SONARQUBE WITH TERRAFORM

# 1) Can SonarQube be used for TERRAFORM IAC?

Yes, In the 9.2 release, SonarQube added support for analyzing CloudFormation and Terraform files.

With these two new languages, SonarQube helps developers secure not just their code, but also their deployments. Because just moving to the cloud doesn't make your application secure. While Azure manages the security of the cloud; it's still up to you to manage what you're putting there. And that means securing not just the code but also how it's deployed.

Among the domains for both CloudFormation and Terraform are the security of your Azure Storage, permissions, traceability, and encryption at rest and at transit.

Additionally, import is supported for reports from Cfn-Lint, Amazon's official CloudFormation linter. The domains for Azure Cloud Terraform analysis include security at rest and at transit, Azure Active Directory, Azure Resource Manager and public network access.

# 2) what kind of features we can achieve via SonarQube TERRAFORM plugin?

**Who's responsible for security?**

Security is a shared responsibility between the cloud provider - such as Amazon Web Services (AWS), Microsoft Azure or Google Cloud Platform (GCP) - and the customer.

In this 'shared model', the cloud provider is responsible for 'security **OF** the cloud,'.

This means the cloud providers are responsible for securing the traditional compute services such as physical hosts, networking and virtualization.

Customers are responsible for 'security **IN** the cloud' and that means platform and resource configuration since that’s under your direct control. When you spin up cloud infrastructures, you're directly controlling the operating environment - this is true whether your instances are server-based or serverless.

Not all cloud developers are aware of this and/or comprehend the significance. One small mistake can expose a lot! The good news is that IaC is just code and at SonarSource, we know a thing or two about helping folks write clean code!

This includes the popular languages and tools you’re using to configure and orchestrate your cloud infrastructures.

**Securing IaC with Sonar**

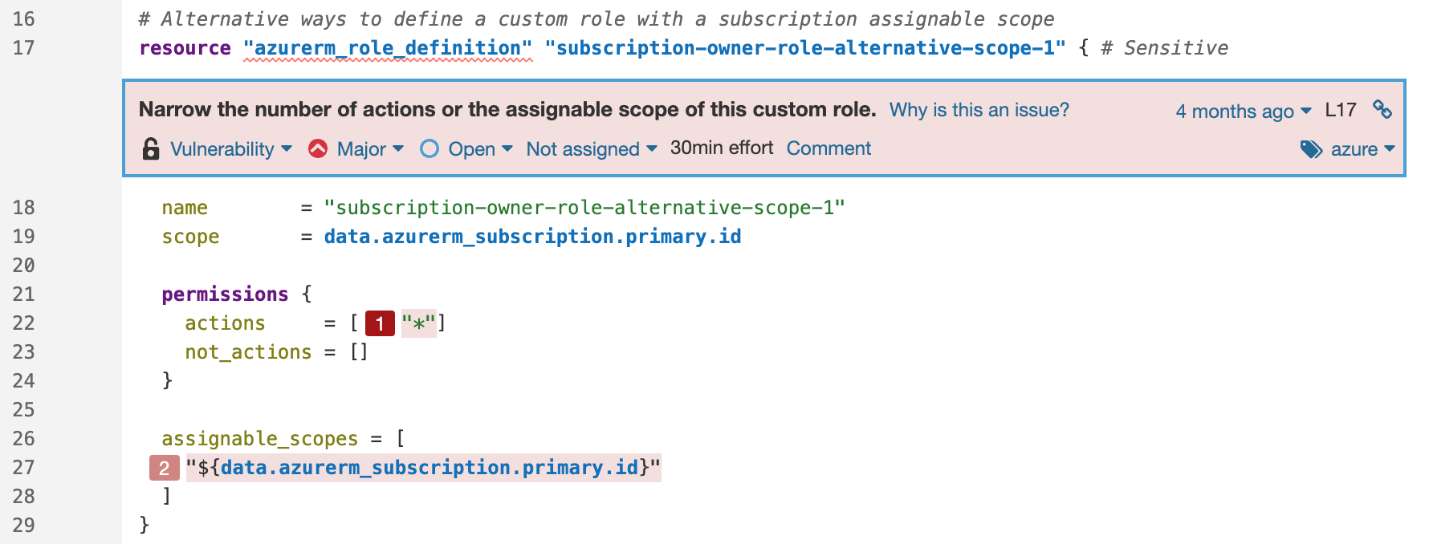
There are added rules to the Sonar solution to detect code smells, bugs and vulnerabilities in your IaC projects. If you’re already using IaC in production or just exploring what it can bring, you’re probably developing cloud-native apps and this is where Sonar really adds value.

With Sonar, you can find and fix issues in your IaC and Sonar can scan the source code in your cloud-native apps as well.

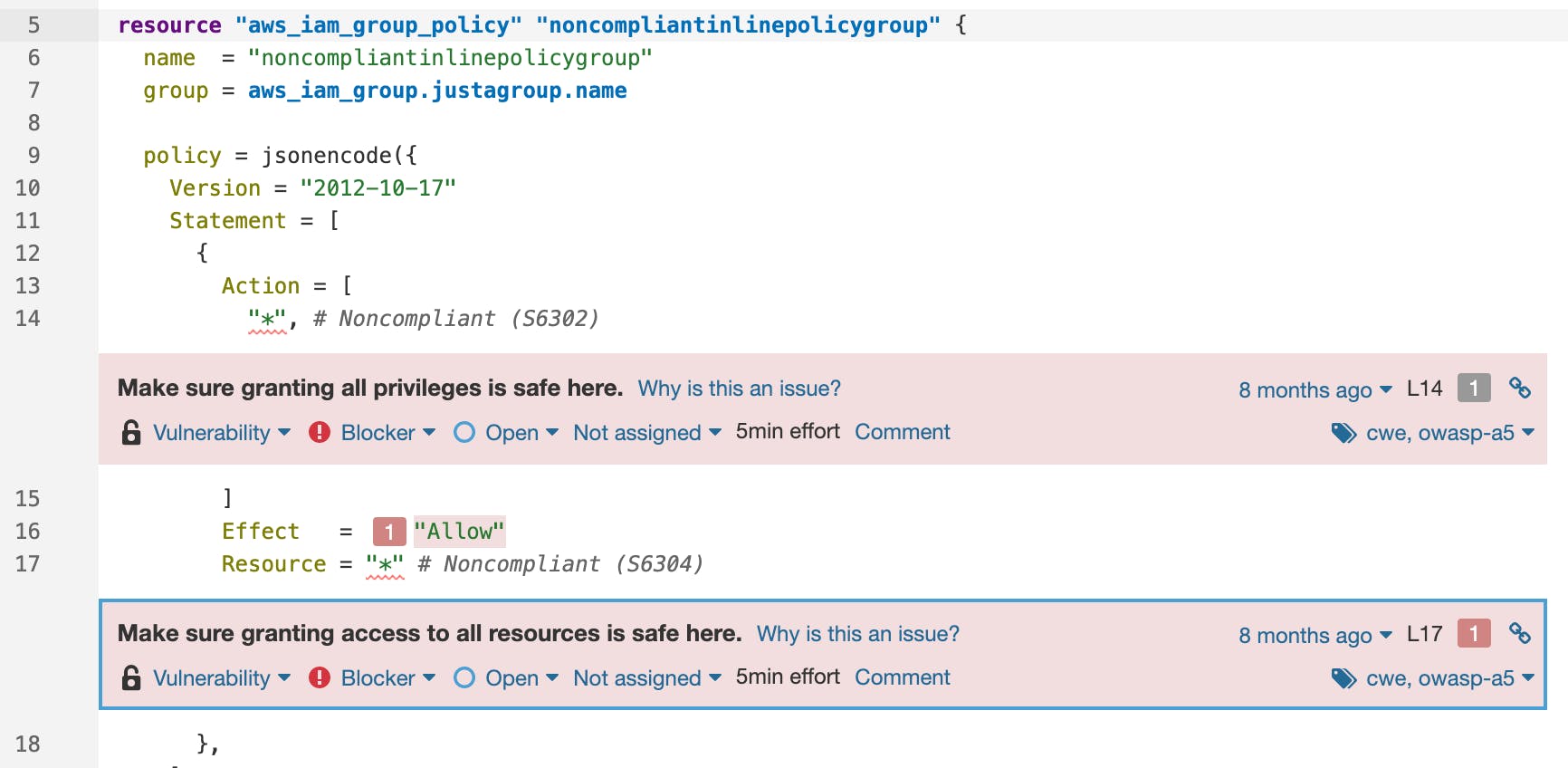
Whether it’s JavaScript running on the backend or Azure functions as part of your microservice, Sonar detects quality and security issues and helps you fix them.

Below are a couple of issue examples caught by our IaC specific rules:

**1) Scope permission vulnerability in Azure with a secondary location**



**2) Authentication vulnerability in AWS**



We know that folks don’t always deploy with a single cloud provider so we have rule coverage for AWS, Azure and Google platforms.

# 3) Configuration steps

There is nothing to set up except a normal analysis with sonar-scanner. This Analyzer searches for .tf files, analyzes them, and shows the results as for any other code.

**Prerequisites:**

The only prerequisite for running SonarQube is to have Java (Oracle JRE 11 or OpenJDK 11) installed on your machine.

**Downloads Sonarqube and Sonar Scanner:**

* LTS version of sonarqube (<https://www.sonarqube.org/downloads>)
* Download sonar-scanner based on your platform (<https://docs.sonarqube.org/latest/analysis/scan/sonarscanner/>)

After the above process, Add the sonar scanner path to environment variables. The SonarScanner is the scanner to use when there is no specific scanner for your build system.

**Run Sonarqube server:**

#For Windows(cmd):

C:\sonarqube\bin\windows-x86-64>StartSonar.bat

#For other OS (terminal):

C:\sonarqube\bin\[OS]>sonar.sh

Once the sonar server up successfully, then Log in to [http://localhost:9000](http://localhost:9000/) with System Administrator credentials (login=admin, password=admin).

**Install a Plugin**

There are two ways to install plugins in SonarQube:

* **Marketplace** - With Community Edition, you can use Marketplace to automatically install plugins from the SonarQube. With commercial editions, you can browse plugins in the Marketplace, but you need to manually install and update your plugins.

If your instance has internet access and you're connected with a SonarQube user with the **Administer System** global permission, you can find the Marketplace at **Administration > Marketplace**. From here:

* Find the plugin you want to install
* Click **Install** and wait for the download to be processed

Once the download is complete, a **Restart** button will be available to restart your instance.

* **Manual Installation** - You need to manually install plugins when using commercial editions of SonarQube.

You can also manually install plugins if your SonarQube instance doesn't have internet access or the plugin you're installing isn't in the Marketplace.

To manually install a plugin:

* Download the plugin you want to install. The version needs to be compatible with your SonarQube version.
* Put the downloaded jar in $SONARQUBE\_HOME/extensions/plugins, and remove any previous versions of the same plugins.
* Restart your SonarQube server.

**Generating and Using Tokens**

**Types of Tokens**

**User Tokens**

These tokens can be used to run analysis and to invoke web services, based on the token author's permissions.

**Project Analysis Tokens**

These tokens can be used to run analysis on a specific project.

In order to create this type of token, the user should have Global Execute Analysis permission or Execute Analysis permission on the token's associated project.

If the token's author loses Execute Analysis permissions for the associated project, the token will no longer be valid for performing an analysis.

The usage of Project Analysis Tokens is encouraged for security reasons.

If such a token were to leak, an attacker would only gain access to analyze a single project or to interact with the related web services requiring Execute Analysis permissions.

**Global Analysis Tokens**

These tokens can be used to run analysis on every project.

In order to create this type of tokens, the user should have Global Execute Analysis Permission.

If the token's author loses the Global Execute Analysis permission, the token will no longer be valid for performing an analysis.

**Generating a token**

You can generate new tokens at **User > My Account > Security**.

The form at the top of the page allows you to generate new tokens, specifying their token type.

Once you click the **Generate** button, you will see the token value. Copy it immediately; once you dismiss the notification you will not be able to retrieve it.

**Revoking a token**

You can revoke an existing token at **User > My Account > Security** by clicking the **Revoke** button next to the token.

**Using a token**

User tokens must replace your normal login process in the following scenarios:

* when running analyses on your code: replace your login with the token in the sonar.login property.
* when invoking web services: just pass the token instead of your login while doing the basic authentication.

In both cases, you don't need to provide a password (so when running analyses on your code, the property sonar.password is optional).

Using a token is the preferred method over using a login and password.

**Provider configuration**

The sonarqube provider is used to configure sonarqube. The provider needs to be configured with a url, and either with user and password or token.

**Example: Authenticate with username and password**

terraform {

required\_providers {

sonarqube = {

source = "jdamata/sonarqube"

}

}

}

provider "sonarqube" {

user = "admin"

pass = "admin"

host = "http://127.0.0.1:9000"

}

**Example: Authenticate with token**

terraform {

required\_providers {

sonarqube = {

source = "jdamata/sonarqube"

}

}

}

provider "sonarqube" {

token = "d4at55a6f7r199bd707h39625685510880gbf7ff"

host = "http://127.0.0.1:9000"

}

**[Argument Reference](https://registry.terraform.io/providers/jdamata/sonarqube/latest/docs" \l "argument-reference)**

The following arguments are supported:

* **user** - (Optional) Sonarqube user. This can also be set via the SONARQUBE\_USER environment variable.
* **pass** - (Optional) Sonarqube pass. This can also be set via the SONARQUBE\_PASS environment variable.
* **token** - (Optional) Sonarqube token. This can also be set via the SONARQUBE\_TOKEN environment variable.
* **host** - (Required) Sonarqube url. This can be also be set via the SONARQUBE\_HOST environment variable.
* **installed\_version** - (Optional) The version of the Sonarqube server. When specified, the provider will avoid requesting this from the server during the initialization process. This can be helpful when using the same Terraform code to install Sonarqube and configure it.
* **tls\_insecure\_skip\_verify** - (Optional) Allows ignoring insecure certificates when set to true. Defaults to false. Disabling TLS verification is dangerous and should only be done for local testing.

**Configuring your project**

Create a configuration file in your project's root directory called sonar-project.properties

# must be unique in a given SonarQube instance

sonar.projectKey=my:project

# --- optional properties ---

# defaults to project key

#sonar.projectName=My project

# defaults to 'not provided'

#sonar.projectVersion=1.0

# Path is relative to the sonar-project.properties file. Defaults to .

#sonar.sources=.

# Encoding of the source code. Default is default system encoding

#sonar.sourceEncoding=UTF-8

**Running SonarScanner from the zip file**

**SonarScanner (v4.7)**

[Linux 64-bit](https://binaries.sonarsource.com/Distribution/sonar-scanner-cli/sonar-scanner-cli-4.7.0.2747-linux.zip)

[Windows 64-bit](https://binaries.sonarsource.com/Distribution/sonar-scanner-cli/sonar-scanner-cli-4.7.0.2747-windows.zip)

[Mac OS X 64-bit](https://binaries.sonarsource.com/Distribution/sonar-scanner-cli/sonar-scanner-cli-4.7.0.2747-macosx.zip)

[Docker](https://hub.docker.com/r/sonarsource/sonar-scanner-cli)

[Any (Requires a pre-installed JVM)](https://binaries.sonarsource.com/Distribution/sonar-scanner-cli/sonar-scanner-cli-4.7.0.2747.zip)

To run SonarScanner from the zip file, follow these steps:

1. Expand the downloaded file into the directory of your choice. We'll refer to it as $install\_directory in the next steps.
2. Update the global settings to point to your SonarQube server by editing $install\_directory/conf/sonar-scanner.properties:

#----- Default SonarQube server

#sonar.host.url=http://localhost:9000

1. Add the $install\_directory/bin directory to your path.
2. Verify your installation by opening a new shell and executing the command sonar-scanner -h (sonar-scanner.bat -h on Windows).

You should get output like this:

usage: sonar-scanner [options]

Options:

-D,--define <arg> Define property

-h,--help Display help information

-v,--version Display version information

-X,--debug Produce execution debug output

If you need more debug information, you can add one of the following to your command line: -X, --verbose, or -Dsonar.verbose=true.

1. Run the following command from the project base directory to launch analysis and pass your **authentication token**:

sonar-scanner -Dsonar.login=myAuthenticationToken

After the analysis is done, you can either browse the provided link to see the sonar report directly [Ex.: <http://localhost:9000/dashboard?id=1>] or go to the project section to see the newly generated sonar report of your project.

# 4) Is there any similar tool available for the same objectives?

Below tools are present for same objectives as static analysis tool.

* Mend (<https://www.mend.io/>)
* Terrascan (<https://github.com/accurics/terrascan>)
* TFLint (<https://github.com/terraform-linters/tflint>)
* oak9 ([www.oak9.io](http://www.oak9.io)) - It focuses on securing your application architecture by analyzing your IaC. It has a number of features including out-of-the-box security and compliance blueprints, integrations across CI/CD toolsets & code-repositories, integrations with different cloud service providers and a lot more.

# 5) Community Support

<https://www.sonarqube.org/community/>

Need to ask a question, report a bug or discuss a feature? Visit our community forum!

**Community Forum**

<https://community.sonarsource.com/>

This is a public forum moderated by SonarSource staff when they have time.

The goal of this community is to give you a place to discuss getting up and running with SonarSource products and then making the most of them for Code Quality and Code Security in your workflows.

# 6) Use Cases

We use SonarQube in our project to basically calculate the code quality report mostly, in that report we test for the bugs, vulnerabilities, code smells, code issues, criticals, blockers, major & minor issues, and also calculate the code coverage of junits.

We also set the quality profile which contains the rules which we set according to the rules we follow in our project and on that basis, we generate the junit coverage report.

* Generating code quality report
* Calculates junit coverage of the codebase very efficiently and precisely
* Highlights the bugs and vulnerabilities in our codebase
* Informs the user of the improvements which can be done to the code to make it cleaner
* SonarQube also suggests remediation and resolution of the problems it highlights

**SonarQube is a flexible tool that supports different use cases in terms of source code assessment:**

* SonarQube is able to analyze projects in many different programming languages
* The provided software may come as pure source code or with a build management system like Maven or Gradle
* Projects can be added as SonarQube stand-alone or with an additional continuous integration (CI) tool involved which is recommended (e.g., Bamboo or Jenkins)

**Sonar covers the 7 sections of code quality:**

1. **Sustainability** - Reduces complexity, possible vulnerabilities, and code duplications, optimizing the life of applications.
2. **Increase productivity** - Reduces the scale, cost of maintenance, and risk of the application; as such, it removes the need to spend more time changing the code
3. **Quality code** - Code quality control is an inseparable part of the process of software development.
4. **Detect Errors** - Detects errors in the code and alerts developers to fix them automatically before submitting them for output.
5. **Increase consistency** - Determines where the code criteria are breached and enhances the quality
6. **Business scaling** - No restriction on the number of projects to be evaluated
7. **Enhance developer skills** - Regular feedback on quality problems helps developers to improve their coding skills

# 7) Assumption

Analysis doesn’t do any downloading. The assumption is that analysis is being triggered from the project root directory and that all relevant code is available in the current directory or in child directories.

# 8) Limitations

* A downtime of the cluster has to be accepted when performing SonarQube upgrade or plugins installations
* There is no way to perform actions on the cluster from a central app - all operations have to be done manually on each node of the cluster
* All application nodes must be stopped when installing, uninstalling or upgrading a plugin
* Plugins are not shared, it means if you install/uninstall/upgrade a given plugin in one application node, you need to do the same actions on the other application node
* Importing a new custom quality profile on SonarQube is a bit tricky, it can be made easier
* Every second time when we want to rerun the server, we have to restart the whole system, otherwise, the server stops and closes automatically