Prerequisites:

- JDK 8 or above
- Maven
- SonarQube Server

Steps:

- Create a Maven Project:
- Create a new Maven project to hold your custom rule. Add dependencies for SonarQube plugins.

```
<dependencies>
    <dependency>
        <groupId>org.sonarsource.sonarqube</groupId>
        <artifactId>sonar-plugin-api</artifactId>
        <version>9.3.0.51899 </version>
    </dependency>
</dependencies>
```

Implement Custom Rule:

Create a Java class that implements the custom rule. The class should extend BaseTreeVisitor and implement the rule logic.

```
@Rule(key = "ThreadSleepCheck")
public class ThreadSleepCheck extends BaseTreeVisitor implements JavaFileScanner {
    @Override
    public void visitMethodInvocation(MethodInvocationTree tree) {
        if
("Thread.sleep".equals(ExpressionUtils.methodName(tree).symbolType().name())) {
            reportIssue(tree, "Avoid using Thread.sleep in production code.");
        }
    }
}
```

Register Rule in Plugin:
Create another Java class to register your custom rule with SonarQube.

```
public class MyJavaRulesPlugin implements Plugin {
    @Override
    public void define(Context context) {
        context.addExtension(ThreadSleepCheck.class);
    }
}
```

Package Plugin:
Package your plugin into a JAR file using Maven.

```
mvn clean package
```

Deploy Plugin:

Copy the generated JAR into the extensions/plugins/ directory of your SonarQube server and restart the server.

Activate Rule:

Log in to your SonarQube dashboard, go to 'Quality Profiles', and activate your custom rule.

Run Analysis:

Finally, run a SonarQube analysis on your project to see if the custom rule detects any instances of Thread.sleep.