Come sfruttare la potenza dell'ereditarietà di AWS CDK per creare un servizio OCR serverless

Matteo Madeddu @ AWS Community Builder - 22 Giugno 2023

Chi sono

Matteo Madeddu, Platform Engineer

Lavoro principalmente su infrastrutture AWS. Quando proprio non c'è alternativa migliore di me nei paraggi per farlo, scrivo codice.

@made2591 su Github/Twitter
madeddu.xyz blog

Agenda

- Platform engineering
- AWS CDK
- Infrastruttura AWS
- Demo
- Q&A

Platform engineering

Per lavorare come platform engineer,
devi seguire una serie di principi

Platform engineering

Per lavorare come platform engineer,
devi seguire una serie di principi

per dirla meglio, non è “obbligatorio”
...ma seguirli semplifica molto il lavoro

Principio n° 1

”Don't repeat yourself” (DRY) is a principle aimed at reducing repetition of information which is likely to change, replacing it with abstractions that are less likely to change.

Principio n° 1

Io sapevo già

”Don't repeat yourself” (DRY) is a principle aimed at reducing repetition of information which is likely to change, replacing it with abstractions that are less likely to change.

Principio n° 2

“Keep it simple” (stupid) (KISS) is a design principle which states that designs and/or systems should be as simple as possible. Wherever possible, complexity should be avoided in a system—as simplicity guarantees the greatest levels of user acceptance and interaction.

Principio n° 2

capirai che novità

“Keep it simple” (stupid) (KISS) is a design principle which states that designs and/or systems should be as simple as possible. Wherever possible, complexity should be avoided in a system—as simplicity guarantees the greatest levels of user acceptance and interaction.

Principio n° 2

"A complex system that works is invariably found to have evolved from a simple system that worked."

Systemantics
How Systems Work and Especially How They Fail
John Gall

Principio n° 2

"A complex system designed from scratch never works and cannot be patched up to make it work. You have to start over with a working simple system."

Systemantics
How Systems Work and Especially How They Fail
John Gall

Principio n° 3

Principio n° 3

ehi... un momento

Principio n° 4

Principio n° 4

ok mi hai incuriosito, ti ascolto

Come faccio a costruire qualcosa che rispetti questi quattro principi in modo efficace?

AWS CDK

github.com/aws/aws-cdk

AWS CDK

The AWS Cloud Development Kit (AWS CDK) is an open-source software development framework to define cloud infrastructure in code and provision it through AWS CloudFormation.

Infrastruttura AWS

Costruiamo il backend di un form di upload

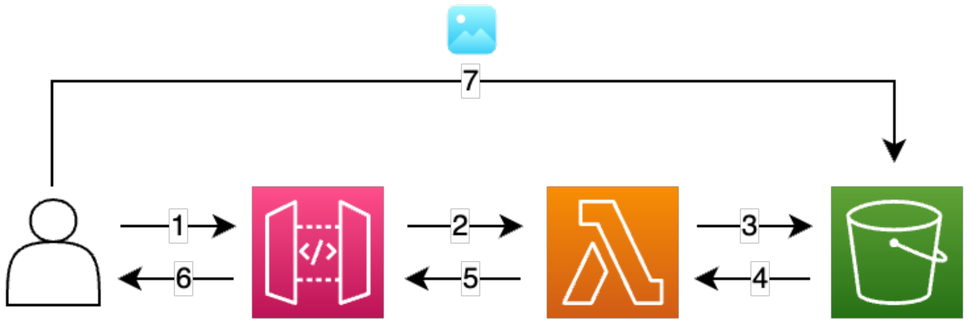
Infrastruttura AWS

Costruiamo il backend di un form di upload

con 45 linee di Typescript

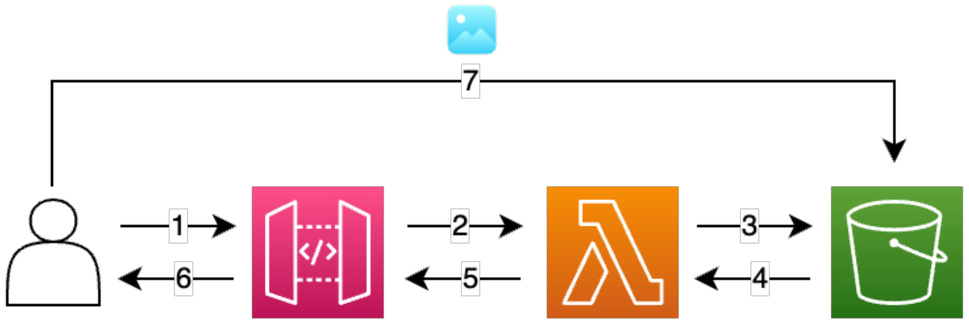
Step 1

Costruiamo il backend di un form di upload



Step 1

Vediamo il codice



Step 1

```
You, 1 minute ago | 1 author (You)
export class UploadFormStack extends cdk.Stack {
  constructor(scope: cdk.App, id: string, props: UploadFormStackProps) {
    super(scope, id, props);

    // create content s3 bucket
    this.contentBucket = new s3.Bucket(this, props.stage.toString() + '-content', {
      encryption: BucketEncryption.S3_MANAGED,
      publicReadAccess: false
    })

    // create lambda policy statement for operating over s3
    var lambdaS3PolicyStatement = new iam.PolicyStatement()
    lambdaS3PolicyStatement.addActions(
      's3:PutObject',
      's3:GetObject'
    )
    lambdaS3PolicyStatement.addResources(
      this.contentBucket.bucketArn + '/*'
    );
```

Step 1

```
44    // create s3 authorizer lambda
45    this.s3AuthLambda = new lambda.Function(this, props.stage.toString() + '-s3-auth', {
46      runtime: lambda.Runtime.NODEJS_18_X,
47      handler: 'index.handler',
48      code: lambda.Code.fromAsset('./s3-authorizer'),
49      environment: { 'S3_BUCKET': this.contentBucket.bucketName },
50      initialPolicy: [lambdaS3PolicyStatement]
51    })
52
53    // give to apigateway permission to invoke the lambda
54    new lambda.CfnPermission(this, props.stage.toString() + '-apigtw-lambda-permission', {
55      functionName: this.s3AuthLambda.functionArn,
56      action: 'lambda:InvokeFunction',
57      principal: 'apigateway.amazonaws.com'
58    })
59
60    // defines an API Gateway REST API resource backed by our s3 uploader function.
61    this.uploadApiAuthorizer = new apigateway.LambdaRestApi(this, props.stage.toString() + '-apigtw', {
62      handler: this.s3AuthLambda,
63      deployOptions: { stageName: props.stage.toString() }
64    });
```

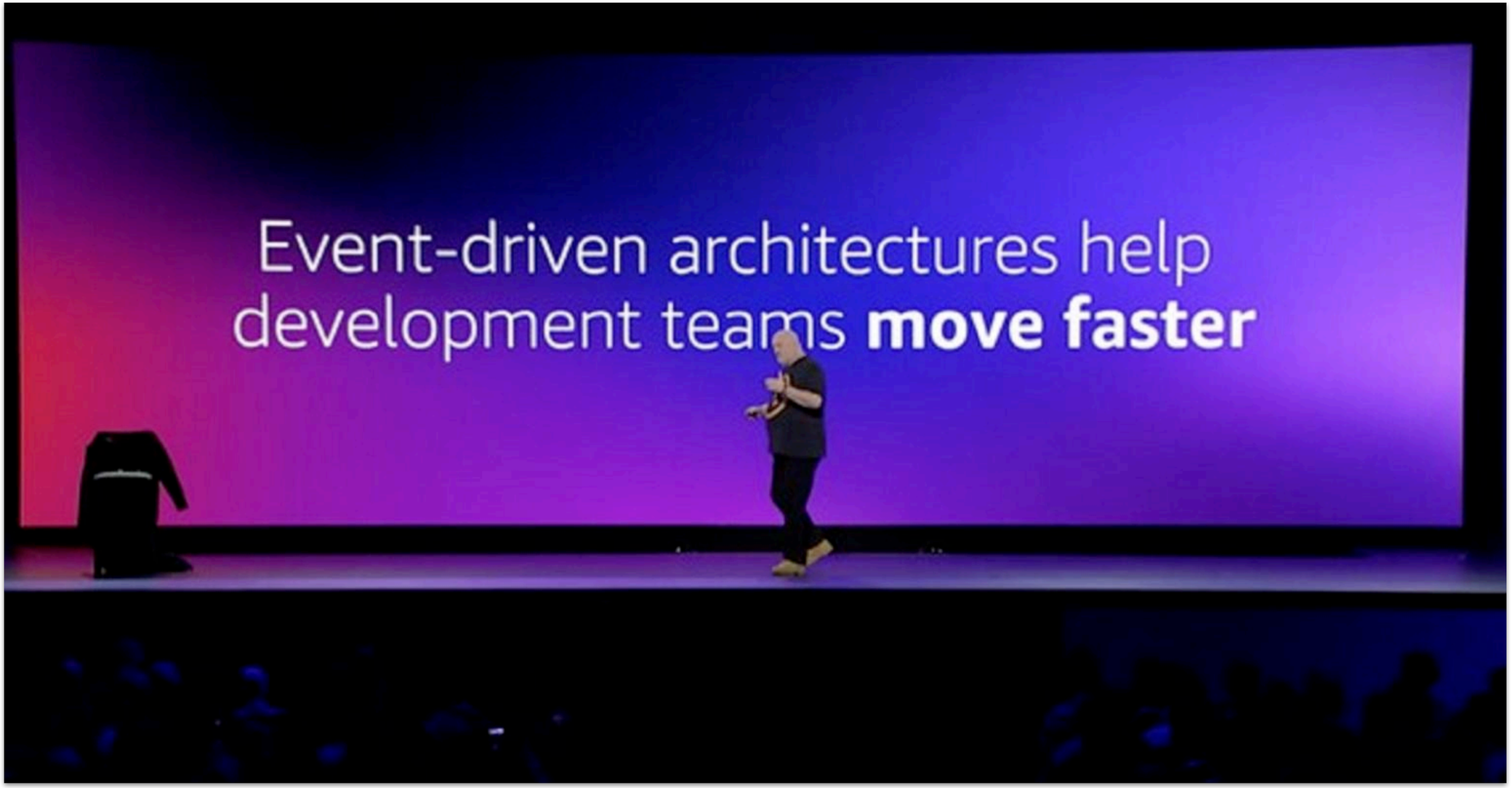
Step 1

```
You, 1 minute ago | 1 author (You)
export class UploadFormStack extends cdk.Stack {
  constructor(scope: cdk.App, id: string, props: UploadFormStackProps) {
    super(scope, id, props);

    // create content s3 bucket
    this.contentBucket = new s3.Bucket(this, props.stage.toString() + '-content', {
      encryption: BucketEncryption.S3_MANAGED,
      publicReadAccess: false
    })

    // create lambda policy statement for operating over s3
    var lambdaS3PolicyStatement = new iam.PolicyStatement()
    lambdaS3PolicyStatement.addActions(
      's3:PutObject',
      's3:GetObject'
    )
    lambdaS3PolicyStatement.addResources(
      this.contentBucket.bucketArn + '/*'
    );
```

Step 1

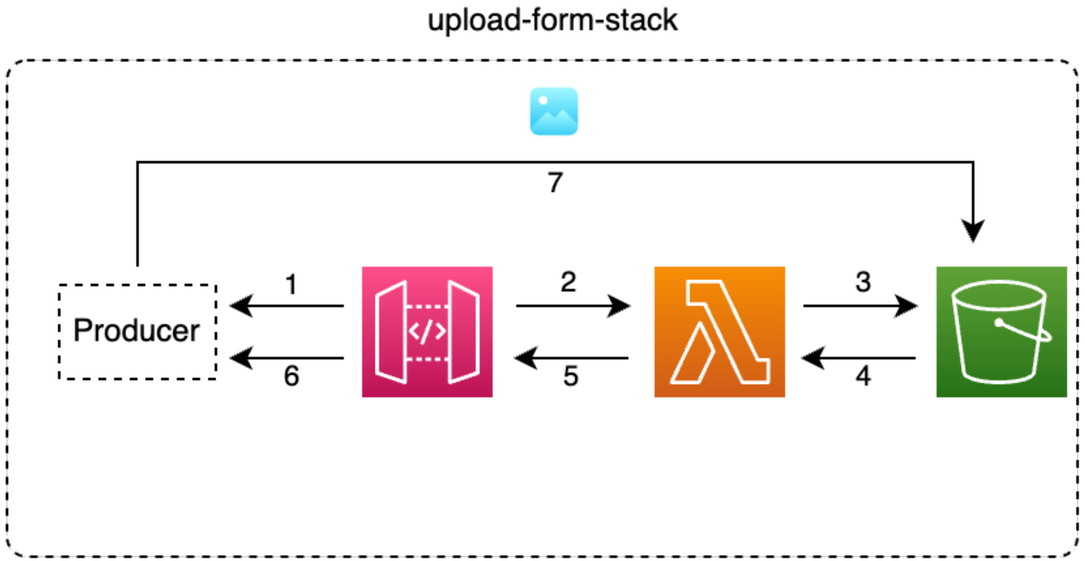
```
44    // create s3 authorizer lambda
45    this.s3AuthLambda = new lambda.Function(this, props.stage.toString() + '-s3-auth', {
46      runtime: lambda.Runtime.NODEJS_18_X,
47      handler: 'index.handler',
48      code: lambda.Code.fromAsset('./s3-authorizer'),
49      environment: { 'S3_BUCKET': this.contentBucket.bucketName },
50      initialPolicy: [lambdaS3PolicyStatement]
51    })
52
53    // give to apigateway permission to invoke the lambda
54    new lambda.CfnPermission(this, props.stage.toString() + '-apigtw-lambda-permission', {
55      functionName: this.s3AuthLambda.functionArn,
56      action: 'lambda:InvokeFunction',
57      principal: 'apigateway.amazonaws.com'
58    })
59
60    // defines an API Gateway REST API resource backed by our s3 uploader function.
61    this.uploadApiAuthorizer = new apigateway.LambdaRestApi(this, props.stage.toString() + '-apigtw', {
62      handler: this.s3AuthLambda,
63      deployOptions: { stageName: props.stage.toString() }
64    });
```

Bello, ma a cosa mi serve un upload-form nel 2023?


Event-driven architectures help
development teams **move faster**

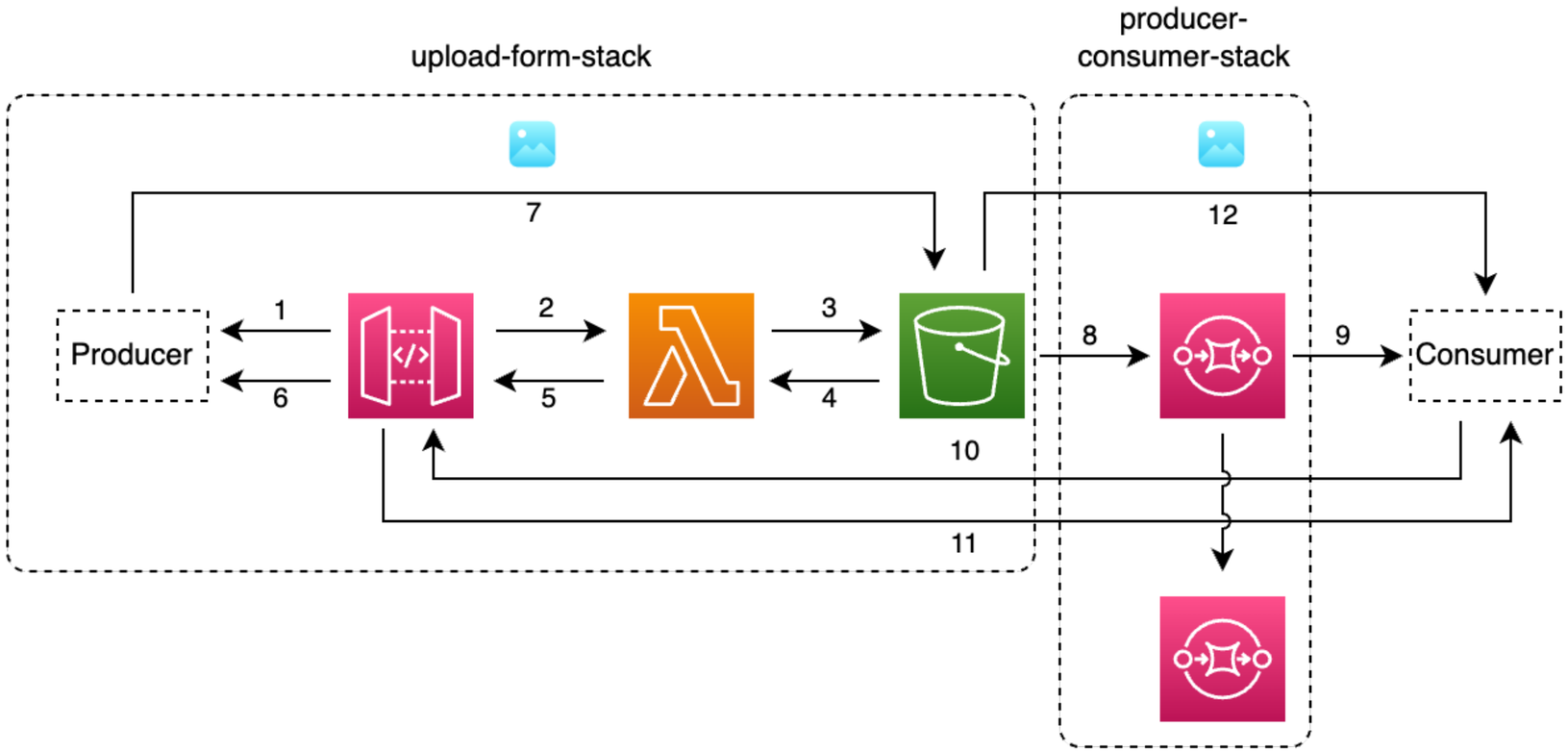
Step 2

Attraverso l'ereditarietà estendiamo lo stack precedente...



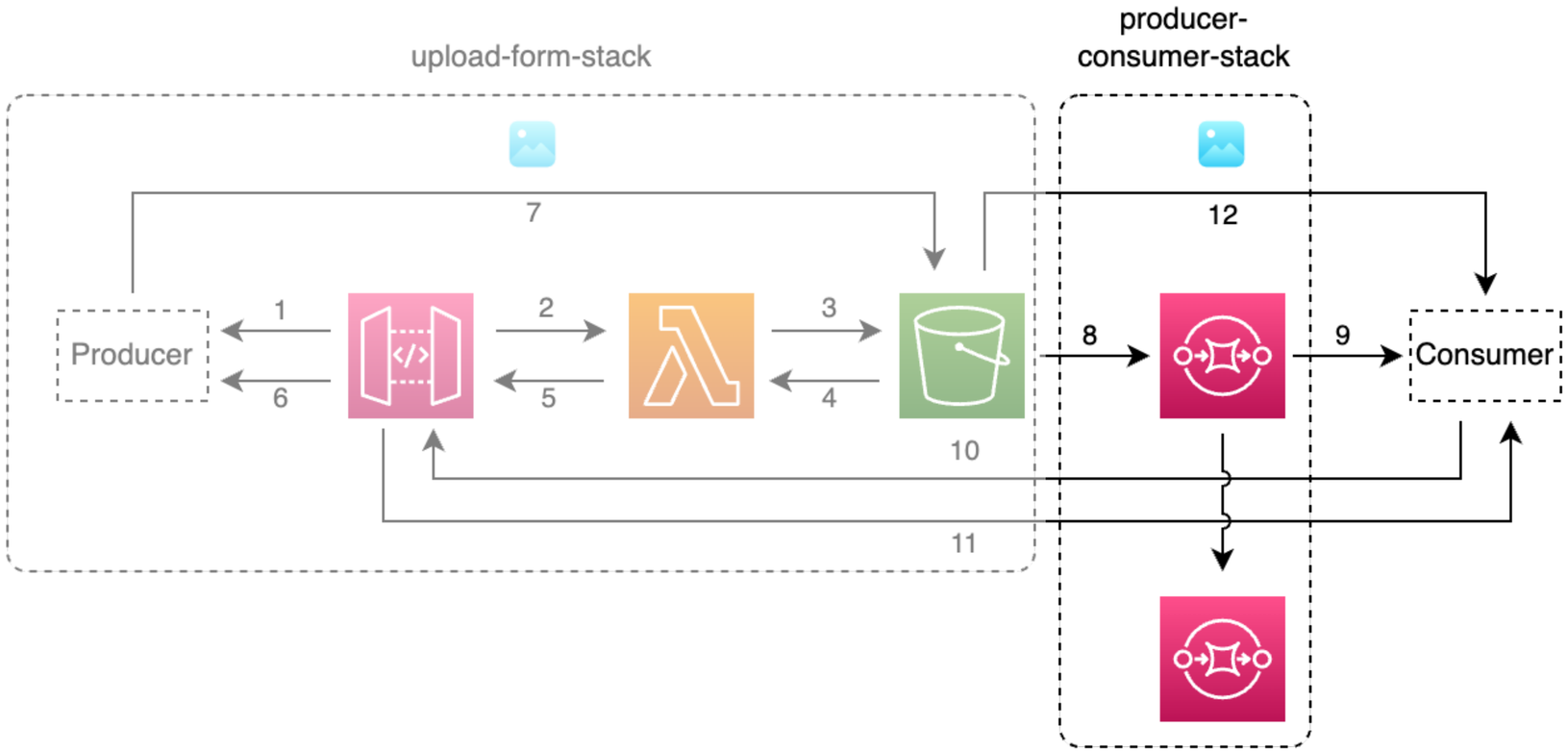
Step 2

...e costruiamo un sistema producer/consumer



Step 2

...e costruiamo un sistema producer/consumer



Step 2

```
18  export class ProducerConsumerStack extends UploadFormStack {
19    constructor(scope: cdk.App, id: string, props: ProducerConsumerStackProps) {
20      super(scope, id, props);
21
22      // create trail to enable events on s3
23      const trail = new cloudtrail.Trail(this, props.stage.toString() + 'content-bucket-trail');
24      trail.addS3EventSelector([{ bucket: this.contentBucket, objectPrefix: "/*",}], {
25        includeManagementEvents: false,
26        readWriteType: ReadWriteType.ALL,
27      });
28
29      // create sqs queue
30      this.sqsQueue = new sqs.Queue(this, props.stage.toString() + "-sqs-content-queue", {
31        retentionPeriod: cdk.Duration.days(14),
32        visibilityTimeout: cdk.Duration.seconds(360),
33        deadLetterQueue: {
34          queue: new sqs.Queue(this, props.stage.toString() + "-sqs-dead-queue",
35          { retentionPeriod: cdk.Duration.days(14) }),
36          maxReceiveCount: 5
37        }
38      });
```

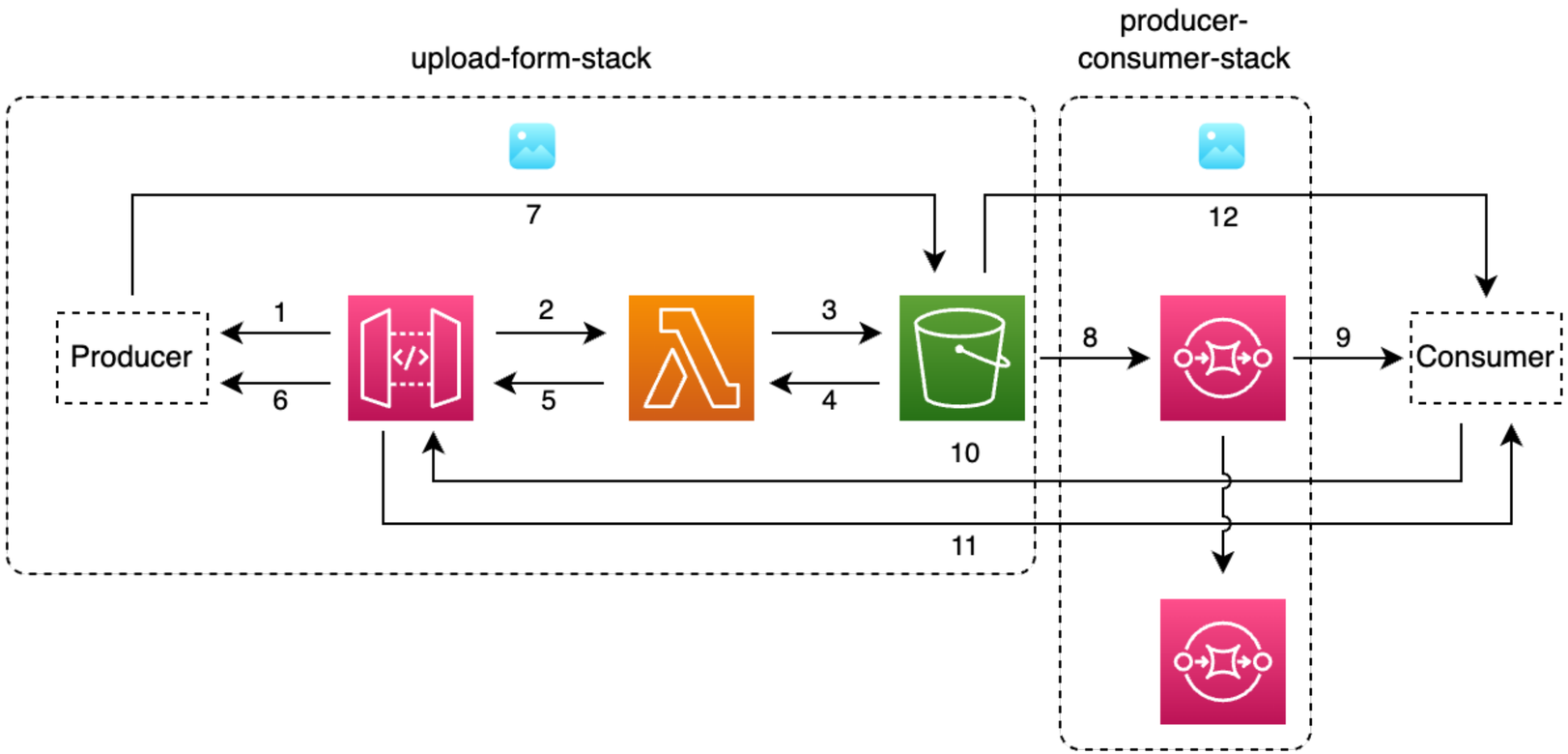
Step 2

```
// create s3 sqs policy to allow notification
new iam.Role(this, props.stage.toString()+'s3-sqs-notification', {
  assumedBy: new iam.ServicePrincipal('s3.amazonaws.com'),
  inlinePolicies: {
    "S3SQSNotification": new iam.PolicyDocument({
      statements: [new iam.PolicyStatement({
        effect: iam.Effect.ALLOW,
        actions: [ 'SQS:SendMessage', ],
        resources: [this.sqsQueue.queueArn],
      })],
    })
  },
  description: 'S3 SQS Notification',
});

// create sqs policy statement to allow cloudwatch pushing the message
var s3SQSStatement = new iam.PolicyStatement()
s3SQSStatement.addActions(
  "sqs:SendMessage",
  "sqs:SendMessageBatch",
  "sqs:GetQueueAttributes",
  "sqs:GetQueueUrl")
s3SQSStatement.addResources(this.sqsQueue.queueArn);
s3SQSStatement.addAnyPrincipal()
s3SQSStatement.addCondition("ArnLike", { "aws:SourceArn": this.contentBucket.bucketArn })

this.contentBucket.addEventNotification(s3.EventType.OBJECT_CREATED, new s3n.SqsDestination(this.sqsQueue), {
  prefix: "input/"
})
```

Vediamo come estendere il backend
creato a un caso d’uso concreto

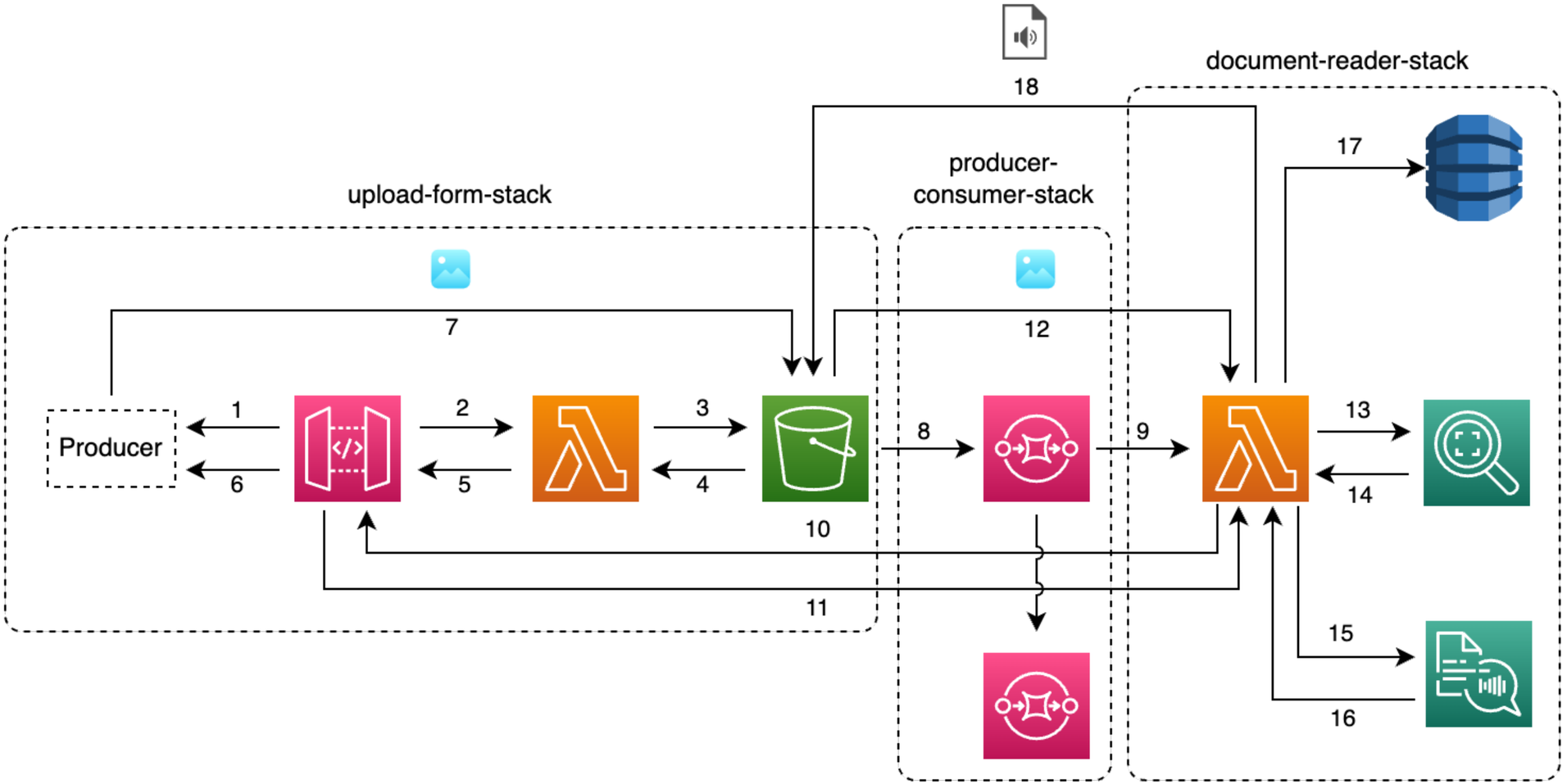
Step 3

Sfruttando servizi managed AWS come Rekognition e Polly...



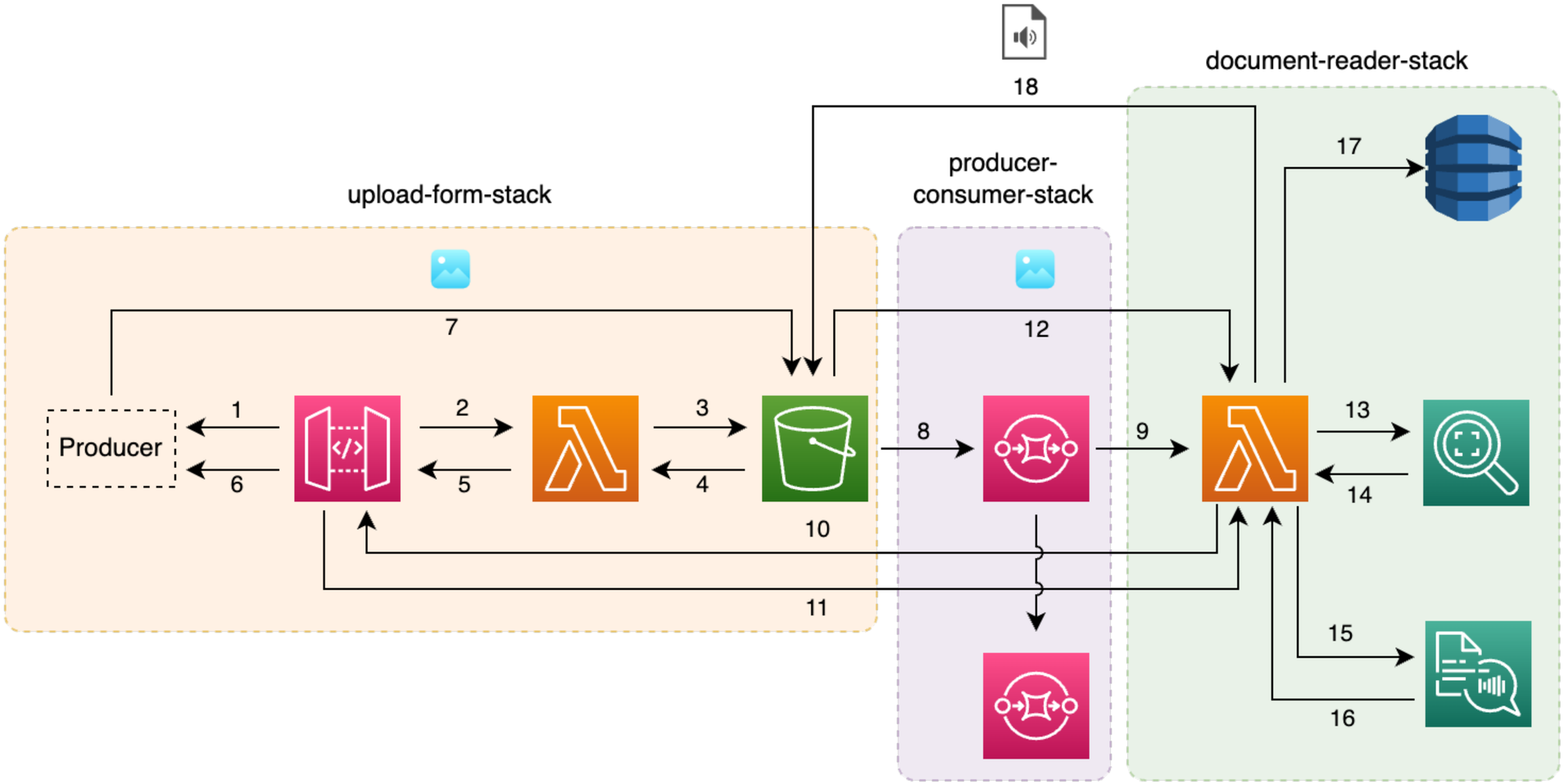
Step 3

...con una terza *extend* costruiamo un OCR serverless



Step 3

...con una terza *extend* costruiamo un OCR serverless



Step 3

...con una terza *extend* costruiamo un OCR serverless

```
21  export class DocumentReaderStack extends ProducerConsumerStack {
22    constructor(scope: cdk.App, id: string, props: DocumentReaderStackProps) {
23      super(scope, id, props);
24
25      // create tables for images, text and registration references
26      this.parsedDocument = new dynamodb.Table(this, props.stage.toString() + '-parsed-document', {
27        readCapacity: props.readCapacity,
28        writeCapacity: props.writeCapacity,
29        partitionKey: {
30          name: props.partitionKey,
31          type: AttributeType.STRING
32        }
33      })
```

Demo

Se Internet e la fortuna ci assistono lo vediamo live

Grazie dell’attenzione 🙏

Q&A

@made2591 su Github/Twitter
madeddu.xyz blog

Tutto il materiale
github.com/made2591/serverless-cdk-stacks

SUMMIT
MILANO