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

OBJECT COMPUTING

18 November - Java 11 runtime now available in AWS Lambda

OBJECT COMPUTING

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.

OBJECT COMPUTING

Monday, December 2nd @ 6:15pm

Wednesday, December 4th @ 10:45am

Cold Start

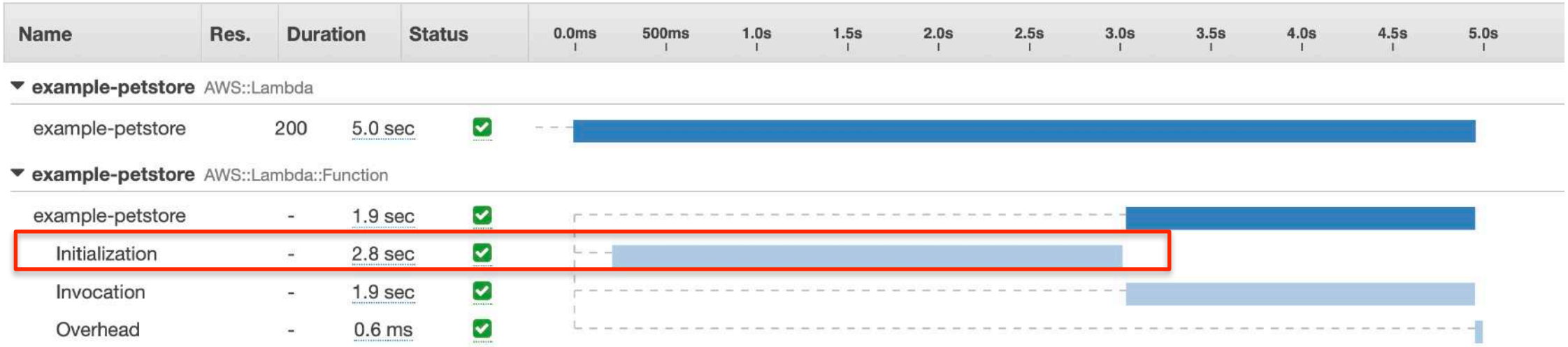
- AWS Lambda—the dynamically scaled and billed-per-execution compute service. Instances of Lambdas are added and removed dynamically.
- When a new instance handles its first request, the response time increases, which is called a **cold start**.
- After that request is processed, the instance stays alive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

The Initialization Step

OBJECT COMPUTING

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

Method	Response	Duration	Age	ID
--	200	5.0 sec	14.1 sec (2019-11-20 14:11:02 UTC)	1-5dd54976-8225456be488238078532a8d



Java and AWS Lambda Tips

OBJECT COMPUTING

- 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.

Java and AWS Lambda Tips

OBJECT COMPUTING

- **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.**

Java and AWS Lambda Tips

OBJECT COMPUTING

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

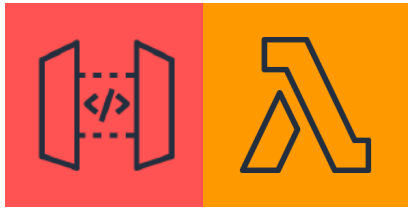
GraalVM

- 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
```

AWS Lambda Runtime

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
```

OBJECT
COMPUTING

Code Samples

OBJECT COMPUTING

- 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

Micronaut AWS Lambda

SINGLE FUNCTION

* @FunctionBean
* FunctionInitializer

AWS API Gateway Lambda Proxy Integration

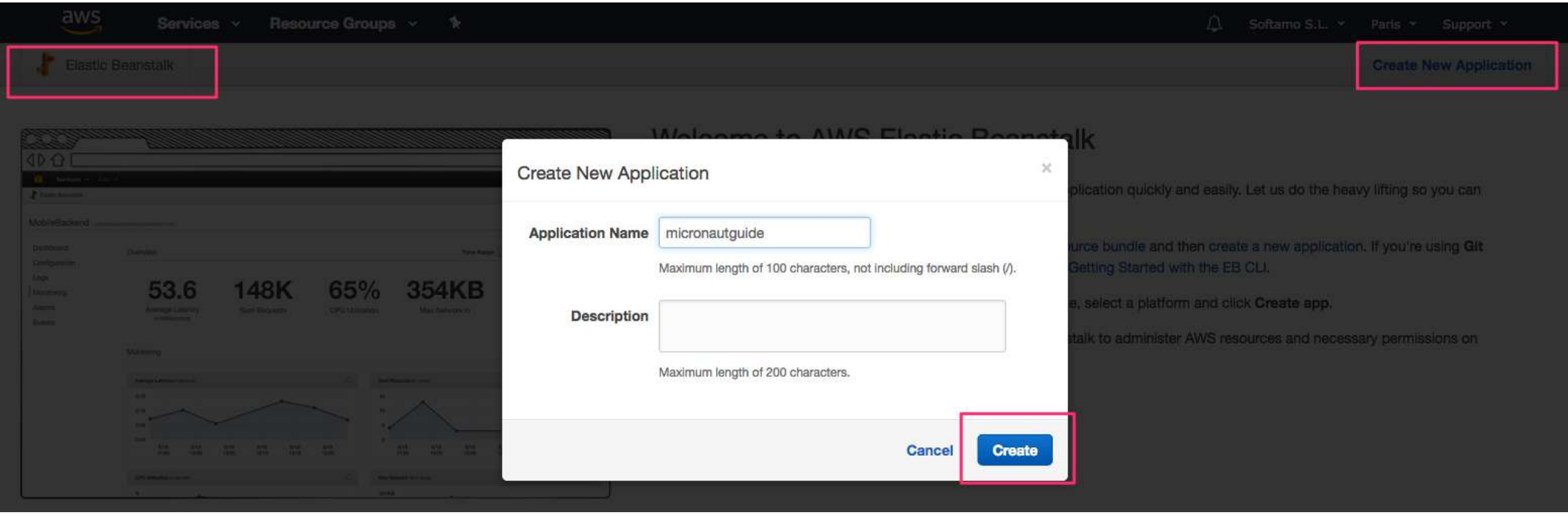
* micronaut-function-aws-api-proxy
* micronaut-function-aws-custom-runtime

Amazon Web Services

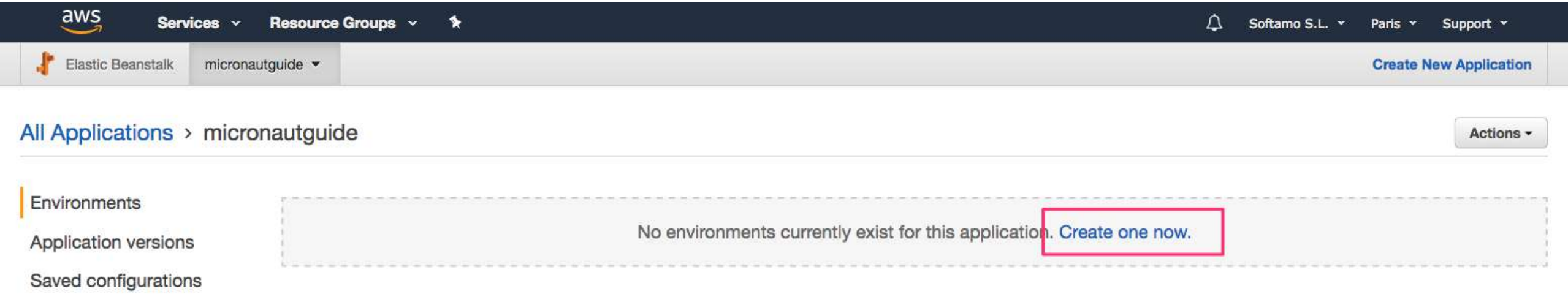
AWS Elastic Beanstalk

OBJECT COMPUTING

AWS Elastic Beanstalk

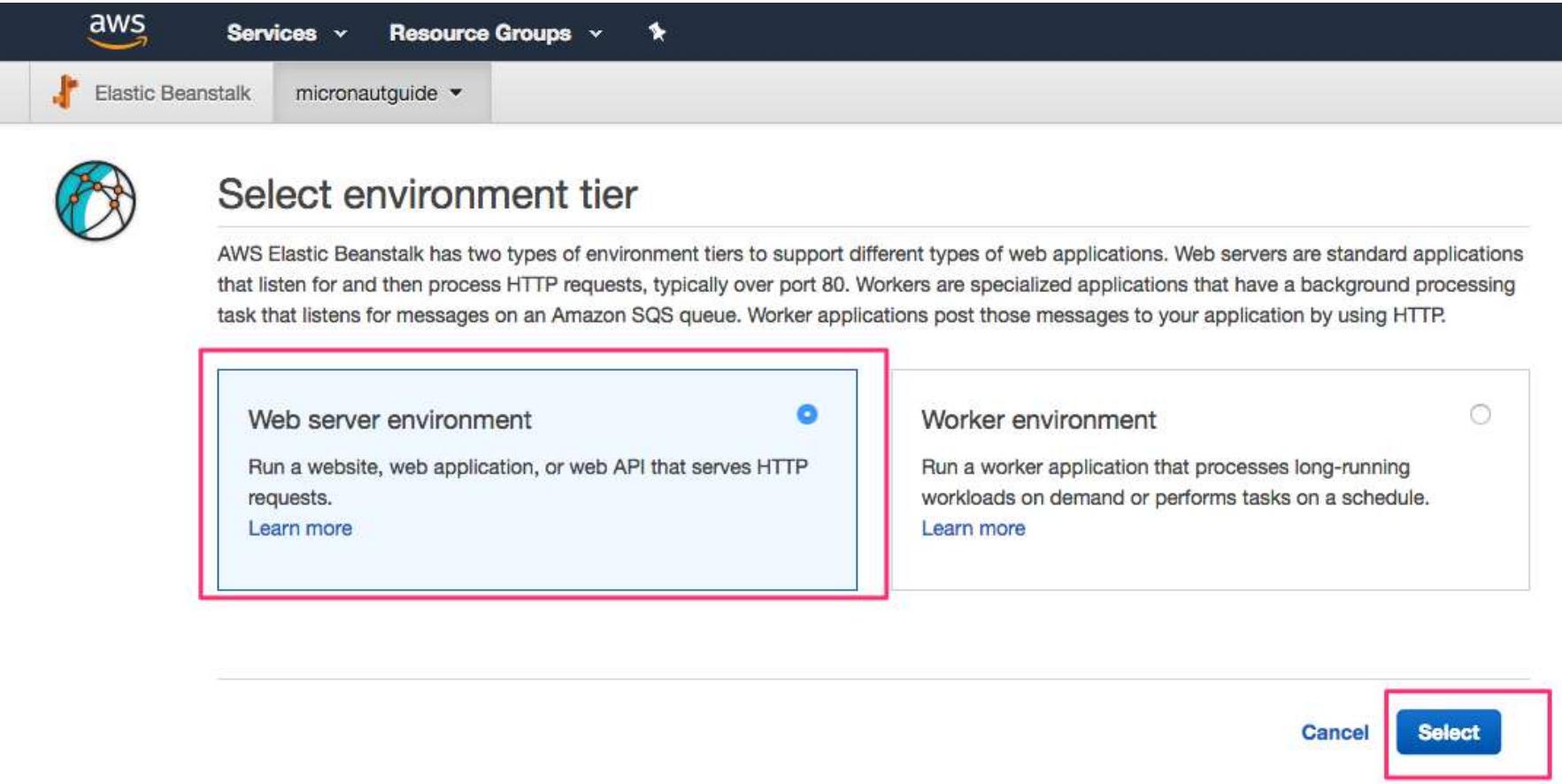


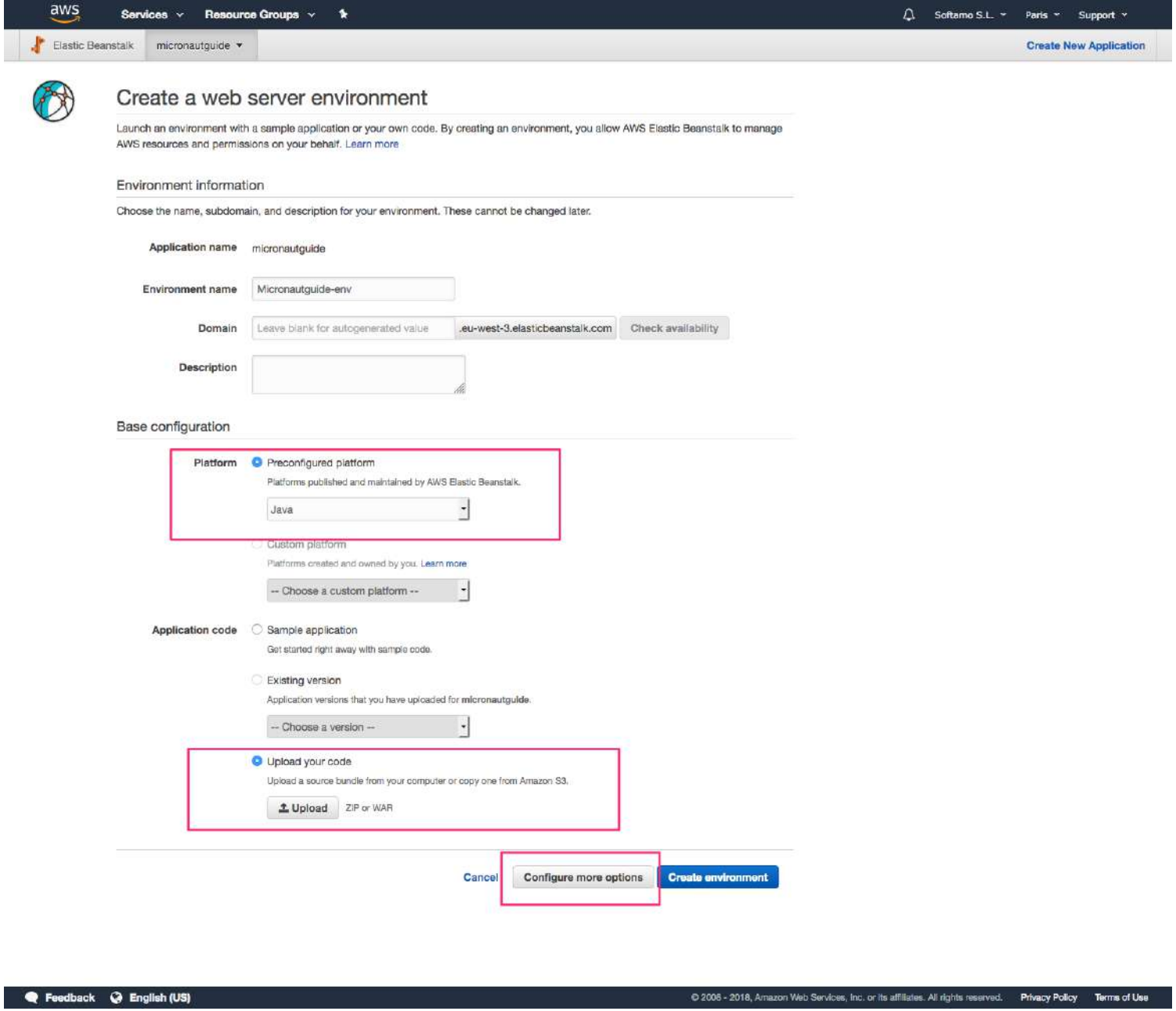
AWS Elastic Beanstalk



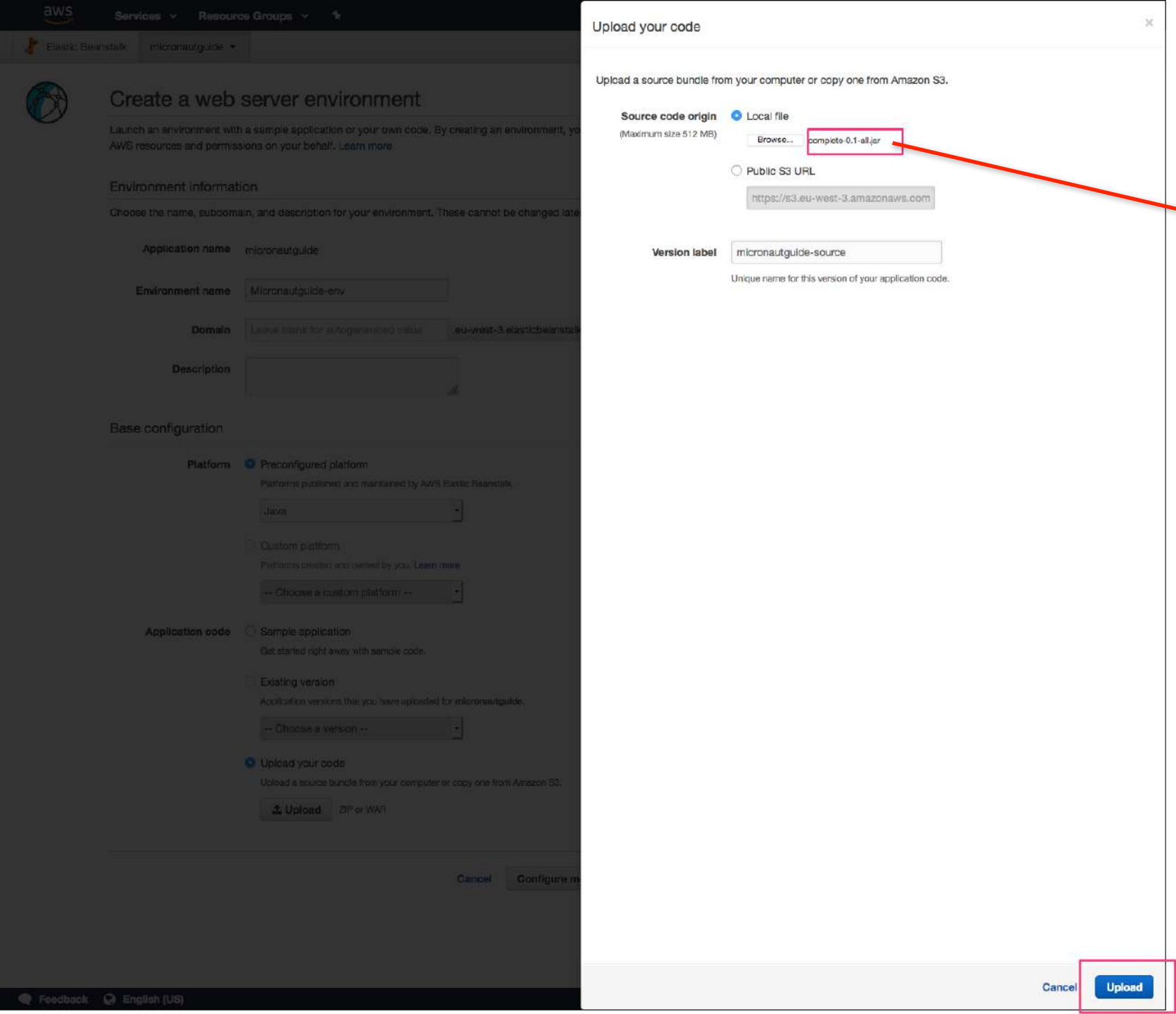
AWS Elastic Beanstalk

OBJECT COMPUTING

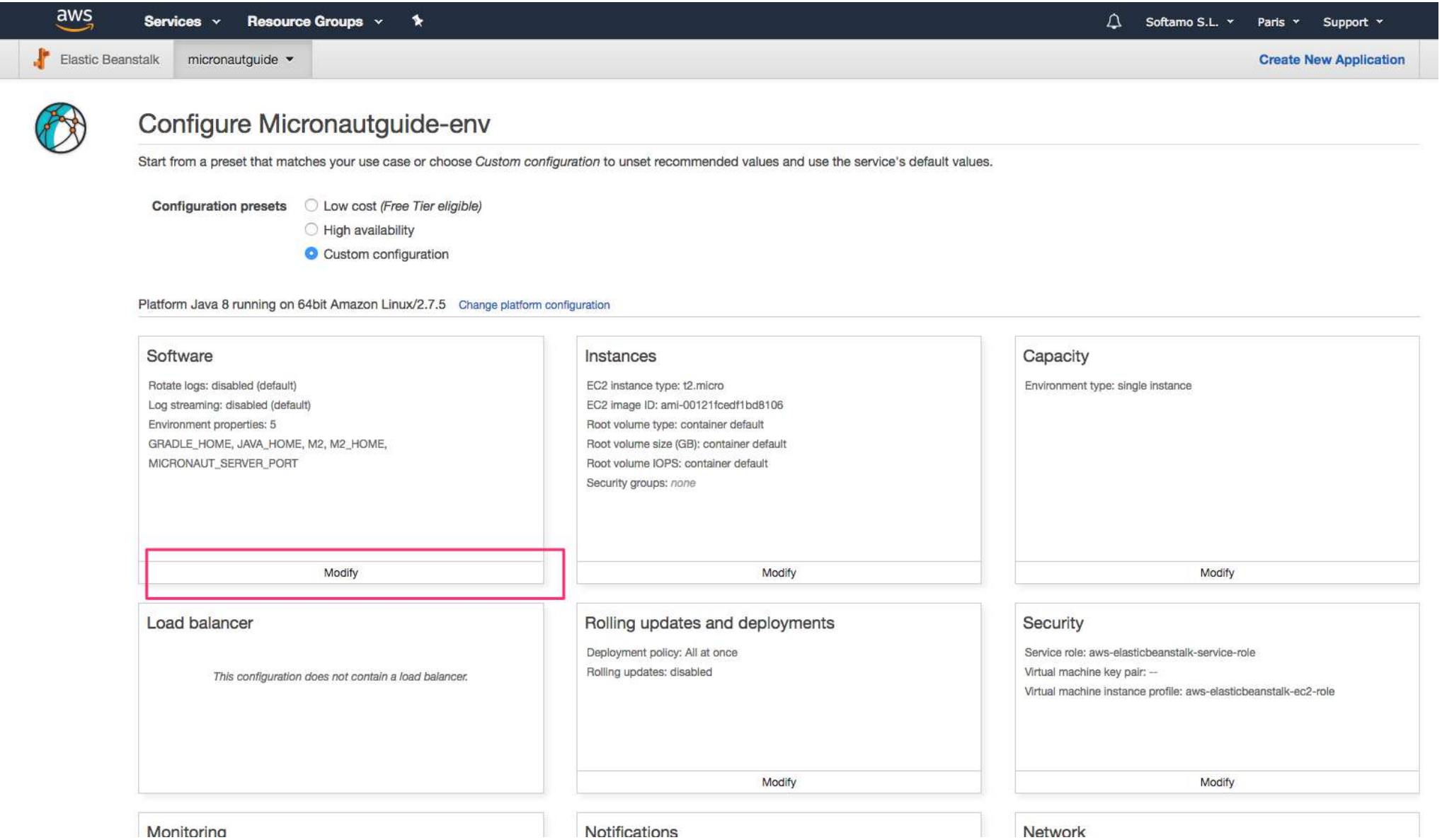




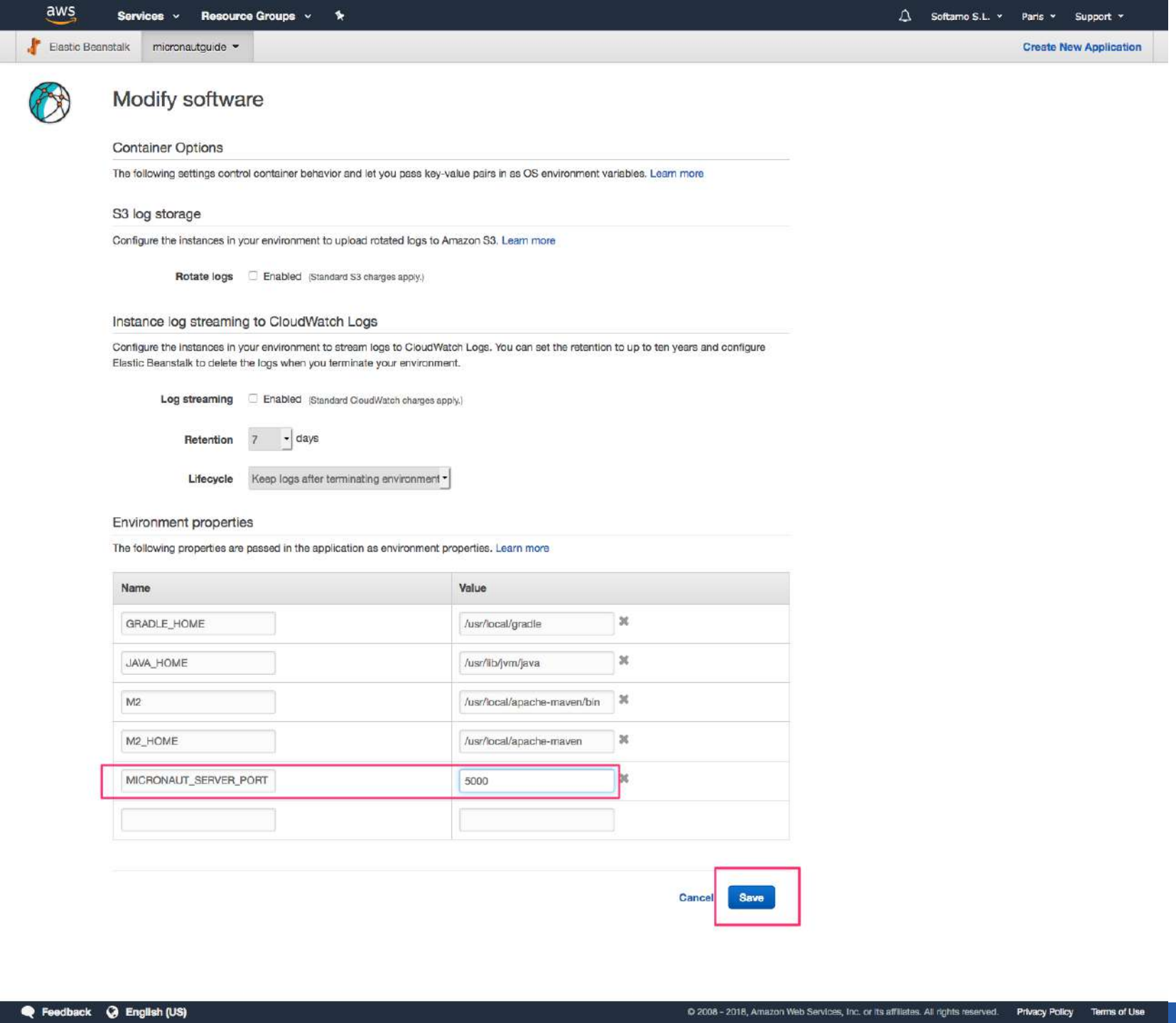
OBJECT COMPUTING



./gradlew shadowJar

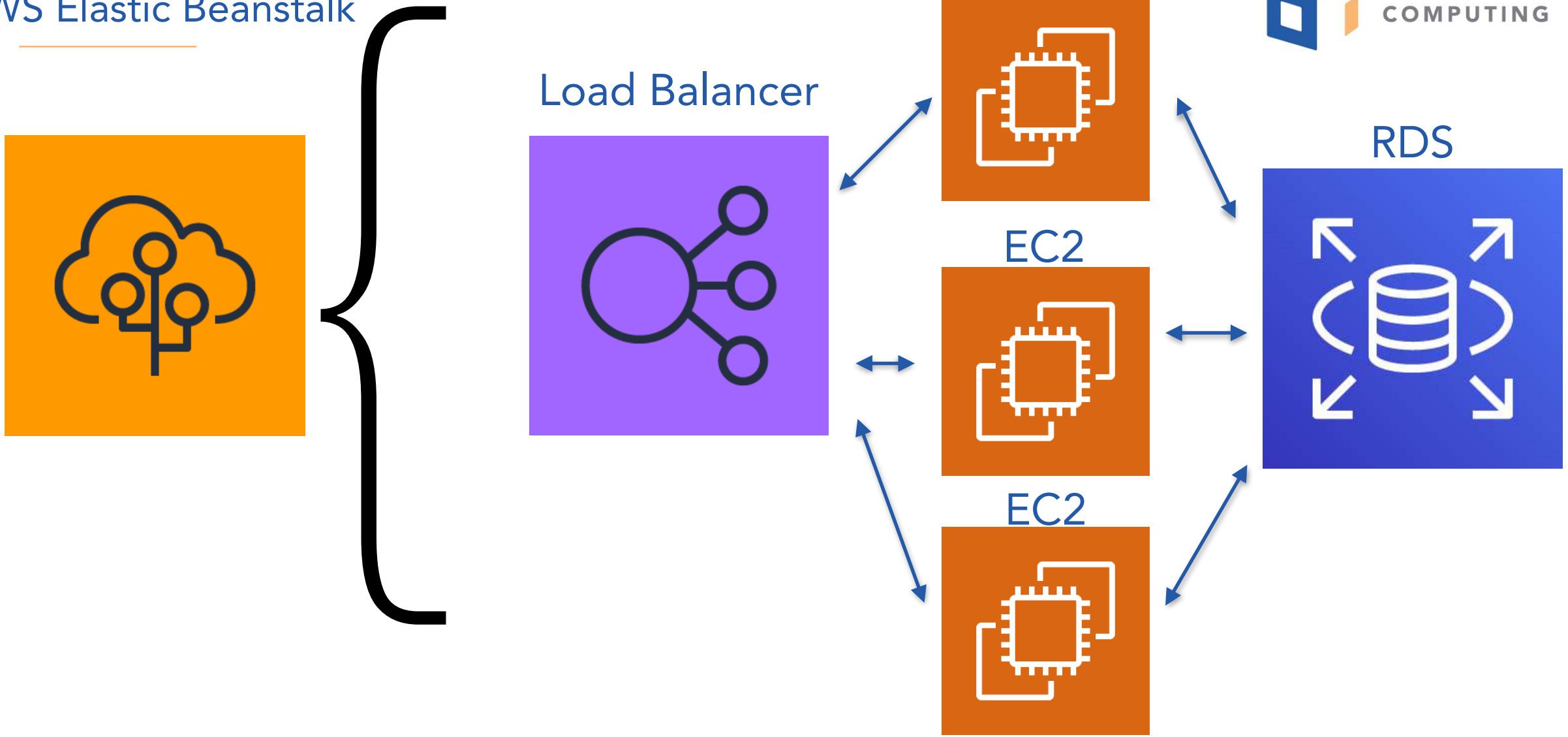


OBJECT COMPUTING

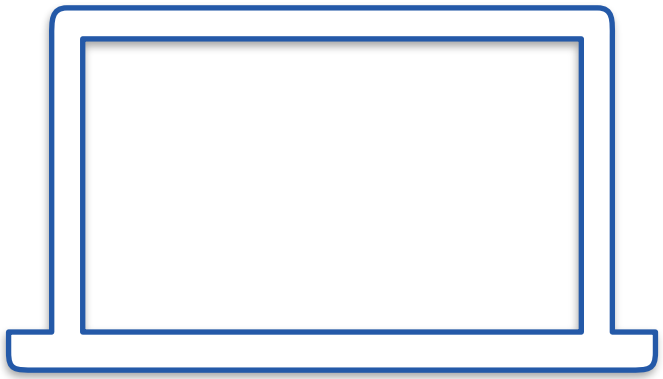


OBJECT COMPUTING

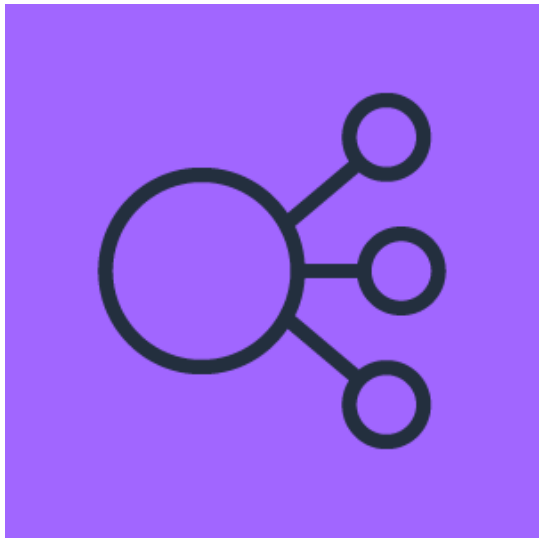
AWS Elastic Beanstalk



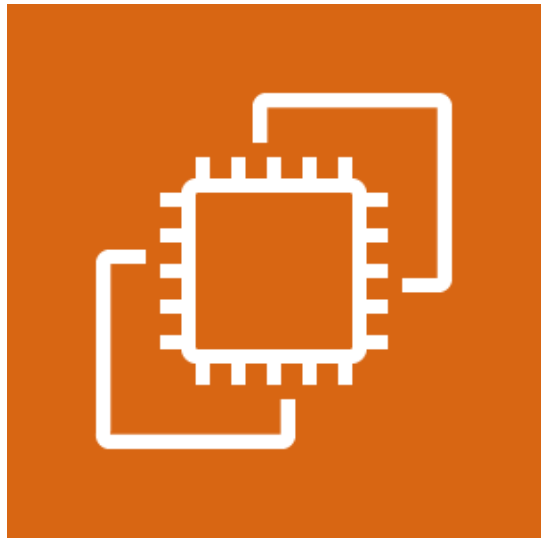
Elastic Load Balancer



124.12.3.231



10.0.0.23



10.0.0.23

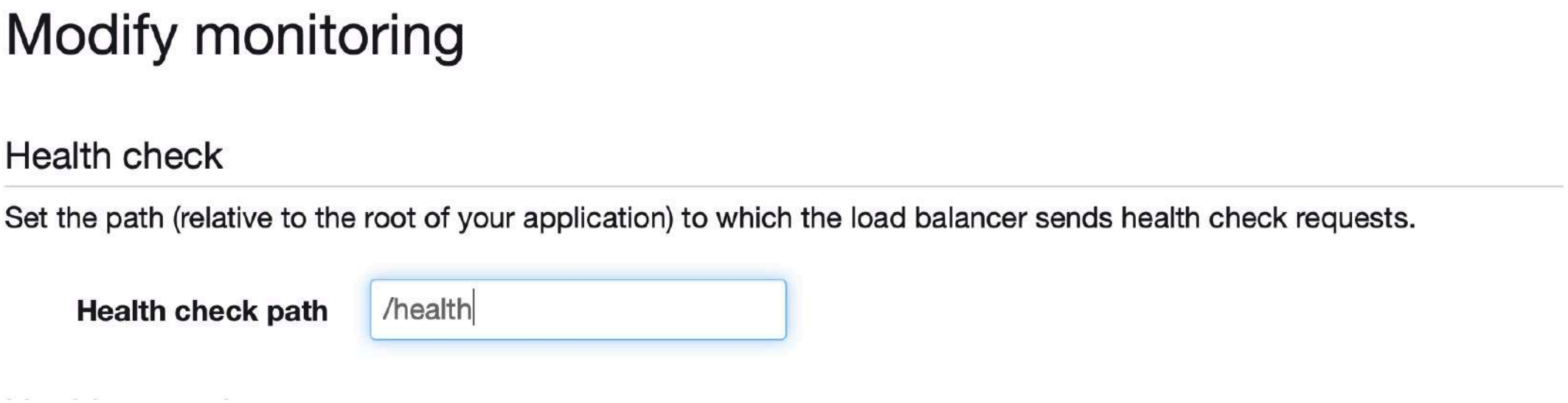
X-Forwarded-For: 124.12.3.231

```
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(HttpRequest request) {
        return httpHostResolver.resolve(request);
    }
}
```

AWS Elastic Beanstalk - Health Check

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.

AWS Elastic Beanstalk - Health Check



AWS Elastic Beanstalk - Health Check

OBJECT COMPUTING

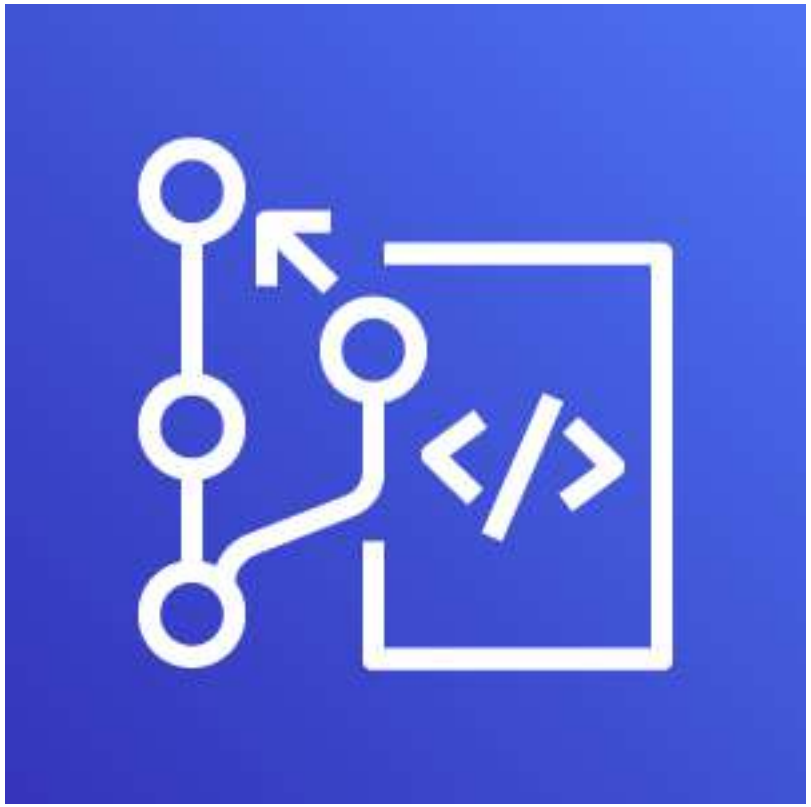
src/main/resources/application.yml

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

build.gradle

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

AWS CodeCommit



AWS CodeCommit

OBJECT COMPUTING

```
$ 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
```

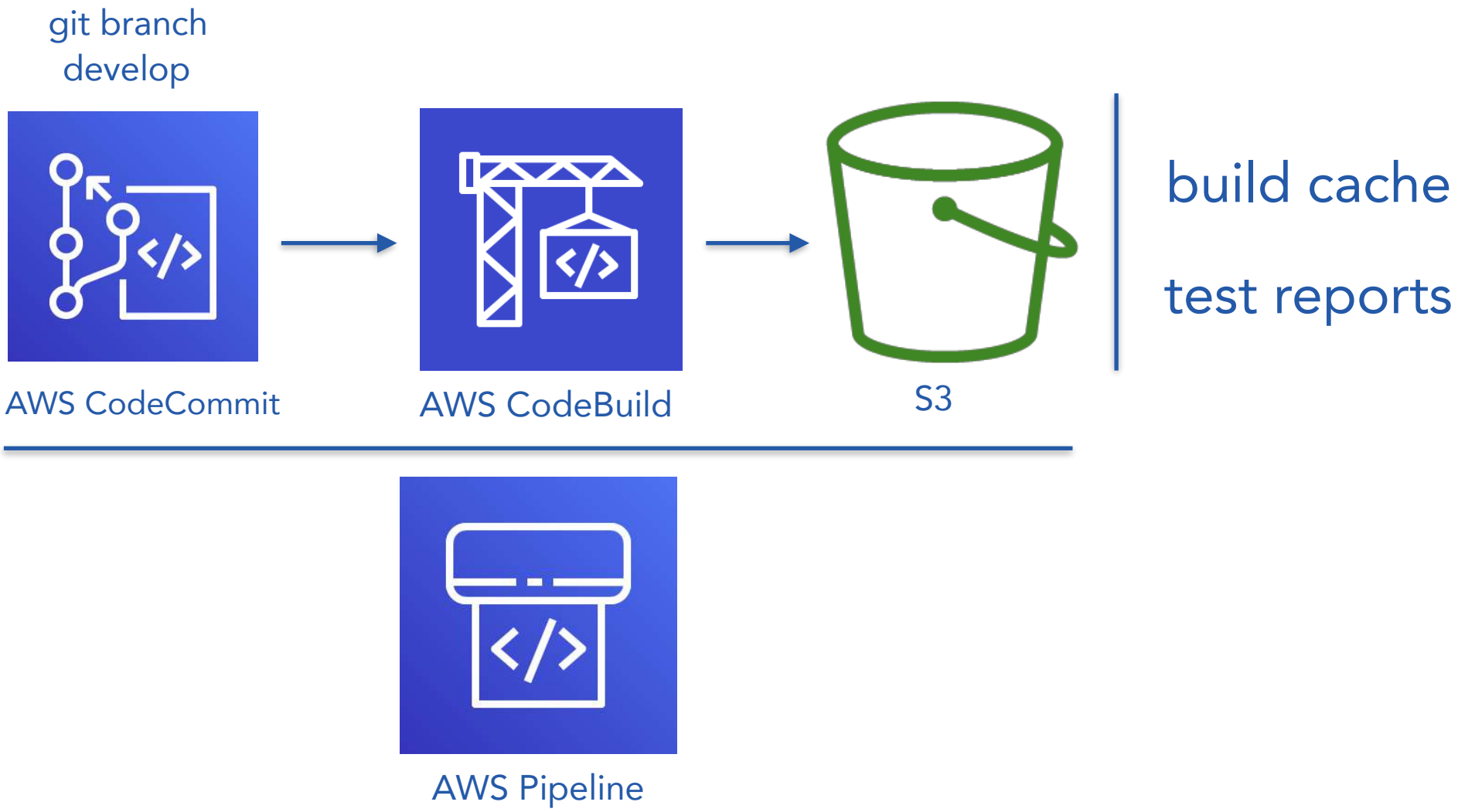
AWS CodeBuild

OBJECT COMPUTING

AWS Pipeline

OBJECT COMPUTING

AWS Pipeline



AWS CodeBuild

OBJECT COMPUTING

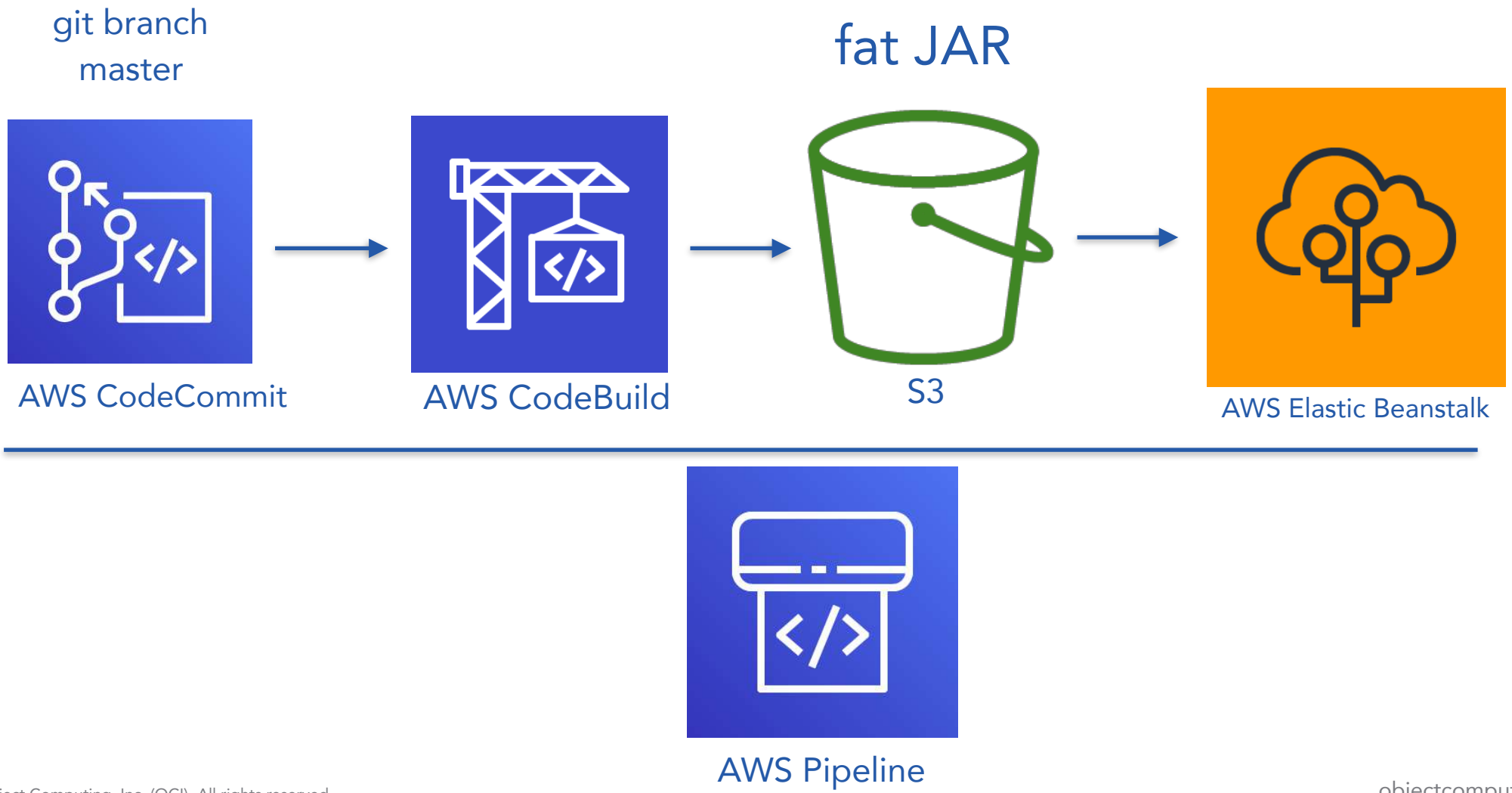
Runtime

Gradle Cache

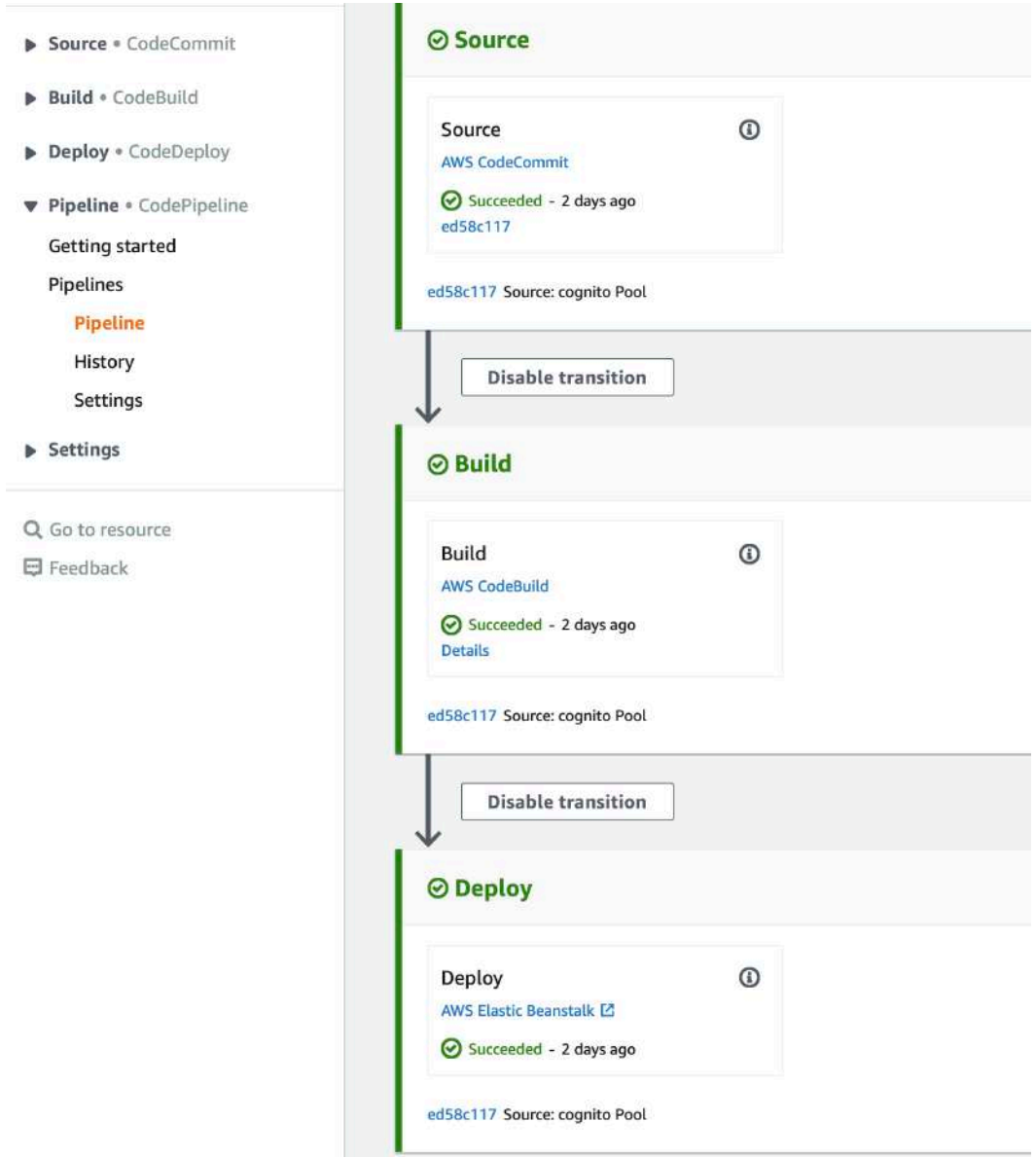
Test Reports

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

AWS Pipeline



AWS Pipeline



OBJECT COMPUTING

AWS CodeBuild

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

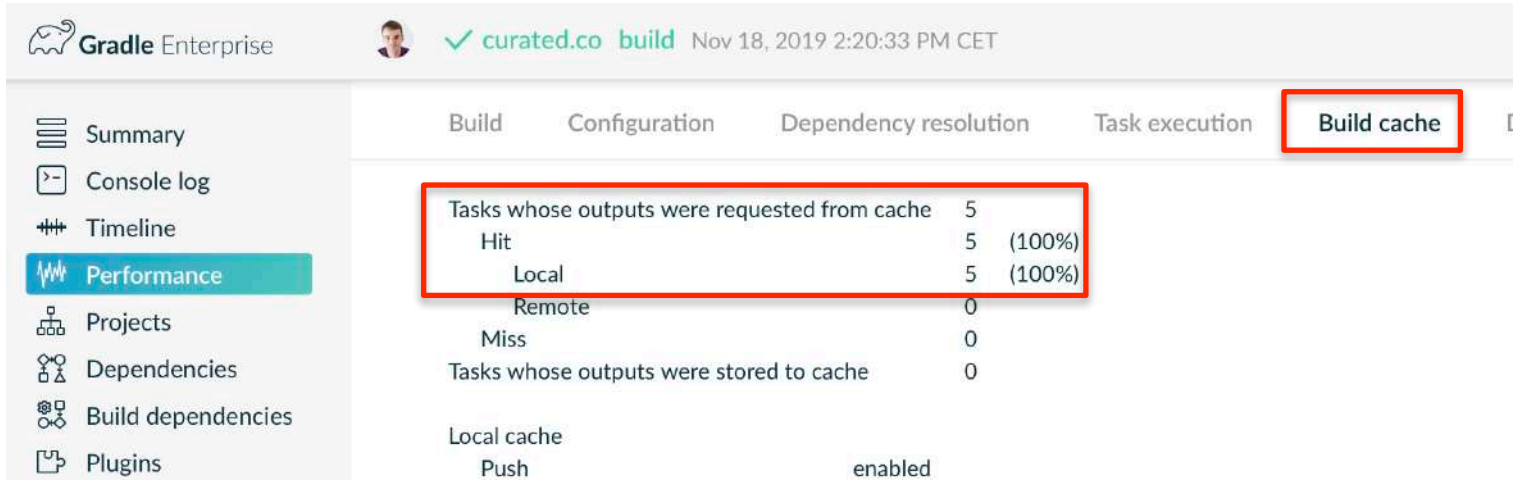
OBJECT COMPUTING

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/**/*'
```



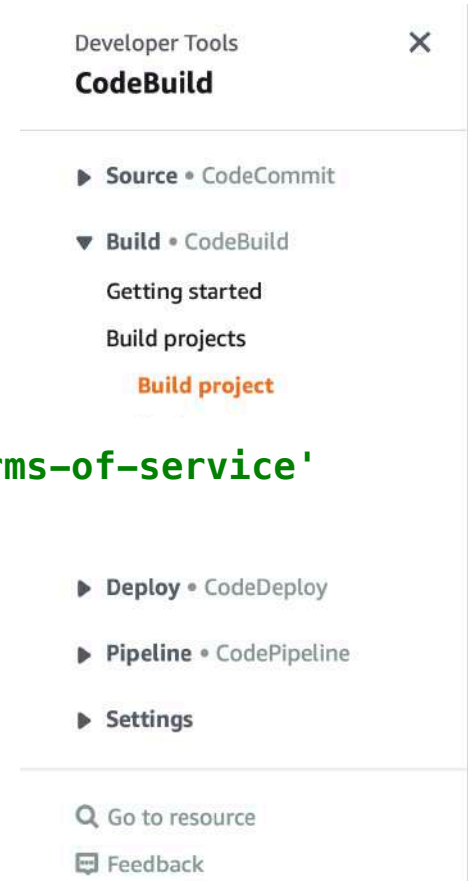
AWS CodeBuild - Build Scans

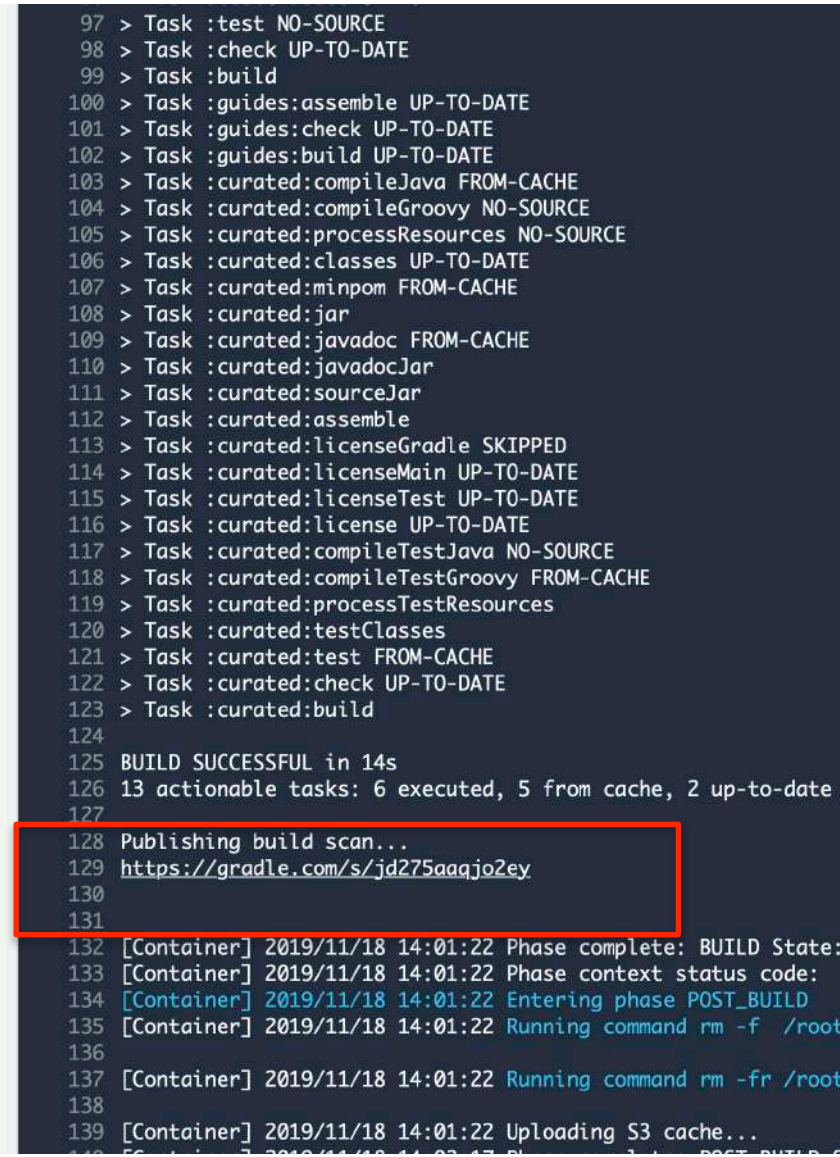
build.gradle

```
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
```





AWS Simple Email Service (SES)

OBJECT COMPUTING

AWS Simple Email Service (SES)

build.gradle

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

AWS Simple Email Service (SES)

OBJECT COMPUTING

```
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)

OBJECT COMPUTING

```
@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"
    ...
}
...
```

S3 - Simple Cloud Storage


```
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'
```

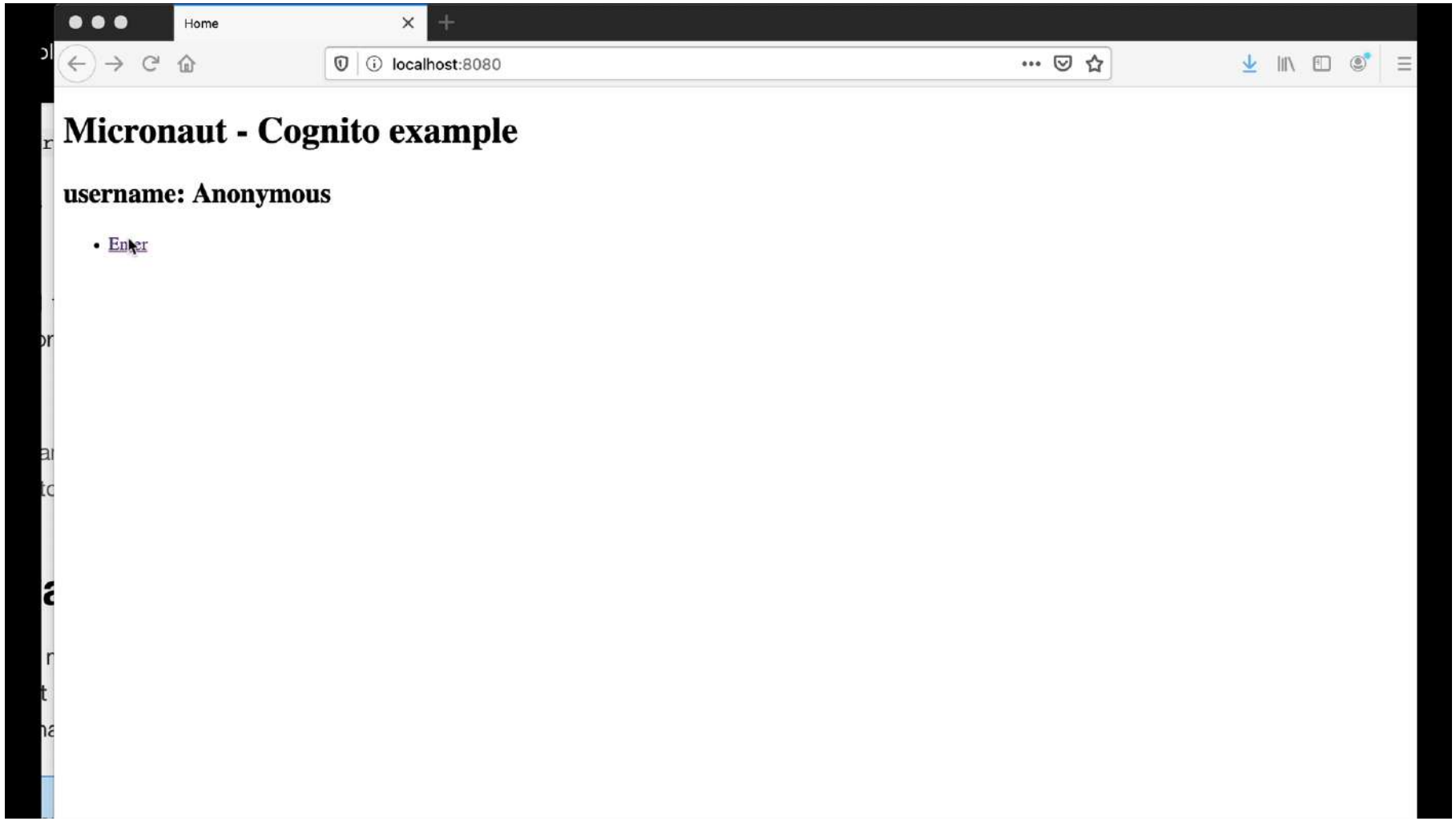
S3 - Simple Cloud Storage


```
@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");
            }
        }
    }
}
```

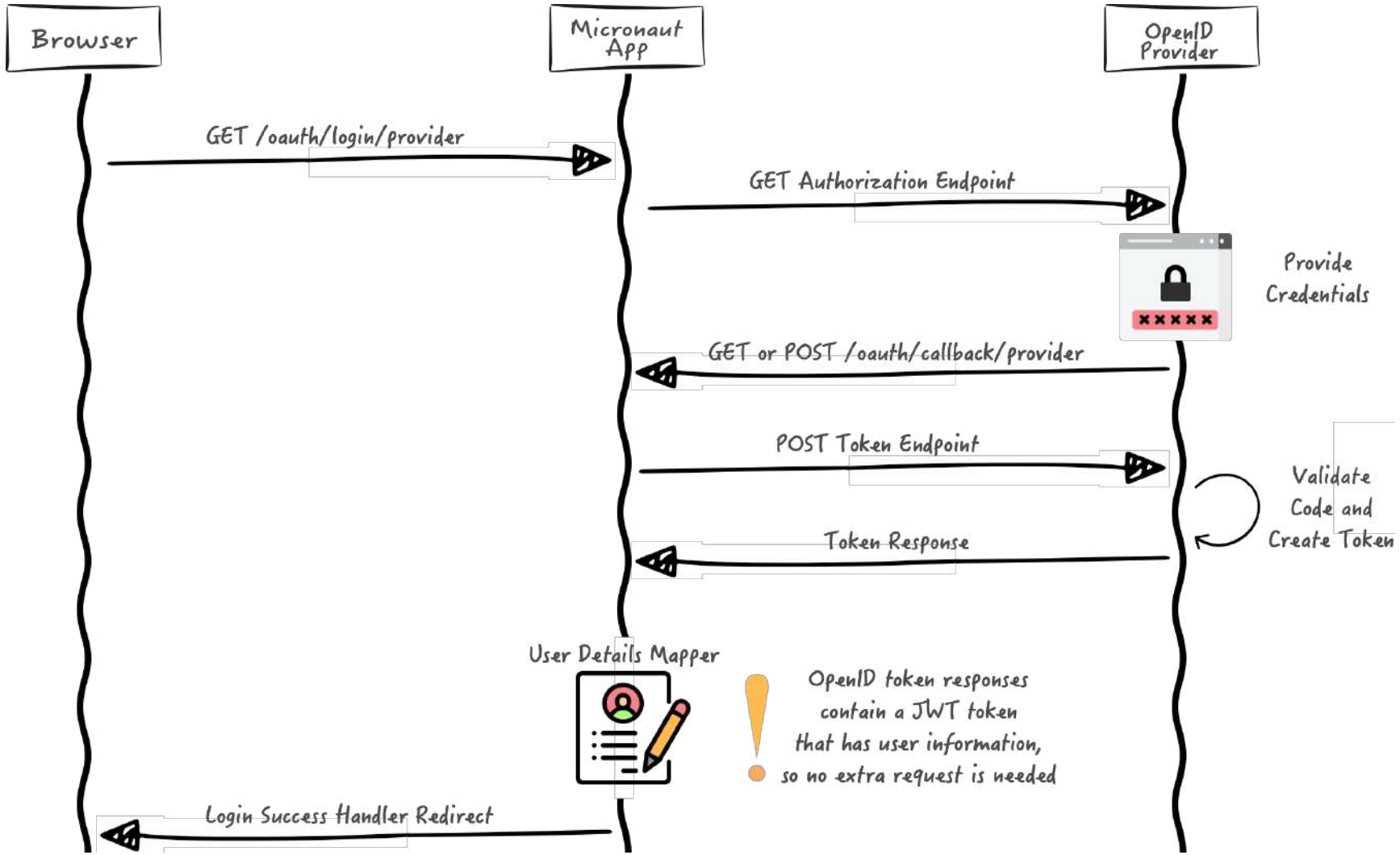
Amazon Cognito

OBJECT COMPUTING

Cognito example



Amazon Cognito



Amazon Cognito

build.gradle

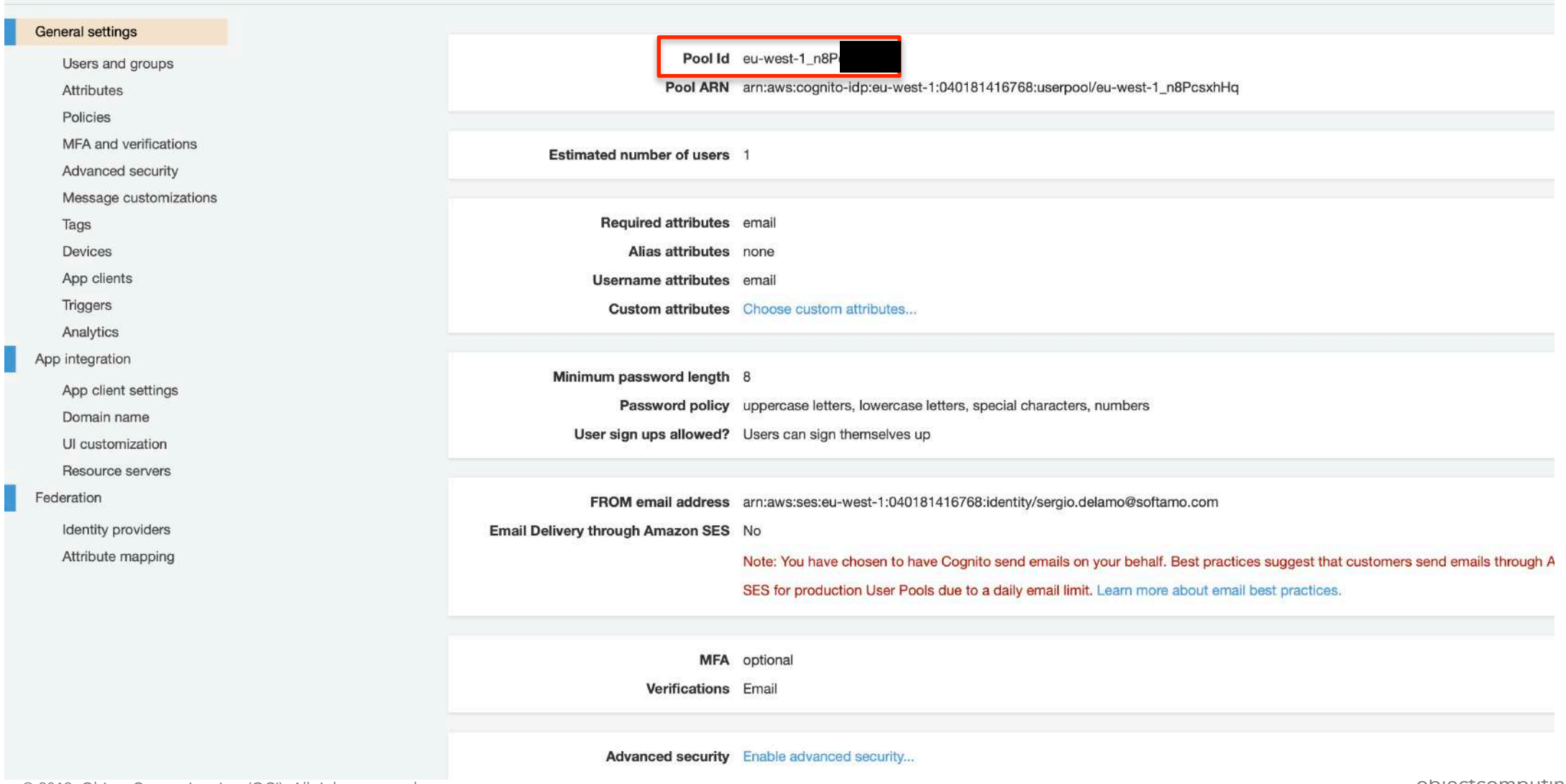
```
...
dependencies {
    ...
    annotationProcessor "io.micronaut:micronaut-security"
    implementation 'io.micronaut:micronaut-security-jwt'
    implementation 'io.micronaut.configuration:micronaut-security-oauth2'

}
...
```

Amazon Cognito

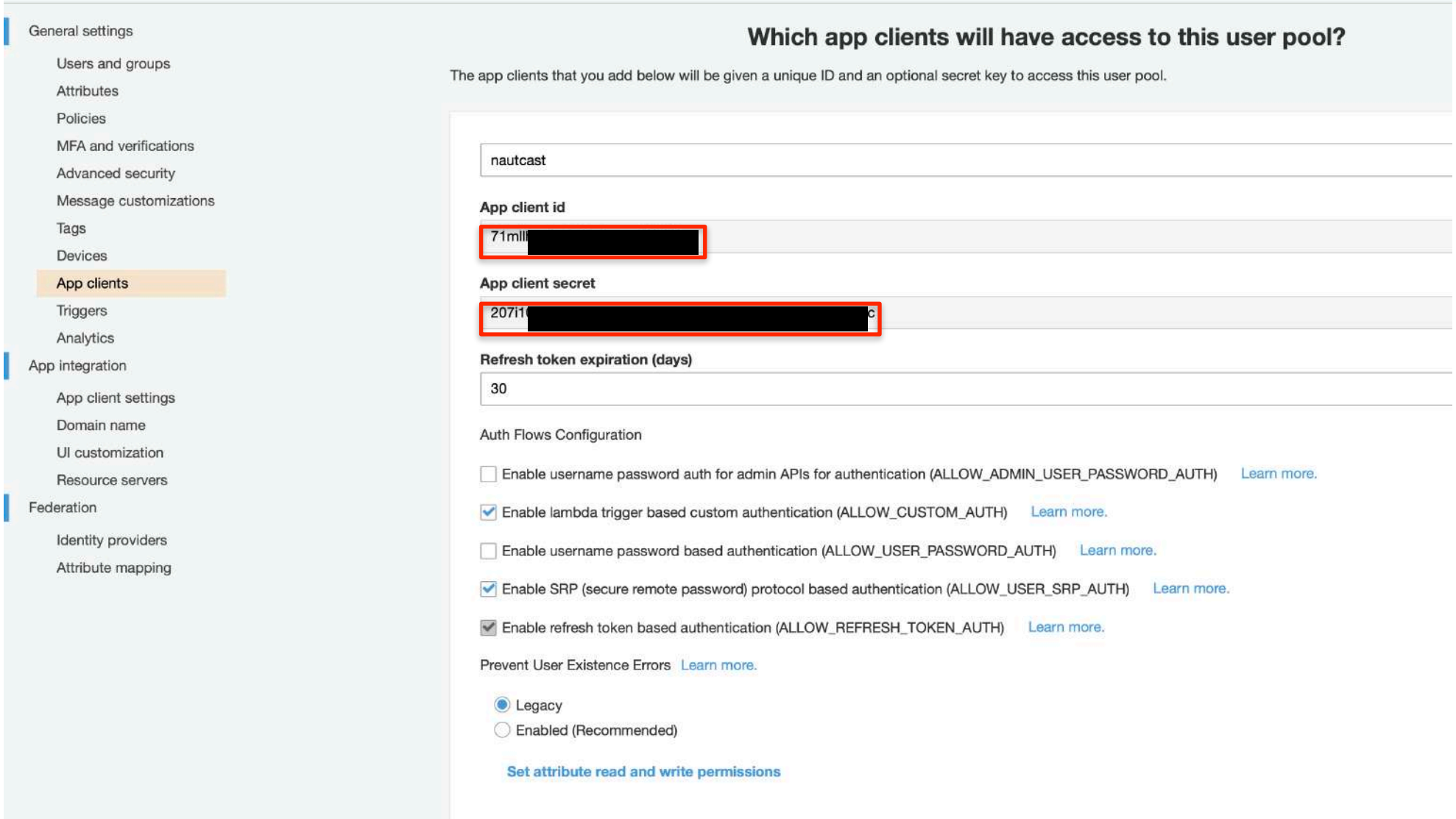
src/main/resources/application.yml

```
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



Amazon Cognito



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}/'
```

Amazon Cognito

build.gradle

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

Amazon Cognito


```
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<>();
    }
}
```

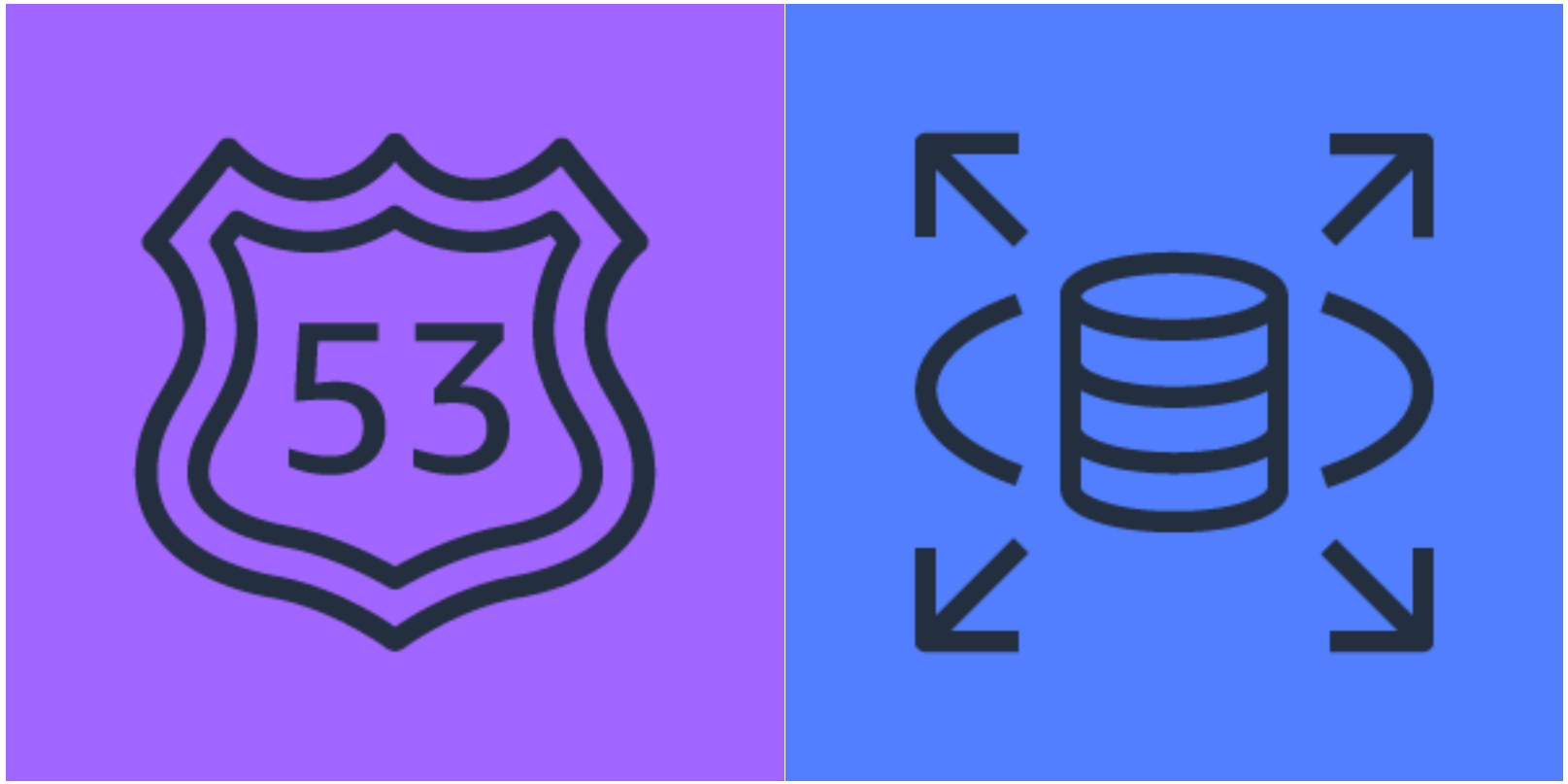
Amazon Cognito

OBJECT COMPUTING

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
- docs.micronaut.io
- guides.micronaut.io
- micronaut.io/faq.html
- github.com/micronaut-projects/micronaut-core
- github.com/micronaut-projects/micronaut-examples
- objectcomputing.com/products/micronaut
- info@micronaut.io

Questions?

OBJECT
COMPUTING

CONNECT WITH US

1+ (314) 579-0066

@objectcomputing

objectcomputing.com