Rocket.Chat 클라우드 배포 **- FinOps** 최적화

프로젝트 개요

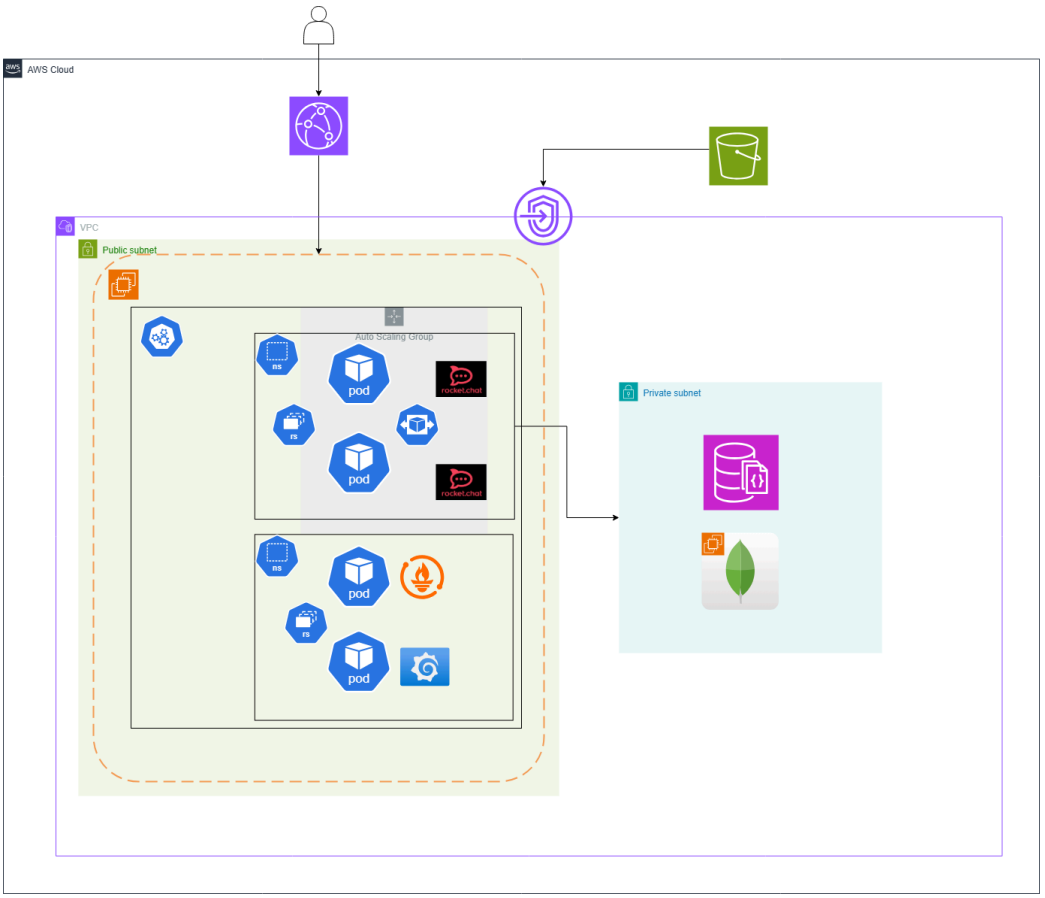
실시간 채팅앱을 AWS 환경에서 클라우드 배포 ,

다양한 구성 변경에 따른 성능과 비용 최적화를 확인하는 것을 목표로 진행함.

주요 목적

- EC2 + k3s 환경에서 Rocket.Chat 모놀리식 파드를 배포하고, k6를 이용해 부하 테스트 수행
- CloudFront 적용 전/후 성능 및 비용 비교
- 데이터 저장소 변경(MongoDB → DocumentDB) 영향 분석
- HPA 적용 시 세션 저장 방식(MongoDB → Redis)에 따른 확장 효율 비교

아키텍처



주요 시나리오

1. **EC2/k3s** 환경에서 **Rocket.Chat** 배포 후 **k6** 부하 테스트 수행
    - 초기 배포 환경에서 부하 및 리소스 사용량 모니터링
    - Prometheus + Grafana로 실시간 지표 확인
2. **CloudFront** 적용 전/후 비교
    - CloudFront를 통해 정적 파일 캐싱 적용
    - 네트워크 지연 및 트래픽 비용 변화 분석
3. **MongoDB → DocumentDB** 전환 비교
    - Private Subnet 내 EC2 MongoDB 운영
    - DocumentDB 전환 후 성능 및 관리 편의성 비교
4. **HPA** 적용 및 세션 저장 방식 비교
    - 모놀리식 Rocket.Chat 파드 오토스케일 적용
    - 세션 저장 MongoDB vs Redis 비교
    - 부하 분산 효율, 확장성, 안정성 확인

결론

- CloudFront 적용 시 정적 파일 접근 지연 최소화 및 비용 절감 확인
- DocumentDB 전환은 운영 관리 편의성 향상, EC2 MongoDB 대비 안정성 증가
- HPA 적용 시 세션 저장 방식을 Redis로 변경하면 부하 분산 효율이 개선됨
- 전체적으로 FinOps 관점에서 비용 효율과 성능 최적화를 동시에 달성 가능함