



# Micronaut and AWS

Combining Micronaut and AWS to Superpower Your Apps

# Speakers

---

## Stefano Buliani (Amazon Web Services)

Principal Business Development Manager

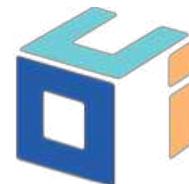
Stefano Buliani works in the Serverless organization at Amazon Web Services helping AWS customers implement new applications that leverage AWS Lambda and Amazon API Gateway. Stefano has been a professional developer for 15 years, primarily focusing on distributed systems and service-oriented architectures using Java, Go, and Rust.



## Sergio del Amo (Object Computing)

Senior Software Engineer, Micronaut and Grails

For the past 6 years, Sergio has been developing Grails applications, Grails Guides, Grails Plugins, and other aspects of the framework. Currently, he is involved with Grails and Micronaut development and supporting our clients on projects. Since early 2015, Sergio has been the author of Groovy Calamari. Most recently, he has taken on the role of conference organizer for Greach.



# Agenda

---

- Java and AWS Lambda
- Distributing a Micronaut FAT JAR to Elastic Beanstalk
- CI Pipeline with CodeCommit - CodeBuild - CodePipeline - S3
- CI/CD Pipeline with CodeCommit - CodeBuild - CodePipeline - ElasticBeanstalk
- Sending Emails with - SES (Simple Email Service)
- Uploading files to S3
- OPEN ID Connect with AWS Cognito



# AWS Lambda

---



<https://aws.amazon.com/blogs/compute/java-11-runtime-now-available-in-aws-lambda/>

We are excited to announce that you can now develop your AWS Lambda functions using the Java 11 runtime. Start using this runtime today by specifying a runtime parameter value of **java11** when creating or updating your Lambda functions.

## SVS403-R - Best Practices for AWS Lambda and Java

---

In this session, we follow a customer's journey as they optimize an AWS Lambda function written in Java to meet their cold start time requirements. We start from a simple yet slow PoC and walk through all of the changes, tricks, and trade-offs we made to reduce the cold start time by over 70%. Finally, we explore new technologies such as Micronaut, Quarkus, and GraalVM that can make Java even faster in Lambda.



The background of this section is a gradient banner for AWS re:Invent 2019, featuring the event name "AWS re:Invent" in large white letters and the dates "DECEMBER 2–6, 2019 | LAS VEGAS, NEVADA" below it.

Monday, December 2nd @ 6:15pm

Wednesday, December 4th @ 10:45am

# Cold Start

---

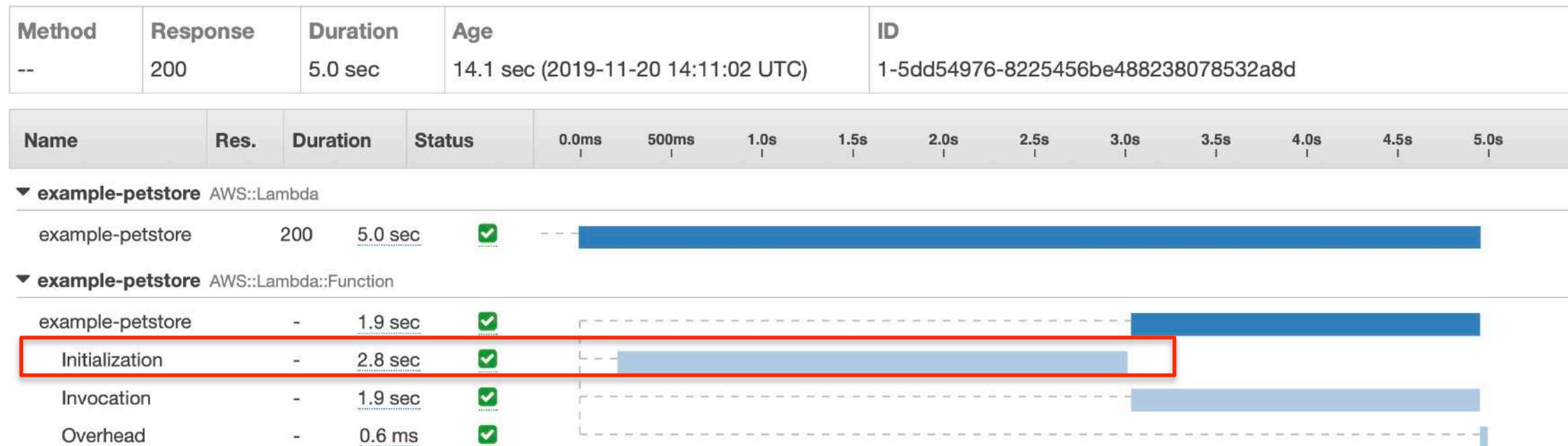
- AWS Lambda—the dynamically scaled and billed-per-execution compute service. Instances of Lambdas are added and removed dynamically.
- When a new instance handles its first request, the response time increases, which is called a **cold start**.
- After that request is processed, the instance stays alive ( $\approx 10$  m) to be reused for subsequent requests.



# The Initialization Step



1. AWS Lambda **starts a JVM**.
2. Java runtime **loads and initializes** handler class.
3. Lambda calls the handler method.

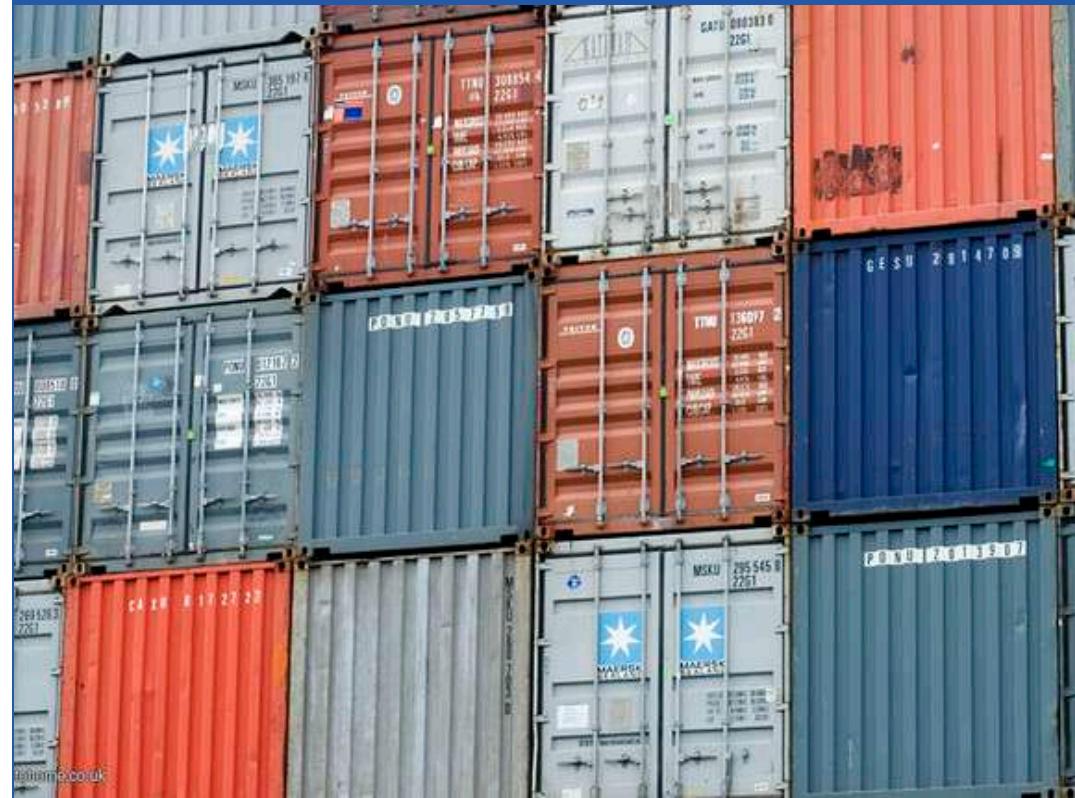


- Increasing the amount of memory allocated to the Lambda function helps.
  - AWS Lambda allocates CPU cycles to a function based on the amount of memory configured.
  - More memory = Higher CPU.
  -  More memory, More Cost.

# Cold Start Problem

---

- Cold start is not just relevant to Serverless, **Containers** also need to start very fast.



- **Front-load classes** during initialization.
- Try to avoid reflection like the plague.
- Switch to the **AWS SDK for Java v2**. Smaller footprint and more modular.
- Provide all known values to avoid auto-discovery.

- Each class is more bytecode to load, I/O access - less is more.
- Remove unused dependencies.
- If you need to, prime dependencies.

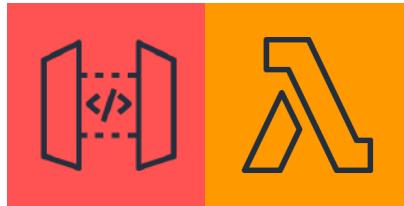
- GraalVM is a universal virtual machine for running applications written in Javascript, Python, Ruby, R, JVM-based languages like Java, Scala, Groovy, Kotlin, Clojure and LLVM languages such as C and C++.
- Compile Java programs to **native executables**.

<https://graalvm.org>



# GraalVM™

# AWS API Gateway Lambda Proxy Integration



In Lambda proxy integration, when a client submits an API request, API Gateway passes to the integrated Lambda function the raw request as-is.

This request data includes the request headers, query string parameters, URL path variables, payload, and API configuration data. The configuration data can include current deployment stage name, stage variables, user identity, or authorization context (if any).



## AWS API Gateway Support

```
mn create-app my-app --features aws-api-gateway
```

```
io.micronaut.aws:micronaut-function-aws-api-proxy
```



A runtime is a program that runs a Lambda function's handler method when the function is invoked. You can include a runtime in your function's deployment package in the form of an executable file named bootstrap.



## Custom GraalVM Native Runtimes

```
mn create-app my-app --features aws-api-gateway-graal
```

```
io.micronaut.aws:micronaut-function-aws-custom-runtime
```



# Code Samples

---



- Pet Store with Micronaut Spring.

<https://github.com/awslabs/aws-serverless-java-container/tree/master/samples/micronaut/pet-store>

- Pet Store Micronaut micronaut-function-aws-api-proxy

<https://github.com/micronaut-guides/micronaut-api-gateway-proxy-lambda>

- Pet Store Micronaut micronaut-function-aws-custom-runtime

<https://github.com/micronaut-guides/micronaut-api-gateway-proxy-lambda-custom-runtime>



## SINGLE FUNCTION

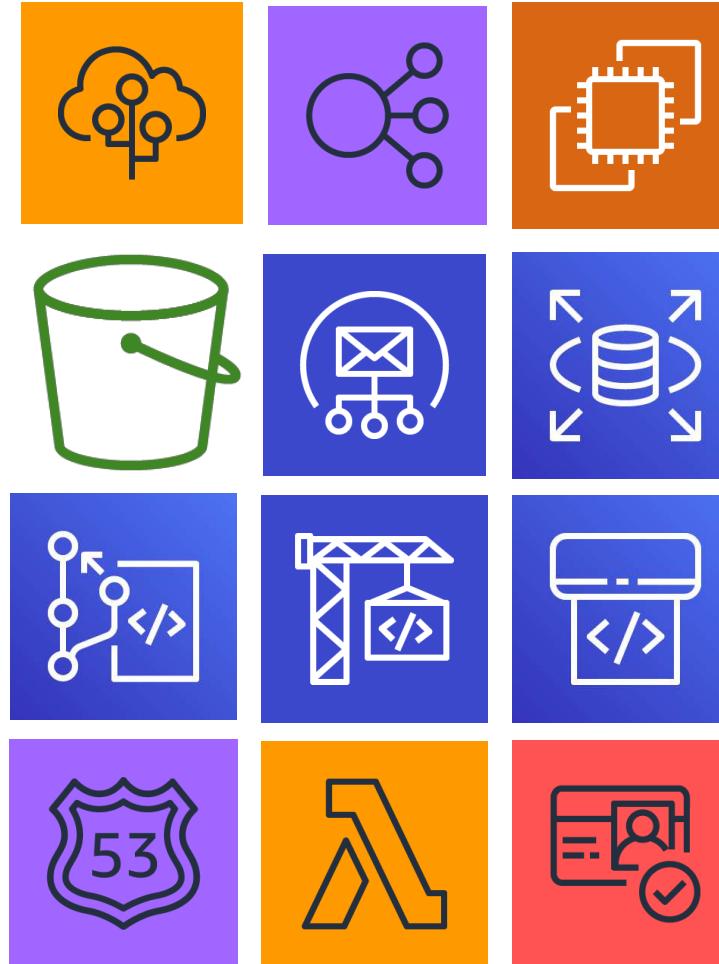
- \* `@FunctionBean`
- \* `FunctionInitializer`



## AWS API Gateway Lambda Proxy Integration

- \* `micronaut-function-aws-api-proxy`
- \* `micronaut-function-aws-custom-runtime`

# Amazon Web Services



# AWS Elastic Beanstalk

---



# AWS Elastic Beanstalk



The screenshot shows the AWS Elastic Beanstalk interface. At the top, there's a navigation bar with the AWS logo, 'Services' dropdown, 'Resource Groups' dropdown, a bell icon, 'Softamo S.L.' dropdown, 'Paris' dropdown, and 'Support' dropdown. Below the navigation is a search bar with the placeholder 'Search Elastic Beanstalk' and a 'Create New Application' button. A red box highlights the 'Elastic Beanstalk' link in the navigation bar and the 'Create New Application' button. The main content area has a title 'Welcome to AWS Elastic Beanstalk' and a sub-section 'Create New Application'. It contains fields for 'Application Name' (set to 'micronautguide') and 'Description', both with character limits. A large text block provides instructions on creating an application. At the bottom are 'Cancel' and 'Create' buttons, with the 'Create' button highlighted by a red box.

Welcome to AWS Elastic Beanstalk

Create New Application

Application Name  Maximum length of 100 characters, not including forward slash (/).

Description  Maximum length of 200 characters.

Cancel Create

application quickly and easily. Let us do the heavy lifting so you can...  
source bundle and then create a new application. If you're using Git...  
Getting Started with the EB CLI.  
e, select a platform and click **Create app**.  
stalk to administer AWS resources and necessary permissions on

Elastic Beanstalk

Services Resource Groups

Softamo S.L. Paris Support

Create New Application

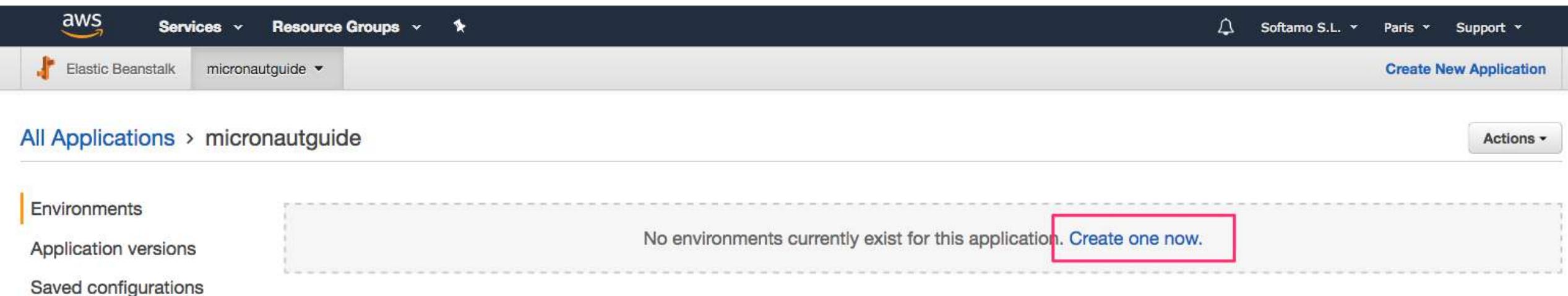
Sales   Services ▾   Resource Groups ▾   ★

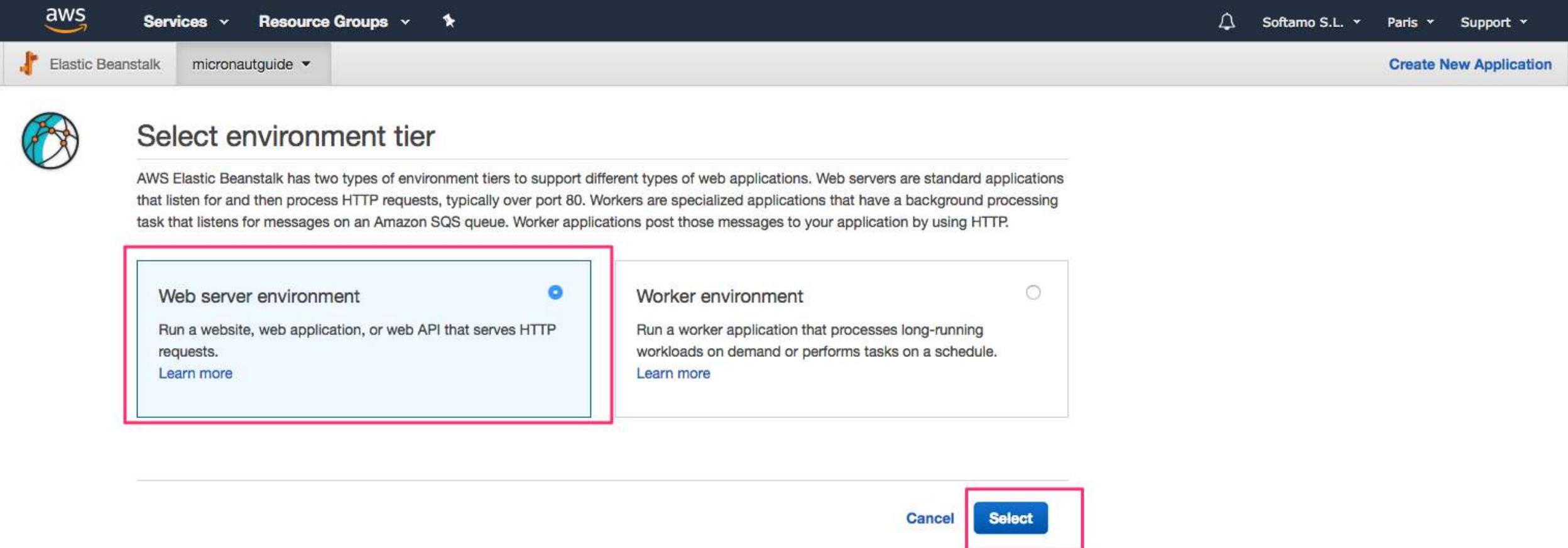
Elastic Beanstalk   micronautguide   Create New Application

All Applications > micronautguide   Actions ▾

Environments   Application versions   Saved configurations

No environments currently exist for this application. [Create one now.](#)





The screenshot shows the 'Select environment tier' step in the AWS Elastic Beanstalk application creation wizard. It offers two choices:

- Web server environment** (selected): Run a website, web application, or web API that serves HTTP requests. [Learn more](#)
- Worker environment**: Run a worker application that processes long-running workloads on demand or performs tasks on a schedule. [Learn more](#)

At the bottom are 'Cancel' and 'Select' buttons, with 'Select' being the primary action.



## Create a web server environment

Launch an environment with a sample application or your own code. By creating an environment, you allow AWS Elastic Beanstalk to manage AWS resources and permissions on your behalf. [Learn more](#)

### Environment information

Choose the name, subdomain, and description for your environment. These cannot be changed later.

**Application name** micronautguide

**Environment name**

**Domain** Leave blank for autogenerated value  [Check availability](#)

**Description**

### Base configuration

**Platform**  Preconfigured platform

Platforms published and maintained by AWS Elastic Beanstalk.

Custom platform

Platforms created and owned by you. [Learn more](#)

**Application code**  Sample application

Get started right away with sample code.

Existing version

Application versions that you have uploaded for **micronautguide**.

Upload your code

Upload a source bundle from your computer or copy one from Amazon S3.

ZIP or WAR

[Cancel](#)

[Configure more options](#)

[Create environment](#)

AWS Services Resource Groups

Elastic Beanstalk micronautguide

## Create a web server environment

Launch an environment with a sample application or your own code. By creating an environment, you AWS resources and permissions on your behalf. Learn more

Environment information

Choose the name, subdomain, and description for your environment. These cannot be changed later.

Application name: micronautguide

Environment name: Micronautguide-env.

Domain: Leave blank for autogenerated value eu-west-3.elasticbeanstalk.com

Description:

Base configuration

Platform: Preconfigured platform

Java

Custom platform: Choose a custom platform --

Application code: Sample application

Get started right away with sample code.

Existing version: Choose a version --

Upload your code: Upload a source bundle from your computer or copy one from Amazon S3.

ZIP or WAR

Cancel Configure next

### Upload your code

Upload a source bundle from your computer or copy one from Amazon S3.

Source code origin: Local file (Maximum size 512 MB)

Browse... complete-0.1-all.jar

Public S3 URL: https://s3.eu-west-3.amazonaws.com

Version label: micronautguide-source

Unique name for this version of your application code.

./gradlew shadowJar

Cancel Upload





## Configure Micronautguide-env

Start from a preset that matches your use case or choose *Custom configuration* to unset recommended values and use the service's default values.

- Configuration presets  Low cost (*Free Tier eligible*)  
 High availability  
 Custom configuration

Platform Java 8 running on 64bit Amazon Linux/2.7.5 [Change platform configuration](#)

<b>Software</b> Rotate logs: disabled (default) Log streaming: disabled (default) Environment properties: 5 GRADLE_HOME, JAVA_HOME, M2, M2_HOME, MICRONAUT_SERVER_PORT	<b>Instances</b> EC2 Instance type: t2.micro EC2 image ID: ami-00121fcfedf1bd8106 Root volume type: container default Root volume size (GB): container default Root volume IOPS: container default Security groups: <i>none</i>	<b>Capacity</b> Environment type: single instance
<a href="#">Modify</a>	<a href="#">Modify</a>	<a href="#">Modify</a>
<b>Load balancer</b> <i>This configuration does not contain a load balancer.</i>	<b>Rolling updates and deployments</b> Deployment policy: All at once Rolling updates: disabled	<b>Security</b> Service role: aws-elasticbeanstalk-service-role Virtual machine key pair: -- Virtual machine instance profile: aws-elasticbeanstalk-ec2-role
	<a href="#">Modify</a>	<a href="#">Modify</a>
<b>Monitoring</b>	<b>Notifications</b>	<b>Network</b>



## Modify software

## Container Options

The following settings control container behavior and let you pass key-value pairs in as OS environment variables. Learn more

## S3 log storage

Configure the instances in your environment to upload rotated logs to Amazon S3. Learn more

Rotate logs  Enabled (Standard S3 charges apply.)

## Instance log streaming to CloudWatch Logs

Configure the instances in your environment to stream logs to CloudWatch Logs. You can set the retention to up to ten years and configure Elastic Beanstalk to delete the logs when you terminate your environment.

Log streaming  Enabled (Standard CloudWatch charges apply.)

Retention  days

Lifecycle

## Environment properties

The following properties are passed in the application as environment properties. Learn more

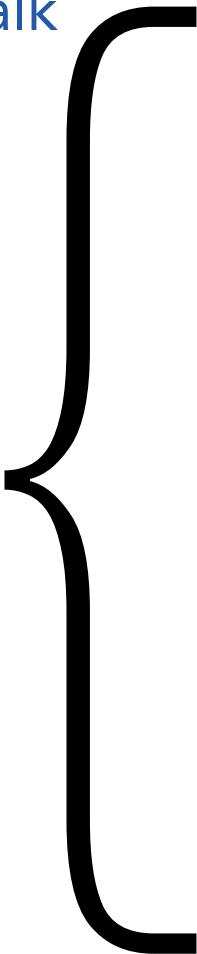
Name	Value
GRADLE_HOME	/usr/local/gradle
JAVA_HOME	/usr/lib/jvm/java
M2	/usr/local/apache-maven/bin
M2_HOME	/usr/local/apache-maven
MICRONAUT_SERVER_PORT	5000

Cancel

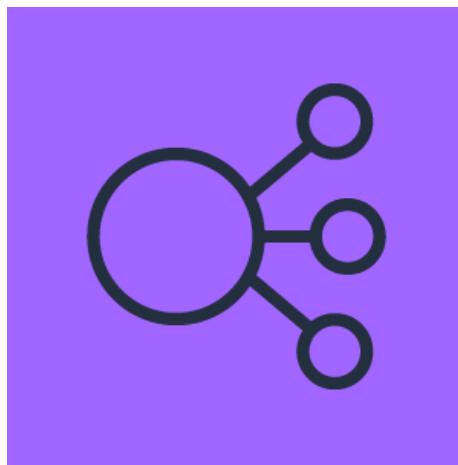
Save



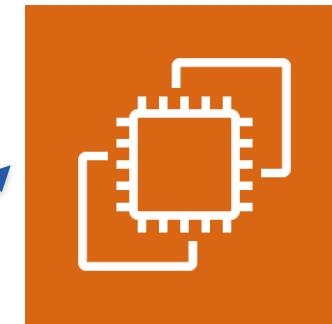
## AWS Elastic Beanstalk



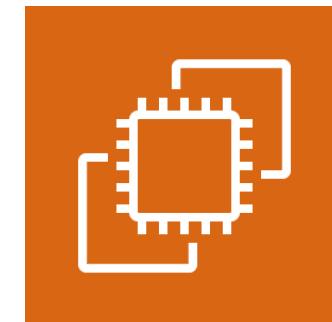
## Load Balancer



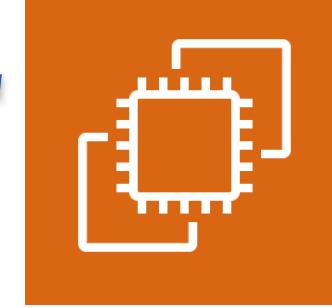
EC2



EC2



EC2



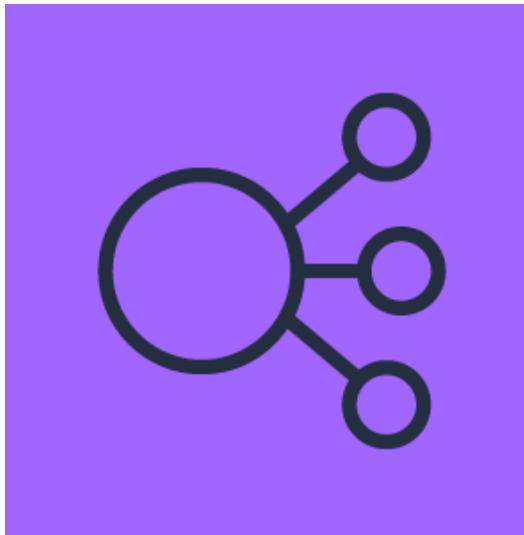
RDS



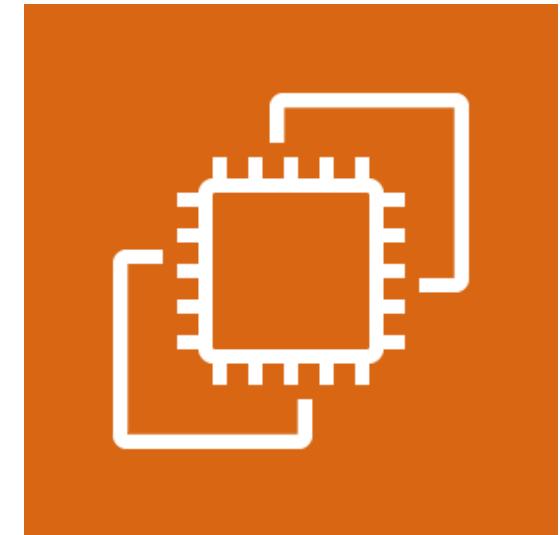
# Elastic Load Balancer



124.12.3.231



10.0.0.23



10.0.0.23

X-Forwarded-For: 124.12.3.231

```
import io.micronaut.http.*;  
import io.micronaut.http.annotation.*;  
import io.micronaut.http.server.util.HttpHostResolver;  
  
{@Controller  
public class HomeController {  
  
    private final HttpHostResolver httpHostResolver;  
  
    public HomeController(HttpHostResolver httpHostResolver) {  
        this.httpHostResolver = httpHostResolver;  
    }  
  
    @Produces(MediaType.TEXT_PLAIN)  
    @Get  
    public String index(HttpServletRequest request) {  
        return httpHostResolver.resolve(request);  
    }  
}}
```



If a health check URL is configured, Elastic Load Balancing expects a GET request that it sends to return a response of 200 OK. The application fails the health check if it fails to respond within 5 seconds or if it responds with any other HTTP status code. After 5 consecutive health check failures, Elastic Load Balancing takes the instance out of service.

## Modify monitoring

### Health check

Set the path (relative to the root of your application) to which the load balancer sends health check requests.

**Health check path**

/health

# AWS Elastic Beanstalk - Health Check

---



```
src/main/resources/application.yml
```

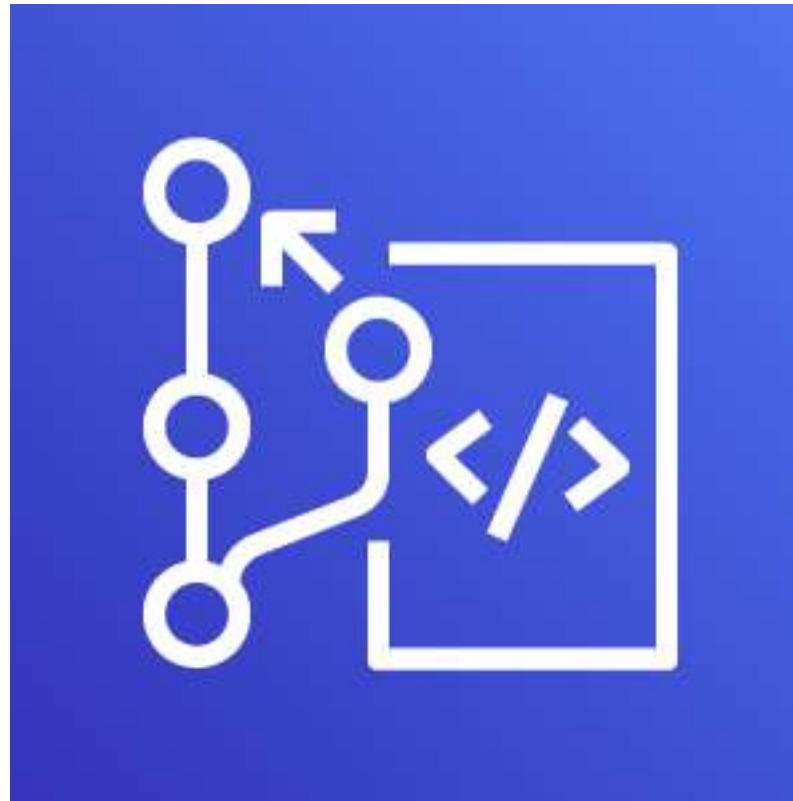
```
endpoints:  
  health:  
    enabled: true  
    sensitive: false
```

```
build.gradle
```

```
...  
dependencies {  
  ...  
  implementation "io.micronaut:micronaut-management"  
}  
...
```

# AWS CodeCommit

---



```
$ git remote -v
origin ssh://git-codecommit.eu-west-1.amazonaws.com/v1/repos/nautcast-webapp (fetch)
origin ssh://git-codecommit.eu-west-1.amazonaws.com/v1/repos/nautcast-webapp (push)
```

```
$ mn --version
Resolving dependencies..
| Micronaut Version: 1.2.6
| JVM Version: 1.8.0_181
```

```
$ mn create-app com.nautcast --inplace
| Generating Java project...
| Application created at /Users/sdelamo/Developer/softamo/nautcast-webapp
```

```
$ git add --all
$ git commit -m "Initial commit"
[master (root-commit) be2c0a8] Initial commit
 13 files changed, 371 insertions(+)
 create mode 100644 .gitignore
```

...

```
$ git push origin master
```

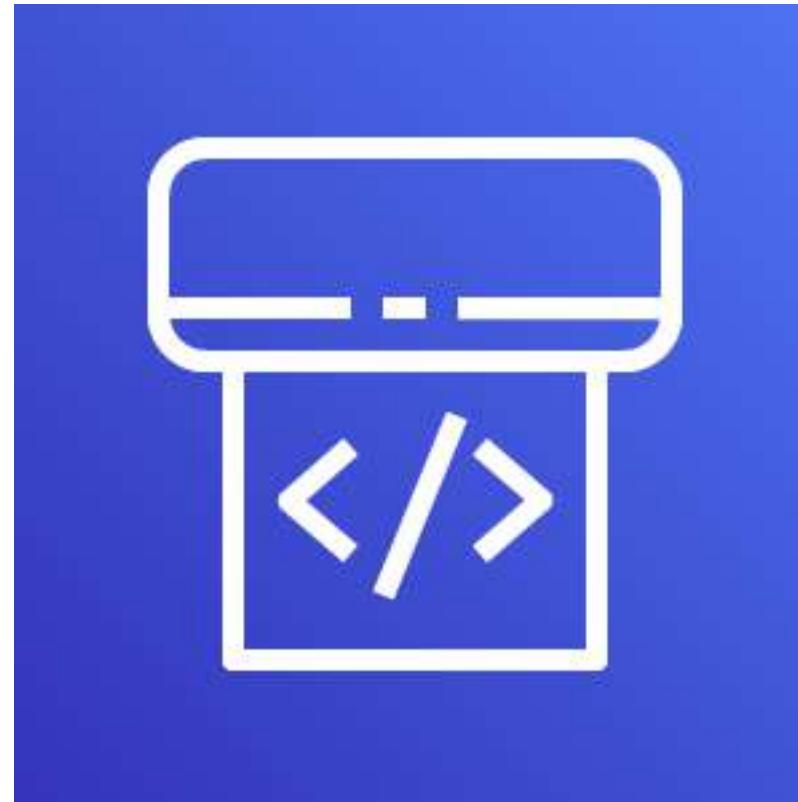
# AWS CodeBuild

---

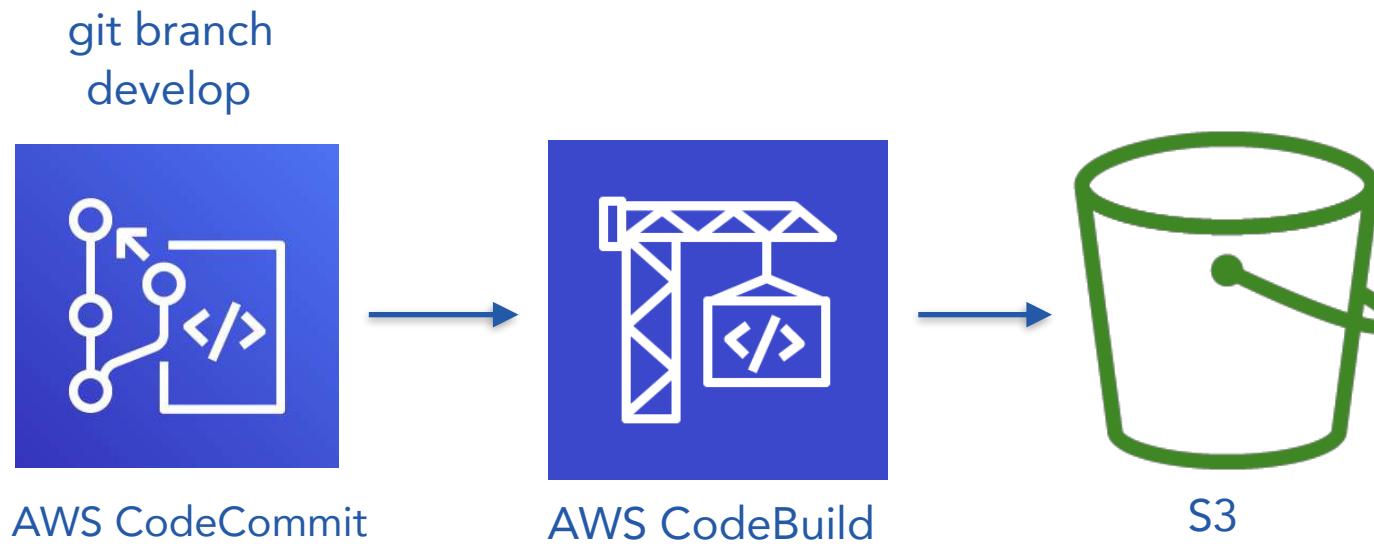


# AWS Pipeline

---



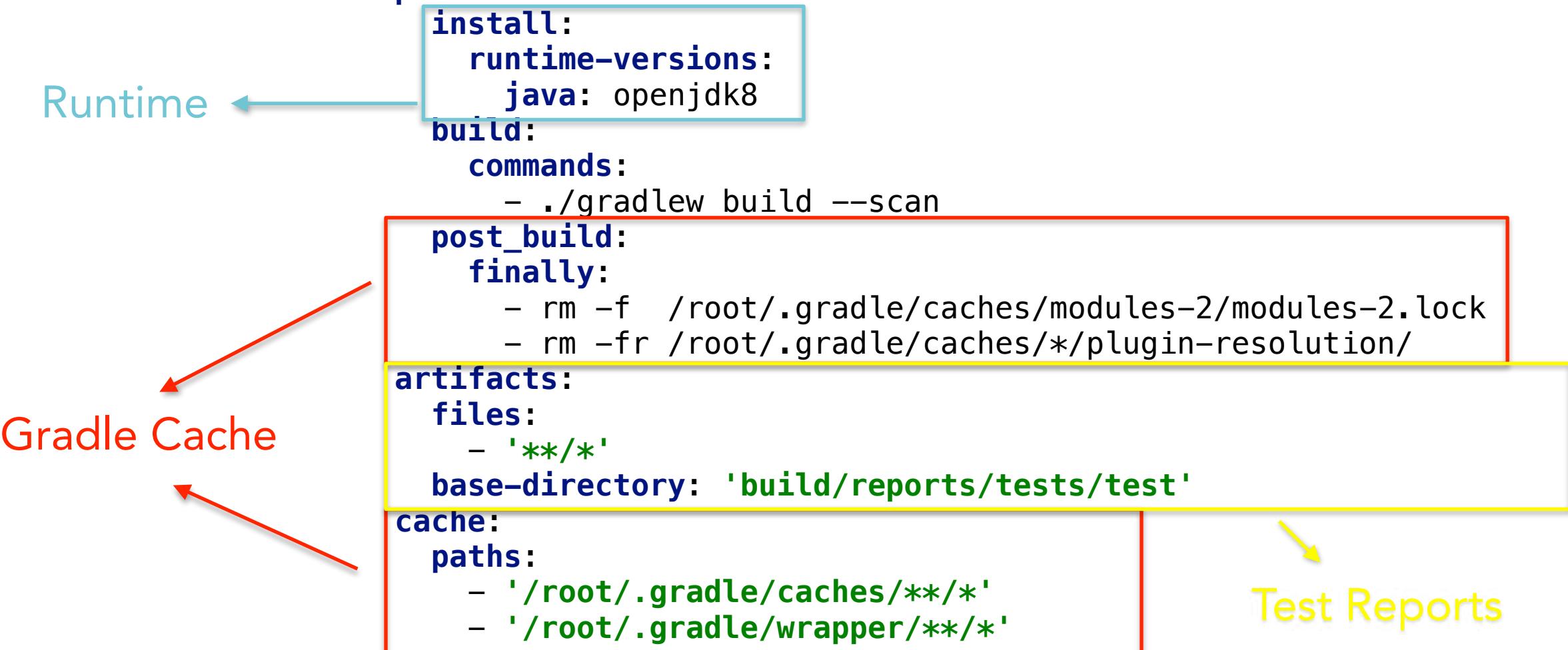
# AWS Pipeline



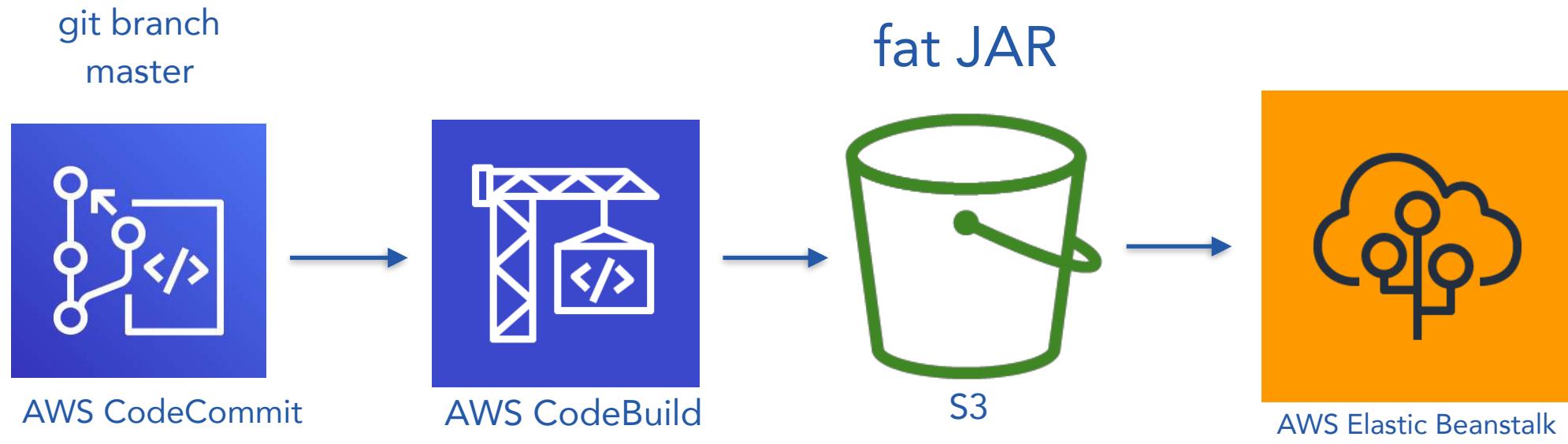
build cache  
test reports



AWS Pipeline

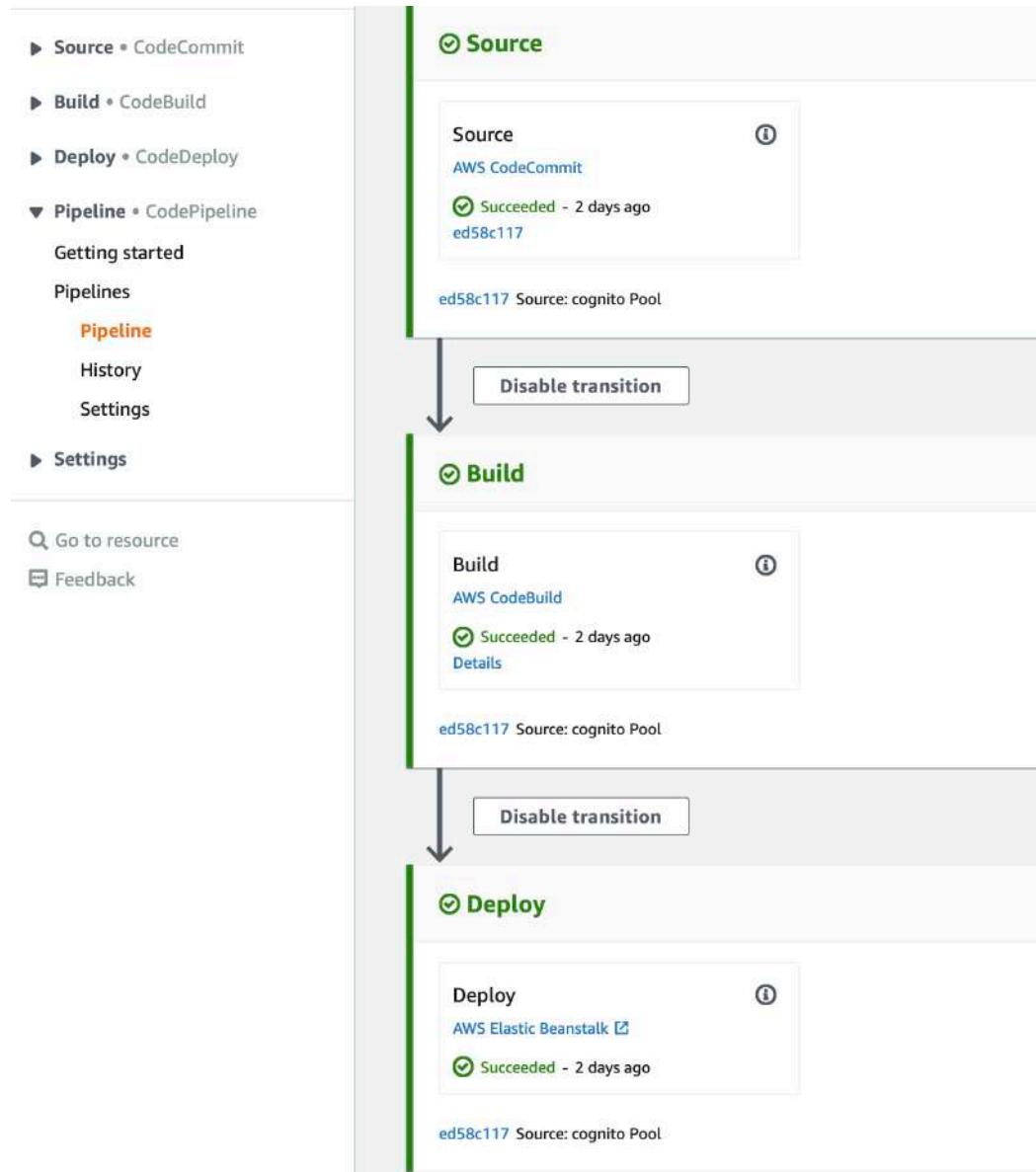


# AWS Pipeline



AWS Pipeline

# AWS Pipeline



FAT JAR

```
buildspec.yml
version: 0.2
phases:
  install:
    runtime-versions:
      java: openjdk8
  build:
    commands:
      - ./gradlew shadowJar --scan
  post_build:
    finally:
      - rm -f /root/.gradle/caches/modules-2/modules-2.lock
      - rm -fr /root/.gradle/caches/*plugin-resolution/
artifacts:
  files:
    - '*-all.jar'
  base-directory: 'build/libs'
cache:
  paths:
    - '/root/.gradle/caches/**/*'
    - '/root/.gradle/wrapper/**/*'
```

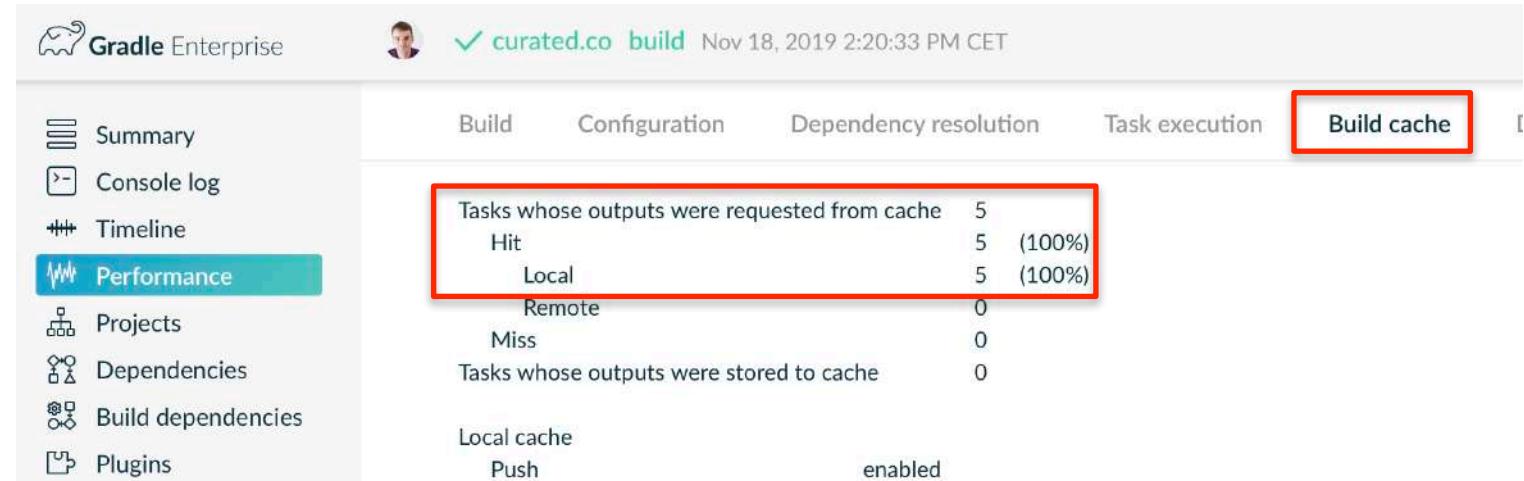
myapp-0.1-all.jar

# AWS CodeBuild - Gradle Cache



```
gradle.properties  
org.gradle.caching=true
```

```
buildspec.yml  
version: 0.2  
phases:  
...  
post_build:  
  finally:  
    - rm -f /root/.gradle/caches/modules-2/modules-2.lock  
    - rm -fr /root/.gradle/caches/*plugin-resolution/  
...  
cache:  
  paths:  
    - '/root/.gradle/caches/**/*'  
    - '/root/.gradle/wrapper/**/*'
```



# AWS CodeBuild - Build Scans

## build.gradle

```
plugins {  
    ...  
    id "com.gradle.build-scan" version "3.0"  
}  
...  
  
buildScan {  
    termsOfServiceUrl = 'https://gradle.com/terms-of-service'  
    termsOfServiceAgree = 'yes'  
}
```

## buildspec.yml

```
version: 0.2
```

### phases:

```
...
```

### build:

#### commands:

```
- ./gradlew build --scan
```



```
97 > Task :test NO-SOURCE  
98 > Task :check UP-TO-DATE  
99 > Task :build  
100 > Task :guides:assemble UP-TO-DATE  
101 > Task :guides:check UP-TO-DATE  
102 > Task :guides:build UP-TO-DATE  
103 > Task :curated:compileJava FROM-CACHE  
104 > Task :curated:compileGroovy NO-SOURCE  
105 > Task :curated:processResources NO-SOURCE  
106 > Task :curated:classes UP-TO-DATE  
107 > Task :curated:minpom FROM-CACHE  
108 > Task :curated:jar  
109 > Task :curated:javadoc FROM-CACHE  
110 > Task :curated:javadocJar  
111 > Task :curated:sourceJar  
112 > Task :curated:assemble  
113 > Task :curated:licenseGradle SKIPPED  
114 > Task :curated:licenseMain UP-TO-DATE  
115 > Task :curated:licenseTest UP-TO-DATE  
116 > Task :curated:license UP-TO-DATE  
117 > Task :curated:compileTestJava NO-SOURCE  
118 > Task :curated:compileTestGroovy FROM-CACHE  
119 > Task :curated:processTestResources  
120 > Task :curated:testClasses  
121 > Task :curated:test FROM-CACHE  
122 > Task :curated:check UP-TO-DATE  
123 > Task :curated:build  
124  
125 BUILD SUCCESSFUL in 14s  
126 13 actionable tasks: 6 executed, 5 from cache, 2 up-to-date  
127  
128 Publishing build scan...  
129 https://gradle.com/s/jd275aaqjo2ey  
130  
131  
132 [Container] 2019/11/18 14:01:22 Phase complete: BUILD State:  
133 [Container] 2019/11/18 14:01:22 Phase context status code:  
134 [Container] 2019/11/18 14:01:22 Entering phase POST_BUILD  
135 [Container] 2019/11/18 14:01:22 Running command rm -f /root  
136  
137 [Container] 2019/11/18 14:01:22 Running command rm -fr /root  
138  
139 [Container] 2019/11/18 14:01:22 Uploading S3 cache...  
140 [Container] 2019/11/18 14:01:22 POST BUILD
```

# AWS Simple Email Service (SES)

---



# AWS Simple Email Service (SES)

## **build.gradle**

```
...
dependencies {
    ...
    implementation "software.amazon.awssdk:ses:2.10.16"
    ...
}
```

```
package example;

import javax.annotation.NonNull;

public interface SesConfiguration {

    @NonNull
    String getRegion();

    @NonNull
    String getSource();

    @NonNull
    String getDestination();
}
```

# AWS Simple Email Service (SES)



```
package example;

import io.micronaut.context.annotation.ConfigurationProperties;
import javax.annotation.Nonnull;
import javax.validation.constraints.NotBlank;

@ConfigurationProperties(SesConfigurationProperties.PREFIX)
public class SesConfigurationProperties implements SesConfiguration {

    @SuppressWarnings("WeakerAccess")
    public static final String PREFIX = "aws.ses";

    @Nonnull
    @NotBlank
    private String region;

    @Nonnull
    @NotBlank
    private String source;

    @Nonnull
    @NotBlank
    private String destination;

    // GETTERS AND SETTERS
}
```

# AWS Simple Email Service (SES)

---



**src/main/resources/application.yml**

```
aws:  
  ses:  
    region: 'eu-west-1'  
    destination: 'delamos@objectcomputing.com'  
    source: 'sergio.delamo@softamo.com'
```

# AWS Simple Email Service (SES)



```
@Singleton
public class EmailService {
    private static final Logger LOG = LoggerFactory.getLogger(EmailService.class);

    private final SesClient ses;
    private final String source;
    private final String destination;

    public EmailService(SesConfiguration sesConfiguration) {
        this.source = sesConfiguration.getSource();
        this.destination = sesConfiguration.getDestination();
        this.ses = SesClient.builder().region(Region.of(sesConfiguration.getRegion())).build();
    }

    public void sendEmail(String subject, String body) {
        SendEmailRequest sendEmailRequest = SendEmailRequest.builder()
            .destination(Destination.builder().toAddresses(destination).build())
            .source(source)
            .message(Message.builder().subject(Content.builder().data(subject).build())
                .body(Body.builder().text(Content.builder().data(body).build()).build())
            .build())
            .build();
        SendEmailResponse response =ses.sendEmail(sendEmailRequest);
        if (LOG.isInfoEnabled()) {
            LOG.info("Sent email with id: {}", response.messageId());
        }
    }
}
```

# S3 - Simple Cloud Storage

---



# S3 - Simple Cloud Storage

## **build.gradle**

```
...
dependencies {
    ...
    implementation "software.amazon.awssdk:s3:2.10.14"
    ...
}
```



```
package example;

import javax.annotation.NonNull;

public interface S3Configuration {

    @NonNull
    String getBucket();

    @NonNull
    String getRegion();
}
```

# S3 - Simple Cloud Storage

---



```
package example;

import io.micronaut.context.annotation.ConfigurationProperties;
import javax.annotation.Nonnull;
import javax.validation.constraints.NotBlank;

@ConfigurationProperties(S3ConfigurationProperties.PREFIX)
public class S3ConfigurationProperties implements S3Configuration {

    @SuppressWarnings("WeakerAccess")
    public static final String PREFIX = "aws.s3";

    @Nonnull
    @NotBlank
    private String bucket;

    @Nonnull
    @NotBlank
    private String region;

    // GETTERS AND SETTERS
}
```

# S3 - Simple Cloud Storage

---



**src/main/resources/application.yml**

```
aws:  
  s3:  
    region: 'eu-west-1'  
    bucket: 'mybucket'
```

# S3 - Simple Cloud Storage



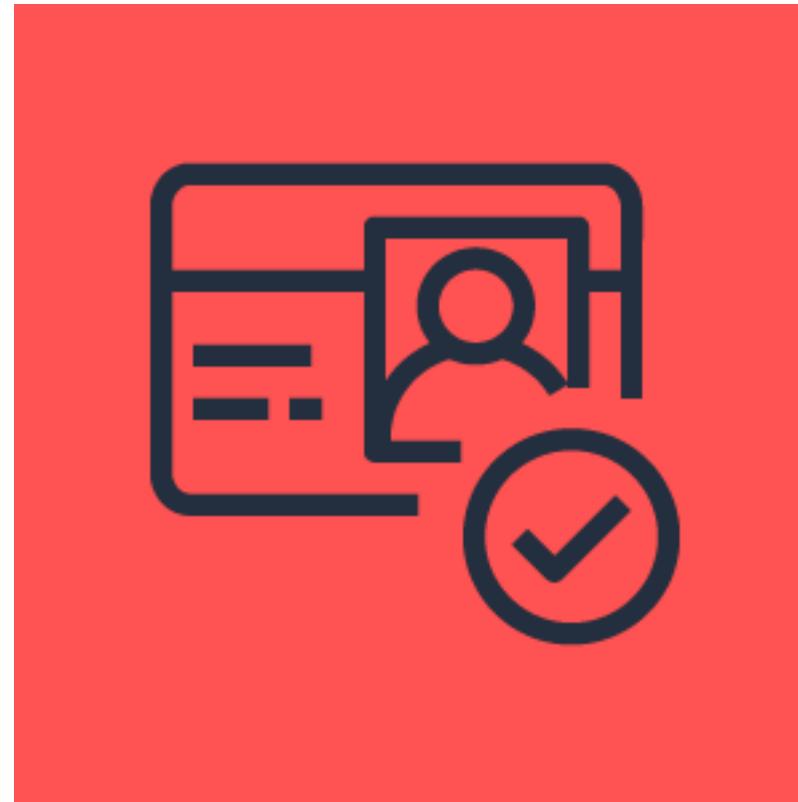
```
@Singleton
public class UploadFileService {
    private static final Logger LOG = LoggerFactory.getLogger(UploadFileService);

    private final S3Client s3;
    private final String bucket;

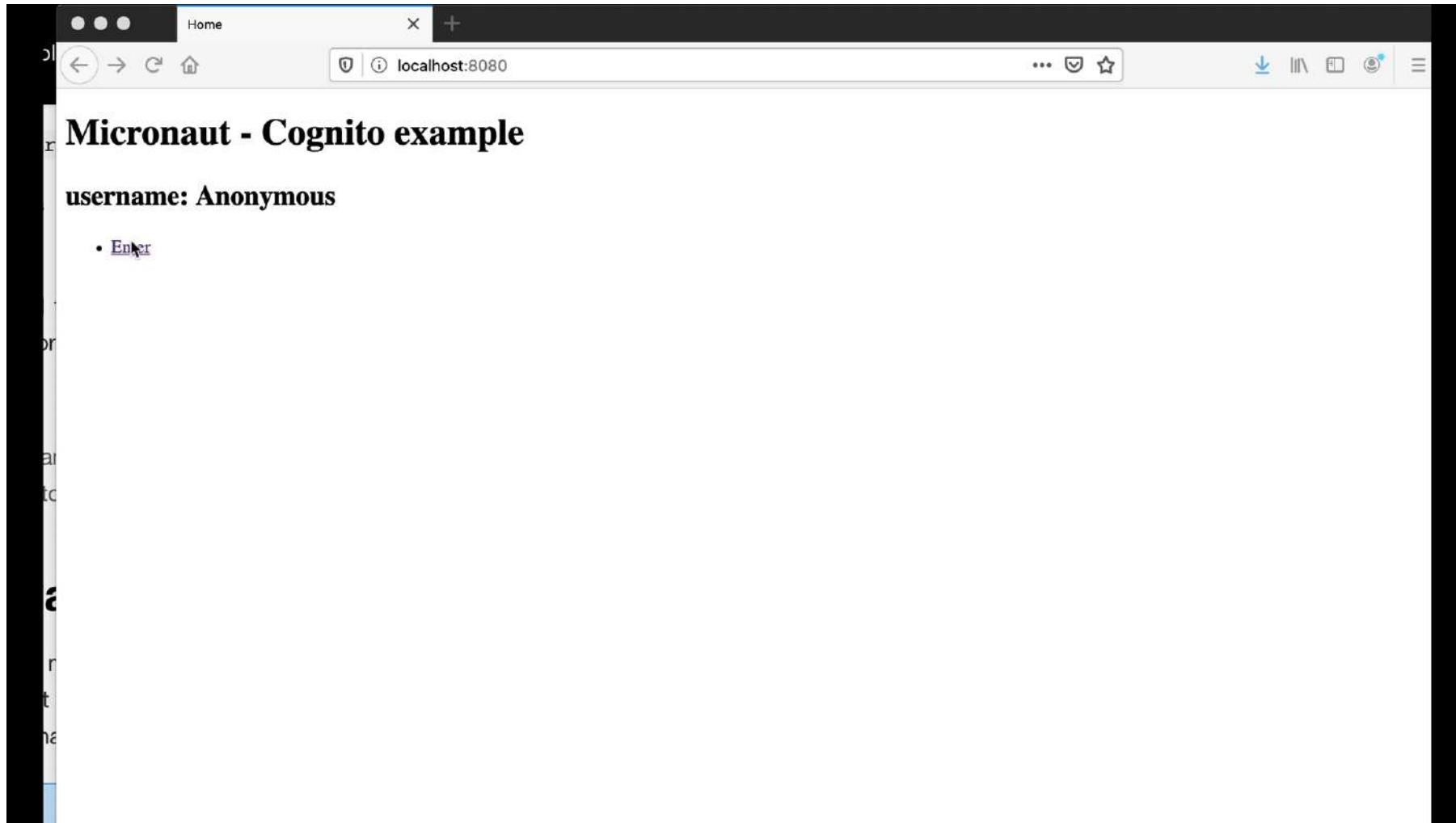
    public UploadFileService(S3Configuration bucket) {
        this.bucket = s3Configuration.getBucket();
        this.s3 = S3Client.builder().region(Region.of(s3Configuration.getRegion())).build();
    }
    public void saveFile(CompletedFileUpload fileUpload) {
        if (LOG.isInfoEnabled()) {
            LOG.info("Attempting to save excel in S3 bucket {}", bucket);
        }
        PutObjectRequest.Builder builder = PutObjectRequest.builder()
            .bucket(bucket)
            .acl(ObjectCannedACL.PUBLIC_READ)
            .key(fileUpload.getFilename());
        try {
            s3.putObject(builder.build(), RequestBody.fromBytes(fileUpload.getBytes()));
        } catch (IOException e) {
            if (LOG.isErrorEnabled()) {
                LOG.error("IO exception grabbing the bytes");
            }
        }
    }
}
```

# Amazon Cognito

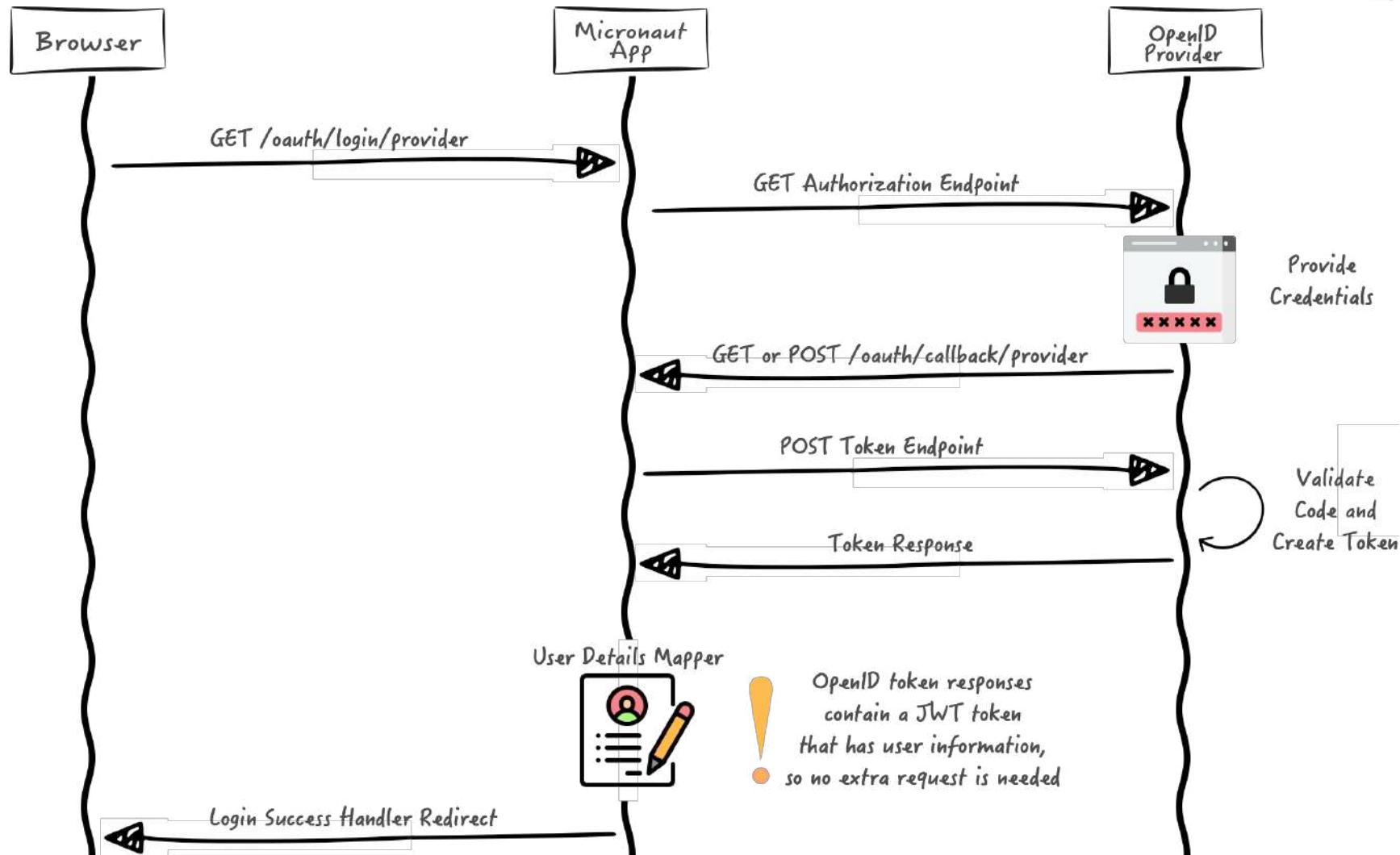
---



# Cognito example



# Amazon Cognito



## build.gradle

```
...
dependencies {
    ...
    annotationProcessor "io.micronaut:micronaut-security"
    implementation 'io.micronaut:micronaut-security-jwt'
    implementation 'io.micronaut.configuration:micronaut-security-oauth2'
}

...
```

```
src/main/resources/application.yml
micronaut:
  security:
    enabled: true
    token:
      jwt:
        enabled: true
        cookie:
          enabled: true
    signatures:
      secret:
        generator:
          secret: pleaseChangeThisSecretForANewOne
  endpoints:
    logout:
      enabled: true
      get-allowed: true
```

# Amazon Cognito



General settings

- Users and groups
- Attributes
- Policies
- MFA and verifications
- Advanced security
- Message customizations
- Tags
- Devices
- App clients
- Triggers
- Analytics

App integration

- App client settings
- Domain name
- UI customization
- Resource servers

Federation

- Identity providers
- Attribute mapping

**Pool Id** eu-west-1\_n8P [REDACTED]

**Pool ARN** arn:aws:cognito-idp:eu-west-1:040181416768:userpool/eu-west-1\_n8PcsxhHq

**Estimated number of users** 1

**Required attributes** email

**Alias attributes** none

**Username attributes** email

**Custom attributes** Choose custom attributes...

**Minimum password length** 8

**Password policy** uppercase letters, lowercase letters, special characters, numbers

**User sign ups allowed?** Users can sign themselves up

**FROM email address** arn:aws:ses:eu-west-1:040181416768:identity/sergio.delamo@softamo.com

**Email Delivery through Amazon SES** No

*Note: You have chosen to have Cognito send emails on your behalf. Best practices suggest that customers send emails through A SES for production User Pools due to a daily email limit. [Learn more about email best practices](#).*

**MFA** optional

**Verifications** Email

**Advanced security** [Enable advanced security...](#)

General settings

Users and groups

Attributes

Policies

MFA and verifications

Advanced security

Message customizations

Tags

Devices

**App clients**

Triggers

Analytics

App integration

App client settings

Domain name

UI customization

Resource servers

Federation

Identity providers

Attribute mapping

## Which app clients will have access to this user pool?

The app clients that you add below will be given a unique ID and an optional secret key to access this user pool.

nautcast

**App client id**  
71ml[REDACTED]

**App client secret**  
207i1[REDACTED]c

**Refresh token expiration (days)**  
30

**Auth Flows Configuration**

Enable username password auth for admin APIs for authentication (ALLOW\_ADMIN\_USER\_PASSWORD\_AUTH) [Learn more.](#)

Enable lambda trigger based custom authentication (ALLOW\_CUSTOM\_AUTH) [Learn more.](#)

Enable username password based authentication (ALLOW\_USER\_PASSWORD\_AUTH) [Learn more.](#)

Enable SRP (secure remote password) protocol based authentication (ALLOW\_USER\_SRP\_AUTH) [Learn more.](#)

Enable refresh token based authentication (ALLOW\_REFRESH\_TOKEN\_AUTH) [Learn more.](#)

**Prevent User Existence Errors** [Learn more.](#)

Legacy

Enabled (Recommended)

[Set attribute read and write permissions](#)

# Amazon Cognito

---



```
src/main/resources/application.yml
micronaut:
  security:
oAuth2:
  enabled: true
  clients:
    cognito:
      client-id: '${OAUTH_CLIENT_ID}'
      client-secret: '${OAUTH_CLIENT_SECRET}'
      openid:
        issuer: 'https://cognito-idp.eu-west-1.amazonaws.com/${OAUTH_POOL_ID}'
```

## build.gradle

```
...
dependencies {
    ...
    implementation "io.micronaut:micronaut-views"
    runtime "org.thymeleaf:thymeleaf:3.0.11.RELEASE"
}
...
```

```
package example;
import io.micronaut.http.annotation.*;
import io.micronaut.views.View;
import javax.annotation.security.PermitAll;
import java.util.*;

@Controller
public class HomeController {

    @PermitAll
    @View("home")
    @Get
    public Map<String, Object> index() {
        return new HashMap<>();
    }
}
```

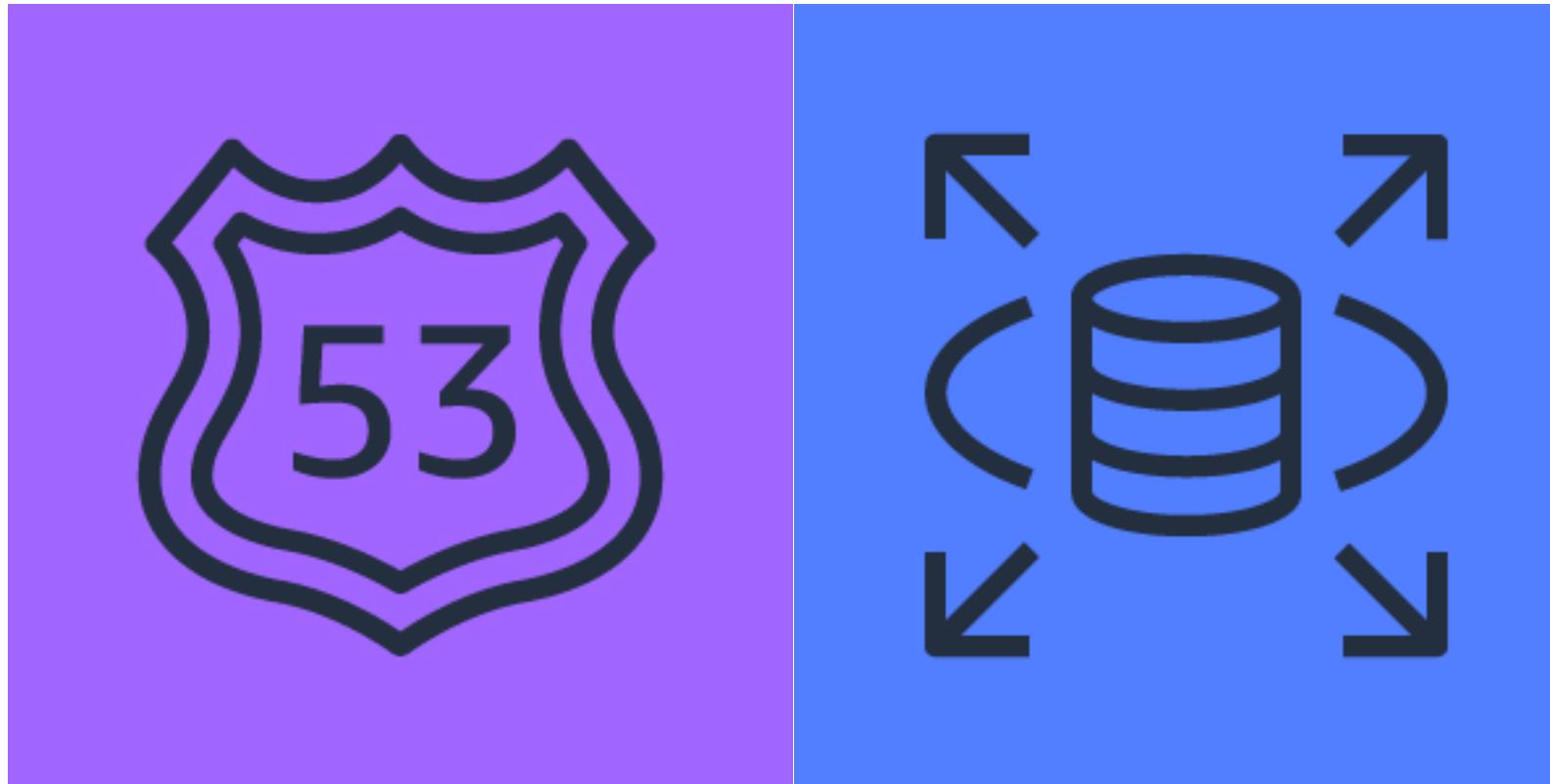
```
src/main/resources/views/home.html
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title>Home</title>
</head>
<body>
    <h1>Micronaut – Cognito example</h1>

    <h2 th:if="${security}">username: <span th:text="${security.attributes.get('email')}"></span></h2>
    <h2 th:unless="${security}">username: Anonymous</h2>

    <nav>
        <ul>
            <li th:unless="${security}"><a href="/oauth/login/cognito">Enter</a></li>
            <li th:if="${security}"><a href="/oauth/logout">Logout</a></li>
        </ul>
    </nav>
</body>
</html>
```

## Other Topics

---



# Micronaut Resources

- [gitter.im/micronautfw](https://gitter.im/micronautfw)
- [docs.micronaut.io](https://docs.micronaut.io)
- [guides.micronaut.io](https://guides.micronaut.io)
- [micronaut.io/faq.html](https://micronaut.io/faq.html)
- [github.com/micronaut-projects/micronaut-core](https://github.com/micronaut-projects/micronaut-core)
- [github.com/micronaut-projects/micronaut-examples](https://github.com/micronaut-projects/micronaut-examples)
- [objectcomputing.com/products/micronaut](https://objectcomputing.com/products/micronaut)
- [info@micronaut.io](mailto:info@micronaut.io)

# Questions?



## CONNECT WITH US

---

- 1+ (314) 579-0066
- @objectcomputing
- objectcomputing.com