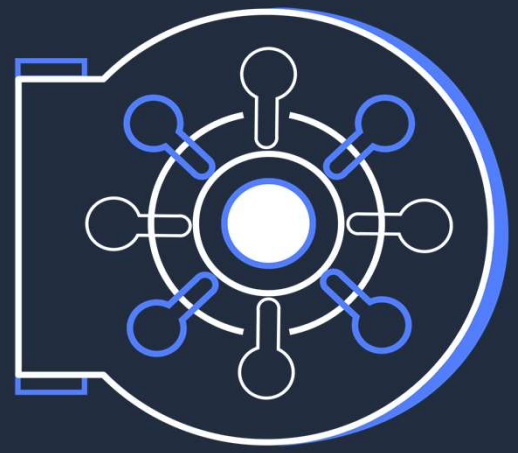
AWS TECHSHIFT

EMBARK



Decoupling Application Logic

AWS TECHSHIFT What is a decoupled application?

A decoupled application architecture allows each component to perform its tasks independently, it allows components to remain completely autonomous and unaware of each other.

AWS TECHSHIFT Decoupling benefits

Smaller blast radius

Faster Deployments

Migration path

More focused teams

More testable code

Easier maintenance

What is a microservice?

"…an approach to developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API."

Martin Fowler

What is a microservice?

Microservices, also known as the microservice architecture, is an architectural style that structures an application as a collection of services that are:

- Highly maintainable and testable
- Loosely coupled
- Independently deployable, without effecting other services
- Organized around business capabilities

AWS TECHSHIFT More benefits from Microservices

Flexible Scaling

Technical Freedom

Agility

Decomposition

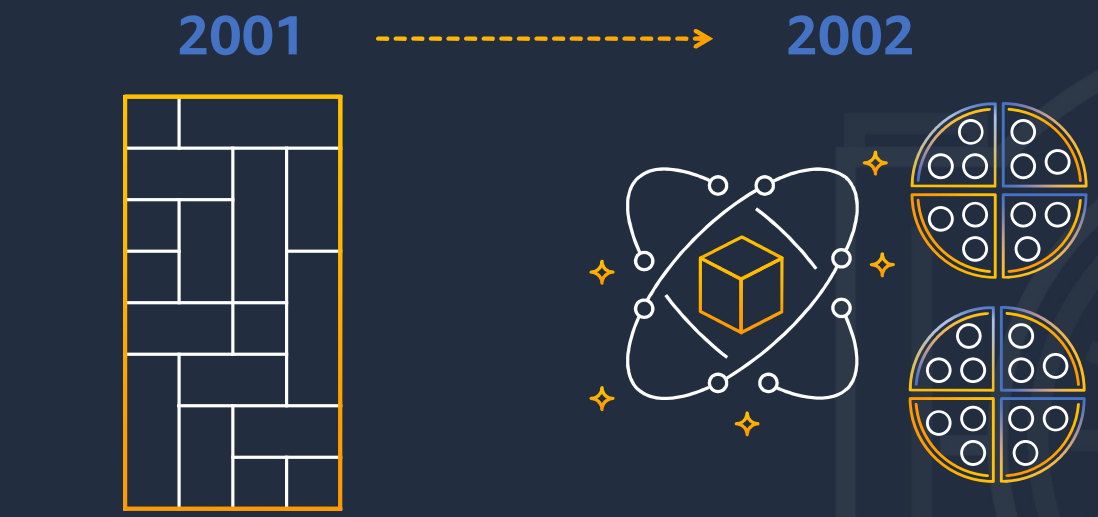
Communication

Observation

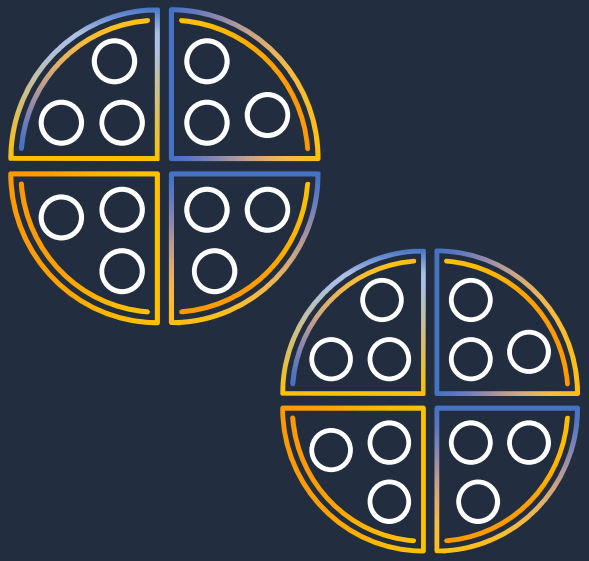
Security

Development transformation at Amazon: 2001–2002

Lesson learned: decompose for agility



AWS TECHSHIFT Amazon: 2001–2002



Full ownership

Full accountability

“DevOps”

Focused innovation

Domain Model

The domain model is derived primarily from the nouns of the user stories, and the system operations are derived mostly from the verbs.

Business Capabilities

Define services corresponding to business capabilities. A business capability is something that a business does in order to generate value.

AWS TECHSHIFT Decomposition Patterns

Domain Model

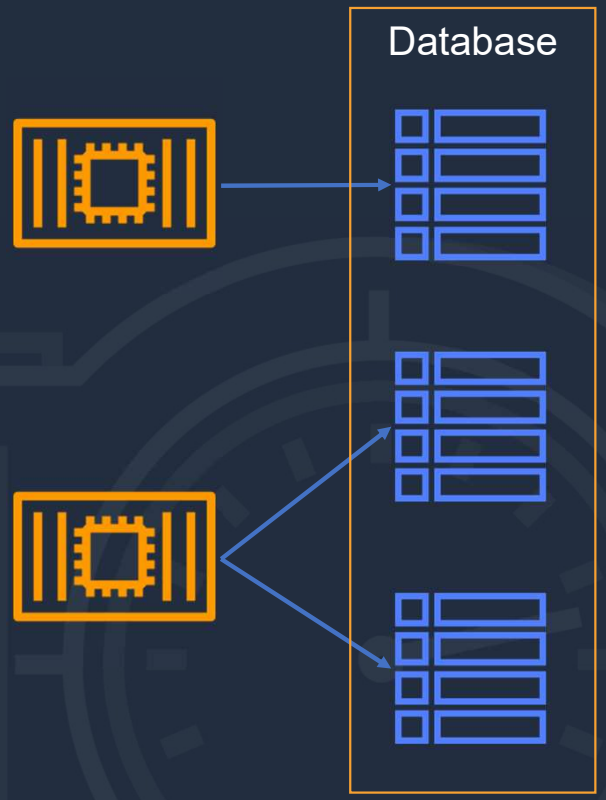
- Transactions
- Orders
- Products
- Departments

Business Capabilities

- Shopping Cart
- Catalogue Management
- Finance Reporting
- Accounts Reconciliation

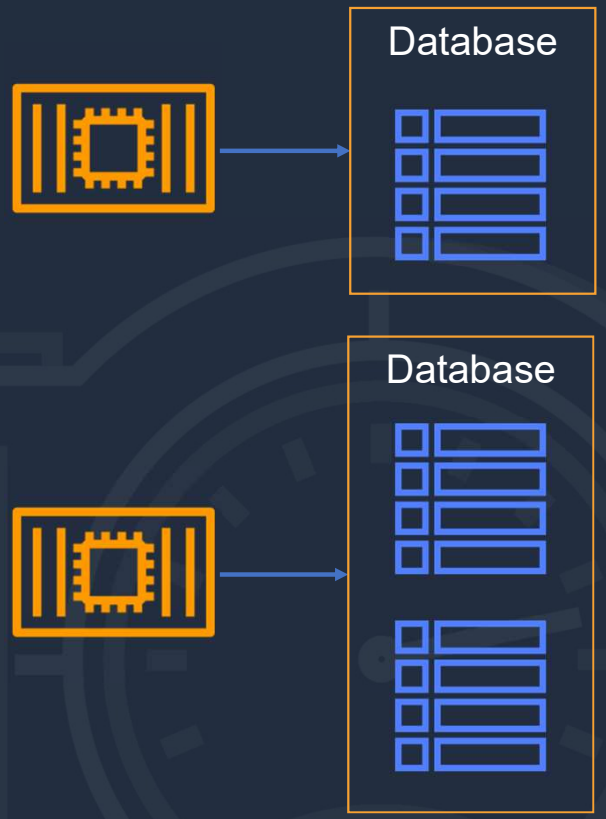
AWS TECHSHIFT Data management

- Private-tables-per-service – each service owns a set of tables that must only be accessed by that service
- Database-per-service – each service has a database schema that's private to that service
- Database-server-per-service – each service has it's own database server.



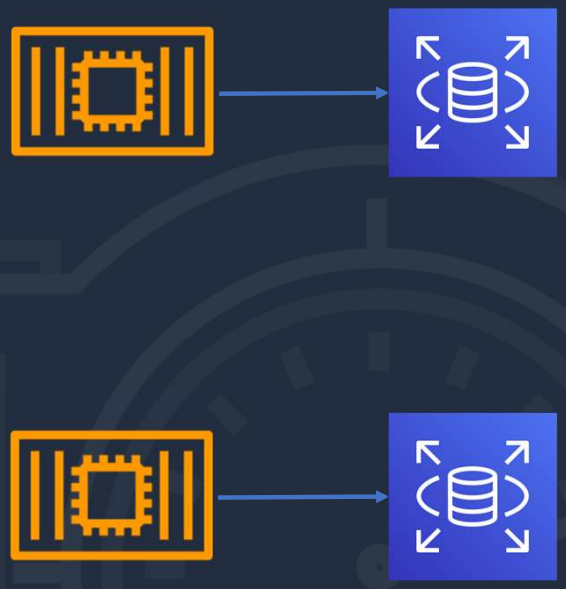
AWS TECHSHIFT Data management

- Private-tables-per-service – each service owns a set of tables that must only be accessed by that service
- Database-per-service – each service has a database schema that's private to that service
- Database-server-per-service – each service has it's own database server.



AWS TECHSHIFT Data management

- Private-tables-per-service – each service owns a set of tables that must only be accessed by that service
- Database-per-service – each service has a database schema that's private to that service
- Database-server-per-service – each service has it's own database server.



AWS TECHSHIFT Communication

Synchronous

RESTful API

HTTP/S

Remote Procedure Calls

Asynchronous

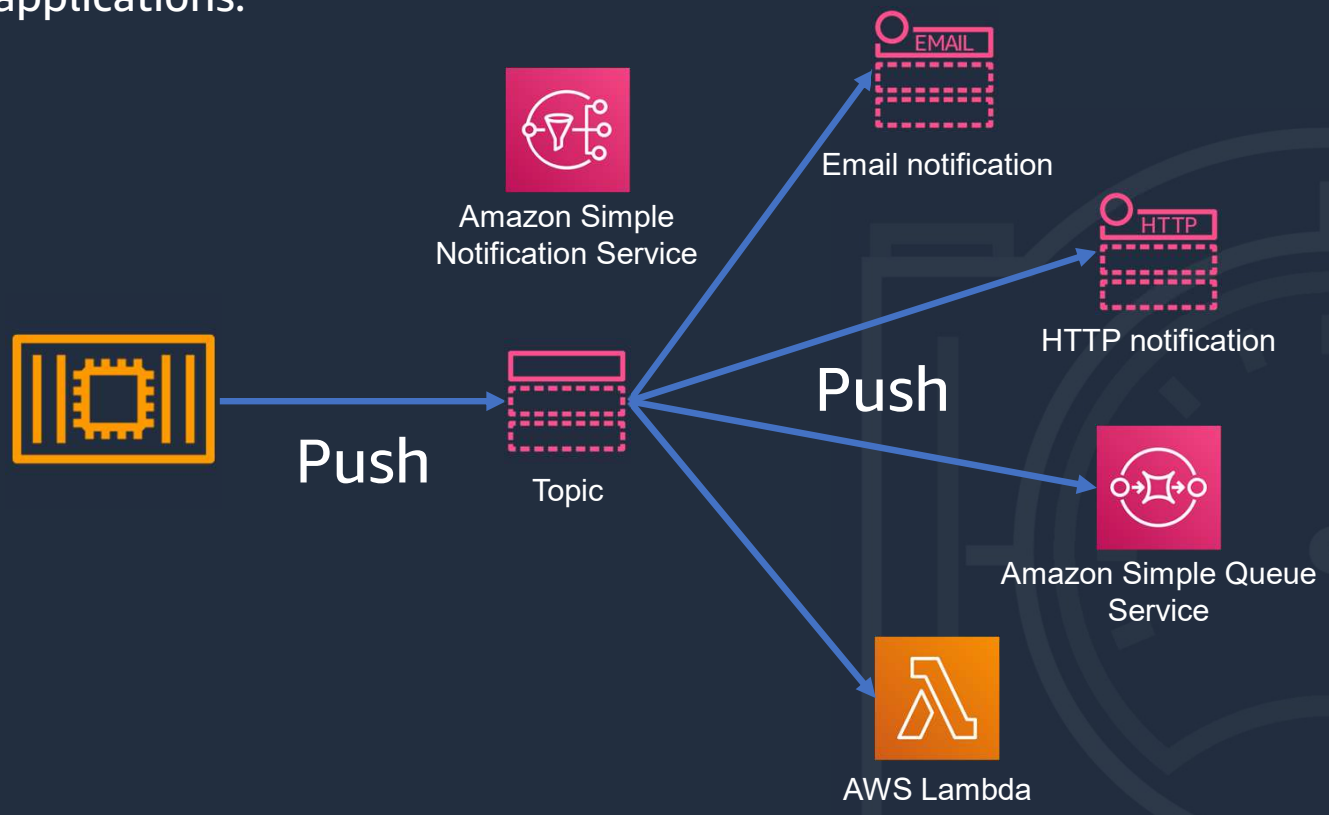
Request/response

Notifications

Publish/subscribe

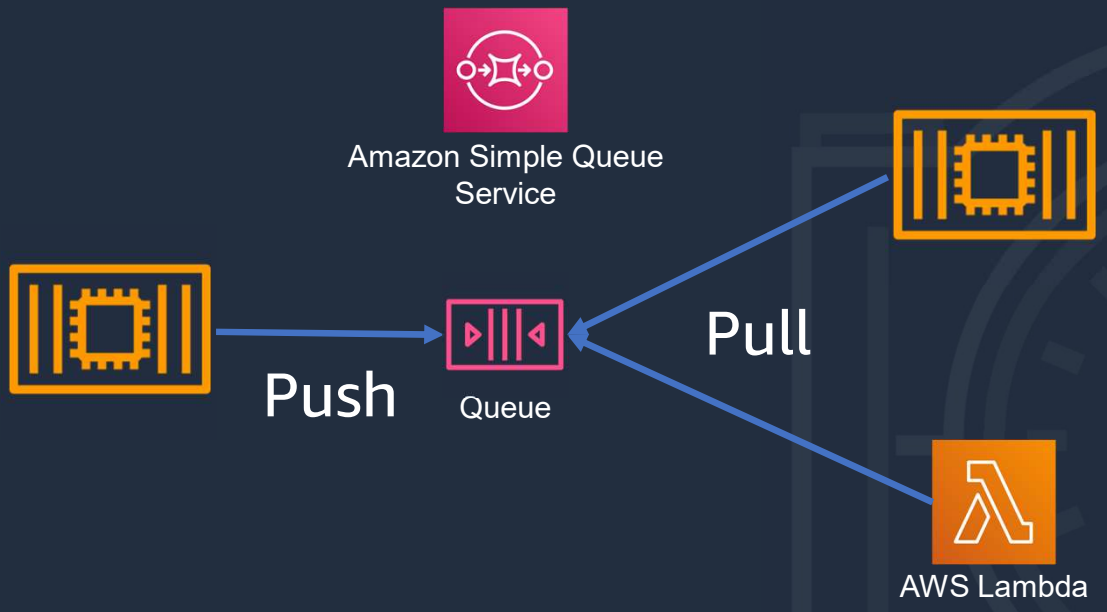
AWS TECHSHIFT Notifications with SNS

Amazon Simple Notification Service (SNS) is a highly available, durable, secure, fully managed pub/sub messaging service that enables you to decouple microservices, distributed systems, and serverless applications.



AWS TECHSHIFT Messaging with SQS

Amazon Simple Queue Service (SQS) is a fully managed message queuing service that enables you to decouple and scale microservices, distributed systems, and serverless applications.



AWS TECHSHIFT API Gateway

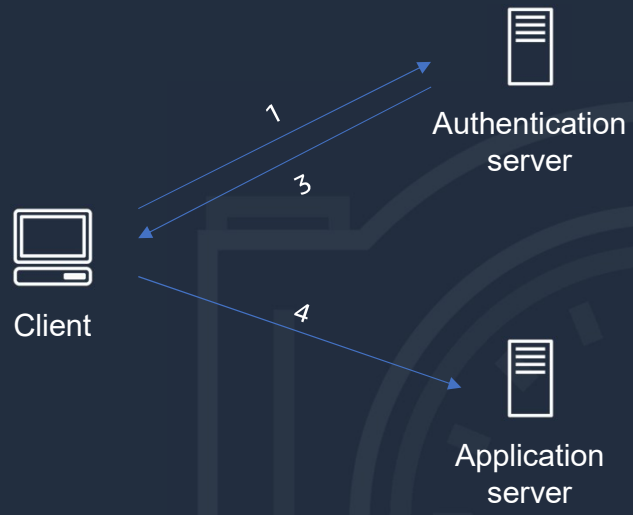
Amazon API Gateway is a fully managed service that makes it easy for developers to create, publish, maintain, monitor, and secure APIs at any scale

- Efficient API Development
- Easy Monitoring
- Performance at Any Scale
- Cost Savings at Scale
- Flexible Security Controls
- Restful API Endpoints
- Serverless APIs
- WebSocket APIs

Token Authentication

Authentication flow

1. User's browser redirected to Authentication Server
2. Authentication server validates users credentials
3. User is issued an authentication token
4. User requests resource from the Application Server using the authentication token



JWT Authentication Tokens

JSON Web Token (JWT) is an open standard (RFC 7519) that defines a compact and self-contained way for securely transmitting information between parties as a JSON object.

This information can be verified and trusted because it is digitally signed.

AWS TECHSHIFT JWT Authentication Tokens

Encoded JWT Token

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ
zdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4
gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJ
SMeKKF2QT4fwpMeJf36POk6yJV_adQssw5c
```

Decoded Header

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

Signature

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  secret
)
```

Decoded Payload

```
{
  "sub": "1234567890",
  "name": "John Doe",
  "iat": 1516239022
}
```

AWS TECHSHIFT Amazon Cognito

Spend your time creating great apps.

Let Amazon Cognito handle authentication.

Secure and scalable user directory

Standards-based authentication

Easy integration with your app

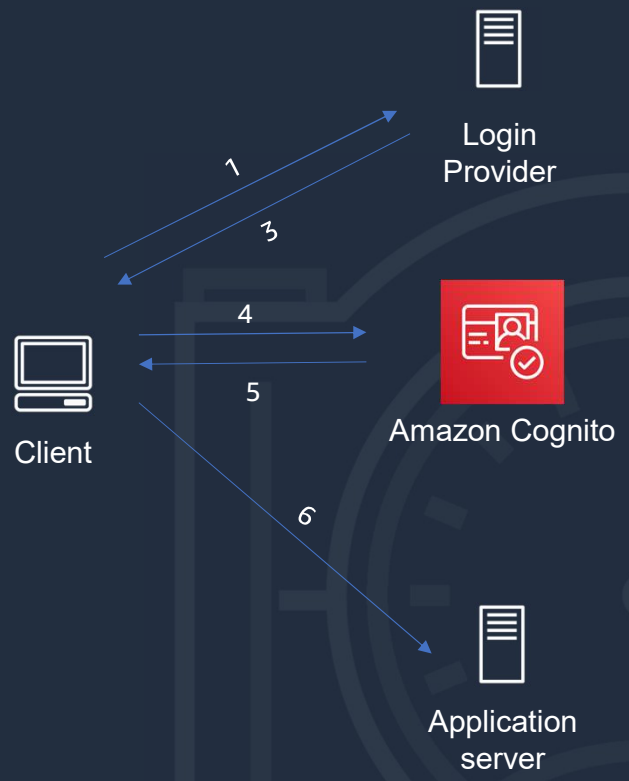
Social and enterprise identity federation

Access control for AWS resources

Security for your apps and users

User Pools Authentication

1. User's browser redirected to Login Provider
2. Login Provider validates users credentials
3. User is issued an ID token
4. Exchange ID token for Cognito token
5. Get Cognito token form Amazon Cognito
6. User Cognito token to request resource from the Application server



Identity Pools Authentication

1. User's browser redirected to Login Provider
2. Login Provider validates users credentials
3. User is issued an ID token
4. Exchange ID token for Cognito token
5. Get Cognito token form Amazon Cognito
6. Exchange Cognito token for temporary AWS access credentials
7. Get temporary AWS access credentials
8. User temporary AWS access credentials to access AWS services

End