# AWS - Infrastruktur mit Java erstellen
# CDK in Action

**Damian Dudziec & Pawel Adaszewski**

**09 November 2023**

# Über uns

**Pawel Adaszewski**
**Senior Software Consultant @ ZEISS Digital Innovation**

AWS:
- Development (CDK)
- Architecture
- IIoT

linkedin.com/in/pawel-Adaszewski

**Damian Dudziec**
**Software Consultant @ ZEISS Digital Innovation**

Enterprise Java
AWS CDK Development

linkedin.com/in/damian-dudziec

# Ziel

**Infrastructure as Code**

- Was ist das?
- Wofür wird es benötigt?
- Was sind die Vorteile?
- Kurzvorstellung ausgewählter Tools

**AWS CDK**

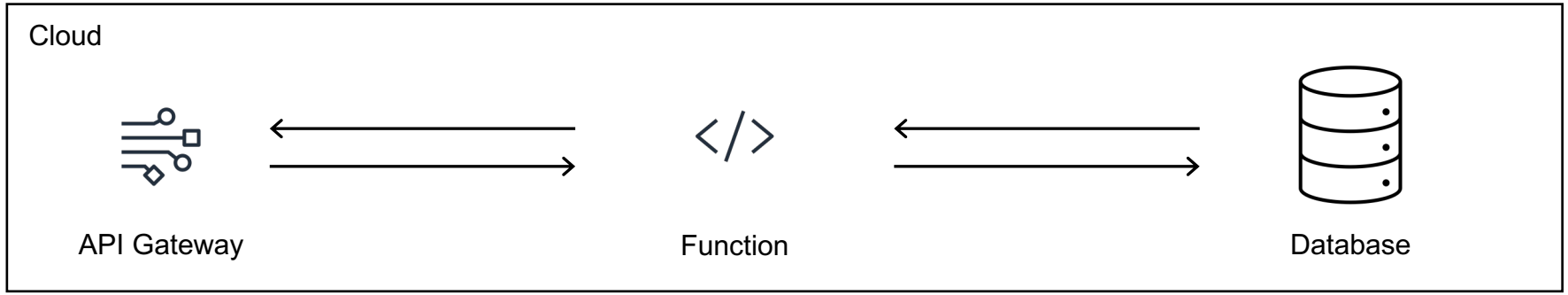
- Geschichte
- Einführung
- Konzepte
- Vorteile

**Hands on: Coding Demo**

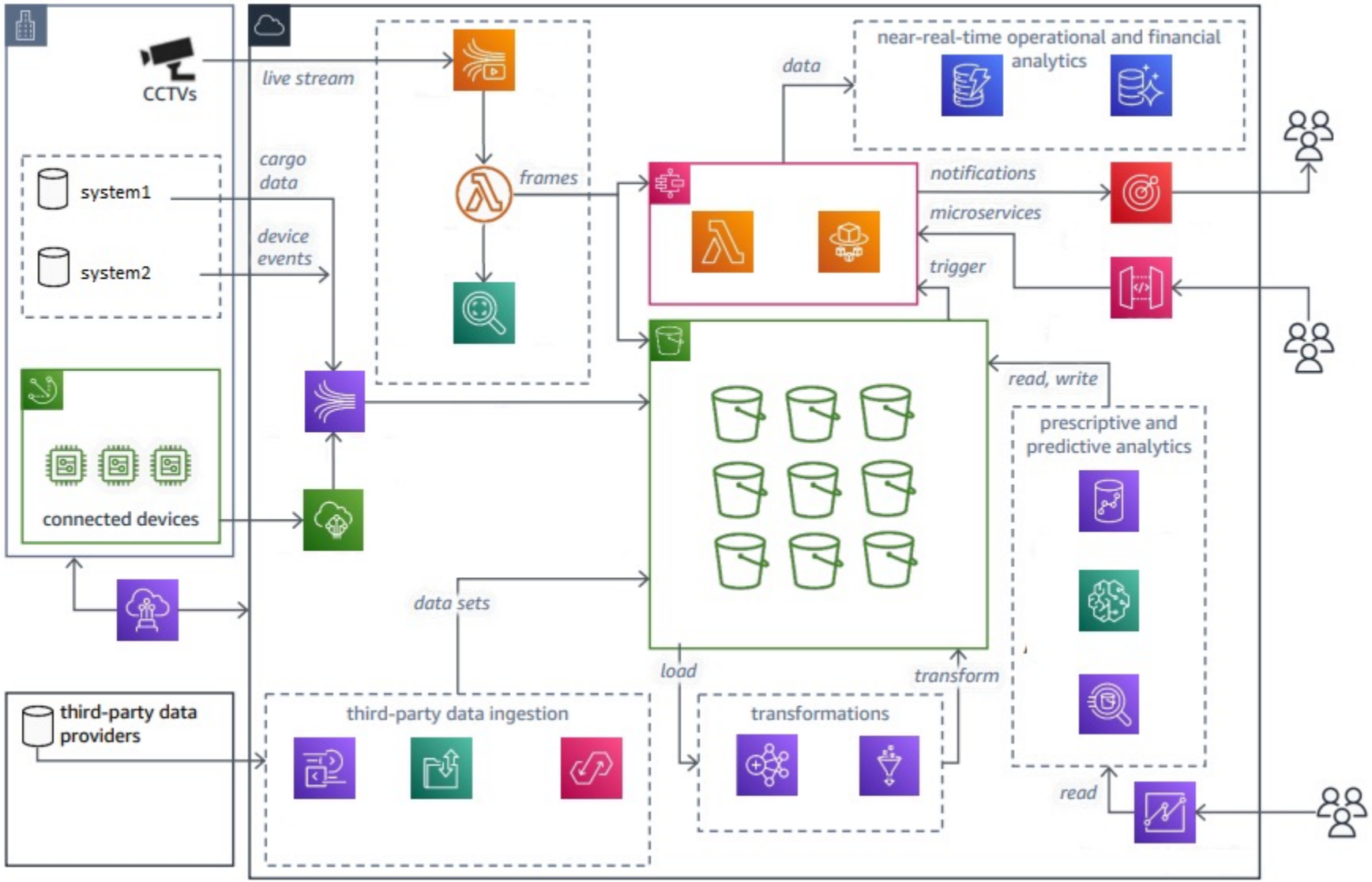
- Architektur: Überblick
- Projekterstellung
- Modularisierung
- Provisionierungsprozess

**Inspiration durch zwei interessante Perspektiven für den Aufbau eigener Infrastrukturen auf AWS mittels IaC**

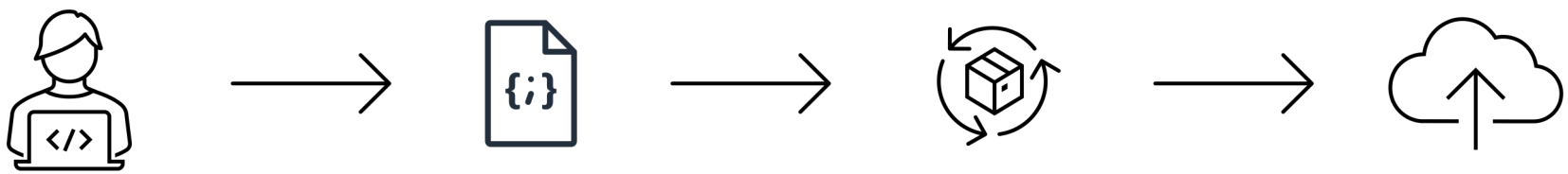
# Problemstellung



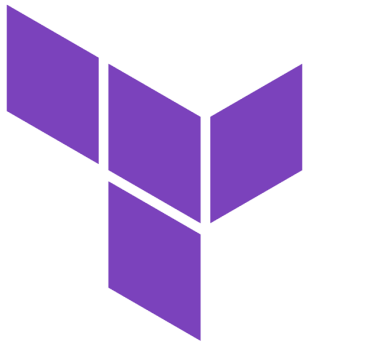
# AWS Infrastructure – was wenn…?

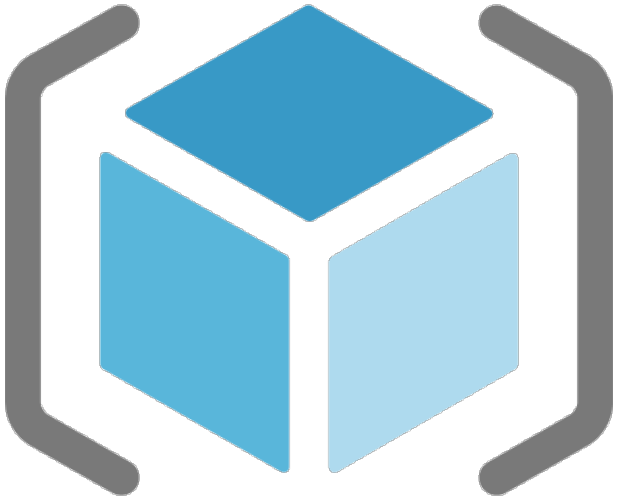


# Infrastructure as Code



# Kurzvorstellung ausgewählter Tools
# Terraform, Kubernetes, ARM Templates



# AWS CDK

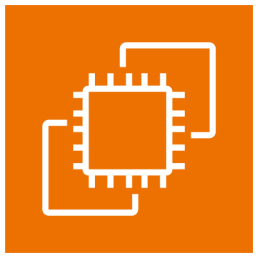
aws

Cloud Development Kit

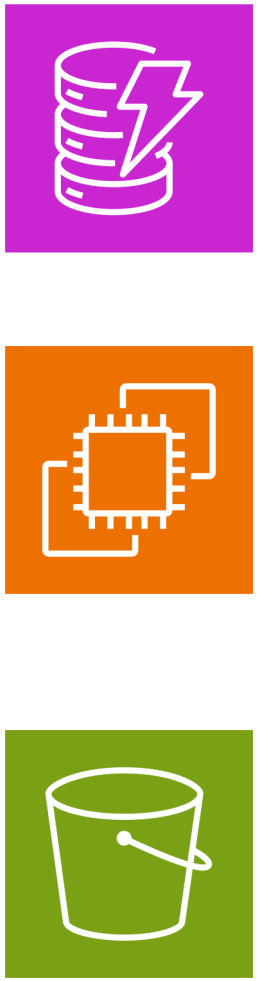
# Amazon Web Services

# Amazon Web Services

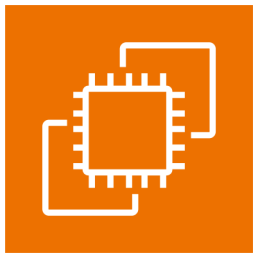
# Amazon Web Services



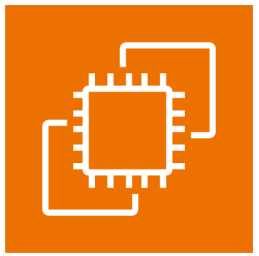
# Amazon Web Services

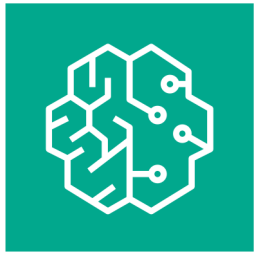


# Amazon Web Services

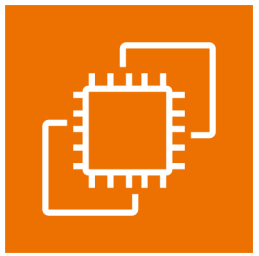


# Amazon Web Services

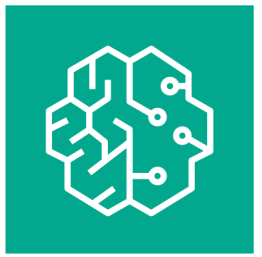




# Amazon Web Services





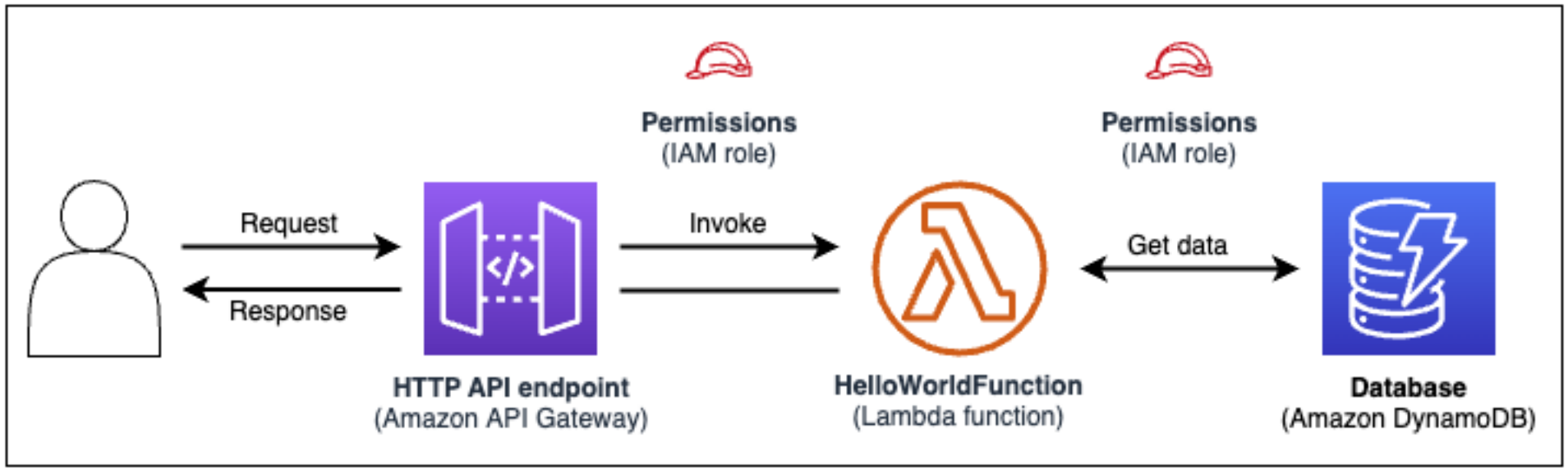

# AWS CDK

aws

Cloud Development Kit

# AWS CloudFormation

# Beispielarchitektur



# Beispielarchitektur: AWS CloudFormation


```
      "SamResourceId": "getAllItemsFunction"
    },
    "Properties": {
      "Code": {
        "S3Bucket": "aws-sam-cli-managed-default-samclisourcebucket-1a4x26zbcdkqr",
        "S3Key": "what-is-app/a6f856abf1b2c4f7488c09b367540b5b"
      },
      "Handler": "src/get-all-items.getAllItemsHandler",
      "Role": {
        "Fn::GetAtt": [
          "getAllItemsFunctionRole",
          "Arn"
        ]
      },
      "Runtime": "nodejs12.x",
      "Tags": [
        {
          "Key": "lambda:createdBy",
          "Value": "SAM"
        }
      ]
    }
  },
  "getAllItemsFunctionRole": {
    "Type": "AWS::IAM::Role",
```

# AWS Serverless Application Model (SAM)

# Beispielarchitektur: AWS SAM

```yaml
AWSTemplateFormatVersion: 2010-09-09
Transform: AWS::Serverless-2016-10-31
Resources:
  getAllItemsFunction:
    Type: AWS::Serverless::Function
    Properties:
      Handler: src/get-all-items.getAllItemsHandler
      Runtime: nodejs12.x
      Events:
        Api:
          Type: HttpApi
          Properties:
            Path: /
            Method: GET
    Connectors:
      MyConn:
        Properties:
        Destination:
          Id: SampleTable
          Permissions:
            - Read
  SampleTable:
    Type: AWS::Serverless::SimpleTable
```

# AWS CDK

aws

Cloud Development Kit

# Beispielarchitektur: AWS CDK

```
Table tableBooksCatalog = new Table(this, "tableBooksCatalog", TableProps.builder().build());

Function listBooks = new Function(this, "listBooksLambda", FunctionProps.builder()
        .runtime(Runtime.NODEJS_18_X)
        .code(Code.fromAsset("lambda"))
        .handler("list.handler")
        .environment(Collections.singletonMap("DYNAMODB_TABLE", tableBooksCatalog.getTableName()))
        .build());

LambdaRestApi booksApi = new LambdaRestApi(this, "booksApi", LambdaRestApiProps.builder()
        .proxy(false)
        .handler(listBooks)
        .build());

Resource resource = booksApi.getRoot().addResource("books");
LambdaIntegration lambdaIntegration = new LambdaIntegration(listBooks);

resource.addMethod("GET", lambdaIntegration);

listBooks.addToRolePolicy(
        new PolicyStatement(PolicyStatementProps.builder()
                .actions(List.of("dynamodb:Scan"))
                .resources(List.of(tableBooksCatalog.getTableArn()))
                .build())
);
```

# AWS CDK – Konstrukte

[redacted]

[redacted]

L1

**CfnBucket =**

# AWS CDK – Konstrukte

L2

**Bucket**

**CfnBucket =**

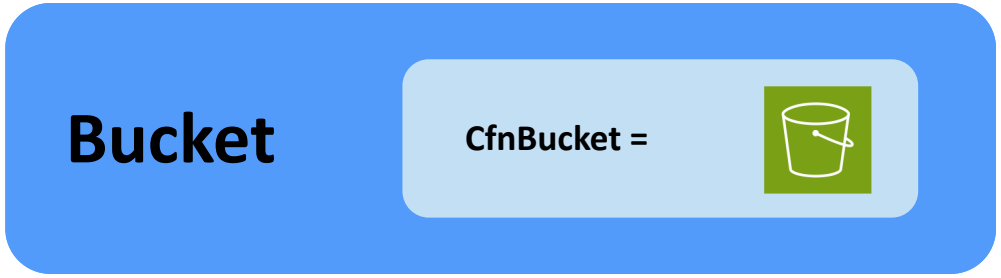
L1

**CfnBucket =**

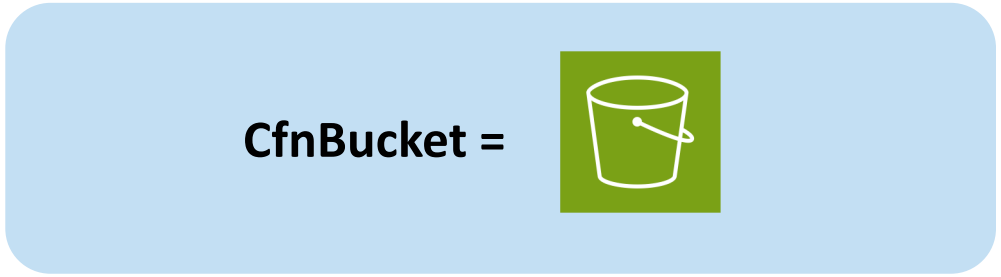
# AWS CDK – Konstrukte

L3

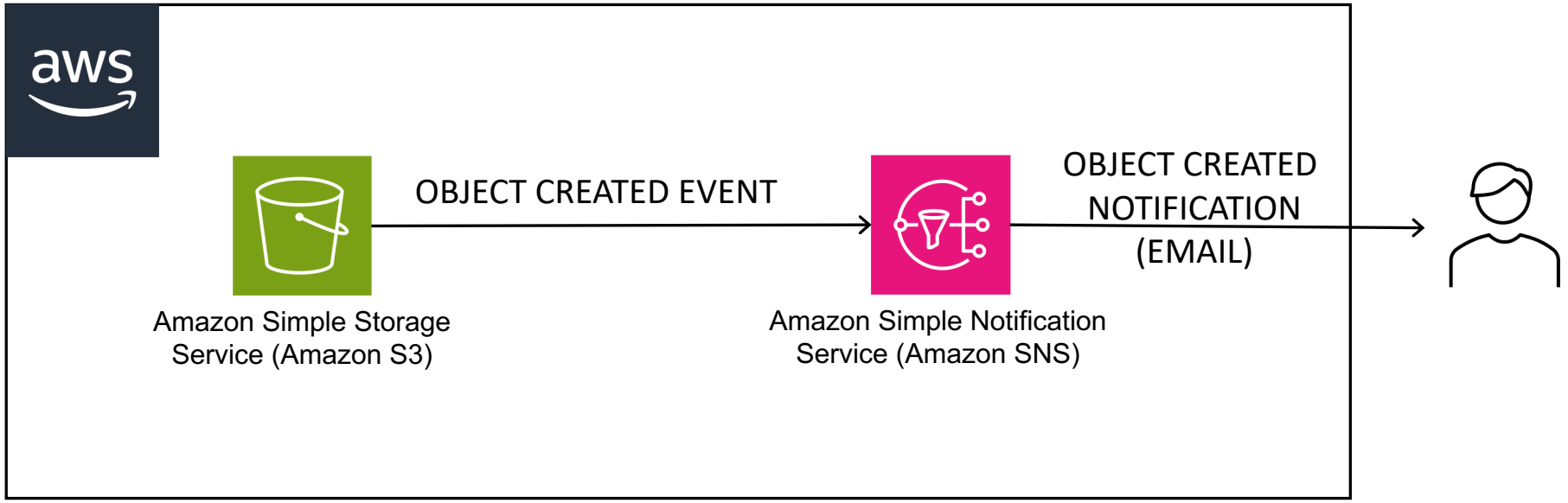
L2



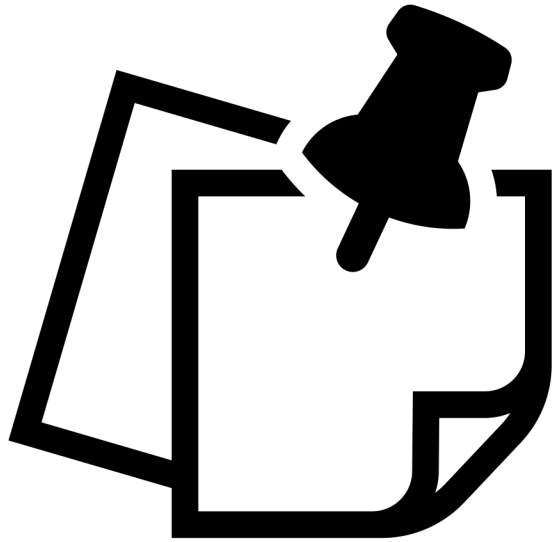

L1



# Live Demo: Architektur



# Zusammenfassung




Seeing beyond