
AWS
re:Invent

ARC306

Building reactive systems on AWS

Sascha Möllering

Specialist Solutions Architect, Containers

AWS EMEA

Michael Hanisch

Head of Technology, DACH

AWS EMEA

AWS re:Invent

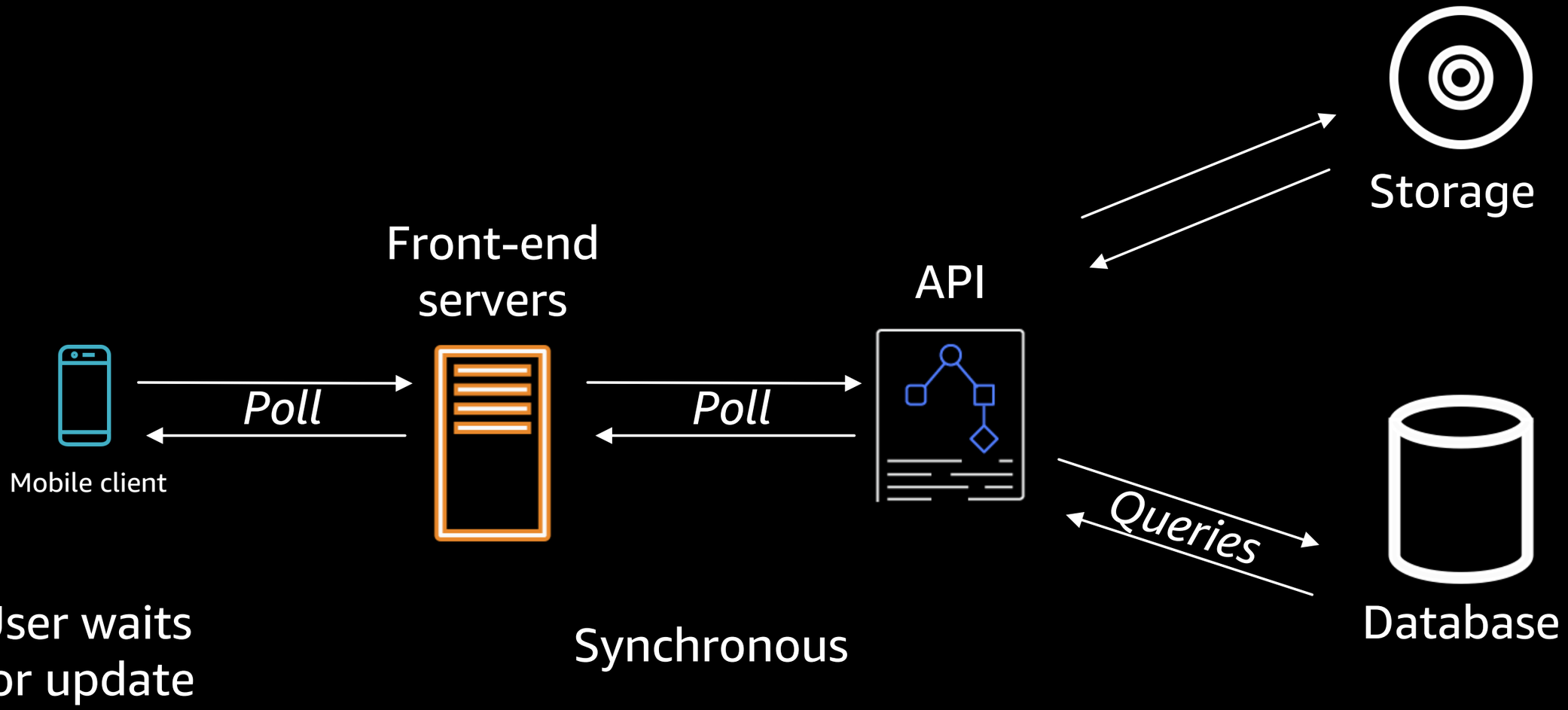
© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

What are reactive systems?



© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

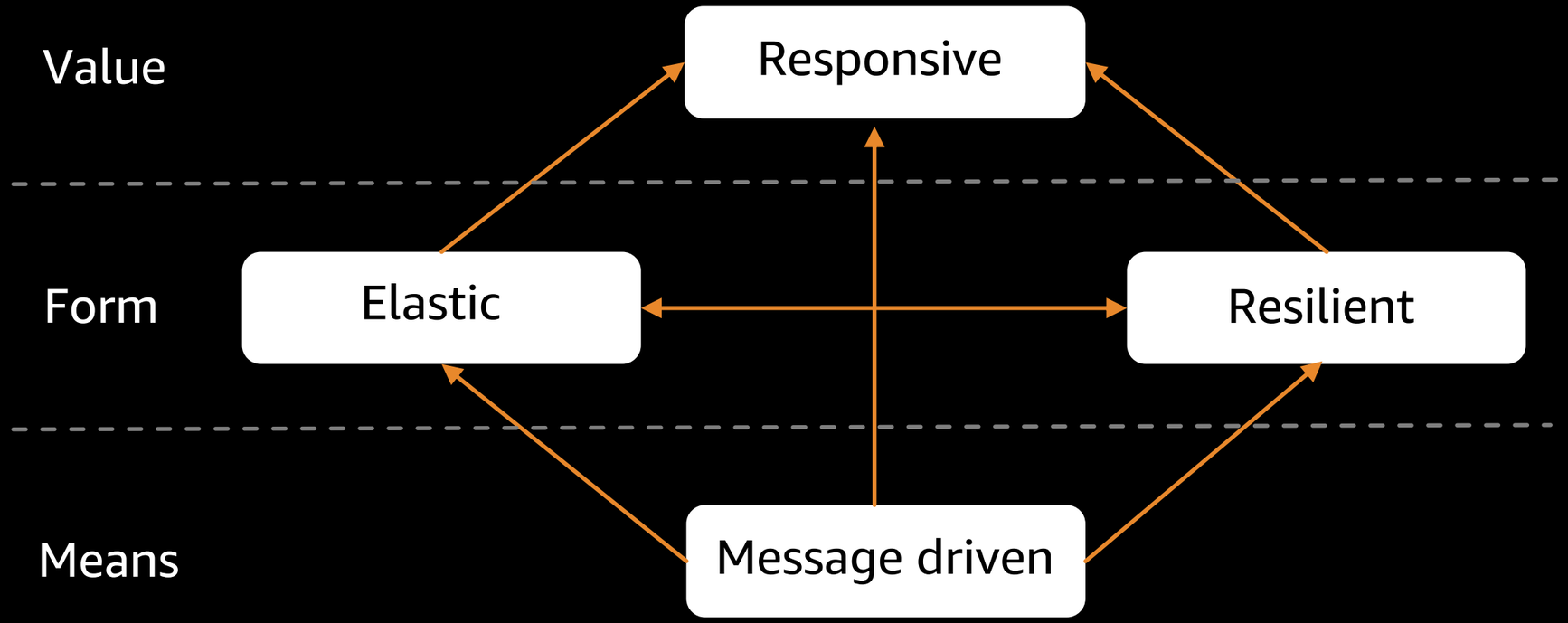
N-tier architecture





Software moves faster today

Reactive applications



Reactive systems

Reactive programming

"A development model focusing on the observation of data streams, reacting on changes, and propagating them."

Reactive system

"An architecture style used to build responsive and robust distributed systems based on asynchronous message passing."

Reactive systems design principles

- Asynchronous message passing
- Nonblocking
- Loosely coupled
- Push-based

Typical use cases



AdTech



Internet of Things



Ecommerce



Gaming



Social networks

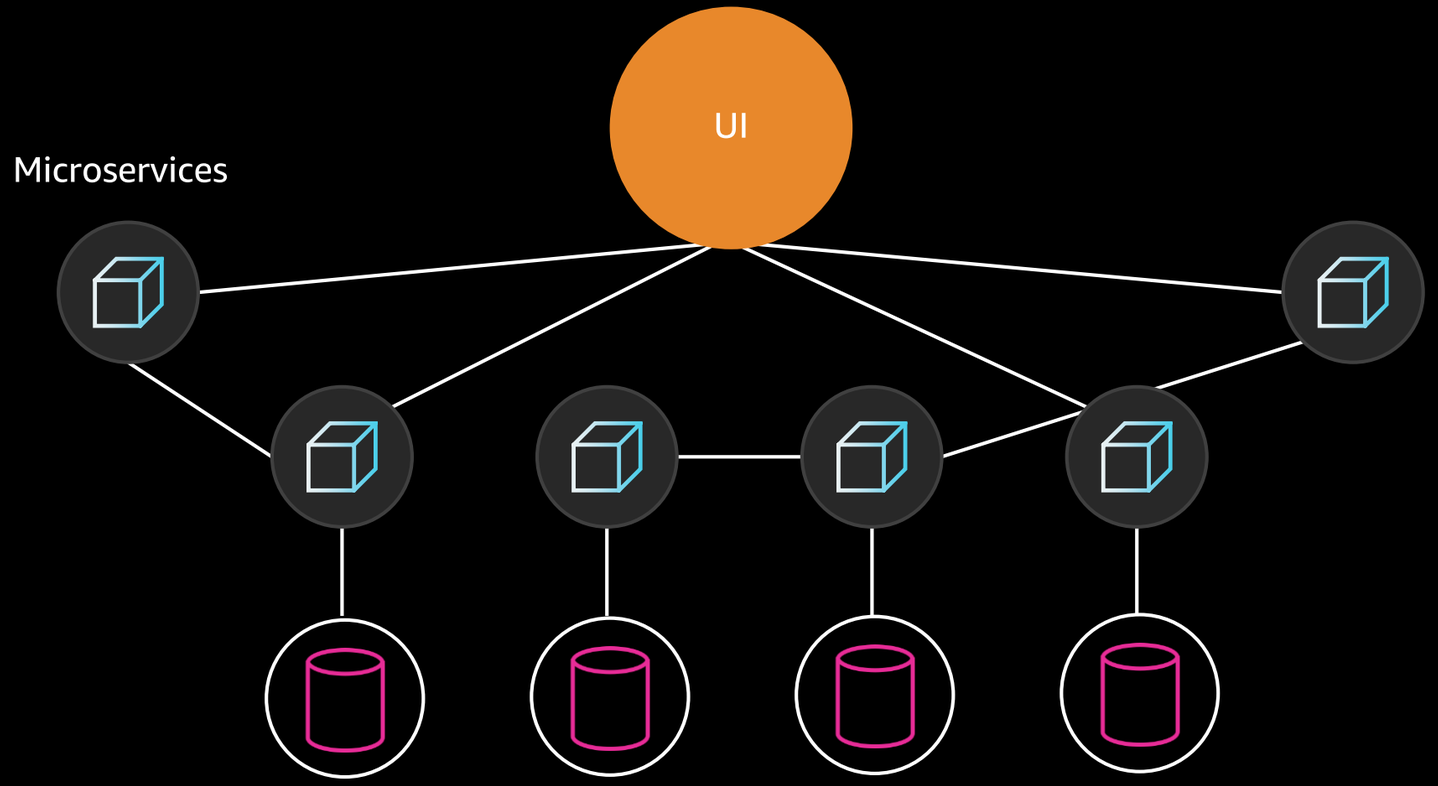
Microservices



AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Reactive systems and microservices



Microservices architecture

Reactive systems on AWS



AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Responsive

“The system responds in a timely manner if at all possible.”



Amazon Elastic Container Service





AWS Fargate



AWS Lambda

Resilient

“The system stays responsive in the face of failure.”



Amazon Elastic Container Service



AWS Lambda



Amazon Kinesis Data Streams



Amazon ElastiCache for Redis



Amazon DynamoDB





AWS Fargate

Elastic

“The system stays responsive under varying workload.”



Amazon Elastic Container Service



AWS Lambda



Amazon DynamoDB





AWS Fargate

Message driven

“Reactive systems rely on asynchronous message passing to establish a boundary between components that ensures loose coupling, isolation, and location transparency.”

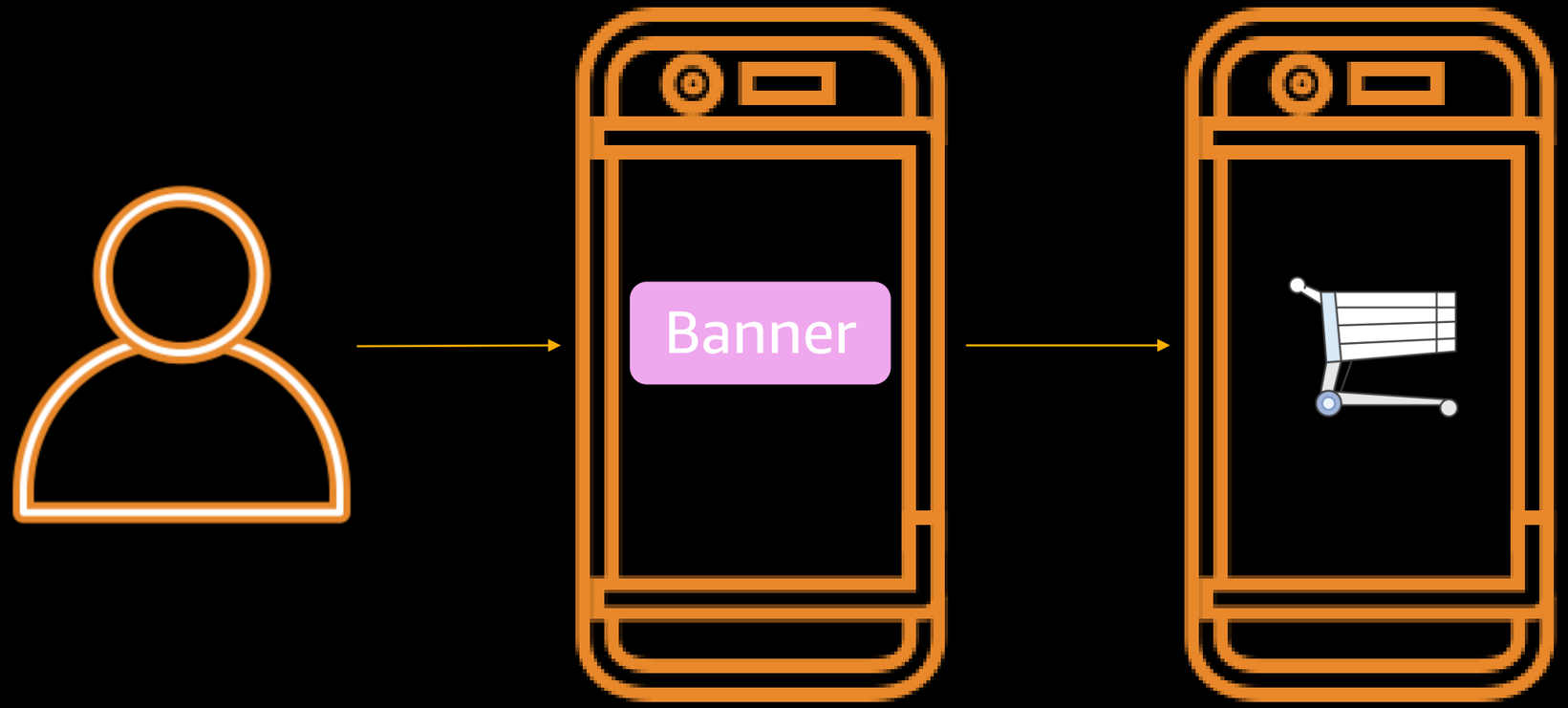


Amazon Kinesis Data Streams



Amazon ElastiCache for Redis

Example use case



Customer visits website

Customer clicks on a banner

Customer buys from online shop

https://github.com/aws-samples/reactive-refarch-cloudformation

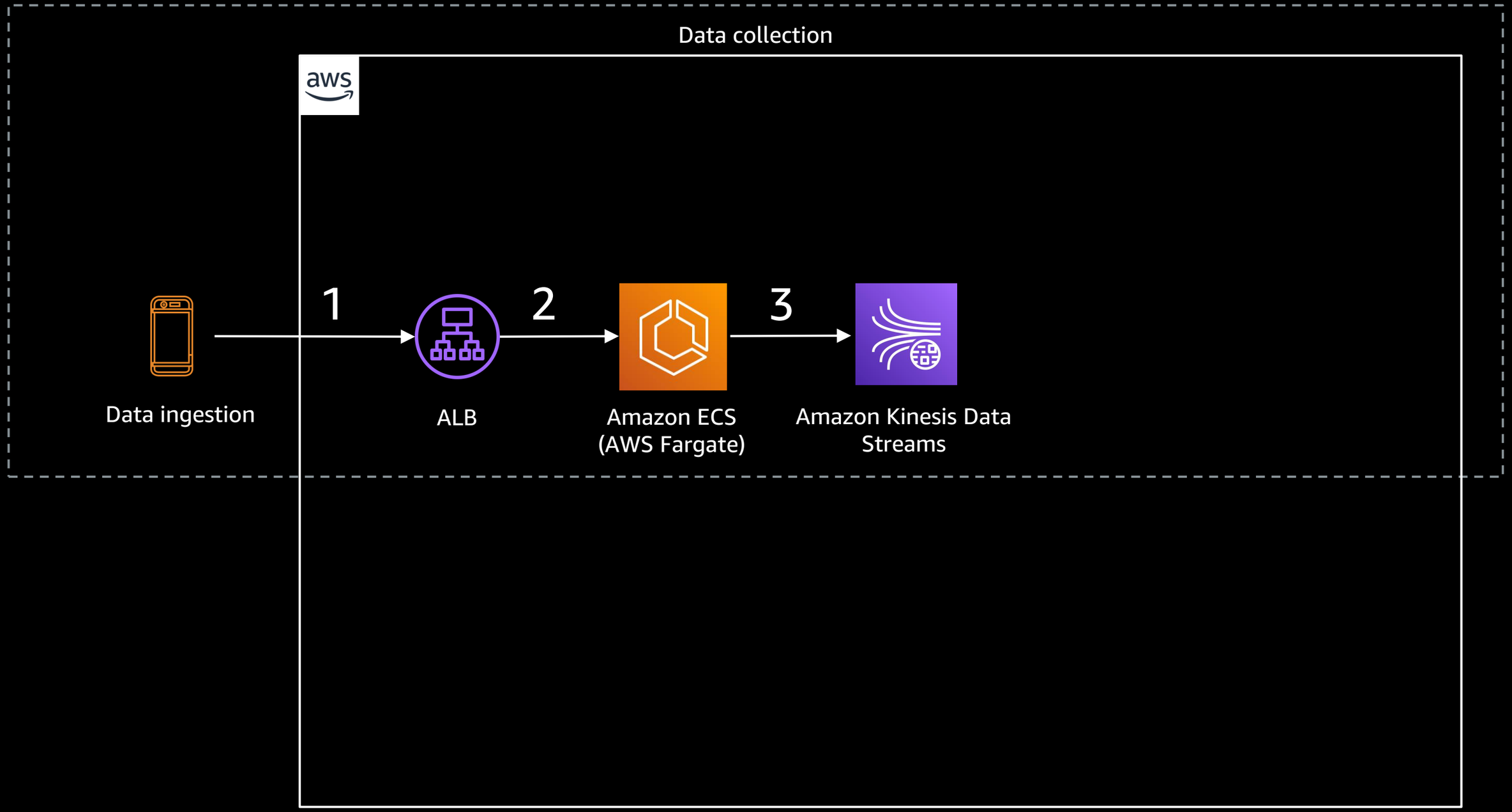
High-level architecture

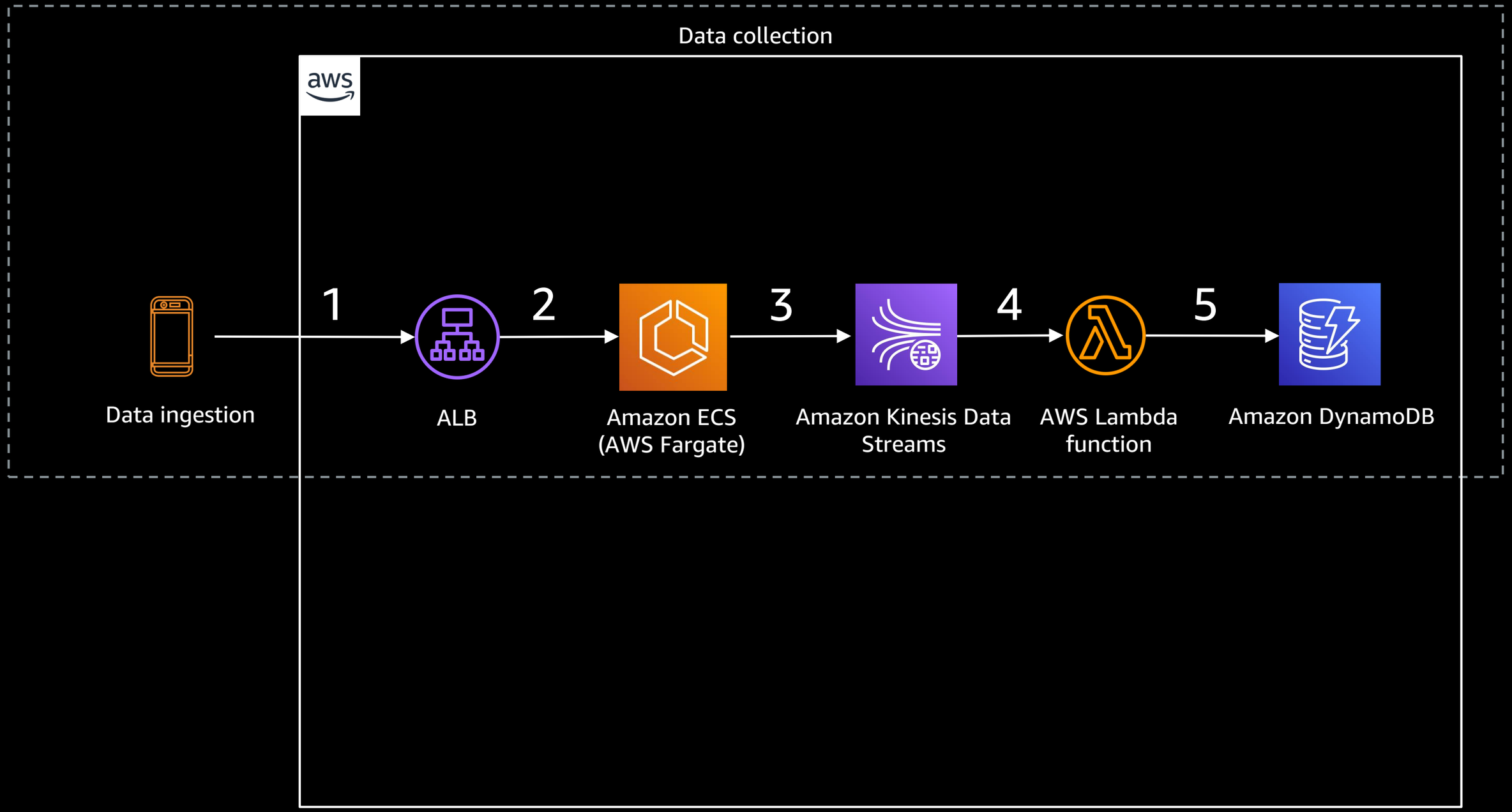


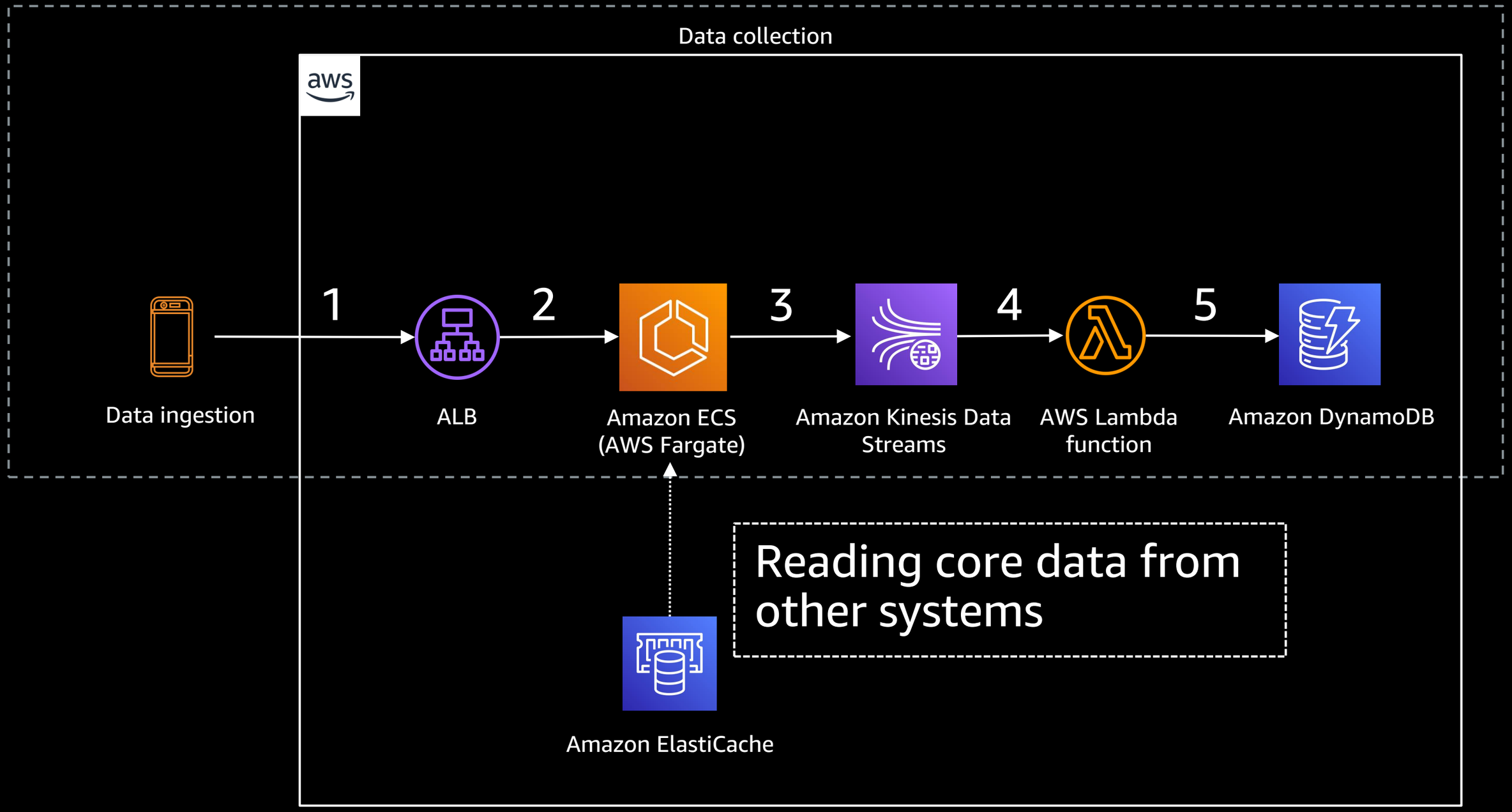
© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

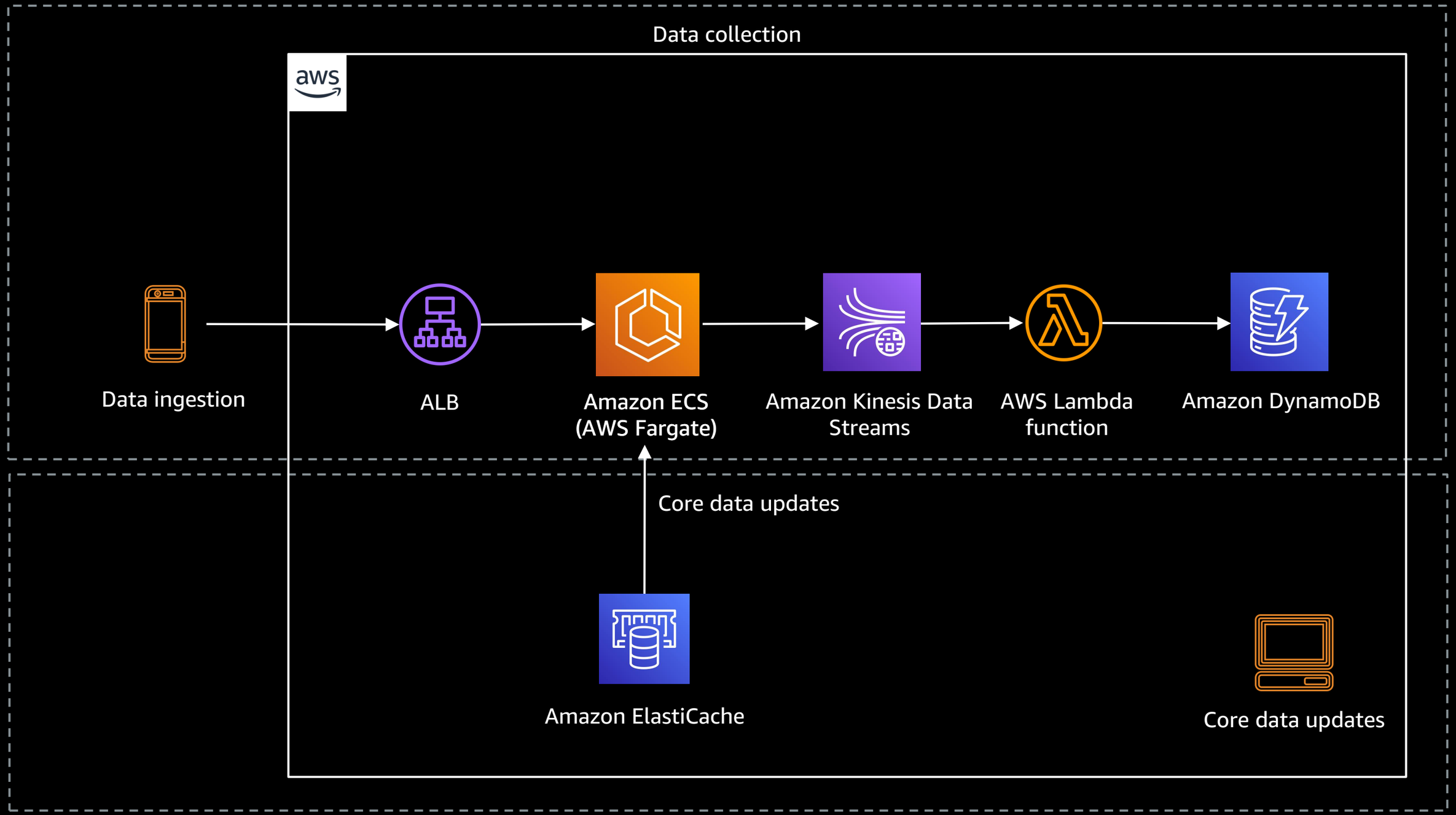

Data collection
aws
1
Data ingestion
Application Load Balancer (ALB)

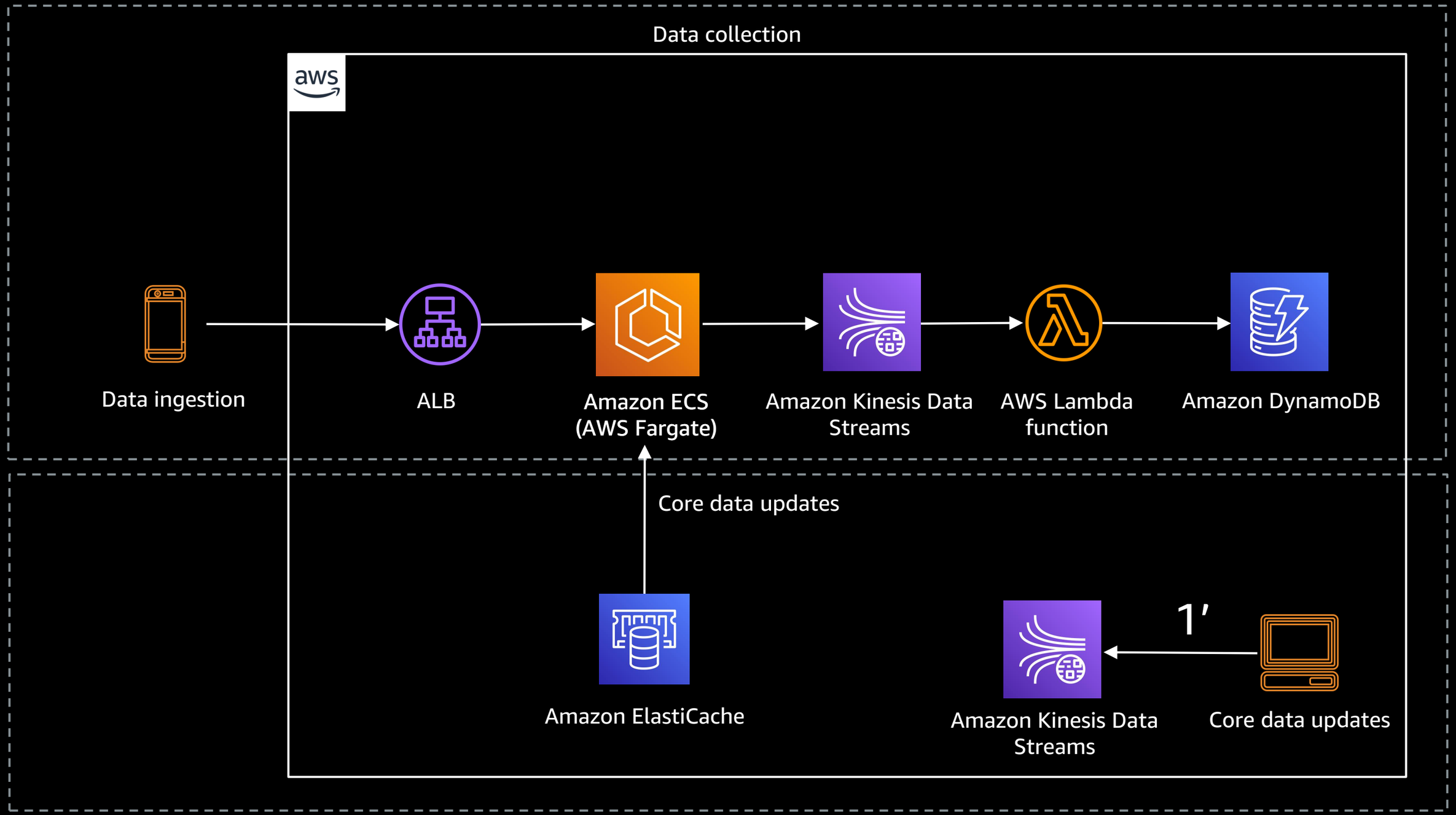

Data collection
aws
1
2
Data ingestion
ALB
Amazon ECS
(AWS Fargate)

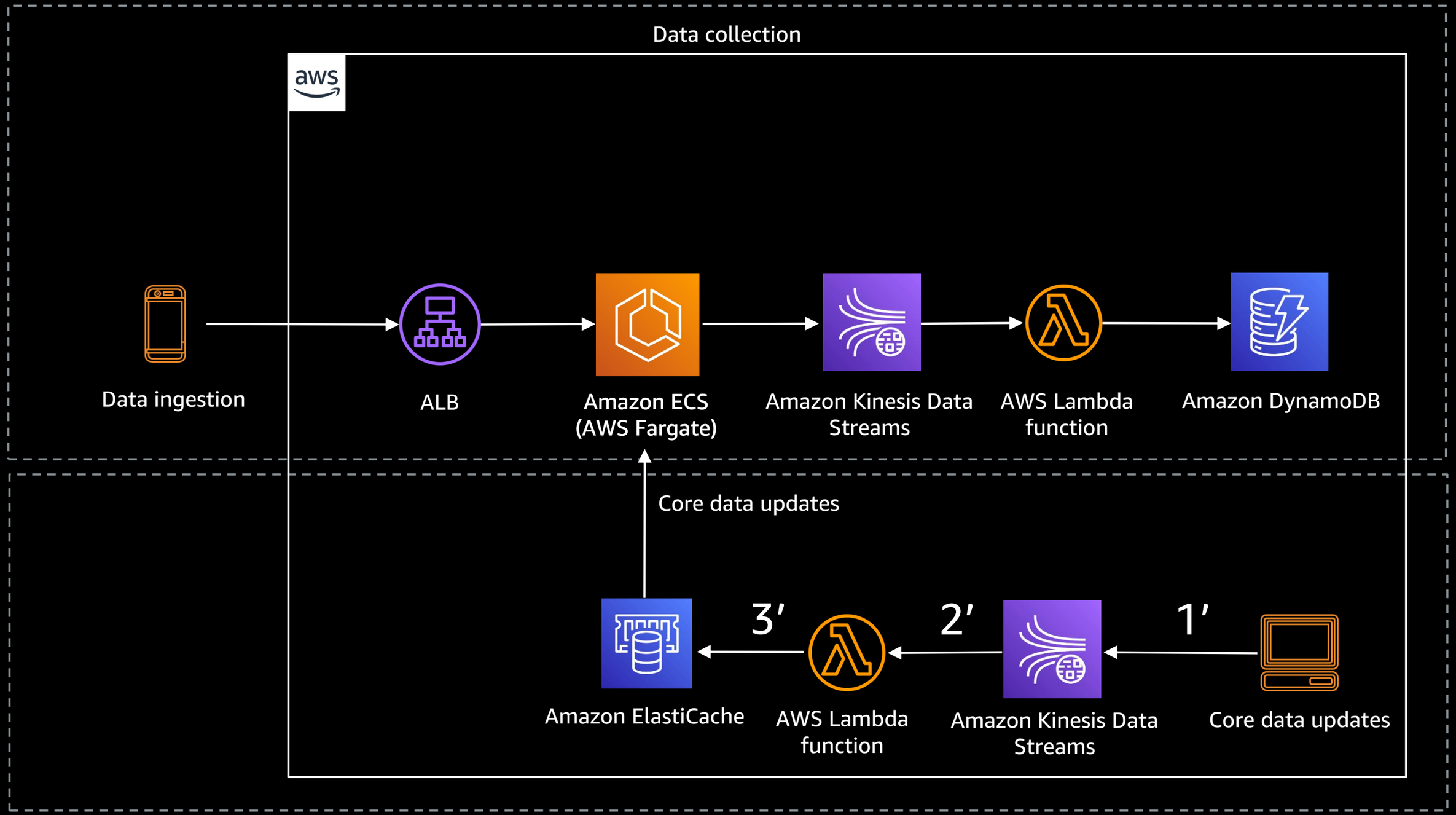

Data collection
aws
1
2
3
Data ingestion
ALB
Amazon ECS
(AWS Fargate)
Amazon Kinesis Data
Streams


Data collection
aws
1
2
3
4
5
Data ingestion
ALB
Amazon ECS
(AWS Fargate)
Amazon Kinesis Data
Streams
AWS Lambda
function
Amazon DynamoDB


Data collection
aws
1
2
3
4
5
Data ingestion
ALB
Amazon ECS
(AWS Fargate)
Amazon Kinesis Data
Streams
AWS Lambda
function
Amazon DynamoDB
Reading core data from
other systems
Amazon ElastiCache


Data collection
aws
Data ingestion
ALB
Amazon ECS
(AWS Fargate)
Amazon Kinesis Data
Streams
AWS Lambda
function
Amazon DynamoDB
Core data updates
Amazon ElastiCache
Core data updates


Data collection
aws
Data ingestion
ALB
Amazon ECS
(AWS Fargate)
Amazon Kinesis Data
Streams
AWS Lambda
function
Amazon DynamoDB
Core data updates
Amazon ElastiCache
1'
Amazon Kinesis Data
Streams
Core data updates

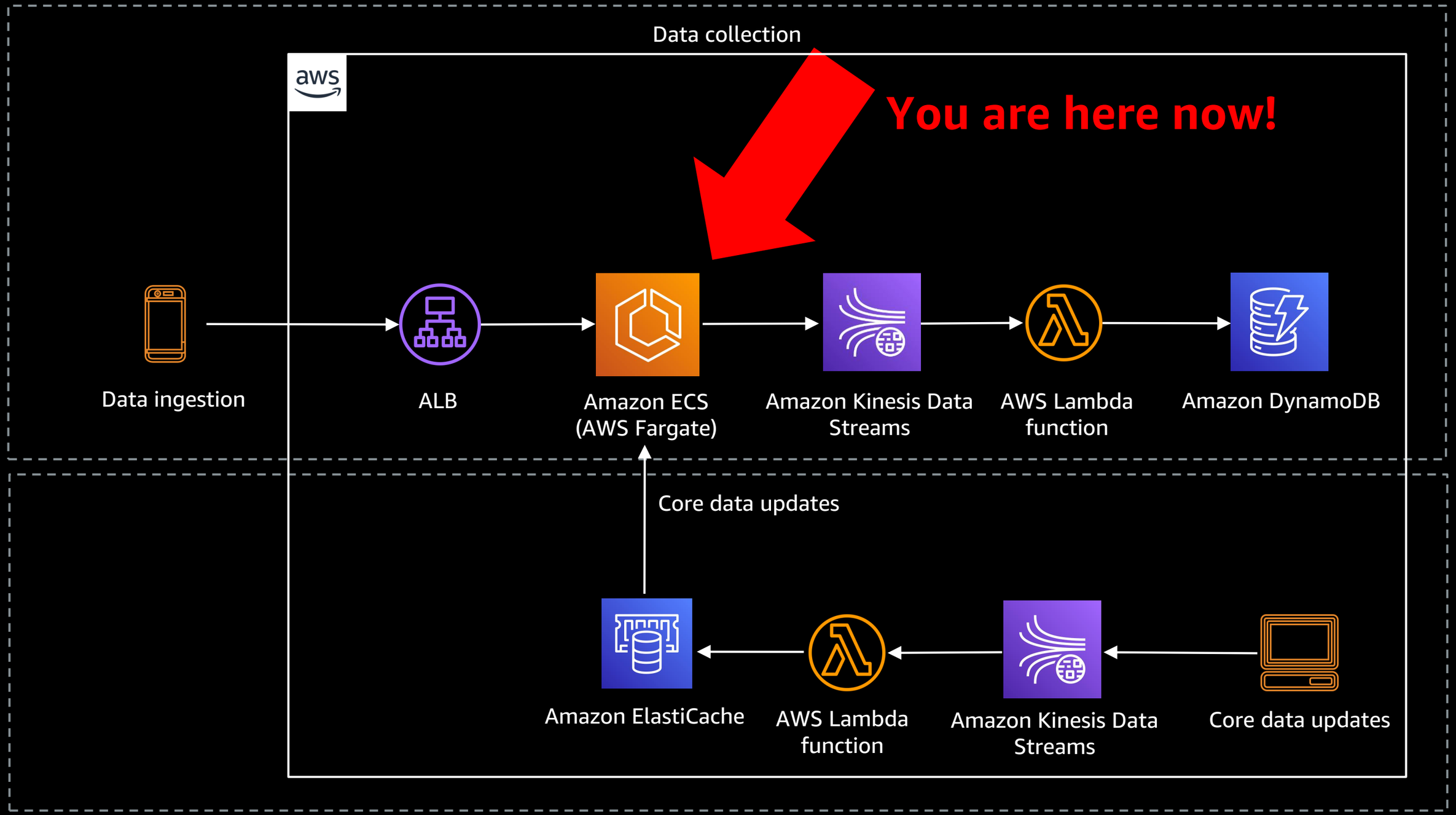

Data collection
aws
Data ingestion
ALB
Amazon ECS
(AWS Fargate)
Amazon Kinesis Data
Streams
AWS Lambda
function
Amazon DynamoDB
Core data updates
Amazon ElastiCache
3′
AWS Lambda
function
2′
Amazon Kinesis Data
Streams
1′
Core data updates

Application architecture



AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.


Data collection
aws
You are here now!
Data ingestion
ALB
Amazon ECS
(AWS Fargate)
Amazon Kinesis Data
Streams
AWS Lambda
function
Amazon DynamoDB
Core data updates
Amazon ElastiCache
AWS Lambda
function
Amazon Kinesis Data
Streams
Core data updates

Design principles

- Use push instead of pull
- Avoid blocking calls
- Decouple your services using async message passing
- Keep state in managed services
- Use caching

Application architecture

“Verticles” as main actors

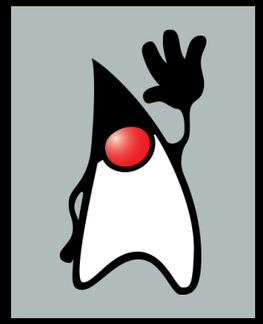
Event-driven, asynchronous

Nonblocking IO

High scale with limited resources

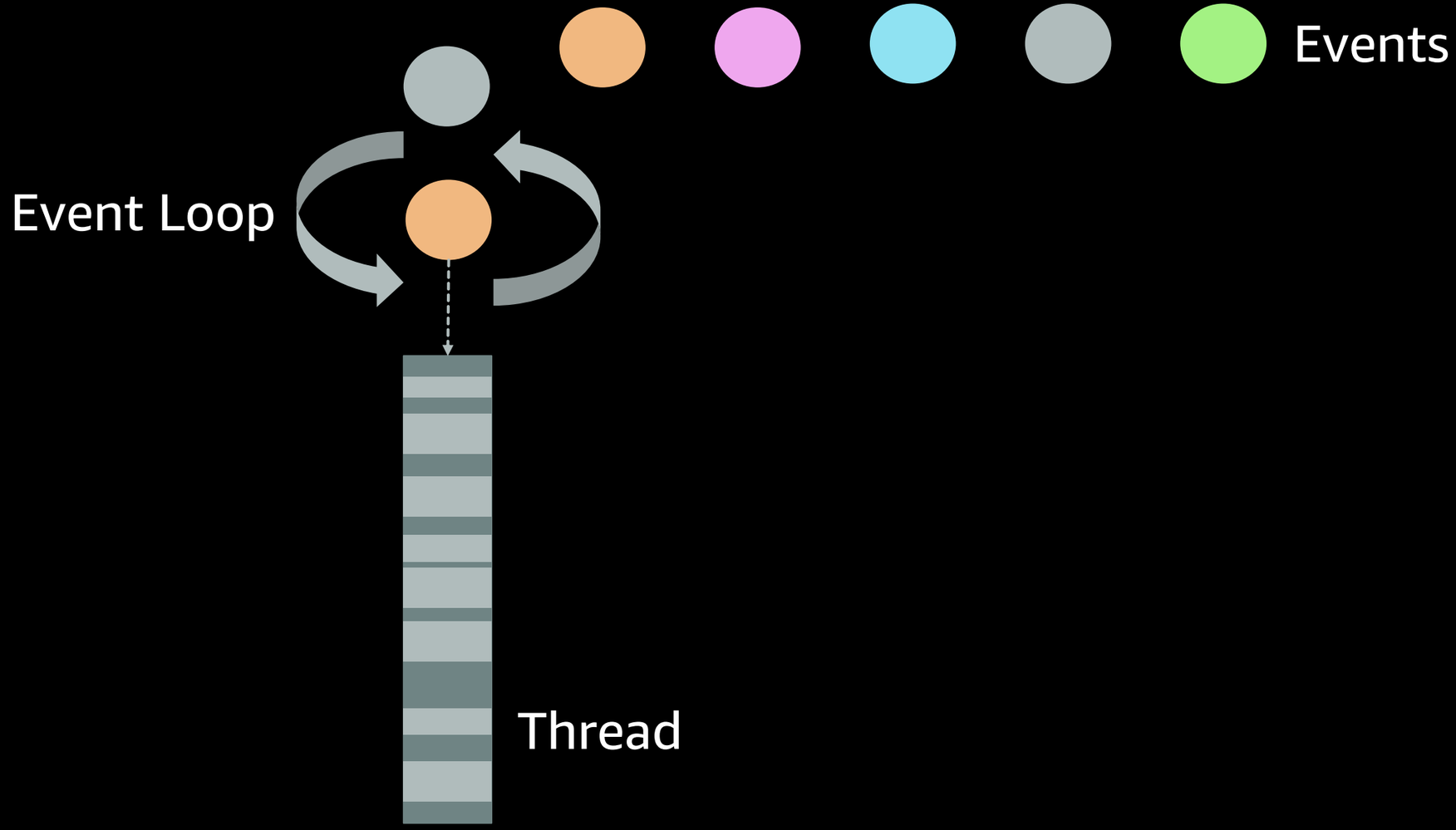
Polyglot language support





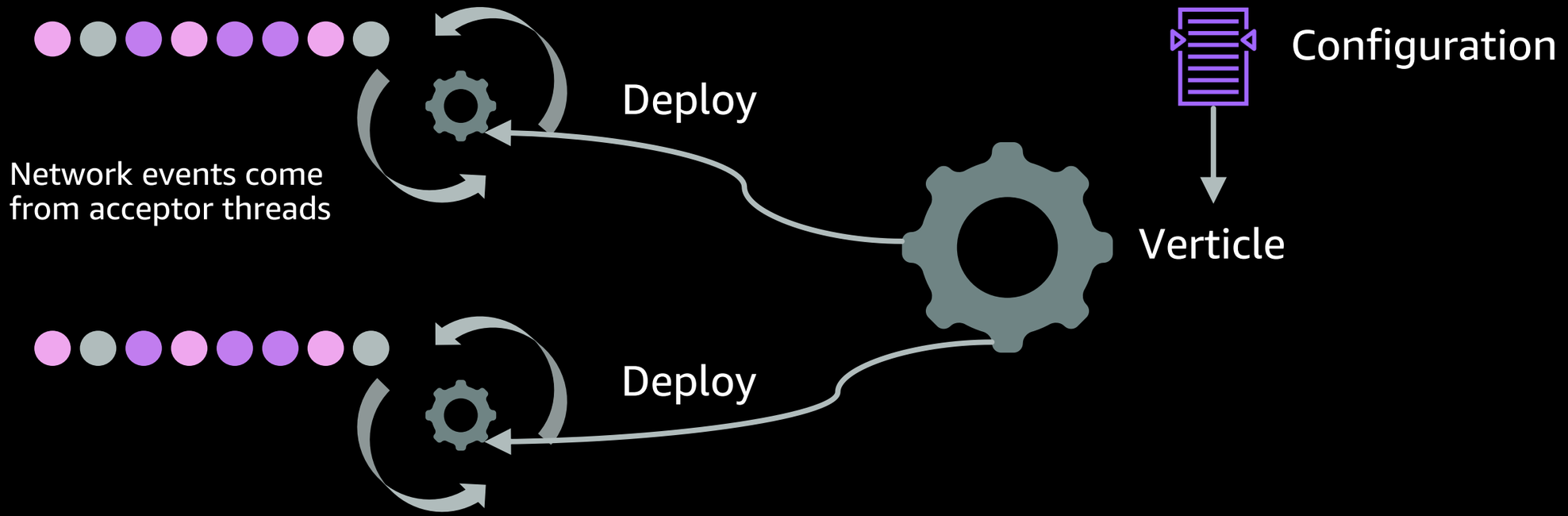
GraalVM™

Application architecture



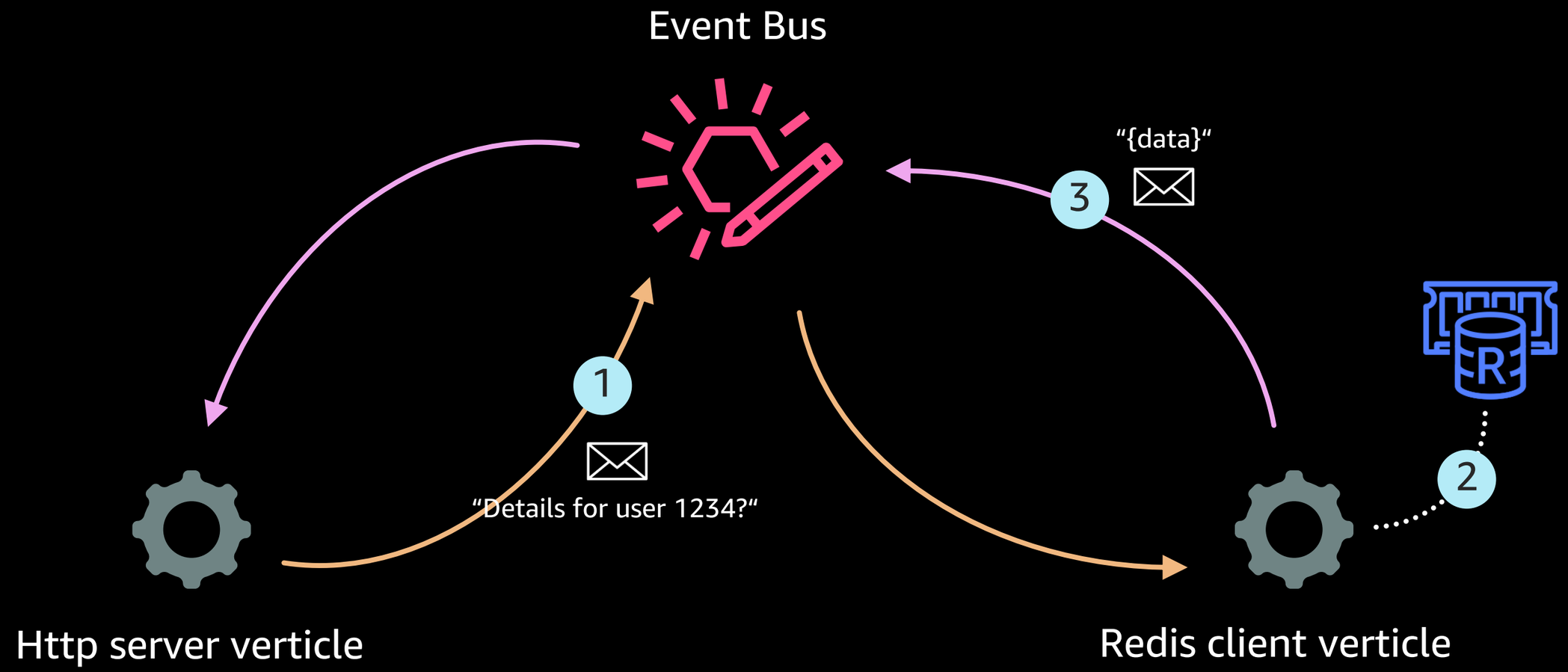
Source: http://vertx.io/docs/guide-for-java-devs/

Application architecture

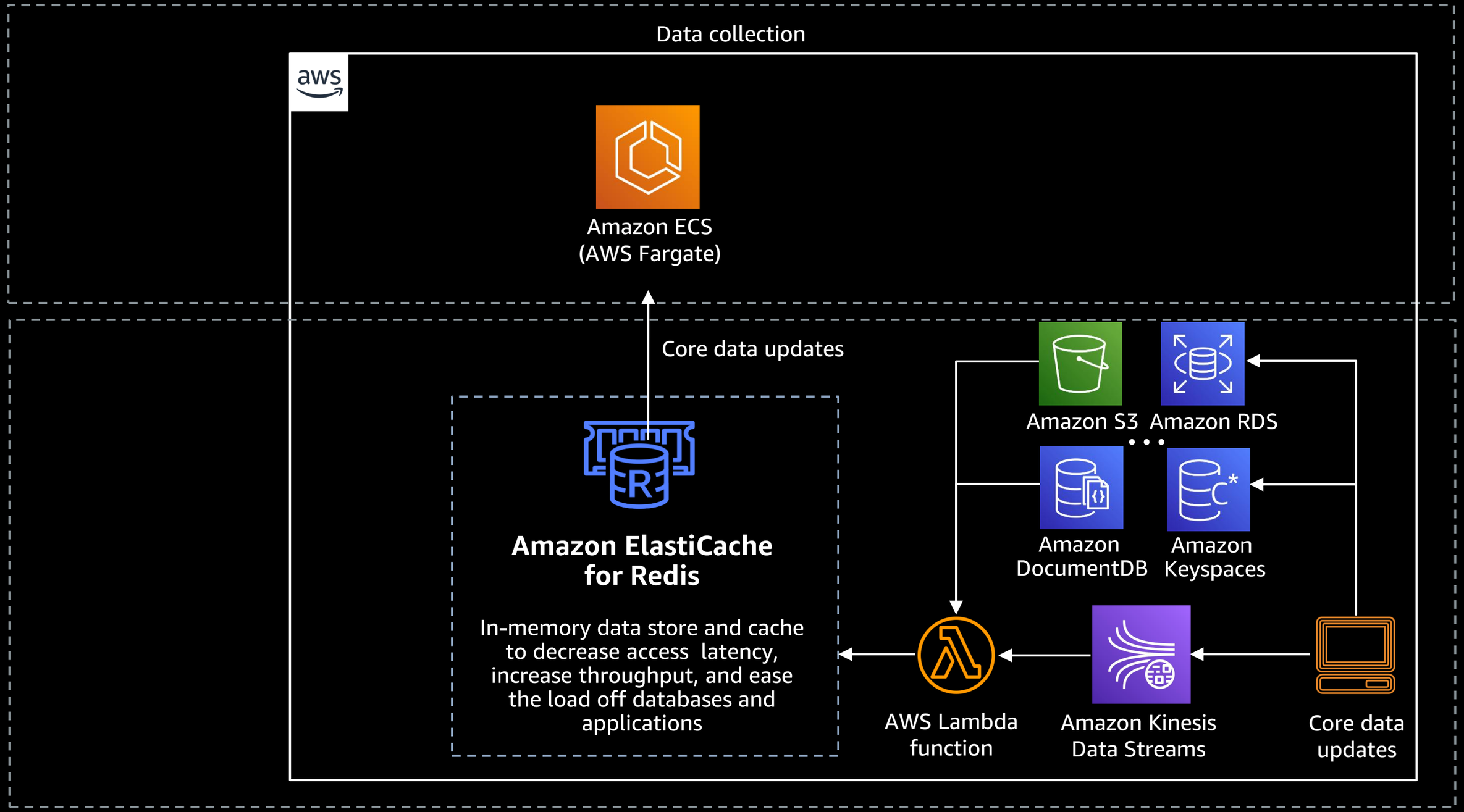


Source: http://vertx.io/docs/guide-for-java-devs/

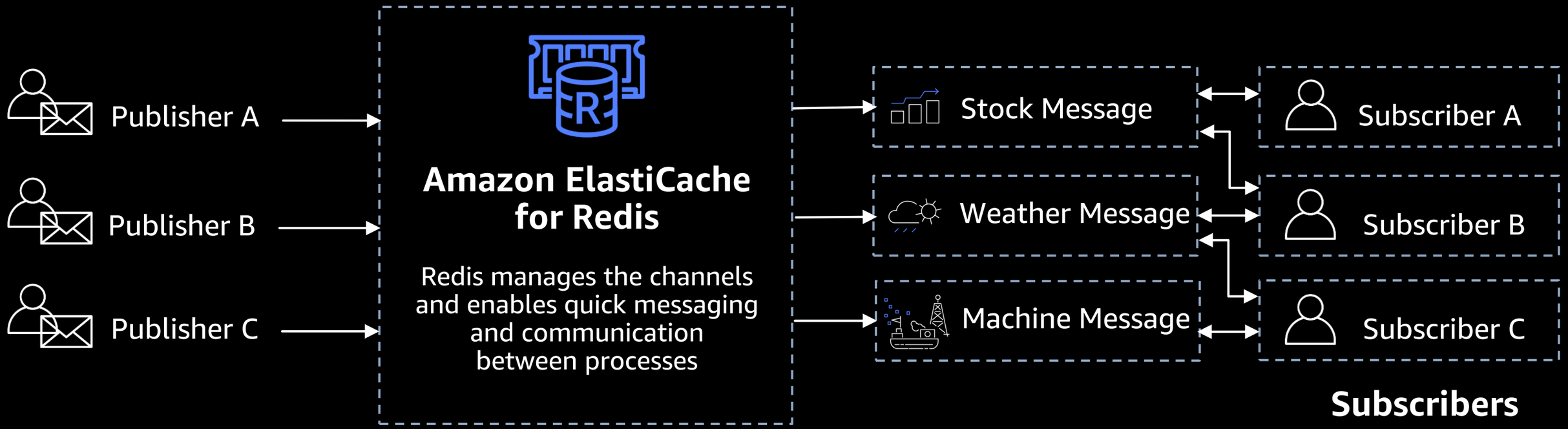
Application architecture



Source: http://vertx.io/docs/guide-for-java-devs/


Data collection
aws
Amazon ECS
(AWS Fargate)
Core data updates
Amazon ElastiCache
for Redis
In-memory data store and cache
to decrease access latency,
increase throughput, and ease
the load off databases and
applications
Amazon S3
Amazon RDS
Amazon
DocumentDB
Amazon
Keyspaces
AWS Lambda
function
Amazon Kinesis
Data Streams
Core data
updates

Application architecture



Subscribers

Subscribers can listen to any channel they've subscribed to

Example code: Redis verticle

```
void registerToEventBusForPubSub(final EventBus eb, final RedisClient redis) {
    vertx.eventBus().<JsonObject>consumer(REDIS_PUBSUB_CHANNEL_VERTX, received -> {
        JsonObject value = received.body().getJsonObject("value");

        String message = value.getString("message");

        JsonObject jsonObject = new JsonObject(message);
        eb.send(CACHE_REDIS_EVENTBUS_ADDRESS, jsonObject);
    });

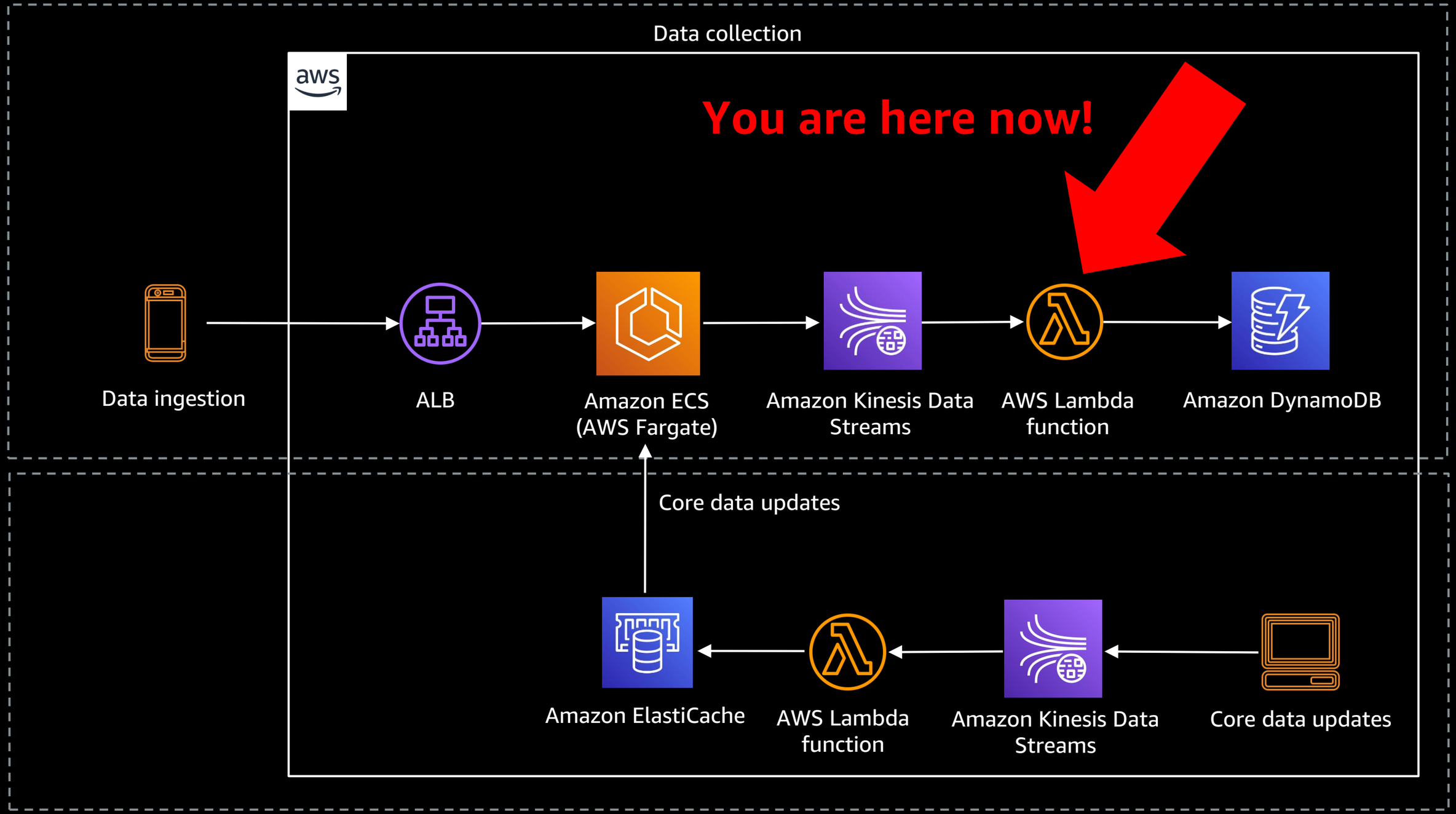
    redis.subscribe(Constants.REDIS_PUBSUB_CHANNEL, res -> {
        if (res.succeeded()) {
            LOGGER.info("Subscribed to " + Constants.REDIS_PUBSUB_CHANNEL);
        } else {
            LOGGER.info(res.cause());
        }
    });
}
```

Subscribe to EB

Consume data from event bus

Send data to cache verticle

Subscribe to Redis channel


Data collection
aws
You are here now!
Data ingestion
ALB
Amazon ECS
(AWS Fargate)
Amazon Kinesis Data
Streams
AWS Lambda
function
Amazon DynamoDB
Core data updates
Amazon ElastiCache
AWS Lambda
function
Amazon Kinesis Data
Streams
Core data updates

Application architecture

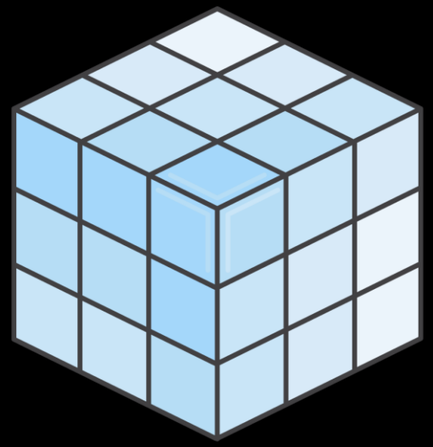


AWS Lambda

Lambda considerations and best practices

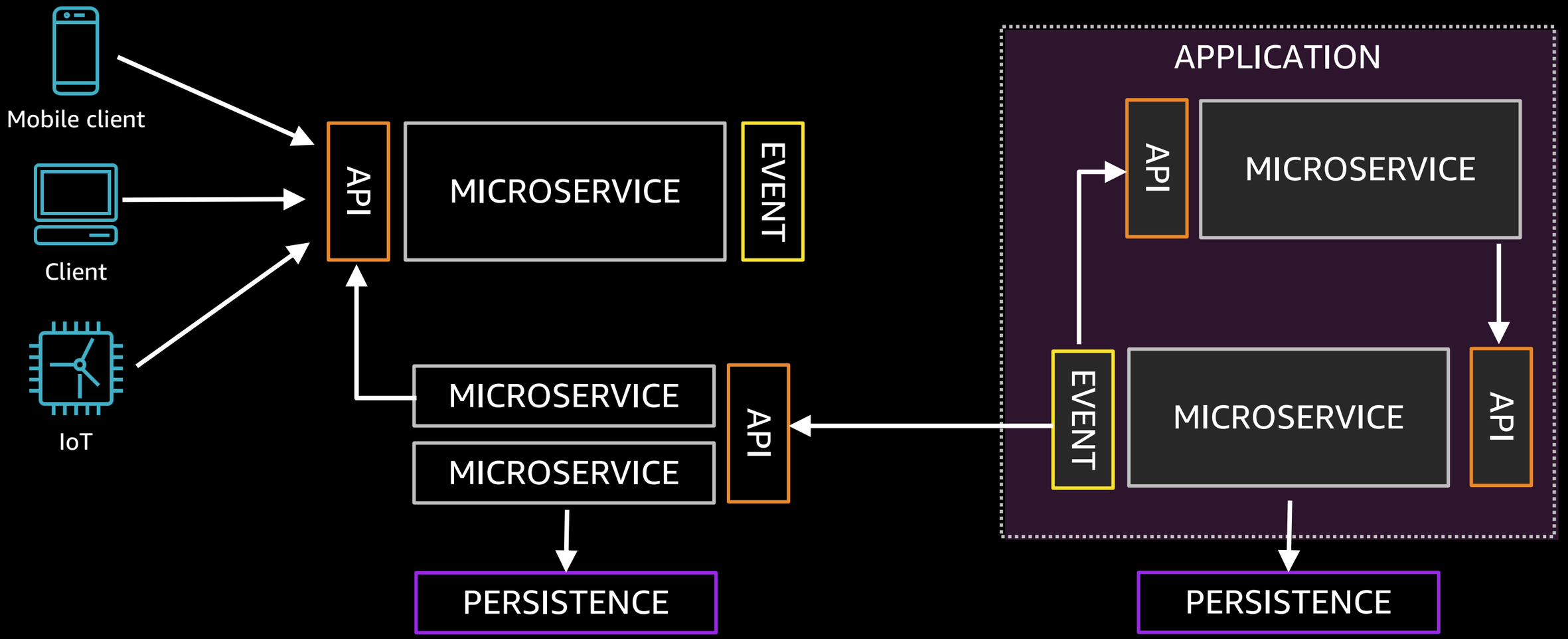
AWS Lambda is stateless – architect accordingly

- Assume no affinity with underlying compute infrastructure
- Local filesystem access (/tmp) and child process may not extend beyond the lifetime of the Lambda request
- AWS Lambda now supports Amazon Elastic File System (Amazon EFS)





Go build something!



Resources

Reference Architecture

Try it out for yourself, deploy the sample architecture

https://github.com/aws-samples/reactive-refarch-cloudformation

Reactive Microservices Architectures on AWS

This blog post provides more explanation about the design of the reference architecture

https://aws.amazon.com/blogs/architecture/reactive-microservices-architecture-on-aws/

Using GraalVM to Build Minimal Docker Images for Java Applications

To minimize the footprint of your Docker images running GraalVM-based applications, read this blog:

https://aws.amazon.com/blogs/opensource/using-graalvm-build-minimal-docker-images-java-applications/

Eclipse Vert.x

Homepage for the Vert.x project

https://vertx.io/

Architecture resources

AWS Well-Architected Framework

The official best practices for architecting in the AWS Cloud

https://aws.amazon.com/architecture/well-architected/

AWS Well-Architected Labs

Hands-on labs to help you learn, measure, and build using architectural best practices

https://wellarchitectedlabs.com/

AWS Architecture Center

Official AWS repository for all architecture resources

https://aws.amazon.com/architecture/

AWS Solutions Library

Vetted reference implementations and well-architected patterns

https://aws.amazon.com/solutions/

Thank you!



AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.



Please complete the session survey

AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.