Deploy

AWS Cloud Club

DDWU ACC Crew

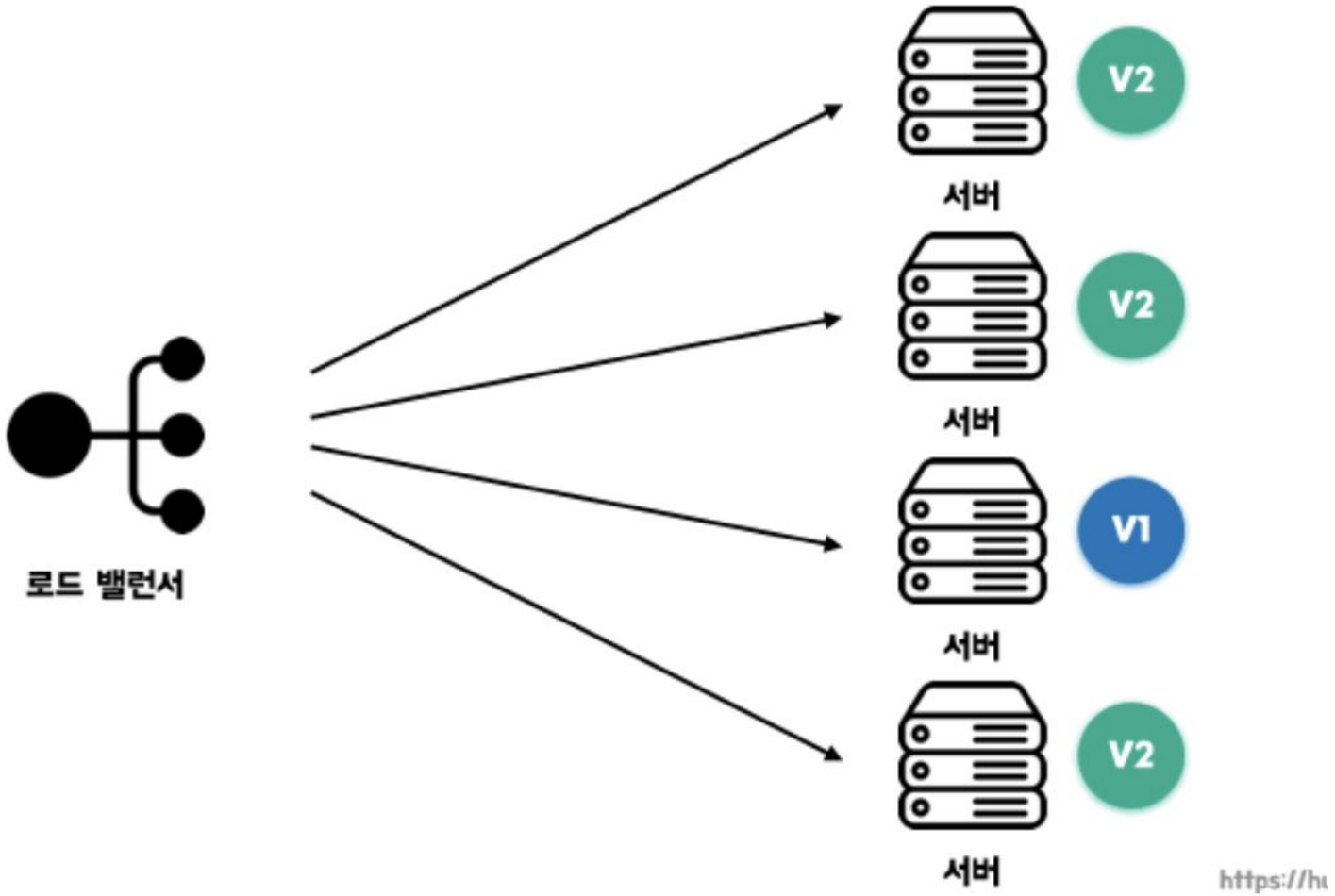
이어진

무중단 배포

서비스가 중단되지 않은 상태(zero-downtime)로, 새로운 버전을 사용자들에게 배포하는 것

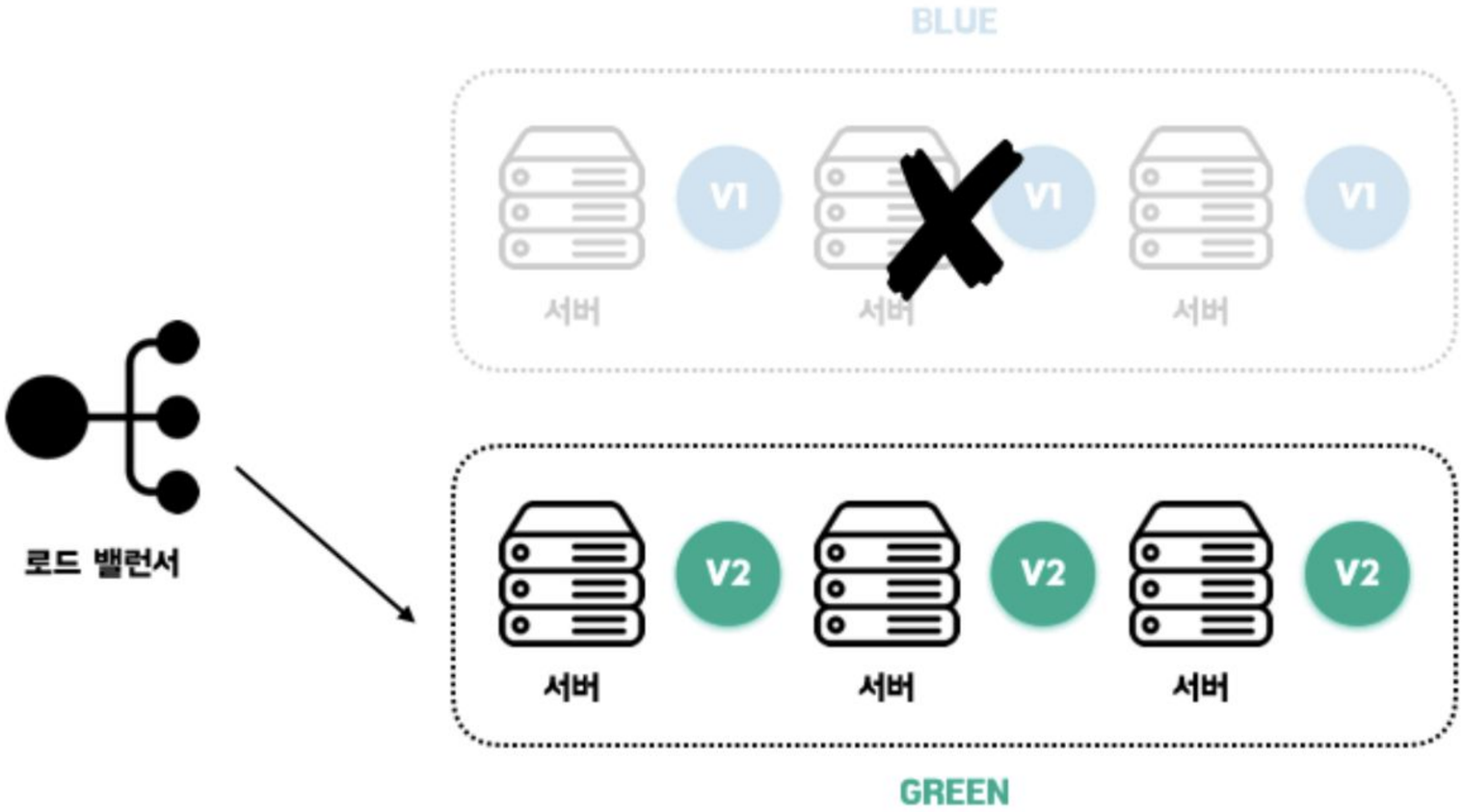
무중단 배포를 하기 위해서는 최소 서버가 2대 이상을 확보해야한다.

배포 전략 - 롤링(Rolling) 배포



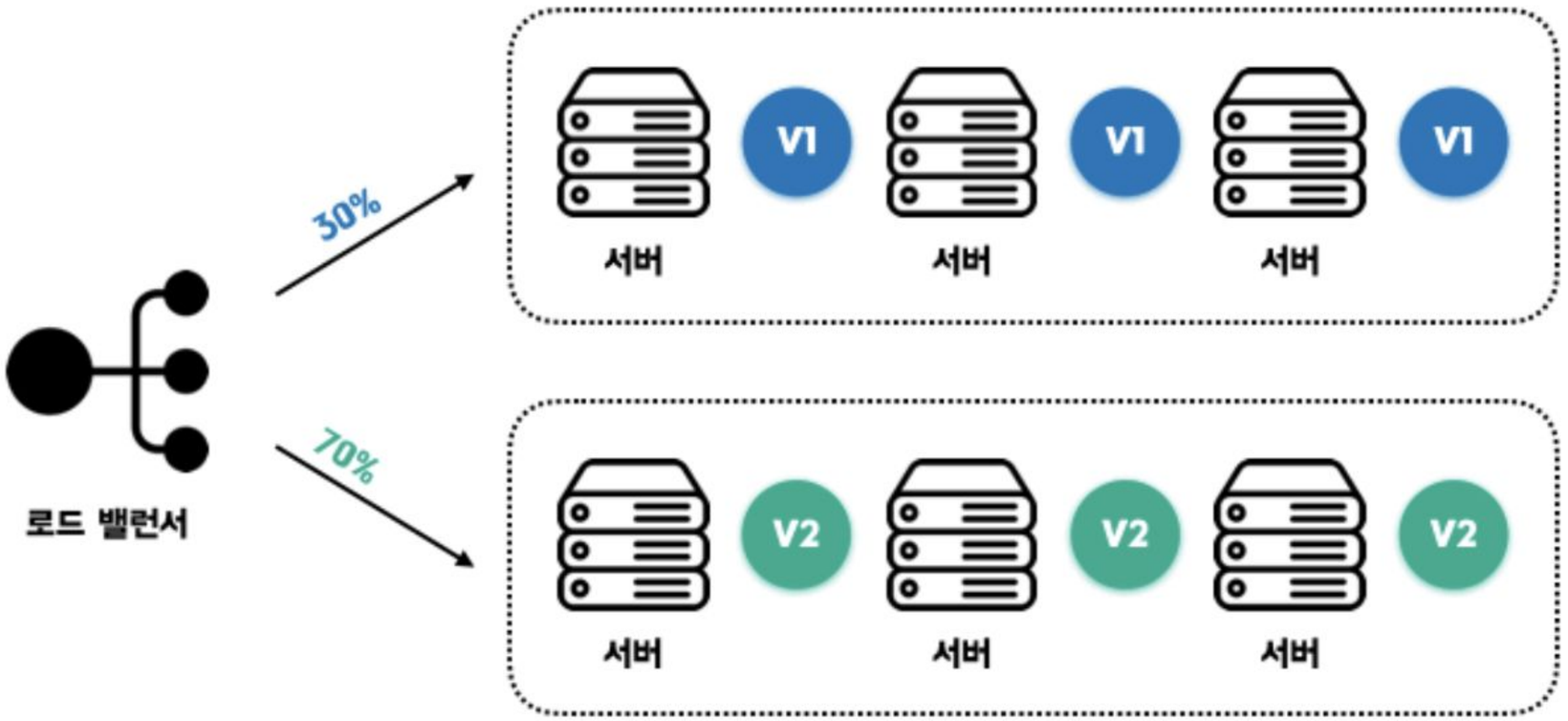
트래픽을 **점진적으로** 구버전에서 새로운 버전으로 옮기는 방식

배포 전략 - Blue/Green 배포



트래픽을 한번에 구버전에서 신버전으로 옮기는 방법이다. Blue/Green 배포 전략에서는 **현재 운영중인 서비스의 환경을 Blue라고 부르고, 새롭게 배포할 환경을 Green**

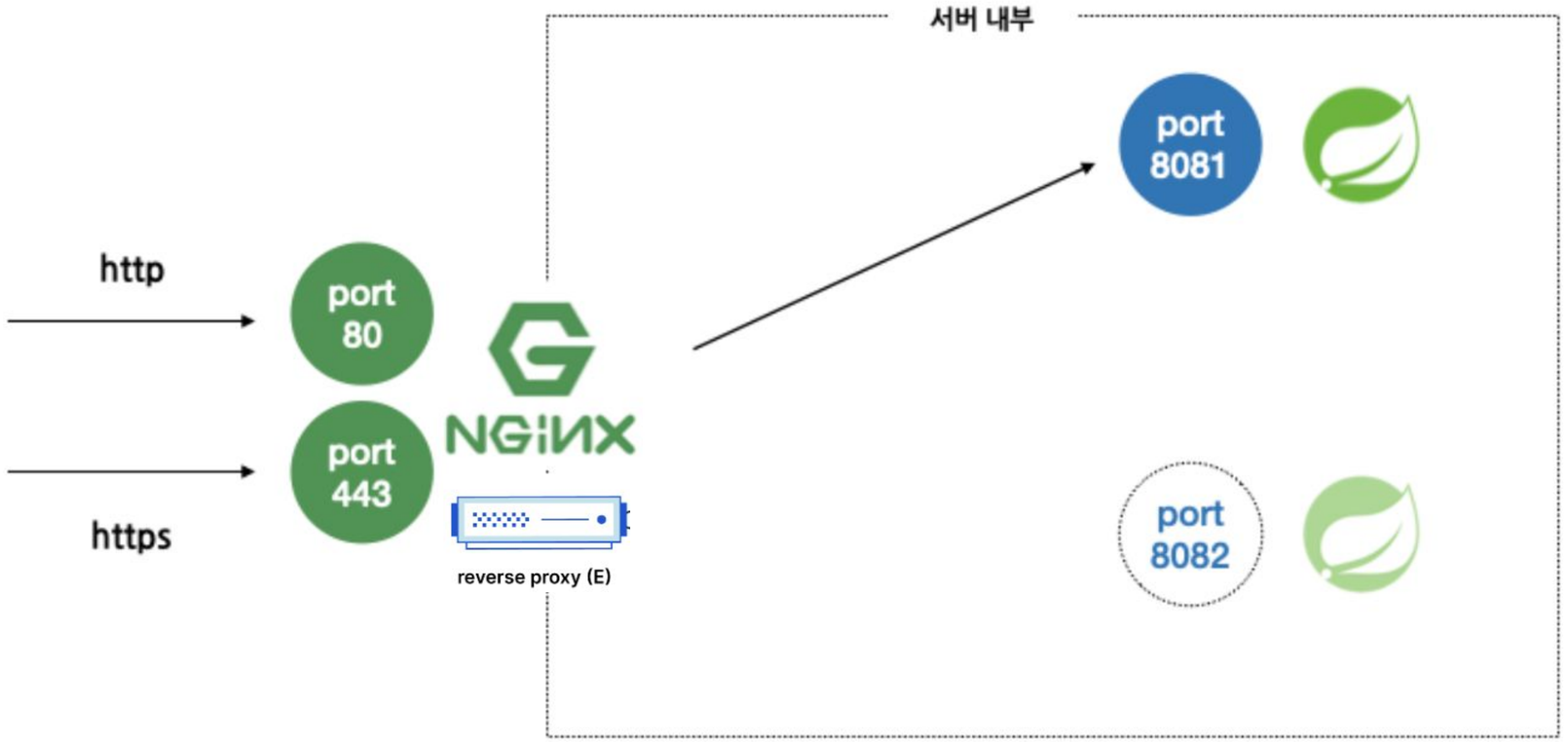
배포 전략 - 카나리(Canary) 배포



점진적으로 구버전에 대한 트래픽을 신버전으로 옮기는 것은 롤링 배포 방식과 비슷하다.
다만 카나리 배포의 핵심은 새로운 버전에 대한 **오류를 조기에 감지**하는 것

Blue/Green 배포 실습

아키텍처



Blue/Green 배포 실습

Dockerfile 작성 (Spring 기준)

```
FROM openjdk:17
COPY ./build/libs/demo-0.0.1-SNAPSHOT.jar app.jar
ENTRYPOINT ["java", "-jar", "app.jar"]
```

Blue/Green 배포 실습

docker-compose.yml 작성 (Spring 기준)

```yaml
version: '3'
services:
  blue:
    container_name: blue
    image: eojin0814/demo
    ports:
      - 8081:8080
    environment:
      - TZ=Asia/Seoul
  green:
    container_name: green
    image: eojin0814/demo
    ports:
      - 8082:8080
    environment:
      - TZ=Asia/Seoul
```

Blue/Green 배포 실습

deploy.sh작성 (Spring 기준)

```
#!/bin/bash

IS_GREEN_EXIST=$(docker ps | grep green)
DEFAULT_CONF=" /etc/nginx/nginx.conf"

# blue가 실행 중이면 green을 up합니다.
if [ -z $IS_GREEN_EXIST ];then
  echo "### BLUE => GREEN ####"
  echo ">>> green image를 pull합니다."
  docker-compose pull green
  echo ">>> green container를 up합니다."
  docker-compose up -d green
  while [ 1 = 1 ]; do
  echo ">>> green health check 중..."
  sleep 3
  REQUEST=$(curl http://127.0.0.1:8082)
    if [ -n "$REQUEST" ]; then
      echo ">>> 🍃 health check success !"
      break;
    fi
  done;
  sleep 3
  echo ">>> nginx를 다시 실행 합니다."
  sudo cp /etc/nginx/nginx-green.conf /etc/nginx/nginx.conf
  sudo nginx -s reload
  echo ">>> blue container를 down합니다."
  docker-compose stop blue

# green이 실행 중이면 blue를 up합니다.
else
  echo "### GREEN => BLUE ###"
  echo ">>> blue image를 pull합니다."
  docker-compose pull blue
  echo ">>> blue container up합니다."
  docker-compose up -d blue
  while [ 1 = 1 ]; do
    echo ">>> blue health check 중..."
    sleep 3
    REQUEST=$(curl http://127.0.0.1:8081)
    if [ -n "$REQUEST" ]; then
      echo ">>> 🍃 health check success !"
      break;
    fi
  done;
  sleep 3
  echo ">>> nginx를 다시 실행 합니다."
  sudo cp /etc/nginx/nginx-blue.conf /etc/nginx/nginx.conf
  sudo nginx -s reload
  echo ">>> green container를 down합니다."
  docker-compose stop green
fi
```

Blue/Green 배포 실습

Github action 배포를 위한 deploy.yml 작성 (Spring 기준)

```yaml
jobs:
  build:

    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@v3
      - name: Set up JDK 17
        uses: actions/setup-java@v3
        with:
          java-version: '17'
          distribution: 'temurin'

      - name: application.yml
        run: |
          mkdir ./src/main/resources
          touch ./src/main/resources/application.yml
          echo "${{ secrets.APPLICATION }}" > ./src/main/resources/application.yml # github actions에서 설정한 값을 applicatio

      - name: gradle build
        run: chmod +x gradlew

      - name: gradle build
        run: ./gradlew build -x test

      - name: docker
        run: |
          docker login -u ${{ secrets.DOCKER_USERNAME }} -p ${{ secrets.DOCKER_PASSWORD }}
          docker build -t ${{ secrets.DOCKER_REPOSITORY }}/${{ secrets.DOCKER_IMAGE }} .
          docker push ${{ secrets.DOCKER_REPOSITORY }}/${{ secrets.DOCKER_IMAGE }}
```

Blue/Green 배포 실습

Github action 배포를 위한 deploy.yml 작성 (Spring 기준)

```yaml
- name: deploy.sh
  uses: appleboy/scp-action@master
  with:
    username: ubuntu
    host: ${{ secrets.AWS_HOST }}
    key: ${{ secrets.AWS_KEY }}
    port: ${{ secrets.AWS_PORT }}
    source: "./scripts/deploy.sh"
    target: "/home/ubuntu/"

- name: docker-compose.yml
  uses: appleboy/scp-action@master
  with:
    username: ubuntu
    host: ${{ secrets.AWS_HOST }}
    key: ${{ secrets.AWS_KEY }}
    port: ${{ secrets.AWS_PORT }}
    source: "./docker-compose.yml"
    target: "/home/ubuntu/"

- name: docker hub depoly
  uses: appleboy/ssh-action@master
  with:
    username: ubuntu
    host: ${{ secrets.AWS_HOST }}
    key: ${{ secrets.AWS_KEY }}
    script: |
      sudo docker pull ${{ secrets.DOCKER_REPOSITORY }}/${{ secrets.DOCKER_IMAGE }}
      chmod 777 ./scripts/deploy.sh
      cp ./scripts/deploy.sh ./deploy.sh
      ./deploy.sh
      docker image prune -f
```

Blue/Green 배포 실습

Github action Log 확인(Spring 기준)

```
52
53  Pulling blue ... done
54  Recreating blue ...
55  out: >>> blue health check 중...
56  err:
57  Recreating blue ... done
```

```
err:   % Total    % Received % Xferd  Average Speed   Time    Time     Time  Curren
err:                                  Dload  Upload   Total   Spent    Left  Speed
err:
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
100    89    0    89    0     0    247      0 --:--:-- --:--:-- --:--:--   252
out: >>> 🌿 health check success !
out: >>> nginx를 다시 실행 합니다.
err: cp: cannot stat '/etc/nginx/nginx-blue.conf': No such file or directory
out: >>> green container를 down합니다.
err: Stopping green ...
out: Total reclaimed space: 0B
err:
Stopping green ... done
```

Blue/Green 배포 실습

Reverse Proxy nginx 설정

```
http {
    include       /etc/nginx/mime.types;
    default_type  application/octet-stream;

     # 백엔드 upstream 설정
    # upstream demo {
    #     server api:8080;
    # }

     # 프론트엔드 upstream 설정
    upstream next-server {
        server 172.0.0.1:3000;
    }

    server {
        listen 80;

            # /api 경로로 오는 요청을 백엔드 upstream 의 /api 경로로 포워딩
        # location /api {
        #     proxy_pass         http://myweb-api/api;
        # }

            # / 경로로 오는 요청을 프론트엔드 upstream 의 / 경로로 포워딩
        location / {
            proxy_pass         http://next-server/;
            proxy_http_version 1.1;
            proxy_set_header Upgrade $http_upgrade;
            proxy_set_header Connection 'upgrade';
            proxy_set_header Host $host;
            proxy_cache_bypass $http_upgrade;
        }
    }
```