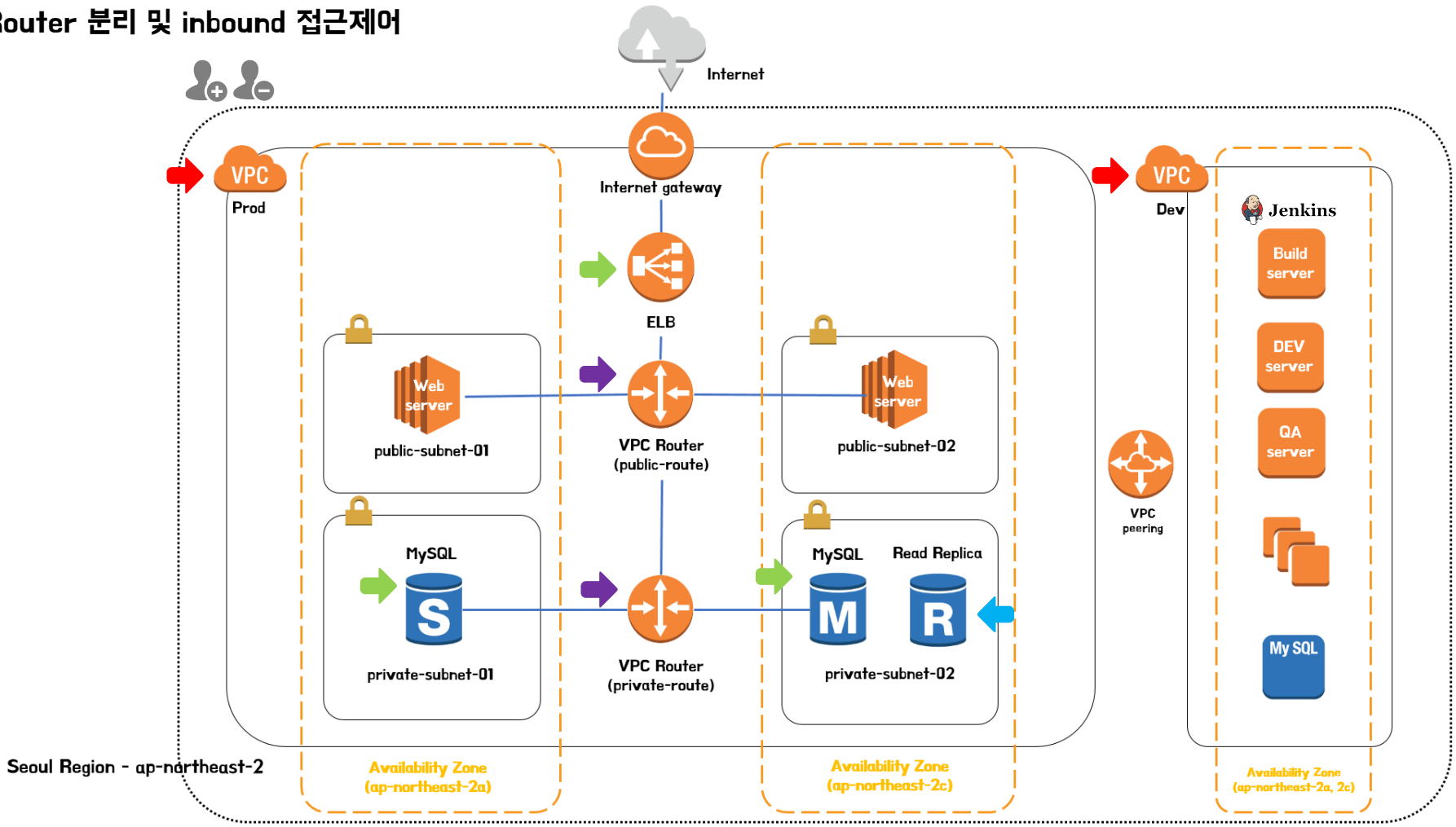
DevOps 의 시작
(Docker + Ansible + Jenkins + AWS 를 이용한 무 중단 배포)

JBUG - 김완철 (데이빗백곰)

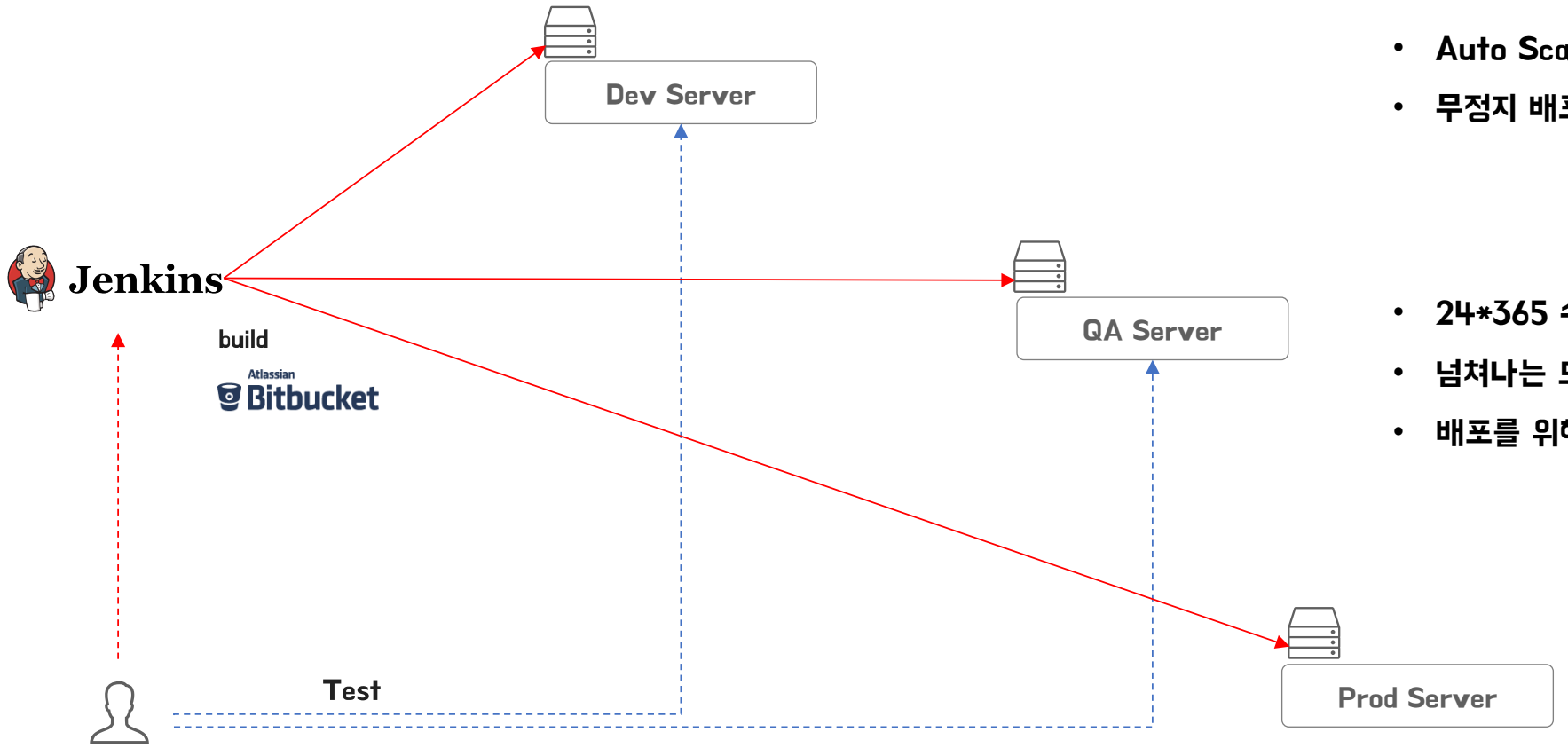
https://github.com/david100gom/docker_hands_on

1. 기본적인 AWS 서버 아키텍처 구성

- Dev(QA) 와 Prod 를 구분하여 VPC 구성
- HA 를 위한 Multi-AZ 구성, RDS M/S 구성 및 Auto Scaling 을 위한 ELB 구성
- Scale out 를 위한 RDS Read Replica 구성
- 기본적인 보안을 위한 public, private Router 분리 및 inbound 접근제어
- IAM 를 통한 Role 및 권한 관리 등등



2. 기본적인 CI/CD 구성



- Jenkins 내에서 git 에서 checkout 한 소스를 컴파일
- 각 DEV, QA 서버 및 PROD 서버 배포

- Auto Scaling 대응 미흡
- 무정지 배포 대응 미흡

- 24*365 수동 모니터링
- 넘쳐나는 모니터링 메일 스레드
- 배포를 위해 새벽에 출근

3. 기존 CI/CD 문제점 해결방안 고민

- AWS Auto Scaling 설정
- 손쉬운 배포를 위한 Docker 도입
- 서버 관리를 위한 Ansible 도입

AWS Auto Scaling

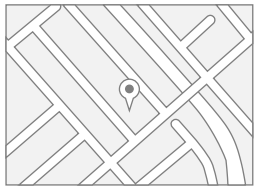
- AWS Auto Scaling 설정 구성하면 추후 관리해야 하는 서버가 늘어남

- 소스 배포의 안정적인 처리를 위한 Docker

- Docker Swarm, Rancher, Kubernetes 등등

- 많아진 서버 관리를 위한 Ansible

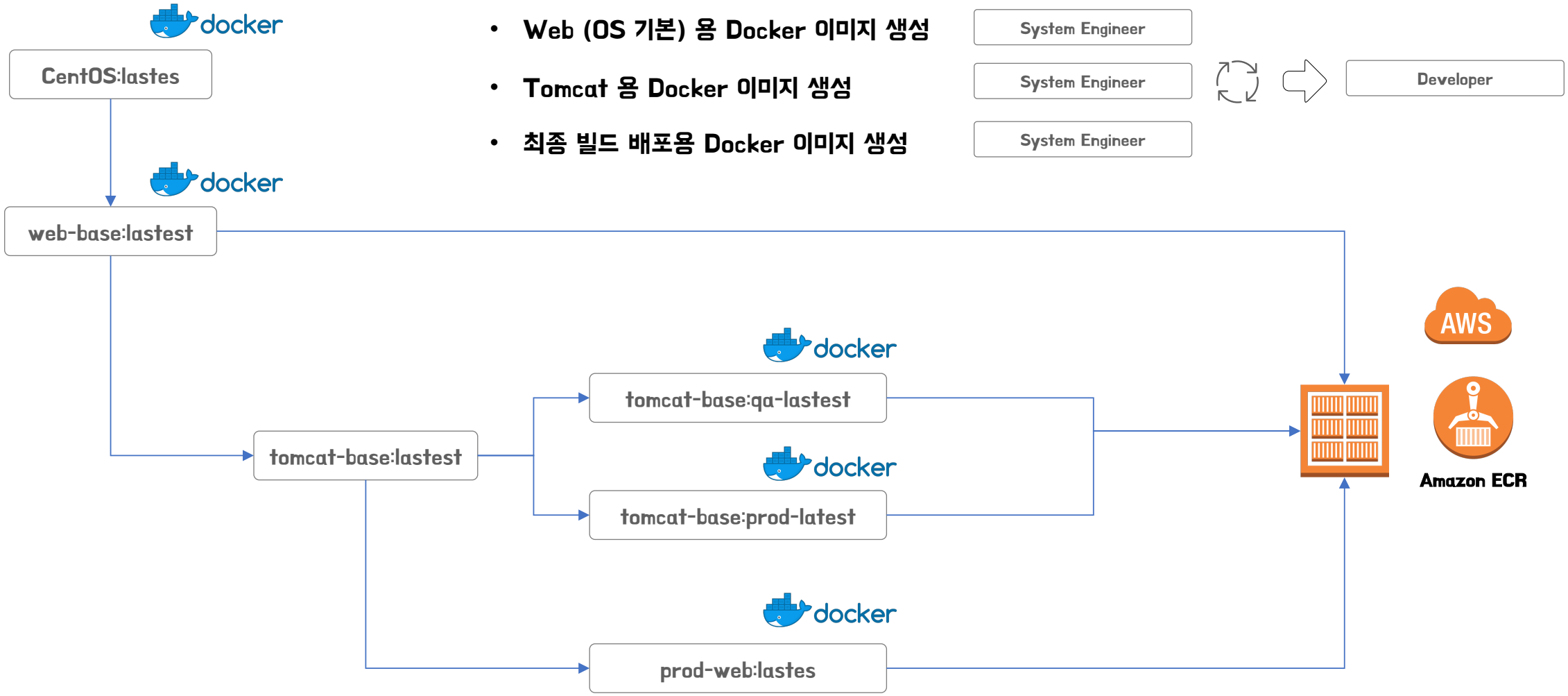
- yml, playbook, tower 등등



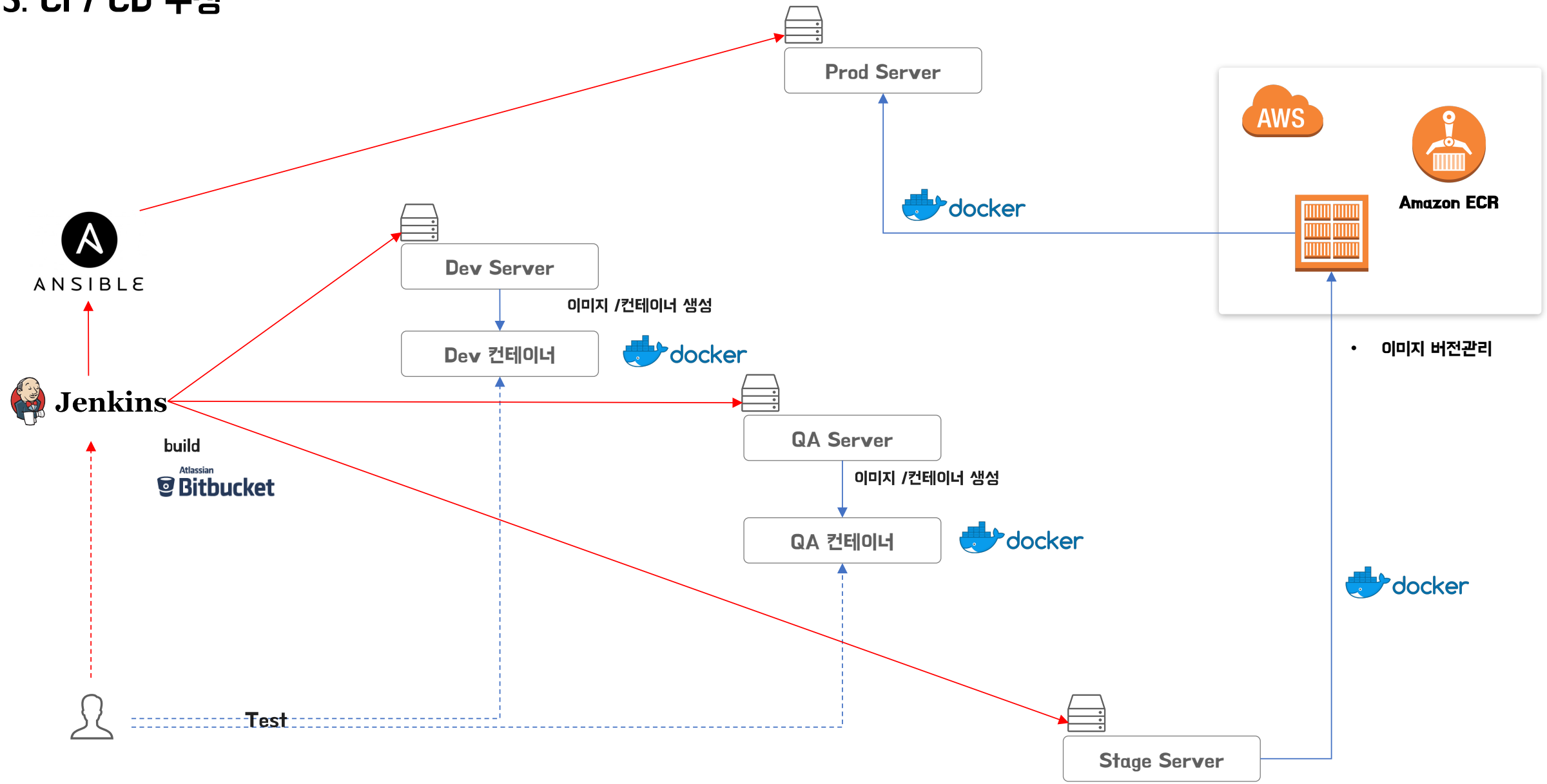
- 관련 기술이 많아 목적 상실 위험

Step by Step 고도화

4. Docker 개발, 서비스 이미지 생성 구성



5. CI / CD 구성


Prod Server
AWS
Amazon ECR
docker
ANSIBLE
Dev Server
이미지 /컨테이너 생성
Dev 컨테이너
docker
Jenkins
build
Atlassian
Bitbucket
QA Server
이미지 /컨테이너 생성
QA 컨테이너
docker
• 이미지 버전관리
docker
Test
Stage Server

6. 빌드/배포 상세 flow

Jenkins

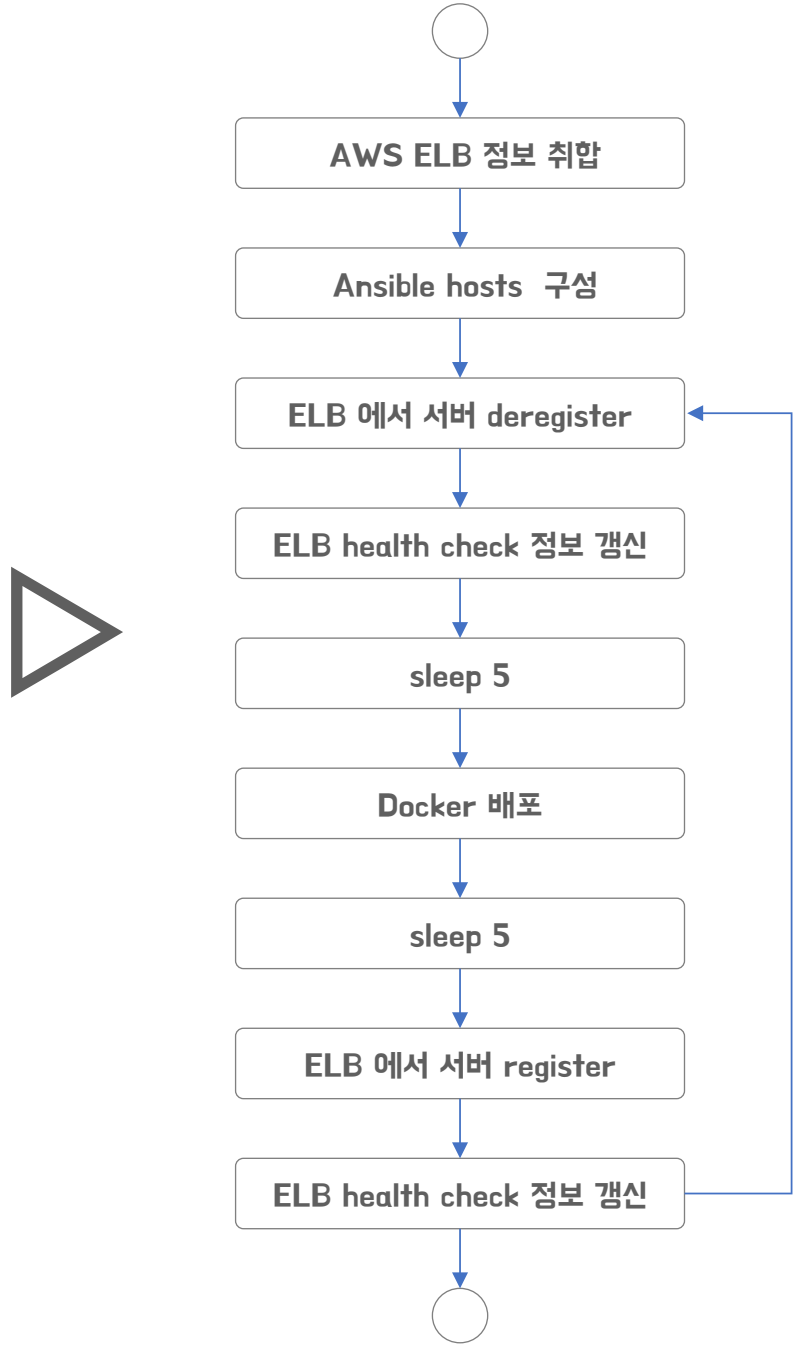
Shell Script

+

AWS CLI

+

ANSIBLE



- 최대한 down time을 회피하기 위한 flow 구성
- ELB 뒤에 HAProxy 가 있다면 HAProxy 컨트롤 고려
- 서비스 목적에 따라 무중단에 대한 개념 및 위험요소 다양
- 다양한 방법 존재 : Blue-Green 배포 등등

7. 추후 고려사항

- 서비스 목적에 따른 DevOps 아키텍처 구성 포인트
- Over Engineering 에 따른 관리 포인트 문제 및 리소스 부족
- Docker, Ansible 등 최신 기술에 대한 내재화에 대한 부담
- 전체 DevOps 아키텍처에 대한 모니터링
- 개발조직의 DevOps 에 대한 마인드 및 기술 수준 평준화

AWS CLI

Amazon ECR

Python (boto)

감사합니다.