구글 폼 – 마켓터 : 편하다.

광고 같은 걸 활용 – 페이스 북, SNS, 구글 애드 기능 활용.

수집은 그쪽으로.

설문을 위해서 인프라를 구축? 비용 효율적이지 않음. 비현실적음.

쉽게 공짜로 쓸 수 있는 폼 사용

다른 컨셉을 틀어서 사용 가능

데이터 레이크 같은 걸 사용?

기술적으로는 크게 나쁘지 않음.

[민경]

크롤러는 계속 돌리는 거냐?

돌리는 이유. 크롤러를 돌려야 하는가.

데이터 카탈로그를 만들기 위해서 사용

크롤러 돌리면 카탈로그를만들 때 아테나에서 스캔하는

파티션 – 년 월일을 어떤 식으로 표현?

정식 형태로 하면 키 : 밸류

구조는 키가 잡혀 있으면 크롤러는 기본적으로 파티션으로 인식

파티션으로 데이터를 나눈다.

파티션으로 나눈다는 의미는 아테나는 해당하는 것만 스캔하게 되어 있다.

아테나가 쿼리학 위해서는 카탈로그를 사용해서 설정.

테이블 정보로 Data Catalog와 함께 파티션 값이 들어가 있다.

파티션은 수동으로 설정해야 한다. Msk repair을 사용해서 스캔한다. 근데, 이건 부하가 많이 걸린다.

그래서 수동으로 해야 하는데, 크롤러가 이를 도와준다.

새로운 날짜가 계속 추가되면 파티션이 없어서 미리 만들어 놓거나 데이터가 추가되면 만든다.

왜 필요하냐? 의도: 그 부분에 대해서 이해하고 있고 이걸 관리할 수 있는지.

파티션이 잡혀 있는 테이블은 파티션 키가 없으면 아테나에 쿼라가 안 된다.

데이터가 있어도 쿼리가 안 되면 파티션 키가 없는 것이다.

키가 잡혀 있는 건 키 기준으로

S3와 RDS

설문 조사의 경우 분석이 중심이 되는 데이터라고 생각하기 때문에 RDS를 사용하지 않았습니다. RDS는 전통적으로 사이트의

바람직하지는 않는다.

Csv 파일을 넣는 방식은

Order : 주문 히스토리 맞아요?

Order는 계속 쌓일 것

그리고 이걸 한 곳에 넣으면 어떻게 되어 있겠는가?

파티션이 어떻게 잡히겠는가?

계속 누적할 것인가?

S3는 객체 스토리지.

버킷의 경로는 키다. ‘/’ 되어 있으면 디렉토리처럼 나눠 놓은 건데, 사실은 경로를 전체 합한 것. 키는 유닉크하다.

파티션을 날짜별로 하려면 구분을 해놨어야 한다.

RAW 데이터이지만, 그 밑에 memebers/orders/products로 설정. 그리고 orders는 날짜 별로 저정해야 한다.

아이디어는 현실적 – 리뷰(내가 보기에는 괜찮아도 리뷰가 없으면 망설임)

* 구매 전환율 가능성이 높다.

[혜성]

어떤 오류?

DMS 권한에 S3 접근 권한이 있는가?

해당 되는 버킷의 policy?

DMS를 쓸 수 있다.

근데 과한감이 없지 않다. 실시간으로 하면 S3로 넣는 게 좋다.

개발자가 귀찮아 할지도

dms에서 s3로 저장될 때 어떻게 될까?

저장 구조가

데이터 관점

OLTP : 내 데이터만 보면 된다. 한정적인 데이터를 본다. 레코드 단위로 저장. 블록 단위로. 블록에는 레코드가 순서대로 저장. 인덱스.

OLAP : 전체 데이터를 보면 된다.

컬럼의 장점

필요한 컬럼만 잘라서 조회

남녀, 필요한 값은 두 개. 고객이 천만명. 거기에 2개.

남자, 여자 저장을 2개만 하면 된다. 어디에 있는 정보면 메타 정보로 가지고 있음. 논리적 압축.

성별 저장은 남자, 여자. 그리고 어디에 있는.

단, 레코드 단위로 저장하면 남자, 여자를 다 저장해야 한다. 저장하고 있는 사이즈가 너무 커진다.

그러므로 컬럼 단위는 다량의 데이터를 처리할 때 더 효율적.

빠르게 비용 효율적으로 있는 방법을 찾아야 한다.

데이터를 일부분 삭제하면 어떻게 될까?

해당 되는 레코드가 있는 오브젝트를 찾아서 새로운 레코드를 빼거나 추가하거나 한 다음에. 그 값에 해당하는 오브젝트를 새로 만든다. 올리면 기존에 있던 오브젝트는 삭제하고 완전 새롭게 한 것들을 다시 헤야 한다.

이런 개념들을 잘 만들어 놓은 게 ICE BUG, 코디니 델타 레이크 등을 설정

오브젝트들을 방금 말한 것들로 해서 변경된 데이터가 실제 현재 사용된 오브젝트라는 것이 맵핑시켜서 관리하고 예전 데이터는 이전 오브젝트로 인식하게 만든다.

클루 카탈로그를 만들 때 그냥 만듦. 하지만, 글루가 레이크 하우스 포맷을 지원한다. 나중에 DMS를 통해서 말어 넣을 때 CDC로 넣으면 포맷이 다르다.

기존 데이터를 알아서 삭제하지는 않는다. INSERT, DELETE, UPDATE하는 방식을 지원하지 않는다. 대신 FLAG가 붙는다. 누적된 값들에 대해서 해당하는 로직을 해야 한다. 그대로 크롤링하면 FLAG가 붙어 있는 테이블이 생성.

아테나가 UPDATE, DELETE가 가능. 하지만, DMS가 자동으로 해주는 건아니다. 디테일이 없다.

다양한 조직들과 일을 해야 하는데, 어쩔 수 없이 옵션으로 사용할 수는 있다.

[건호]

엑세스 로그가 지속적으로 쌓인다?

엑세스 로그를 가지고 오면 JSON 파일이 들어가 있는데, JSON 파일이 계속해서.

파일명은? 누적이라는 값이라는 게 동일한

버퍼 사이즈가 잘못된 거 같고. 레코드 하나 당 파일 하나씩 쌓인다?

근데 Fire hose를 바로 실행? 키네시스 안 쓰나?

샤드 부분을 빼면 데이터 스트림을 쓰는 이유를 모른다.

느리다? Fire host로 전송되는데, 느리다. 왜냐하면 버퍼 타임이 설정되어 있다.

버퍼 타임 및 사이즈를 충족해야 하는지 떨구는데 이게 충족되지 않으니까 떨구는 것.

테스트 하려면 1분 이내로 설정

스트림을 쓰는 이유.

카프카의 관점은 db라고 인식을 하기도 한다.

DB는 데이터를 저장하는 시스템. 데이터를 저장하는 것이다. 스트림으로 옮기는 것도 있지만, 카프카는 동시에 데이터를 저장한다.

왜 저장?

일시적으로 데이터를 모아서 한 번에 처리. 부하도를 조정하기 위함.

예를 들어서 팀장이 일을 받았음. 근데 막 패시하면 밑에서 부하가 높아진다.

개별적으로 일을 줬음. 근데 전달이 안 됐음. 누락이 된다. 안정적인 스트림이 구축되어 있어서.

카프카는 땡겨 가는 방식. 특정 시점에서부터 데이터를 가지고 올 수도 있다.

스트림이 있으면 이러한 역할을 한다.

엑세스로 저장될 때는 JSON으로

키 – 벨류는 어떻게?

ACCESS 로그에서는 가지고 올 수 있는 게 많이 없다. 그래서 이걸 가공해서 더 많은 데이터, 레퍼러 데이터, 클릭 IP 등을 가지고 온다.

누가 뭘 클릭했는지 모른다. 이걸로는 누가 접근했는지 알 수가 없다. IP로 맵핑한느 건 쉽지 않다. 실제 실무에서는 유저 정보의 세션 ID 값과 IP 정보를 맵핑 테이블을 만들어 놓는다. 이걸 저장해놨다가 반본적으로 들어오는 IP는 추적해서 식별

단순한 IP 정보로는 아무것도 할 수가 없음

이 사람의 성별이 뭔지, 나이, 지역과 같은 다양한 정보.

이탈 방지. 어디로 들어와서 어디로 나가는지. 타임 스탬프 별로

요즘에 페이지 네이션이라는 옵션을 조심해야 한다. 이걸 자동으로 켜져 있음. 엄청 비싸다. 무조건 끄자.

주제를 잘 버무리면 좋을 거 같다.

RNR을 사용해서 뭘 하고 싶다.

특정 서스를 기준으로.

파이프 라인 영역, 수집 영역에 대해서만 어떻게 하면 좋을지.

민경, 혜성이 비슷하지만 다르다.

둘이 합치자.

지원 – 설문(현실적으로는 어려움)

- 어떻게 보면 수집을 해서 시트에 있는 데이터를 로딩

- 소스로 많이 쓰는 것중 하나가 엑셀 로딩하는 작업

- 엑셀을 로딩해서 파싱해서, 쿼리

- 좋은 거는 그 데이터를 바로 그냥 S3에 밀어 넣으면 좋다.

- 이런 부분이 추가되면 나쁘지 않다.

RDS에 넣는 것보다 스트림 형식으로 하는 게 모범적인 아키텍처

DB에 있는 데이터를 가지고 와야 하기도 한다.

고객 정보라던지.

엑세스 로그는 별로 좋은 로그가 아니다. 필요할 수는 있지만. 엑세스 로그와 비슷하게 로그를 쌓겠금 스크립트를 만든다.

구글 애널리틱스 – 헤더를 설정한다.

엑세스 로그가 아니라 실제 넣는 것들을 던질 수 있다.

상세한 정보들이 쌓인다.

쉽게 할 수 있는 방법.

자바 스크립트로 하나 만들어서 거기에 현재 세션 아이디나 상품 ID, 카테고리 ID를 넣는다.

작은 홈페이지. 소스가 간단하다.

클릭할 때 호출되는 자바 스크립트 함수가 있다. 필요한 파라미터를 넣고 던진다.

필요한 파라미터 정보를 다 받아서 수집성으로 넣는다.

이런 식으로 호출되는 걸 하나 만든다. 이걸 액세스 로그 형태로 유사하게 조금 더 디테일하게. 일반 액세스 로그는 아무 의미가 없다.

데이터

수집(설문, 엑셀로 가지고 온다. 수기든 자동이든 일단 s3로 가겠끔)

DB – S3로 가게(DMS는 비효율적. 과한 감이 없지 않음. 스냅샷. 고객 정보, 일반적인 메타 정보, snap shot.)

실제 트랜젝션에 대한 로그(엑세스 로그만을 가지고는 많은 걸 볼 수 없다. 실제 로그를 수집하는 고민(핵심)을 해야 한다. 스크립트로 만들어서 한다. 그리고 수집 서버가 있어야 하고 데이터가 s3로 가야 한다. Fire hose가 될 수도 있고 Stream이 될 수도 있고.)

Glue Crawler를 통해 데이터 카탈로그를 만든다. 메타 데이터, 파티션을 고려, 별도로 이를 위한 Lambda를 사용한 파티션. 이걸 기반으로 셋팅.

데이터 마트 구성

시각화(Quick Sight)

대시 보드를 어느 수준까지?

판매율이지만 리뷰인지 등등한 속성 별로 추이가 다르다. 디테일이 달라질 수 있다. 포맷은 하나지만, 리뷰, 구매에 대한 것

데이터 레이크. 하나의 버킷에 넣어도 되고 버킷을 쪼개도 된다. 고객 정보 – 민감. 버킷을 따로 만들어서 설정해야 한다. 이런 걸 부각시킨다.

민감 데이터이기 때문에 구분했다.

S3 권한 있는 사람. 암호화.

엑셀에서 가려면 파싱하려면 코딩한다. 람다 활용해서.

일단은 시각화는 필수이다. 얼마나 잘했는가? 이걸 위한 작업은 부수적이다.

현실적으로 할 수 있는 부분 없는 부분. 각 요소들을 구체화.

일종의 WBS를 만들자.

테스트 별로 나누고 이를 구체화한다.

수집은 다른 의미.

현실적으로 솔루션 쪽에서 하게 되어 있다. 솔루션 회사에서는 수집만 한다. 때문에 우리는 일단 처음에는 필요한 데이터를 가지고 와서 데이터 분석을 하고 시각화를 한다.

RNR을 최대한 구성 – 어느 정도 진행

대시 보드를 구성하고 테스트를 먼저 한다. 이게 낫다. 왜냐하면 실제 프로덕이 아니라서 실제 데이터가 뭔지 모른다. 원래는 고객이 가지고 있는 데이터를 먼저 조사한다. 이를 확인하고 뽑아낼 수 있는지. 이 부분을 먼저 고민한다.

가장 효율적인 방법은 시각화를 먼저 간단하게 만든 다음 가상의 시나리오를 만드는 게 좋다.

자체적으로 하지는 않는다. 우리가 그들이 사용하는 EC2에 접근하지는 않는다. DB 접속해서. 접속할 권한이 있으면 Sage Maker로 방법은 하기 나름. Glue로 바로 사용.

결국에는 비용적인 부분, 성능적인 부분, 이걸 사용하는 사람들의 스킬 셋. 이 사람이 쿼리에 익숙? 파이선 언어에 익숙? 이 유저들이 운영을 해야 하니까. 그들이 쓰는 스킬 셋에 맞춘 거가 제일 좋다.

람다 – 파이썬. 필요하면 자바 써도 되는데? 설정하기 나름이다.

아마도 소스 코드

고객이 해야 한다. 관리를 못하는 조직이 있다. 운영에 대한 부분. 메가존이나 베스핀에서 말하는 운영은 서버 이슈, 모니터링 및 대응. EC2, CPU, 메모리에 대한 이슈.

데이터는 운영 관점이 리포트. 충분히 할 수 있는데, 일부 대행을 해준다. 대시 보드를 추가 수정한다거나 꾸준히 일반 실무자들이 필요로 하는 걸 한다.

인사이트가 나온다는 의미. 궁금한 게 생기고 얻을 게 있다는 의미. 새로운 뷰를 요청하는 게 정상. 계속 발견하고 싶어해야. 자기들이 꾸준히 해야 하는데, 그런 기술 내제화가 안 되어 있으면 별도 요청할 수 있다.

시나리오 자유롭다?

Ctc – 서버 운영 관점, 보안 관점.

데이터 아키텍처를 요청하는 고객은 인사이트를 제외하고 뭘 더 중점으로? 사실은 기본만 리소스는 프라이빗에. 정말 기본적인 것만.

앤드 포인트를 최소화 – 보안 최소화

성능은 당연 비용은 줄이면 좋다.

성능이라는 의미는 배치 쪽 성능. 대시 보드 성능일 수도 있다(내일 아침까지 기다려야해?) 배치성 최근 데이터를 빠르게 볼 수 있느냐?

퀵 사이트는 실시간은 아니다. 퀵 사이트를 실시간처럼 스파이시를 쓰느냐 마느냐. 얼마나 자주 갱신행 하느냐?

실시간이라는 게 30분일 수도 있다.

고객사 성향도 설정하기 나름이다. 발표할 때 얼마나 잘 합당한 솔루션을 제시했는가를 시나리오에서 꾸며줘야 한다.

교육에 맞는 많은 걸 배운 것처럼 보이게금

이런 것들이 다 되면 인터뷰를 볼 때 많은 도움이 된다.

조직이 인프라쪽 팀, 보안 팀,

1시부터

**건호**

* 코드 커밋: git 보다 많은 걸 하지 않는다.
* git hub를 사용하던 걸 굳이 코드 커밋으로 이동할 필요가 없기는 한다.
* 웹 서버가 퍼블릭에 있어도 상관 없음?
  + 퍼블릭에 있다는 건 외부에 노출된다는 의미.
  + 웹 서버의 소스 코드는 대단치 않다. 털러도 됨
  + 근데 외부 공격자가 퍼블릭에 있는 ec2에 다이렉트로 붙어버린다. 엄청난 공격!
  + 프라이빗에 있어도 ANY라는 의미는 VPC가 격리되어 있음
* 보안 그룹이라든가 전부다 명시를 해야 한다.
  + 처음부터 잘 만들어두면 관리하기는 편하다.

**지원**

* 리전 간 백업
* 스냅샷 자체가 RDS에서 S3를 떨구는 것
* AWS 백업 - 수동으로 했던 것을 서비스로 빼 놓은 것. 이걸 활용하자. 그냥 S3로 내리는 건 좀 옛날 방식.
* 백업의 기본은 native 서비스에서 최대한 활용. 여기서 제공하지 못한 걸 꼭 해야 할 때.
* 최대한 백업을 논의할 때는 aws 서비스에서 최대한 활용할 수 있는 걸 확인. 추가적인 요건이 있는데 이 부분을 커스텀으로.

**ec2 타입**

* t 타입 : 버스팅
  + t 타입의 취지. 일반 프로덕에서 잘 안 쓰인다. 크래딧이 언리미티드 옵션이 생겨가지고. 크래딧이 소모가 되면. 서버가 죽는다.
  + 크래딧이 소모되는 순간 성능 제한을 걸어버린다. 트래픽은 계속 받고 있는 상황.
  + 사용자 입장에서는 서버의 성능이 떨어진다.
  + 원래는 취지는 저렴하게 사용한다. 기준이 있음.
  + 타입 별로 특징이 다 있다.
  + i : 데이터 트래픽
* 비슷한 가격대로 사양을 조정
* 그냥 싸게 만든 것. 100% 안 쓸거라는 게 전제가 되어 있다. 기준을 잡은 것보다 덜 쓴다. 트래픽이 올라가. 기준보다 더 써. 크래딧이 소모가 되면서 계속/
* 크래딧이 쌓이는 양. 성능이 높아질 수록 많이 크래딧이 쌓인다.
* 싸서 쓴다는 게 좋을지도. 트래픽이 많이 나오면 쓸 수 있다는 것.

**현업**

* 트래픽이 몰린다. 일종의 입출력 쪽, 네트워크냐 디스크냐.
* 대부분은 m 타입을 사용한다. cpu를 많이 사용 - app 서버에서 비즈니스 로직 처리, 메모리 캐싱(데이터) - DB
* 주로 m을 사용한다.

**코드 파이프라인**

* 굳이 다르게?
* WEB 서버 배포하는 데 ECS를 사용?
* S3
  + CDN

**Cloud Front**

* 빠르게 한다.

**디스크 캐시**

* 왜 m2가 빠른가?
  + 보드 하나에 모든 걸 넣는다. 하나에 잘 해놓는다.
  + 결론은 디스크는 느릴 수밖에 없다. 메모리는 빠르다.
  + 디스크보다 메모리가 비싸고 캐시가 더 비싸다.
  + 가장 좋은 건 캐시에 넣는 게 좋다. 근데 문제는 비싸다.

**s3 쓰는 이유**

* 싸다.
* 접근을 자주 하면 느리다. 그래서 비용이 높아진다.
* 한 번 캐시된 건 빨리 접근하게 한다.

**테라폼**

* 배포용으로 주로 쓰신다.
* 정리를 잘하는 게 중요하다.\

**버킷 주체**

* 관리 주체에 따라서 달라진다.
* 버킷을 더 만든다고 해서 달라지지는 않는다.

**cpu 알람 기준**

* 서비스에 따라서 다르다.
* 서버를 비용 최적화한다고 하면 대부분 50% 언더로 쓰는 게 안전
* 빡빡하게 쓰면 상시 60% ~ 70%
* 일반적인 트래픽이 있으면 일반적으로 예측을 한다. 패턴이 있고 어느 정도로 예측을 하고 벗어나면 설정
* 정하기 나름이다.

**백업**

* aws 백업
* s3에서 글래시어
  + IA, 라이프 사이클을 정리. 기준이 뭔지? 실제 데이터는 1년 ~ 2년 사용. 그 이전은 법률 상 저장은 해야 한다.
  + 빈도가 좀 있다고 하면 글래시어에 넣으면 안 된다.
  + 정책이 정해져야 구현할 수 있다.
  + 가이드 제공 - 고객이 직접 수명 주기를 설정할 수 있게

**혜성**

**ECS**

* 이미지가 왜 없나?
* 용량 공급자
* 랜덤 포트 많이 쓰나?
  + 컨테이너 인스턴스에서 포트를 띄울 때

**질문**

* 동적 포트를 맵핑했을 때 포트가 임의로 변경된다.
  + 외부에는 3만 번대로 들어오면 내부에서 80번으로

**케이스**

* AWS에서 이미 만들어진 롤 중에 생성이 안 되는 롤이 있음

**임계치**

* 컨테이너를 기준으로 스케일링
* 전체 리소스를 쪼개서 사용

**X-Ray**

* 이런 걸로 모니터링.
* 이게 맞으니까 이걸 사용하라는 의미.
* 시각화가 안 되면 모니터링이 안 된다. 내가 이 상황이 어떤지를 인지하겠금. 경보 알람만 받게 하면 모니터링은 안 된다.

**Cloud Front**

* 기본적으로 shield standard가 기본적으로 제공
  + DDOS로 종료되면 어떤 일이 일어날까? 서비스 장애
  + 무슨 일이 일어나나? AWS 서비스 중단. 여기에 올라가 있는 모든 서비스가 장애가 난다. 그래서 죽으면 절대 안 된다.
* Route 53 사용 이유
  + 왜 도메인을 쓸까? ip가 아닌 일반적인 문자로 접근하기 위해서
* Cloud Front도 마찬가지
  + 어떤 앤드 포인트로 접근해야 하는데, 기본적으로 여러 개 산제되어 있다.
* 장애가 나면 안 된다.
* ddos는 계속해서 진행. aws에서 어딘가에 처리가 되고 있다.
* cloudfront에 shield를 붙이는 게 좋다.
* waf는 alb 단에서 해도 괜찮다.

**Cloud Trail**

* 사용자나 서비스 어떤 서비스에 접근했는지를 확인하는 감사 시스템
* AWS 서비스가 수행한 작업
* 콘솔에서 수행하는 작업들은 다른 명령어로 가능하다.
* AWS CLI - 명령어. 무슨 명령어? 서비스에 대한 동작
* 사용자가 이상 행동을 했을 때
  + 콘솔에서 실행하는 건 모두 CLI로 작업할 수 있다.
* S3에는 명령어로 가능하다.
  + 호출된 건건이 과금. 원래는 다 API 호출으로. 기록이 된다.
* API 호출은 모두 기록이 된다. 콘솔로 할 수 있는 건 다 CLI가 된다. CLI는 모두 API 호출.

**정리**

* 필수 과제: 잘 어필을 해야 한다.
* 기본 것들이 잘 되면 좋은 점수를 받지 않을까? 하기 나름이다. 보너스적인 측면

**추가 질문**

**Glue Visual ETL 작업**

* 시각적으로 알 수 있어서 좋음
* 근데 자동화 하려면 노트북 등을 통해서 하는 게 좋지 않을까?
* 이 부분은 공통 영역이라 합의가 필요하다.
* 하면 좋다. 다만, 할 수 있는지는 다음 영역
* 데이터 파이프라인을 자동화했다는 걸 어필
* 핵심은 하되 나머지는 닿는 데까지 하는 것. 이걸 잘 선택하는 게 좋다.
* 워커 플로우 안 쓰는 곳은 없음 → 뭘 쓰는 써야 한다.

**Data Lake**

* 이걸 구성하는 게 핵심이다.
* DATA Lake → Glue Crawler → Data Catalog → Quick Sight
* 그 다음에 어디서 데이터를 받아올 건지를 골러야 한다.
* 자동화 선택
  + work flow
  + mwaa
  + Stp func
  + Glue work
  + DBT
* 제한된 시간 속에서 최대한 잘 포장

**CSV vs JSON**

* JSON은 비효율적

해야 할 것과 버려야 할 것을 잘 조율