

# AWS 탐험기

몇 주 간의 걸친 AWS 삽질기

발표자 : 다람쥐

발표날짜 : 2020년 1월 9일 토요일



**최근 AWS 작업**

## 최근 AWS 작업

# 사이드 프로젝트

1. AWS EC2
2. AWS CodeDeploy
3. AWS RDS
4. AWS ElastiCache
5. AWS Route 53
6. AWS ACM
7. AWS Elastic LoadBalancer (EC2 내 존재)
8. AWS S3
9. AWS CloudFront

## 최근 AWS 작업 - 사이드 프로젝트

# AWS EC2 & CI/CD 구축

### 1. API 서버 전용

1. REST API 서버
2. Sequelize & AWS RDS ( 정규화 & 반정규화, 인덱스 설정 )
3. AWS ElastiCache 로 응답값 캐싱 ( Master, ReadOnly 구분하여 사용 )
4. Sequelize-CLI 로 Migration 적용

### 2. CI/CD 구성

3. PM2 다운타임 없도록 프로세스 이벤트 수정
4. NginX 적용

## 최근 AWS 작업 - 사이드 프로젝트

# AWS EC2 & CI/CD 구축

1. API 서버 전용
2. CI/CD 구성
  1. master 브랜치에 푸시될 시 빌드
  2. 압축하여 AWS S3 버킷에 전송
  3. AWS CodeDeploy 실행하여 AWS EC2 CodeAgent 로 전송
  4. AWS EC2 CodeAgent 에서 압축 해제 후 스크립트 실행
  5. NPM 빌드 & PM2 재실행 & 200 Heath Check
3. PM2 다운타임 없도록 프로세스 이벤트 수정
4. NginX 적용

## 최근 AWS 작업 - 사이드 프로젝트

# AWS EC2 & CI/CD 구축

1. API 서버 전용
2. CI/CD 구성
3. PM2 다운타임 없도록 프로세스 이벤트 수정
  1. 기존 프로세스 종료와 새 프로세스 서버 사전 작업 (DB 싱크) 까지 딜레이 시간 존재
  2. 새 프로세스 서버 사전 작업이 완료될 때 까지 대기
  3. 사전 작업 완료 될 시 프로세스 통신으로 기존 프로세스 종료
4. NginX 적용

## 최근 AWS 작업 - 사이드 프로젝트

# AWS EC2 & CI/CD 구축

1. API 서버 전용
2. CI/CD 구성
3. PM2 다운타임 없도록 프로세스 이벤트 수정
4. NginX 적용
  1. HTTP 80 포트로 Node 3000 포트로 포워딩하는 프록시 서버

## 최근 AWS 작업 - 사이드 프로젝트

# AWS Route 53 & 도메인 작업

1. 도메인 구매하여 호스팅 등록
2. 호스팅 하위 도메인 레코드 등록
  1. Web 서버 : dear-world.live
  2. Api 서버 : api.dear-world.live
3. API 서버 A 레코드로 호스팅 영역 등록
  1. EC2 퍼블릭 주소를 api.dear-world.live 도메인에 A 레코드로 등록



## 최근 AWS 작업 - 사이드 프로젝트

# AWS S3 & CloudFront 정적 웹 호스팅 배포

1. AWS S3 & Github Actions 로 정적 웹 호스팅 배포
  1. AWS S3 웹 호스팅 전용 퍼블릭으로 버킷 생성
  2. Github Actions 으로 웹 프론트 빌드 결과 압축하여 S3 버킷으로 전송
2. AWS CloudFront 로 도메인 작업
  1. AWS S3 와 연동하여 Distributions 생성
  2. AWS CloudFront 도메인 주소를 AWS Route 53 A 레코드로 등록

## 최근 AWS 작업 - 사이드 프로젝트

# API 서버, WEB 서버 HTTPS 작업

1. AWS ACM 으로 도메인 인증서 발급
  1. 기본 dear-world.live, 추가 api.dear-world.live 으로 인증서 발급
2. AWS CloudFront 에 인증서 적용
  1. Distribution Edit 화면에서 인증서 적용
3. AWS EC2 Elastic LoadBalancer 생성하여 인증서 적용
  1. ELB 생성하여 실행 중인 EC2 인스턴스와 연결
  2. HTTPS 인증서 등록
  3. 반드시 EC2 가용 영역과 맞는 지 확인할 것

# 최근 AWS 작업

## 사내

1. **AWS S3**
2. **AWS Lambda**
3. **AWS API Gateway**
4. **AWS CloudFront**
5. **AWS ACM**
6. **AWS Route 53**

## 최근 AWS 작업

# 사내

**목적 : 구글 스토리지 서비스에서 AWS 스토리지 서비스로 이전**

## 최근 AWS 작업 - 사내

# 바이너리 파일 업로드

1. AWS S3 버킷 생성
  1. us-east-1 리전으로 생성
  2. 퍼블릭으로 생성
  3. 어디서든 접근할 수 있도록 권한 설정
2. AWS Lambda 함수 생성
3. AWS API Gateway 연결
4. AWS Lambda 바이너리 파일 업로드 코드 업로드

## 최근 AWS 작업 - 사내

# 바이너리 파일 업로드

1. AWS S3 버킷 생성
2. AWS Lambda 함수
  1. 같은 리전 us-east-1 으로 함수 생성
  2. Node.js 12 선택
    1. 풍부한 npm 라이브러리 설치 가능
    2. 비동기 처리
3. AWS API Gateway 연결
4. AWS Lambda 바이너리 파일 업로드 코드 업로드

## 최근 AWS 작업 - 사내

# 바이너리 파일 업로드

1. AWS S3 버킷 생성
2. AWS Lambda 함수
3. AWS API Gateway 연결
  1. 리소스와 메서드 생성
  2. 멀티파트 헤더 허용
  3. 요청 응답 화면에서 멀티파트 매핑 템플릿, 패스 스루로 설정
  4. 무조건 JSON 데이터로 람다 함수로 넘어감
  5. 따라서 바이너리 데이터를 한 필드에 base64 인코딩하여 전송 (?!)
4. AWS Lambda 바이너리 파일 업로드 코드 업로드

## 최근 AWS 작업 - 사내

# 바이너리 파일 업로드

1. AWS S3 버킷 생성
2. AWS Lambda 함수
3. AWS API Gateway 연결
4. AWS Lambda 바이너리 파일 업로드 코드 업로드
  1. 필드로 넘어온... 바이너리 데이터를 직접 파싱해야 함...
  2. 'multipart' 파싱 모듈 사용
    1. 제대로 된 게 없었음
    2. 지금 쓴 모듈도 예외처리 코드 삽입하여 커스텀
  3. 기존 규칙에 따라 S3 버킷에 저장



최근 AWS 작업 - 사내

# AWS CloudFront & Lambda@Edge

<https://~~~/~~~.jpeg?width=100&height=100&format=png&quality=20>

**최근 AWS 작업 - 사내**

# **AWS CloudFront & Lambda@Edge**

- 1. AWS CloudFront와 AWS S3 연동**
  - 1. AWS ACM 연동하여 HTTPS 설정**
- 2. AWS Lambda 생성하여 Lambda@Edge 배포**
- 3. AWS CloudFront 캐싱 전략 수정**

## 최근 AWS 작업 - 사내

# AWS CloudFront & Lambda@Edge

1. AWS CloudFront와 AWS S3 연동
2. AWS Lambda 생성하여 Lambda@Edge 배포
  1. AWS Lambda 에 코드 업로드
  2. Lambda@Edge 배포로 AWS CloudFront Origin Response 로 등록
  3. 모든 복제된 리전에 배포하기 까지 수 분 걸림...
  4. S3 리소스 접근 -> 이미지 정보 쿼리스트링에 맞게 조작 -> 캐싱하고 응답
  5. X-Cache 응답 헤더에 Miss from cloudfront, Hit from cloudfront
3. AWS CloudFront 캐싱 전략 수정

최근 AWS 작업 - 사내

# AWS CloudFront & Lambda@Edge

1. AWS CloudFront와 AWS S3 연동
2. AWS Lambda 생성하여 Lambda@Edge 배포
3. AWS CloudFront 캐싱 전략 수정
  1. 캐싱할 쿼리스트링 목록 저장

## 최근 AWS 작업 - 사내

# AWS Route53 도메인 연결

1. API Gateway 도메인 주소 연결
  1. AWS API Gateway 콘솔에서 도메인 생성하여 배포한 API 와 매핑
  2. AWS Route53 호스팅 영역에서 하위 도메인으로 레코드 등록
2. CloudFront 도메인 주소 연결
  1. AWS Route53 호스팅 영역에서 하위 도메인으로 레코드 등록

**감사합니다.**