1. 파트1: 환경 자동화
   1. 수동 구성의 문제점
      1. 수동 프로세스: 관리 콘솔을 통해 AWS 환경을 구성하는 것
      2. 문제점 및 우려 사항
         1. 안정성
         2. 재현가능성
         3. 설명서
   2. 모범 사례
      1. 환경 자동화: 수동 프로세스를 제거하면 시스템의 확장성 및 일관성과 조직의 효율성을 개선
      2. 삭제 가능한 리소스 사용: 서버 및 기타 구성요소를 임시 리소스로 간주
   3. 코드형 인프라: 인프라를 생성하는 데 적용되는 SW개발 기술, 사례 및 도구
   4. 인프라를 코드로 취급할 때의 이점
      1. 반복성 / 재사용성
      2. 유지관리성, 일관성 및 병렬화
2. 파트2: AWS의 코드형 인프라(AWS CloudFormation)
   1. CloudFormation: 코드형 인프라: 템플릿을 통해 AWS 리소스를 시작, 구성 및 연결할 수 있음
      1. 템플릿
         1. 리소스를 설명하는 JSON 형식 파일
         2. 소스코드로 취급, 저장소에 저장
         3. YAML 형식 템플릿 지원
      2. AWS CloudFormation 엔진
         1. AWS 서비스 구성 요소
         2. 템플릿을 리소스 스택으로 해석
      3. 스택: 단계
         1. 리소스 모음
         2. 콘솔에서 추적 및 확인 가능
         3. 교차 스택 참조
3. 파트3: 리소스는 어떻게 여러 템플릿으로 그룹화해야 합니까?
   1. 템플릿 구성
      1. 소유권과 App 수명 주기를 기준으로 템플릿에 리소스를 지정
      2. 리소스를 자체 템플릿으로 분리
      3. 의도적으로 해당 리소스를 유형을 중앙에서 제어하려는 것이 아닌한, 단일 템플릿을 App 전체에서 공유하지 말 것!
      4. 보안 리소스가 포함된 템플릿을 다른 템플릿과 별도의 저장소에 저장
4. 파트4: CloudFormation에서 직접 지원하지 않는 리소스 기능은 어떻게 됩니까?
   1. CloudFormation: 사용자 정의 리소스 기능을 활용하여 자체 로직을 스택 생성의 일부로 추가
   2. Elastic Beanstalk: 웹 App용 서비스. 로드 밸런싱 코드 배포등을 자동으로 처리