亚马逊云科技

Serverless Design Principles

a guide to effective architectural choices

Luca Mezzalira

Principal Serverless Specialist Solutions Architect
亚马逊云科技

Luca Mezzalira

Principal Serverless Specialist at AWS

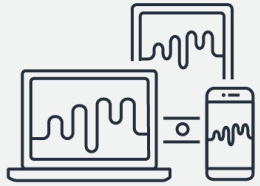
International Speaker

Author

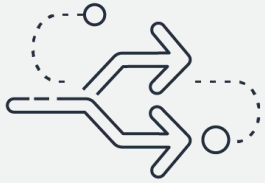
Distributed Systems
are LIVING systems

Distributed systems goals

Organization scalability

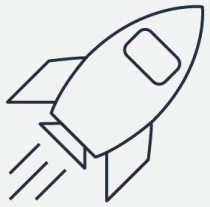


Business Agility



Faster feedback loop

Reduce external dependencies



Reduce blast radius

How does serverless fit in distributed systems?

Serverless is a STRATEGY

n

Serverless Portfolio*

APPLICATION PRIMITIVES – COMPUTE AND DATASTORES


Amazon S3


AWS Lambda


AWS Fargate


Amazon DynamoDB


Amazon Aurora Serverless


Amazon Kinesis

APPLICATION INTEGRATION


Amazon SNS


Amazon API Gateway


AWS Step Functions


Amazon EventBridge


Amazon SQS

Developer Tools


AWS CloudFormation


AWS Cloud9


AWS CodePipeline


AWS Config


AWS CloudTrail


Amazon CloudWatch


AWS X-Ray


AWS Serverless Application Repository

SECURITY AND ADMINISTRATION


AWS IAM


AWS SSO


Amazon GuardDuty


Amazon Inspector


Amazon VPC


AWS WAF


AWS Shield

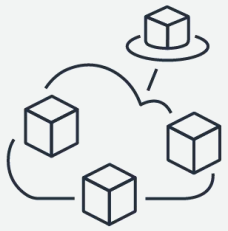

Amazon Cognito

Where can serverless help?

Focus on business value

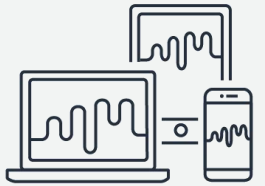
Automatic scaling

Security and isolation by design



Managed infrastructure

Lower Total Cost of Ownership (TCO)



Business agility

How to design Serverless applications

CONNECTED DIMENSIONS in distributed systems

Organization

Culture

System Architecture

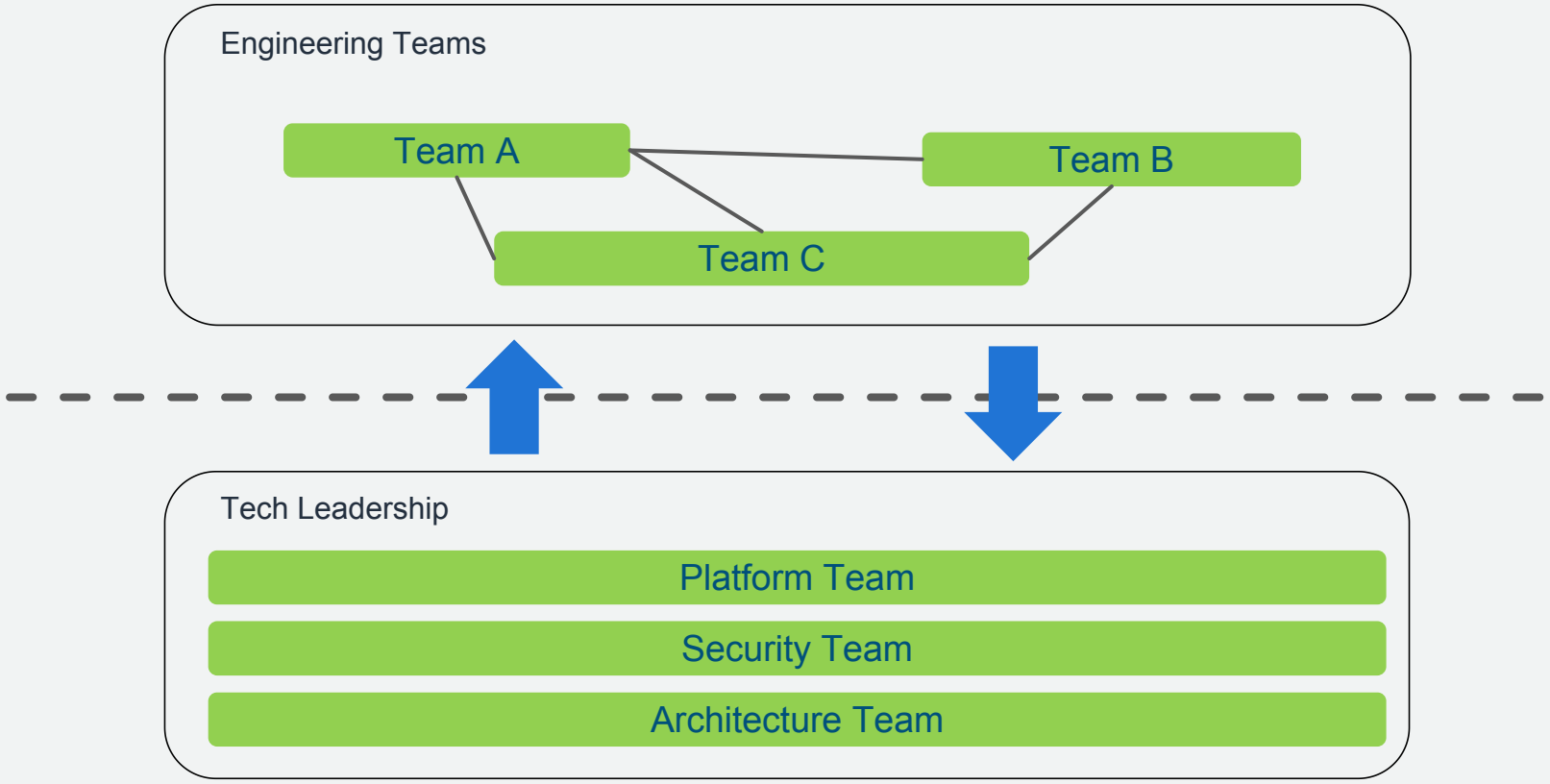
Organization

> “Any organization that designs a system will produce a design whose structure is a copy of the organization's communication structure.”

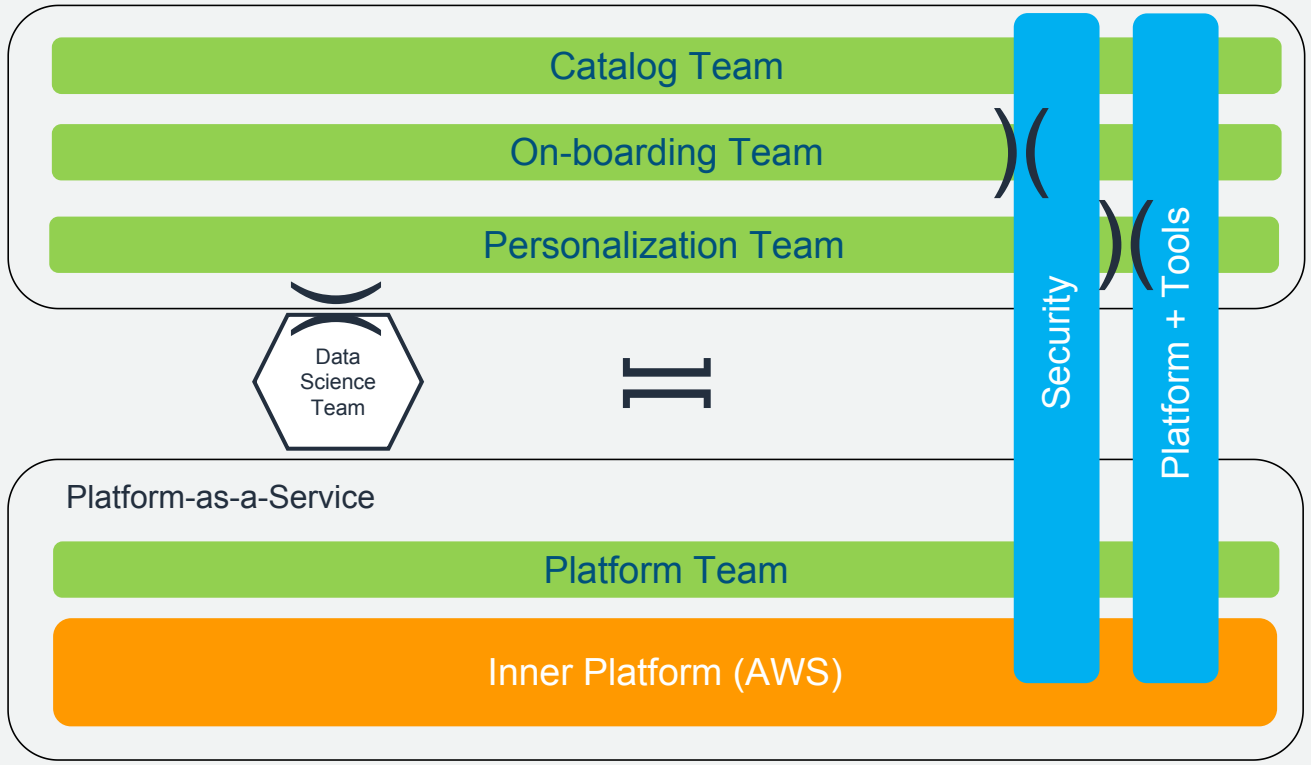
Melvin E. Conway

1967

Centralized mindset



Decentralize mindset



Key:

Stream-aligned team – A team with a business-aligned objective

Enabling Team - An Enabling team helps a stream-aligned team to overcome obstacles.

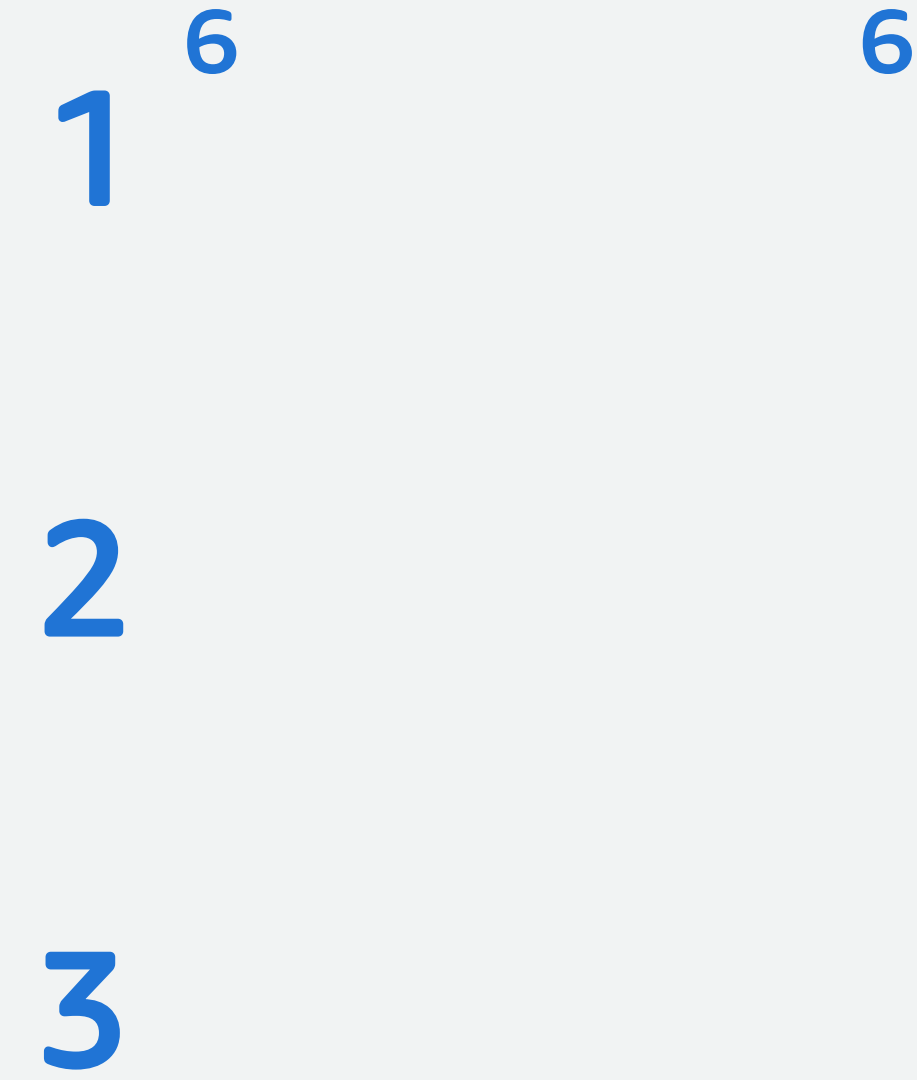
Complicated Subsystem Team – A team with specialist skills that facilitate acute functionality

Facilitating

Federated Service (i.e. X-as-a-Service)

Serverless is an extension of your enablement teams

Enablement teams focus with Serverless



Culture

Decentralize & Empower

Developing a culture of serverless-first

Form CCOE

Form Cloud Center of Excellence.

1

Quick Wins

Deliver strategic “light house” modernized workloads

2

Leadership Support

Establish clear vision and support from leadership

3

Best Practices

Build reusable patterns, reference architecture, and shared services

4

Evangelize

Community Building and Enablement

5

Reorganize

Decentralize CCOE function and federate across the organization

6

System Architecture

Evolutionary Architecture

Architecture begins with

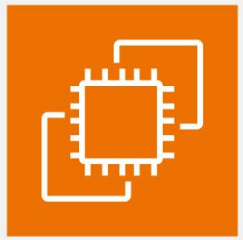
1 3/8

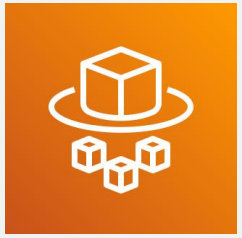
2

3

fk

A spectrum of compute for different needs



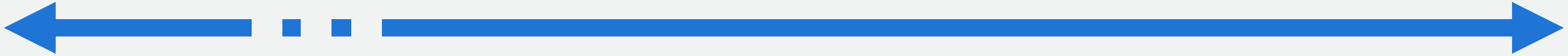


6 5

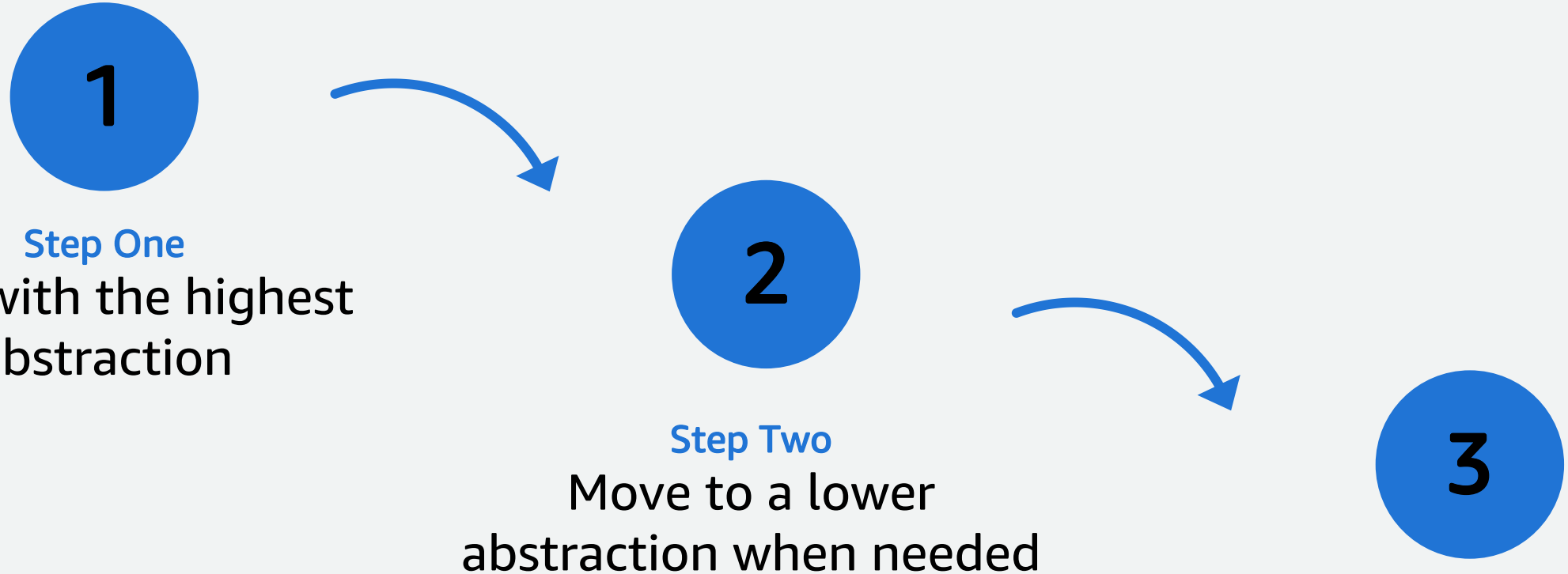
fl x ⅐

fl x n

fl x



Selecting services that fit your strategy



Step One
Start with the highest abstraction

Step Two
Move to a lower abstraction when needed

Step Three
Iterate and evolve

Design Principles

Modularity

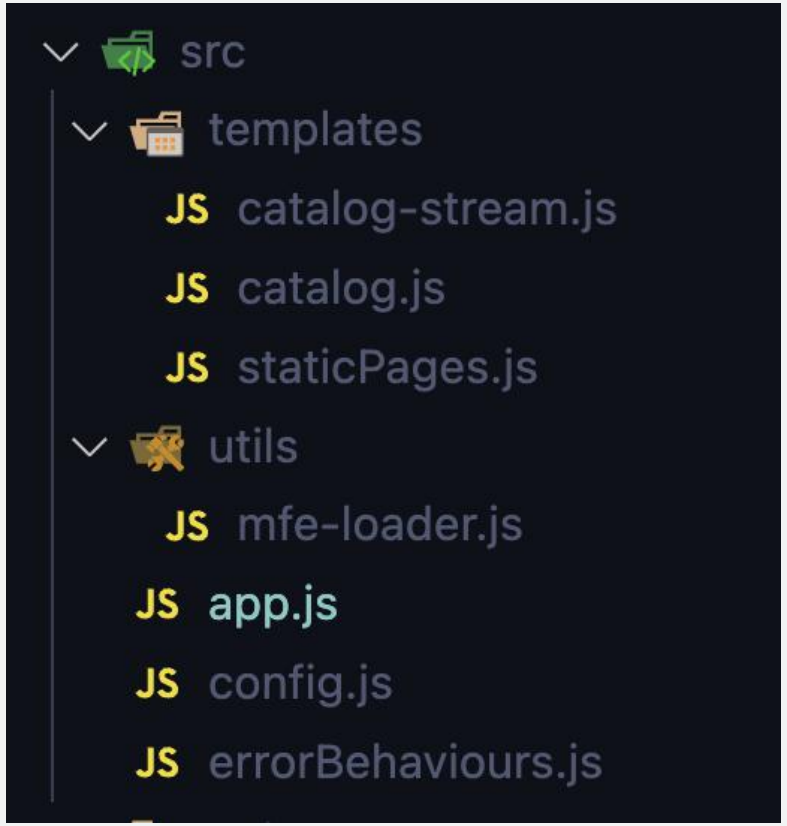
“Modularity: the quality of consisting of separate parts that, when combined, form a complete whole.”

Cambridge Dictionary

"A system lacks modularity when a tweak to one of its components affects the functioning of others."

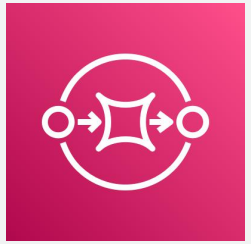
Cambridge Dictionary

Modularity using code



- Strong encapsulation
- Large usage of design patterns
- Decouple business logic from environment
- Developers discipline

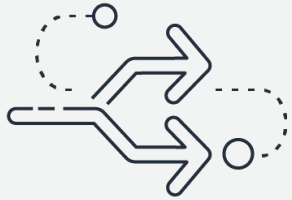
Modularity using infrastructure



- More options to express your intents
- Configuration over code
- Many common built-in behaviors
- More control on what to develop

Architecture and patterns enabled by Serverless

Microservices



Event-Driven Architectures

Data Architectures

Integration Patterns

How to design a workload using Serverless

Business requirements

- Gift code service for an e-commerce
- Gift codes can be generated by the system or 3rd party companies
- For every gift code consumption we need to
  - Notify the customer support team
  - Update the user’s account history
  - If the gift code was issued by 3rd party company notify them

Workload Characteristics

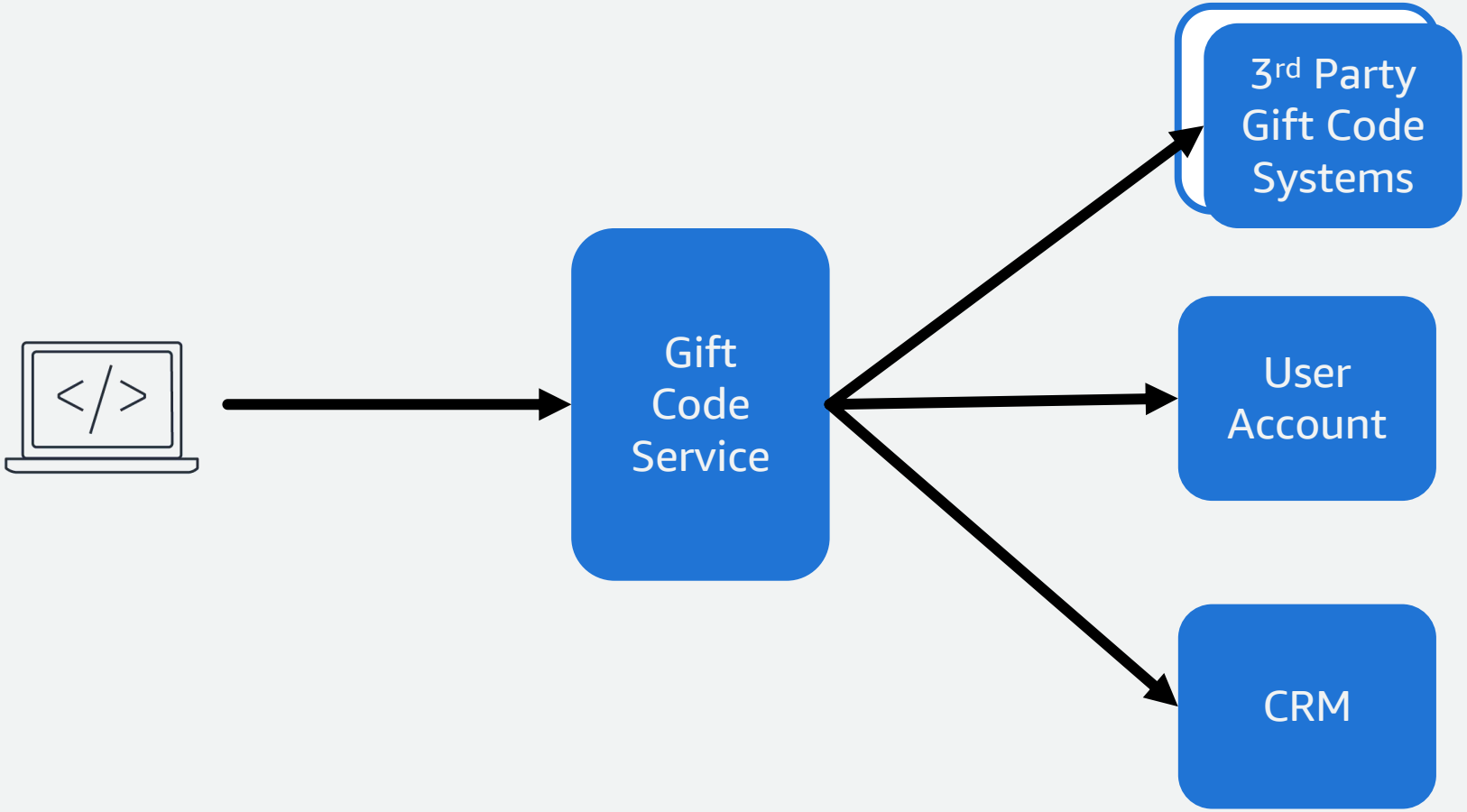
- 99.99% availability on critical path
- 99.9% availability on the rest of the system

- Events to communicate across bounded context

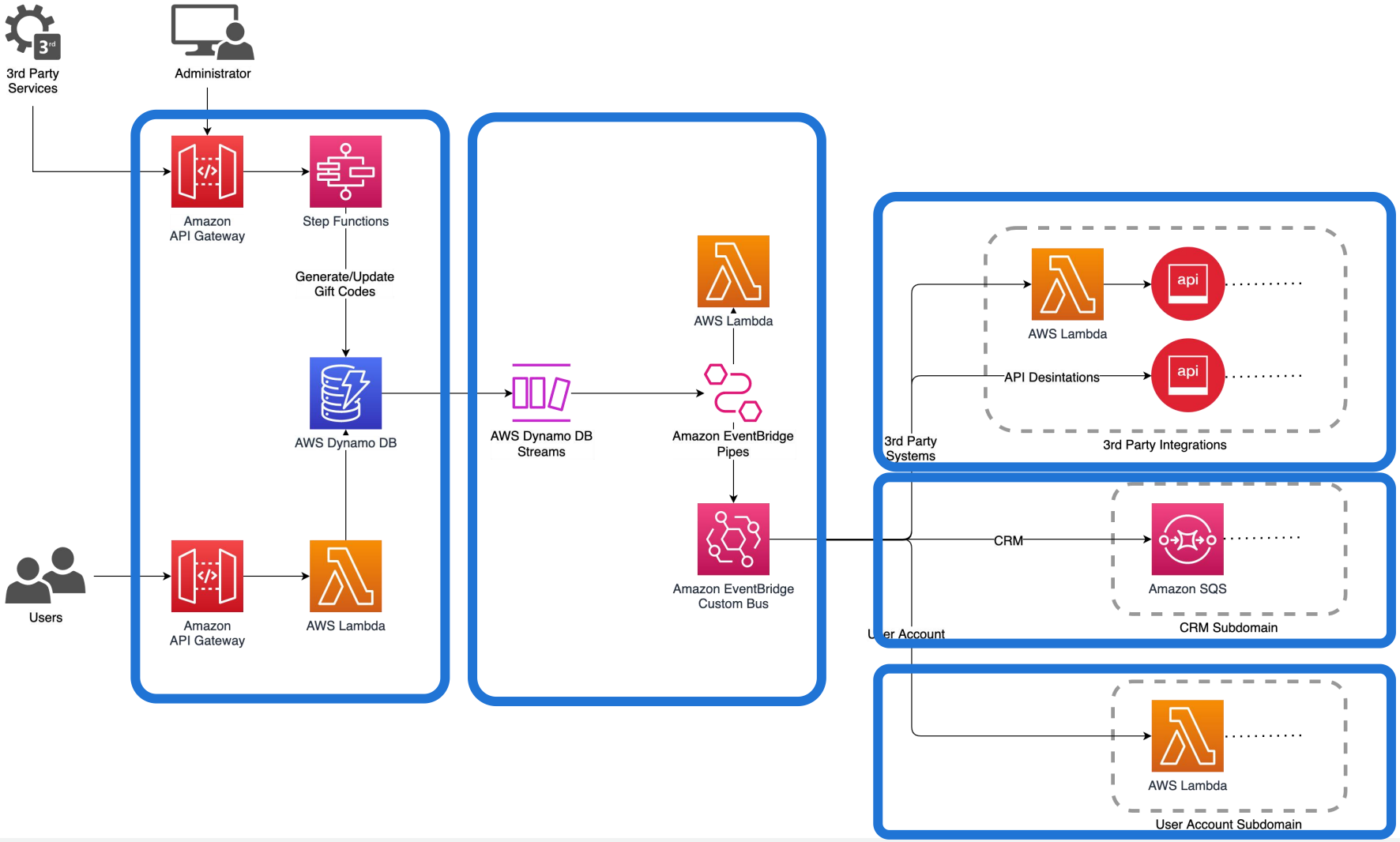
- Under 1 second response time for the user facing APIs
- Scale to up 3000 TPS with 50% headroom

…

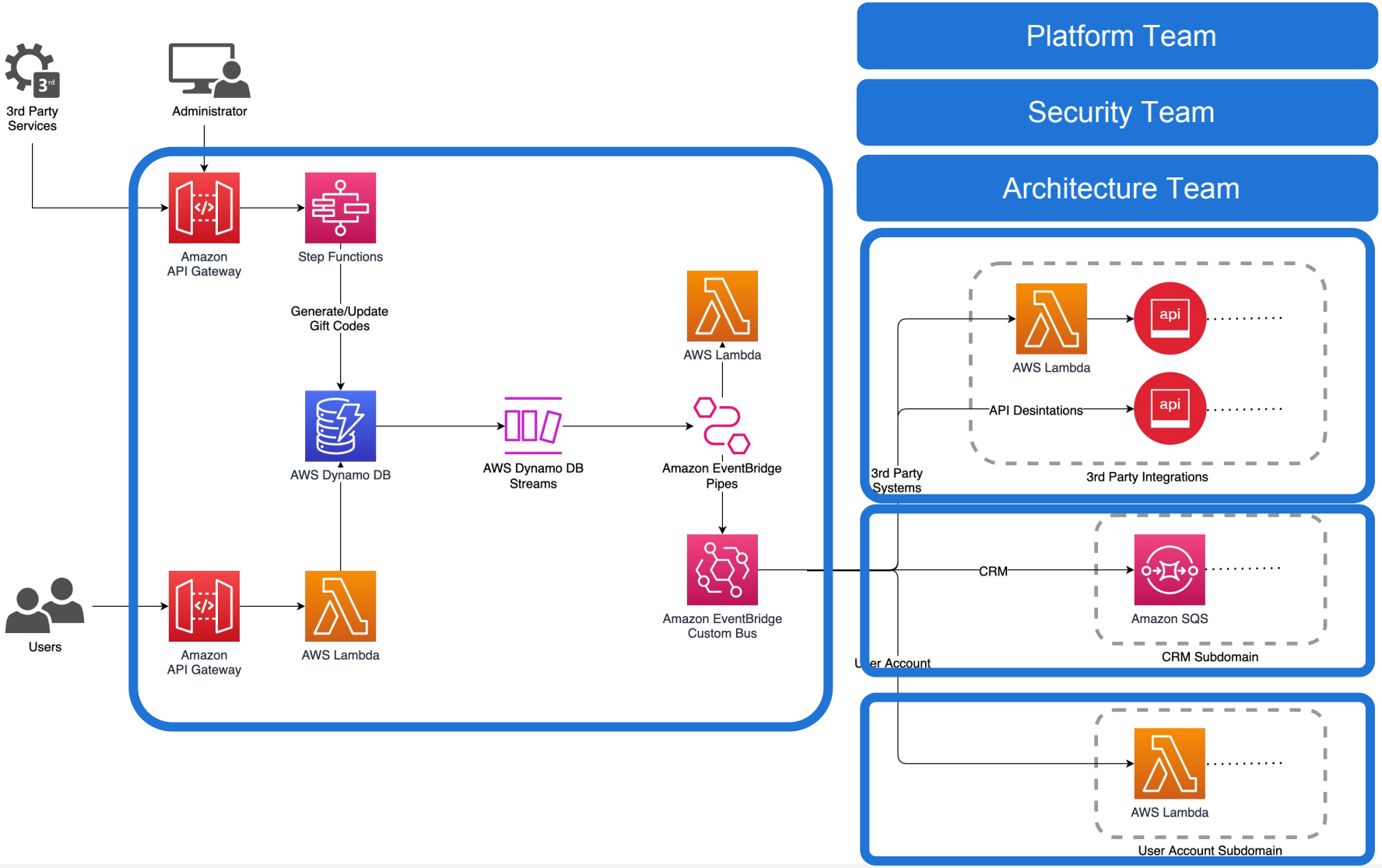
High-level architecture



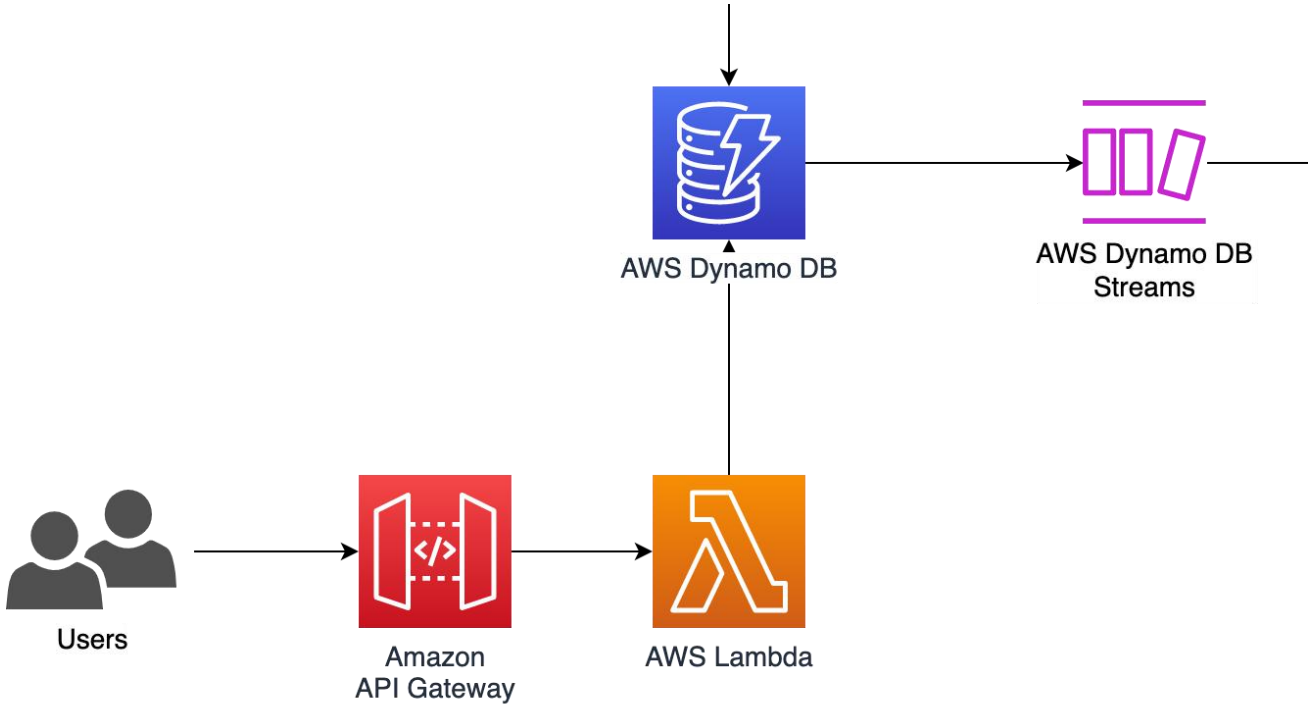
Architectural characteristics



Team topology

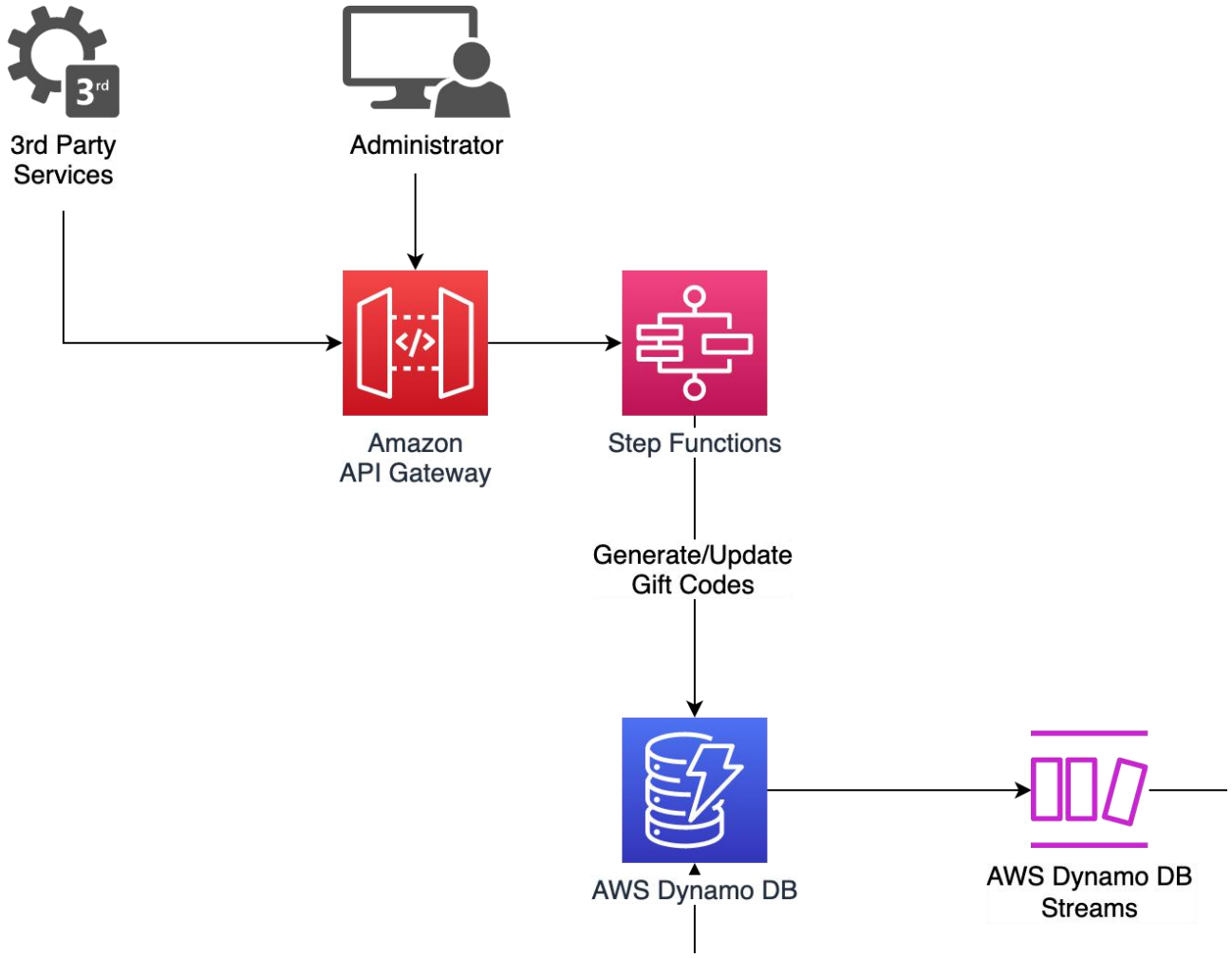


Serverless implementation



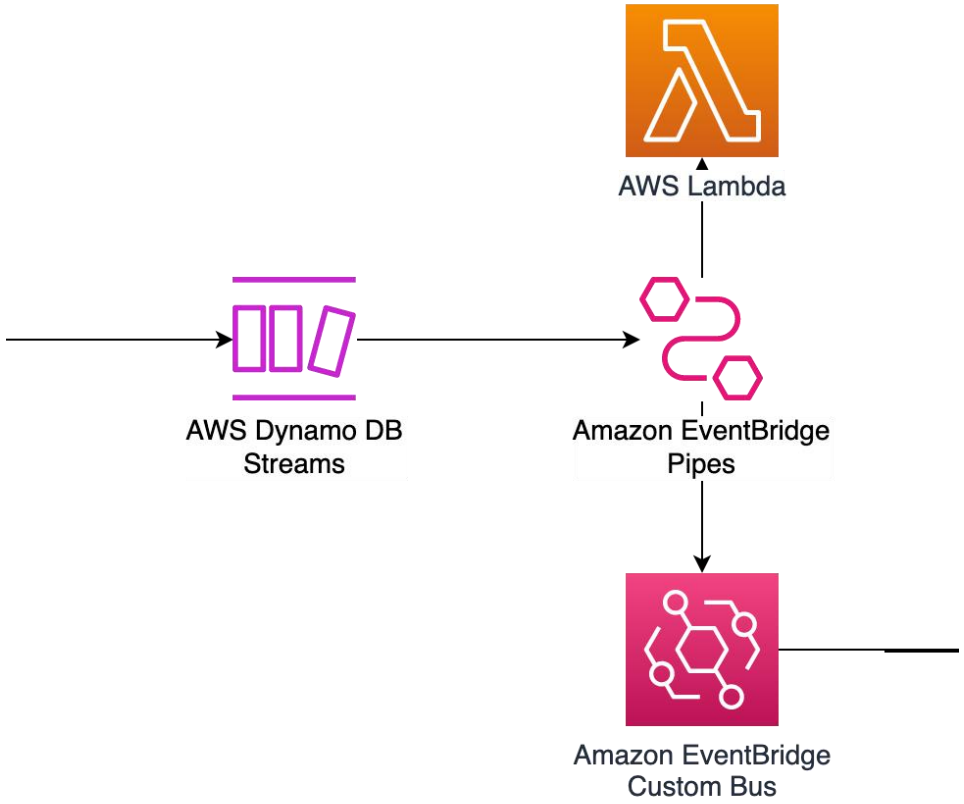
- Synchronous API
- Quick acknowledgment for the users
- Only part that needs to scale based on users traffic
- DynamoDB Streams becomes the glue with the asynchronous part of the system

Serverless implementation



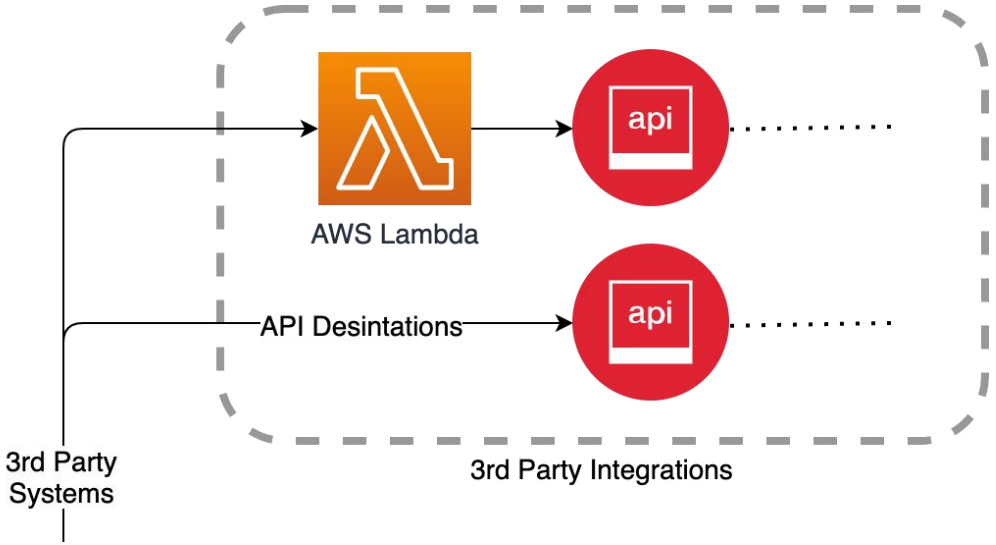
- Synchronous API
- Step Functions orchestrates different services to generate or update gift codes
- API Gateway helps to mitigate eventual traffic spikes from 3rd party services

Serverless implementation



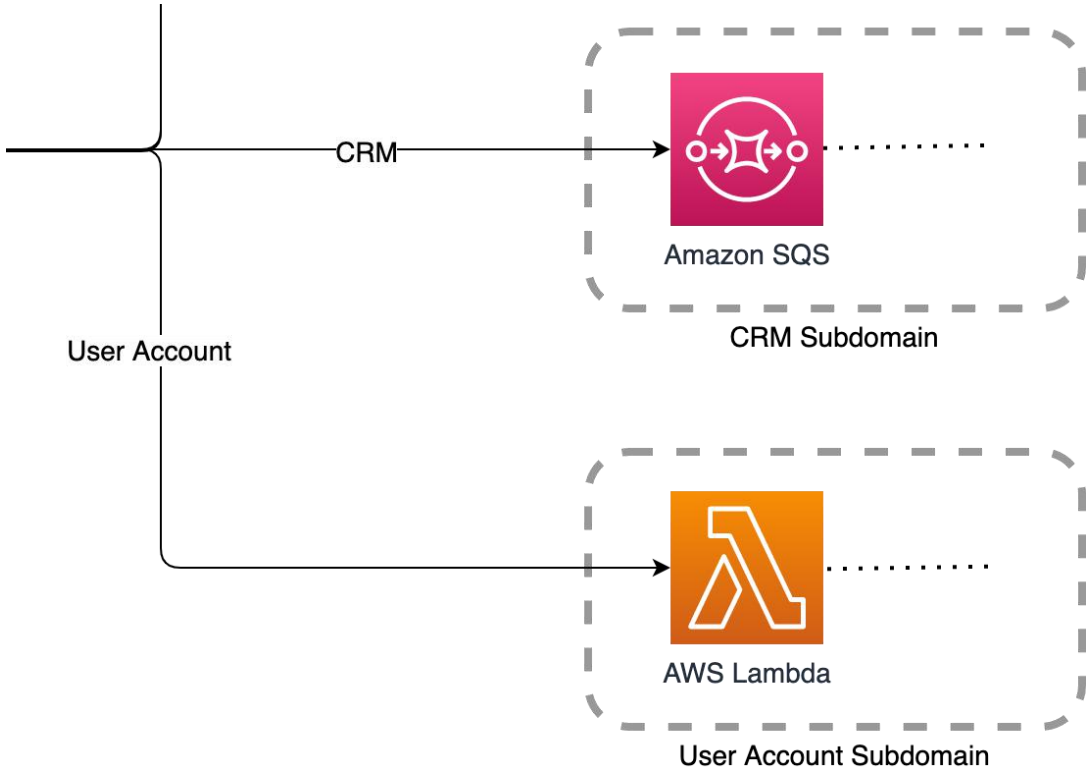
- Events allow the decoupling of producers and consumers
- DynamoDB Streams notifies every change in the DynamoDB table
- EventBridge Pipes enrich the information received for downstream services
- EventBridge is the message broker

Serverless implementation



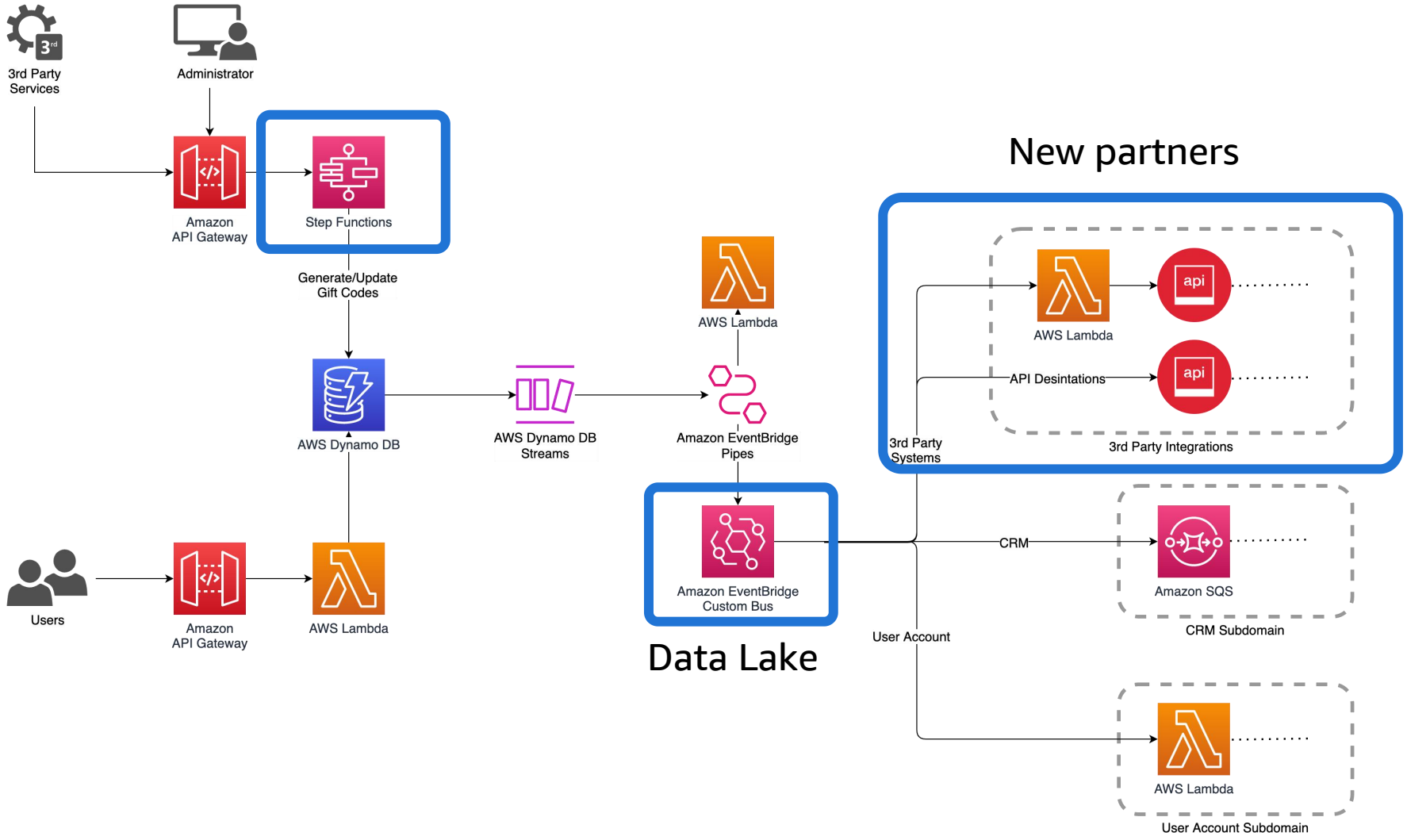
- Some 3rd party systems accepts an API calls in the format defined by your system
- Some others require to translate from JSON to XML or any other format
- More architectural patterns are also applicable

Serverless implementation



- CRM has API limits so a queue helps us to ease the traffic
- User account uses a Lambda function to manipulate the event and then integrate inside their bounded context

Ready for the future…



Express your architecture characteristics and business requirements into infrastructure focusing on YOUR BUSINESS goals

谢谢

Luca Mezzalira

lmezza@amazon.com

亚马逊云科技