The project is here:

~~C:\Users\claud\OneDrive\Documents\SSO\sb-saml-demo~~

C:\projects\spring-boot-saml-sp2

A screenshot of a computer

Description automatically generated with medium confidence

# Create the spring boot app

Documentation here

<https://developer.okta.com/code/java/spring_security_saml/>

In Visual Studio Code, display the command palette and type the command Spring Initializr: Create a gradle project. Respond to the questions, to create the project with the following options:

* Project: Gradle
* Spring Boot: 3.0.0 (SNAPSHOT)
* Dependencies: Spring Web, Spring Security, Thymeleaf

Save the project file in a new folder like sb-saml-demo, and reopen the folder in Visual Studio Code.

Display the HomeController.java file in Visual Studio Code. Then you can run the app with F5 to check that everything is fine, and display the home page in a browser:

<http://localhost:8080>

Graphical user interface, application, website

Description automatically generated

Then add the code as specified here.

<https://developer.okta.com/code/java/spring_security_saml/#create-a-spring-boot-app-with-saml-support>

In the build.gradle file, add the saml2 line in the dependencies object:

dependencies {

    implementation 'org.springframework.boot:spring-boot-starter-security'

    implementation 'org.springframework.boot:spring-boot-starter-thymeleaf'

    implementation 'org.springframework.boot:spring-boot-starter-web'

    implementation 'org.thymeleaf.extras:thymeleaf-extras-springsecurity6'

    implementation 'org.springframework.security:spring-security-saml2-service-provider'

    testImplementation 'org.springframework.boot:spring-boot-starter-test'

    testImplementation 'org.springframework.security:spring-security-test'

}

## Creating your keys and certificates

In a terminal window in Visual Studio Code run the following:

rsa:2048 -new -nodes -x509 -days 3650 -keyout private.key -out public.cer

openssl req -newkey rsa:2048 -nodes -keyout local.key -x509 -days 365 -out local.crt

The local.key file is private key, and the local.crt is a public key, or a certificate.

This will generate a PKCS#8 format private-key and public key pair.

Text

Description automatically generated

Move those files in your ressouces folder.

You will now have a certificate and private key file in your resources folder which is conveniently located on our classpath in Spring Boot by default.

Finally, make sure to copy the public key (certificate) of your identity provider (if you have one) and place it in the credentials folder naming it identity-provider-certificate.crt.

How it works:

When creating an Okta SAML2 integration application in an Okta account (an organization), we generate a metadata uri that must be configured in the corresponding SAML2 application. This URI looks like:

https://dev-38785465.okta.com/app/exk6fmgk9u8BTZOg15d7/sso/saml/metadata

This is what enables the SAML2 app to interact with the Okta organization.

To be able to log out a user with Okta, we must generate a PKCS#8 format private-key and public key pair, that are stored in the SAML2 app. The public key is a certificate which must also be stored in the Okta SAML2 application.

# Testing the deep link

## Use case 1 – not authenticated

In this use case I am not authenticated, and I attempt to display the URL

<http://localhost:8080/check>

I am redirected to the okta loginpage

Graphical user interface, application

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| Step | What | Status | Comment |
| 1 | GET http://localhost:8080/check | 302  (redirect) | Redirected to the Location in the response headers:  http://localhost:8080/saml2/authenticate/okta |
| 2 | GET http://localhost:8080/saml2/authenticate/okta | 200 |  |
| 3 | POST <https://dev-38785465.okta.com/app/dev-38785465_springbootsaml_1/exk6fmgk9u8BTZOg15d7/sso/saml>  The SAML request:  <?xml version="1.0" encoding="UTF-8"?>  <saml2p:AuthnRequest xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol" AssertionConsumerServiceURL="http://localhost:8080/login/saml2/sso/okta" Destination="https://dev-38785465.okta.com/app/dev-38785465\_springbootsaml\_1/exk6fmgk9u8BTZOg15d7/sso/saml" ForceAuthn="false" ID="ARQ307da4d-49e6-4a07-8e65-1dcaf2e73cfd" IsPassive="false" IssueInstant="2022-09-26T14:56:05.778Z" ProtocolBinding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST" Version="2.0">  <saml2:Issuer xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion">  http://localhost:8080/saml2/service-provider-metadata/okta  </saml2:Issuer>  </saml2p:AuthnRequest> | 200 | This is a SAML request with a RelayState. |
| 4 | GET https://login.okta.com/discovery/iframe.html | 200 |  |
| 5 | GET https://dev-38785465.okta.com/app/dev-38785465\_springbootsaml\_1/exk6fmgk9u8BTZOg15d7/sso/saml | 200 |  |
| 6 | GET https://dev-38785465.okta.com/app/dev-38785465\_springbootsaml\_1/exk6fmgk9u8BTZOg15d7/sso/saml | 200 |  |
| 7 | GET https://login.okta.com/discovery/iframe.html | 200 | Display of the login page in okta.  The following steps are after the authentication with okta. |
| 1 | GET https://dev-38785465.okta.com/auth/services/devicefingerprint | 200 | <https://dev-38785465.okta.com/auth/services/devicefingerprint> |
| 2 | GET https://dev-38785465.okta.com/login/token/redirect?stateToken=02.id.7AksoSEHsVWxSP7BYqAiRjkOiyPnXBpB39NgqeQf | 200 |  |
| 3 | POST <http://localhost:8080/login/saml2/sso/okta>  The SAML response:  <?xml version="1.0" encoding="UTF-8"?>  <saml2p:Response Destination="http://localhost:8080/login/saml2/sso/okta" ID="id30051927742649281301902832" IssueInstant="2022-09-26T16:24:34.458Z" Version="2.0" xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol">  <saml2:Issuer Format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity" xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion">  http://www.okta.com/exk6fmgk9u8BTZOg15d7  </saml2:Issuer>  <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">  <ds:SignedInfo>  <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>  <ds:SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256"/>  <ds:Reference URI="#id30051927742649281301902832">  <ds:Transforms>  <ds:Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>  <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>  </ds:Transforms>  <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>  <ds:DigestValue>  ZBdvCkXZJ6Rn8C+BWqvJk/ok2FHLgoxlOaTnIJXbR84=  </ds:DigestValue>  </ds:Reference>  </ds:SignedInfo>  <ds:SignatureValue>  67AkIrz6S0/wh6qbPMLJAYweHwEs...  </ds:SignatureValue>  <ds:KeyInfo>  <ds:X509Data>  <ds:X509Certificate>  MIIDqDCCApCgAwIBAgIGAYMdLZYtMA0GCSqGSIb3DQEBCwUAMIGUMQswCQYDVQQGEwJVUzETMBEG  A1UECAwKQ2FsaWZvcm5pYTEWMBQGA1UEBwwNU2FuIEZyYW5jaXNjbzENMAsGA1UECgwET2t0YTEU  MBIGA1UECwwLU1NPUHJvdmlkZXIxFTATBgNVBAMMDGRldi0zODc4NTQ2NTEcMBoGCSqGSIb3DQEJ  ARYNaW5mb0Bva3RhLmNvbTAeFw0yMjA5MDgxMjU2NTBaFw0zMjA5MDgxMjU3NTBaMIGUMQswCQYD  VQQGEwJVUzETMBEGA1UECAwKQ2FsaWZvcm5pYTEWMBQGA1UEBwwNU2FuIEZyYW5jaXNjbzENMAsG  A1UECgwET2t0YTEUMBIGA1UECwwLU1NPUHJvdmlkZXIxFTATBgNVBAMMDGRldi0zODc4NTQ2NTEc  MBoGCSqGSIb3DQEJARYNaW5mb0Bva3RhLmNvbTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoC  ggEBAO264iHSgJ7UdkeIvCcwUWHsNsfX5phow+80QfiXU7MH6PRusI15oJs3nB36IQucmV7kgdTJ  HN7s7rMxFZRHebmj/31aARVnr5KSrMpmKF1HrBSkdOAB0sYbSdVjsD+mtmb642SEKkLz8hBJ2OXs  nx/1RyTh7MDi6rwLRZaRsSwETgBrF8OZ6exHO3L0W+wt/z6ODZipgtCZFi1NVrJEtooXEleKW1dE  m6o6eaD9/YQmP08SHSA9WHIbN+mZM4N7RAomn0+1hQzg+AkKQjw9+jmjpTve8AaB8J2CxTvDab9T  8Kof2UqnY6nXoH1vevFJwdm6pWUSWLQ+5J2jypkapL0CAwEAATANBgkqhkiG9w0BAQsFAAOCAQEA  PZu4hk0D4+3zaZ/OtXVV8zuSuLM/0e5vXami6j7kQkAFJUlpP31Hccbvh5+Ps2xTUAidEYREf+hd  pqFESCyKqLHFGwlUy1DPn2MHXGHbB31uqCl9WaZkzMBeA7QAGzf4Mwhm1UmH2vYS0I18pRXK5cJq  G2BKu92vxy5l4V1TXMHYBcGNiB3U6BEWSG1XLWasKUEL+dmHPyb+ubLdlzcTGxRtGGi16ruKZbap  sP5YB4nzd5KU2mqo+qzxZ9jweSatV5+ng7SXXfEjGa/i/hcOoKZwAn0g157nFLumW4neIKJ7mSc2  OLp3gHsf5kvMvj62liw/56VlOSmWH5xYp0YC7Q==  </ds:X509Certificate>  </ds:X509Data>  </ds:KeyInfo>  </ds:Signature>  <saml2p:Status xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol">  <saml2p:StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>  </saml2p:Status>  <saml2:Assertion ID="id3005192774401327617803591" IssueInstant="2022-09-26T16:24:34.458Z" Version="2.0" xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion">  <saml2:Issuer Format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity" xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion">  http://www.okta.com/exk6fmgk9u8BTZOg15d7  </saml2:Issuer>  <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">  <ds:SignedInfo>  <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>  <ds:SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256"/>  <ds:Reference URI="#id3005192774401327617803591">  <ds:Transforms>  <ds:Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>  <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>  </ds:Transforms>  <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>  <ds:DigestValue>  Mj8MiEw3UOGhlkKn/nGkLnnB8aPTPAfE/59V8ASHzJ4=  </ds:DigestValue>  </ds:Reference>  </ds:SignedInfo>  <ds:SignatureValue>  vqzFTz5UWUX8/MQL0L4QVF3eF4LDp4PVo...  </ds:SignatureValue>  <ds:KeyInfo>  <ds:X509Data>  <ds:X509Certificate>  MIIDqDCCApCgAwIBAgIGAYMdLZYtMA0GCSqGSIb3DQEBCwUAMIGUMQswCQYDVQQGEwJVUzETMBEG  A1UECAwKQ2FsaWZvcm5pYTEWMBQGA1UEBwwNU2FuIEZyYW5jaXNjbzENMAsGA1UECgwET2t0YTEU  MBIGA1UECwwLU1NPUHJvdmlkZXIxFTATBgNVBAMMDGRldi0zODc4NTQ2NTEcMBoGCSqGSIb3DQEJ  ARYNaW5mb0Bva3RhLmNvbTAeFw0yMjA5MDgxMjU2NTBaFw0zMjA5MDgxMjU3NTBaMIGUMQswCQYD  VQQGEwJVUzETMBEGA1UECAwKQ2FsaWZvcm5pYTEWMBQGA1UEBwwNU2FuIEZyYW5jaXNjbzENMAsG  A1UECgwET2t0YTEUMBIGA1UECwwLU1NPUHJvdmlkZXIxFTATBgNVBAMMDGRldi0zODc4NTQ2NTEc  MBoGCSqGSIb3DQEJARYNaW5mb0Bva3RhLmNvbTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoC  ggEBAO264iHSgJ7UdkeIvCcwUWHsNsfX5phow+80QfiXU7MH6PRusI15oJs3nB36IQucmV7kgdTJ  HN7s7rMxFZRHebmj/31aARVnr5KSrMpmKF1HrBSkdOAB0sYbSdVjsD+mtmb642SEKkLz8hBJ2OXs  nx/1RyTh7MDi6rwLRZaRsSwETgBrF8OZ6exHO3L0W+wt/z6ODZipgtCZFi1NVrJEtooXEleKW1dE  m6o6eaD9/YQmP08SHSA9WHIbN+mZM4N7RAomn0+1hQzg+AkKQjw9+jmjpTve8AaB8J2CxTvDab9T  8Kof2UqnY6nXoH1vevFJwdm6pWUSWLQ+5J2jypkapL0CAwEAATANBgkqhkiG9w0BAQsFAAOCAQEA  PZu4hk0D4+3zaZ/OtXVV8zuSuLM/0e5vXami6j7kQkAFJUlpP31Hccbvh5+Ps2xTUAidEYREf+hd  pqFESCyKqLHFGwlUy1DPn2MHXGHbB31uqCl9WaZkzMBeA7QAGzf4Mwhm1UmH2vYS0I18pRXK5cJq  G2BKu92vxy5l4V1TXMHYBcGNiB3U6BEWSG1XLWasKUEL+dmHPyb+ubLdlzcTGxRtGGi16ruKZbap  sP5YB4nzd5KU2mqo+qzxZ9jweSatV5+ng7SXXfEjGa/i/hcOoKZwAn0g157nFLumW4neIKJ7mSc2  OLp3gHsf5kvMvj62liw/56VlOSmWH5xYp0YC7Q==  </ds:X509Certificate>  </ds:X509Data>  </ds:KeyInfo>  </ds:Signature>  <saml2:Subject xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion">  <saml2:NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified">  claude.bonnaud.action3d@gmail.com  </saml2:NameID>  <saml2:SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">  <saml2:SubjectConfirmationData NotOnOrAfter="2022-09-26T16:29:34.458Z" Recipient="http://localhost:8080/login/saml2/sso/okta"/>  </saml2:SubjectConfirmation>  </saml2:Subject>  <saml2:Conditions NotBefore="2022-09-26T16:19:34.458Z" NotOnOrAfter="2022-09-26T16:29:34.458Z" xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion">  <saml2:AudienceRestriction>  <saml2:Audience>  http://localhost:8080/saml2/service-provider-metadata/okta  </saml2:Audience>  </saml2:AudienceRestriction>  </saml2:Conditions>  <saml2:AuthnStatement AuthnInstant="2022-09-26T16:24:34.458Z" SessionIndex="id1664209474456.2017179619" xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion">  <saml2:AuthnContext>  <saml2:AuthnContextClassRef>  urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport  </saml2:AuthnContextClassRef>  </saml2:AuthnContext>  </saml2:AuthnStatement>  </saml2:Assertion>  </saml2p:Response> |  | POST of the SAML response.  I am authenticated. |
| 4 | GET http://localhost:8080/ | 200 | I am redirected to my home page so the deep link did not work. |

## Use case 2 – already authenticated

|  |  |  |  |
| --- | --- | --- | --- |
| Step | What | Status | Comment |
| 1 | GET http://localhost:8080/check | 200 | The deep link URL is directly displayed. |

## Use case 3 – after a signed out

|  |  |  |  |
| --- | --- | --- | --- |
| Step | What | Status | Comment |
| 1 | check  GET http://localhost:8080/check | 302  (redirect) | Redirected to the Location in the response headers:  http://localhost:8080/saml2/authenticate/okta |
| 2 | Okta  GET http://localhost:8080/saml2/authenticate/okta | 200 |  |
| 3 | Saml  POST <https://dev-38785465.okta.com/app/dev-38785465_springbootsaml_1/exk6fmgk9u8BTZOg15d7/sso/saml>  The SAML request:  <?xml version="1.0" encoding="UTF-8"?>  <saml2p:AuthnRequest xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol" AssertionConsumerServiceURL="http://localhost:8080/login/saml2/sso/okta" Destination="https://dev-38785465.okta.com/app/dev-38785465\_springbootsaml\_1/exk6fmgk9u8BTZOg15d7/sso/saml" ForceAuthn="false" ID="ARQ05b3b70-d6cf-495a-9457-9ad6ef8400a2" IsPassive="false" IssueInstant="2022-09-26T16:37:05.098Z" ProtocolBinding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST" Version="2.0">  <saml2:Issuer xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion">  http://localhost:8080/saml2/service-provider-metadata/okta  </saml2:Issuer>  </saml2p:AuthnRequest> | 200 | This is the SAML request with a relay state.  This RelayState parameter is meant to be an opaque identifier that is passed back without any modification or inspection.  **RelayState:**  c5228ba0-f3ae-48d4-a54b-7f6e41890e09 |
| 4 | Iframe.html  GET https://login.okta.com/discovery/iframe.html | 200 |  |
| 5 | Devicefingerprint  GET https://dev-38785465.okta.com/auth/services/devicefingerprint | 200 |  |
| 6 | GET https://dev-38785465.okta.com/login/token/redirect?stateToken=02.id.\_zhWnKgFMBvSsZfowAsIIDVIdZZas-x5EqhN6mQL | 200 |  |
| 7 | POST http://localhost:8080/login/saml2/sso/okta | 302  (redirect) | Redirected to the Location in the response headers:  <http://localhost:8080/check>  The payload contains the RelayState.  **RelayState:**  947d1bd8-dac5-4838-bcae-daf2b867b37f |
| 8 | GET http://localhost:8080/check | 200 | Redirected to the page we wanted. The deep link works! |

# Deploy in Google Cloud

Follow the steps here.

<https://codelabs.developers.google.com/codelabs/cloud-app-engine-springboot?continue=https%3A%2F%2Fdevelopers.google.com%2Flearn%2Fpathways%2Fjava-cloud-fundamentals%23codelab-https%3A%2F%2Fcodelabs.developers.google.com%2Fcodelabs%2Fcloud-app-engine-springboot#1>

I am deploying the spring boot app here.

C:\projects\spring-boot-saml-sp2

It is a maven project I have created with the following parameters:

Spring boot 2.7.4

Java

com.example

spring-boot-saml-sp2

Jar

17

Dependencies: Spring Bott DevTools, Spring Security, Spring Web, Thymeleaf

I have created the following project for this on Google Cloud.

Project number: 627438010007 Project ID: spring-boot-saml-sp

I have executed the following commands:

gcloud auth list

gcloud config list project

curl https://start.spring.io/starter.tgz \

-d bootVersion=2.3.0.RELEASE \

-d dependencies=web \

-d baseDir=spring-boot-saml-sp2 | tar -xzvf -

cd spring-boot-saml-sp2

# Deploy in Azure

Cost too much.

# Using VS Code with Google CloudPlatform

<https://medium.com/@alex.burdenko/vs-code-happens-to-be-my-favorite-code-editor-and-ive-been-lucky-to-participate-so-many-diverse-952102856a7a>

I have installed the following extensions in VS code:

Graphical user interface, text, website

Description automatically generated

Text

Description automatically generated

In visual studio terminal I have executed:

gcloud config set project spring-boot-saml-sp

Use the [gcloud beta compute ssh command](https://cloud.google.com/sdk/gcloud/reference/beta/compute/ssh) to connect to a VM that has security keys enabled:

gcloud beta compute ssh instance-1

# Deploy an existing VS Code directory to GitHub

<https://techobservatory.com/how-to-push-code-from-visual-studio-code-to-github/>

Pre-requisites:

Install Git on the PC.

Have a GitHub account.

https://github.com/login

Username : claude-bonnaud

email address: claude.bonnaud.action3d@gmail.com

Password : !SOFTREPO2me!

Log in to your GitHub account.

Click on the button Create a new repository.

My repository with the name spring-boot-saml-sp2 is here:

<https://github.com/claude-bonnaud/spring-boot-saml-sp2>

Open the folder into VS Code.

Click on the Initialize Repository button from the “Source Control” tab in the sidebar.

Graphical user interface, text, application, chat or text message

Description automatically generated

Now, under the Source Control panel, enter any commit message. I’m going to type “First Commit.” Then, click on the ✓ icon next to Source Control heading or press Ctrl + Enter.

When prompted with a pop-up as shown in the image below, choose Yes.

To push the code from VS code to GitHub, Inside the Source Control panel, click on the Overflow button with the ellipsis icon. Then, from the Pull, Push menu option, select Push to.

On the right corner, you will be prompted to add a remote. Click on Add Remote button.

Next, on the top-center, a palette will appear. Paste the copied GitHub repository URL and press Enter.

<https://github.com/claude-bonnaud/spring-boot-saml-sp2>

Then, type any remote name and press Enter.

# Deploy spring boot app on Google Cloud

<https://www.infiproton.com/post/multiple-services-to-deploy-spring-boot-app-in-gcp>

Use the browser to display the Google Cloud console and authenticate yourself if it is not yet done.

https://console.cloud.google.com/

On the left menu select Compute Engine / VM instances.

On the line for the instance-1, expand the SSH menu and select Open in browser window.

This will open a window SSH console into the VM that enables you to install the spring boot app.

Graphical user interface, text, application, email

Description automatically generated

In the console type the following commands.

Update the Ubuntu software packages

sudo apt-get update

Installing the Java 17 on Ubuntu VM.

sudo apt install openjdk-17-jdk openjdk-17-jre

Check the version of java.

java -version

Locate your Java installation on Ubuntu.

update-alternatives --config java

With the location of the Java install on the clipboard, open up the server’s environment file with Nano:

**sudo nano /etc/environment**

Paste the JAVA\_HOME assignment at the bottom of the file:

JAVA\_HOME="/usr/lib/jvm/java-11-openjdk-amd64/bin/java"

Then force the Ubuntu terminal to reload the environment configuration file:

**source /etc/environment**

You should then be able to echo the JAVA\_HOME environment variable in an Ubuntu terminal window:

**echo $JAVA\_HOME**

Install Git.

sudo apt-get install git

git –version

Now, Clone the Spring App from Github.

git clone https://github.com/claude-bonnaud/spring-boot-saml-sp2.git

Change the directory to Spring App

cd spring-boot-saml-sp2

Grant the permission to ./mvnw ( maven wrapper ) to run on the VM

./mvnw package

Change the directory to the target

cd target

Now, Run the Spring Boot App as the background process in the VM

nohup java -jar spring-boot-saml-sp2-0.0.1-SNAPSHOT.jar &

Ctrl-C

To get the process id of that java process run

ps -A |grep java

It displays:

claude\_+ 8265 3.7 16.2 2772548 162740 pts/1 Sl 20:46 0:15 java -jar spring-boot-saml-sp2-0.0.1-SNAPSHOT.jar

claude\_+ 8355 0.0 0.0 5136 704 pts/1 S+ 20:53 0:00 grep java

To kill the java process if you want to.

kill -9 8265

Expose 8080 port on the VM to make the Spring App accessible to the Internet (see the section below).

Verify that Spring Boot App is running:

<http://34.173.212.174:8080>

You can exit the console.

exit

# install java on ubuntu

<https://www.theserverside.com/blog/Coffee-Talk-Java-News-Stories-and-Opinions/How-do-I-install-Java-on-Ubuntu>

# Allow the port 8080 for the VM on Google Cloud

<https://console.cloud.google.com/networking/networkinterfaces/zones/us-central1-a/instances/instance-1?networkInterface=nic0&project=spring-boot-saml-sp&cloudshell=true&tab=APPLICABLE_FIREWALL_POLICIES&analysisTab=CONNECTIVITY_TEST>

On the left menu select VM instances. Click on the 3 vertical dots on the instance-1 line and select View network details.

Graphical user interface, text, application, email

Description automatically generated

Under the FIREWALLS section, expand the vcp-firewall-rules entry, and click on default-allow-http.

Click the edit button, set the 8080 TCP Port and save the config.

Graphical user interface, text, application

Description automatically generated