
aws
SUMMIT
KOREA | MAY 10-11, 2022

T12S1

2200만 사용자를 위한 채팅 시스템 아키텍처

서호석
솔루션즈 아키텍트
AWS

변규현
SW 엔지니어
당근마켓

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Agenda

아키텍처 현대화

당근마켓 채팅 시스템의 현대화

데이터분석과 모니터링

맺음말

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

아키텍처 현대화

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

아키텍처 현대화

500,000,000

향후 3~4년 동안
출시될 새로운 앱

지난 40년간
개발된 총 앱 수

향후 몇 년 동안 기업 및 기관들은 이전 40년 동안 개발된 수를 합친 것보다 5억 개 이상의 새로운 앱을 구축할 것입니다.

*Source: IDC PlanScape. Oct 2020

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

아키텍처 현대화

수백만명의 사용자로 확장

글로벌 가용성

밀리초 단위로 응답

페타바이트의 데이터 처리

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

아키텍처 현대화

출시 시간 단축

혁신 향상

향상된 안정성

비용 절감

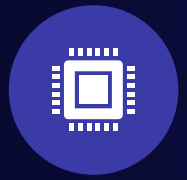
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

아키텍처 현대화

느슨하게 결합된 서비스로 설계

경량화된 컨테이너 혹은 서버리스 함수로 패키징

상태 비저장(Stateless) 서비스와 상태 저장(Stateful) 서비스를 분리하여 설계



탄력적이며 회복력 있는 셀프-서비스 인프라스트럭처에 배포

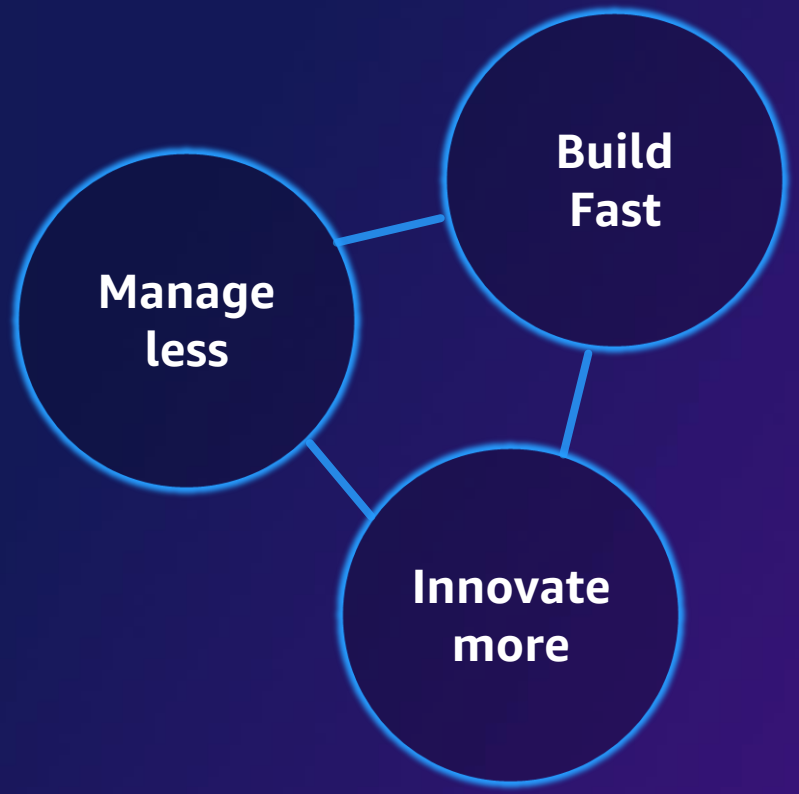
서버 및 운영체제 종속성에서 격리

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

아키텍처 현대화

현대화를 위한 세 가지 경로

1. 관리형 컨테이너 서비스로 플랫폼 변경
2. 서버리스 아키텍처에서 새롭고 안전한 앱 빌드
3. 최신 Dev+Ops / 클라우드 네이티브 모델로 전환

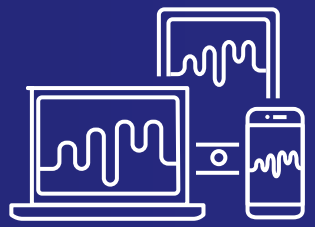


aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

혁신적 기술 관리와 새로운 기능 구축의 균형 유지

혁신 속도의 가속화

데이터를 활용 극대화



새로운 온라인 고객 경험 구축

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

애플리케이션 빌드 및 운영 방식 현대화

- 모듈화된 아키텍처 패턴
- 서버리스 운영 모델
- 애자일 개발 프로세스

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS는 현대화를 위한 모든 솔루션을 제공합니다

이미 실행 중이거나 애플리케이션을 컨테이너로 이동하는 것을 고려하고 있습니까?

워크로드를 **관리형 AWS 컨테이너 서비스로** 이동하여 운영을 단순화하고 관리 오버헤드를 줄입니다.

새로운 애플리케이션이나 기능을 구축하고 계십니까?

AWS Lambda와 같은 **서버리스 기술** 사용

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

당근마켓 채팅 시스템의 현대화

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

당근마켓 소개

월평균 1600만 명의 사용자

하루 평균 사용 시간 20분

1억 2천만 번의 연결

중고 직거래로 시작한 당근마켓은
국내 최대의 지역 생활 커뮤니티
서비스로 나아가고 있어요.

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

당근마켓의 성장

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

당근하는 새로운 방법, 당근페이

22.02.14 전국 서비스 오픈

- 중고거래 송금
- 계좌번호 없이, 채팅창에서 바로 송금
- 생활편의 서비스 결제
- 선물하기, 로컬 커머스, 제휴 서비스도 간편 결제

“동네 생활을 더 편하게, 이웃을 더 가깝게 하는 금융”

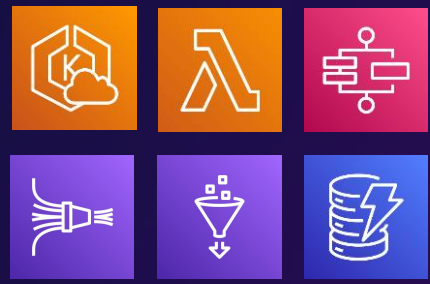
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

당근마켓 성장과 함께 서비스 복잡도의 증가

Ruby on Rails/PostgreSQL 로 구성된 모노리틱 시스템

Amazon ECS

서비스 성장과 함께 개발되는 다양한 서비스, 그리고 채택된 기술들



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

그리고 늘어가는 장애

_emergency

Marco 17:38
채팅암된다는데여

Seapy (씨피) 20:17
쿠베쪽이 이상이 있는거 같은데요.

Miti 17:39
Prod 클러스터가 좀 이상한 것 같네여

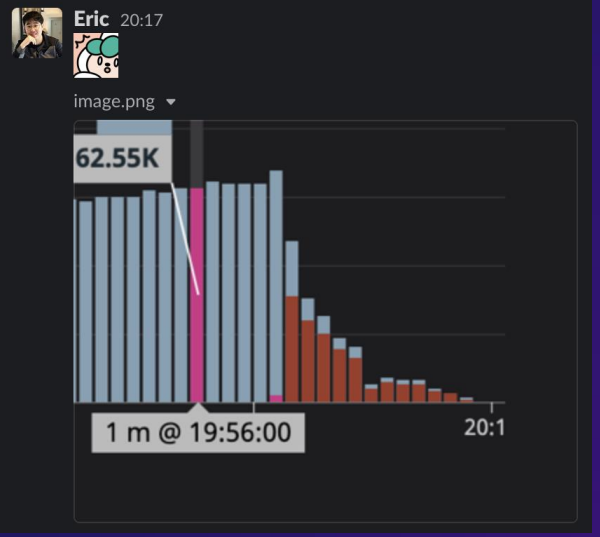
Ethan (에단) 20:16
8시경 부터 디비 커넥션 늘고 있어요

Paul 16:51
@channel ㅈㅓㄴ체장아

Stark (스타크) 19:56
어디 문제인지 apm 보신 분 계시나요

디비 문제인지 병목 위치부터 찾으면 좋을 것 같아요

Ethan (에단) 20:13
@here 장애요



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

변화하는 서비스의 형태

AS-IS

- 앱 사용자를 위한 서비스
- 유저와 직결되는 기능에 초점
- 한 서버에서 모든 기능을 구현
- 신규 기능은 모놀리틱 서비스에서 함수 형태로 구현
- 규모가 커지며 길어지는 배포 주기

TO-BE

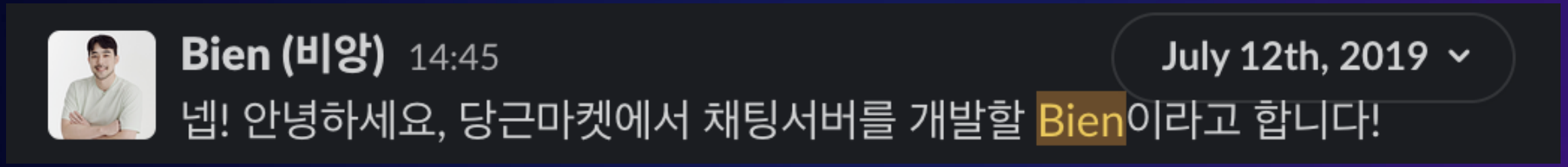
- 앱 사용자를 위한 서비스
- 사내 개발자를 위한 서비스
- 개발자를 위한 다양한 기능을 제공해야함
- API를 호출하는 형태로 지원
- 경량화 된 서비스 구성

→ **마이크로서비스의 도입**

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

당근마켓의 첫번째 마이크로서비스 시작

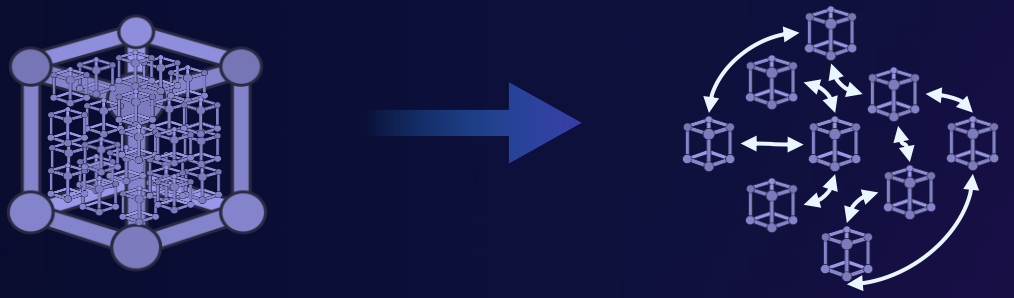


aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

당근마켓의 첫번째 마이크로서비스 시작

- Main DB에서 60%가 채팅 데이터
- Index가 대부분의 용량을 차지함 (전체의 약 50%)

채팅 데이터 분리로 성능 향상 및 서비스 안정성 강화를 목표로 함
당근에서 첫번째로 시작한 Micro service 분리 프로젝트



aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

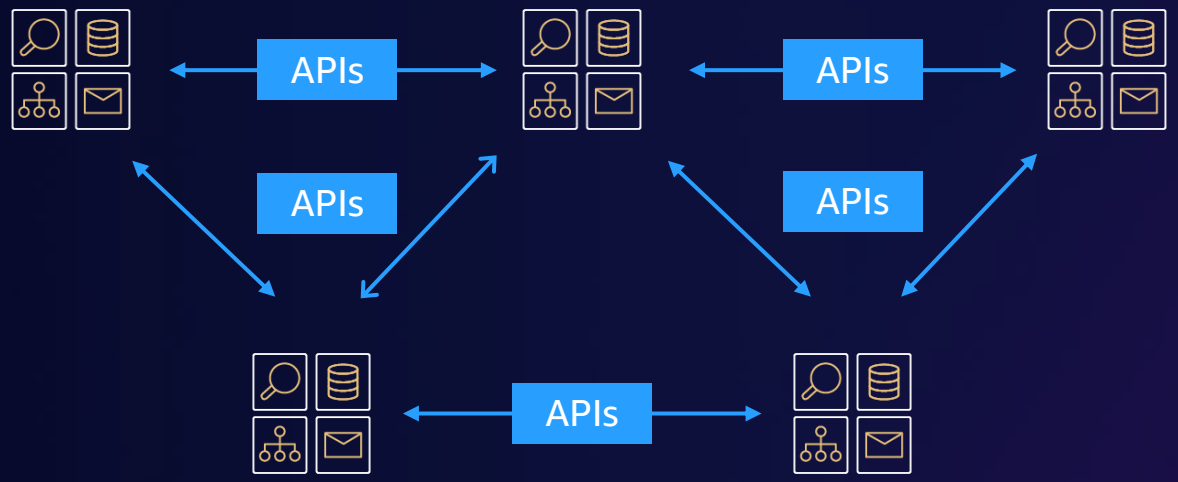
당근마켓 채팅 시스템의 현대화 Phase 1

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

채팅 마이크로서비스 개발하기

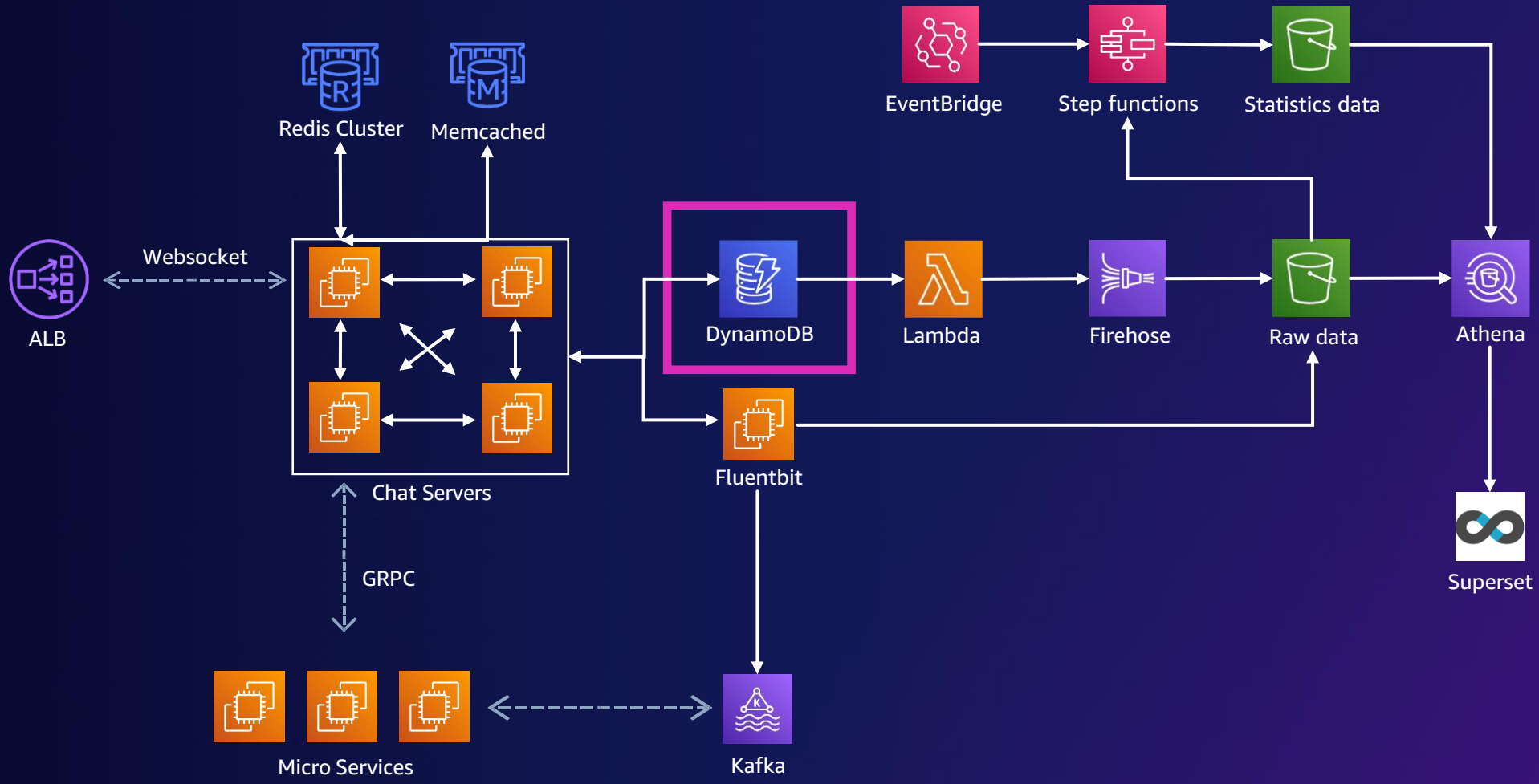
채팅 데이터 분리(데이터 모델링, 마이그레이션...)

기존 API 재구현



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

채팅 데이터 분리하기



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

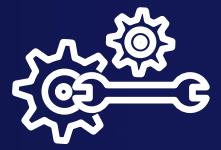
채팅 데이터 분리하기

Database Research 진행

- Managed Service일 것
- 최대한 운영에 대한 시간을 아껴야 함
- 데이터 용량 확장에 용이할 것

향상된 보안

강력한 컴플라이언스



관리 편의성

운영 비용 절감

지속적인 혁신

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

채팅 데이터 분리하기

데이터를 분리하기 위해선 옮겨갈 데이터베이스를 선택해야 함
고가용성, 안정성, Auto scaling, 성능

빠르게 성장하기 때문에 on-demand 요금
•이 모든 것을 지원하는 것은 DynamoDB

완전 관리형

모든 상황에서
일관된 성능

고가용성 및 내구성

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

채팅 데이터 분리하기

DynamoDB를 사용하면서 어려운 점

- 일반적인 SQL을 사용하지 못함
- 데이터를 가져오기 위해선 Query와 Scan으로 가져와야 함
- Query는 인덱스를 건 Item만 가져올 수 있음
- Scan으로 filter를 걸 수 있음.
- 다만 실제로 데이터를 전부 읽으면서 필터링해서 가져오기 때문에 비용문제가 생길 수 있음.
  실서비스에서는 사용하면 안됨

인덱스를 잘 두어야하기 때문에 설계에 대해서 고민이 많이 필요함
Full scan으로 전체 데이터에서 필터링하고 싶은 경우 DynamoDB 기능으로는 거의 불가능함

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

채팅 데이터 분리하기

설계시 고려한 점

- 분석이 용이할 것
- 데이터 파이프 라인을 구축해야 함

INDEX 설계

- 채팅방 하나가 Partition 하나로 설계. Sort key로 메시지를 구분한다.
- 한 채팅방에 들어올 수 있는 인원이 제약되기 때문에 Hot partition issue에서 자유로움

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

기존 API의 재구현

- Golang으로 모든 API를 재작성 및 연동
- 작은 Util 함수들도 전부 사용 형태에 따른 Test case 작성
- 서비스가 커짐에 따라 API 수준의 Test도 전부 Code로 작성(REST/gRPC/Websocket)
- 마이크로 서비스와 연동하는 클라이언트도 전부 테스트 케이스 작성
- 현재까지 약 1250여개의 TEST Case 작성

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

당근마켓 채팅 시스템의 현대화 Phase 2

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

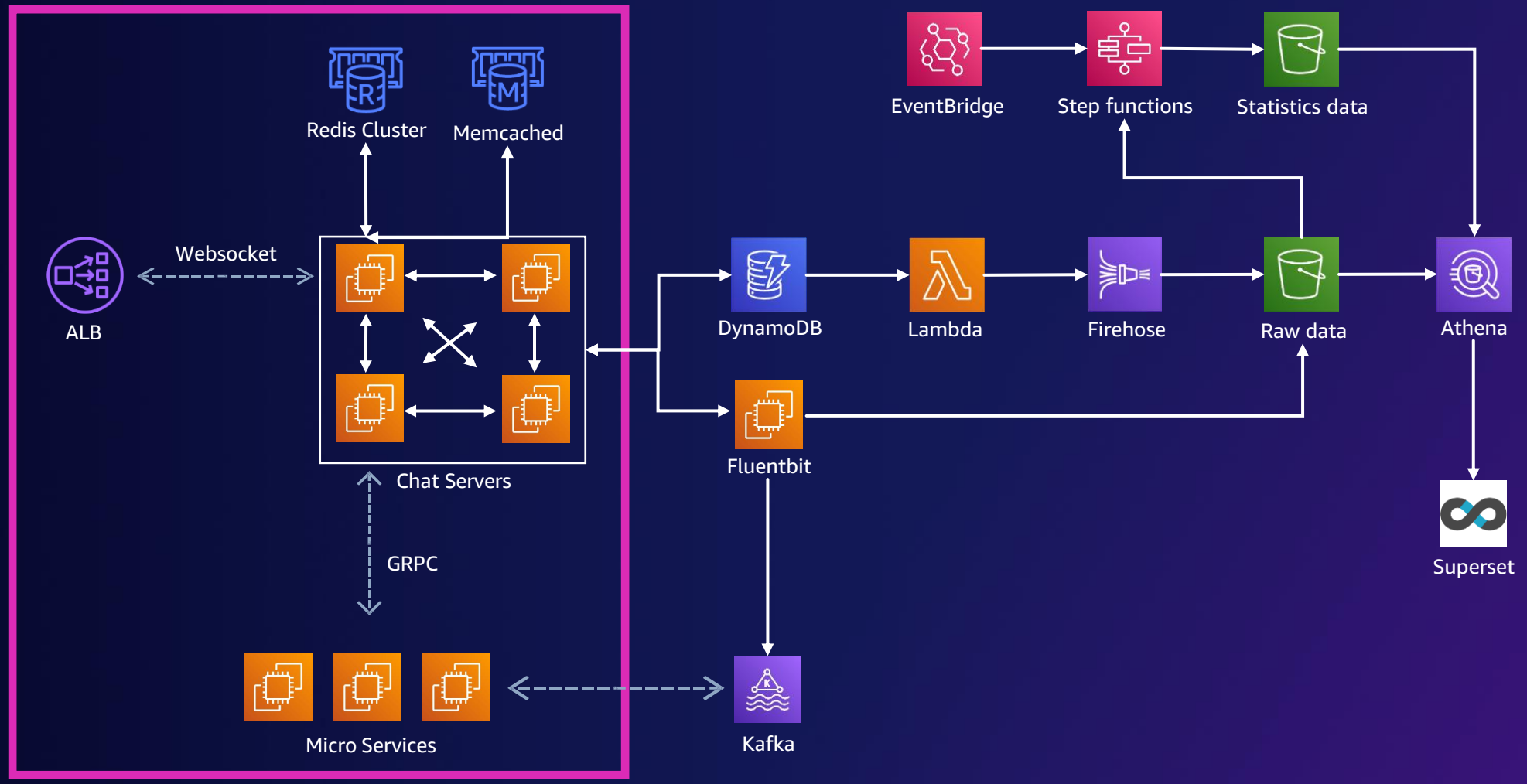
Websocket을 통한 실시간 메시징 시스템

프로토콜 변경
REST API -> Websocket

{ REST:API } vs 

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Websocket을 통한 실시간 메시징 시스템



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Websocket을 통한 실시간 메시징 시스템

ElastiCache - Redis

- Websocket을 사용하기 위한 Session Store
- Scale out을 위해 Cluster mode로 사용
- Stateless Server 로 사용 가능함

Amazon ElastiCache

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

당근마켓 채팅 시스템의 현대화 Phase 3

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

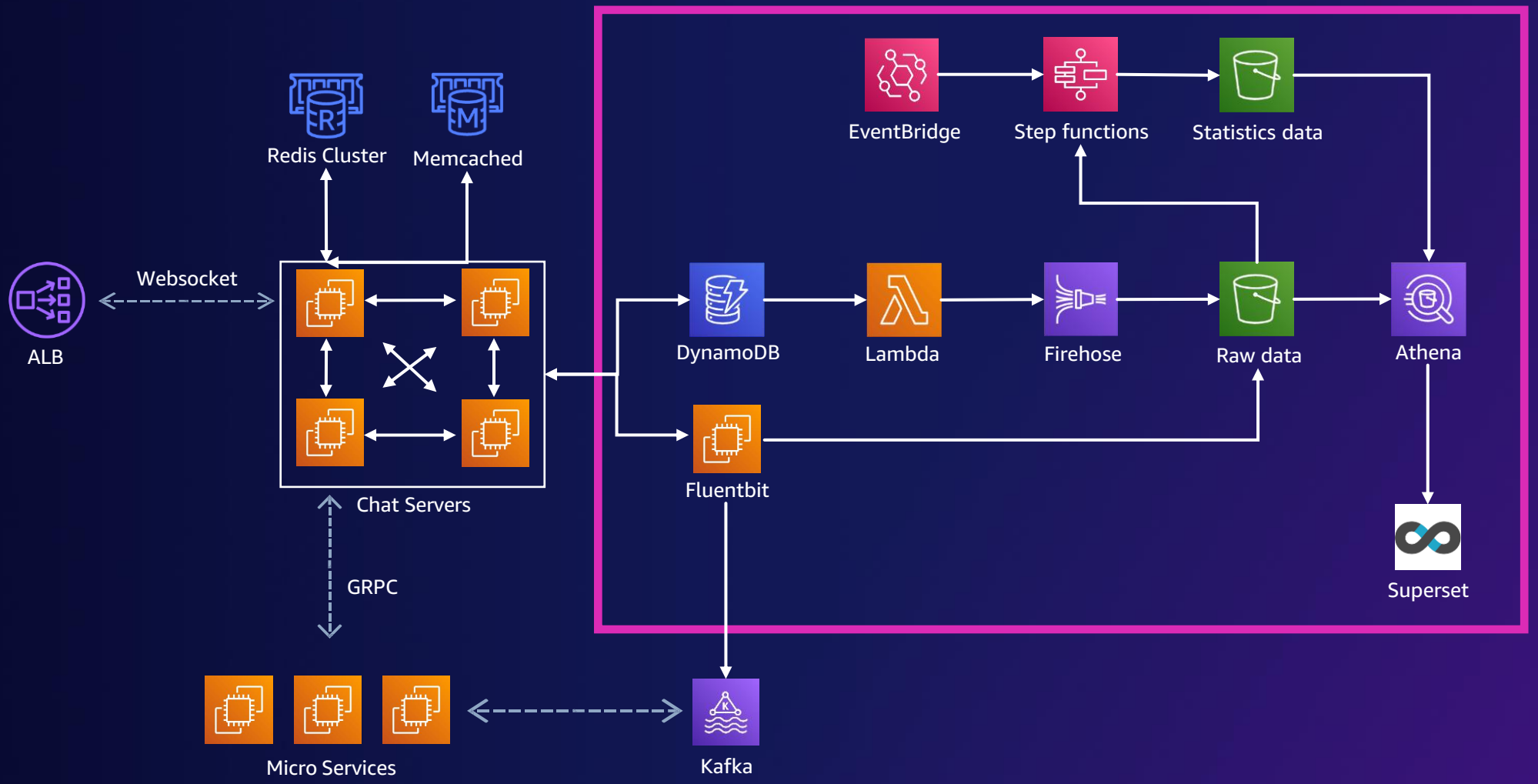
데이터 분석 PIPELINE - 구성요소

구성요소

- DynamoDB Stream
- Lambda
- Firehose
- S3
- Athena

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

데이터 분석 PIPELINE - 아키텍처



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Amazon DynamoDB Stream

- DynamoDB 테이블에서 시간 순서에 따라 Change Data Capture함
- 이 정보를 최대 24시간 동안 로그에 저장
- 로그와 데이터 항목은 변경 전후 거의 실시간으로 나타나므로 애플리케이션에서 이러한 로그와 데이터에 액세스할 수 있음

DynamoDB Stream

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS Lambda

- AWS Lambda는 새로운 DynamoDB Stream Record가 감지될 때마다 Stream을 폴링함
- Lambda 함수를 동기식으로 호출함
- 모양이 DynamoItem 형식이라 사용하기 어려움
- S3에 넣어서 사용하고 싶은데, Lambda가 진행하기엔 파일 크기가 너무 작음(1KB 정도)
- Firehose를 통해 S3에 넣는 방법을 선택(Serverless Service)

AWS Lambda

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS Lambda does

Input

- DyanmoDB Stream에서 나온 값은 Records로 감싼 형태
- SequenceNumber/SizeBytes 등 분석에 필요하지 않은 데이터가 존재
- Key: Value 형태가 아닌 Key: {Type: Value} 형태


```
{
  "Records": [
    {
      "awsRegion": "ap-northeast-2",
      "dynamodb": {
        "ApproximateCreationDateTime": 1578325846,
        "Keys": {
          "pk": {
            "S": "my_partition_key"
          },
          "sk": {
            "S": "my_sort_key"
          }
        },
        "NewImage": {
          "attr1": {
            "S": "new_attr1"
          },
          ...
        },
        "OldImage": {
          ...
        },
        "SequenceNumber": "232419000000000016191110770",
        "SizeBytes": 38,
        "StreamViewType": "NEW_AND_OLD_IMAGES"
      },
      "eventSourceARN": "..."
    }
  ]
}
```


Output

- NewImage에 들어있는 Key/Value만 사용
- 불필요한 값은 모두 제거
- 데이터 저장 크기 감소
- 추후에 데이터 조회에 용이한 형태로 저장함
- Input으로 들어온 Structured data보다 조회 성능이 뛰어남


```
{
  "pk": "my_partition_key",
  "sk": "my_sort_key",
  "attr1": "new_attr1"
}
```


aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Amazon Kinesis Data Firehose

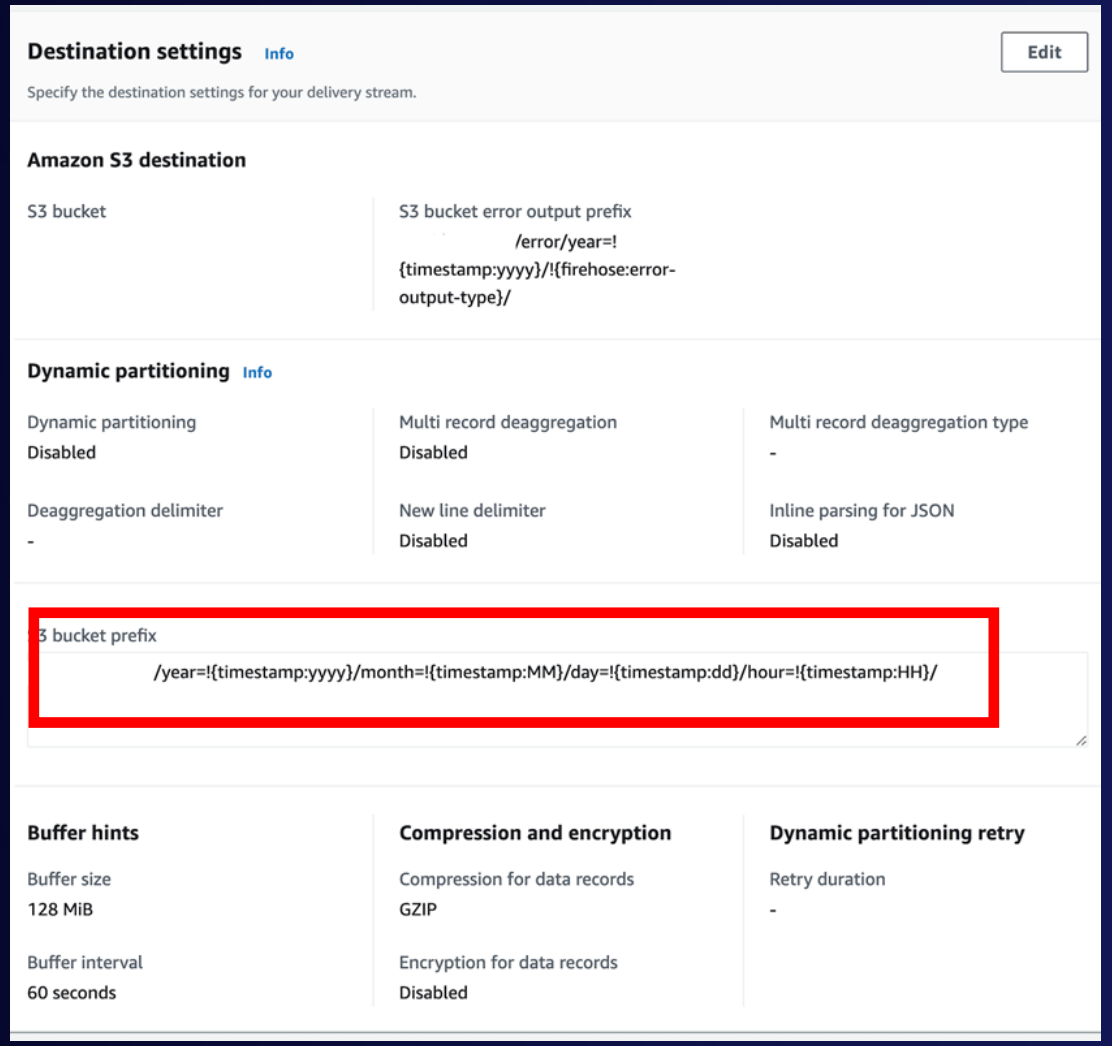
- 데이터를 Amazon S3로 전송하기 전에 수신되는 데이터를 Apache Parquet 및 Apache ORC와 같은 Columnar 기반 형식으로 변환하도록 전송 스트림 구성이 가능함
- 서버리스형 스트림 서비스
- S3로 저장 지원
- GZIP, ZIP, SNAPPY 압축형식 지원
- 날짜 및 시간으로 Partitioning하여 S3에 저장 가능

Amazon Kinesis Data Firehose

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Amazon Kinesis Data Firehose – 설정정보



- S3 Bucket의 Prefix를 설정
- 년/월/일/시간 단위로 파티셔닝
- Athena 로 조회할 경우 파티셔닝된 S3 Object만 읽어서 비용 효율적임
- GZIP으로 S3 Object 압축
- Buffer Size와 Buffer interval로 적절한 object size 설정

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Serverless Framework

- 생성된 Firehose와 DynamoDB Stream의 ARN을 넣어서 Serverless Application 형태로 배포

```yaml
service: demo-app
frameworkVersion: "2"

provider:
  name: aws
  runtime: go1.x
  memorySize: 128
  timeout: 6
  versionFunctions: false
  logRetentionInDays: 30
  deploymentBucket:
    name: serverless-deployment-bucket
  stage: ${self:custom.appEnv}
  region: ap-northeast-2
  iam:
    role:
      statements:
      - Effect: "Allow"
        Action:
          - "firehose:PutRecordBatch"
        Resource:
          - ${self:custom.myDynamoTable1.firehoseArn}
          - ${self:custom.myDynamoTable2.firehoseArn}

custom:
  appEnv: production
  myDynamoTable1:
    name: my-dynamo-table-1-stream
    dynamoArn: arn:aws:dynamodb:ap-northeast-2:1234567890:table/my-dynamo-table-1/stream/2020-02-28T05:00:00.000
    firehoseArn: arn:aws:firehose:ap-northeast-2:1234567890:deliverystream/my-dynamo-table-1-stream
  myDynamoTable2:
    name: my-dynamo-table-2-stream
    dynamoArn: arn:aws:dynamodb:ap-northeast-2:1234567890:table/my-dynamo-table-2/stream/2020-02-28T05:00:00.000
    firehoseArn: arn:aws:firehose:ap-northeast-2:1234567890:deliverystream/my-dynamo-table-2-stream
  tags:
    Environment: "prod"
    Country: "kr"
```

```yaml
functions:
  one-stream:
    handler: bin/stream
    tags: ${self:custom.tags}
    environment:
      DELIVERY_STREAM_NAME: ${self:custom.myDynamoTable1.name}
    events:
      - stream:
          type: dynamodb
          batchSize: 20
          arn: ${self:custom.myDynamoTable1.dynamoArn}
  two-stream:
    handler: bin/stream
    tags: ${self:custom.tags}
    environment:
      DELIVERY_STREAM_NAME: ${self:custom.myDynamoTable2.name}
    events:
      - stream:
          type: dynamodb
          batchSize: 20
          arn: ${self:custom.myDynamoTable2.dynamoArn}

package:
  patterns:
    - '!./**'
    - './bin/**'
```

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Amazon Athena

- Apache Presto 기반의 서버리스형 데이터 조회 서비스
- S3에 저장된 데이터를 조회할 수 있음
- 다양한 형식의 포맷 지원(CSV, JSON, ORC, Avro, Parquet)
- 파티셔닝된 데이터 조회 지원

Amazon Athena

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS Glue

- 완전 관리형 ETL(추출, 변환 및 로드) 서비스
- Athena에서 쿼리할 테이블을 관리
- PostgreSQL DB의 테이블을 Describe해서 스키마를 읽을 수 있음.
- S3에 있는 데이터를 크롤링해서 스키마를 읽을 수 있음.
- Pyspark를 사용한 script를 통해 ETL 진행
- 자체적인 Scheduler로 매일 ETL을 할 수 있음.

AWS Glue

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS Glue

- Glue Job을 통해 PostgreSQL에 있는 Table을 S3로 Dump
- Glue Crawler로 Glue Table 생성
- Athena를 통해 DynamoDB Event와 PostgreSQL Table을 조인하여 원하는 데이터 분석

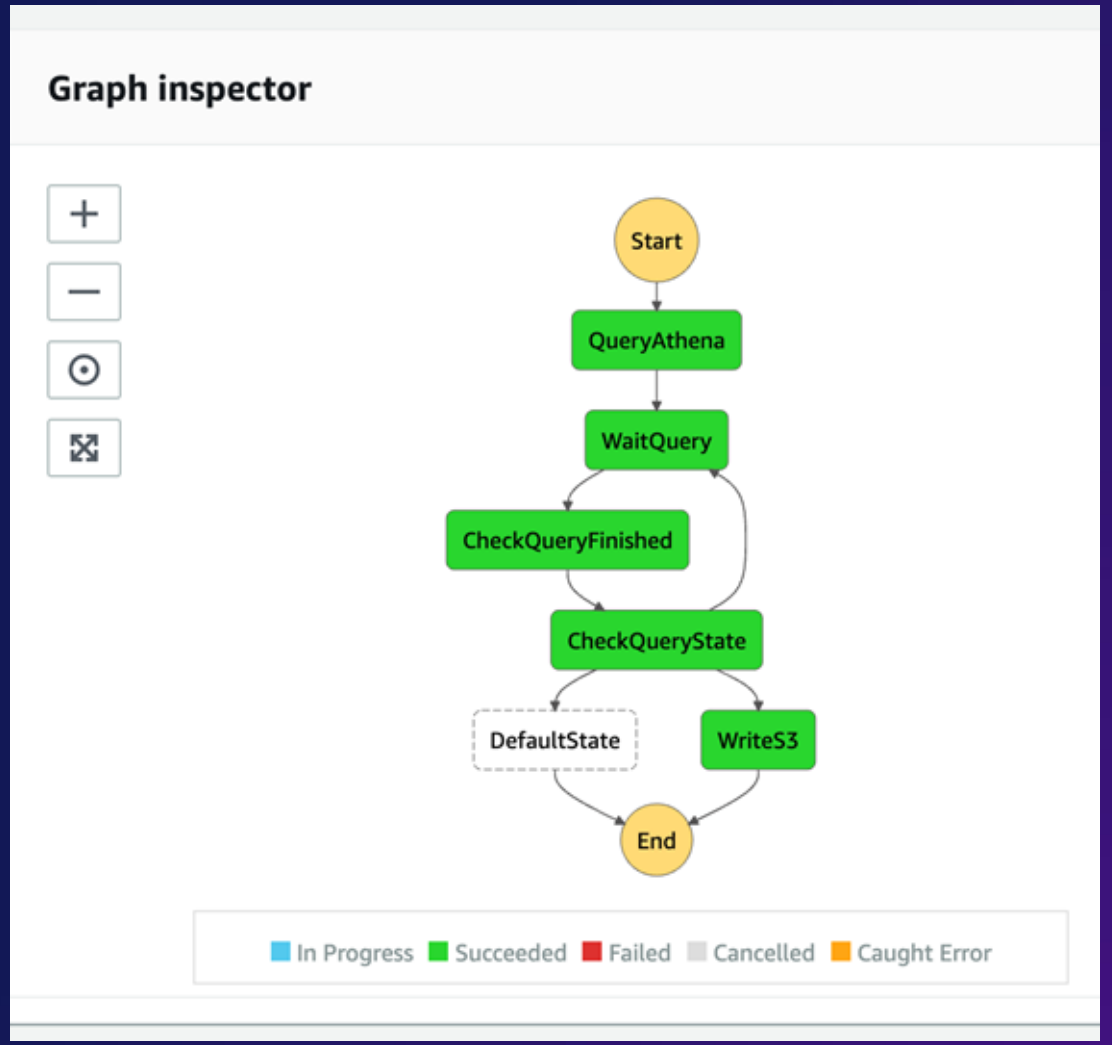
```python
import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job
from datetime import date

## @params: [JOB_NAME]
args = getResolvedOptions(sys.argv, ['JOB_NAME'])

sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)
job.init(args['JOB_NAME'], args)

d = date.today().strftime("year=%Y/month=%m/day=%d")

database_name = "database_name"
table_list = [
    "table_name",
]
target_s3 = "s3://your-s3-bucket/" + d

for table_name in table_list:
    datasource0 = glueContext.create_dynamic_frame.from_catalog(
        database=database_name,
        table_name=table_name,
        transformation_ctx="datasource0")
    applymapping1 = ApplyMapping.apply(
        frame=datasource0,
        mappings=[("code", "string", "code", "string"),
                  ...
                  ("status", "int", "status", "int")],
        transformation_ctx="applymapping1")
    resolvechoice2 = ResolveChoice.apply(frame=applymapping1,
                                         choice="make_struct",
                                         transformation_ctx="resolvechoice2")
    dropnullfields3 = DropNullFields.apply(
        frame=resolvechoice2, transformation_ctx="dropnullfields3")
    datasink4 = glueContext.write_dynamic_frame.from_options(
        frame=dropnullfields3,
        connection_type="s3",
        connection_options={"path": target_s3},
        format="parquet",
        transformation_ctx="datasink4")

job.commit()
```

RDB에서 S3로 보내는 코드

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

데이터 분석하기

- 분석 통계를 뽑는 Workflow는 AWS Step functions를 사용
- S3에 있는 데이터를 Athena를 통해 쿼리하고 결과를 S3에 저장
- Eventbridge Scheduler를 통해 매일 자정에 Trigger됨



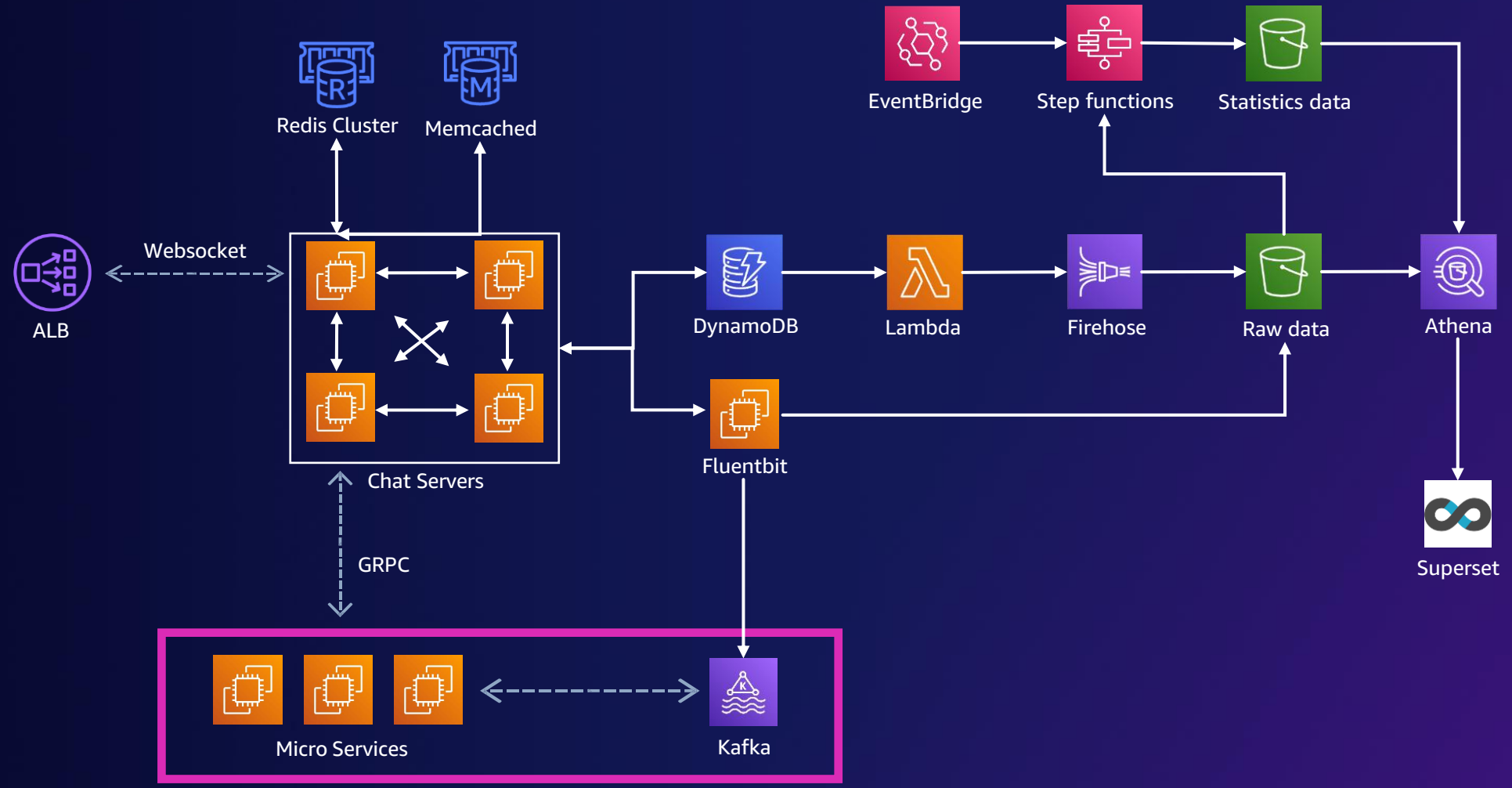
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

채팅 데이터를 다른 서비스에서 활용하기

- Fluentbit를 통해 Managed Kafka로 보냄
- Fluentbit는 안정적으로 이벤트를 전달하는 버퍼역할.
- 다양한 output을 지원하는데, kafka/s3/firehose 등 대부분을 지원함
- Kafka에 보내주는 것을 보장해주기 때문에 kafka장애 시에도 이벤트가 유실되는 것을 방지함

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

채팅 데이터를 다른 서비스에서 활용하기



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

FluentBit - Publisher

- 초경량
- 다양한 Output을 둘 수 있음(Kafka, S3, Firehose, Datadog, ElasticSearch...)
- Filter/Parser 등 지원
- TCP를 사용하는 자체 프로토콜 지원

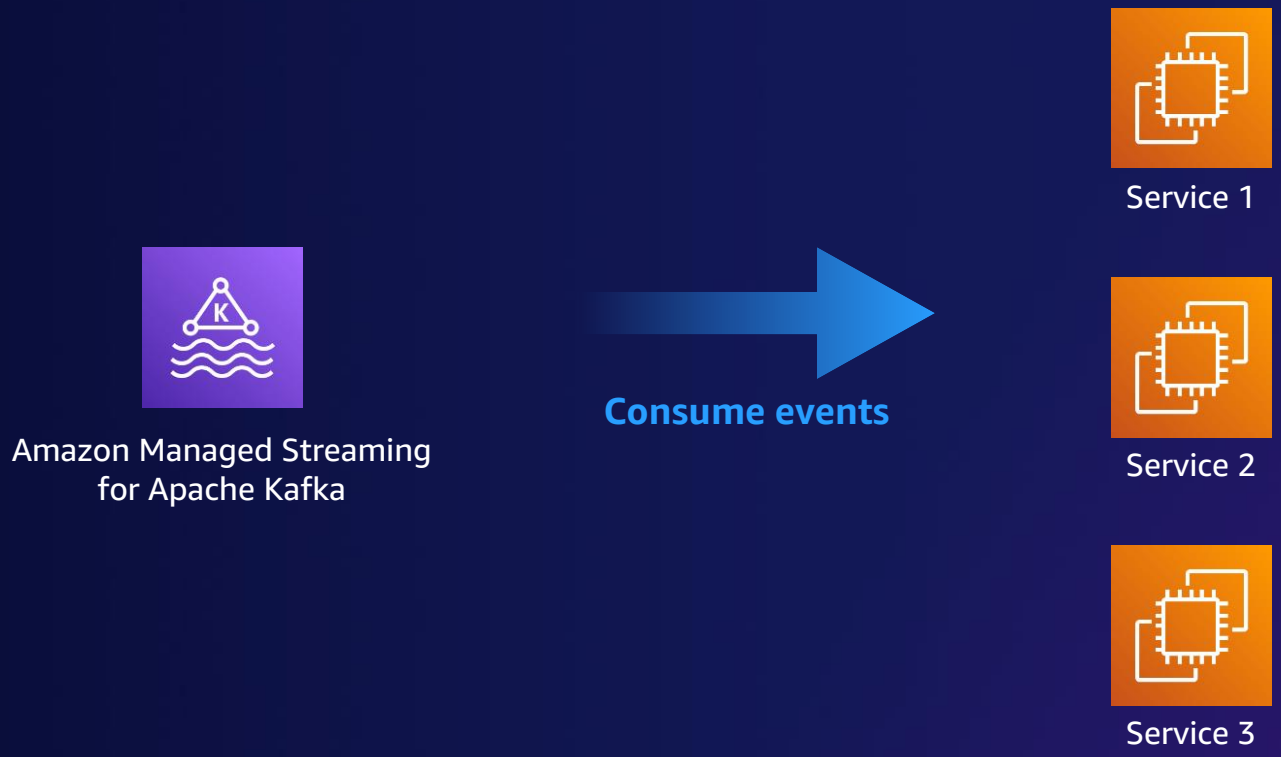
```
logger, err := fluent.New(fluent.Config{
    FluentHost:   cfg.FluentHost,
    FluentPort:   cfg.FluentPort,
    WriteTimeout: time.Second * 1,
    MaxRetry:     3,
    Async:        false,
})
if err != nil {
    return nil, errors.Wrap(err, "fluent.New() failed")
}
```

```
if err := p.logger.Post(match, map[string]interface{}{
    "event": b,
}); err != nil {
    return errors.Wrap(err, "logger.Post() failed")
}
return nil
```

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

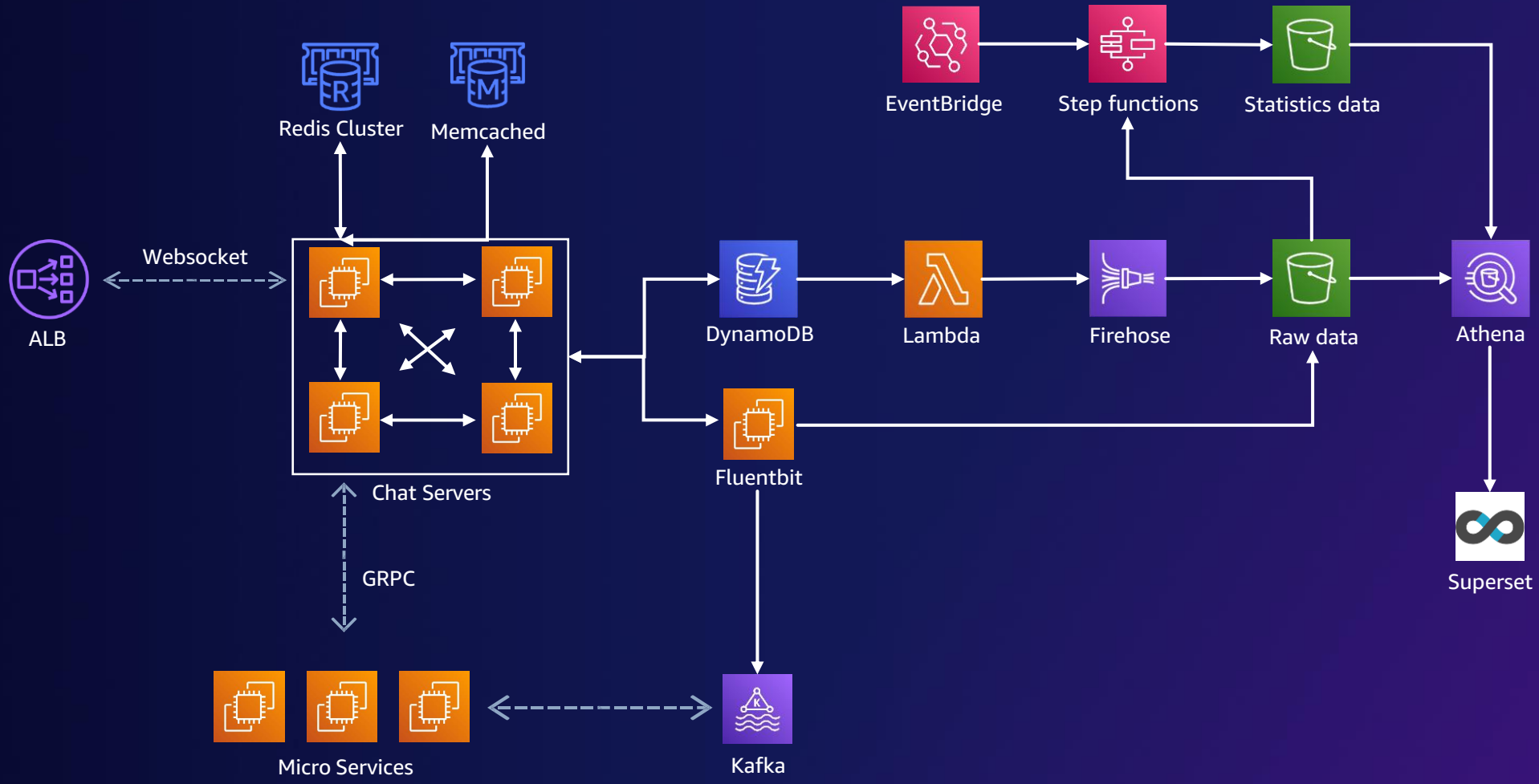
채팅 데이터를 다른 서비스에서 활용하기

- 각 서비스는 kafka에 publish된 event에 대해서 consumer를 구현하여 소비함



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

현재 채팅 시스템 아키텍처



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

맺음말

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

맺음말

아키텍처 현대화 이점

MSA 아키텍처 필요성

서버리스 이점

채팅 시스템 현대화 아키텍처

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

여러분의 소중한 피드백을 기다립니다.

강연 종료 후, 강연 평가에 참여해 주세요!

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

감사합니다

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

비즈니스 커뮤니케이션의 변화

원격 의료 및 식료품 배달과 같은 서비스로 고객들에게 더 많은 유연성을 제공합니다.

고객들은 그들이 원하는 방식으로 의사소통하기를 원합니다.

비디오는 다양한 고객들의 요구를 충족하기 위한 공통 메커니즘이 되었습니다.

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS 커뮤니케이션 개발자 서비스들은 고객들이 커뮤니케이션 지원 애플리케이션과 서비스를 구축할 수 있도록 지원합니다.

AWS 커뮤니케이션 개발자 서비스

트랜잭션 | 비동기 | 실시간

Amazon Chime SDK

Amazon Pinpoint

Amazon SES

Build and host your own **OR** build your own and host with AWS

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.