Long Ngo
Posted on Mar 4, 2023 • Edited on Nov 5, 2023

15

Integrate Spring boot Version 3 with Parameter Store

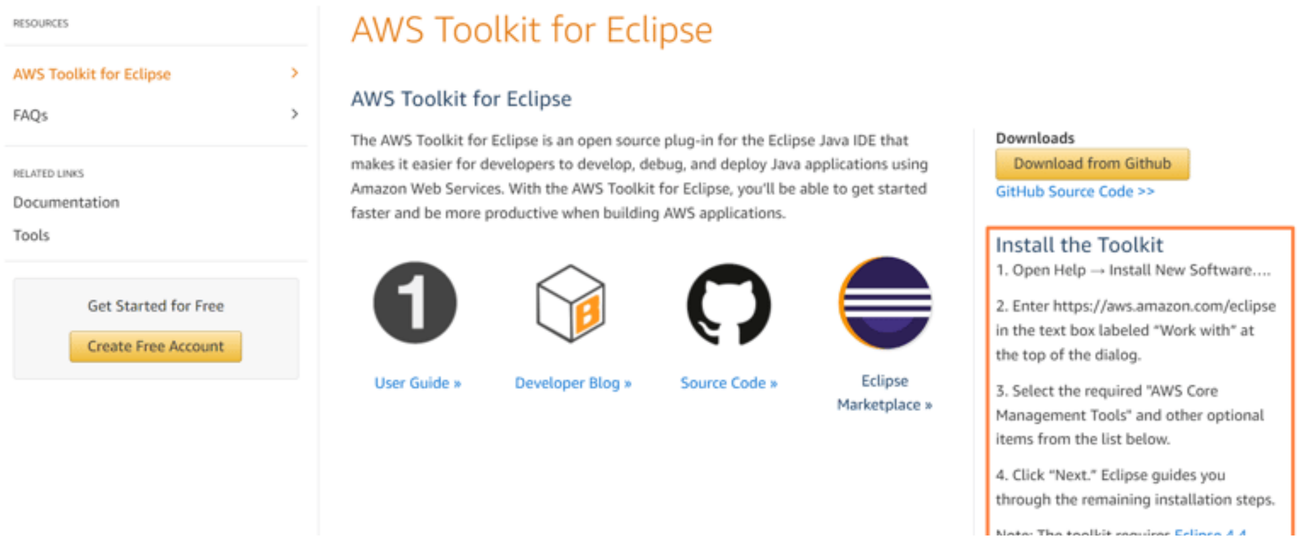
1. Introduction

AWS Parameter Store is a service provided by Amazon Web Services (AWS) that help us store and manage parameters and secrets for our applications. It provides a secure, centralized location for storing and accessing sensitive data such as database credentials, API keys, and configuration data.
So we can use this service to manage multiple deployment property files for a Spring Boot Application.

2. Implementation

2.1. Provide access to the development environment

We need to provide access key and secret key for IDE (I will use Eclipse) to access the Parameter Store feature. You can refer this post [AWS Toolkit for Eclipse](#) and follow these steps to install AWS Toolkit for Eclipse IDE.



AWS Toolkit for Eclipse

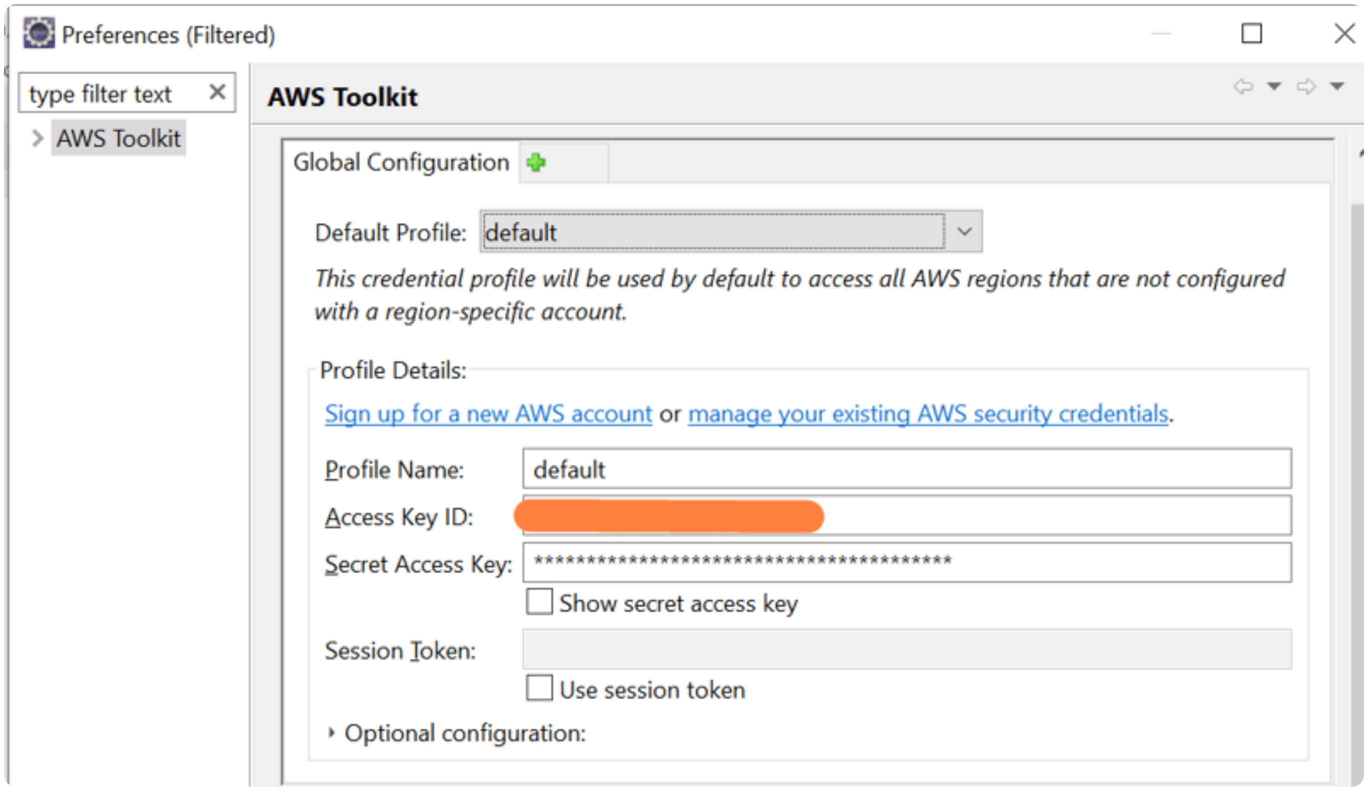
AWS Toolkit for Eclipse

The AWS Toolkit for Eclipse is an open source plug-in for the Eclipse Java IDE that makes it easier for developers to develop, debug, and deploy Java applications using Amazon Web Services. With the AWS Toolkit for Eclipse, you'll be able to get started faster and be more productive when building AWS applications.

Install the Toolkit
1. Open Help → Install New Software....
2. Enter https://aws.amazon.com/eclipse in the text box labeled "Work with" at the top of the dialog.
3. Select the required "AWS Core Management Tools" and other optional items from the list below.
4. Click "Next." Eclipse guides you through the remaining installation steps.

Note: The toolkit requires Eclipse 4.4 (Luna) or higher

After install successfully, we need to add key pair into AWS Toolkit. You can click the Icon of AWS Toolkit on the header and choose Preferences. If you are already had an access key on your computer, it will be display like this.


Preferences (Filtered)

type filter text

AWS Toolkit

AWS Toolkit

Global Configuration

Default Profile: default

This credential profile will be used by default to access all AWS regions that are not configured with a region-specific account.

Profile Details:

Sign up for a new AWS account or manage your existing AWS security credentials.

Profile Name: default

Access Key ID: [redacted]

Secret Access Key: **

Show secret access key

Session Token:

Use session token

Optional configuration:

If you don't want to use this key, you can add manually another key by press the plus button next to the Global Configuration.

2.2. Create a Spring boot project

We can use Spring Starter Project from Spring Tool Suite (can find at Eclipse Marketplace) or Spring Initializer to create a Spring boot project with Spring boot version 3

Then, we should add some dependency that need to use to integrate with Parameter Store.

```
 <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-bootstrap</artifactId>
        </dependency>
        <dependency>
```

```
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-aws-parameter-store-config</artifactId>
            <version>2.2.6.RELEASE</version>
        </dependency>
```

Carefully check you add these dependencies exactly. Because for a Spring Boot version 3, we should use **spring-cloud-starter-bootstrap** to work with Parameter Store. If you don't use it, the issues relative to mismatch version can occur. And finally, we need to use **spring-cloud-dependencies** because Parameter Store work on Spring Cloud.

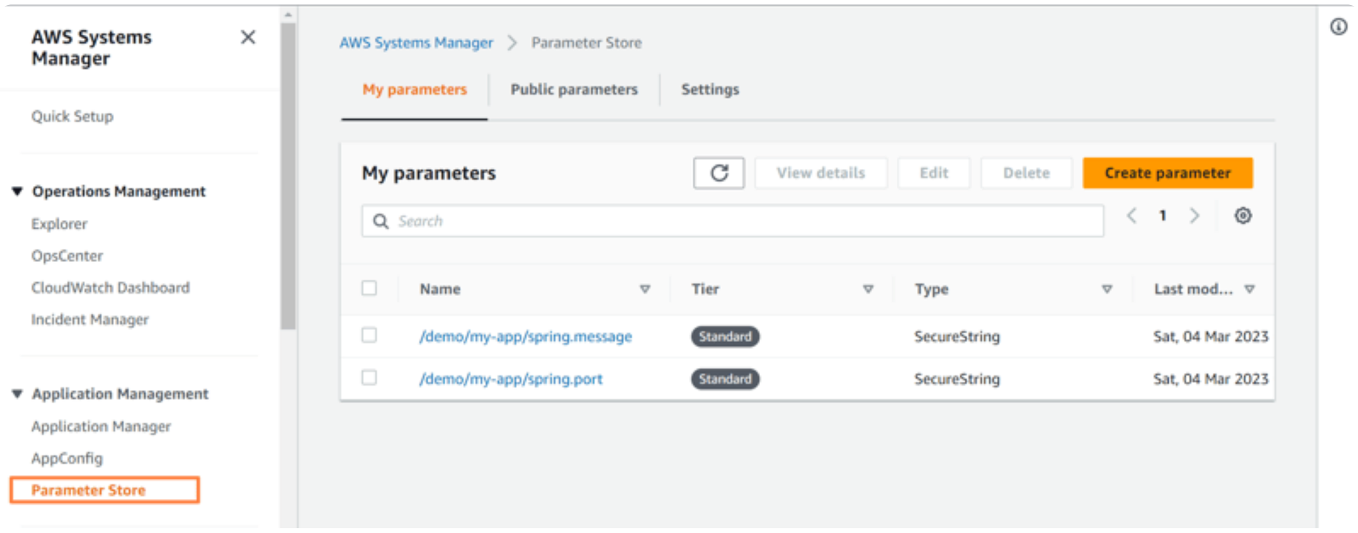
```
<dependencyManagement>
        <dependencies>
            <dependency>
                <groupId>org.springframework.cloud</groupId>
                <artifactId>spring-cloud-dependencies</artifactId>
                <version>2022.0.1</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
        </dependencies>
    </dependencyManagement>
```

2.3. Read the parameter from Parameter Store

We need to define application name in application.properties file. For example is `spring.application.name=my-app`. By default, Spring defined the syntax to work with parameter on Parameter Store with format:
`/config/<name-of-the-spring-application>_<profile>/<parameter-name>`
If the default parameter convention does not fit our needs. We can create a `bootstrap.properties` to override.

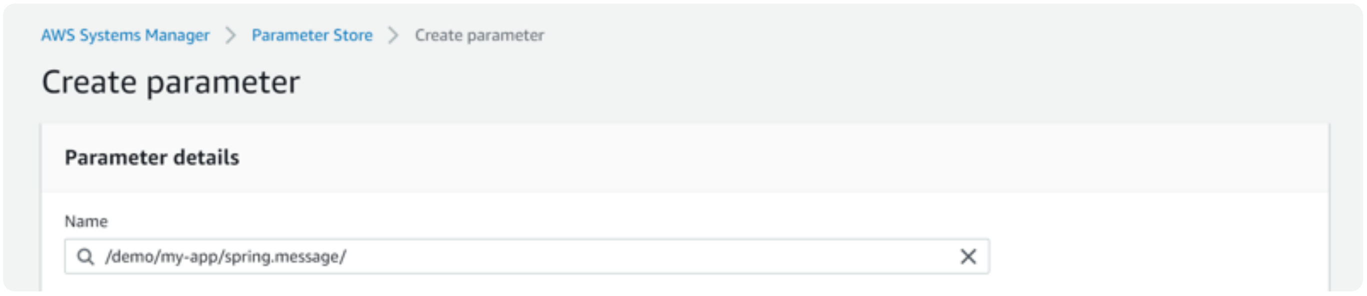
```
aws.paramstore.enabled=true
aws.paramstore.prefix=/demo
aws.paramstore.name=my-app
aws.paramstore.profileSeparator=
```

Now, we will create some parameter and add some code for testing.
In the AWS Console of AWS System Manager, you can choose Parameter and click on Create Parameter button.



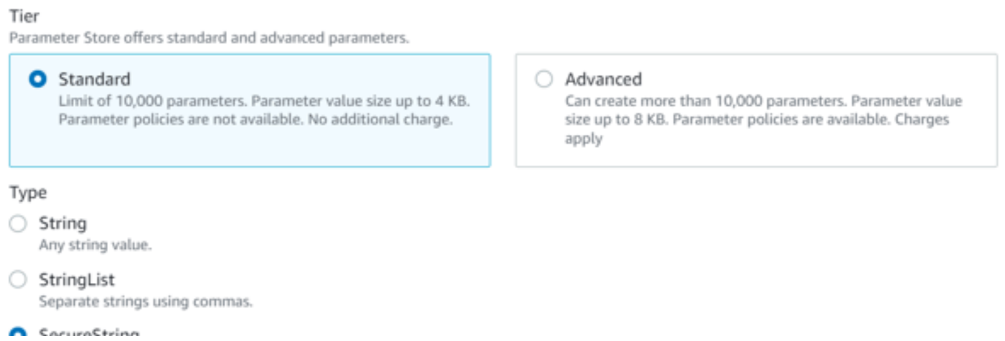
Name	Tier	Type	Last mod...
/demo/my-app/spring.message	Standard	SecureString	Sat, 04 Mar 2023
/demo/my-app/spring.port	Standard	SecureString	Sat, 04 Mar 2023

I already have some parameters in there, so I don't create anymore. You can create by input the name of parameter.



Name: /demo/my-app/spring.message/

Select the type, recommend for SecureString.


Tier
Parameter Store offers standard and advanced parameters.

[✓ Standard] Limit of 10,000 parameters. Parameter value size up to 4 KB. Parameter policies are not available. No additional charge.

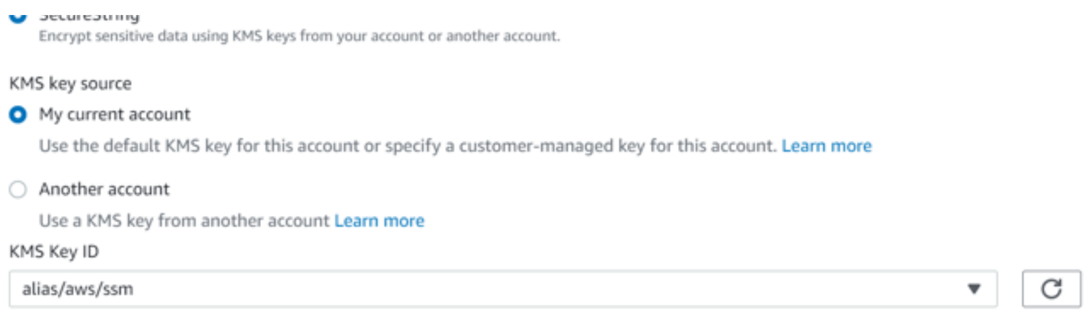
Advanced: Can create more than 10,000 parameters. Parameter value size up to 8 KB. Parameter policies are available. Charges apply

Type

String: Any string value.

StringList: Separate strings using commas.

[✓ SecureString]


SecureString
Encrypt sensitive data using KMS keys from your account or another account.

KMS key source

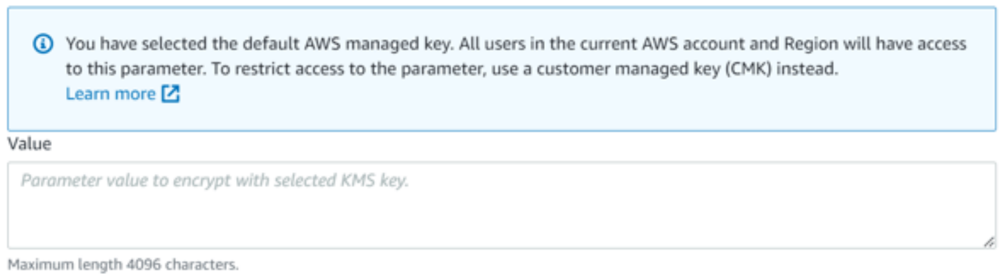
[✓ My current account]
Use the default KMS key for this account or specify a customer-managed key for this account. Learn more

Another account
Use a KMS key from another account Learn more

KMS Key ID

alias/aws/ssm

And finally, input the value for this parameter.


You have selected the default AWS managed key. All users in the current AWS account and Region will have access to this parameter. To restrict access to the parameter, use a customer managed key (CMK) instead.
Learn more

Value

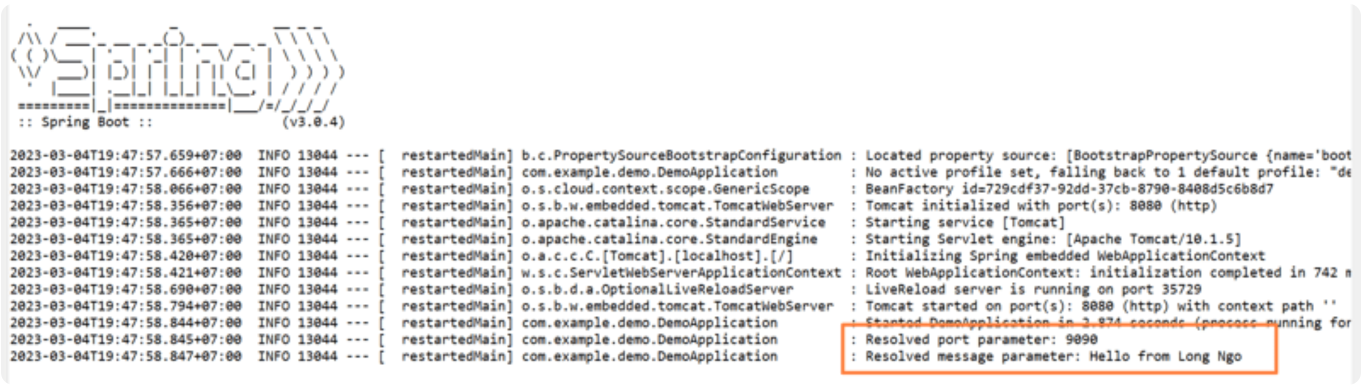
Parameter value to encrypt with selected KMS key.

Maximum length 4096 characters.

For testing purpose, the `@Value` annotation will be used to resolve the value of parameters. We expect the `port` variable will have value of `/demo/my-app/spring.port` and `message` variable corresponding for `/demo/my-app/spring.message` .

```
@SpringBootApplication
@Slf4j
public class DemoApplication implements CommandLineRunner {

    @Value("${spring.port}")
    private String port;

    @Value("${spring.message}")
    private String message;


    public static void main(String[] args) {
        SpringApplication.run(DemoApplication.class, args);

    }

    @Override
    public void run(String... args) throws Exception {

        log.info("Resolved message port: {}", port);
        log.info("Resolved message parameter: {}", message);


    }

}
```

Starting the application and view on the console log, we can see the value that we created on AWS Parameter Store



```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v3.0.4)

2023-03-04T19:47:57.659+07:00  INFO 13044 --- [  restartedMain] b.c.PropertySourceBootstrapConfiguration : Located property source: [BootstrapPropertySource {name='boot
2023-03-04T19:47:57.666+07:00  INFO 13044 --- [  restartedMain] com.example.demo.DemoApplication         : No active profile set, falling back to 1 default profile: "de
2023-03-04T19:47:58.066+07:00  INFO 13044 --- [  restartedMain] o.s.cloud.context.scope.GenericScope     : BeanFactory id=729cdf37-92dd-37cb-8790-8408d5c6b8d7
2023-03-04T19:47:58.356+07:00  INFO 13044 --- [  restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8080 (http)
2023-03-04T19:47:58.365+07:00  INFO 13044 --- [  restartedMain] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2023-03-04T19:47:58.365+07:00  INFO 13044 --- [  restartedMain] o.apache.catalina.core.StandardEngine    : Starting Servlet engine: [Apache Tomcat/10.1.5]
2023-03-04T19:47:58.420+07:00  INFO 13044 --- [  restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2023-03-04T19:47:58.421+07:00  INFO 13044 --- [  restartedMain] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 742 m
2023-03-04T19:47:58.690+07:00  INFO 13044 --- [  restartedMain] o.s.b.d.a.OptionalLiveReloadServer       : LiveReload server is running on port 35729
2023-03-04T19:47:58.794+07:00  INFO 13044 --- [  restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8080 (http) with context path ''
2023-03-04T19:47:58.844+07:00  INFO 13044 --- [  restartedMain] com.example.demo.DemoApplication         : Started DemoApplication in 2.874 seconds (process running for
2023-03-04T19:47:58.845+07:00  INFO 13044 --- [  restartedMain] com.example.demo.DemoApplication         : Resolved port parameter: 9090
2023-03-04T19:47:58.847+07:00  INFO 13044 --- [  restartedMain] com.example.demo.DemoApplication         : Resolved message parameter: Hello from Long Ngo
```

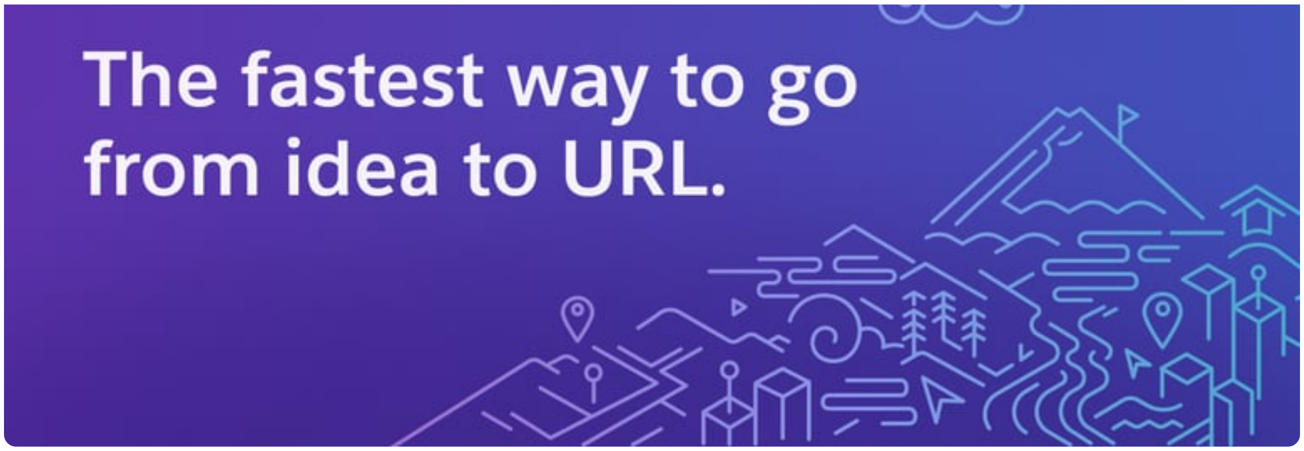
3. Summary

Using the Parameter Store with Spring Boot application, we can manage the value of properties easily. It can work similar to read values from `application.properties` file by `@Value` annotation. Also help to reduce the mismatch issues between development processing in local environment and deploy processing on AWS Cloud.

The implementation of all these examples can be found in my GitHub

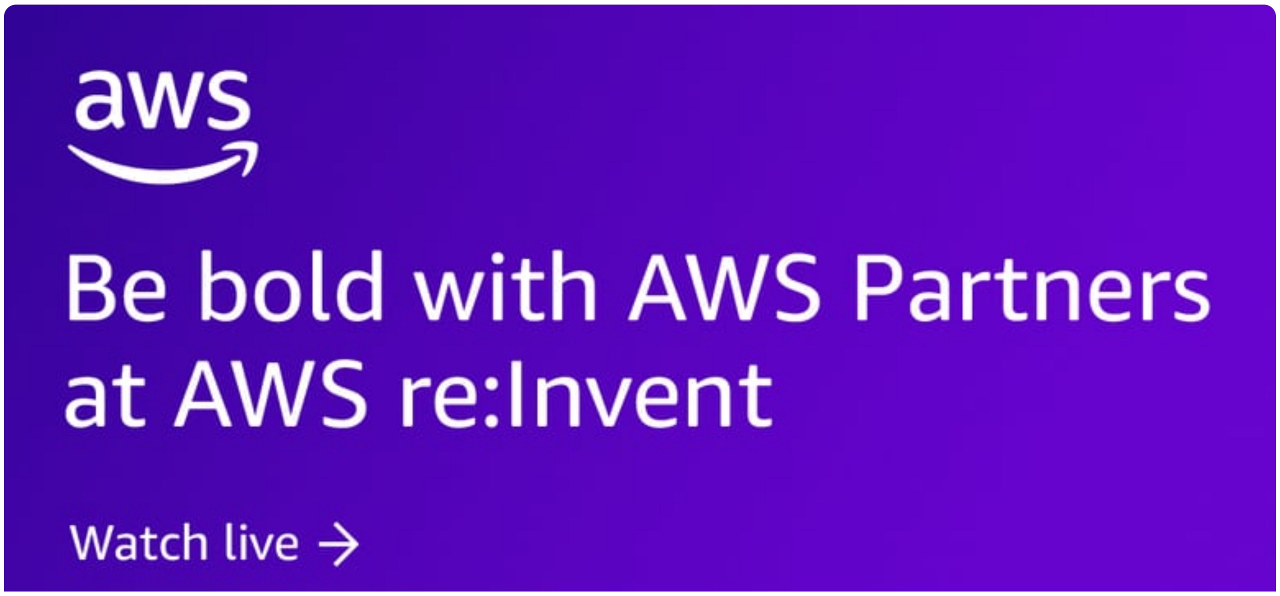
Happy Coding :)

Heroku PROMOTED ...



Top comments (0)

Code of Conduct • Report abuse



Long Ngo

PRONOUNS
He/Him

JOINED
Dec 5, 2022

More from Long Ngo

API Gateway integrate privately with ECS microservice

Lazy load caching strategy example using Redis

Upload large file with Multipart Upload feature

Auth0 PROMOTED

Check out all the new updates to the Free Plan at Auth0
✓ 25k MAUs
✓ Unlimited* Okta Connections and Social Connections
✓ A Custom Domain
And much more...
Auth0 by Okta

Auth0 now, thank yourself later. 😌

Scaling made simple. With up to 25k MAUs and unlimited Okta connections, our Free Plan lets you focus on what you do best—building great apps.

Learn more