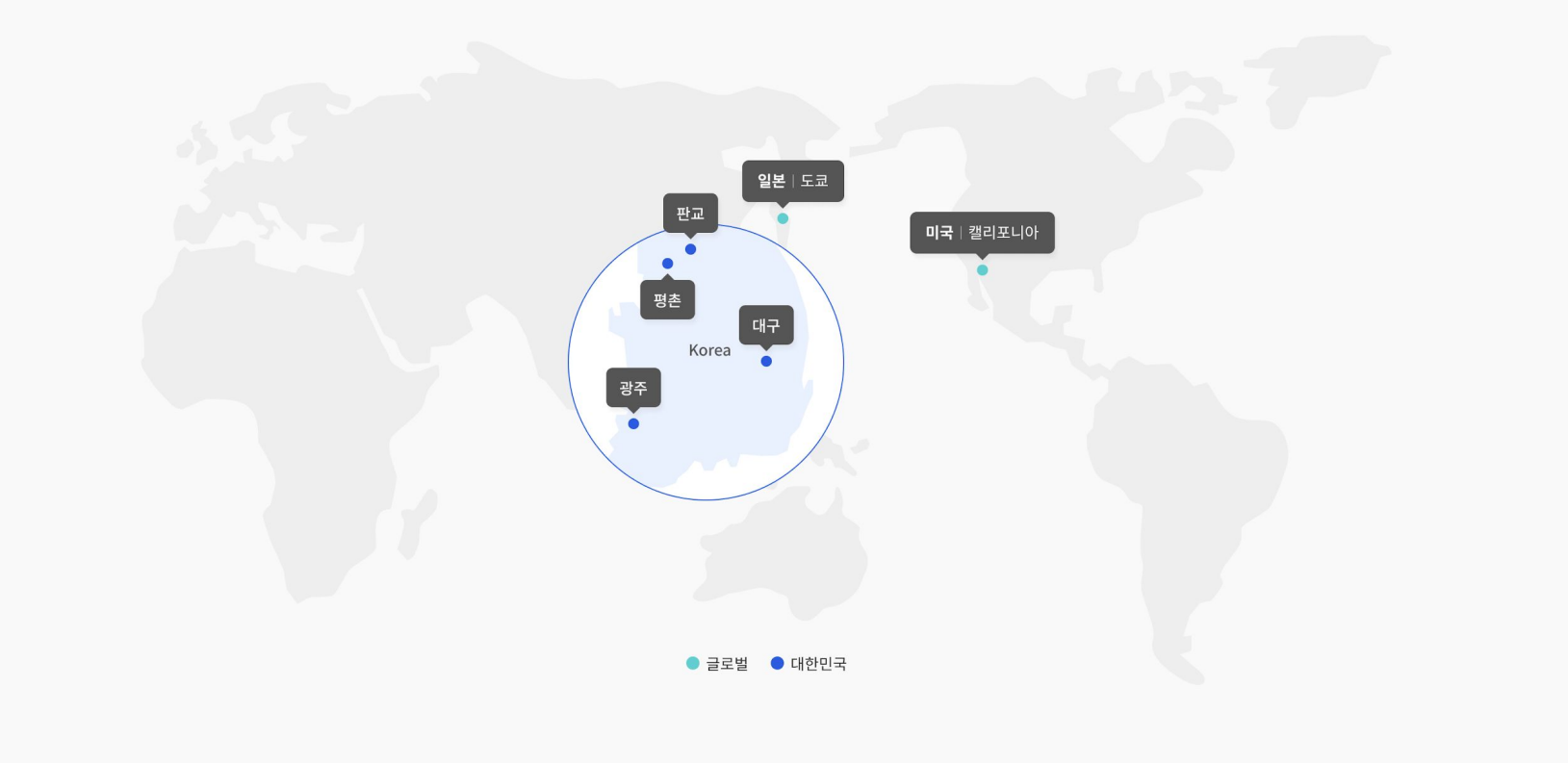
NHN Cloud 한윤범

NHN

목차

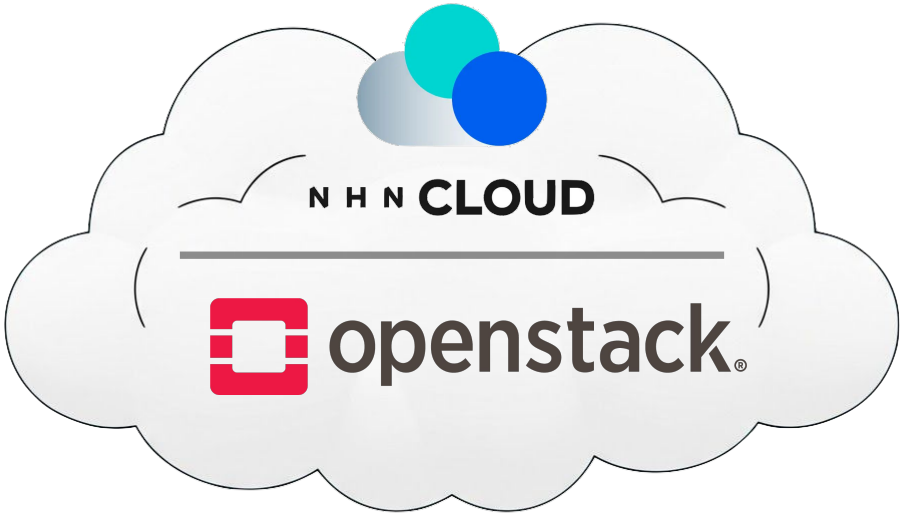
NHN Cloud의 OpenStack 환경

NHN Cloud 리전 구성



NHN Cloud의 인프라 서비스

NHN Cloud 의 인프라 서비스 (IaaS)는 OpenStack 기반으로, 자체 기술력을 더해 다양한 인프라 기능을 제공합니다



NHN Cloud의 인프라 서비스에서 제공하는 기능

Compute

- Instance
- Ephemeral Storage Instance
- GPU Instance
- Cloud Functions
- Instance Template
- Image
- Image Builder
- Auto Scale
- Virtual Desktop

Network

- VPC
- Network Interface
- Flow Log
- Floating IP
- Network ACL
- Security Groups
- Load Balancer
- NAT Instance
- Transit Hub
- Internet Gateway
- Peering Gateway
- Colocation Gateway
- NAT Gateway
- VPN Gateway(Site-to-Site VPN)
- Service Gateway
- Traffic Mirroring
- Private DNS
- DNS Plus
- Direct Connect

Storage

- Block Storage
- NAS
- NAS (offline)
- Object Storage
- Backup
- Storage Gateway
- Data Transporter

NHN Cloud의 OpenStack 구성 - Component

그 외 ceilometer, aodh, gnocchi 등 다양한 컴포넌트 사용

NHN Cloud의 OpenStack 구성 - Node

in-house
IaaS Service

VM

Baremetal

Container

NHN Cloud는 OpenStack을 어떻게 구축할까?

OpenStack 구성에 많이 쓰는 도구

kolla-ansible

NHN Cloud IaaS가 사용하는

배포 / 구성 도구

왜 SaltStack인가?

SaltStack은 사내 배포 도구의 표준

OpenStack 외 다른 서비스들도 배포하고 설정을 관리 해야함

다양한 상황에 유연한 대응을 위해 자체 배포 / 구성 코드가 필요했음

우리는 무엇을 배포하는가?

OpenStack

RabbitMQ / HAProxy / Keepalived / ELK / 모니터링 도구

자체 개발한 IaaS 서비스

리눅스 시스템 설정 값

배포 코드는 구성만 담당하는가?

SaltStack Orchestration

- 여러 minion 과 서비스에 걸친 작업을 순서 / 의존성을 가지고 실행하는 절차서
- 단순한 설정 관리를 넘어 복잡한 워크플로 자동화를 지원
  - 예) api -> agent 순으로 배포 | 다수 서버 순차 업데이트

노드에서 필수로 검사해야 하는 항목 확인

- 설치해야 하는 패키지가 저장소에 존재하는 지 확인
- network 및 database ACL(Access Control List) 검사

NHN Cloud IaaS의 Multi Region 배포

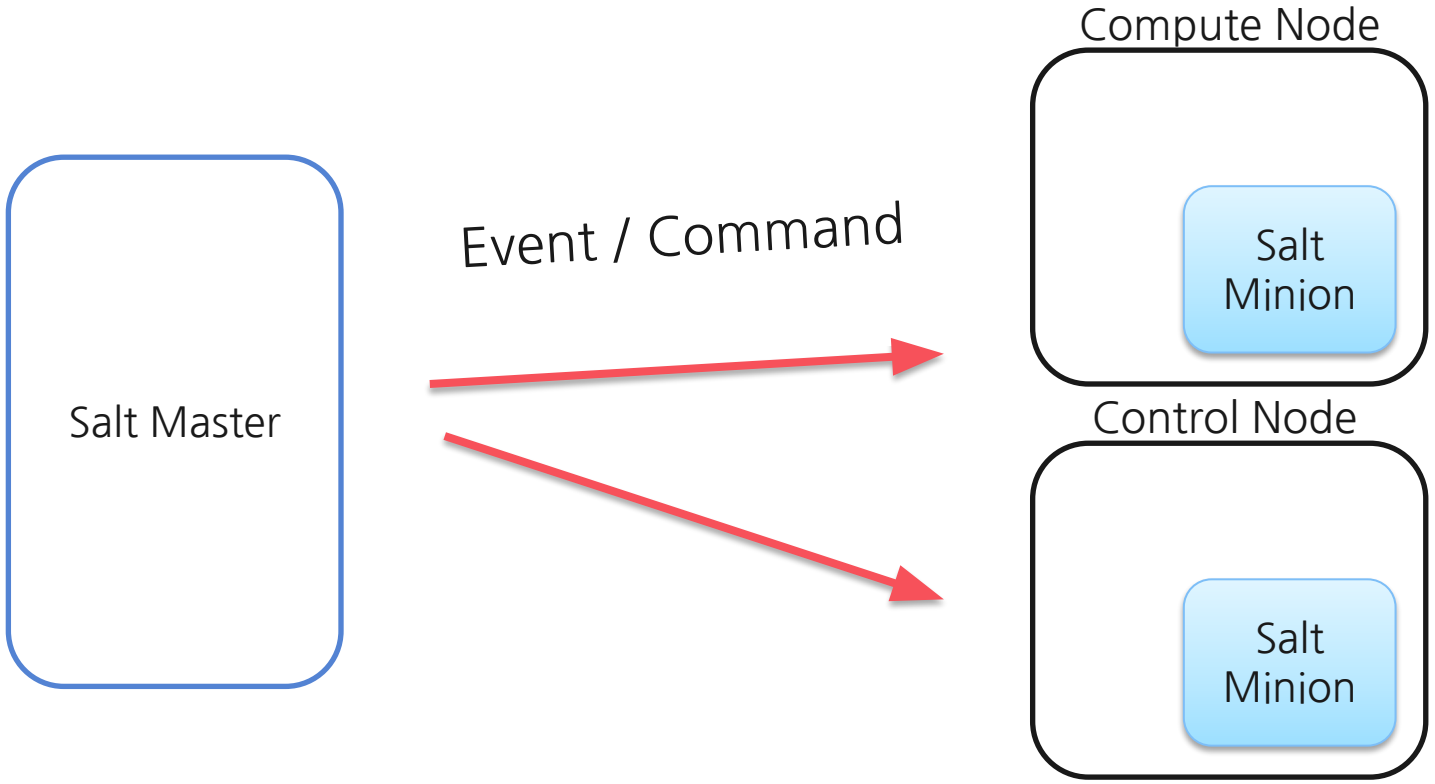
NHN Cloud IaaS의 Multi Region 구성 전략

하나의 배포 코드와

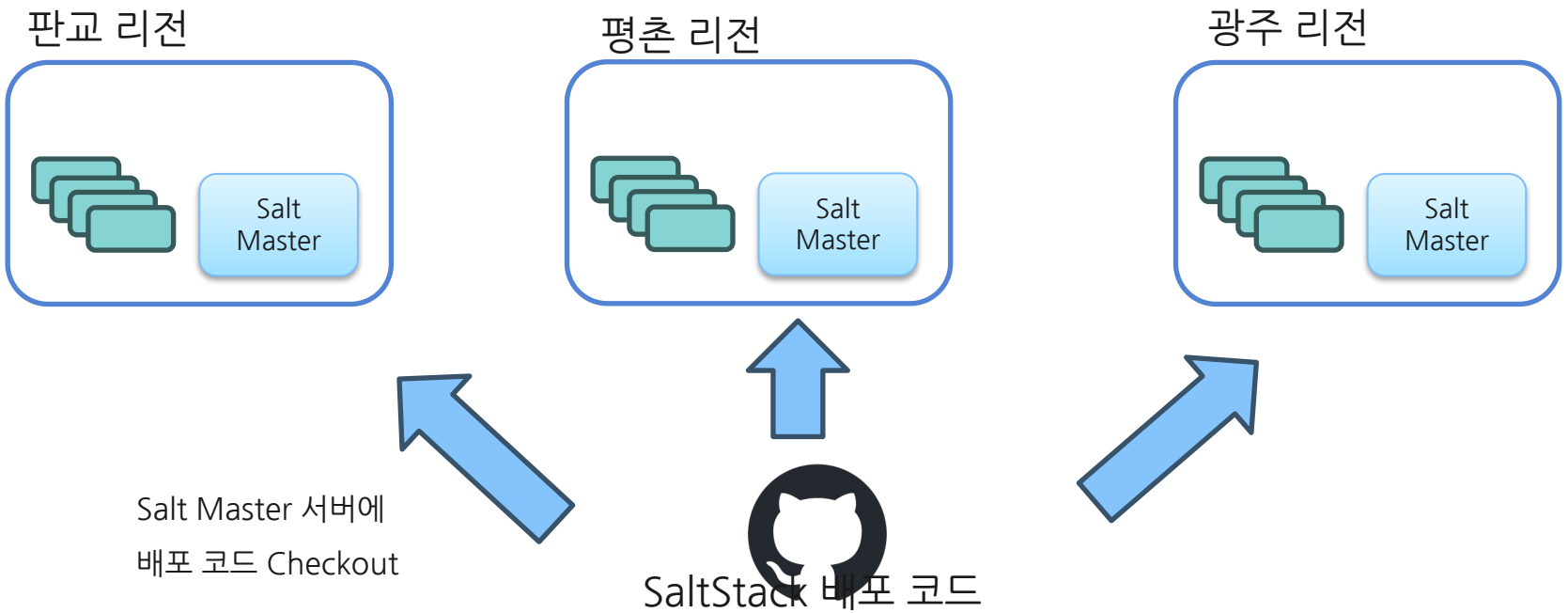
하나의 오픈스택 패키지로

모든 리전을 같은 형상으로 구성한다

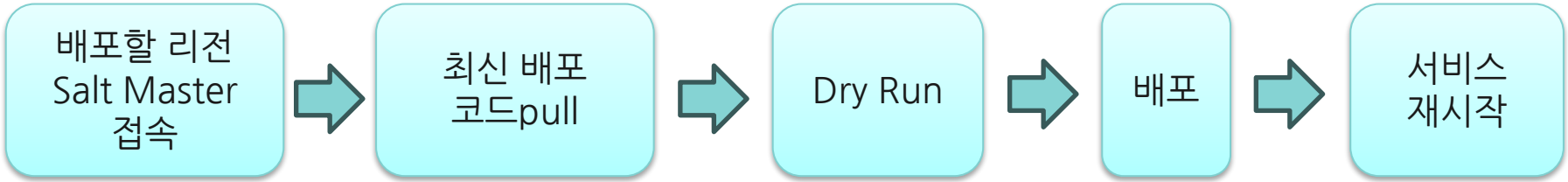
SaltStack 의 구조



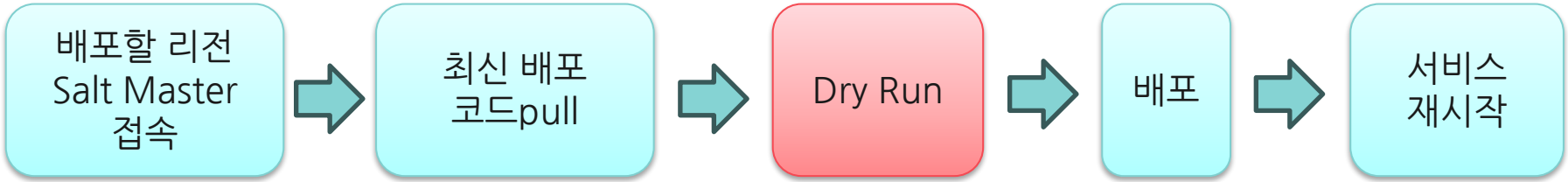
NHN Cloud Multi Region 배포 환경



NHN Cloud Multi Region 배포 환경 - 작업자 측면



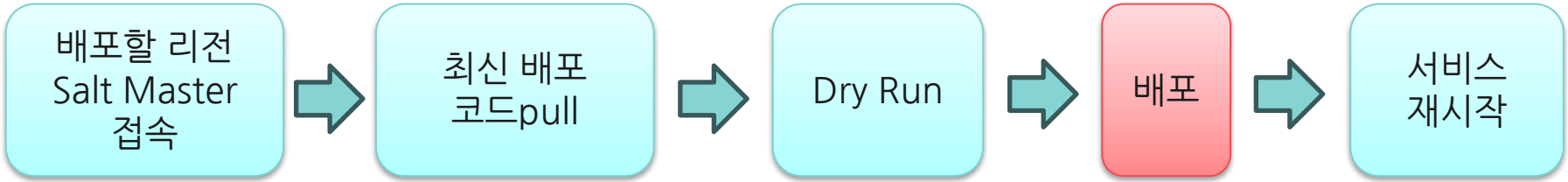
NHN Cloud Multi Region 배포 환경 - 작업자 측면



Dry Run

- 배포 코드와 시스템 현재 상태를 비교하여 어떤 변경(diff)이 생기는지 파악하는 과정
  - 배포 코드를 반영하기 전, 코드 이상 여부를 확인하는 단계
  - 노드 마다 변경사항이 나오기 때문에 수 백대 노드 대상으로 검증하는 데 시간이 오래 걸림

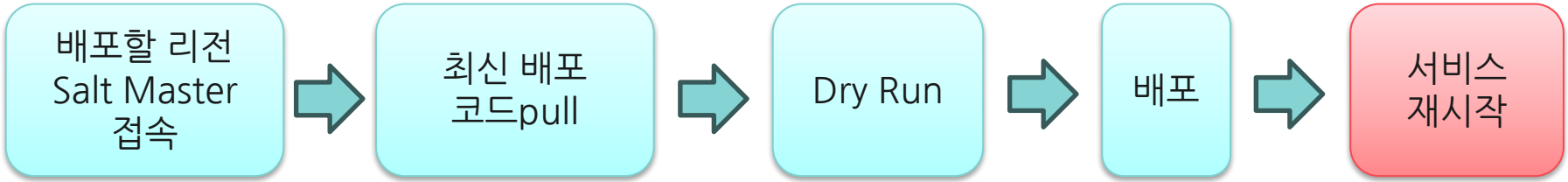
NHN Cloud Multi Region 배포 환경 - 작업자 측면



배포

- 배포 코드를 시스템에 반영
- 배포 결과를 보고 의도한 대로 반영이 잘 되었는지 한번 더 검수하는 과정이 있음 (사람이 눈으로)
- 가끔 Dry Run은 문제 없는데, 배포할 때 배포 코드가 잘 동작하지 않는 경우가 있음

NHN Cloud Multi Region 배포 환경 - 작업자 측면



서비스 재시작

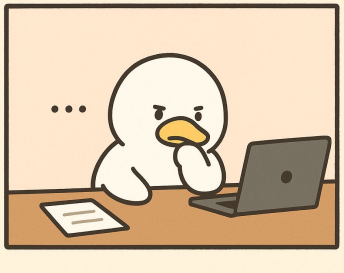
- 변경 내용을 반영하기 위해 서비스 데몬을 재시작
- 안전한 배포를 위해 서비스 데몬을 Load Balancer에서 제외, 재시작, 투입, 모니터링 순으로 작업 수행

NHN Cloud Multi Region 배포 환경 - 작업자 측면

배포할 리전 Salt Master 접속 → 최신 배포 코드pull → Dry Run → 배포 → 서비스 재시작

얼마전까지 이 모든 과정을 사람이 직접 명령어를 치며 수행했다

하지만 리전이 늘어나고 배포할 컴포넌트가 늘어나면서 작업 준비와 실행의 부담은 더 커졌다



Multi Region 배포를 위한

배포 코드 측면의 전략

SaltStack 로 만드는 배포 코드

SaltStack

Deploy Code

Pillar

- 환경별 설정 관리를 위한 데이터 저장소
- 비공개 데이터
    - 비밀번호 / 키 값 등

State

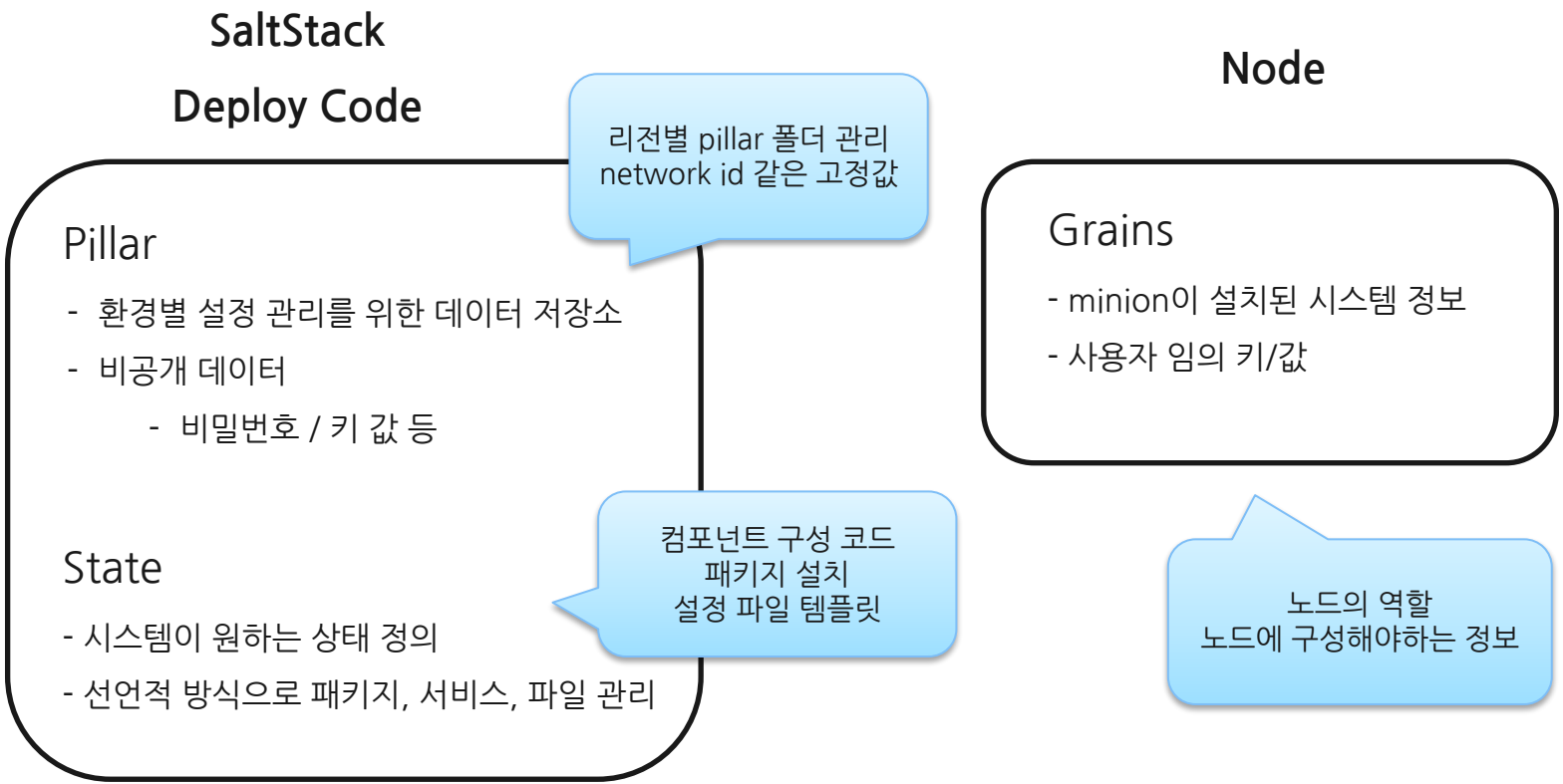
- 시스템이 원하는 상태 정의
- 선언적 방식으로 패키지, 서비스, 파일 관리

Node

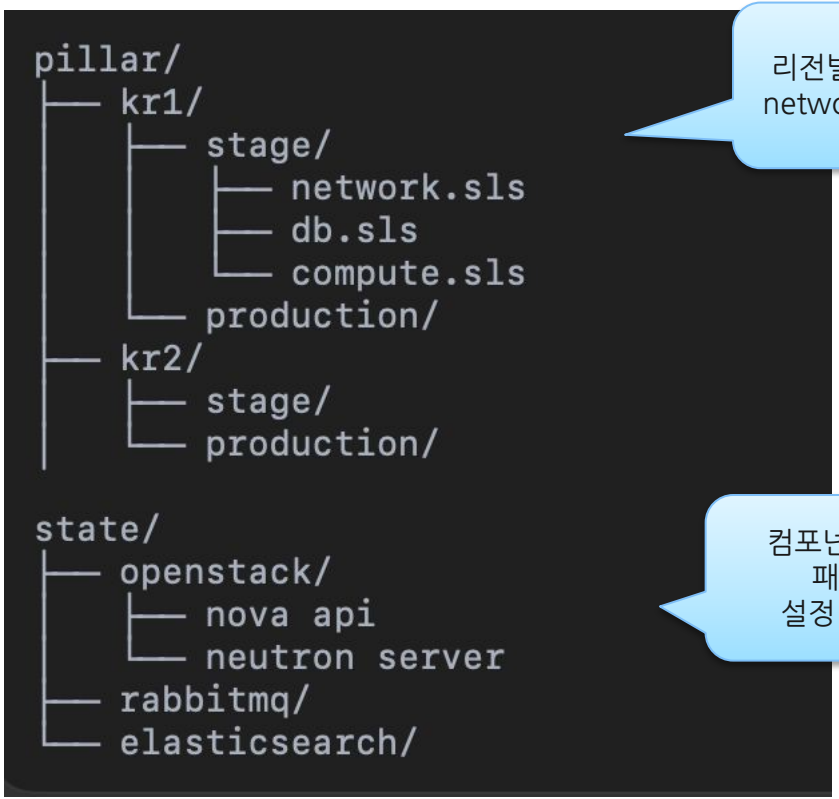
Grains

- minion이 설치된 시스템 정보
- 사용자 임의 키/값

SaltStack 로 만드는 배포 코드



SaltStack 로 만드는 배포 코드



리전별 pillar 폴더 관리
network id 같은 고정값

컴포넌트 구성 코드
패키지 설치
설정 파일 템플릿

리전마다 출시하는 서비스가 달라진다

리전마다 출시하는 IaaS 서비스 혹은 기능이 달라지거나, 변화한다

예) 특정 리전에는 NAT Gateway 서비스가 제공되지 않는다

-> (미래 요구사항) 어느 순간에 NAT Gateway 서비스를 제공해야 한다

리전마다 출시하는 서비스가 달라진다

하나의 컴포넌트 배포 코드에서 리전에 따라

서비스 / 기능 제공 여부를 어떻게 결정할까?

처음에 했던 단순한 방법

리전 이름으로 기능 배포 여부를 결정

리전이 늘어날 수록 if 조건문이 알아보기 힘들고, 배포 누락이 발생하기도 한다

```
/etc/neutron/neutron.conf:
    - file: salt/openstack/neutron-server/files/neutron.conf
    - context:
      {% if grains['region'] in ['KR1', 'KR2', 'KR3'] %}
        natgw_feature1: pillar['natgw']['feature1']
      {% endif %}
```

기능을 on/off 하는 스위치 - features

리전마다 features.sls 에 기능 on/off 를 표시하고, state에서 분기한다

```
pillar/
└── kr1/
    └── production/
        └── features.sls
```

```
features:
  - natgw: True
  - transitgw : False
  - physical lbaas: True
```

```
/etc/neutron/neutron.conf:
    - file: salt/openstack/neutron-server/files/neutron.conf
    - context:
      {% if pillar['features']['natgw'] %}
        natgw_feature1: pillar['natgw']['feature1']
      {% endif %}
```

배포 코드 관리 방법

코드 자체는 git 으로 관리하는데

언제 어느 리전에 어떤 리비전의 배포 코드가 어떤 요청에 의해

적용되었는지 추적 관리 필요

NHN Cloud IaaS의 배포 종류

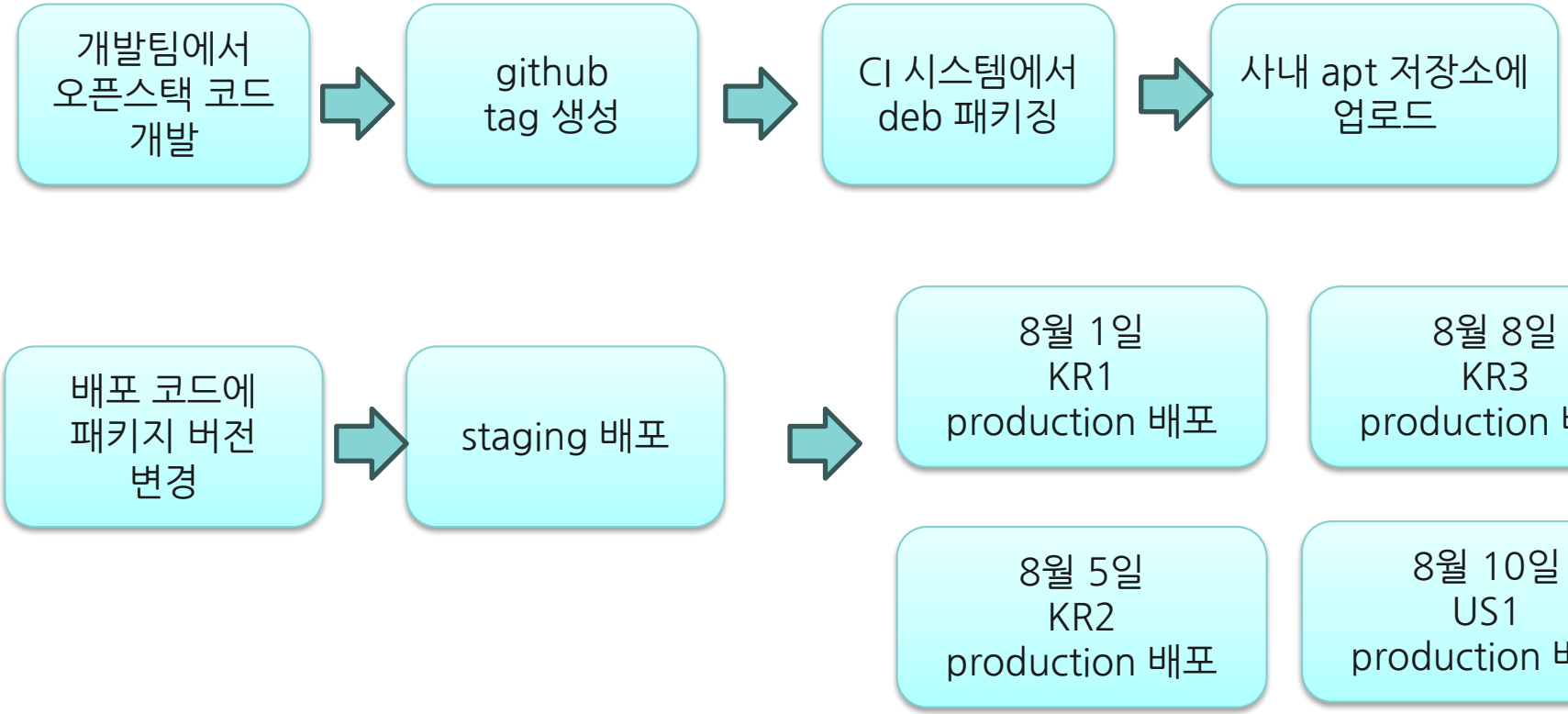
신규 리전 구성

새로운 기능 출시 / 버그 패치를 위한

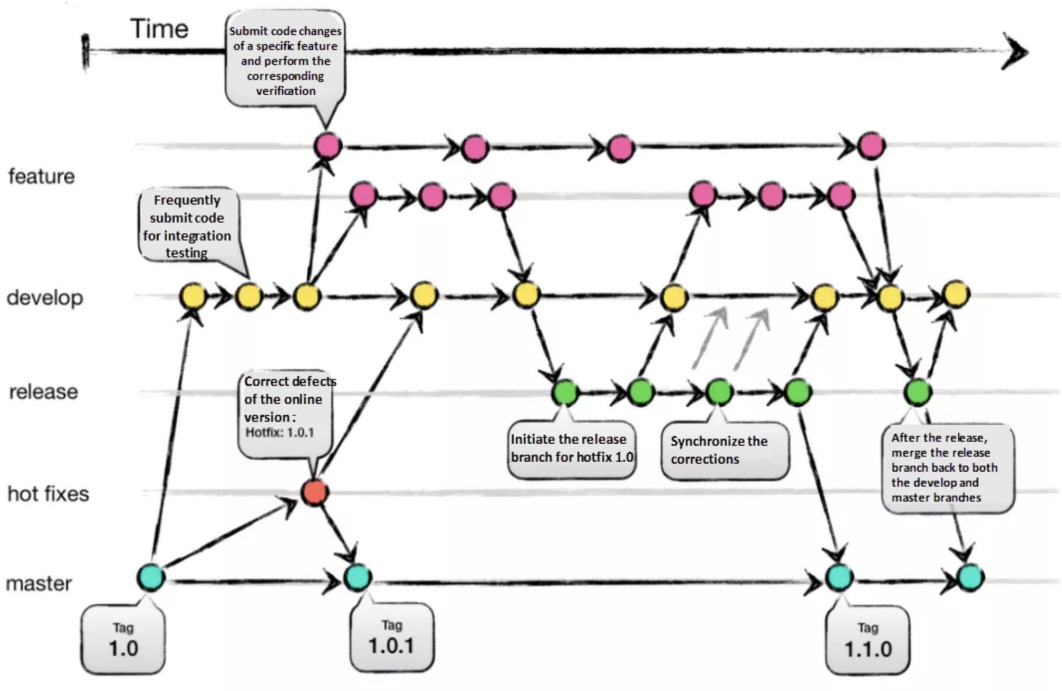
정기 /비정기 점검

긴급 핫픽스

오픈스택 코드가 서비스 서버까지 배포되는 과정

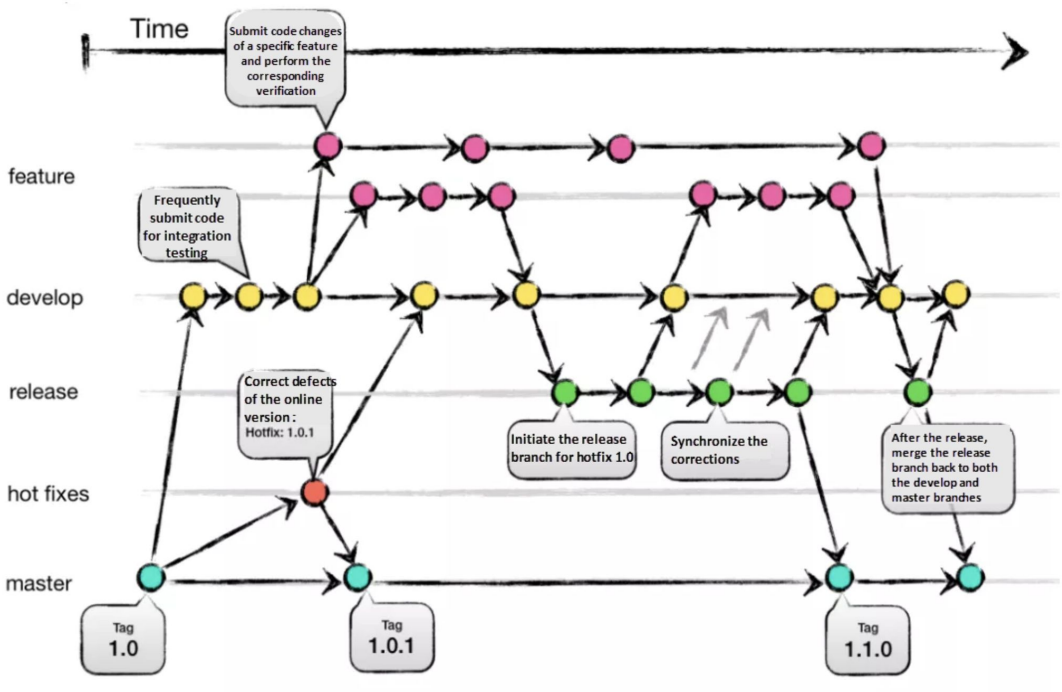


일반적인 git branch 전략

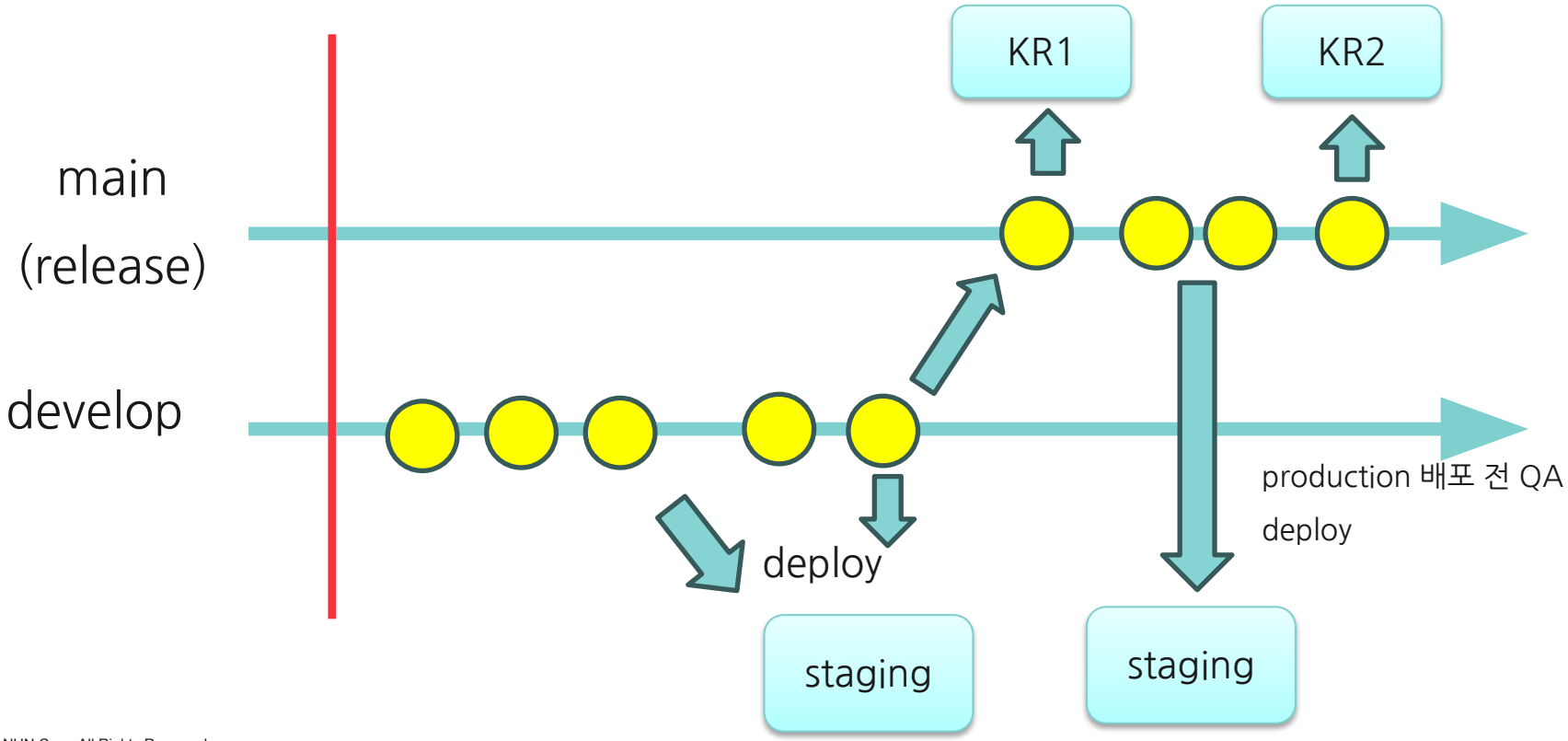


일반적인 git branch 전략

리전마다 배포 시점도 다르고, 리비전도 다른데.. 어떻게 효율적으로 관리하지?



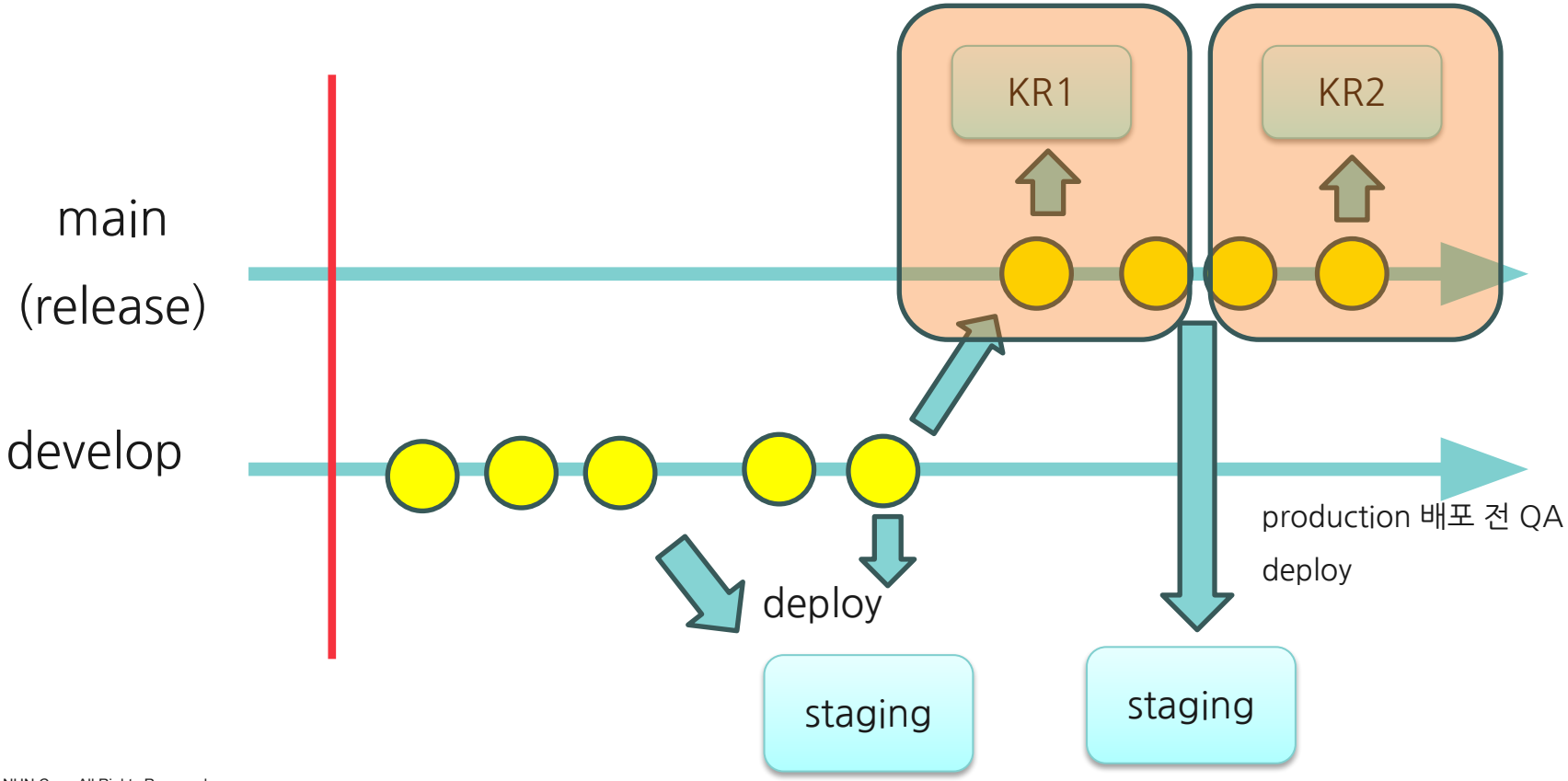
우리의 상황



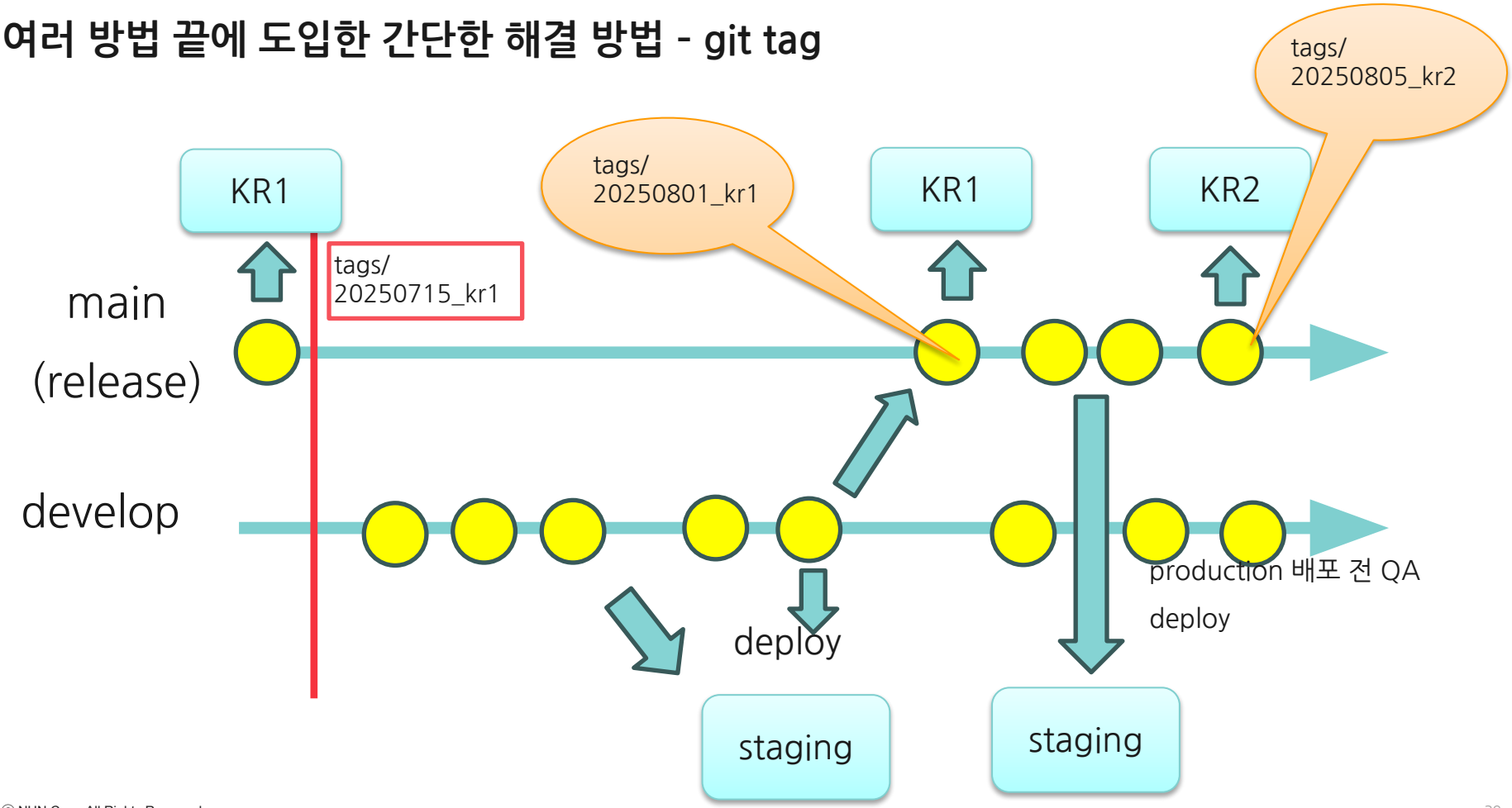
우리의 상황

리전마다 배포된 코드가 다르다

어느 리전에 어떤 코드가 배포되었는지 관리해야 한다



여러 방법 끝에 도입한 간단한 해결 방법 - git tag



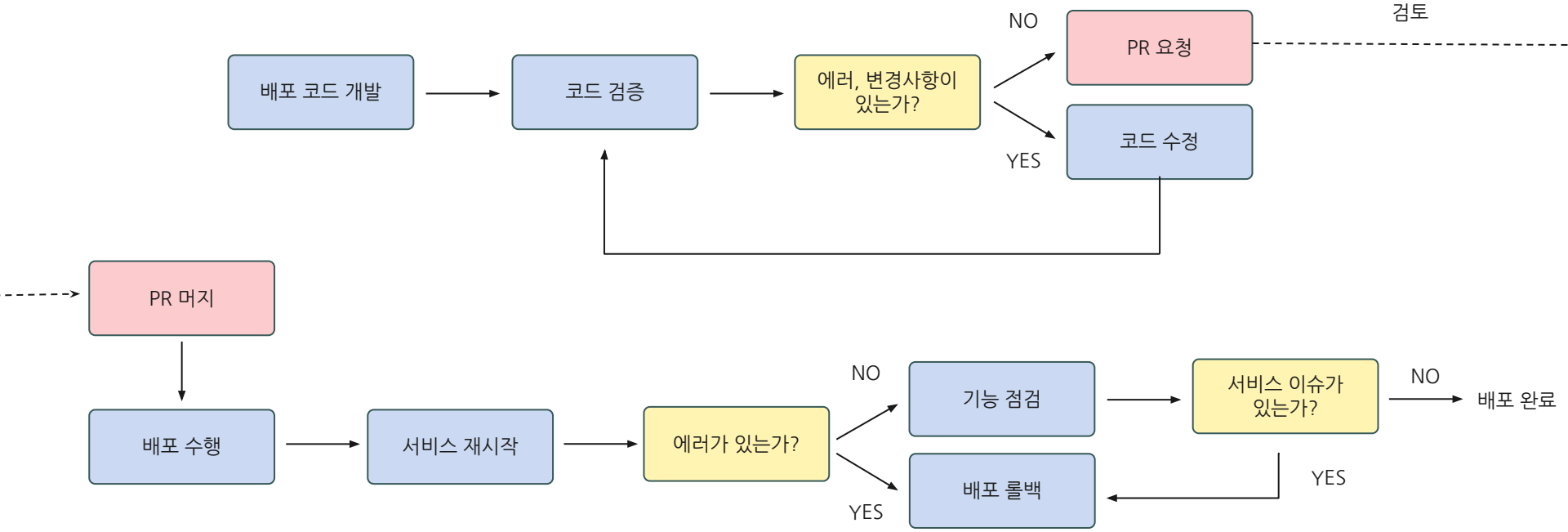
코드의 발전은

배포 준비 과정에서 많은 개선을 가져왔다

이제 우리에게 남은 건

배포 자동화 및 고도화

수동 배포 과정



수동 배포 과정

Deploy

salt master 서버 접속 (리전: A)
git pull
git checkout ~

salt ‘target’ test.ping
salt ‘target’ state.highstate saltenv=base test=true
salt ‘target’ state.highstate saltenv=base test=false

salt ‘target’ pkg.version pkg_name
salt ‘target’ cmd.run ‘cat /etc/nova/nova.conf | grep ~’

salt master 서버 접속 (리전: B)
git pull
git checkout ~

salt ‘target’ test.ping
salt ‘target’ state.highstate saltenv=base test=true
salt ‘target’ state.highstate saltenv=base test=false

salt ‘target’ pkg.version pkg_name
salt ‘target’ cmd.run ‘cat /etc/nova/nova.conf | grep ~’
…

Restart

(리전: A)
salt ‘target’ cmd.run ‘systemctl status nova-service’
salt ‘target’ cmd.run service.disable
salt ‘target’ service.restart nova-service
salt ‘target’ service.check
salt ‘target’ cmd.run service.enable

(리전: B)
salt ‘target’ cmd.run ‘systemctl status nova-service’
salt ‘target’ cmd.run service.disable
salt ‘target’ service.restart nova-service
salt ‘target’ service.check
salt ‘target’ cmd.run service.enable
…

수동 배포의 한계

반복 작업을 어디에? 모든 리전에!

단순 반복작업

시간 부족

휴먼 폴트 가능성 UP

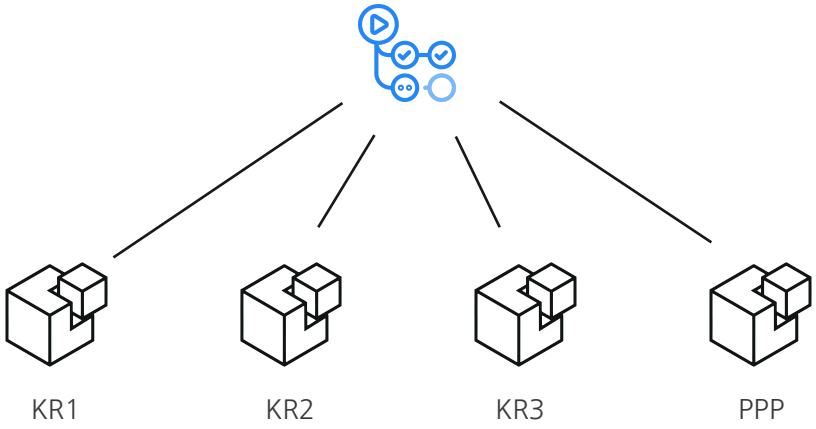
점점 스파게티 코드로..

VDI로 접속해야 하는 환경

B 설정도 추가해주세요~

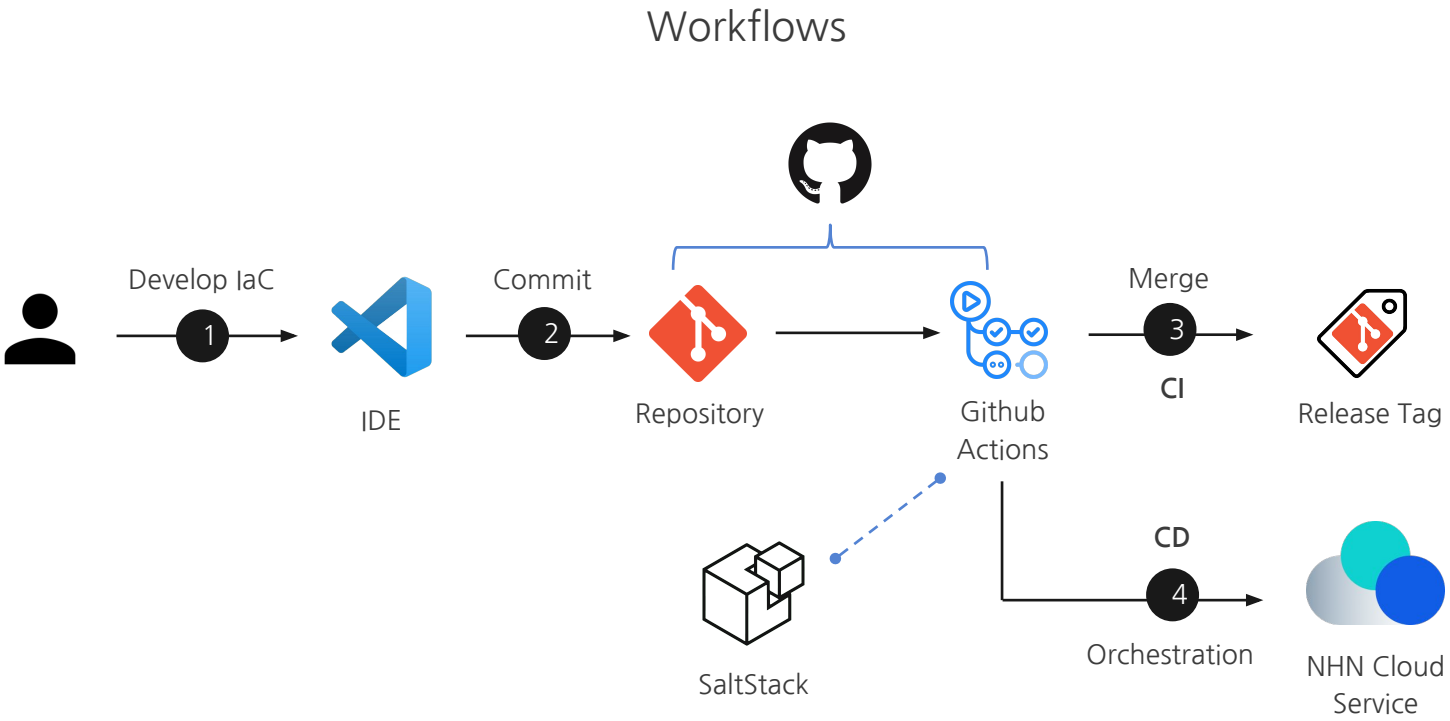
Github Actions 도입

Self-Hosted Runners

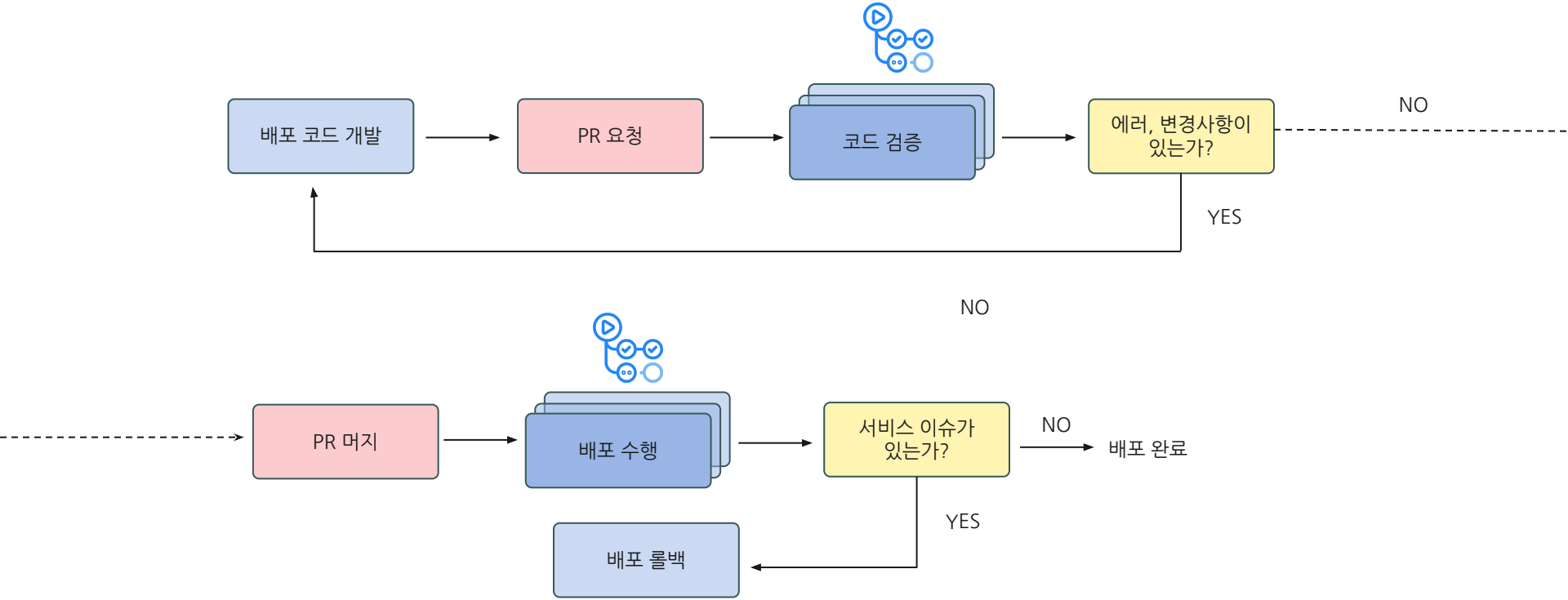


SaltStack Master 서버

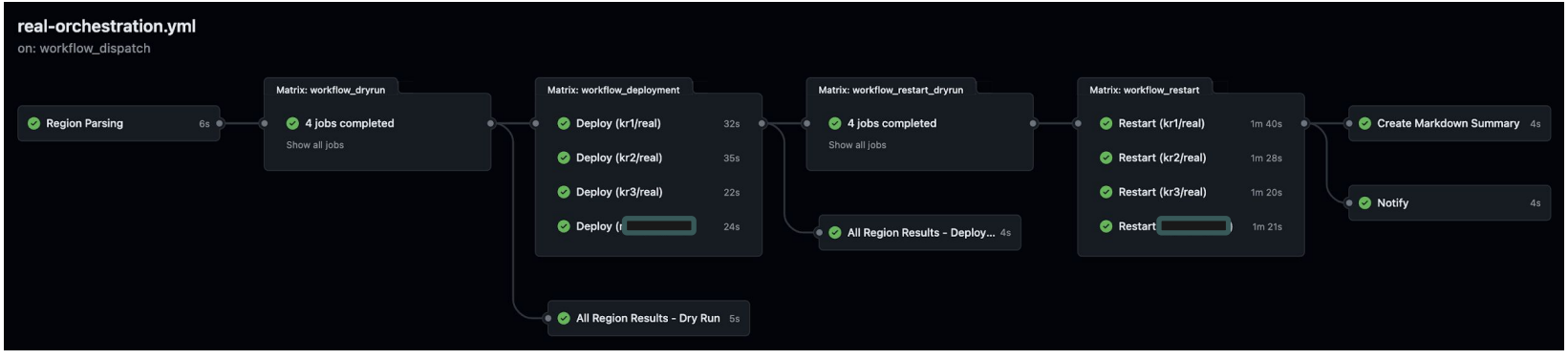
GitHub Actions 도입



GitHub Actions 도입



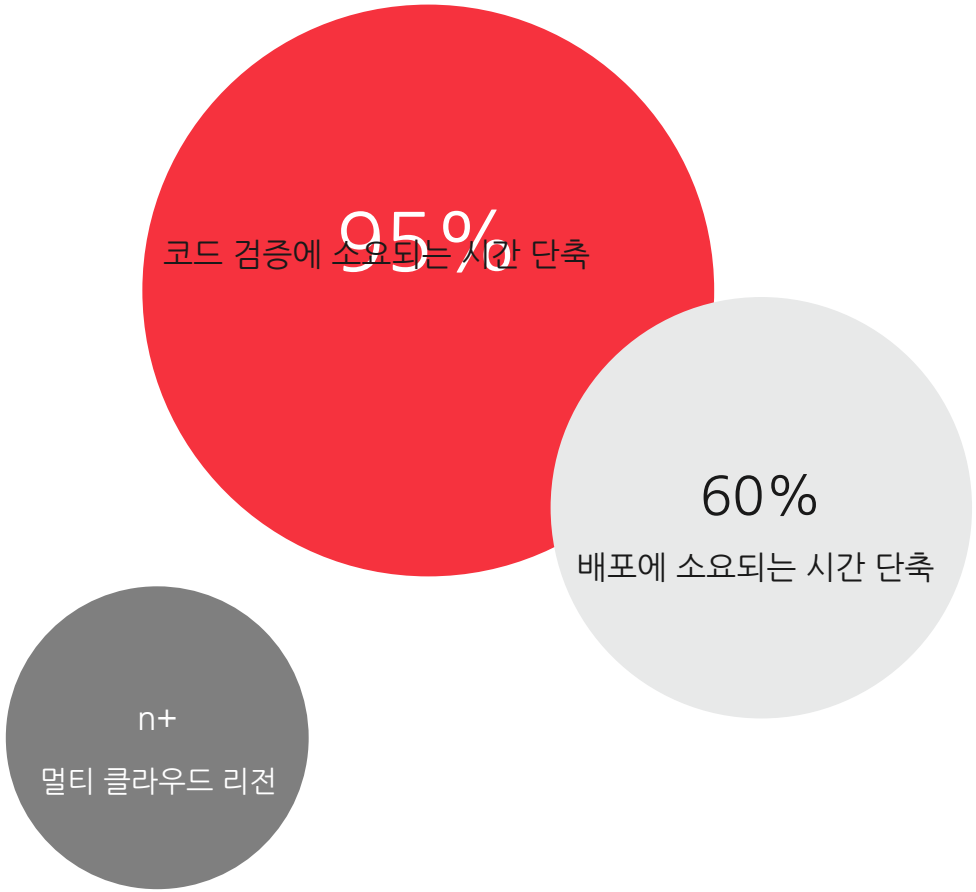
GitHub Actions 도입




```
72
73  ----------
74  ID : /etc/neutron/neutron.conf
75  NAME : [file][managed] /etc/neutron/neutron.conf
76  ▶ details...
96
97  ----------------------------------------
98  Summary
99  ----------------------------------------
100 # of minions targeted: 2
101 # of minions changed: 2
102 # of minions that did not return: 0
103 # of minions with warnings: 0
104 # of minions with errors: 0
105 ----------------------------------------
```

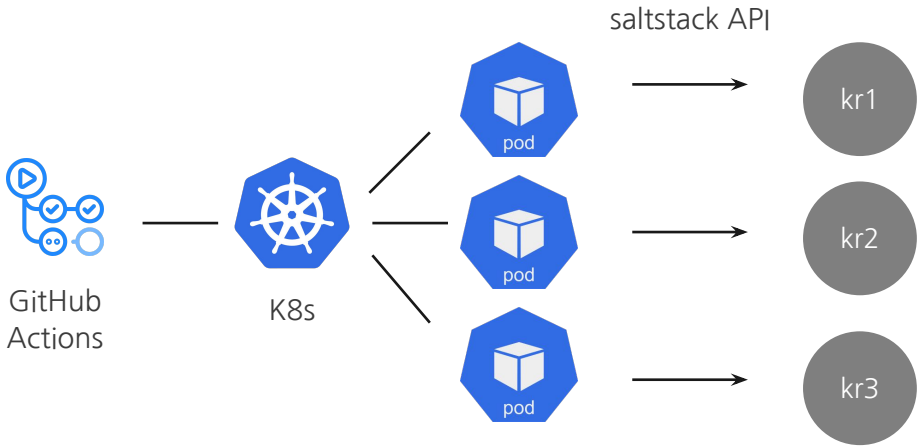
```
73  ----------
74  ID : /etc/neutron/neutron.conf
75  NAME : [file][managed] /etc/neutron/neutron.conf
76  ▼details...
77        Change-1:
78            comment : The file /etc/neutron/neutron.conf is set to be changed
79            Note: No changes made, actual changes may
80            be different due to other states.
81            Target: all hosts
82            Changes:
83                -----
84                diff:
85                    ---
86                    +++
87                    @@ -148,7 +148,7 @@
88
89                     [database]
90                     connection = ***neutron-
91                    -max_pool_size = 50
92                    +max_pool_size = 150
93                     max_overflow = 100
94                     pool_timeout = 10
95
```

GitHub Actions 도입 후 개선 효과



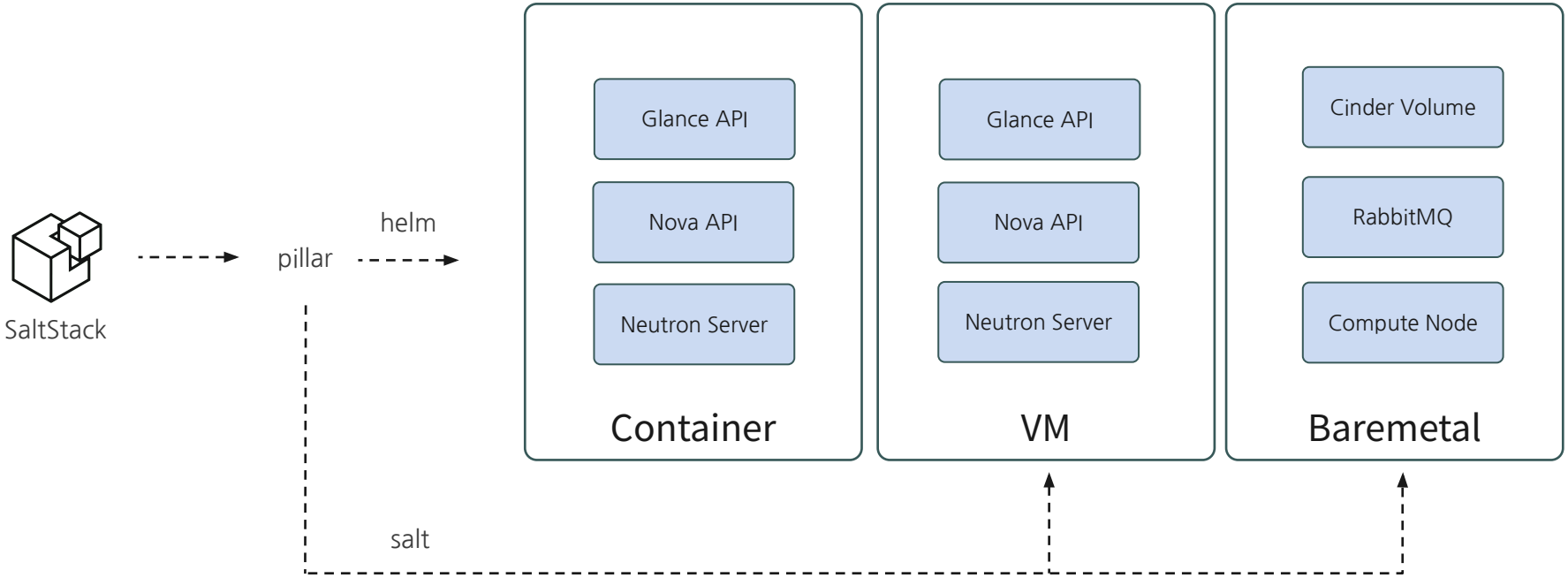
한계 및 보완점

요청에 따라 자동으로 Runner를 확장, 축소할 수 있도록 컨테이너화 (Actions Runner Controller, ARC)



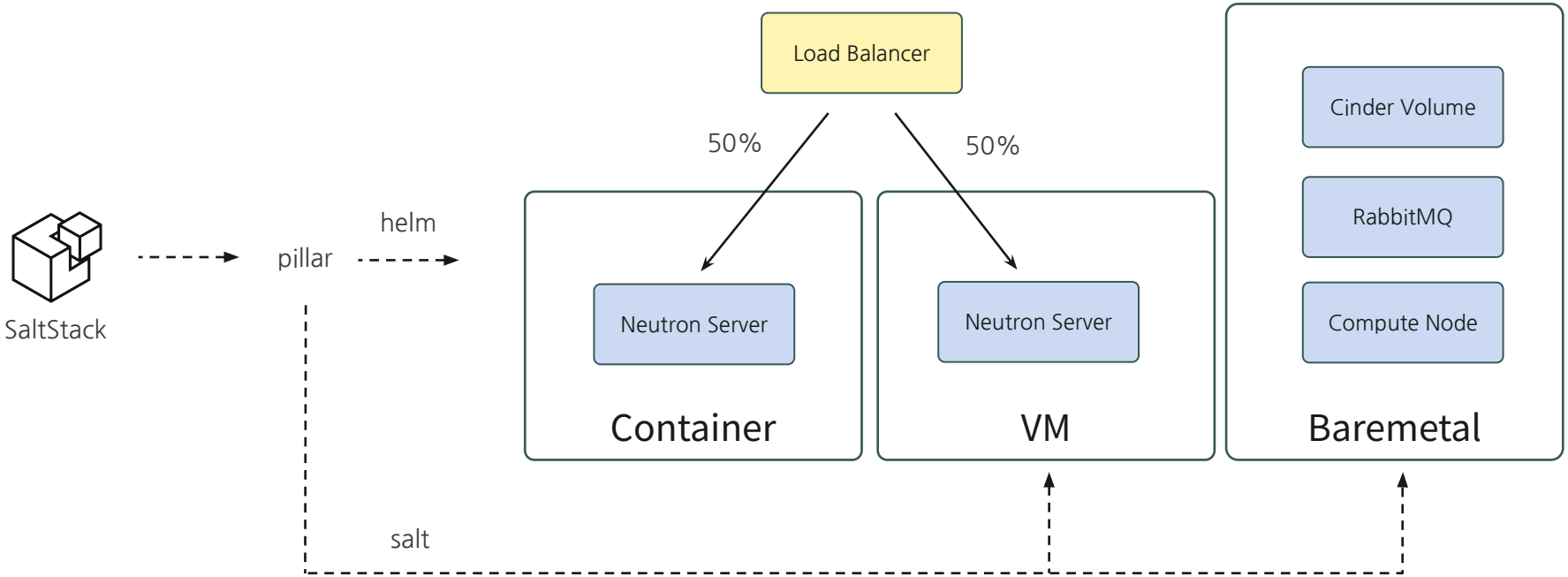
앞으로의 여정

VM, 베어메탈, 컨테이너가 공존하는 배포 형태를 관리



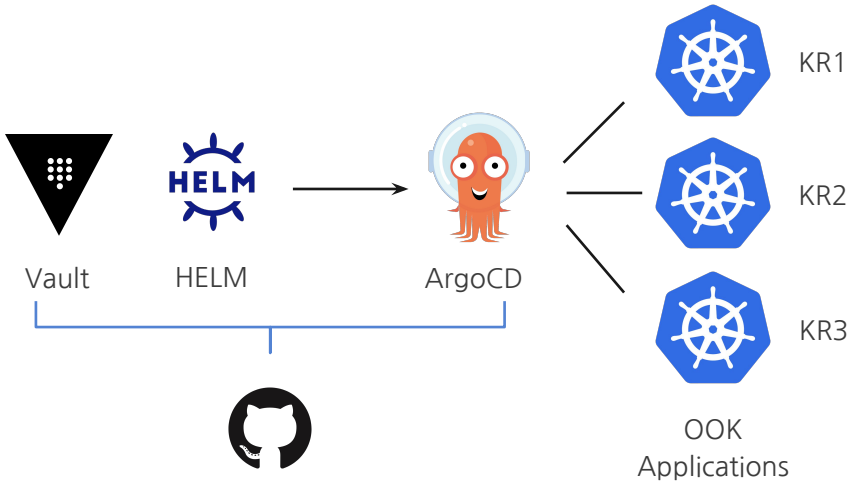
앞으로의 여정

VM, 컨테이너의 투입 비율을 조정하면서 점차 컨테이너 환경으로 전환



앞으로의 여정

Openstack On Kubernetes (OOK) 배포를 위한 GitOps 기반의 운영



(마무리) SaltStack을 이용한 Multi Region 배포 전략

pillar로 리전마다 사용하는
고정값 / 시크릿 값 관리

복잡한 워크플로우 실행을
위한 salt orchestration

1개 컴포넌트의 배포코드는
단 하나

자동화 / 여러 리전 동시 배포를
위한 GitHub Actions

리전마다 달라지는 기능을
쉽게 제어하는 features.sls

vm/pm/container 등 다양한
인프라 환경 대응을
위한 배포 코드

(마무리) SaltStack을 이용한 Multi Region 배포 전략

리전과 기능이 빠르게 늘어나더라도

유연하게 대응할 수 있게 되어

NHN Cloud 인프라 서비스를 안정적으로 확장하고 있다

위한 배포 코드

사람을 구합니다

[NHN Cloud] 시스템 엔지니어링

Tech | Infra E... | 경력 | 정규 | 채용시까지

지원하기

NHN Cloud

NHN Cloud는 대한민국을 대표하는 클라우드 서비스로서 데이터 센터부터 플랫폼, 애플리케이션 서비스에 이르기까지 고객이 필요한 모든 클라우드 서비스를 제공 하는 클라우드 서비스 제공사(CSP)입니다. 탄탄한 기술력으로 폭발적인 성장을 이뤄온 NHN Cloud는 이제 국내를 넘어 글로벌 테크 기업으로의 성장을 목표로 하고 있습니다.

이런 업무를 해요 (주요 업무)

- 오픈스택 기반 NHN Cloud IaaS 배포 및 운영
- 시스템/서비스 모니터링 및 트러블슈팅, 모니터링 및 운영도구 구축/개발/운영

이런 분들을 찾고 있어요 (자격 요건)

- 시스템 엔지니어링 경력을 5년 이상 보유하신 분
- 리눅스 OS 설치 및 시스템/서비스 운영, 트러블슈팅, 서버/시스템 모니터링 구축/운영 경험 보유하신 분
- IaC 도구, CI/CD 도구를 이용한 시스템 운영 자동화 경험, 프로그래밍 개발 스킬 (Bash, Python, Java 등) 보유하신 분

End of Document

감사합니다

NHN