**Replacing withSonarQubeEnv Plugin with SonarQube API Calls**

This document outlines how to replace the withSonarQubeEnv method from the SonarQube Scanner plugin with direct API calls for running SonarQube scans. This approach provides greater flexibility and eliminates dependency on the plugin.

**Prerequisites**

1. **SonarQube Access**:
   * SonarQube server URL.
   * Credentials (token or username/password) with privileges to trigger scans.
2. **SonarScanner CLI**: Installed on Jenkins build nodes.
3. **SonarQube Project Key**: Unique project keys for each build type.
4. **SonarQube API Token**: Generate from SonarQube under My Account > Security > Generate Tokens.

**Replacing withSonarQubeEnv**

**Steps**

1. **Install SonarScanner CLI**.
2. **Use SonarScanner CLI or APIs** to handle scans and results.
3. **Integrate with Jenkins pipeline**.

**SonarQube Analysis Script**

Create a reusable script (run-sonar-scan.sh) for executing scans with SonarScanner CLI:

#!/bin/bash

# Script to execute SonarQube analysis

# Input Parameters

SONAR\_HOST=$1

SONAR\_TOKEN=$2

PROJECT\_KEY=$3

PROJECT\_NAME=$4

PROJECT\_BRANCH=$5

# Run SonarScanner

sonar-scanner \

-Dsonar.projectKey="${PROJECT\_KEY}" \

-Dsonar.projectName="${PROJECT\_NAME}" \

-Dsonar.branch.name="${PROJECT\_BRANCH}" \

-Dsonar.host.url="${SONAR\_HOST}" \

-Dsonar.login="${SONAR\_TOKEN}" \

$@

**Jenkins Pipeline Changes**

Replace withSonarQubeEnv with a shell execution to call the above script.

**Example for Maven Build**

pipeline {

agent any

environment {

SONAR\_HOST = 'https://your-sonarqube-server.com'

SONAR\_TOKEN = credentials('sonar-api-token')

}

stages {

stage('Build') {

steps {

sh 'mvn clean package'

}

}

stage('SonarQube Scan') {

steps {

script {

def projectKey = "maven-project"

def projectName = "My Maven Project"

def branchName = "main"

sh """

./run-sonar-scan.sh \

$SONAR\_HOST \

$SONAR\_TOKEN \

$projectKey \

$projectName \

$branchName

"""

}

}

}

}

}

**Example for Gradle Build**

pipeline {

agent any

environment {

SONAR\_HOST = 'https://your-sonarqube-server.com'

SONAR\_TOKEN = credentials('sonar-api-token')

}

stages {

stage('Build') {

steps {

sh './gradlew build'

}

}

stage('SonarQube Scan') {

steps {

script {

def projectKey = "gradle-project"

def projectName = "My Gradle Project"

def branchName = "main"

sh """

./run-sonar-scan.sh \

$SONAR\_HOST \

$SONAR\_TOKEN \

$projectKey \

$projectName \

$branchName

"""

}

}

}

}

}

**Example for NodeJS/ReactJS Build**

pipeline {

agent any

environment {

SONAR\_HOST = 'https://your-sonarqube-server.com'

SONAR\_TOKEN = credentials('sonar-api-token')

}

stages {

stage('Build') {

steps {

sh 'npm install'

sh 'npm run build'

}

}

stage('SonarQube Scan') {

steps {

script {

def projectKey = "node-project"

def projectName = "My NodeJS Project"

def branchName = "main"

sh """

./run-sonar-scan.sh \

$SONAR\_HOST \

$SONAR\_TOKEN \

$projectKey \

$projectName \

$branchName

"""

}

}

}

}

}

**Using SonarQube APIs**

**Upload Results**

Use the **/api/ce/submit** endpoint to upload analysis results manually if necessary:

curl -X POST -u "${SONAR\_TOKEN}:" \

-F "projectKey=${PROJECT\_KEY}" \

-F "file=@/path/to/sonar-report.zip" \

"${SONAR\_HOST}/api/ce/submit"

**Monitor Analysis Results**

Use the **/api/ce/task** endpoint to monitor the status of an analysis task:

curl -u "${SONAR\_TOKEN}:" \

"${SONAR\_HOST}/api/ce/task?id=TASK\_ID"

Replace TASK\_ID with the value returned in the response of the /submit API call.

**Tailored Configurations for Build Types**

**Maven**

mvn sonar:sonar \

-Dsonar.projectKey=my-maven-project \

-Dsonar.host.url=https://your-sonarqube-server.com \

-Dsonar.login=$SONAR\_TOKEN

**Gradle**

./gradlew sonarqube \

-Dsonar.projectKey=my-gradle-project \

-Dsonar.host.url=https://your-sonarqube-server.com \

-Dsonar.login=$SONAR\_TOKEN

**NodeJS/ReactJS**

sonar-scanner \

-Dsonar.projectKey=my-node-project \

-Dsonar.sources=src \

-Dsonar.host.url=https://your-sonarqube-server.com \

-Dsonar.login=$SONAR\_TOKEN

**Benefits of API-Based Approach**

1. **Flexibility**: Decouple pipelines from the plugin version.
2. **Customizability**: Add additional logic such as custom dashboards or notifications.
3. **Security**: Manage credentials at a granular level.

Let me know if additional details are required!