

# 1장. AWS - 배포 v3 시작 전 준비사항

## ● Region 설정 및 종료

서울 서버가 가장 가깝다... 리전에서

1. 엘라스틱빈스톡 → 환경종료 : 여러 개 있으면 비용이 발생하니까...
2. 엘라스틱빈스톡 → 애플리케이션 종료(보안그룹 때문에 환경이 종료되지 않는 경우 있음.)
3. RDS 종료
4. EC2 혹시나 켜져있는 것이 있으면 종료
5. 탄력적IP 켜져 있다면 삭제
6. 로드밸런서(비용 듬) 종료
7. 좌측 메뉴에서 Elastic block Store - 볼륨하고 스냅샷 삭제
8. 보안그룹 - default 만 남기고 전부 삭제
  - 보안그룹 3개라면
  - 『 A보안그룹, B보안그룹(RDS), C보안그룹(EC2) → B가 C를 의존하고 있다면, C 삭제 후 B 삭제』
  - 하나씩 체크해 가면서 삭제해야 한다.
9. 제일 중요한 것...
  - 리전을 서울로 맞추는 것 ....

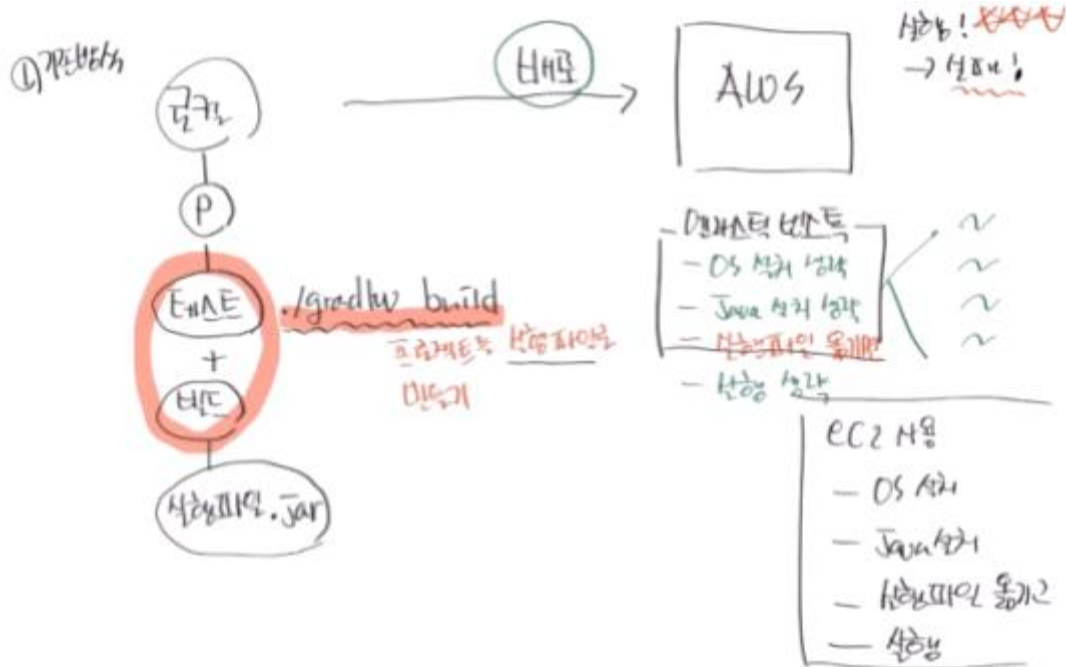
## 2장. AWS - 배포 v3 AWS 요금정책

▶ 검색 → 결제 또는 청구서 → 좌측 메뉴 → 청구서 : 자주 확인해 보고 청구금액 있는지 확인.

1. AWS 프리티어(1년간 매달 750시간 무료) 비용 정책
  - 사용범위 추가
  - 무료가 아닌 것을 사용하면 비용 청구 됨.
2. 엘라스틱 빈스톡 서비스의 비용
  - EC2 (한대를 켜두면 24시간 + 31일 = 744시간)
  - EC2 (한대를 켜두면 24시간 + 31일 = 744시간)
    - 두 대 켜 경우 총 한 달에 약 1500시간.... 그러니까 15일 이내만 돌려야하고
    - 실습이 끝나면 종료해야 비용청구가 안됨...
  - S3(파일 저장소)
    - 신규가입 첫 해 매월 5G 무료, 자료의 입출력 2,000건 소요 시 100G 까지 무료 -- 걱정안해도 됨.
  - RDS(데이터베이스) : 한달 750시간 무료. 한대 켜 놓을 거니까... 괜찮음..
  - 로드밸런서(두개 사용예정 - Application, Network Load Balancer) : 프리티어 기준 매월 750시간 사용
    - 두개 사용할 거니까... 15일만 사용... 사용 후 바로 종료

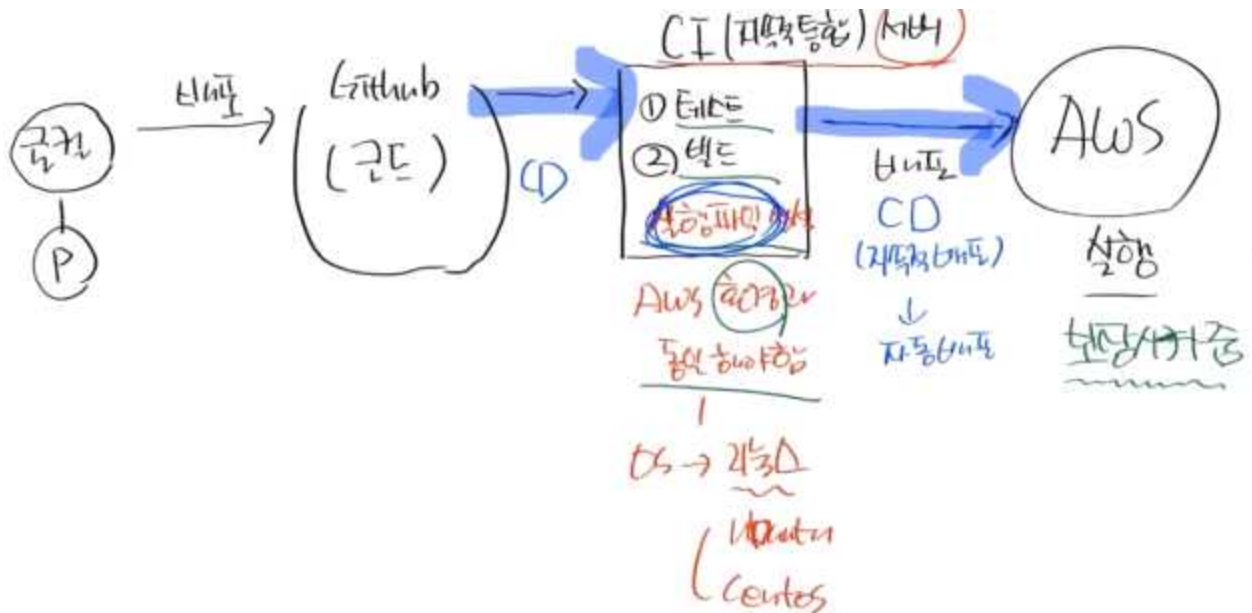
### 3장. AWS - 배포 v3 전체구성 알아보기

▶ 기존 방식 : 로컬프로젝트 → 테스트 → 빌드(./gradlew build) → 실행파일 .jar 생성 → AWS 배포  
 - 운영체제의 환경 차이(테스트는 윈도우 - 배포는 리눅스)로 인해 배포 후 실행 단계에서 실패할 확률이 생긴다.



### ● 새로운 방식의 배포

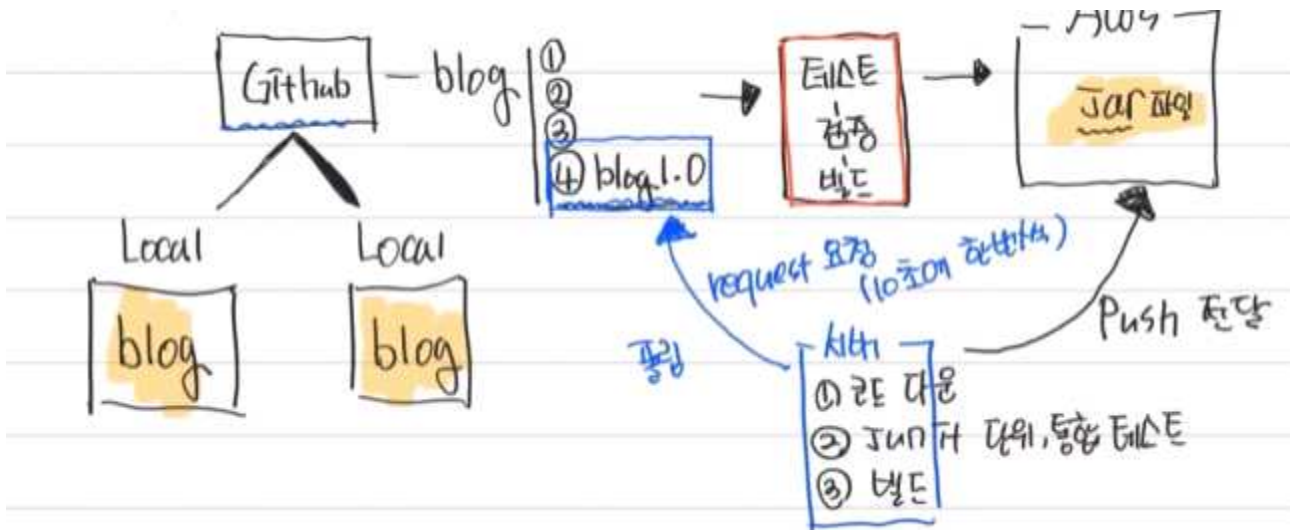
▷ aws-v3 프로젝트를 로컬에서 생성 후 → Github으로 배포 → CI(지속적 통합 : Continuous Integration) 서버에 던짐 → 그 후 자동배포(CD : Continuous Delivery)



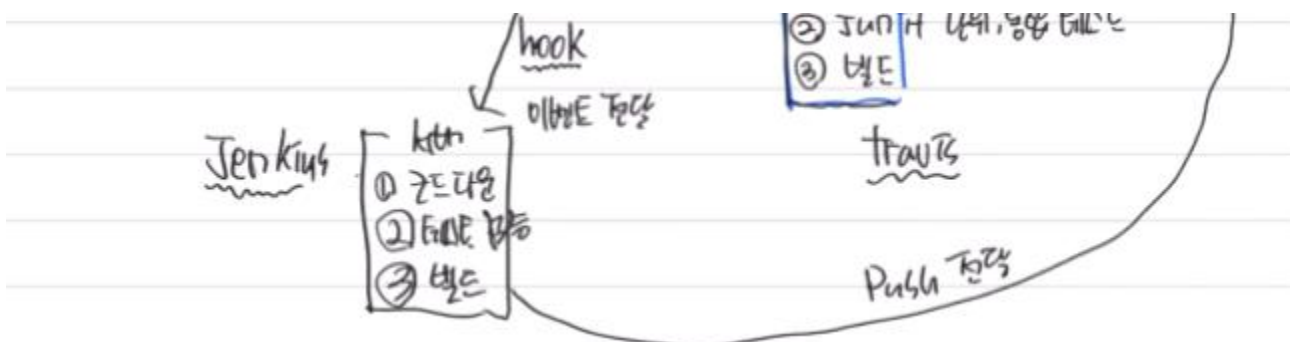
기존 EC2 사용 시에는 AWS에 로그인 후 진행했었으나, CI에서 던지는 것은 로그인을 거치지 않고 CI에서 AWS에 배포를 하는 것이기 때문에 CI에서 CD를 하려면 Access key가 필요하다... 그래서 IAM 이란 개념을 이해하고 진행해야 한다.

## 4장. AWS - 배포 v3 CI/CD란

- CI : 지속적 통합 - Continuous Integration
- CD : 지속적 배포 - Continuous Delivery



- ① 폴링 : 깃헙에 변화가 있는지 지속적으로 Request 한 후 변화가 생기면 그 코드를 내려받아 Junit으로 단위테스트, 통합 테스트 후 문제가 없으면 AWS에 배포(Travis)
- ② WebHook : 깃헙에 WebHook 을 설정 → 내 코드 변화가 생기면 → 서버에 이벤트를 전달해서 서버가 알아차리게 함.(Jenkins)

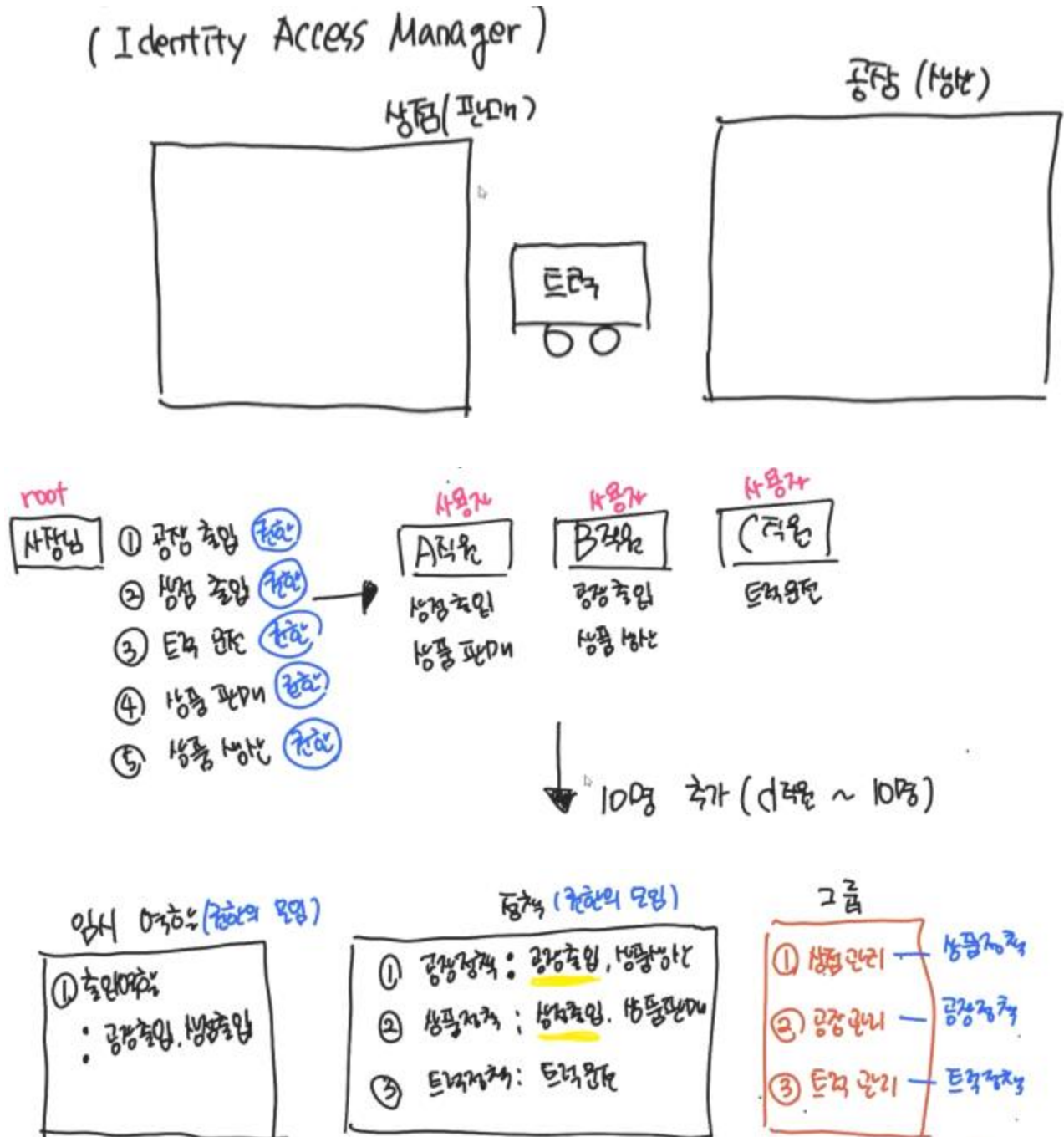


혹이 좀 낫긴 하다. 왜 일까요? 변화를 지속적으로 감지하는 것 보다 변화가 생겼으니 처리해라 하는 방법이 낫지요.... 이런 것을 무료로 사용할 수 있는 방법 그것이 CI 서버를 구축하는 것이다. Github Action을 사용하면 두 서비스를 사용하지 않아도 가능하다.

## 5장. AWS - 배포 v3 IAM 알아보기

### ● AWS 사용자, 정책, 그룹, 역할 이해하기

① IAM (Identity Access Manager)



사용자가 아닌 것(서비스)은 역할을 만든다. 즉 트럭에 출입 역할을 부여한다. 트럭은 사용자가 아닌 서비스 이므로...

## 6장. AWS - 배포 v3 github 프로젝트 fork하기

- 본인 깃헙에서 처리 → 코드를 다운로드 하는 것이 아닌 다른 방법을 사용..  
→ Fork는 두개 서로 다른 깃헙 계정의 레포끼리 코드를 복제할 때 사용  
▷ github.com/zzzmini/aws3 → 깃헙 우 상단 Fork → 나의 깃헙으로 가져오기

## 7장. AWS - 배포 v3 RDS생성

EC2로 가서

보안그룹 이름(sg... 어쩌구 - sg는 Security Group)을 나중에 외우기도 힘들고 봐도 식별이 어려우니 이를 알기 쉽도록 생성한다.

- ① 좌측 메뉴 → 네트워크 및 보안 → 보안그룹 → 우상단 → 보안그룹 생성  
▷ 보안그룹이름 : security-group-aws-v3(sg로 시작하면 보안그룹 생성이 안됨.. 예약어라 그런가 봄.)

보안 그룹 이름 정보

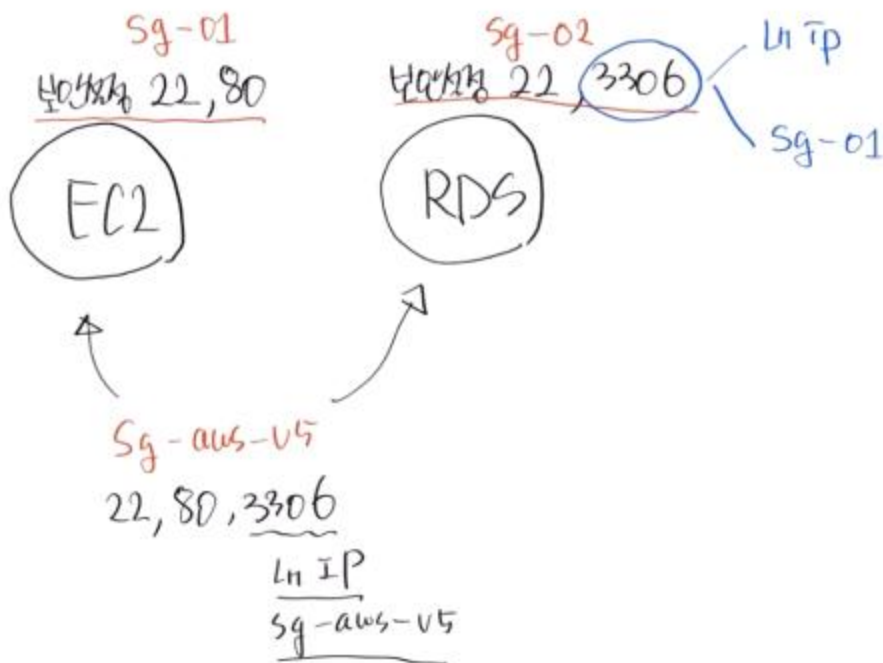


- ▷ 설명 : 같이  
▷ VPC 정보 포함해서 애들 정보를 기록해 놓자.. 메모장에....  
  
▷ 아래 인바운드 규칙추가  
→ 보안 그룹에는 보안그룹 이름과 포트 80, 22, 3306(내 ip와 같은 그룹이면 다 들어올 수 있게 설정)  
② 인바운드 규칙에 추가  
▷ SSH 22번 포트 누구나(Anywhere)  
▷ HTTP 80번 포트 누구나(Anywhere)  
▷ MySQL 3306 - 자동으로 생성 → 애는 내 IP로 .... 하고..  
  
③ 아래 보안그룹생성 버튼 클릭 → 생성 후 화면 중 하단 인바운드 규칙 → 우측 인바운드 규칙 편집  
→ 규칙추가 단추 클릭 → 3306포트 추가

MYSQL 추가 선택하고.... 그룹에 좀 전에 생성한 위의 그룹명을 선택하면 AWS가 관리하는 그룹명으로 변경되어 나타난다.



▷ 향후 위 시큐리티 그룹을 지정하면.... 22번, 80번, RDS 접근이 가능한 그룹이 생성 됨.



위 처럼 그룹을 권한별로 나누어 정하는 것이 정책상 맞으나 단순하게 처리하기 위해 아래처럼 하나의 보안그룹에 묶어서 처리하기로 한다.

## ● RDS 검색 - 생성합니다

▷ RDS 들어와서 데이터베이스 생성 단추 클릭 - MySQL - 프리티어

- ① DB 인스턴스 식별자 정보 : aws-v3-mysqldb
- ② 마스터 사용자 이름 : zzzmini
- ③ 마스터 암호 설정 :

- ④ 연결 → 퍼블릭 액세스 - 예 : 외부에서 접근 가능하기 위해 설정 함.
- ⑤ 기존 VPC 보안그룹 : 생성한 보안그룹(security-group-aws-v3) 맞춰주고, 기존 default 는 삭제
- ⑥ 추가구성 할 거 없음. → 맨 밑 데이터베이스 생성 단추 클릭

▶ aws-v3-mysqldb하단 VPC 정보도 메모장에 기록 → 생성 완료되면 우측 VPC 보안그룹도 보인다.



■ 첫번째 할 것은 MobaXTerm으로 EndPoint 정보를 이용해 접속 DB도 만들고 등등을 해야 함.

- ① HeidiSQL 실행해 주고
- ▷ 신규 세션 만들고 → aws-v3-mariadb 적고 우측
  - 호스트명 : Endpoint
  - 사용자 및 암호는 아까 RDS 생성할 때 사용자 정보 입력 → 변경사항 저장하고.. 열림..

▷ 상단 쿼리 열고

지난번 만든 DDL 구문 하나씩 실행

- ㉠ DB 만들고
- ㉡ DB 사용 설정하고
- ㉢ 테이블 생성하고
- ㉣ UTF8 설정하고

```
CREATE DATABASE book

USE book

create table book (
  id bigint auto_increment primary key,
  title varchar(255),
  content varchar(255),
  author varchar(255)
);

SELECT * FROM Book

SHOW VARIABLES LIKE 'c%'

INSERT INTO book(title, content, author)
```

VALUES ('제목1','내용1','저자1')
----------------------------

생성하고 좌측 트리에서 생성된 것 확인....

▷ 타임 존 확인하고 변경하기 (9시간 느리니까)

SELECT NOW();

▶ 다시 AWS RDS로 가서

→ 좌측 파라미터그룹 메뉴 선택 - 우측 메뉴 파라미터 그룹 생성

- 파라미터 그룹 패밀리 : mysql 8.0

- 이름 : aws-v3-mysqldb-group 생성 → 생성된 파라미터 그룹 클릭 → 우 상단 편집

많은 데이터베이스 관련 세팅이 있다. 파라미터에서 time\_zone으로 검색

- time\_zone 에서 Asia/Seoul 입력하면 셋팅 고춧...

※ 파라미터 그룹은 default가 있으나 애는 편집이 안되서 새롭게 만들어 설정해야 함.

## ● 타임존 적용하기

- 만들 파라미터 그룹을 적용하려면..

① 좌측 데이터베이스 메뉴 선택 → 해당 DB 식별자 선택 → 우 상단 작업에서 수정 누르고...

② 아래쪽으로 쭉욱 내려가면 추가구성 → DB 파라미터 그룹 → 좀 전에 만든 파라미터 그룹으로 연결..

③ 맨 아래 적용 → DB 인스턴스 수정 → 수정 예약 → 즉시 적용 → DB 인스턴스 수정 단추 클릭

※ 최종 적용하려면 Amazon RDS 를 재부팅을 해야 함.

데이터베이스 → 우 상단 → 작업 → 재부팅 → 확인

(바로 안 되면 좀 기다렸다가.... 아까 정해졌던 파라미터 그룹 재설정 시간이 필요하므로)

다시 HeidiSQL 완전 종료 후 재접하고,

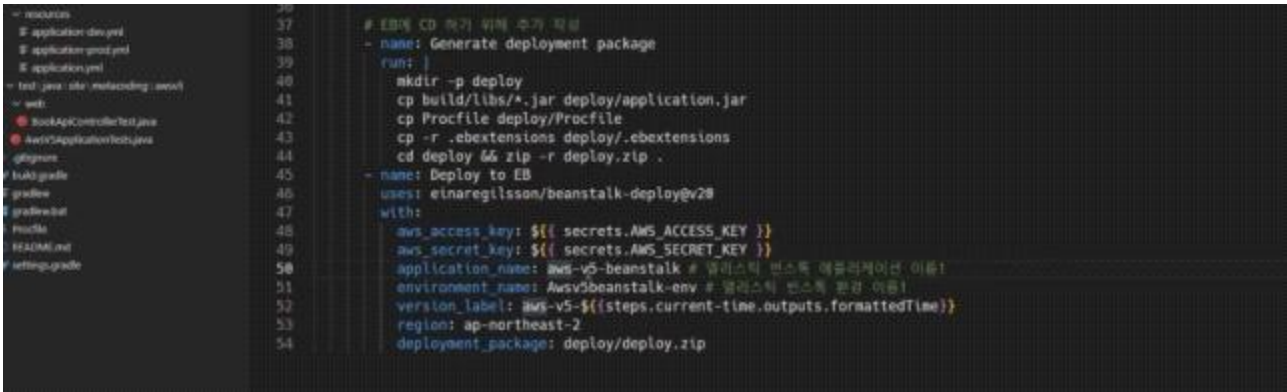
SELECT NOW();

실행해 보면 시간 설정 완료 된다.



## 8장. AWS - 배포 v3 엘라스틱 빈스톡 생성

▶ 검색에서 Elastic Beanstalk 검색 → 우측 시작하기 → Create Application



```
37 # EB에 CD 하기 위해 추가 작성
38 - name: Generate deployment package
39   run: |
40     mkdir -p deploy
41     cp build/libs/*.jar deploy/application.jar
42     cp Procfile deploy/Procfile
43     cp -r .ebextensions deploy/.ebextensions
44     cd deploy && zip -r deploy.zip .
45 - name: Deploy to EB
46   uses: einaregilsson/beanstalk-deploy@v20
47   with:
48     aws_access_key: ${ secrets.AWS_ACCESS_KEY }
49     aws_secret_key: ${ secrets.AWS_SECRET_KEY }
50     application_name: aws-v3-beanstalk # 엘라스틱 빈스톡 애플리케이션 이름!
51     environment_name: AwsV3beanstalk-env # 엘라스틱 빈스톡 환경 이름!
52     version_label: aws-v5-${ steps.current-time.outputs.formattedTime }
53     region: ap-northeast-2
54     deployment_package: deploy/deploy.zip
```

배포 프로젝트에 설정한 환경변수의 이름과 일치 시켜야 함.

첫째, 1단계 - 환경구성

→ aws-v3-beanstalk-env 구성

- 애플리케이션 이름 : aws-v3-beanstalk

- 플랫폼 : java

▷ 아래 쪽 다음 클릭

※ 사전 설정 → 구성 사전 설정 → 사용자 지정 구성 선택

(인스턴스를 2개 구성하기 위함... 실습 종료 후 삭제해야 비용발생 안함.)

둘째, 2단계 - 서비스 액세스 구성

- 서비스 역할 : 기존 서비스 역할 및 사용

① aws-elasticbeanstalk-service-role

② ec2 키페어

③ 프로파일 : aws-elasticbeanstalk-ec2-role

셋째, 3단계 - 네트워킹, 데이터베이스 및 태그설정

① VPC 설정 : vpc-030eaf76cc74f155d

② 인스턴스 설정 → 퍼블릭 IP 주소 : 활성화됨 체크

③ 인스턴스 서브넷 - 두개 지정(로드밸런서 때문)

-- ap-northeast-2a, ap-northeast-2b

넷째, 4단계 - 인스턴스 트래픽 및 크기 조정 구성

- EC2 보안그룹 → security-group-aws-v3

▶ 용량 → 오토 스케일링 그룹 → 환경 유형 → 밸런싱된 로드

▶ 최소 2개(같은 인스턴스 복제) 최대 4개(Auto Scaling - 자동으로 증가시킬 인스턴스 수)

## ▼ 용량 정보

환경의 컴퓨팅 파워와 오토 스케일링 설정을 구성하여 사용되는 인스턴스 수를 최적화합니다.

## 오토 스케일링 그룹

### 환경 유형

단일 인스턴스 또는 로드 밸런싱된 환경을 선택합니다. 단일 인스턴스 환경에서 애플리케이션을 개발 및 테스트하여 비용을 절감한 다음 애플리케이션이 프로덕션에 투입할 준비가 되면 로드 밸런싱된 환경으로 업그레이드할 수 있습니다. [자세히 알아보기](#)

밸런싱된 로드 ▼

### 인스턴스

2 최솟값

4 최댓값

## ▶ 로드 밸런서 유형 → 애플리케이션 로드 밸런서

- 로드 밸런서 유형 : Application Load Balancer 로 선택(https 용)

## 로드 밸런서 유형

### ☒ 애플리케이션 로드 밸런서

애플리케이션 계층 로드 밸런서 - 프로토콜, 포트, 환경 프로세스에 대한 경로를 기반으로 HTTP 및 HTTPS 트래픽을 라우팅합니다.

### ☐ Classic Load Balancer

이전 세대 - HTTP, HTTPS 및 TCP

### ☐ Network Load Balancer

애플리케이션을 위한 최고성능 및 고정 IP 주소.

### ☒ 전용

Elastic Beanstalk가 이 환경 전용으로 생성하는 로드 밸런서를 사용합니다.

### ☐ 공유됨

계정의 누군가가 생성한 로드 밸런서를 사용합니다. 로드 밸런서는 여러 Elastic Beanstalk 환경에서 공유할 수 있습니다.

## ▶ 리스너 : 어떤 포트로 요청이 들어오는지 체크... 여기서는 80포트를 모니터링 함.

8080으로 요청 들어오면 모름... 일 안하겠지요.

### 리스너

로드 밸런서에 대한 리스너를 지정할 수 있습니다. 각 리스너는 지정된 프로토콜을 사용하여 지정된 포트에서 수신 클라이언트 트래픽을 환경 프로세스로 라우팅합니다. 기본적으로 로드 밸런서는 포트 80에서 표준 웹 서버로 구성됩니다.

작업 ▼

리스너 추가

	리스너 포트 ▲	리스너 프로토콜 ▼	SSL 인증서 ▼	기본 프로세스 ▼	활성화됨 ▼
<input type="radio"/>	80	HTTP	—	default	<input checked="" type="checkbox"/>

▶ 프로세스 : 상태 검사 경로 80포트 루트(/)로 신호를 보내서 200OK가 떨어지는지 지속적으로 확인하는 프로세스, 즉 80포트로 들어오면 EC2 두개 프로세스 중 한곳에 라우팅하는 것을 결정한다. 이는, 프로젝트에 @GetMapping("/") 가 만약 없다면 프로세스 검사에서 실패를 리턴한다.(404에러) 인터넷에 well Known port 검색해 보면 다양한 정해진 포트 구성을 살펴볼 수 있다.

### 프로세스

각 환경 프로세스에 대해 로드 밸런서가 요청을 프로세스로 라우팅하는 데 사용하는 프로토콜 및 포트를 지정할 수 있습니다. 또한 로드 밸런서가 프로세스 상태 확인을 수행하는 방법을 지정할 수도 있습니다.

작업 ▼

프로세스 추가

	이름 ▼	포트 ▼	프로토콜 ▼	HTTP 코드 ▼	상태 확인 경로 ▼	고정성 ▼
<input type="radio"/>	default	80	HTTP		/	Disabled

- ▷ 리스너, 프로세스 안건들.
- ▷ 맨 아래 쪽 우측 → 다음 클릭

다섯째, 5단계 - 업데이트, 모니터링 및 로깅 구성

▶ 롤링 업데이트 및 배포 → 배포정책 : 변경불가

## 애플리케이션 배포

Amazon Elastic Beanstalk가 소스 코드 변경 사항 및 소프트웨어 구성 업데이트를 배포하는 방법을 선택합니다. 자세히 알아보기 [\[?\]](#)

### 배포 정책

한 번에 모두	▲
한 번에 모두	✓
롤링	
추가 배치를 사용한 롤링	
변경 불가	
트래픽 분할	변경 불가

배포 전략

새 애플리케이션 버전의 %

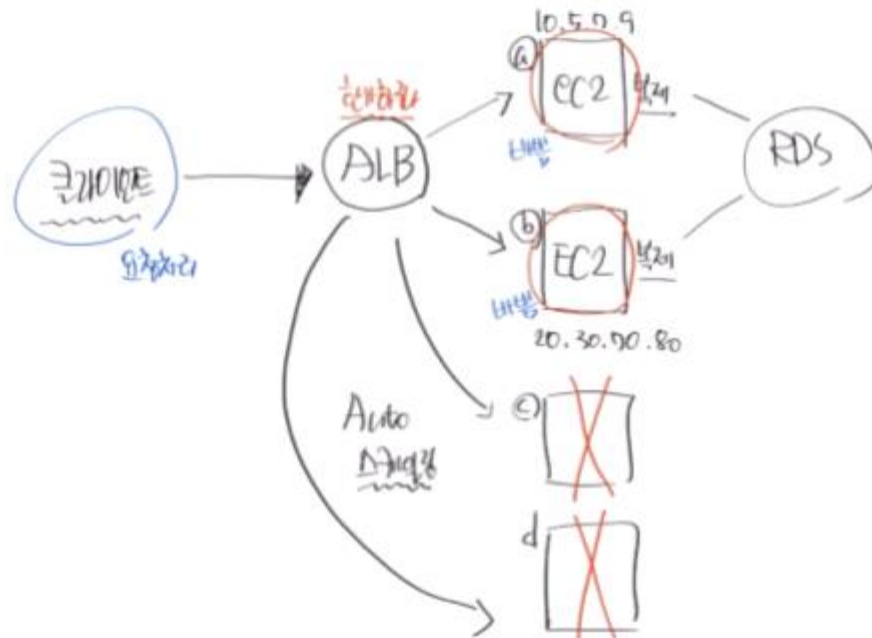
### ▶ Environment properties

- RDS\_HOSTNAME : RDS EndPoint 복붙
- RDS\_PORT : 3306
- RDS\_DB\_NAME : book
- RDS\_USERNAME : zzzmini
- RDS\_PASSWORD : 알아서

위 자료는 크롬 새 탭 열고 RDS 로 찾아 들어가서 확인하던가, 메모장 참고 자료 확인.

직접 프로젝트 코드에 삽입해 놓으면 노출되므로 환경 속성으로 셋팅함. → 다음

▶▶▶▶▶▶▶▶ 제출



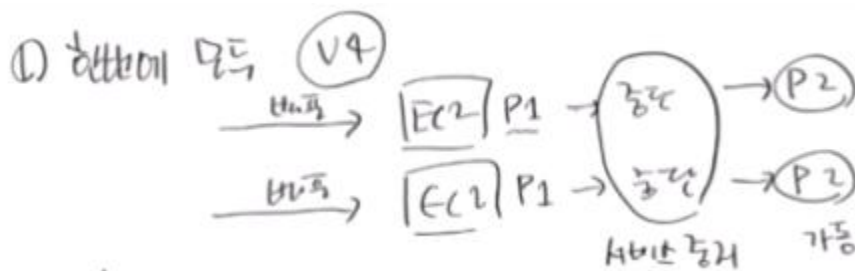
ALB(Application Load Balancer)

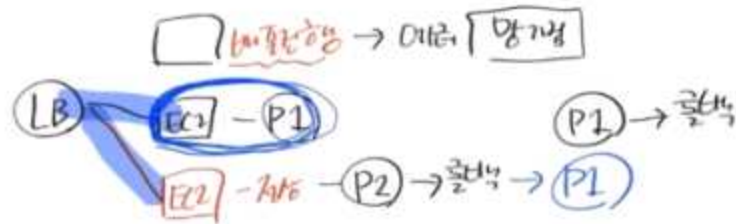
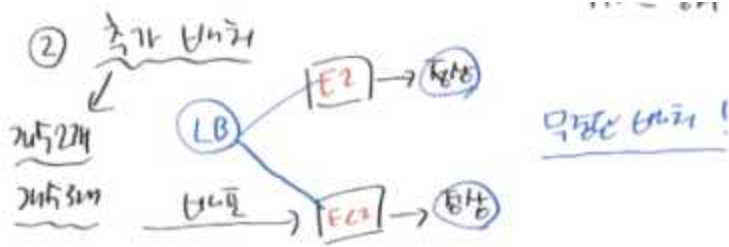
- OSI 8계층의 Application 계층 : 소프트웨어 적으로 존재
- 부하분산

## 9장. AWS - 배포 v3 롤링이란

- 롤링(배포전략) : 무중단 배포전략 수립 가능 → 이전 애플리케이션 배포 관련 추가 설명.

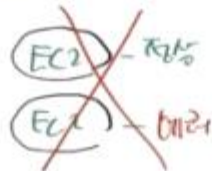
서비스가 멈추면 여러가지 문제가 발생하므로 서비스를 중단하지 않고 업데이트를 시행하는 방법이 필요.





자원소모가 적다. 제일 많이 사용되긴 한다.

③ 블루/그린 → 변경가능



콜백이 끝나면  
자원이 많이 든다

기존 서비스를 블루(2개)가 둘 때, 새로운 배포를 위해 새로운 배포서버 2개(그린)를 만들어 배포하고 성공하면, 기존 서버를 종료하고 새로운 서버 서비스 시작하는 방법. 롤백 쉽고 자원의 사용이 많아진다. 이러한 방법을 롤링 배포(무중단, 배포형식 - 변경 없음... 설정 시 적용된다.)

## 10장. AWS - 배포 v3 현재 구성 살펴보기

- ▶ 셋팅은 전체적으로 마무리 되었음.
- ▶ 엘라스틱 빈스톡 페이지에서 좌측 환경 클릭



## ♣ 지금까지의 구성 정리

### ① DNS 주소확인

② EC2 page → 좌측 로드밸런서 클릭 → 하단 DNS 이름 있음... 복사

????? 근데 IP 주소는 없다.

이 주소를 크롬 열고 주소창에 붙이면 - Congratulation 창 보임.... 웹서버 잘 돌고 있다...

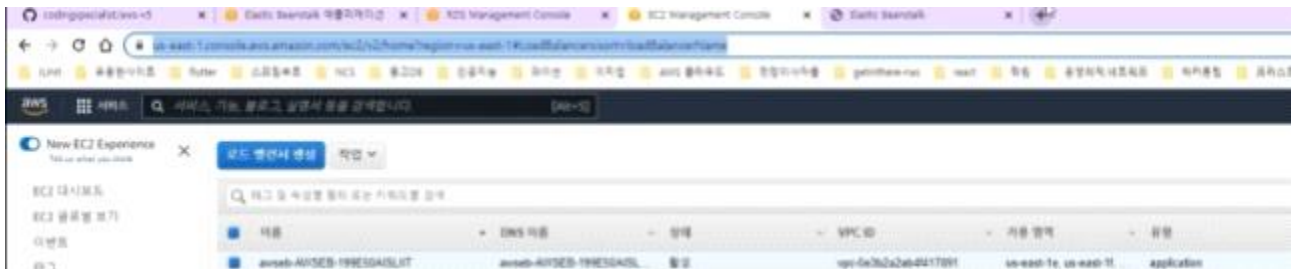
F12 열고 우측 네트워크 탭 선택 클릭 - 중간 IP 확인



위의 IP는 로드밸런서(ALB)의 IP 주소이다.

주소가 없는 이유는 주기적으로 변경되기 때문이다. DNS로 그냥 접근.. 일반적으로 DNS는 IP주소가 매핑이 되어야 하는데 고정 IP가 없으면 서비스가 불가하다. 그래서 ALB 앞에 NLB(Network Load Balancer)를 두어야 하는데, 애는 고정 아이피를 받을 수 있다. 그래서 NLB를 설치해서 사용해야 일관성 있는 서비스제공이 가능해 진다.

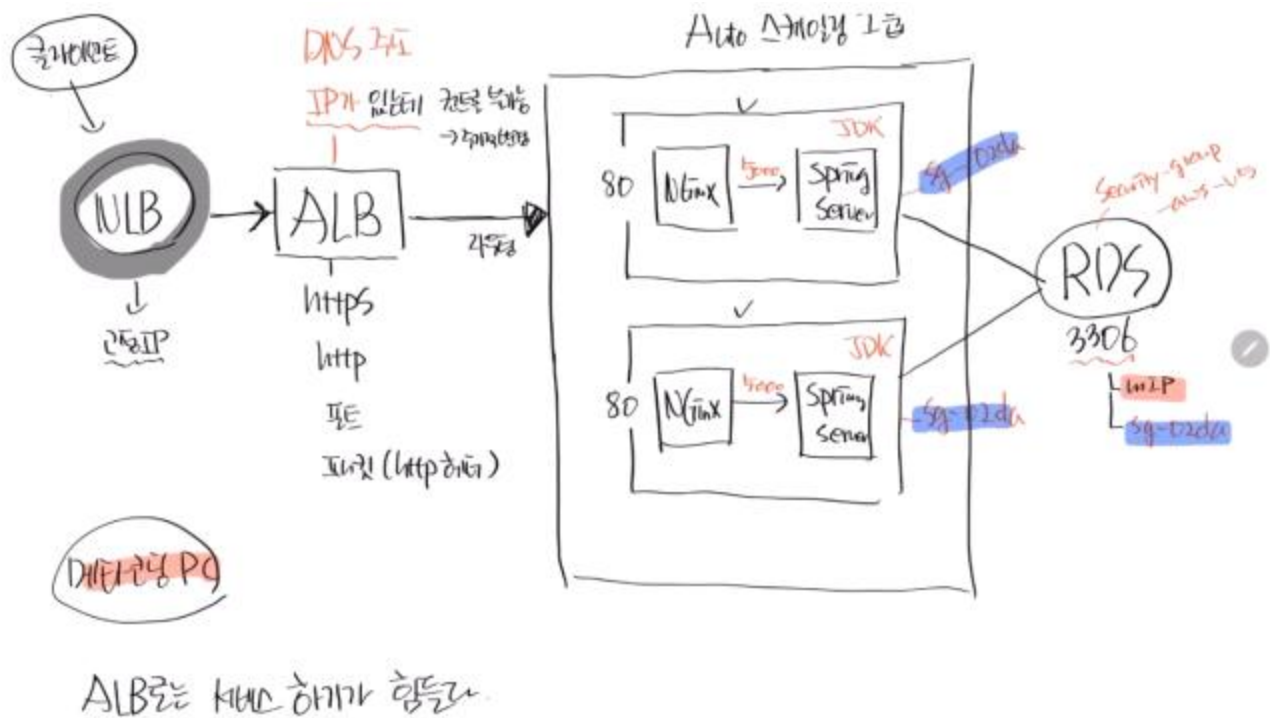
## ▶ 현재 열려있는 창 주소 복사해서 새로운 탭 만들고 붙여넣기



## ▶ 인스턴스 클릭하면... 두개의 EC2가 보인다



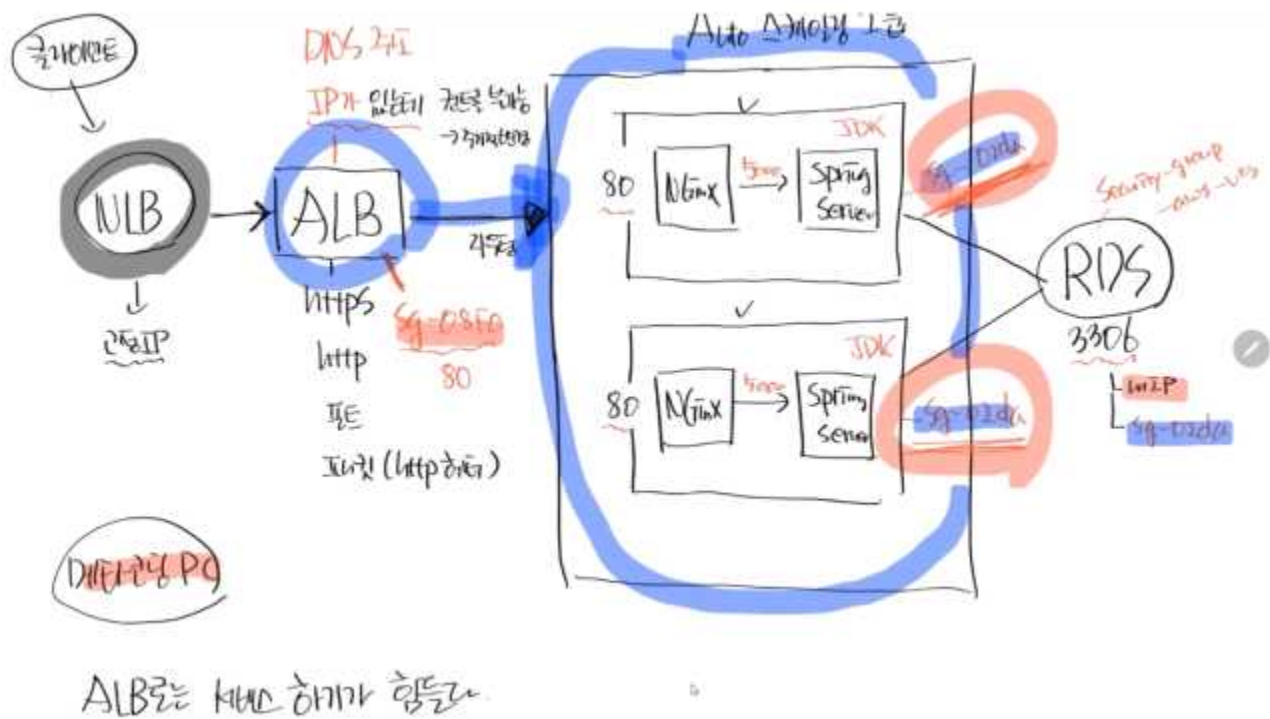
정리하면, ALB를 통해 내부 EC2 인스턴스 두 곳으로 접근하는데, 문제는 ALB IP 주소가 유동적이어서 ALB 앞에 NLB를 두어서 처리를 해야 함.



## ● 시큐리티 보안그룹 설정이해

- 왜 EC2 인스턴스 IP로 접근이 불가능한가?
- 엘라스틱빈스톡이 만들어 지면서 자동으로 보안그룹이 만들어 졌다.
- EC2에는 vpc-030eaf76cc74f155d 어찌구 하는 보안그룹만 접근이 가능하도록 설정했다.
- EC2의 VPC는 vpc-030eaf76cc74f155d 애로 두 인스턴스가 같다. 그니까 EC2 접근 가능한 보안그룹은 로드밸런서의 보안그룹이다.





EC2는 로드밸런서를 통해서만 접근이 가능하도록 설정한거다.

RDS도 설정이 끝났으면 내 IP로도 접속이 안되도록 보안그룹을 변경하는 것이 좋다.

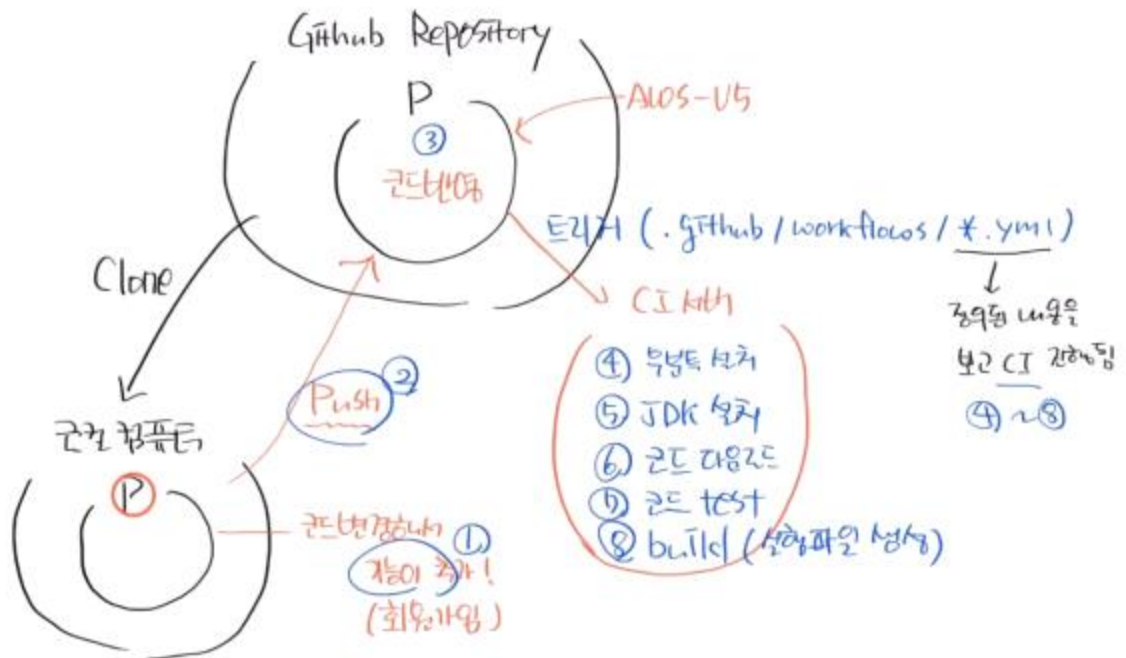
## 11장. AWS - 배포 v3 github 액션 사용해보기

▶ CI 서버 설정해 보자

① 깃 클라켄 사용.

② 내 컴에 적절한 폴더를 만들고, 아까 fork된 파일을 깃헙에서 다운로드 함....

## Github Action (CI / CD 등)



③ CI를 만들려면.. deploy.yml 이 필요하다.(예 위치는 /.github/workflows/뭐뭐.yml)

→ 파일이름 중요하지 않다.)

```
name: aws-v3
on:
  push:
    branches:
      - main

# https://github.com/actions/setup-java
# actions/setup-java@v2 는 사용자 정의 배포를 지원
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout
        uses: actions/checkout@v3
      - name: Set up JDK 17
        uses: actions/setup-java@v3
        with:
          java-version: 17
          distribution: zulu
```

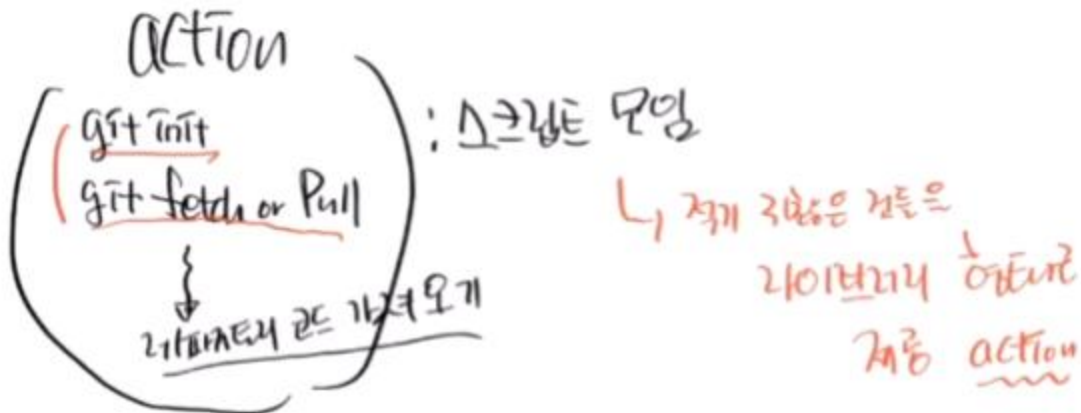
```

- name: Permission
  run: chmod +x ./gradlew
- name: Build with Gradle
  run: ./gradlew clean build -x test

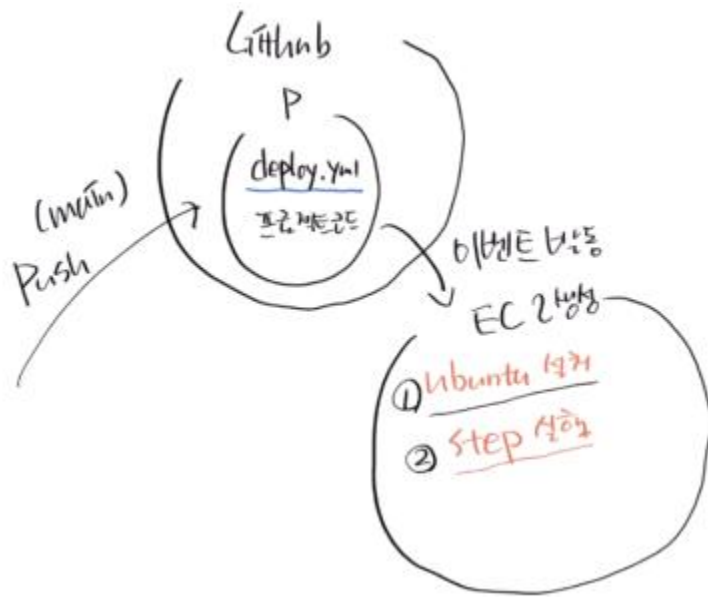
# UTC 변경(한국시간 + 9H)
- name: Get current time
  uses: 1466587594/get-current-time@v2
  id: current-time
  with:
    format: YYYY-MM-DDTHH-mm-ss
    utcOffset: "+09:00"
- name: Show Current Time
  run: echo "CurrentTime=${{steps.current-time.outputs.formattedTime}}"

```

- action은 스크립트의 모음이다.



구글에 `actions/checkout@v3`로 검색해 보면 최신 버전의 라이브러리를 확인 할 수 있다.



## Step

- ① Action → 코드 다운로드
- ② Action → JDK11 설치
- ③ Script → gradlew  
실행 권한 부여
- ④ Script → build  
↓  
test 실행 파일 생성

- ④ 깃 클라켄 와서 → 깃 커밋 → 푸시
- ⑤ 깃 헵에 Actions 로 이동



- ⑥ 클릭하고 빌드 눌러보면 build 라는 job 이 만들어져 있다. 그 job 안에 위의 단계가 하나씩 정리되어 있다.

하나씩 눌러보면 처리가 된다. -- 다 되면 초록색 볼 들어옴.

▶▶▶▶▶▶ CI(Continuous Integration)는 이게 고깃...

이후에 AWS 파일에 jar 파일을 던져야 된다.

## 12장. AWS - 배포 v3 엘라스틱빈스톡 배포(CD)

### ● IAM 생성 - CI 서버에서 AWS로 접근하기 위한 인증서

① 검색 - IAM 검색

② 좌측 액세스관리 → 사용자 → 사용자 생성

- 사용자 이름 : 적당히

- AWS Management Console 에 대한 사용자 액세스 권한 제공 → IAM 사용자를 생성하고 싶음 : 액세스 키로 접근하는 방식 사용함.

- 콘솔 암호 → 자동 생성된 암호

- 사용자는 다음 로그인 시 새 암호를 생성해야 합니다. → 기본 체크 놔둬.

▷ 다음

- 권한 설정 → 직접 정책 연결 클릭 → elastic 검색 → AdministrationAccess-AWSElasticBean 선택

- 아래 다음 버튼 클릭 → 사용자 생성

**암호 검색**

아래에서 사용자의 암호를 보고 다운로드하거나 AWS 관리 콘솔에 로그인하기 위한 사용자 지정을 이메일로 보낼 수 있습니다. 지금이 이 암호를 확인 및 다운로드할 수 있는 유일한 시간입니다.

**콘솔 로그인 세부 정보**

콘솔 로그인 URL  
https://185169865129.signin.aws.amazon.com/console

사용자 이름  
zzzmini

콘솔 암호  
\*\*\*\*\* 표시

이메일 로그인 지침

취소 .csv 파일 다운로드 사용자 목록으로 돌아가기

③ IAM 페이지로 다시 돌아와서 → 좌측 액세스 관리 → 사용자 → 액세스 키 만들기

aws 서비스 검색 [알트+S]

**Identity and Access Management(IAM)**

IAM 검색

대시보드

- 액세스 관리
- 사용자 그룹
- 사용자

**zzzmini 정보**

요약

ARN arn:aws:iam::185169865129:user/zzzmini	콘솔 액세스 MFA 없이 활성화됨	액세스 키 1 액세스 키 만들기
생성됨 September 15, 2023, 11:03 (UTC+09:00)	마지막 콘솔 로그인 안 함	

- ④ 액세스 키 모범 사례 및 대안 → AWS 컴퓨팅 서비스에서 실행되는 애플리케이션 → 다음
- ⑤ 태그 Pass → 액세스 키 만들기

- ▶▶▶▶▶▶▶ 생성 완료
- ▶ 외부로 노출하지 말고...
- ▶ 반드시 csv 파일 다운로드 단추 눌러 다운로드 받아놓고...
- ▶ 안해 놓으면 못찾음.

※ 애들을 깃헙에 등록해 놓자 → 그럼 배포 시 CI 서버에 전달됨. → 그럼 AWS 에 로그인 되는 거지요.

- ▷ AWS\_ACCESS\_KEY
- ▷ AWS\_SECRET\_KEY

## ● IAM 인증서 깃헙에 등록하기

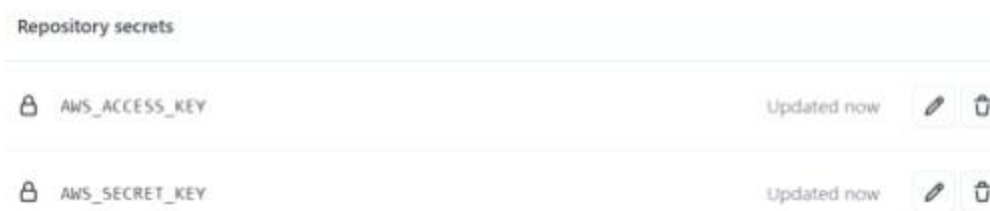
① 깃헙에 와서 → 해당 리포지토리로 이동 → 위쪽 Settings 클릭 → 좌측 메뉴 아래 Secrets and variables → Actions

: 요기 생성

② 위쪽 녹색 → New Repository Secret 클릭

▷ name : AWS\_ACCESS\_KEY → value : Access key ID  
▷ name : AWS\_SECRET\_KEY → value : Secret access key

여기에 생성하면 CI Server로 전송된다.



모두 완료 되었으니 재배포를 해 보자

## ● 재배포하기

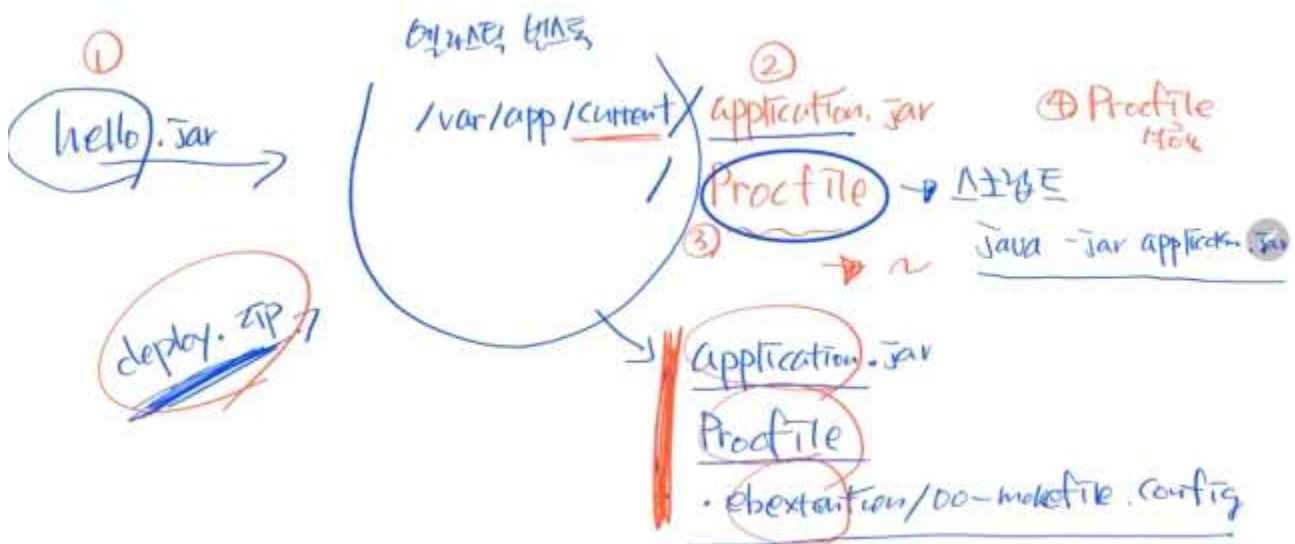
① 다음 거 복붙해서 사용한다.

→ 마지막 스텝이 CD 이다.

```
# EB에 CD 하기 위해 추가 작성
- name: Generate deployment package
  run: |
    mkdir deploy
    cp build/libs/*.jar deploy/application.jar
    cp Procfile deploy/Procfile
    cp -r .ebextensions deploy/.ebextensions
    cd deploy && zip -r deploy.zip .
- name: Deploy to EB
  uses: einaregilsson/beanstalk-deploy@v21
  with:
    aws_access_key: ${ secrets.AWS_ACCESS_KEY }
    aws_secret_key: ${ secrets.AWS_SECRET_KEY }
    application_name: aws-v3-beanstalk-test # 엘리스틱 빈스톡 애플리케이션 이름!
```

```
environment_name: Aws-v3-beanstalk-test-env # 엘리스틱 빈스톡 환경 이름!
version_label: aws-v3- $\{\{\text{steps.current-time.outputs.formattedTime}\}\}$ 
region: ap-northeast-2
deployment_package: deploy/deploy.zip
```

※ run: | 명령어를 여러 줄 적을 수 있다.



```
1 files:
2   "/sbin/appstart":
3     mode: "000755"
4     owner: webapp
5     group: webapp
6     content: |
7       #!/usr/bin/env bash
8       JAR_PATH=/var/app/current/application.jar
9
10      # run app
11      java -Dspring.profiles.active=prod -Dfile.encoding=UTF-8 -jar $JAR_PATH
```





⑤ 로그는 깃헙과 엘라스틱빈스톡에 같이 볼 수 있다.

⑥ 두개의 인스턴스가 더 생기고 새로 생기는 그린에 문제가 없다면, 블루가 종료되고 그린 실행된다.  
F5를 중간에 눌러서 확인해 본다.



⑦ 그림 → 로드밸런서가 그린인스턴스로 변경된다.

아까 양쪽 IP의 내용을 화면으로 확인해 보면 중간에 한개는 새로운 서비스를, 한개는 블루에 있던 서비스를 실행하고 있다는 것을 확인해 볼 수 있다. 롤백 등 여러가지 조치를 통과하면서 시간이 많이 걸릴 수 있다.

⑧ 아마존에서 S3 검색

▶ 좌측 버킷 → 우측 계정 스냅샷 → 버킷을 확인해 보면 여태껏 진행했던 내용들이 저장되어 있다.

▶▶▶▶▶▶ 배포 완료.....

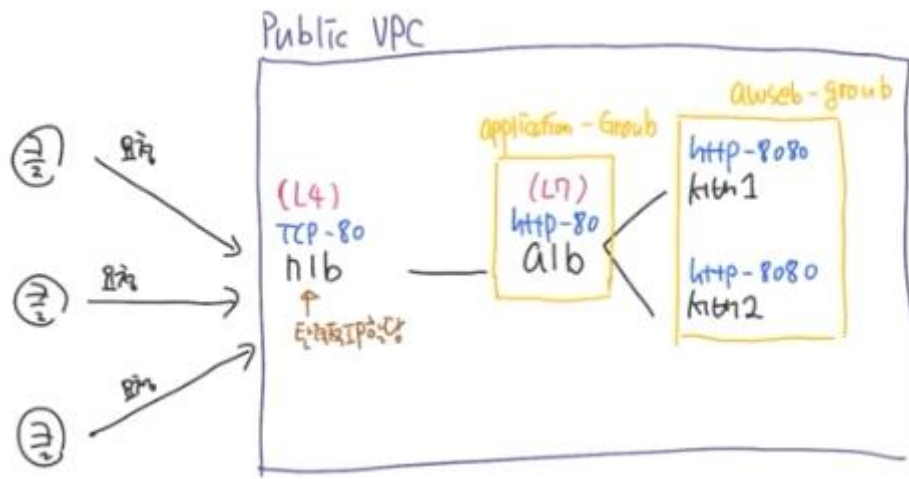
→ 좌측 환경으로 이동. 실행해 보고 사이트 잘 뜨는지 DB 연동되는지 확인도 해야지요...

## 13장. AWS - 배포 v3 네트워크 로드밸런서 고정 IP 설정

### ● GithubAction + Beanstalk + NLB(Network Load Balancer) + 고정IP(탄력적IP 할당)

▷ ALB와 NLB와 차이

- Elastic Load Balancer 는 고정IP를 지원하지 않음.



① 첫째, EC2로 이동

→ 좌측에 탄력적 IP → 생성 → 생성된 탄력적 IP를 로드밸런서에 달거야.

※ 먼저 탄력적 IP를 셋팅해야 로드밸런서에 탄력적 IP를 할당할 수 있다.

② 좌측 로드밸런서 → 우 상단 로드밸런서 생성 → NLB 선택

- 이름 : aws-v3-nlb

- 네트워크 매핑 : north-east-np2 AZ 선택(아무거나 한 개만 선택 가능하다.)

- IPv4 주소 : 탄력적 IP 주소 사용 → 생성한 IP 주소 보임.

③ 리스너 및 라우팅

- 리스너 TCP-80

▶ port 80 → Security Group 에서 Default action 을 Application Load Balancer 로 보내주어야 한다.  
애를 설정하기 위해

바로 밑에 『대상 그룹 생성』 클릭 - 그룹 세부 정보 지정(새창 열림)

- 기본 구성 → 대상 유형 선택 → application load balancer : check

- name : aws-v3-alb

▶▶ 우측 하단 다음 ...

- 대상 등록 → Application Load Balancer 등록

- Choose..... → 새로 만든 거 잡아주고

▶▶ 우측 하단 대상 그룹 생성



신규로 타겟 그룹이 생성된다. 원래 창으로 돌아가서

### ③ 리스너 및 라우팅

- 리스너 TCP-80

▶ port 80 → 우측 새로그침. → 우측 Security Group 에서 aws-v3-alb 보임... 선택

▶ 모든 NLB로 들어오는 요청은 위의 aws-v3-alb로 갈거임.

▶▶▶▶▶ 맨 아래 로드밸런서 생성 버튼 클릭 → 로드 밸런서 보기 (NLB 생성 완료)

이제 접속을 탄력적 IP로 접속하면 됨.....

### 시간이 좀 걸림.....

프로비저닝(준비 중) - 사용자 요구에 맞게 시스템 자원을 할당, 배치, 배포해 두었다가 필요시 시스템을 즉시 사용할 수 있는 상태로 미리 준비해 두는 것을 말함.