대규모 사용자들을 감당하기 위한 서비스 설계

대규모 서비스 개발 및 운영을 위한 실전 엔지니어링 워크숍

목차

대규모 서비스 개발 및 운영을 위한 실전 엔지니어링 워크숍

1. 워크숍 개요

목적

- **대규모 부하를 견딜 수 있는 서비스 구현**
- **실전 엔지니어링 경험 축적**
- **팀원들 간의 협업 및 능력 향상**
- **면접에서 점수 딸 수 있는 스토리 만들기**
- **다른 곳에서는 할 수 없는 경험해보기**

학습 목표

- **실무 경험 축적**
  - 대규모 서비스 운영 기술 직접 적용
  - 부하 테스트를 통한 성능 최적화 경험
- **협업 능력 향상**
  - 다양한 전문 분야의 팀원들과 협업
  - 문제 해결 능력 및 커뮤니케이션 스킬 강화
- **엔지니어링 역량 증진**
  - 각종 기법, 도구, 서비스 사용 경험
  - 클라우드 서비스 활용 능력 향상

2. 프로젝트 개요

수행 배경

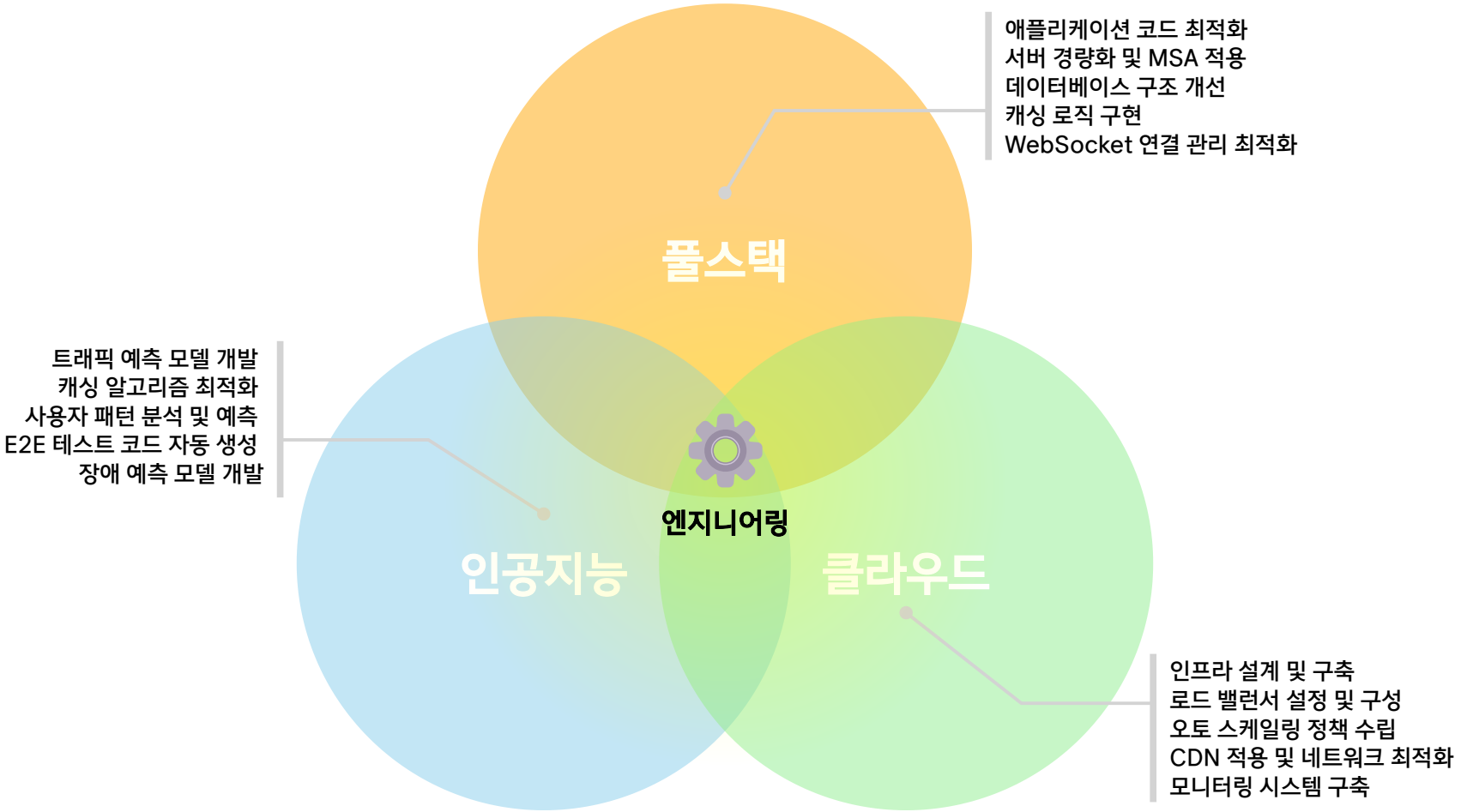
- **현실 세계의 대규모 서비스 운영 경험 부족**
  - 이론보다 실전에 가까운 환경에서의 학습 필요
  - 팀 단위의 협업을 통한 실력 향상 도모
- **생성 AI가 작성한 코드의 리뷰와 관리 능력 필요성 증가**
  - ChatGPT, Claude 등이 생성한 코드의 품질 검증
  - AI 생성 코드와 기존 코드베이스의 통합 전략 수립
- **레거시 시스템 리엔지니어링은 피할 수 없는 과제**
  - 기존 시스템의 현대화 및 개선 작업 경험 필요
  - 레거시 코드 분석과 리팩토링 능력 배양

2. 프로젝트 개요

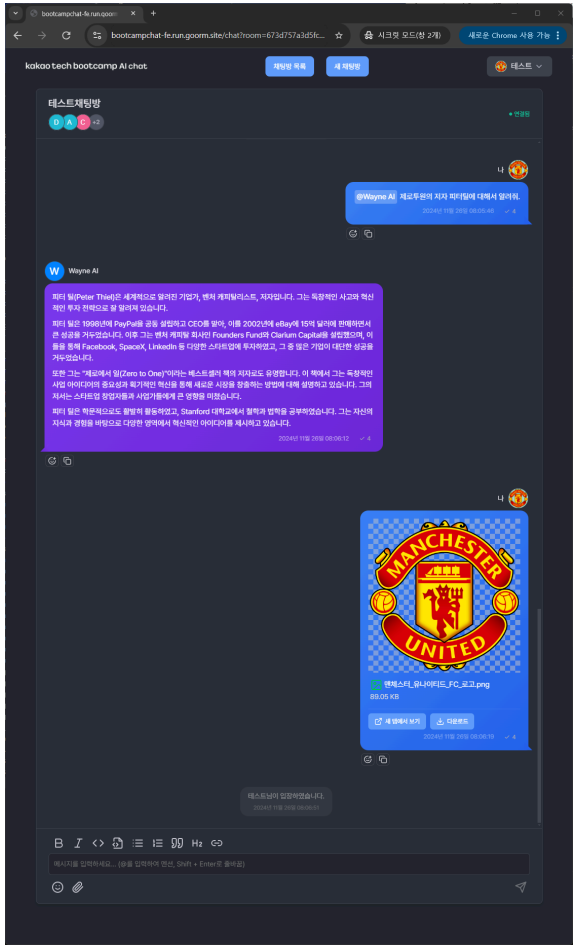
채팅 서비스에 대한 부하테스트

- **동시 접속자 수 증가** (변경될 수도 있음)
  - 초기 100명에서 시작하여 매 30초마다 100명씩 증가
  - 최대 3,000명까지 순차적 증가 목표
  - 각 사용자당 최소 1개 이상의 WebSocket 연결 유지
- **제한된 리소스로 최대한의 효율 달성**
  - 최대 30대의 t3.small 인스턴스 사용
  - 다양한 아키텍쳐 시도

3. 팀 구성 및 역할 분담


애플리케이션 코드 최적화
서버 경량화 및 MSA 적용
데이터베이스 구조 개선
캐싱 로직 구현
WebSocket 연결 관리 최적화
풀스택
트래픽 예측 모델 개발
캐싱 알고리즘 최적화
사용자 패턴 분석 및 예측
E2E 테스트 코드 자동 생성
장애 예측 모델 개발
엔지니어링
인공지능
클라우드
인프라 설계 및 구축
로드 밸런서 설정 및 구성
오토 스케일링 정책 수립
CDN 적용 및 네트워크 최적화
모니터링 시스템 구축

4. KakaoTechBootcamp AI Chat 서비스 소개



- **실시간 통신 기능 구현** (Socket.io 사용)
  - 사용자 간 메시지를 **실시간**으로 주고받을 수 있도록 **Socket.io**를 활용하여 양방향 통신 시스템 구축
- **사용자 인증 및 관리 체계 구축**
  - **JWT(Json Web Token)**를 사용하여 **사용자 인증**과 **세션 관리**를 구현하여 보안성 강화
- **멀티미디어 전송 기능 추가**
  - 채팅 내에서 **이미지**, **파일** 등의 **멀티미디어 콘텐츠 전송** 기능 추가
- **채팅방 생성 및 그룹 채팅 기능 구현**
  - 사용자가 직접 **채팅방을 생성**하고, 다수의 참여자와 함께하는 **그룹 채팅** 지원
- **멘션 기능 구현**
  - 채팅 중 특정 사용자를 **@아이디** 형태로 **멘션**하여 상대방에게 알림을 보내는 기능 추가
- **Redis 도입**
  - **Redis**를 활용하여 **세션 관리**, **실시간 데이터 캐싱**, **메시지 큐 처리** 등을 통한 성능 최적화
- **읽음 처리 기능 추가**
  - 상대방이 메시지를 읽었는지 확인할 수 있는 **읽음 확인** 기능 추가로 사용자 경험 향상
- **이모지 리액션 기능 구현**
  - 메시지에 대한 반응을 표현할 수 있는 **이모지 리액션** 기능 추가
- **프로필 이미지 변경 기능 구현**
  - 사용자가 **프로필 이미지를 업로드 및 변경**할 수 있어 개인화된 계정 제공
- **회원 관리 체계 구축**
  - 관리자 전용 페이지에서 회원 정보 관리, **권한 설정**, **계정 제재** 등의 기능 수행
- **중복 로그인 방지 기능 구현**
  - 동일한 계정으로 여러 곳에서 동시에 로그인할 수 없도록 **중복 로그인 방지** 기능 추가로 보안성 강화
- **99% 생성형 AI로만 개발**
  - 생성형 AI로만 개발되었음을 인지하고, **모르는 사람이 작성한 코드, 혹은 AI가 작성한 코드**라는 점을 바탕으로 최적화 시도

4. KakaoTechBootcamp AI Chat 서비스 소개

서비스 시연

https://ktb-chat-test.goorm.team/

E2E 테스트 시연

`@jerry.kim`

4. KakaoTechBootcamp AI Chat 서비스 소개

기술 스택

- **프론트엔드**: React
- **백엔드**: Node.js (Express.js), Socket.IO
- **데이터베이스**: MongoDB
- **캐싱**: Redis
- **기본 앱 코드**: 모든 팀에게 동일한 채팅 애플리케이션 코드 제공
  - https://github.com/goorm-dev/ktb-BootcampChat
- **기본 테스트 코드**: 모든 팀에게 동일한 부하 테스트 코드 제공
  - https://github.com/goorm-dev/ktb-stresstest

5. High Availability

정의

- 시스템이 중단 없이 지속적으로 운영될 수 있는 능력
- 장애 발생 시에도 서비스의 연속성을 보장하는 방식
- 99.9% 이상의 가용성을 목표로 하는 시스템 설계 철학

필요성

- 서비스 중단으로 인한 비즈니스 손실 방지
- 사용자 경험 및 신뢰도 향상
- 시스템 안정성 확보
- 장애 복구 시간 최소화

핵심 원칙

- 단일 장애점 제거 (No Single Point of Failure)
- 장애 감지 및 자동 복구
- 확장성과 유연성 확보
- 데이터 정합성 보장

5. High Availability

주요 HA 구성 요소

- **로드 밸런싱**
  - 로드 밸런싱을 통한 트래픽 분산
  - 세션 지속성(Session Affinity) 관리
  - Health Check 및 장애 노드 자동 제외
- **데이터 복제**
  - MongoDB Replica Set 구성
  - Redis Sentinel을 통한 자동 장애 조치
  - 데이터 백업 및 복구 전략
  - 실시간 데이터 동기화
- **서비스 이중화**
  - 다중 가용 영역(Multi-AZ) 구성
  - Active-Active 구조 설계
  - 무중단 배포 전략
  - 장애 격리(Failure Isolation)
- **자동화된 복구**
  - Auto Scaling 구성
  - 서비스 헬스 체크 자동화
  - 장애 감지 및 자동 복구
  - 블루-그린 배포 전략
- **모니터링 및 알림**
  - 실시간 성능 모니터링을 통한 메트릭 수집
  - 임계치 기반 알림 설정
  - 로그 집중화 및 분석

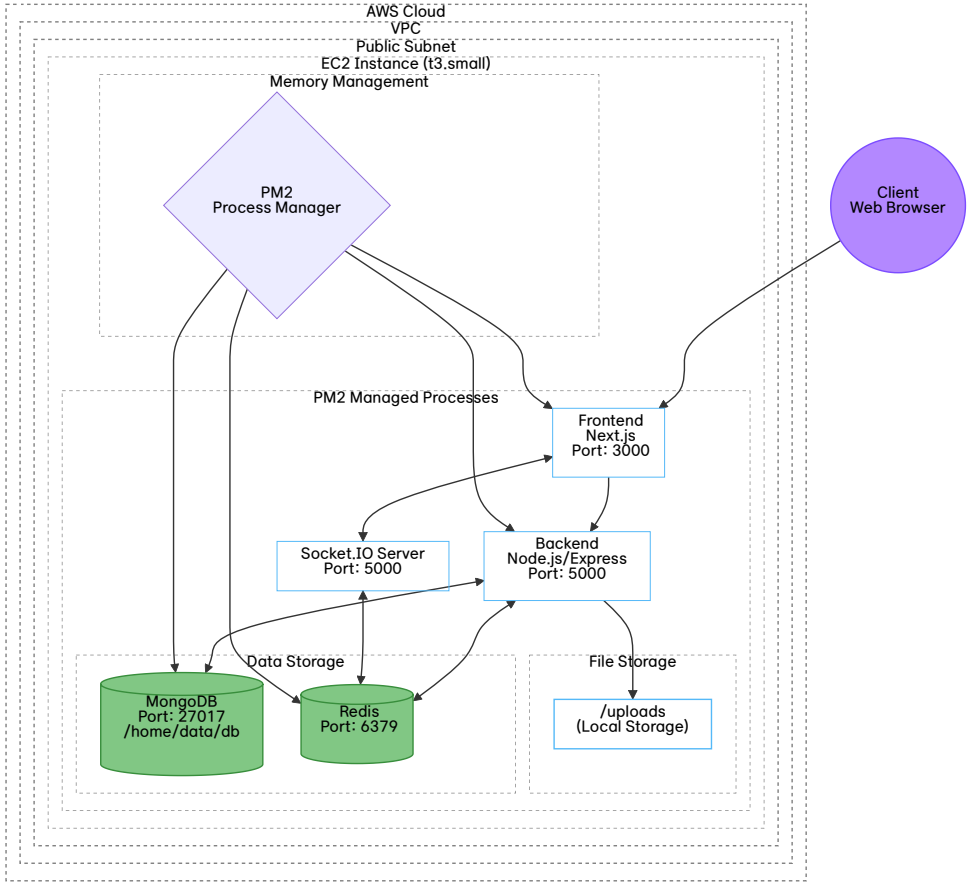
6. 워크샵 상세 내용

시도해볼 HA 향상 기술

- **로드 밸런싱** (NGINX, AWS ELB)
- **오토 스케일링** (AWS Auto Scaling)
- **데이터베이스 캐싱** (Redis)
- **CDN 활용**
- **서버 경량화** (Docker 컨테이너화)
- **MSA 적용**
- **메시지 큐 활용** (RabbitMQ, Kafka)
- **비동기 프로세싱 구현**

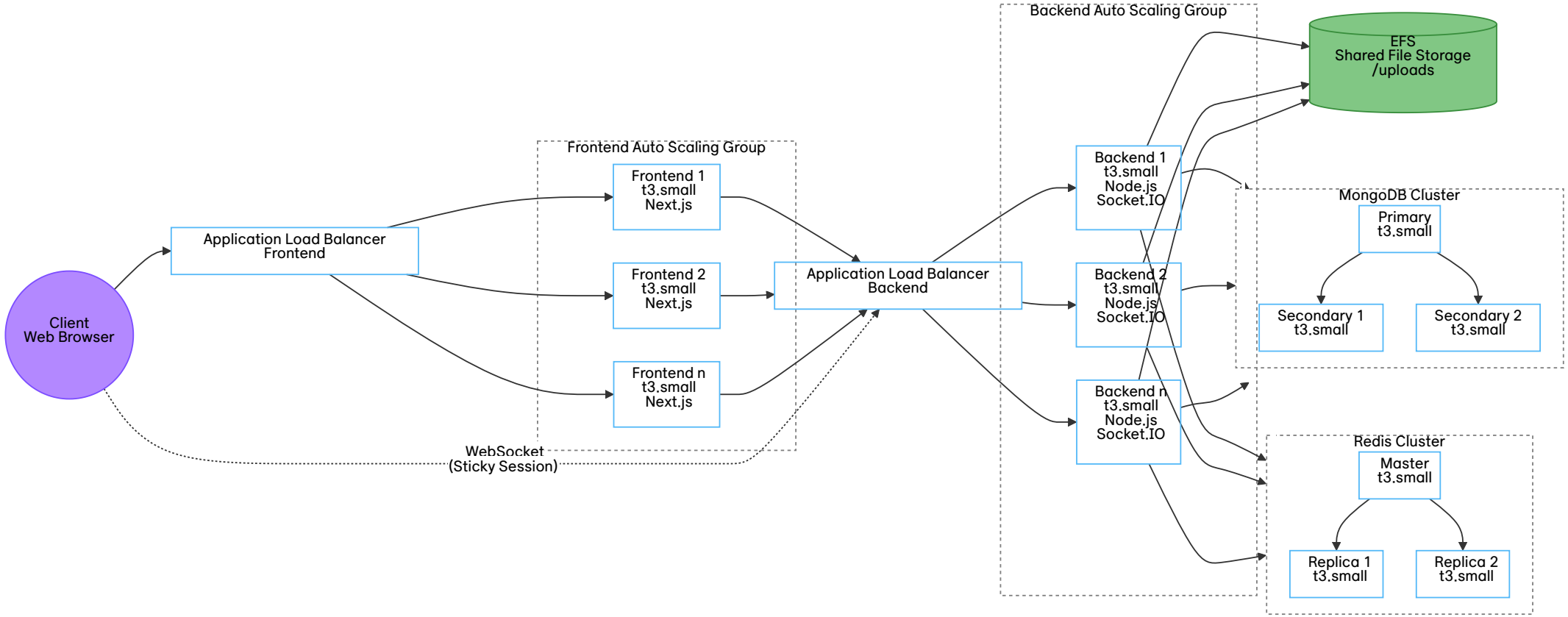
6. 워크샵 상세 내용

기본 아키텍쳐



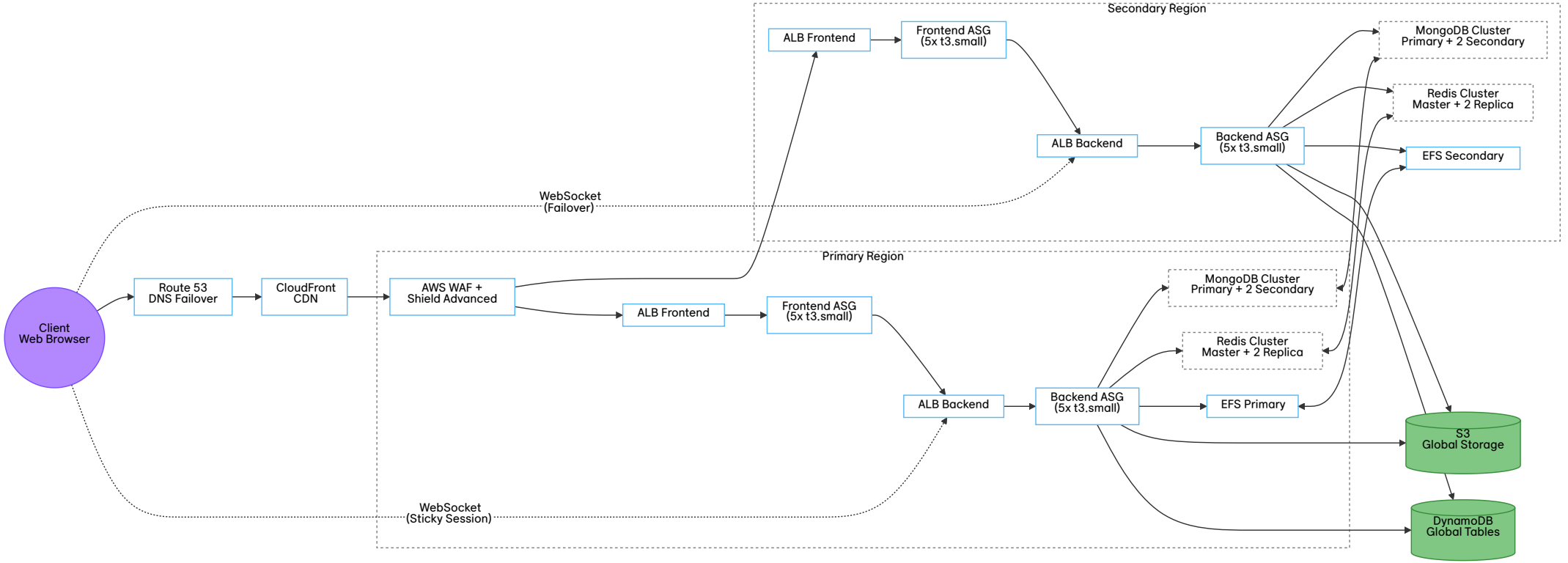
6. 워크샵 상세 내용

아키텍쳐 예시 1 (10대 정도 사용, 실제로 이렇게 하실 수는 없음)



6. 워크샵 상세 내용

아키텍쳐 예시 2 (20대 정도 사용, 실제로 이렇게 하실 수는 없음)



7. 규칙

- chat.goorm-ktb-001.goorm.team ~ chat.goorm-ktb-022.goorm.team 으로 제공
- AWS 계정은 팀장에게 개별 전달

공정한 경쟁을 위해	폭넓은 성장 경험을 위해	기본 제공 사항
프로젝트 관련 Q&A는 팀당 3회로 제한	애플리케이션 코드 수정 자유	팀당 AWS 계정 1개 부여
실전 테스트는 팀당 1회로 제한	외부 라이브러리 사용 자유	팀당 도메인 1개 부여
t3.small 인스턴스만 사용 가능 (최대 30대)	아키텍처 변경 자유	기본 모니터링 대시보드는 클라우드 워치로
총 AWS 사용 비용 $300 이내로 제한	테스트 시나리오 수정 자유	초기 애플리케이션 코드 제공
외부 서비스는 AWS 제품군으로만 제한	모니터링 도구 선택 자유	프로젝트 진행 관련 Q&A는 자유롭게 하되 답변은 전체 공개

- 실전 테스트와 Q&A는 목요일 오후에 몰리기 전에 해주시길 당부

7. 규칙

도메인 구성

- 팀별 서브도메인: chat.goorm-ktb-001.goorm.team ~ chat.goorm-ktb-022.goorm.team
- Route53 설정 완료
- ACM 인증서 발급 및 LB 설정 가능

AWS 계정 권한

- EC2
- Elastic Load Balancing
- Auto Scaling
- Route 53
- ACM
- S3
- CloudFront
- CloudWatch

8. 채점 기준

- 먼저 접속 불가해진 쪽이 패배
- (30초 이내 범위에서) 동시에 접속 불가해진 것으로 파악되었을 때는 무승부
- 무승부 시에는 결과 레포트를 바탕으로 채점
  - 성공/실패 개수, response_time나 session_length 값들로 판단

9. 일정 및 세부 계획

※ 변동될 수도 있습니다.

일차	시간	내용
1일차	14:00 – 18:00	기술 교육 및 오리엔테이션
2일차	09:00 – 18:00	E2E 테스트 작성 강의 및 실습, 팀별 엔지니어링 작업 1일차
3일차	09:00 – 18:00	팀별 엔지니어링 작업 2일차
4일차	09:00 – 18:00	팀별 엔지니어링 작업 3일차
5일차	11:00 – 12:30	1라운드(26팀)
	12:30 – 13:30	2라운드(16팀)
	13:30 – 14:30	점심 식사
	14:30 – 15:15	3라운드(8팀)
	15:15 – 15:45	준결승(4팀)
	15:45 – 16:15	결승(2팀)
	16:15 – 17:15	시상 및 마무리

9. 일정 및 세부 계획

토너먼트 진행 방식

- **22팀** 동시에 부하 테스트 시작
- 상위 **16팀** 선정 후 **1시간** 엔지니어링 시간 부여
- **16팀 → 8팀**: 재테스트 후 상위 8팀 선정
- **30분** 엔지니어링 시간 후 재테스트
- **8팀 → 4팀**: 상위 4팀 선정
- **15분** 엔지니어링 시간 후 재테스트
- **4팀 → 2팀**: 상위 2팀 선정
- **5분** 엔지니어링 시간 후 최종 테스트
- **우승 팀** 선정

(토너먼트 대진표는 당일에 추첨)

10. 시상

순위	상금	상격
대상	30만원 (빕스 상품권)	카카오 대표이사 상
최우수상	20만원 (빕스 상품권)	구름 대표이사 상
우수상	카카오 굿즈	구름 대표이사 상
버그픽스상	10만원 (빕스 상품권)	구름 대표이사 상
기능혁신상	10만원 (빕스 상품권)	구름 대표이사 상

- 수상자들에게는 우수 수료생 평가 시 가점 부여 예정
- 버그픽스상은 가장 많은 버그 레포트와 픽스를 한 팀에 부여
- 기능혁신상은 참신한 신규 기능을 1개 이상 개발하여 추가한 팀에 부여

감사합니다!

Q&A

질문이 있으시면 자유롭게 해주세요.