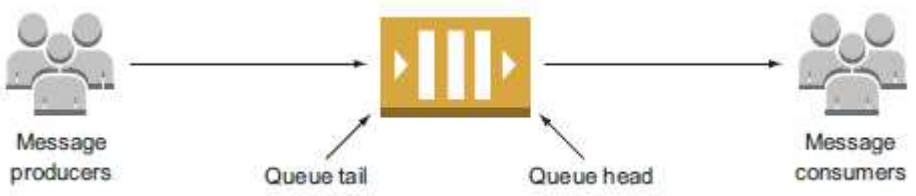
3

Alta disponibilidad

3.7 Desacoplando la arquitectura

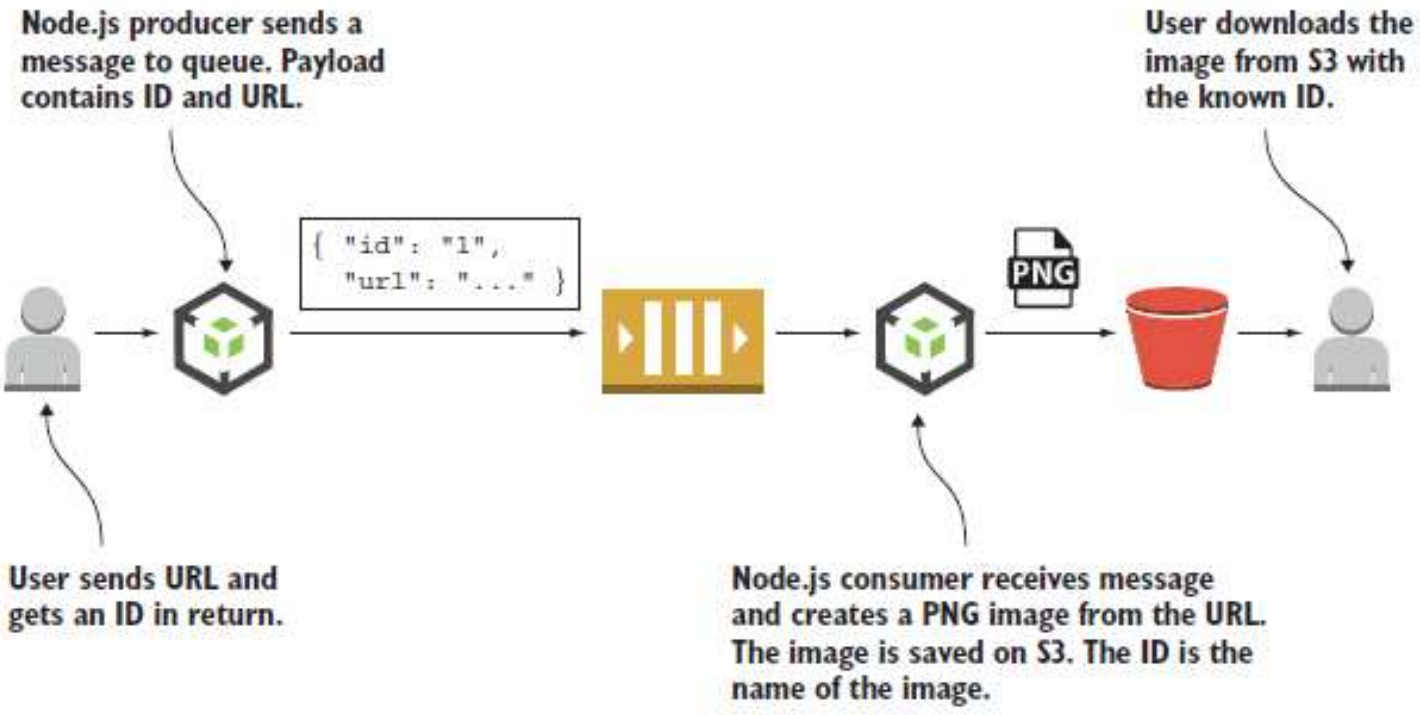
Mondragon Unibertsitatea
Goi Eskola Politeknikoa

- Desacople asíncrono mediante colas de mensajes
  - SQS
  - Requiere modificar el código de la aplicación
  - Productor/Consumidor
  - Mensajes se guardan durante un tiempo limitado
  - Las lecturas de mensajes se deben de notificar
  - Es un servicio autogestionado
  - No se asegurar el orden de los mensajes
  - Es un servicio autoescalado



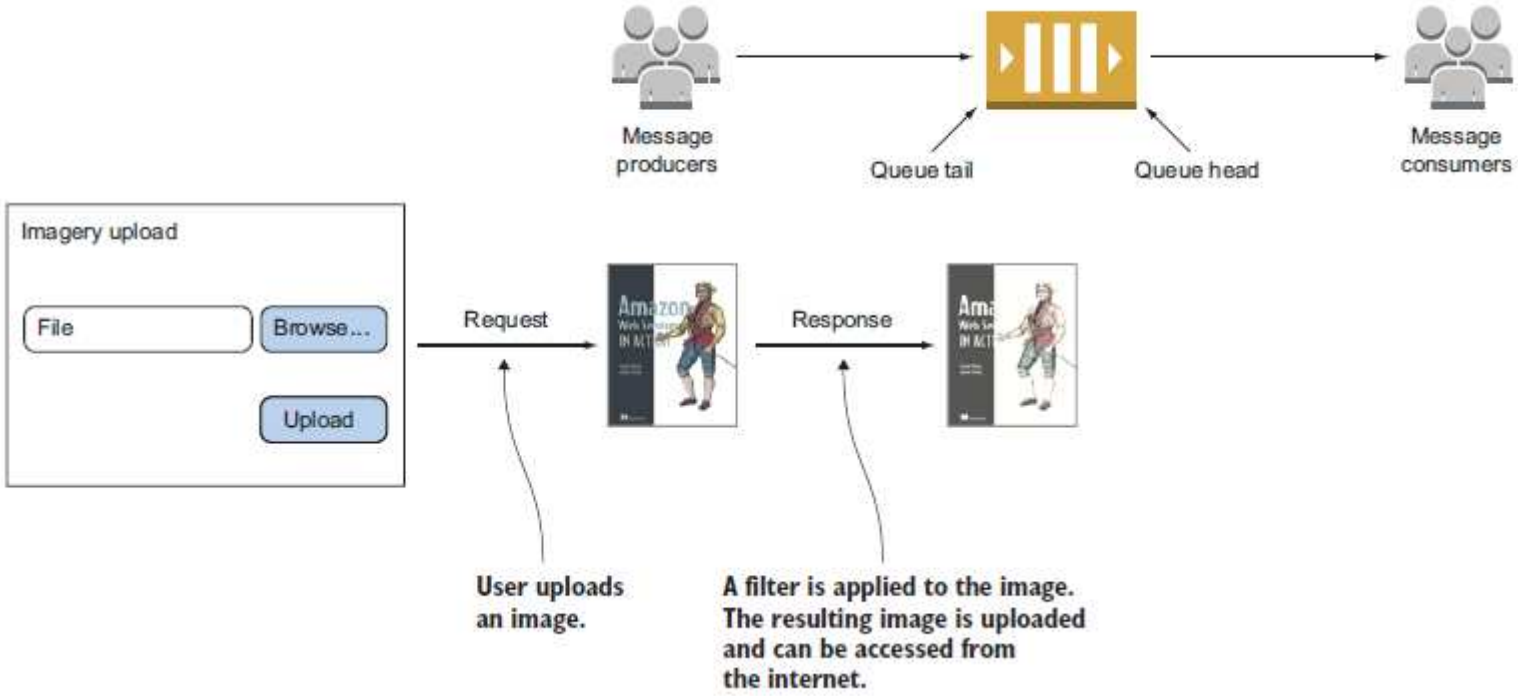
3.7 Desacoplando la arquitectura

- Ejemplo SQS : URL2PNG



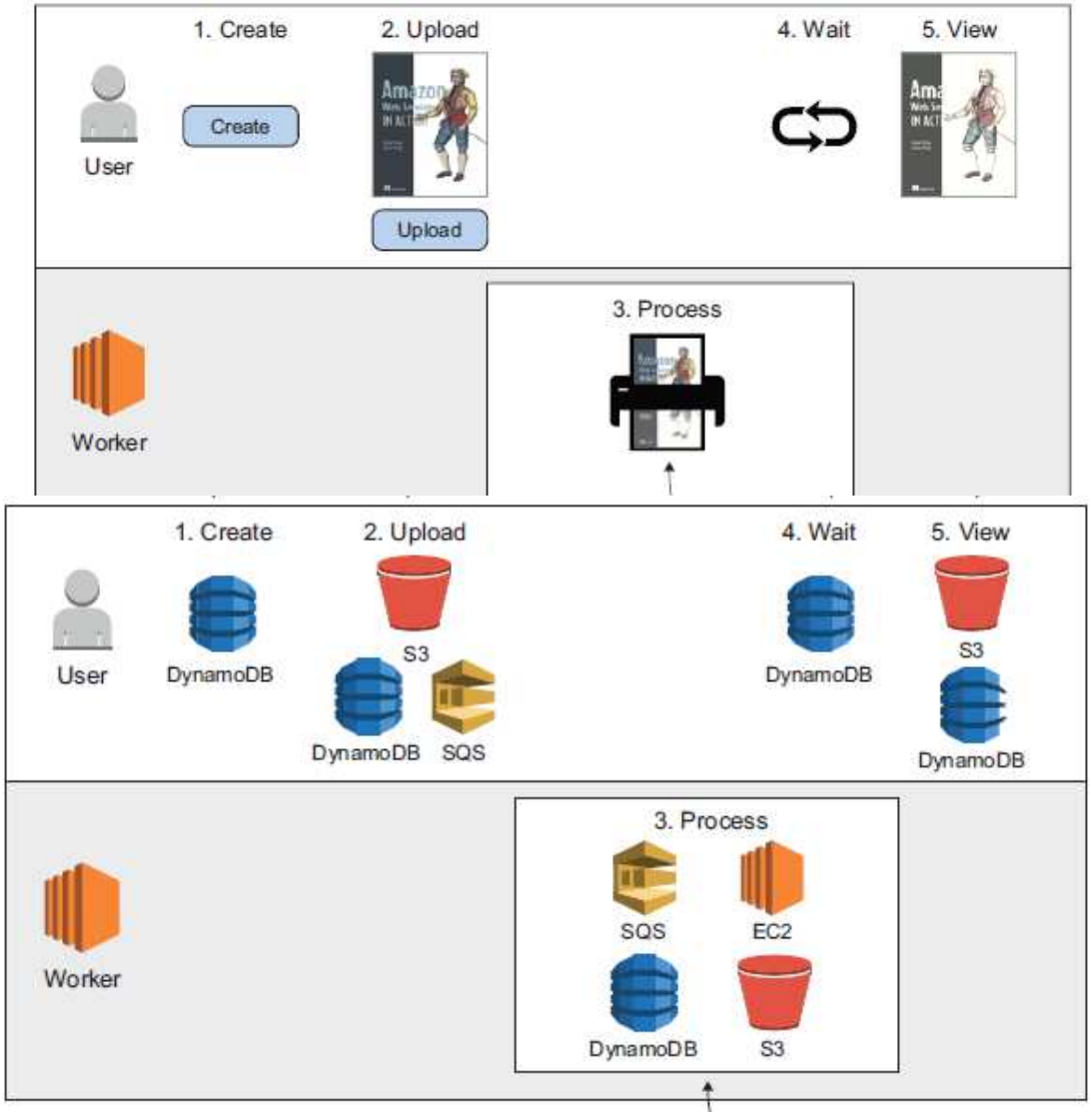
3.7 Desacoplando la arquitectura

- Arquitectura asicnrono con tolerancia a fallos
- Aplicación galería de imágenes
  - Subir imagen y procesar a S3
  - Ver imágenes
  - Arquitectura asíncrono y tolerante a fallos



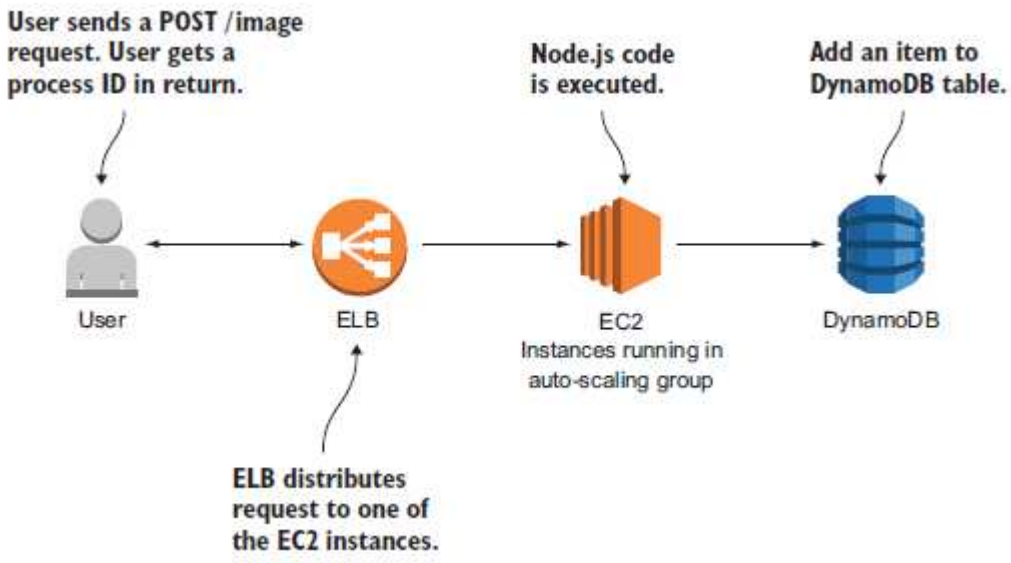
3.7 Desacoplando la arquitectura

- Arquitectura asíncrono con tolerancia a fallos



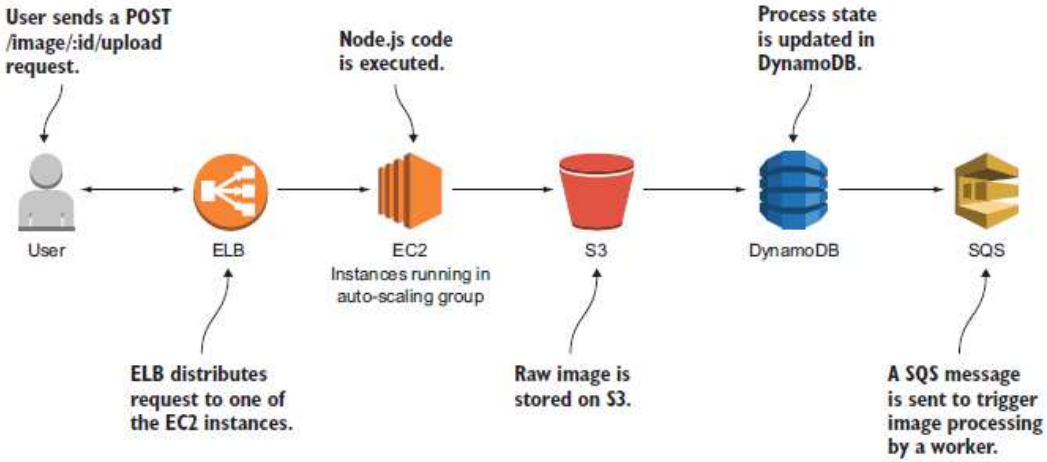
3.7 Desacoplando la arquitectura

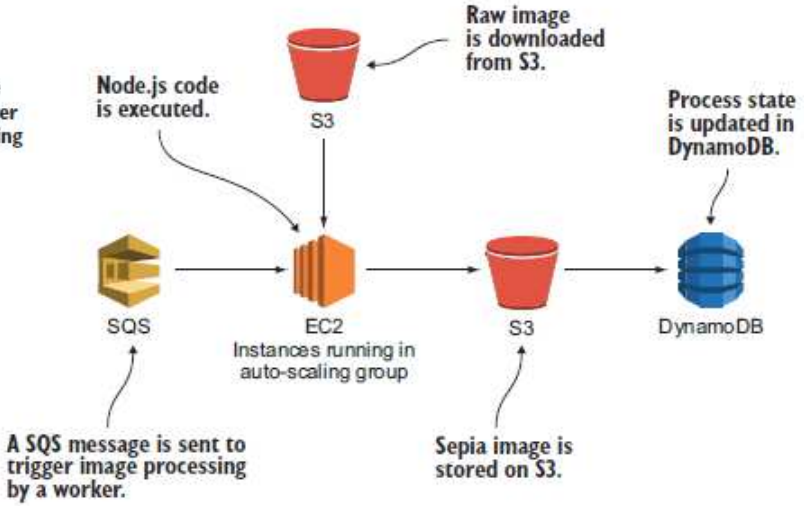
- Arquitectura asíncrono con tolerancia a fallos
- API REST : Servidor node.js + express.js



3.7 Desacoplando la arquitectura

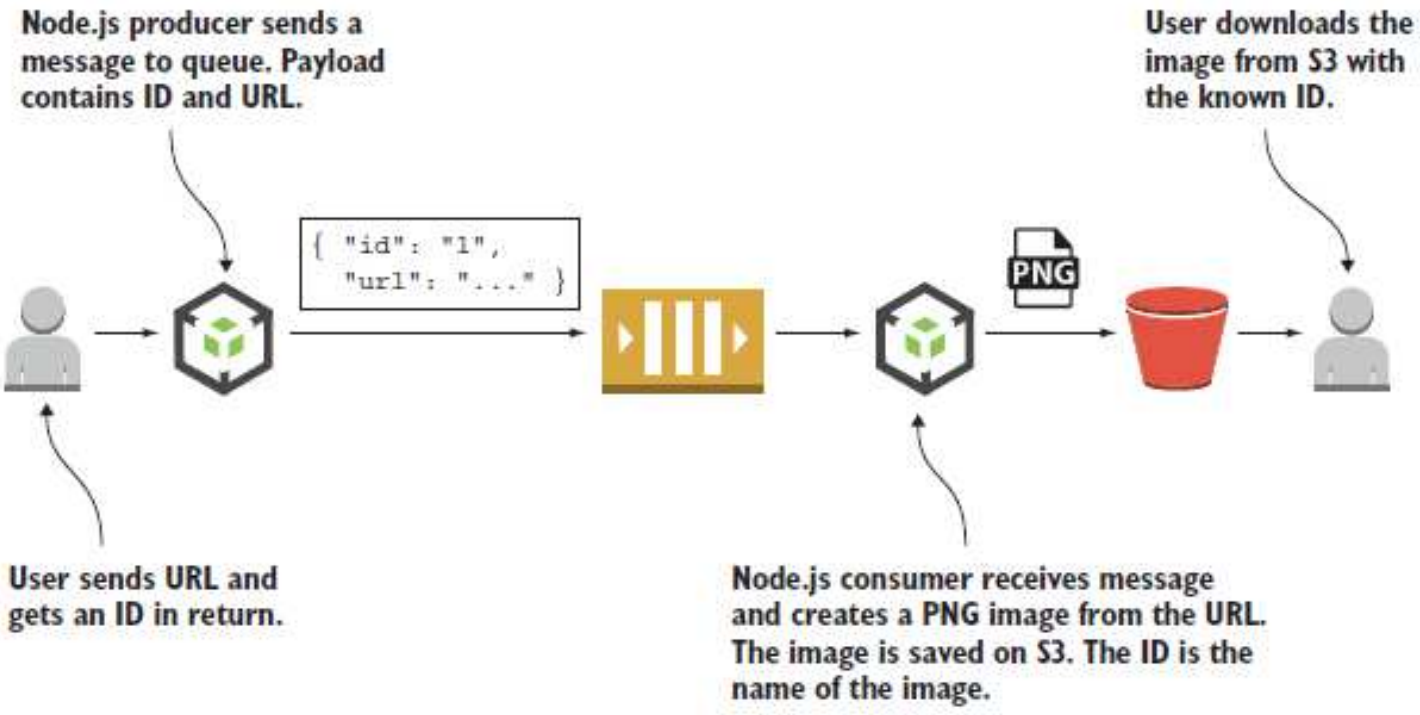
- Arquitectura asíncrono con tolerancia a fallos
- Worker recibe mensaje procesa e indica al servidro via post
- El servidor actualiza la información
- Y cuando el front-end hace refresh actualiza la imagen





3.7 Desacoplando la arquitectura

- Ejemplo SQS : URL2PNG



3.7 Desacoplando la arquitectura

Mondragon Unibertsitatea
Goi Eskola Politeknikoa

- Ejemplo SQS : URL2PNG : Código productor
- Utilizaremos el CLI
- Crear bucket
  - $aws s3 mb s3://bucket-name
- Configurar bucket como página WEB
  - aws s3 website s3://bucketname --index-document index.html --error-document error.html

```
PS C:\Users\jaagirre> aws configure
AWS Access Key ID [****************6C7F]: AKIAWD4
AWS Secret Access Key [****************j7D1]: au0
Default region name [us-west-2]: eu-west-1
Default output format [json]:
```

```
PS C:\Users\jaagirre> aws s3 ls
C:\Program Files\Amazon\AWSCLI\.\dateutil\parser\_parser.py:1175: UnicodeWarning: Unicode equal
convert both arguments to Unicode - interpreting them as being unequal
2019-09-17 17:04:03 cf-templates-6smw15w7vf9n-us-east-1
2019-02-08 17:22:39 cf-templates-6smw15w7vf9n-us-west-2
2019-09-12 14:32:40 facturas-jaagibas-master
2019-09-12 14:33:19 facturas-master-jaagibas
2019-10-01 16:30:23 url2png-jaagirre
2019-09-30 16:50:51 wordpress-master-practicas
```

3.7 Desacoplando la arquitectura

Mondragon Unibertsitatea
Goi Eskola Politeknikoa

- Ejemplo SQS : URL2PNG : Código productor
- Crear cola de mensaje

```
PS C:\Users\jaagirre> aws sqs create-queue --queue-name url2png
{
    "QueueUrl": "https://eu-west-1.queue.amazonaws.com/420693608596/url2png"
}
```

3.7 Desacoplando la arquitectura

Mondragon Unibertsitatea
Goi Eskola Politeknikoa

- Ejemplo SQS : URL2PNG : Código consumidor
- Instalar en local Node.js
- Descargar código cliente AWSInAction
- Para editar/leer código “Visual Code”
  - Configurar la región en el cliente de la cola **index.js**
  - Configurar la URL de la cola SQS y el nombre del bucket en **config.json**

3.7 Desacoplando la arquitectura

Mondragon Unibertsitatea
Goi Eskola Politeknikoa

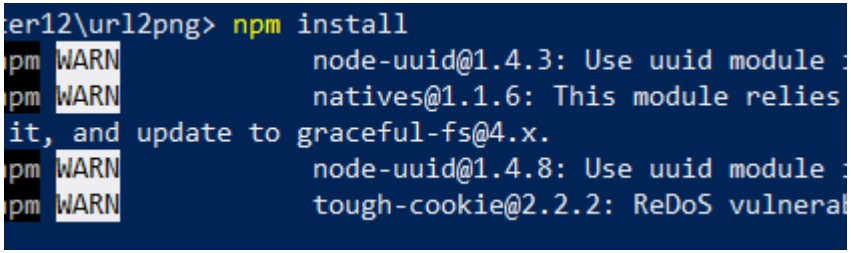
- Ejemplo SQS : URL2PNG : Código consumidor
- La aplicación crea un objeto AWS SQS
- Crea un mensaje con
  - Crea un id aleatorio
  - URL para convertir en imagen
- Envía mensaje
  - Función Callback

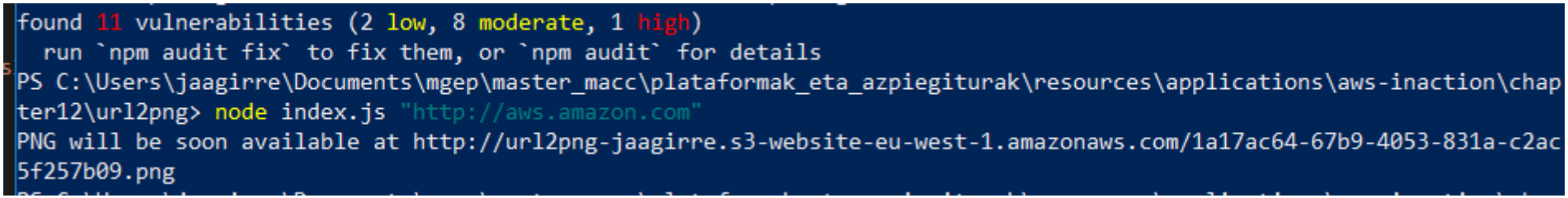
```
var AWS = require('aws-sdk');
var uuid = require('node-uuid');
var config = require('./config.json');
var sqs = new AWS.SQS({
    "region": "eu-west-1"
});

if (process.argv.length !== 3) {
    console.log('URL missing');
    process.exit(1);
}

var id = uuid.v4();
var body = {
    "id": id,
    "url": process.argv[2]
};

sqs.sendMessage({
    "MessageBody": JSON.stringify(body),
    "QueueUrl": config.QueueUrl
Complexity is 3 Everything is cool!
}, function(err) {
    if (err) {
        console.log('error', err);
    } else {
        console.log('PNG will be soon available at http://' + config.Bucket + '.s3-website-eu-we
    }
});
```

3.7 Desacoplando la arquitectura

Mondragon Unibertsitatea
Goi Eskola Politeknikoa

- Ejemplo SQS : URL2PNG : Código productor
- Para ejecutar la aplicación
- Instalar dependencias: aws-sdk, …
  - npm install
- Ejecutar aplicación cliente
  - Node index.js “http://urlquesequiera”

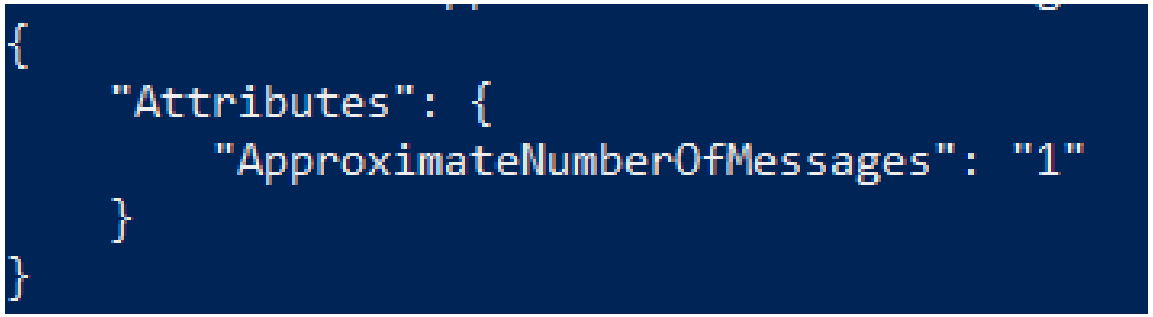




3.7 Desacoplando la arquitectura

- Ejemplo SQS : URL2PNG : Código productor
- Comprobar que el mensaje etsa en la cola

aws sqs get-queue-attributes --queue-url https://eu-west-1.queue.amazonaws.com/420693608596/url2png --attribute-names ApproximateNumberOfMessages



3.7 Desacoplando la arquitectura

- Ejemplo SQS : URL2PNG : Código consumidor

```
function run() {
    Complexity is 14 You must be kidding
    receive(function(err, message) {
        if (err) {
            throw err;
        } else {
            if (message === null) {
                console.log('nothing to do');
                setTimeout(run, 1000);
            } else {
                console.log('process');
                Complexity is 8 It's time to do something...
                process(message, function(err) {
                    if (err) {
                        throw err;
                    } else {
                        Complexity is 4 Everything is cool!
                        acknowledge(message, function(err) {
                            if (err) {
                                throw err;
                            } else {
                                console.log('done');
                                setTimeout(run, 1000);
                            }
                        });
                    }
                });
            }
        }
    });
}
```

```
function receive(cb) {
    var params = {
        "QueueUrl": config.QueueUrl,
        "MaxNumberOfMessages": 1,
        "VisibilityTimeout": 120,
        "WaitTimeSeconds": 10
    };
    Complexity is 5 Everything is cool!
    sqs.receiveMessage(params, function(err, data) {
        if (err) {
            cb(err);
        } else {
            if (data.Messages === undefined) {
                cb(null, null);
            } else {
                cb(null, data.Messages[0]);
            }
        }
    });
}
```

```
function acknowledge(message, cb) {
    var params = {
        "QueueUrl": config.QueueUrl,
        "ReceiptHandle": message.ReceiptHandle
    };
    sqs.deleteMessage(params, cb);
}
```

3.7 Desacoplando la arquitectura

- Ejemplo SQS : URL2PNG : Código consumidor

```
function process(message, cb) {
    var body = JSON.parse(message.Body);
    var file = body.id + '.png';
    Complexity is 9 It's time to do something...
    webshot(body.url, file, function(err) {
        if (err) {
            cb(err);
        } else {
            Complexity is 6 It's time to do something...
            fs.readFile(file, function(err, buf) {
                if (err) {
                    cb(err);
                } else {
                    var params = {
                        "Bucket": config.Bucket,
                        "Key": file,
                        "ACL": "public-read",
                        "ContentType": "image/png",
                        "Body": buf
                    };
                    Complexity is 3 Everything is cool!
                    s3.putObject(params, function(err) {
                        if (err) {
                            cb(err);
                        } else {
                            fs.unlink(file, cb);
                        }
                    });
                }
            });
        }
    });
}
```

3.7 Desacoplando la arquitectura

- Ejemplo SQS : URL2PNG : Ejecución

```
ter12\url2png> node index.js "http://www.mondragon.edu/es/escuela-politecnica-superior"
PNG will be soon available at http://url2png-jaagirre.s3-website-eu-west-1.amazonaws.com/7c7f5df0-eb17-4475-9385-75ab
1308038a.png
```

```
ter12\url2png> node .\worker.js
process
done
nothing to do
nothing to do
nothing to do
nothing to do
process
done
nothing to do
```

3.7 Desacoplando la arquitectura

- Ejemplo SQS : URL2PNG : Ejecución
- Borrar recursos creados

aws sqs delete-queue --queue-url https://eu-west-1.queue.amazonaws.com/420693608596/url2png

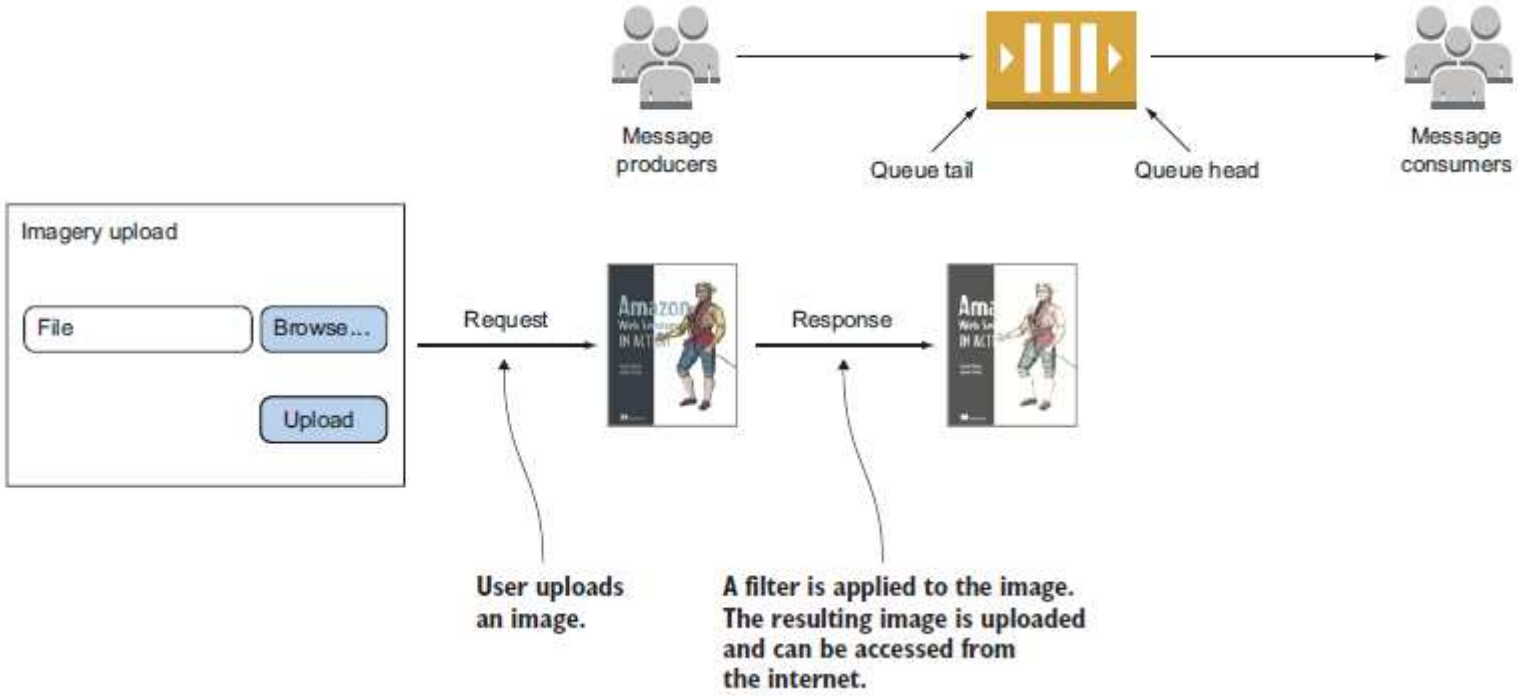
aws sqs list-queues

aws s3 rb --force s3://url2png-jaagirre

```
delete: s3://url2png-jaagirre/1a17ac64-67b9-4053-831a-c2ac5f257b09.png
delete: s3://url2png-jaagirre/7c7f5df0-eb17-4475-9385-75abd308038a.png
delete: s3://url2png-jaagirre/ecf8a8d4-d464-4415-8f4d-e93b0820a326.png
remove_bucket: url2png-jaagirre
```

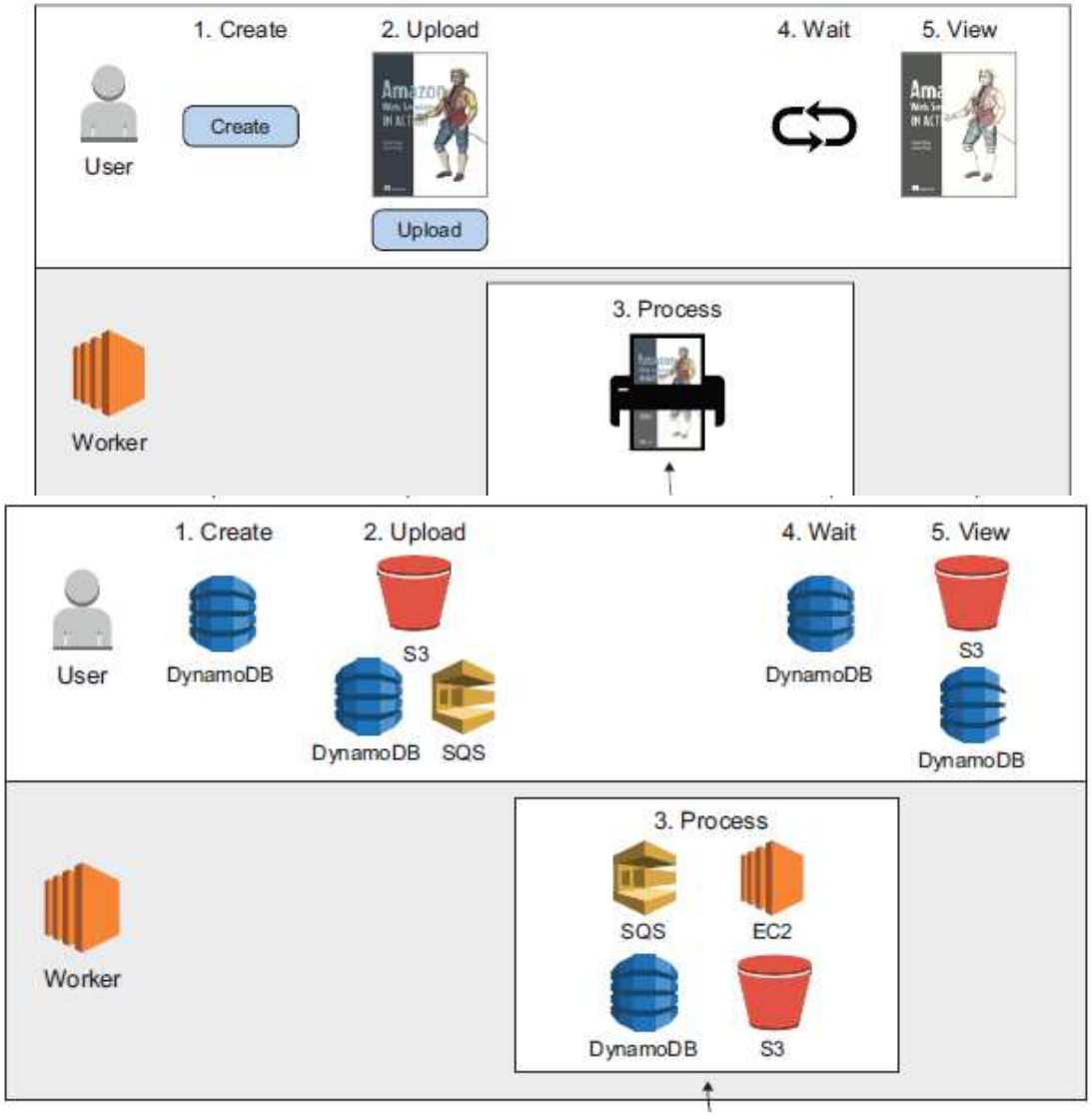
3.7 Desacoplando la arquitectura

- Arquitectura asicnrono con tolerancia a fallos
- Aplicación galería de imágenes
  - Subir imagen y procesar a S3
  - Ver imágenes
  - Arquitectura asíncrono y tolerante a fallos



3.7 Desacoplando la arquitectura

- Arquitectura asíncrono con tolerancia a fallos

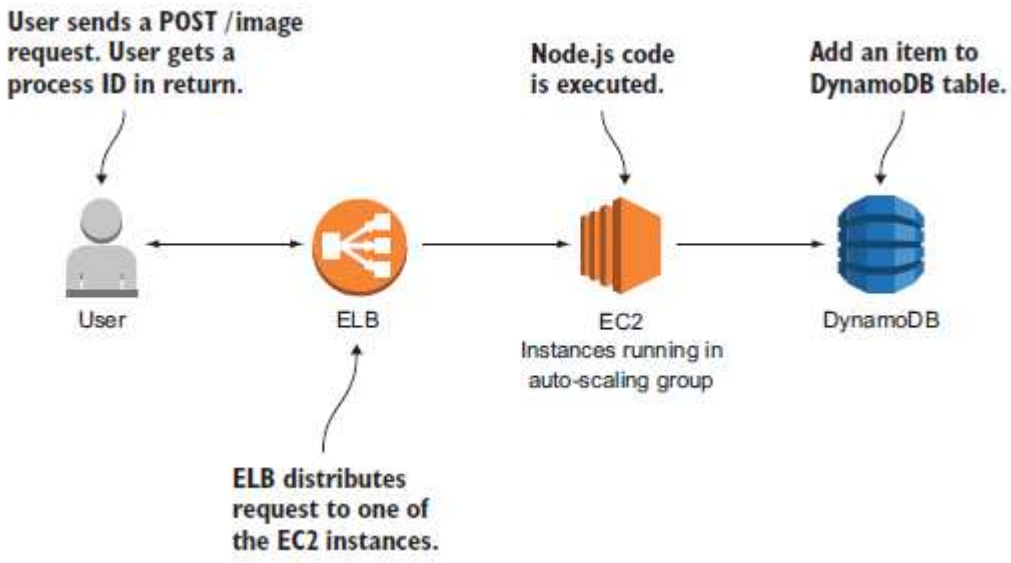


3.7 Desacoplando la arquitectura

- Arquitectura asíncrono con tolerancia a fallos
- Aplicación WEB : Consumidor API REST
- Aplicación : Servidor API REST
  - Implementado por server.js
  - Consumido por página WEB
    - `POST /image`—A new image process is created when executing this route.
    - `GET /image/:id`—This route returns the state of the process specified with the path parameter `:id`.
    - `POST /image/:id/upload`—This route offers a file upload for the process specified with the `path` parameter `:id`.
- Worker : Procesamiento de imagen

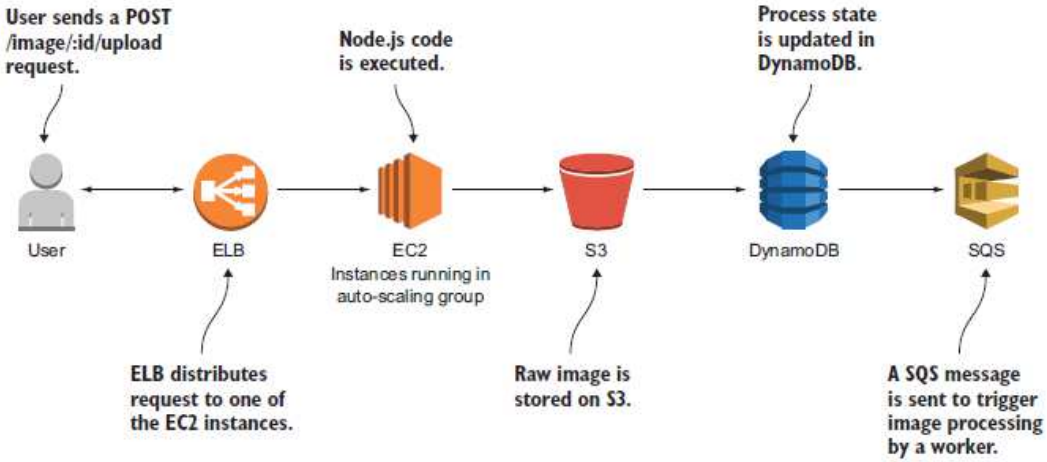
3.7 Desacoplando la arquitectura

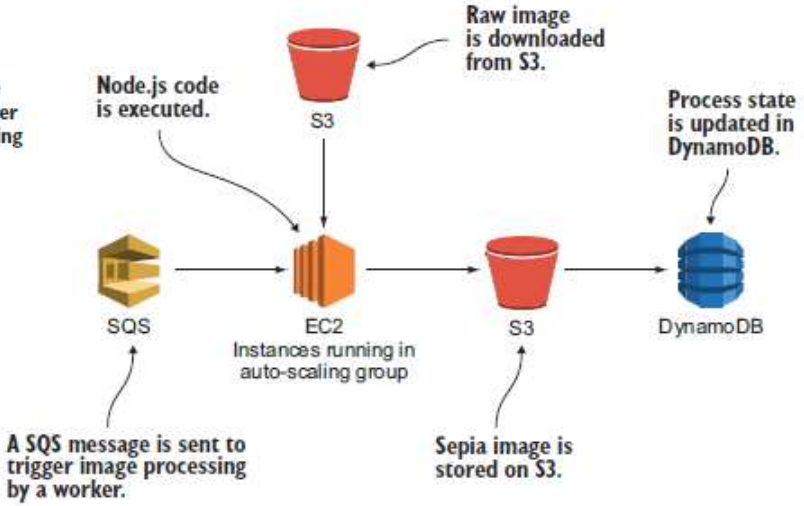
- Arquitectura asíncrono con tolerancia a fallos
- API REST : Servidor node.js + express.js



3.7 Desacoplando la arquitectura

- Arquitectura asíncrono con tolerancia a fallos
- Worker recibe mensaje procesa e indica al servidro via post
- El servidor actualiza la información
- Y cuando el front-end hace refresh actualiza la imagen





3.7 Desacoplando la arquitectura

- Arquitectura asíncrono con tolerancia a fallos
- Despliegue aplicación mediante CLOUDFORMATION y ElasticBeanstalk
- Analizar fichero CLOUDFORMATION template.json
  - Cloudformation para crear el bucket S3 CON accountId
  - La cola SQS
  - La base de datos
  - La instancia EC2 de workder y servidor con páginas estaticas via ElastinBeanStalk con ROLE y autoescalado
    - SQS, S3 , CLOUDWATCH
  - En elasticbeanstak ahí que ofrecer el .zip

3.7 Desacoplando la arquitectura

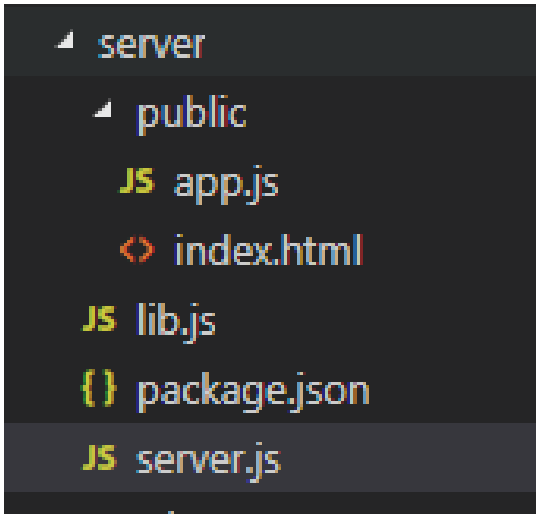
- Arquitectura asíncrono con tolerancia a fallos
- API REST : Servidor node.js + express.js
- Inicialización de conexión a servicios AWS

```
var express = require('express');
var bodyParser = require('body-parser');
var AWS = require('aws-sdk');
var uuid = require('node-uuid');
var multiparty = require('multiparty');

var lib = require('./lib.js');

var db = new AWS.DynamoDB({
  "region": "us-east-1"
});
var sqs = new AWS.SQS({
  "region": "us-east-1"
});
var s3 = new AWS.S3({
  "region": "us-east-1"
});
```

3.7 Desacoplando la arquitectura

- Arquitectura asíncrono con tolerancia a fallos
- API REST : Servidor node.js + express.js
- Creación de servidor express
  - Configurar directorio de paginas estáticas para el frontend

```
var app = express();
app.use(bodyParser.json());
app.use(express.static('public'));
```




```
app.listen(process.env.PORT || 8080, function() {
  console.log("Server started. Open http://localhost:" + (process.env.PORT || 8080) +
  " with browser.");
});
```

3.7 Desacoplando la arquitectura

- Arquitectura asíncrono con tolerancia a fallos
- API REST : Servidor node.js + express.js
- POST /image
  - Genera un ID
  - Y envía el ID con estado creado y fecha
  - Responde con json
- El cliente es el FRONTEND

```
app.post('/image', function(request, response) {
  var id = uuid.v4();
  db.putItem({
    "Item": {
      "id": {
        "S": id
      },
      "version": {
        "N": "0"
      },
      "created": {
        "N": Date.now().toString()
      },
      "state": {
        "S": "created"
      }
    },
    "TableName": "imagery-image",
    "ConditionExpression": "attribute_not_exists(id)"
  Complexity is 4 Everything is cool!
  }, function(err, data) {
    if (err) {
      throw err;
    } else {
      response.json({"id": id, "state": "created"});
    }
  });
});
```

3.7 Desacoplando la arquitectura

- Arquitectura asíncrono con tolerancia a fallos
- API REST : Servidor node.js + express.js
- GET /image/:id
  - Obtiene la imagen
- El cliente es la WEB

```
app.get('/image/:id', function(request, response) {
  Complexity is 4 Everything is cool!
  getImage(request.params.id, function(err, image) {
    if (err) {
      throw err;
    } else {
      response.json(image);
    }
  });
});
```

```
function getImage(id, cb) {
  db.getItem({
    "Key": {
      "id": {
        "S": id
      }
    },
    "TableName": "imagery-image"
  Complexity is 5 Everything is cool!
  }, function(err, data) {
    if (err) {
      cb(err);
    } else {
      if (data.Item) {
        cb(null, lib.mapImage(data.Item));
      } else {
        cb(new Error("image not found"));
      }
    }
  });
}
```

3.7 Desacoplando la arquitectura

- Arquitectura asíncrono con tolerancia a fallos
- API REST : Servidor node.js + express.js
- POST /image/:id/upload
- El cliente de este servicio es el WORKER

```
app.post('/image/:id/upload', function(request, response) {
  Complexity is 5 Everything is cool!
  getImage(request.params.id, function(err, image) {
    if (err) {
      throw err;
    } else {
      var form = new multiparty.Form();
      form.on('part', function(part) {
        uploadImage(image, part, response);
      });
      form.parse(request);
    }
  });
});
```

3.7 Desacoplando la arquitectura

- Arquitectura asíncrono con tolerancia a fallos
- API REST : Servidor node.js + express.js
- POST /image/:id/upload
  - Crea URL S3
  - Sube imagen a S3
  - Una vez subida actualiza la base de datos

```
function uploadImage(image, part, response) {
  var rawS3Key = 'upload/' + image.id + '-' + Date.now();
  s3.putObject({
    "Bucket": process.env.ImageBucket,
    "Key": rawS3Key,
    "Body": part,
    "ContentLength": part.byteCount
  Complexity is 12 You must be kidding
  }, function(err, data) {
    if (err) {
      throw err;
    } else {
      db.updateItem({
        "Key": {
          "id": {
            "S": image.id
          }
        },
        "UpdateExpression": "SET #s=:newState, version=:newVersion, ra
        "ConditionExpression": "attribute_exists(id) AND version=:oldV
        "ExpressionAttributeNames": {
          "#s": "state"
        },
        "ExpressionAttributeValues": {
          ":newState": {
            "S": "uploaded"
          },
          ":oldVersion": {
            "N": image.version.toString()
          },
          ":newVersion": {
            "N": (image.version + 1).toString()
          },
          ":rawS3Key": {
            "S": rawS3Key
          },
          ":stateCreated": {
            "S": "created"
          },
          ":stateUploaded": {
            "S": "uploaded"
```

3.7 Desacoplando la arquitectura

- Arquitectura asíncrono con tolerancia a fallos
- API REST : Servidor node.js + express.js
- POST /image/:id/upload

```
    }, function(err, data) {
      if (err) {
        throw err;
      } else {
        sqs.sendMessage({
          "MessageBody": JSON.stringify({"imageId": image.id, "desi
          "QueueUrl": process.env.ImageQueue,
        Complexity is 4 Everything is cool!
        }, function(err) {
          if (err) {
            throw err;
          } else {
            //response.json(lib.mapImage(data.Attributes));
            response.redirect('/#view=' + image.id);
            response.end();
          }
        });
      }
    });
  }
});
}
```

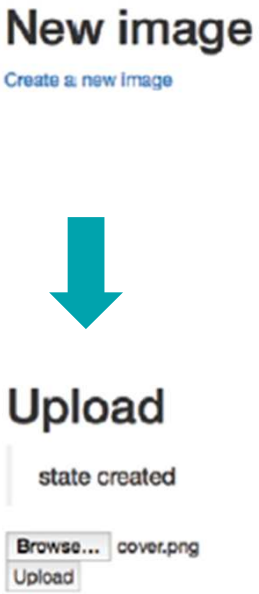
3.7 Desacoplando la arquitectura

- Arquitectura asíncrono con tolerancia a fallos
- Front-end : Index.html , app.js


```
<html lang="en">
  <head>
    <title>Imagery | AWS in Action: chapter 13</title>
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap.
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap-
  </head>
  <body>
    <div id="new" class="collapse">
      <h1>New image</h1>
      <p><a href="#">Create a new image</a></p>
    </div>
    <div id="upload" class="collapse">
      <h1>Upload</h1>
      <p><blockquote></blockquote></p>
      <form method="post" action="/image/blank/upload" enctype="multipart/form-data">
        <input name="file" type="file" accept="image/*">
        <input type="submit" value="Upload">
      </form>
    </div>
    <div id="view" class="collapse">
      <h1>View</h1>
      <p><blockquote></blockquote></p>
      <p><img src=""></p>
      <p><a href="#" class="refresh">Refresh</a> - <a href="/">New image</a></p>
    </div>
    <script src="https://code.jquery.com/jquery-2.1.4.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/js/bootstrap.min.js"></script>
    <script src="app.js"></script>
  </body>
</html>
```

3.7 Desacoplando la arquitectura

- Arquitectura asíncrono con tolerancia a fallos
- Front-end : Index.html , app.js
- Al clickar en el link “Create new Image”
  - Llama al POST /image
    - Si ha ido bien recibe el id de la imagen y especifica vista “update”




```
$('#new a').click(function() {
  $.post('/image', function(data) {
      updateImage(data);
      show('upload');
      window.location.hash = '#upload='+ data.id;
    })
    .fail(function() {
      alert('error');
    });
  return false;
});
```

```
var hash = window.location.hash.substr(1).split("=");
if (window.location.hash.length > 0) {
  if (hash[0] === 'upload' && hash.length === 2) {
    $.get('/image/' + hash[1], function(data) {
      updateImage(data);
      show('upload');
    })
    .fail(function() {
      alert('error');
    });
  } else if (hash[0] === 'view' && hash.length === 2) {
    $.get('/image/' + hash[1], function(data) {
      updateImage(data);
      show('view');
    })
    .fail(function() {
      alert('error');
    });
  } else {
    show('new');
  }
} else {
  show('new');
}
```