Deployment Strategy

15th 이미나

✳ Deployment Strategy

새 버전의 애플리케이션을 제공하는 방법

- 최근에는 마이크로 서비스 환경의 개발이 늘어나면서 서비스를 더 작게 만들고 자주 배포를 하는 방식으로 변화되었다.
- 사용자 경험, 테스트 안정성, 리소스 등 여러 고려 사항에 따라 적절한 배포 전략을 사용할 수 있다.

✳ Deployment Strategies

1. Big-Bang Deployment
2. Blue/Green Deployment
3. Rolling Deployment
4. Ramped Slow Rollout
5. Canary Deployment
6. Shadow Deployment
7. A/B Testing

✳ Big-Bang Deployment

특징

- 애플리케이션의 대부분을 한 번에 업데이트 하는 방식
- 오랜 기간동안 개발 및 테스트를 수행
- 서버를 내리고, 버전 업데이트 후 다시 서버를 올리는 단순한 방법

장단점

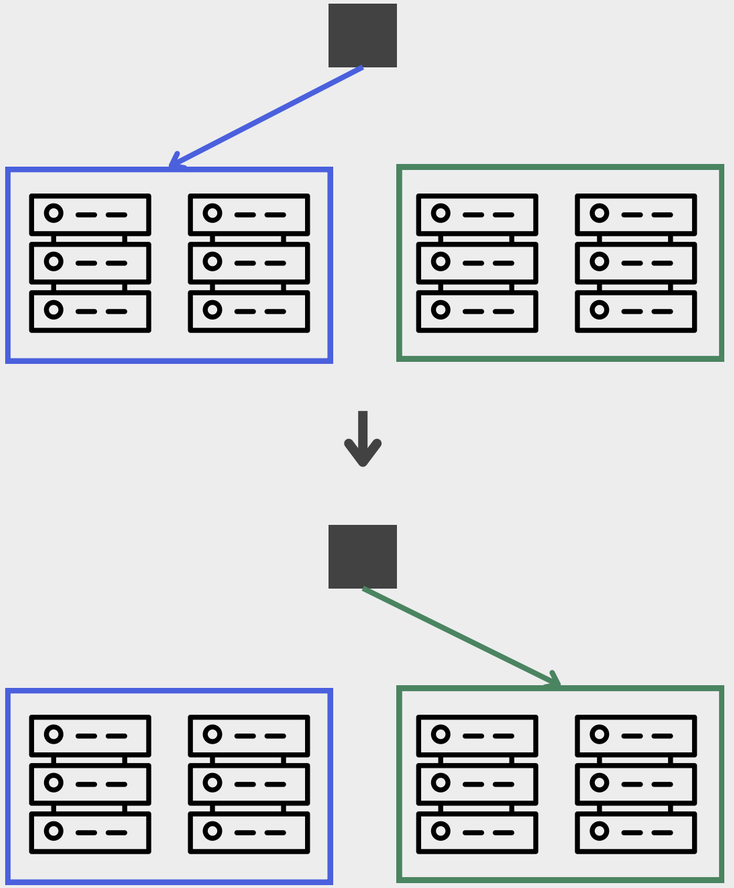
- 성공만 하면 빠른 시간내로 배포 완료 가능
- 모든 사용자가 동일 버전을 사용할 수 있음

- 서버를 내렸다 다시 올리기 때문에 다운타임 발생
- 롤백이 어려움
- 테스트 기간을 가지기 어려움

✳ Blue/Green Deployment

특징

- 블루(기존), 그린(신규)
- 기존 버전과 동일한 신규 버전 인스턴스를 만들어 로드밸런서를 통해 한 번에 모든 트래픽을 신규 버전쪽으로 전환



장단점

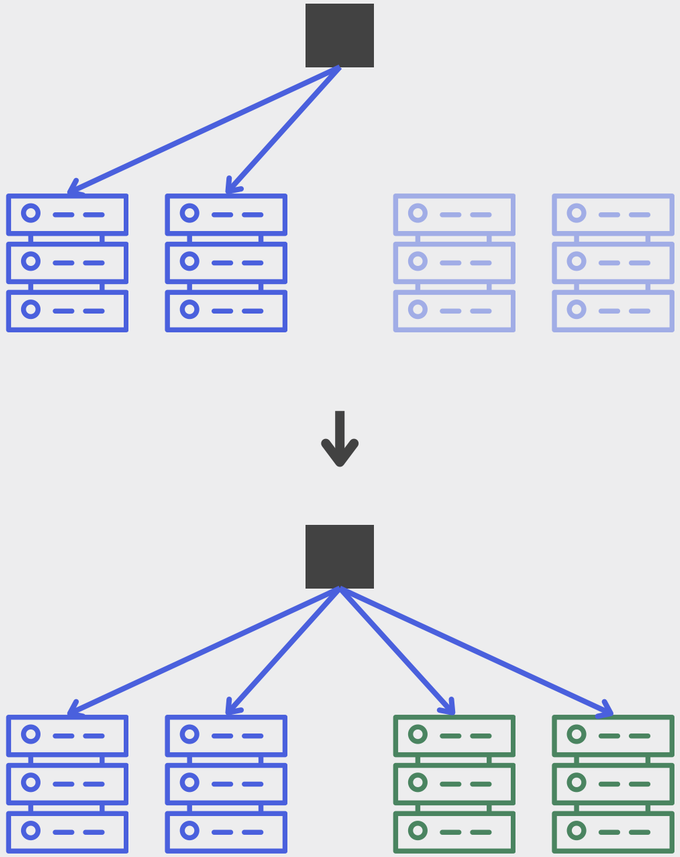
- 기존 버전의 인스턴스를 다음 배포에 재활용 가능
- 다운타임이 없고 롤백이 쉽다

- 리소스가 2배 필요
- 트래픽 스위칭을 해야하기 때문에 설계가 복잡해진다

✳ Rolling Deployment

특징

- 버전을 점진적으로 업데이트 하는 방식
- 무중단 배포 방법 중 하나
- 배포 초기에 새로운 버전의 문제점을 조기 발견할 수 있음
- 유사한 방법으로 Ramped Slow Rollout이 있음



장단점

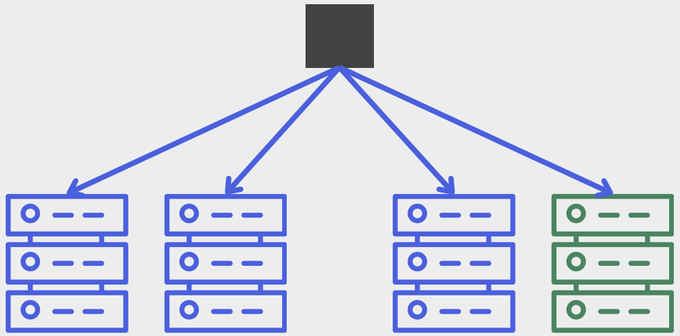
- 다운타임 없음
- 롤백이 쉽다

- 새 버전 배포시 인스턴스의 수가 감소해 트래픽이 몰릴 수 있음
- 배포 진행 중 기존 버전과 새 버전이 공존해서 호환성 문제 발생 가능

✳ Canary Deployment

특징

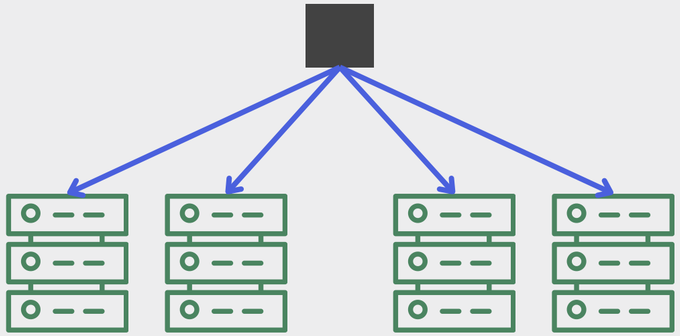
- 새 버전을 완전히 배포하기 전 일부 사용자에게 배포
- 기존 버전과 새 버전 간 트래픽 분할
- 배포 초기에 새로운 버전의 문제점을 조기 발견할 수 있음



장단점

- 모든 사용자에게 새 버전을 제공하기 전 안정성 확보
- A/B 테스트와 성능 모니터링에 유용

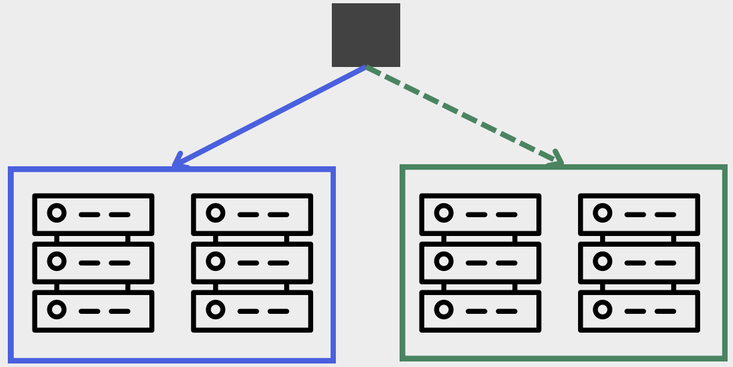
- 구현의 어려움
- 호환성 문제 발생 가능



✳ Shadow Deployment

특징

- 신규 버전에 실제 트래픽을 반영해 테스트
- 사용자에게 신규 버전을 미리 노출시키지 않음
- 테스트 중심의 배포 방식



장단점

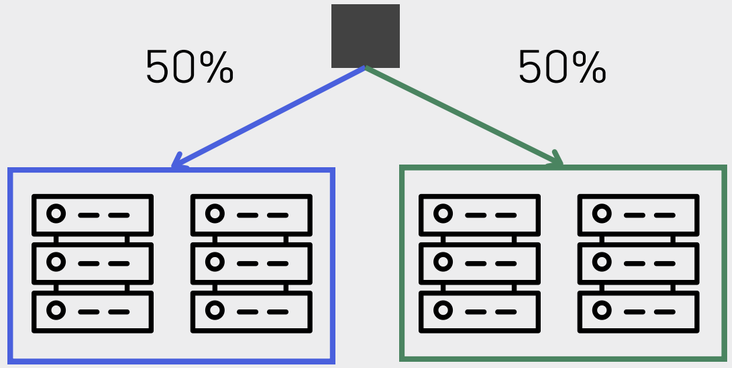
- 테스트 및 검증을 통해 안정성 향상
- 실제 트래픽으로 테스트 가능

- 많은 리소스 필요

✳ A/B Testing

특징

- 서로 다른 애플리케이션을 동시에 배포
- 트래픽 비중을 컨트롤 해 어떤 버전이 좋은지 테스트
- 안전한 배포보단 테스트에 초점



✳ 무중단 배포 사례 1

데브시스터즈 Blue-Green + Canary 도입

기존엔 K8S에서 Rolling 방식을 사용하고 있었으나, 배포가 몰리거나 핫픽스를 해야 하면 모두가 힘든 상황 발생

새로운 배포 전략을 위한 요구사항

- 롤백이 쉬워야 한다.
- 신 서버에 대한 모니터링 및 디버깅이 가능해야 한다
- 신 서버의 Warm-Up이 가능해야 한다.
- 업데이트에 걸리는 시간이 단축되어야 한다.

파이프라인

1. 구 서버와 같은 스펙으로 신 서버를 프로비저닝 (B/G)
2. 약간의 트래픽을 신 서버로 보냄 (Canary)
3. 신 서버를 집중 모니터링, 디버깅 후 트래픽을 늘림
4. 신 서버가 안정되면 구 서버 종료

✳ 무중단 배포 사례 2

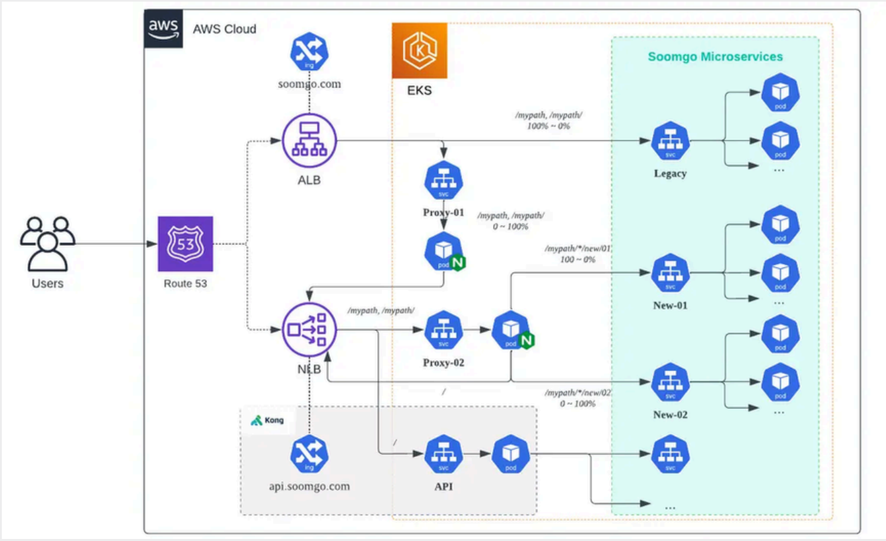
숨고 Canary 도입

Blue/Green 방식으로 MSA 환경에서 안정성을 유지하고 있었으나, 레거시 서비스를 제거하는 미션인 테라포밍 프로젝트의 원활한 진행을 위해 테라포밍 프로젝트를 대상으로 Canary 전략 추가 채택

ALB, NLB로 서비스하고 요청 경로별 카나리 배포 전략을 이용할 수 있도록 도입
Nginx를 프록시 서버로 활용

적용 과정

1. 카나리 배포 대상 선정 및 서비스 개발
2. 코드 리뷰 후 배포 준비
3. 카나리 배포 진행
4. 카나리 대상 서비스 상세 모니터링
5. 신규 버전 서비스의 상태 판단 후 비율 조정해 2~4 과정 반복



✳ 번외 : PM2

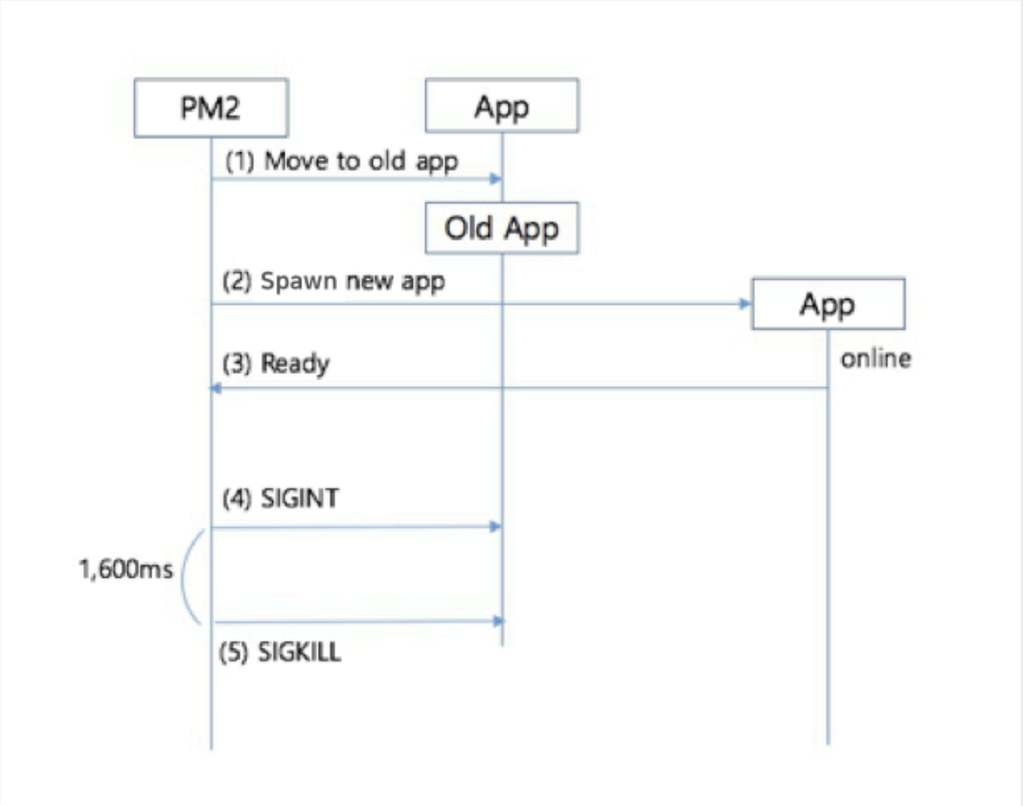
pm2는

- Node.js 프로세서를 관리해주는 패키지
- reload 기능을 통해 무중단 배포 가능

프로세스 재시작 과정

- 기존 0번 프로세스를 _old_0 프로세스로 옮김
- 새로운 0번 프로세스 생성 후 준비 완료되면 ready 전송
- 마스터 프로세스는 필요없어진 _old_0에 SIGINT
- 만약 시간이 지나도 종료되지 않으면 SIGKILL을 보내 강제 종료
- 총 프로세스 개수만큼 반복

→ Rolling 방식과 유사



✳ Reference

https://www.youtube.com/watch?v=eyzzwHcAZSYh
https://llshl.tistory.com/47h
ttps://facerain.github.io/canary-deployment/canary-deployment/
https://devops.com/what-is-a-shadow-deployment/
https://tech.devsisters.com/posts/blue-green-canary-deployment/
https://blog.soomgo.com/posts/6673bb8c52107866fb86a785?source=post_list
https://engineering.linecorp.com/ko/blog/pm2-nodejs
https://rb.gy/m31j65