

토이 프로젝트 보고서 배포 전략 성능 시뮬레이션

카카오 클라우드 엔지니어 4기 윤순상

CONTENTS

카카오 클라우드 스쿨 엔지니어 4기
토이 프로젝트 결과 보고서

01

프로젝트 개요

02

요약 보고서

03

현황, 문제점, 개선사항

04

실시내용

05

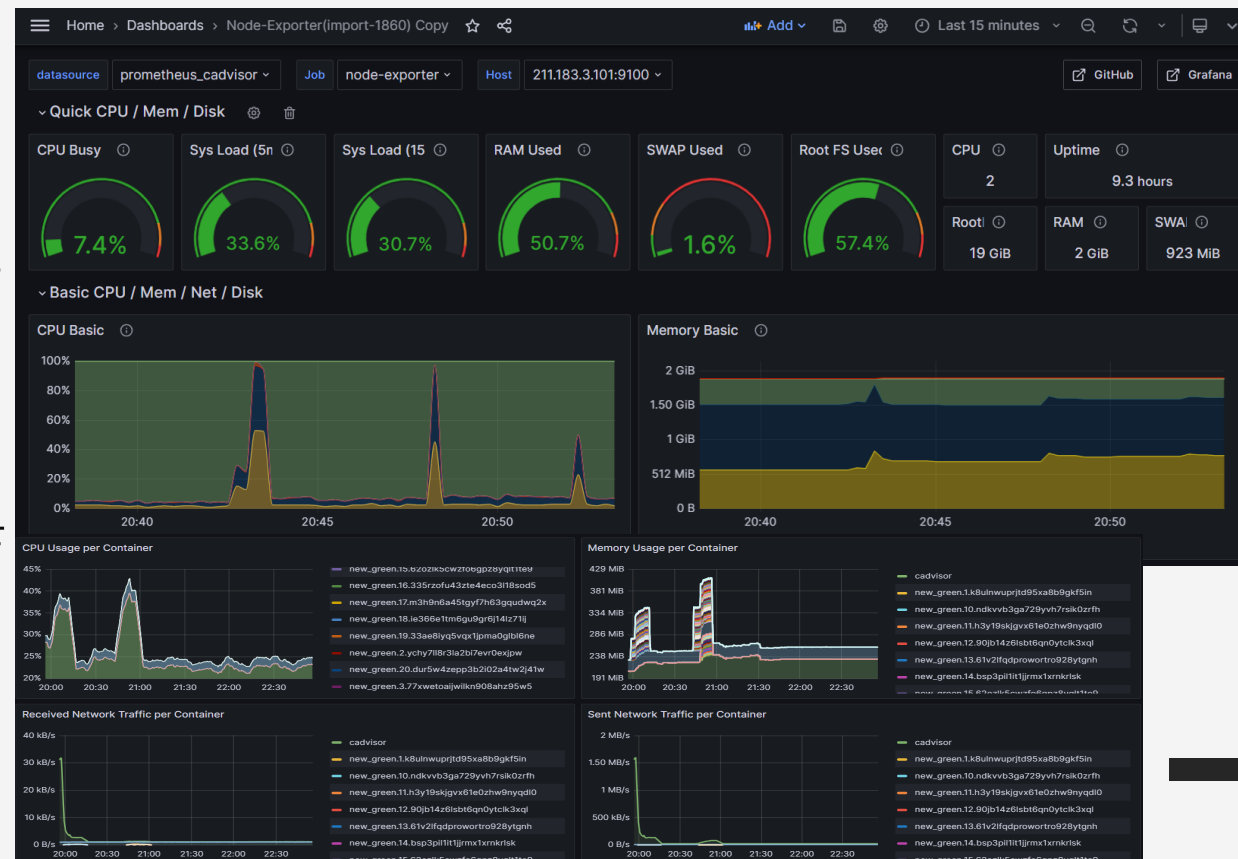
Kubernetes

01 프로젝트 개요

■ 프로젝트 개요

- 프로젝트 / 배포 전략 성능 시뮬레이션
- 진행자 / 개인 프로젝트 - 윤순상
- 특징 / Blue/Green, Canary 배포에 대해 비교 분석
- 설명 / 서비스를 제공하기 위해 환경 구축을 하는데 있어서 배포 전략을 고려하는 것은 매우 중요하다. 주어진 환경에서 알맞은 배포 전략을 찾을 때 비용 책정과 예측이 중요하다. 먼저 간단한 프로토타입 형태로 배포 전략을 구성하여 서비스 배포를 시뮬레이션 해보고 기본 Metric 정보를 수집 및 분석한다.

■ 프로젝트 결과물



02 요약 보고서

■ 프로젝트 개요

Blue/Green, Canary 배포 전략의 성능 비교

■ 프로젝트 과제

1. 로드 밸런싱을 어떻게 해야 하는가?
2. CAdvisor, Node-Exporter의 설치 - Metrics 정보
3. Prometheus와 Grafana 연동

■ 프로젝트 문제 및 효과

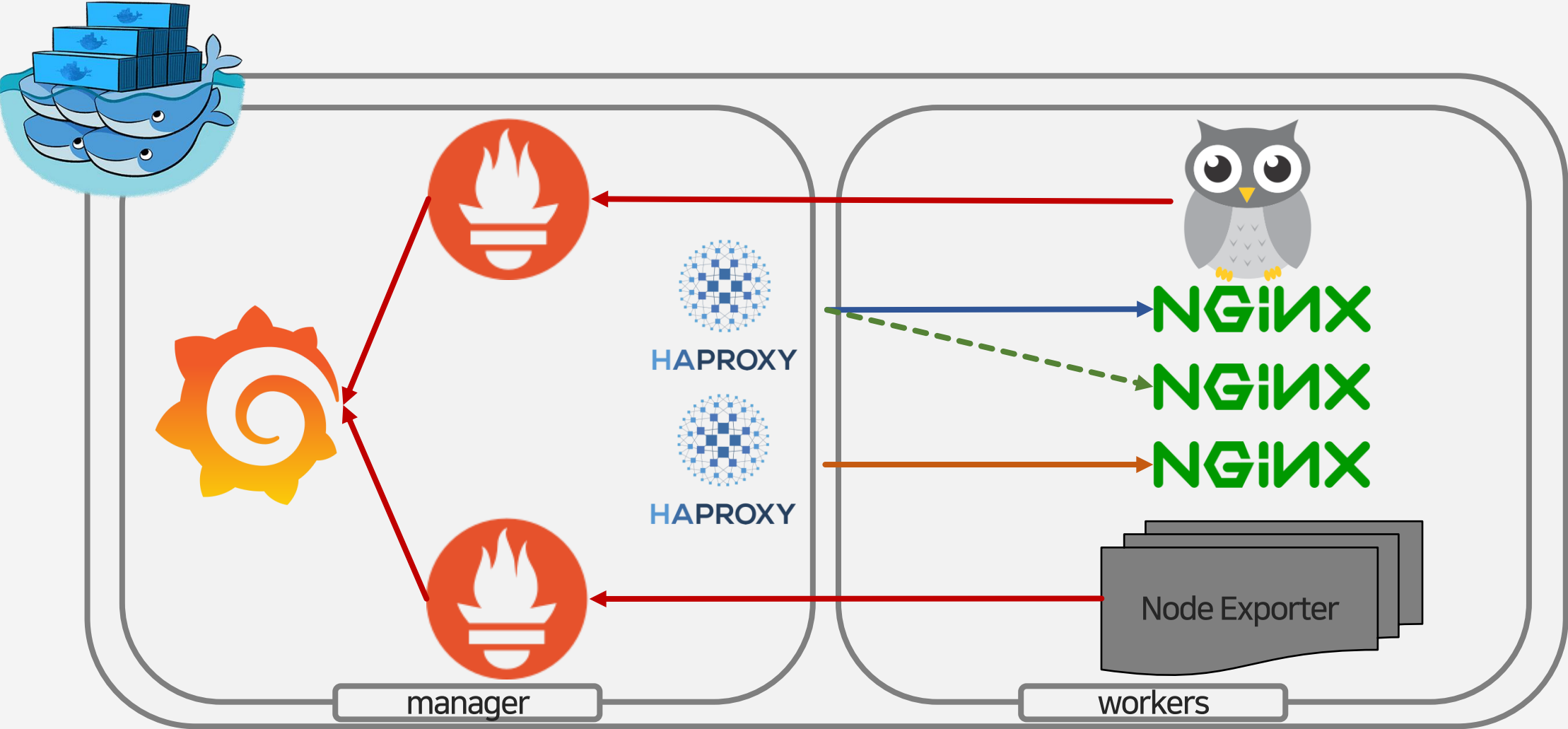
1. 다양한 배포 전략이 존재
2. 서비스를 구성할 때 어떤 배포 전략을 채택하여 환경을 구성 해야 할까?
3. 간단한 프로토타입 배포 전략을 구성하여 대략적인 서비스의 비용을 시뮬레이션 하기
4. 주어진 환경에 맞는 배포 전략을 선택하는데 근거로 활용할 수 있다.

■ 향후 과제

1. 그 밖에 다양한 배포 전략에 맞춰 환경을 구성하고 테스트하기
2. LB의 로드 밸런싱 업데이트 속도를 향상 시킬 수 있을까?
3. 배포 전략별로 어떻게 비용이 산정되는지 구성하기

요약 보고서

카카오 클라우드 스쿨 엔지니어 4기
토이 프로젝트 결과 보고서



03 현황, 문제점, 개선사항

현황, 문제점, 개선사항

로드 밸런싱을 어떻게 해야 하는가?

카카오 클라우드 스쿨 엔지니어 4기
토이 프로젝트 결과 보고서

■ 동적 로드 밸런싱

- Docker-swarm에서 dockercloud/haproxy 이미지를 사용하면 모든 Docker 웹 서비스의 로드 밸런싱을 쉽게 구성할 수 있다.
- 생성되는 모든 웹 서비스에 대해서 동적으로 로드밸런싱 정보를 업데이트한다.
- 두 서비스가 공존하는 Blue/Green에서 두 서비스에게 모두 RoundRobin으로 부하 분산을 하게 된다.
- Canary의 경우 로드 밸런싱의 비율을 조절할 수 있어야 하지만 동적으로 할 경우 불가능하다.

■ 정적 로드 밸런싱

- HAproxy 설정파일을 작성해 놓고 이를 토대로 로드 밸런싱을 구성하는 방법이다.
- 즉, 나만의 HAproxy 도커 이미지를 작성해야 한다.
- 이렇게 하면 원하는 대로 Blue/Green 배포 테스트에서 로드 밸런싱 방향을 변경할 수 있고
- Canary 배포의 경우 로드 밸런싱 비율을 자유롭게 구성할 수 있게 된다.
- 또한 HAproxy는 도커 서비스명으로 부하 분산을 구현할 수 있어 매우 간결하다.

현황, 문제점, 개선사항

로드 밸런싱을 어떻게 해야 하는가?

카카오 클라우드 스쿨 엔지니어 4기
토이 프로젝트 결과 보고서

■ os 컨테이너

- 처음에는 centos:7로 구성한 os 컨테이너를 제작했다.
- 그러나 이 경우 이미지의 크기가 haproxy 컨테이너 보다 5배 정도 무겁다.
- 또한 Canary 배포의 경우 부하 분산 비율을 조정할 때 새로운 haproxy.cfg를 가지고 도커 이미지를 만들 때 많은 시간이 소요된다.

■ haproxy 컨테이너

- HAproxy 이미지를 가지고 구성하면 매우 가벼운 haproxy 도커 이미지를 생성할 수 있다.
- 방식도 매우 간편하고 빠르게 생성하고 배포할 수 있다.

```
andrewyss/canary    default  4840dd92eff5  8 minutes ago  122MB
andrewyss/proxy     green   8e487e822d8e  20 hours ago   584MB
andrewyss/proxy     blue    cffd900d8b72  20 hours ago   584MB
haproxy:latest로 더 간편하게 만드는 것이 가능하다. 용량도 훨씬 작다!
```

04 실시내용

■ 네트워크 및 볼륨 설정

```
● user1@manager:~/toy$ docker network ls | grep overlay
hbb1l2o1x0lz    bg_deploy      overlay        swarm
pghvriasc8lo    cn_deploy      overlay        swarm
1biazkdppjh6    ingress        overlay        swarm
316dmwke4blk    monitor        overlay        swarm

● user1@manager:~/toy$ docker volume ls
DRIVER    VOLUME NAME
local     grafana_vol
local     prometheus_cad
local     prometheus_node
```

■ HAproxy Docker image

[Dockerfile] - 용도에 맞게 일부분만 수정
FROM haproxy:latest

COPY haproxy_green.cfg
/usr/local/etc/haproxy/haproxy.cfg

EXPOSE 80
EXPOSE 8080

CMD ["haproxy", "-f",
"/usr/local/etc/haproxy/haproxy.cfg"]

[haproxy.cfg] - blue/green
... (일부생략)
frontend main
bind *:80
default_backend blue_backend

backend blue_backend
balance roundrobin
server web_blue **web_blue:80** check

[haproxy.cfg] - canary
... (일부생략)
frontend main
bind *:80
default_backend canary_backend

backend canary_backend
balance roundrobin
server web2_canary
web2_canary:80 check **weight 1**
server web2_normal
web2_normal:80 check **weight 9**

andrewyss/proxy:blue

andrewyss/proxy:green

andrewyss/canary:default

■ blue.yml, green.yml

version: "3.7"

services:

blue:

image: andrewyss/deploytest:blue

deploy:

replicas: 20

placement:

constraints: [node.role != manager]

environment:

SERVICE_PORTS: 80

networks:

- bg_deploy

proxy:

image: andrewyss/proxy:blue

depends_on:

- blue

- green

ports:

- "80:80"

- "8080:8080"

networks:

- bg_deploy

deploy:

mode: global

placement:

constraints: [node.role == manager]

networks:

bg_deploy:

external: true

...(일부생략)

green:

image: andrewyss/deploytest:green

deploy:

replicas: 20

placement:

constraints: [node.role != manager]

environment:

SERVICE_PORTS: 80

networks:

- bg_deploy

...(일부생략)

new

web

■ HAproxy 부하 분산 모니터링 – Blue/Green

HAProxy version 2.9.4-4e071ad, released 2024/01/31

Statistics Report for pid 8

> General process information

pid = 8 (process #1, nbproc = 1, nbthread = 4)

uptime = 0d 0h03m48s; warnings = 1

system limits: memmax = unlimited; ulimit-n = 1048576

maxsock = 1048576; maxconn = 524270; reached = 0; maxpipes = 0

current conns = 2; current pipes = 0/0; conn rate = 1/sec; bit rate = 615.607 kbps

Running tasks: 0/23 (0 niced); idle = 100 %

active UP

active UP, going down

active DOWN, going up

active or backup DOWN

active or backup DOWN for maintenance (MAINT)

active or backup SOFT STOPPED for maintenance

backup UP

backup UP, going down

backup DOWN, going up

not checked

Display option:

Scope :

Hide "DOWN" servers

Refresh now

CSV export

JSON export (schema)

External resources:

Primary site

Updates (v2.9)

Online manual

stats

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server									
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle	
Frontend				1	2	-	2	4		524 270	4		2 920	111 776	0	0	1					OPEN									
Backend	0	0		0	0		0	0		52 427	0	0s	2 920	111 776	0	0		0	0	0	0	3m48s UP		0/0	0	0			0		

main

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server									
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle	
Frontend				0	2	-	0	4		524 270	2		7 800	2 392	0	0	1					OPEN									

green_backend

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server									
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle	
new_green	0	0	-	0	7		0	1		-	13	3m21s	7 800	2 392	0	0		0	0	0	0	3m48s UP	L4OK in 1ms	1/1	Y	-	0	0	0s	-	
Backend	0	0		0	7		0	1		52 427	13	3m21s	7 800	2 392	0	0		0	0	0	0	3m48s UP		1/1	1	0		0	0s		

14

■ HAproxy 부하 분산 모니터링 - Canary

HAProxy version 2.9.4-4e071ad, released 2024/01/31

Statistics Report for pid 7

> General process information

pid = 7 (process #1, nbproc = 1, nbthread = 4)
uptime = 0d 0h09m27s; warnings = 1
system limits: memmax = unlimited; ulimit-n = 1048575
maxsock = 1048575; maxconn = 524269; reached = 0; maxpipes = 0
current conns = 2; current pipes = 0/0; conn rate = 1/sec; bit rate = 423.910 kbps
Running tasks: 0/26 (0 niced); idle = 100 %

- active UP

active UP, going down

active DOWN, going up

active or backup DOWN

active or backup DOWN for maintenance (MAINT)

active or backup SOFT STOPPED for maintenance
- backup UP

backup UP, going down

backup DOWN, going up

not checked

Note: "NOLB"/"DRAIN" = UP with load-balancing disabled.

Display option:

Scope :

- Hide 'DOWN' servers
- Refresh now
- CSV export
- JSON export (schema)

- External resources:
- Primary site
 - Updates (v2.9)
 - Online manual

stats																															
	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server									
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle	
Frontend				1	2	-	2	4	524 269	19			52 469	2 171 488	0	0	9					OPEN									
Backend	0	0		0	1		0	1	52 427	1	0	0s	52 469	2 171 488	0	0		1	0	0	0	9m27s UP		0/0	0	0		0			

main	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server									
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle	
Frontend				0	2	-	0	3	524 269	10			42 209	13 085	0	0	6					OPEN									

canary_backend	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server									
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle	
web2_canary	0	0	-	0	1		0	1	-	7	7	8s	4 221	2 204		0		0	0	0	0	9m27s UP	L4OK in 1ms	1/1	Y	-	0	0	0s	-	
web2_normal	0	0	-	0	7		0	1	-	63	63	8s	37 988	10 881		0		0	0	0	0	9m27s UP	L4OK in 1ms	9/9	Y	-	0	0	0s	-	
Backend	0	0		0	8		0	1		52 427	70	70	8s	42 209	13 085	0	0		0	0	0	9m27s UP		10/10	2	0		0	0	0s	

■ canary.yml

```
version: "3.7"

services:
  canary:
    image: andrewyss/deploytest:cnb
    deploy:
      replicas: 5
      placement:
        constraints:
          [node.labels.deployment.type == canary]
      environment:
        SERVICE_PORTS: 80
      networks:
        - cn_deploy
    networks:
      cn_deploy:
        external: true

  normal:
    image: andrewyss/deploytest:cnb
    deploy:
      replicas: 15
      placement:
        constraints:
          [node.labels.deployment.type == normal]
      environment:
        SERVICE_PORTS: 80
      networks:
        - cn_deploy

  proxy:
    image: andrewyss/canary:default
    depends_on:
      - normal
      - canary
    # Blue/Green 서비스와 같이 올리기 때문에 호스트 포트 연결은 다르게 해주었다.
    ports:
      - "81:80"
      - "8081:8080"
    networks:
      - cn_deploy
    deploy:
      mode: global
      placement:
        constraints:
          - "node.role == manager"
```

web2

■ index.yml

KAKAO Cloud Engineer 4 Docker Project

Blue/Green Deployment Test [OLD]

KAKAO Cloud Engineer 4 Docker Project

Blue/Green Deployment Test [NEW]

KAKAO Cloud Engineer 4 Docker Project

Canary Deployment Test [OLD]

KAKAO Cloud Engineer 4 Docker Project

Canary Deployment Test [NEW]

■ docker-compose.yml : Cadvisor, Node-Exporter

```
version: '3.8'
services:
  node-exporter:
    image: prom/node-exporter:latest
    container_name: node-exporter
    restart: always
    volumes:
      - /proc:/host/proc:ro
      - /sys:/host/sys:ro
      - /:/rootfs:ro
    command:
      - '--path.procfs=/host/proc'
      - '--path.rootfs=/rootfs'
      - '--path.sysfs=/host/sys'
      - '--collector.filesystem.mount-points-exclude=^/(sys|proc|dev|host|etc)($|/)'
    ports:
      - 9100:9100
```

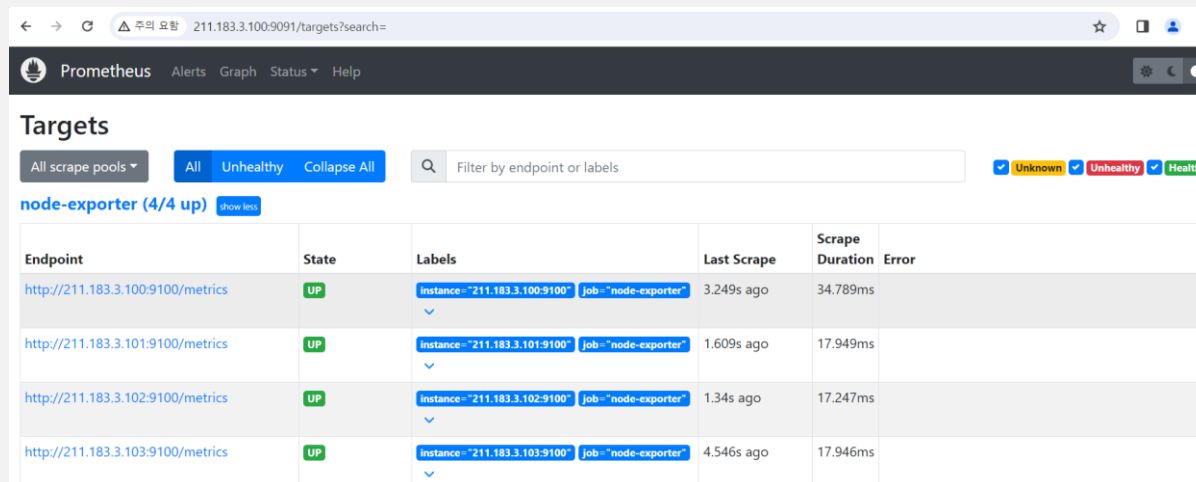
```
cad:
  image: google/cadvisor:latest
  container_name: cadvisor
  restart: always
  volumes:
    - /:/rootfs:ro
    - /var/run:/var/run:rw
    - /sys:/sys:ro
    - /var/lib/docker:/var/lib/docker:ro
  ports:
    - "8000:8080"
```

[prometheus.yml] - 프로메테우스 생성 시 필요
Node-Exporter용은 따로 또 작성 port는 9100

```
global:
  scrape_interval: 5s
  external_labels:
    monitor: 'my-monitor'
scrape_configs:
  - job_name: 'CAdvisor'
    static_configs:
      - targets:
        - '211.183.3.101:8000'
        - '211.183.3.102:8000'
        - '211.183.3.103:8000'
```

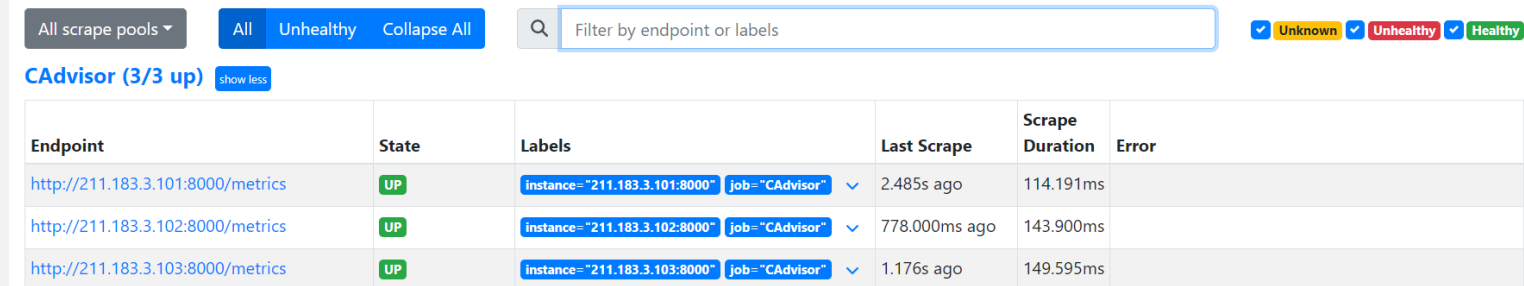
각 node에 별도로 생성

■ prometheus targets



The screenshot shows the Prometheus web interface. The browser address bar displays '211.183.3.100:9091/targets?search='. The Prometheus header includes navigation links for Alerts, Graph, Status, and Help. The 'Targets' section is active, showing a list of scrape pools. The 'node-exporter (4/4 up)' pool is selected, displaying a table of targets. All targets are in an 'UP' state. The table columns are Endpoint, State, Labels, Last Scrape, Scrape Duration, and Error.

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://211.183.3.100:9100/metrics	UP	instance="211.183.3.100:9100" job="node-exporter"	3.249s ago	34.789ms	
http://211.183.3.101:9100/metrics	UP	instance="211.183.3.101:9100" job="node-exporter"	1.609s ago	17.949ms	
http://211.183.3.102:9100/metrics	UP	instance="211.183.3.102:9100" job="node-exporter"	1.34s ago	17.247ms	
http://211.183.3.103:9100/metrics	UP	instance="211.183.3.103:9100" job="node-exporter"	4.546s ago	17.946ms	



The screenshot shows the Prometheus web interface for the 'CADvisor (3/3 up)' scrape pool. The interface includes the same navigation and filtering options as the previous screenshot. The table below lists three targets, all of which are in an 'UP' state.

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://211.183.3.101:8000/metrics	UP	instance="211.183.3.101:8000" job="CADvisor"	2.485s ago	114.191ms	
http://211.183.3.102:8000/metrics	UP	instance="211.183.3.102:8000" job="CADvisor"	778.000ms ago	143.900ms	
http://211.183.3.103:8000/metrics	UP	instance="211.183.3.103:8000" job="CADvisor"	1.176s ago	149.595ms	

■ prometheus, grafana

version: "3.7"

services:

metheus_node:

image: prom/prometheus:latest

ports:

- "9091:9090"

deploy:

mode: global

placement:

constraints: [node.role == manager]

networks:

- bg_deploy

- cn_deploy

- monitor

volumes:

- prometheus_node:/prometheus

- ./conf_node/./etc/prometheus/

restart: always

metheus_cad:

image: prom/prometheus:latest

ports:

- "9092:9090"

deploy:

mode: global

placement:

constraints: [node.role == manager]

networks:

- bg_deploy

- cn_deploy

- monitor

volumes:

- prometheus_cad:/prometheus

- ./conf_cad/./etc/prometheus/

restart: always

grafana:

image: grafana/grafana:latest

depends_on:

- metheus_node

- metheus_cad

ports:

- "3000:3000"

deploy:

mode: global

placement:

constraints: [node.role == manager]

environment:

- GF_SECURITY_ADMIN_USER=admin

-

GF_SECURITY_ADMIN_PASSWORD=test123

networks:

- monitor

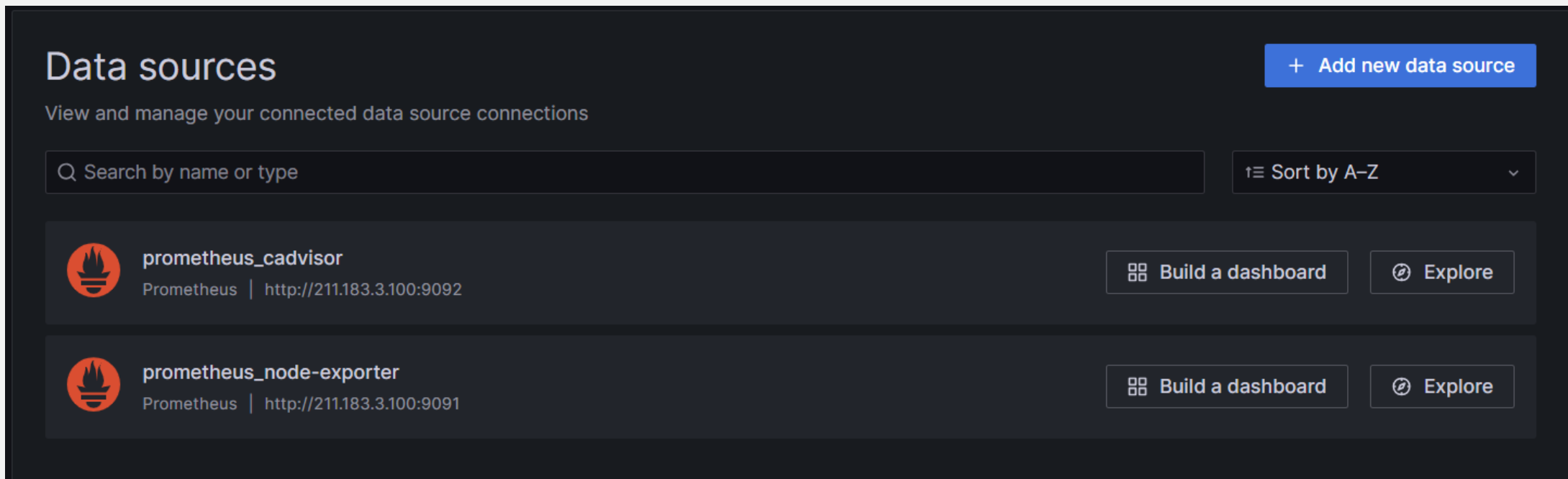
volumes:

- grafana_vol:/var/lib/grafana

...(이하 네트워크, 볼륨 선언 생략)

monitor

■ Grafana Prometheus 등록

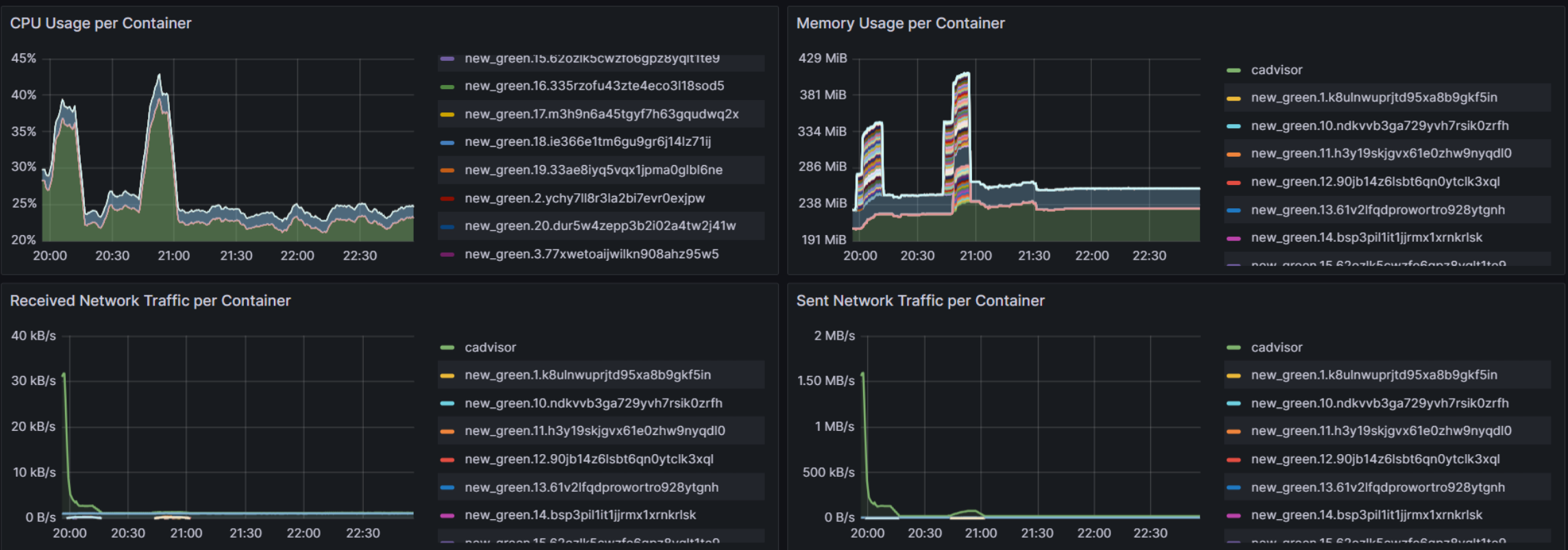


The screenshot displays the 'Data sources' configuration page in Grafana. At the top right, there is a blue button labeled '+ Add new data source'. Below the title, a subtitle reads 'View and manage your connected data source connections'. A search bar with the placeholder 'Q Search by name or type' and a sort dropdown menu set to 'Sort by A-Z' are located below the subtitle. Two data sources are listed:

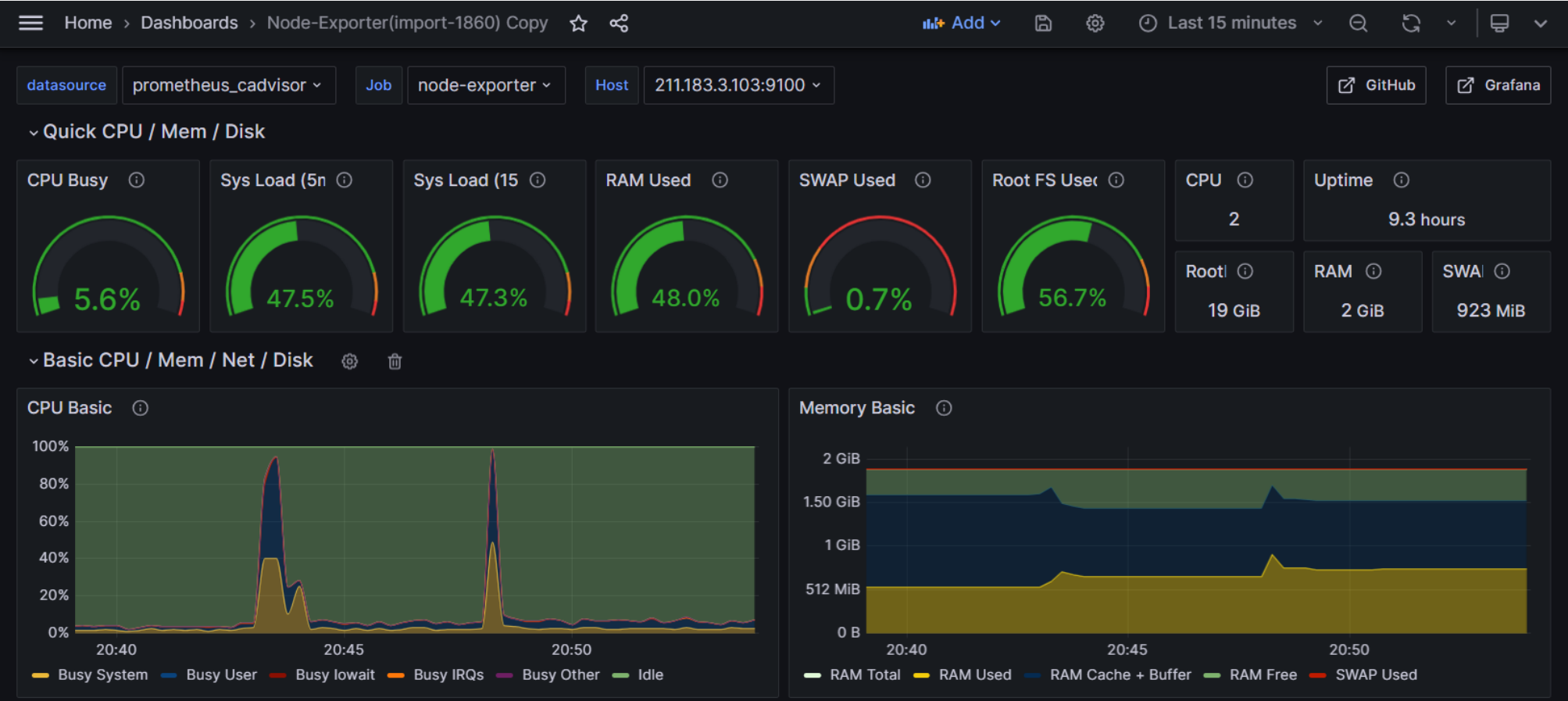
- prometheus_cadvisor**: A Prometheus data source with the URL `http://211.183.3.100:9092`. It includes buttons for 'Build a dashboard' and 'Explore'.
- prometheus_node-exporter**: A Prometheus data source with the URL `http://211.183.3.100:9091`. It also includes buttons for 'Build a dashboard' and 'Explore'.

프로젝트 지표

카카오 클라우드 스쿨 엔지니어 4기
토이 프로젝트 결과 보고서



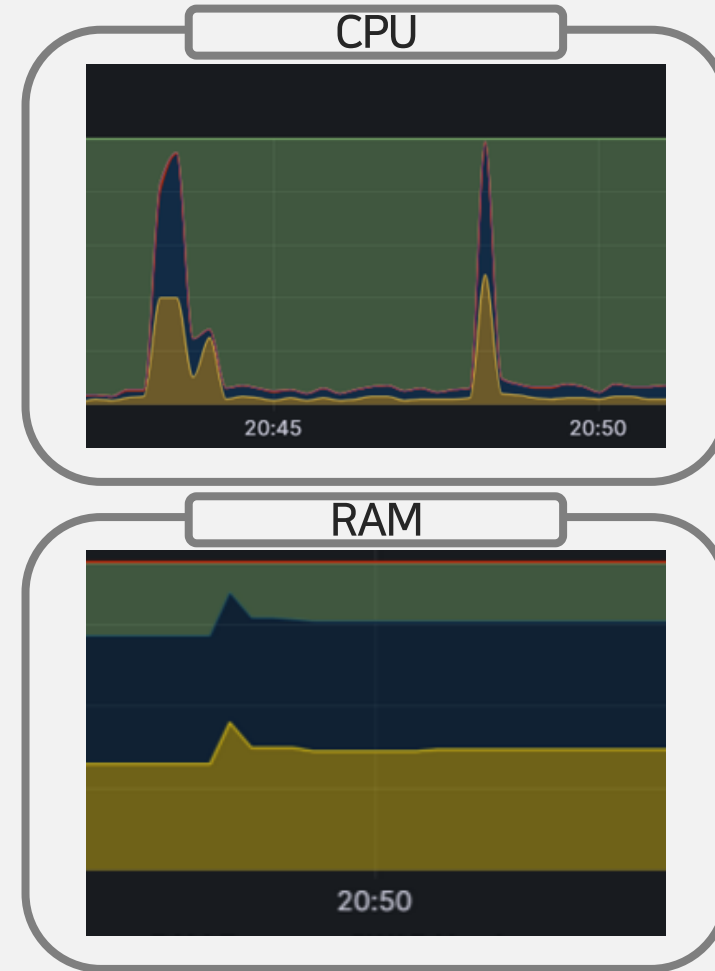
- Grafana 공유 대시보드 활용 -



- Grafana 공유 대시보드 활용 -

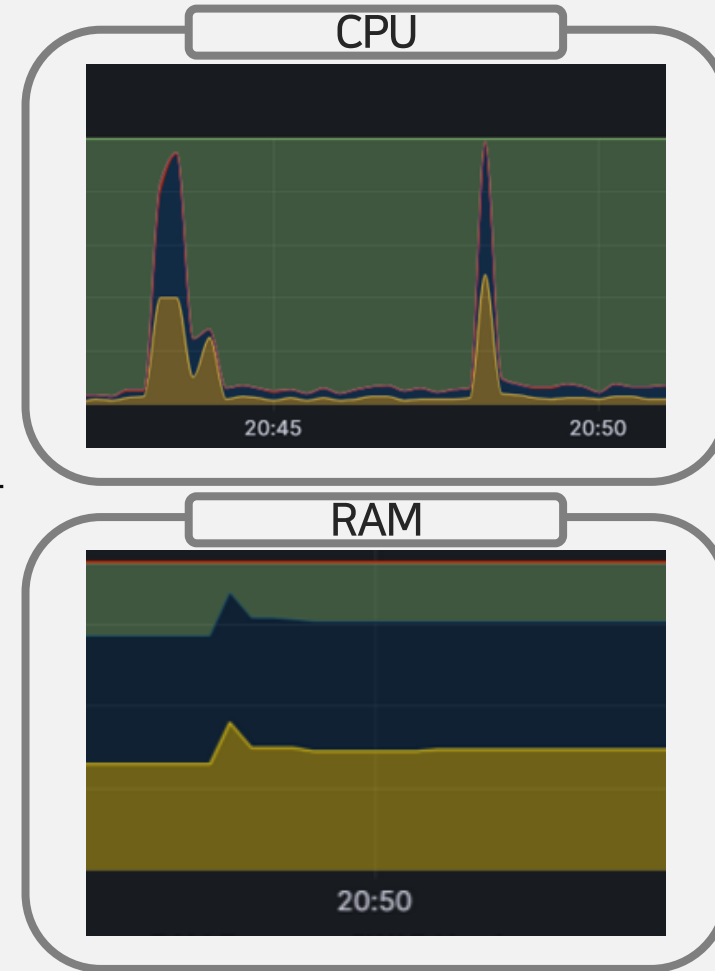
■ Blue/Green Deployment

- 확실히 Blue 컨테이너와 같은 크기의 Green 컨테이너를 생성하기 때문에 리소스를 많이 잡아먹고 Metric그래프가 갑자기 올라가는 모습을 볼 수 있다.
- 만약 서비스 환경의 스펙이 좋지 못하다면 Blue/Green 배포 전략은 서버에 부담이 될 수 있다.
- 하지만 두 서비스 컨테이너를 동시에 Running하기 때문에 이전 버전의 복구가 훨씬 빠르고 간편하다.
- 그러나 만약 roll back을 해야 할 상황이 발생한다면 리소스를 낭비하고 서버에 부담을 주는 것은 사실이다.



■ Canary Deployment

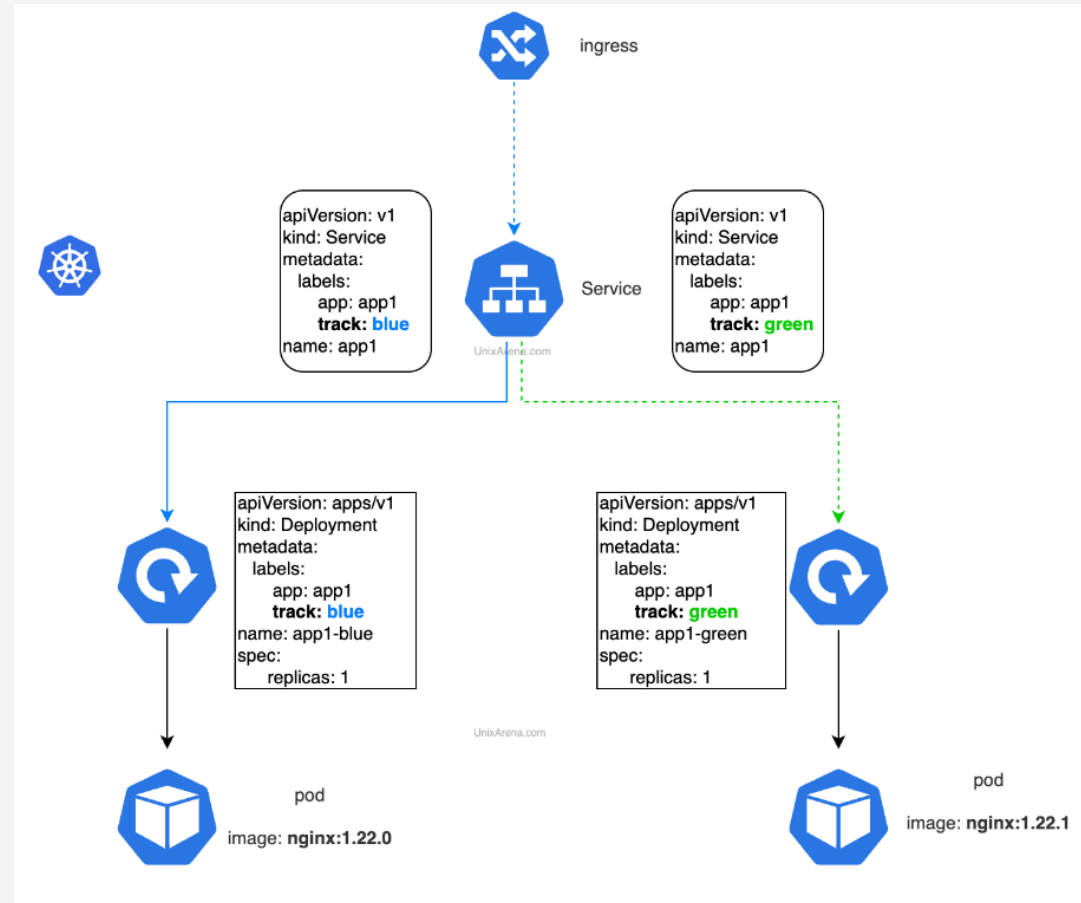
- Canary 배포를 진행할 때는 Metric 그래프의 변화가 거의 탐지 되지 않는다.
- 노드들의 특정 labels에 한해서 업데이트를 진행 하는 것이기 때문에 사용되는 리소스는 그대로라고 볼 수 있다.
- 그러나 만약 업데이트 도중 과도한 traffic이 발생한다면 나머지 웹 서버의 부담이 크게 늘어날 수 있다.
- 또한 Canary는 부하 분산의 비율로 분산을 조절하지만 결과적으로 두 서비스가 동시에 운영이 되는 것이기 때문에 이에 대한 조치가 필요하다.
- 누군가는 new버전을 경험하고 누군가는 old버전을 경험
- 따라서 이에 대한 연구 및 장치가 필요하고 다양한 방식이 존재한다.



05 Kubernetes

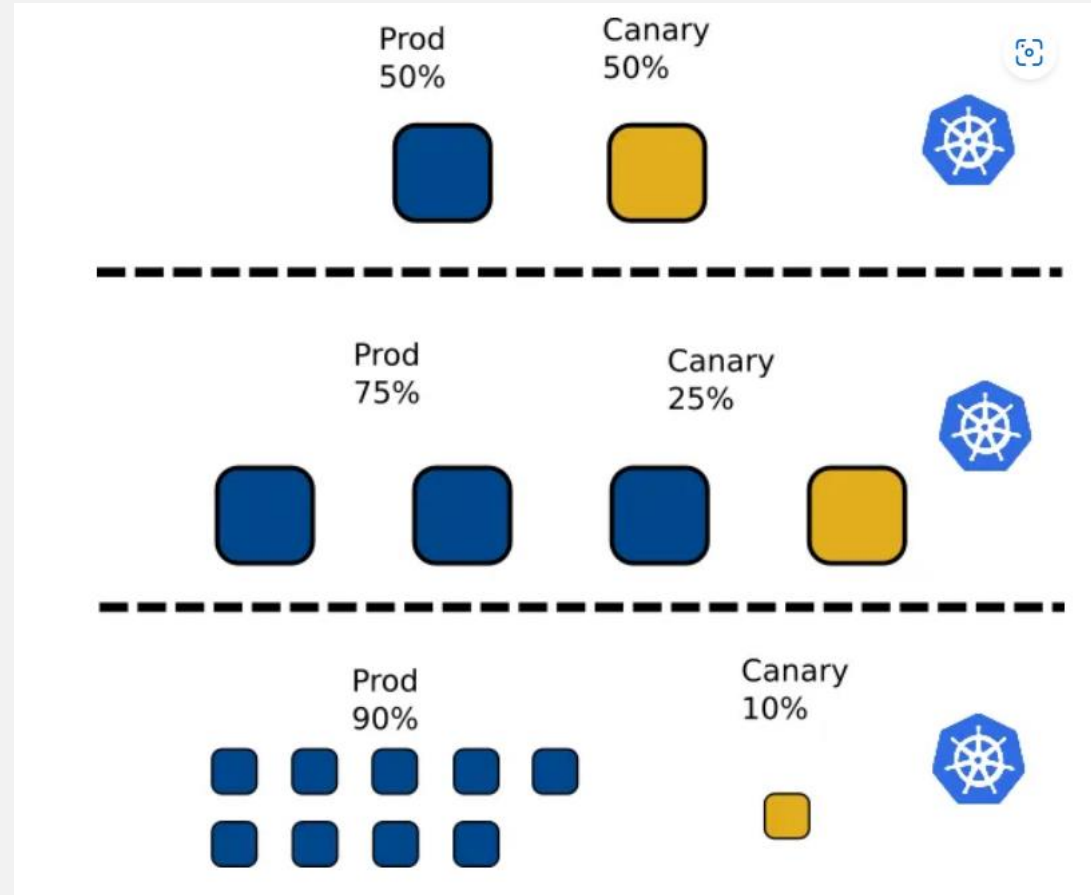
■ Blue/Green Deployment

- k8s에서는 2가지 작업 환경을 구성한다.
- 현재 서비스 중인 blue환경과 R&D를 하는 green환경을 구성한다.
- Green환경에서 테스트 과정을 끝나치면 로드 밸런싱을 변경하면 되는데 k8s에서는 selector가 이를 담당한다.
- Selector의 정보만 업데이트하면 바로 green 환경으로 로드 밸런싱이 업데이트 되어 매우 쉽게 Blue / Green 배포 전략을 수행할 수 있다.



■ Canary Deployment

- Canary 배포 전략은 고려해야할 사항이 많아 매우 복잡한 배포 전략 중 하나다.
- 누구에게 v2를 경험하게 할지, 이전버전으로 되돌리기 UX/UI가 필요한가 등등...
- 그래서 docker-swarm 환경에서는 구현하기에는 상당히 어려울 수 있다. 하지만 k8s에서는 좀 더 쉽게 Canary 배포를 수행할 수 있다.
- k8s는 pod라는 단위의 서비스로 구성되고 canary pod를 늘려가면서 로드 밸런싱의 비율을 점진적으로 증가시킬 수 있다.



■ 참고 자료

- [깃허브](#) - 향후 모든 내용 업로드 예정
- [프로메테우스 그래파나 구성](#)
- [Node-Exporter \(공유 대시 보드 출처\)](#)
- [Grafana - CAdvisor \(공유 대시보드 출처\)](#)
- [k8s - Blue/Green Deployment](#)
- [k8s - Canary Deployment](#)

T h a n k y o u

카카오 클라우드
엔지니어 4기 - 윤순상
