Continuous Integration

Tuleap provides Continuous Integration to teams via a deep integration with Jenkins.

Tuleap team recommends to avoid big Jenkins instances shared by many projects and many teams as the security model of Jenkins doesn't allow to have a strict split of areas. That is to say, to guaranty that 2 concurrent teams that share the same Jenkins sever cannot have access to the code of each other.

In case of doubt the strategy of "One Tuleap Project, one Jenkins Master" is safe and efficient.

Jenkins Configuration

> Attention
>
> The currently minimal Jenkins version is **2.361.4**
>
> The currently minimal Tuleap version should be **12.11**
>
> Both Jenkins and Tuleap servers **must** be in https and certificate must be either valid or, at least, trusted.
>
> If you cannot have valid certificates:
>
> - Jenkins must trust Tuleap server certificate
> - Jenkins must run behind a reverse proxy that does TLS termination
> - Tuleap must trust Jenkins server certificate

This section will cover how to configure a Jenkins server to be used efficiently with Tuleap. We assume a fresh Jenkins instance that was just installed.

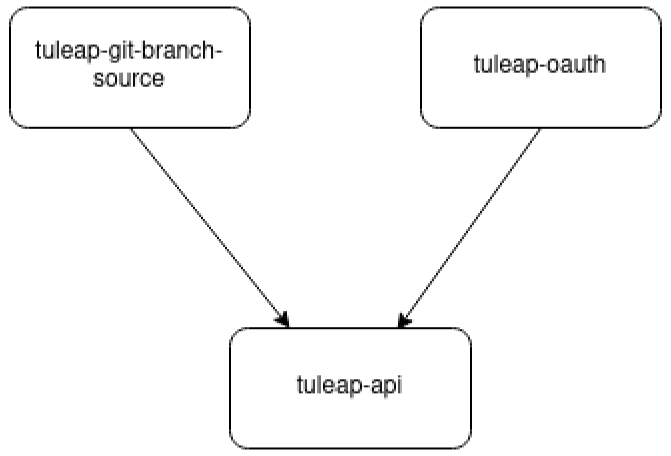
Some adaptations might be needed if you modify an existing Jenkins server (be very careful with authentication to not lock yourself out of Jenkins).

PLUGINS INSTALLATION

Three plugins should be installed:

- Tuleap Git Branch Source
- Tuleap Authentication
- Tuleap API

`Tuleap API` is the plugin responsible for calling the Tuleap API. `Tuleap Git Branch Source` is responsible for easing and automating the discovery and job creation for repositories, branches, pull requests (PRs) in Git respositories hosted on your Tuleap instance. `Tuleap Authentication` is responsible for delegating the authentication of Jenkins users to your Tuleap instance.



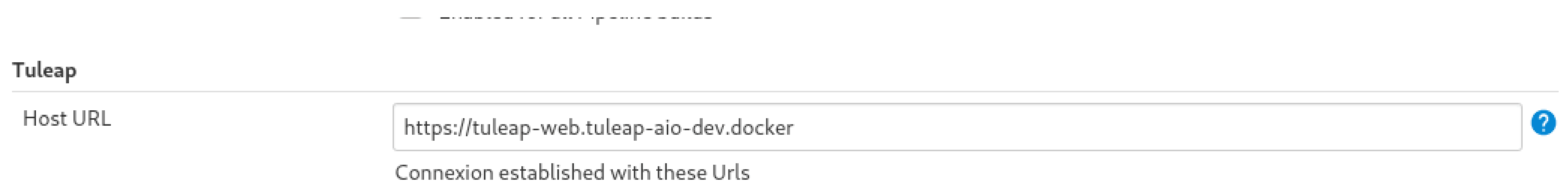
> `Tuleap API` is automatically installed if you install `Tuleap Git Branch Source` or `Tuleap Authentication`.

Ideally, you will want to install the Pipeline plugin as well, as the targeted workflows make an extensive use of `Jenkinsfile`. The examples provided below assume you have this plugin installed on your Jenkins instance.

Finally, you might also need to install other plugins related to your pipeline of email notifications, artifact publishing to Artifactory, etc. This is business specific to each project and not covered in this documentation.

PLUGINS CONFIGURATION

The plugins configuration can be done thanks to Tuleap API plugin. First, you need to associate your Jenkins server with a Tuleap server. This is done in “Manage Jenkins > System configuration”.



Tuleap

Host URL: https://tuleap-web.tuleap-aio-dev.docker

Connexion established with these Urls

If the connection to the Tuleap server is successful you will see `Connexion established with these Urls` message in the Jenkins interface (as in the previous screenshot). Otherwise you will get a stacktrace (Jenkins…) with, hopefully, an error message that will help to diagnose the problem.

Most common issues are related to DNS (is your server name valid and can Jenkins access it) and TLS (does jenkins trust Tuleap server).

Pull request notification configuration

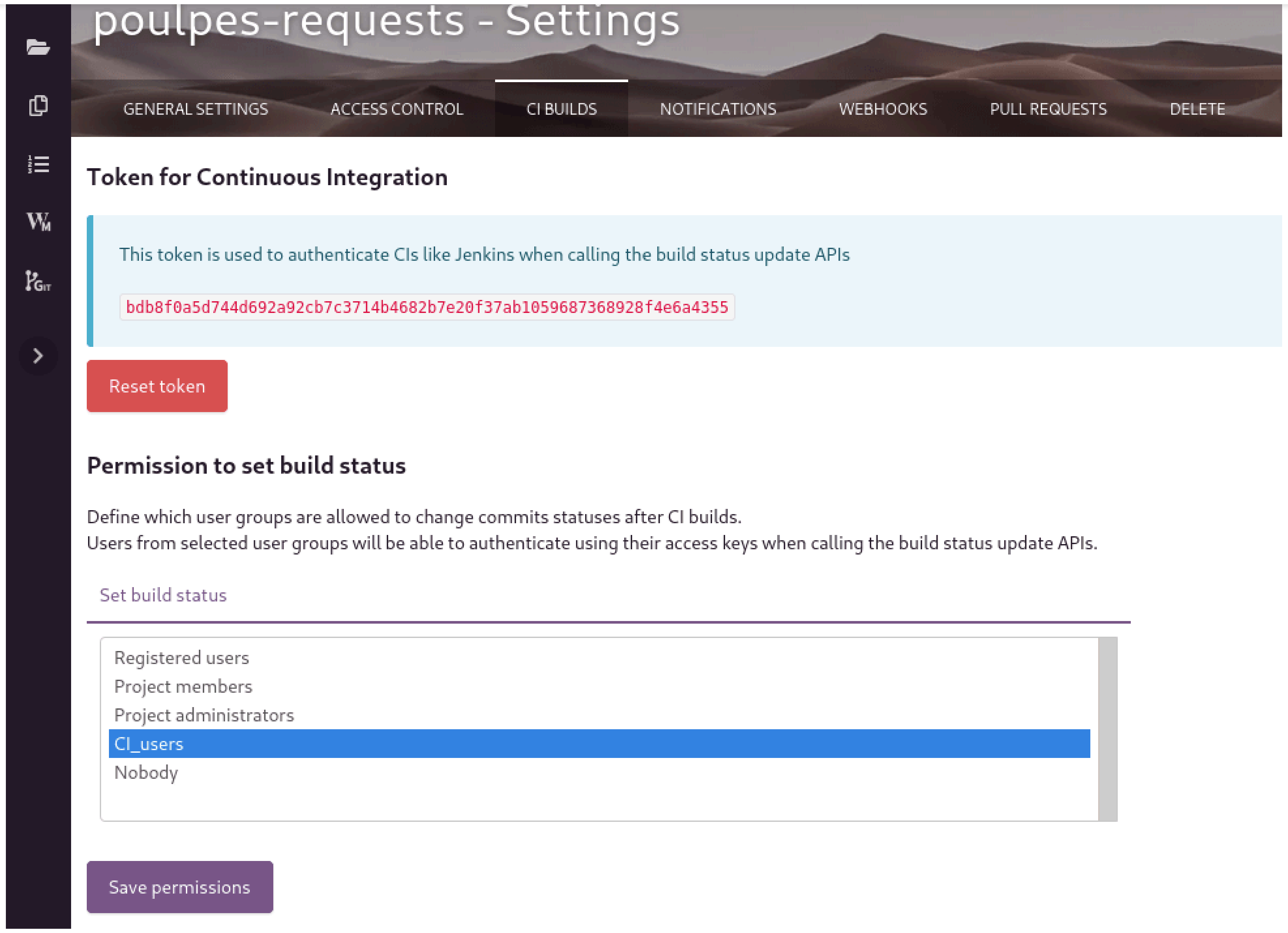
Tuleap API Plugin provides the pipeline command to send the build result of your Jenkins job to a pull request in Tuleap.

Before using the command, you will need to provide Jenkins a way to authenticate on Tuleap’s side to ensure that it’s your Jenkins’s job that recorded the build status.

There are currently two ways to proceed:

- Either using the CI token of the repository
- Or using an access key of a user belonging to a user group authorized to set build statuses.

To do so, go into repository *Settings > CI Builds*:



Token for Continuous Integration

This token is used to authenticate CIs like Jenkins when calling the build status update APIs

`bdb8f0a5d744d692a92cb7c3714b4682b7e20f37ab1059687368928f4e6a4355`

Permission to set build status

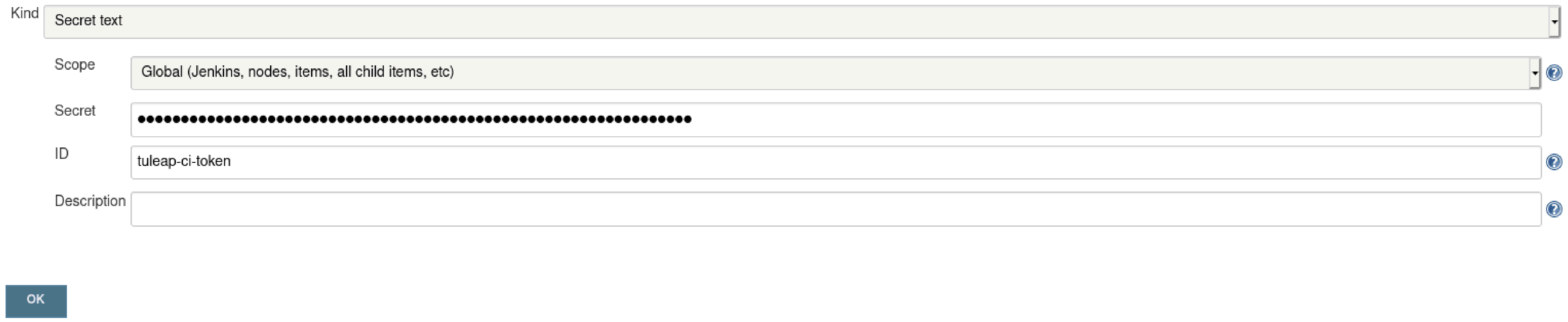
Define which user groups are allowed to change commits statuses after CI builds.
Users from selected user groups will be able to authenticate using their access keys when calling the build status update APIs.

Registered users
Project members
Project administrators
CI_users
Nobody

USING THE CI TOKEN

The CI token has to be saved on Jenkins's side as a `Secret text`. In our example we will add the credential in the "global" domain. In Jenkins, go to *Credentials > global > Add Credentials*, select the `Secret text` type and fill in the inputs:

- `Secret` must contains the CI token retrieved previously.
- `ID` input is the ID that you want for the credential, the value will be used in the Jenkinsfile. Here it will be `tuleap-ci-token`. If you do not fill this field, you have to use the credential ID generated by Jenkins instead
- `Description` is optional. This field will not be used in our example.



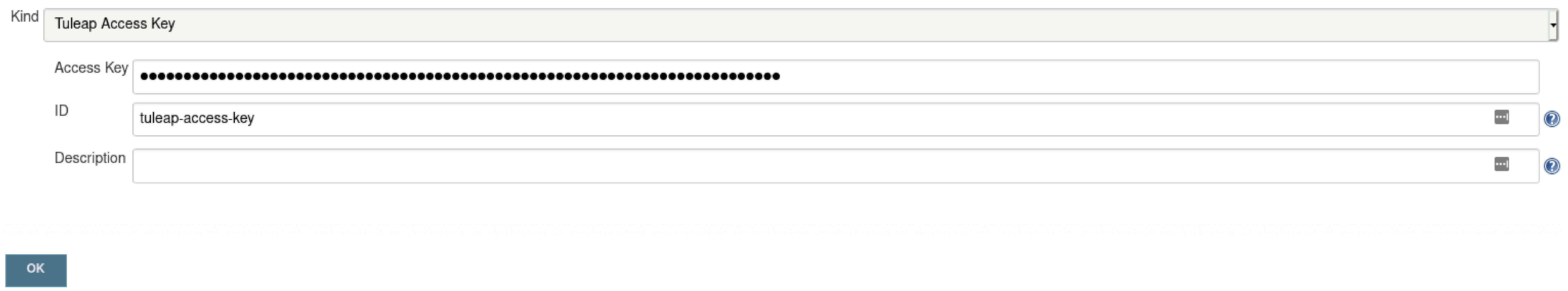
Kind: Secret text
Scope: Global (Jenkins, nodes, items, all child items, etc)
Secret: ••
ID: tuleap-ci-token
Description:

See the Credentials plugin documentation for more information about credentials management in Jenkins.

USING AN ACCESS KEY

In Jenkins, go to *Credentials > global > Add Credentials*, select the `Tuleap Access Key` type and fill in the inputs:

- `Description` is optional. This field will not be used in our example.



Kind: Tuleap Access Key

Access Key: ••

ID: tuleap-access-key

Description:

> Note
>
> If you want to use the same access key as the one provided during the registration of your repository on Jenkins, no need to follow the steps above since it is already registered.

Make sure that the user owning the access key is member of one of the groups having the permission to set builds statuses (see *Settings > CI Builds* in your repository on Tuleap).

SEND BUILDS RESULTS

To send the result you have to add the following command in your Jenkinsfile:

```
tuleapNotifyCommitStatus status: 'BUILD_STATUS',
                         repositoryId: 'GIT_REPOSITORY_ID',
                         credentialId: 'CRED_ID_OF_CI_TOKEN_OR_ACCESS_KEY'
```

Where:

- `BUILD_STATUS` is either 'pending', 'failure' or 'success'.
- `GIT_REPOSITORY_ID` is the ID of the concerned git repository. The ID can be found in the URL query (repo_id=XXX) in the 'Settings' menu of the repository.
- `CRED_ID_OF_CI_TOKEN_OR_ACCESS_KEY` is the Jenkins credential ID of the CI token OR the access key of a user who has the permission to set builds statuses on your repository. In the example below, it is `b9a4832d-f26d-46ca-acd7-15b935de9069`

Here is an example Jenkinsfile using the tuleapNotifyCommitStatus command:

```
// The git repository id is 1
pipeline {
    // Add you own build/test stages instead of this
    stages {
        stage('Build') {
            steps {
                tuleapNotifyCommitStatus status: 'pending', repositoryId: '1', credentialId: 'b9a4832d-f26d-46ca-acd7-15b935de9069'

                sh('make all')
            }
        }
    }
    post {
        failure {
            tuleapNotifyCommitStatus status: 'failure', repositoryId: '1', credentialId: 'b9a4832d-f26d-46ca-acd7-15b935de9069'
        }
        success {
            tuleapNotifyCommitStatus status: 'success', repositoryId: '1', credentialId: 'b9a4832d-f26d-46ca-acd7-15b935de9069'
        }
    }
}
```

The concerned Pull Request is updated with the last build status from Jenkins:

Jack Peralta (peraltaj) 2020-09-14 11:08 +0 -0

9e419faf4a8cc2614ae536e995fa0528dad44b01

pr1 → master

Last CI Status

✖ Failure on 2020-09-18 16:16

See Pull requests documentation for more information.

Jenkins configuration for test automation

First of all you will need an Access Key to access Tuleap API. Generate one for your user and then create a new credential in Jenkins selecting “Tuleap Access Key” as “Kind” and giving it an ID or copying the one generated by Jenkins. For this example we’ll consider that you chose `tuleap-access-token` as the ID.

Create a new Jenkins job “Pipeline” and point it to your SCM repository (you might want to use `jenkins-tuleap-bot` credentials to access the repo). You should also allow it to be triggered remotely. Check the “Trigger builds remotely” checkbox in the “Build Triggers” section and provide a secret Authentication token.

With Tuleap API you can also send to TTM Junit XML output. The automated tests which are concerned will be automatically updated.

In your Jenkinsfile you have to add the following command:

```
tuleapSendTTMResults filesPath: 'XML_FILES',
                     campaignId: 'CAMPAIGN_ID',
                     credentialId: 'CRED_ID_OF_ACCESS_KEY'
```

Where :

- XML_FILES are the Junit XML outputs.
- CAMPAIGN_ID is the TTM concerned campaign ID
- CRED_ID_OF_ACCESS_KEY is the Jenkins credentials ID of your access key. Here it is `tuleap-access-token`

Here is an example Jenkinsfile using the tuleapSendTTMResults command:

```
 pipeline {
   agent any
   parameters {
     string(name: 'campaignId', defaultValue: '', description: 'ID of the campaign')
   }
   stages {
      stage('Reporting') {
         steps {
             tuleapSendTTMResults filesPath: 'outputs/*.xml', campaignId: "${params.campaignId}", credentialId: 'tuleap-access-token'
         }
      }
   }
}
```

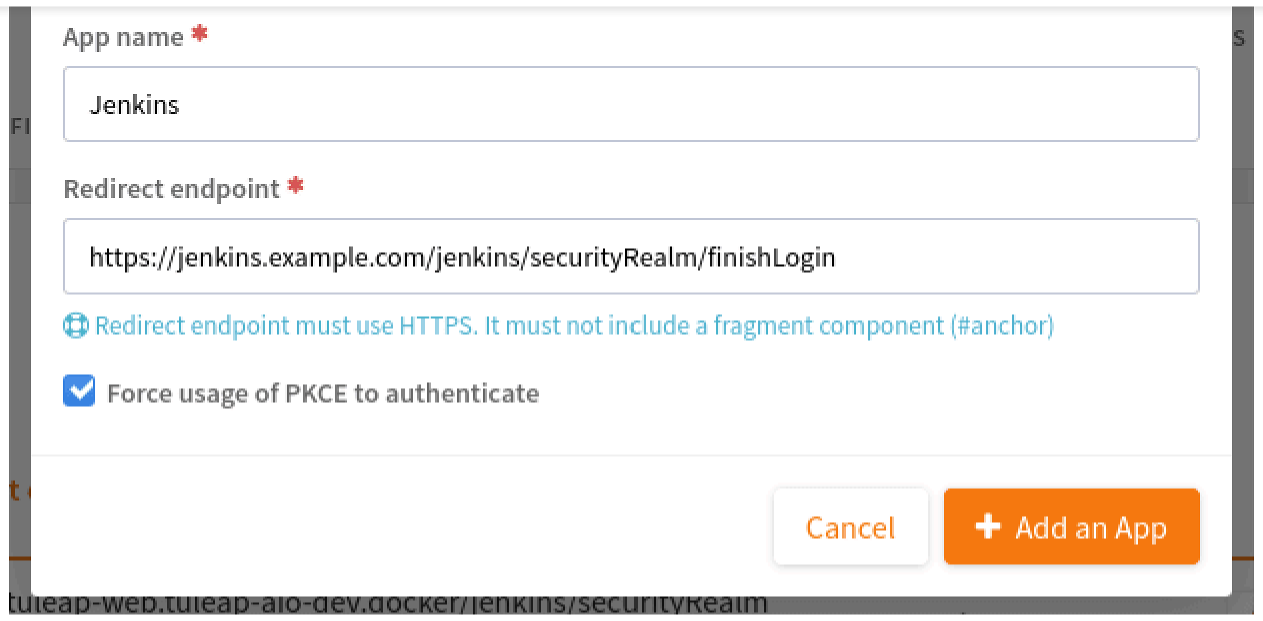
See Test Automation documentation, for more information.

AUTHENTICATION CONFIGURATION

This section requires that your Tuleap server has OAuth2 Server plugin installed.

First, on your Tuleap server, in one of your project, you need to create a new OAuth2 app. The app will ask for a callback URL. This callback URL is your Jenkins server base URL (eg. https://jenkins.example.com/jenkins) + `/securityRealm/finishLogin`. The plugin allows the PKCE usage for the authentication. You can force its usage at the creation of the OAuth2 app.

Keep the generated Client Secret securely until the next step.



App name *

Jenkins

Redirect endpoint *

https://jenkins.example.com/jenkins/securityRealm/finishLogin

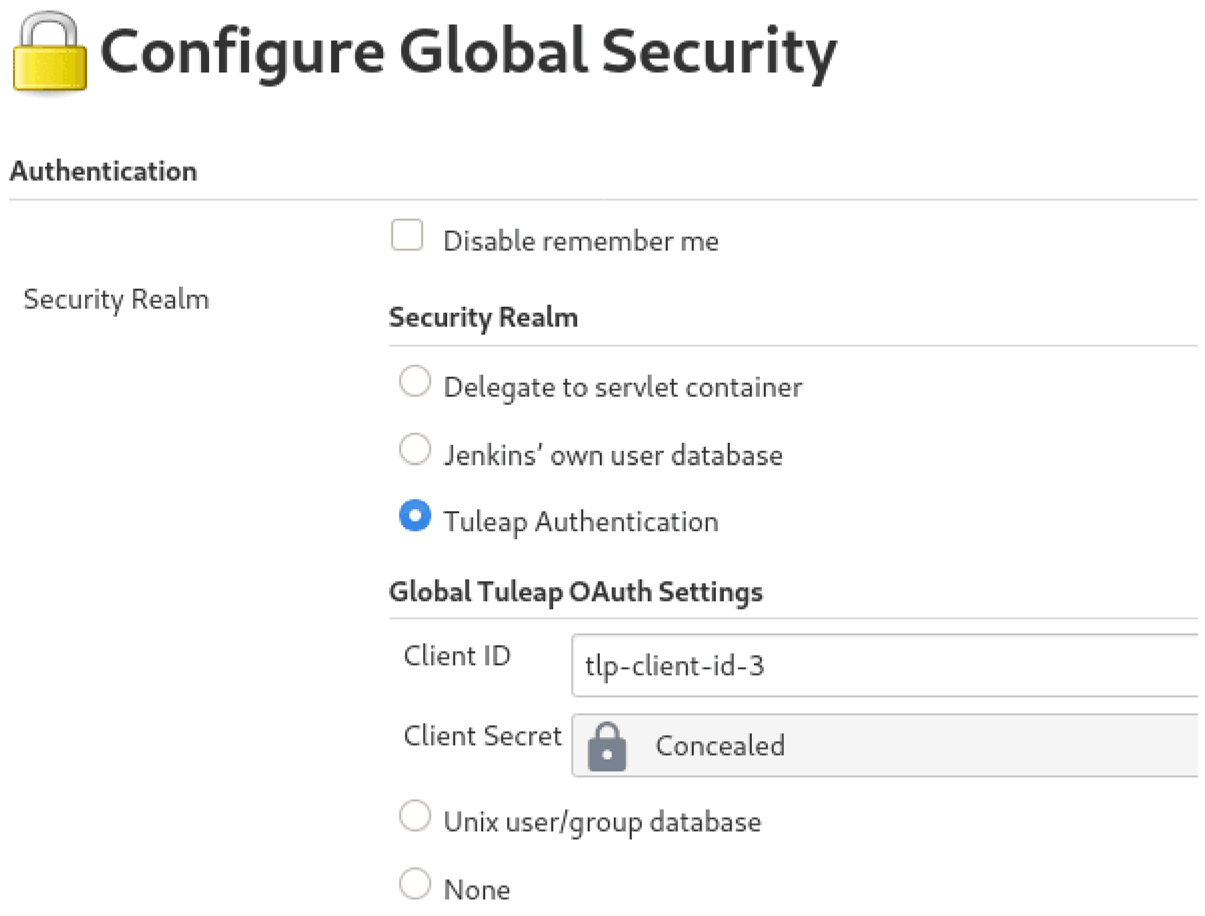
Redirect endpoint must use HTTPS. It must not include a fragment component (#anchor)

☑ Force usage of PKCE to authenticate

Then Jenkins, go In *Manage Jenkins > Configure Global Security*, and select *Tuleap Authentication* and fill:

- Client ID
- Client Secret

With the values provided by Tuleap.



Configure Global Security

Authentication

□ Disable remember me

Security Realm

Security Realm

○ Delegate to servlet container

○ Jenkins' own user database

● Tuleap Authentication

Global Tuleap OAuth Settings

Client ID: tlp-client-id-3

Client Secret: Concealed

○ Unix user/group database

○ None

Ensure that *Authorization* (bellow *Authentication section*) is still set to *Anyone can do anything* and click save.

You should then be able to login on Jenkins with you Tuleap credentials and still have access to *Manage Jenkins*.

> Note
>
> If you locked yourself out of Jenkins you can start over by disabling security.

AUTHORIZATION CONFIGURATION

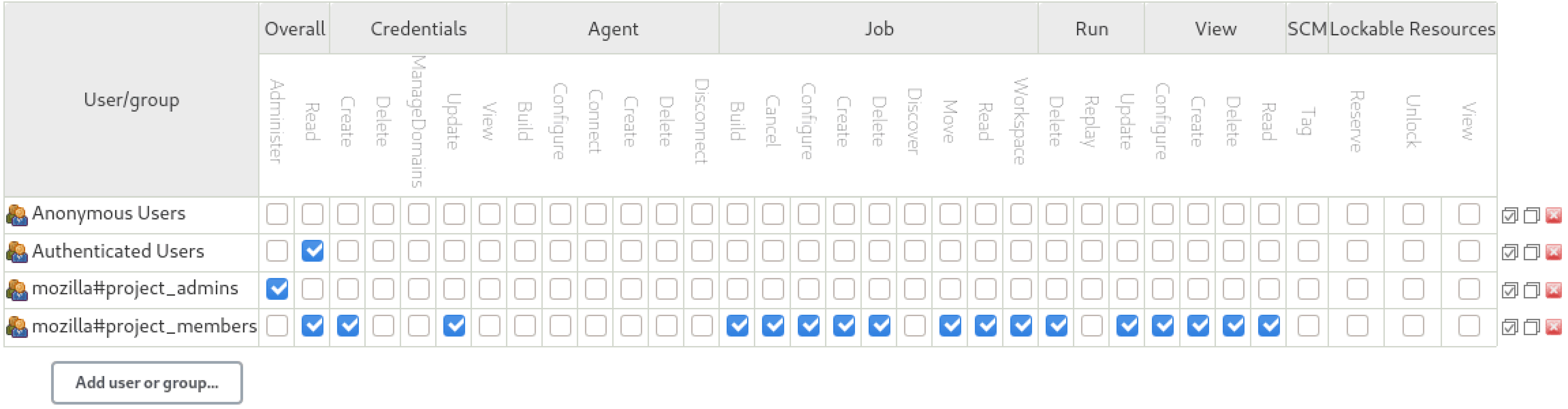
This section depends on the previous one. If you don't use Tuleap Authentication, you cannot do the following configurations.

In the Authorization Matrix you can reference:

- Tuleap users with their `login_name`
- Tuleap user groups in the form `project-shortname#user-group-name`

In the following screenshot you have a good start point in term of permissions management with the default groups defined in Tuleap:

- Access to Jenkins must be authenticated. Controlled by Tuleap => only the people who have access to Tuleap will have access to Jenkins
- The people who can access can only Read your jobs. That correspond to a “Public” project (or “Public including Restricted” if you have restricted users in Tuleap).
- The project members of the Tuleap mozilla project can manage jobs, builds and credentials.
- The project administrators of Tuleap mozilla project can administer Jenkins (plugins, grants & co).



User/group	Overall		Credentials					Agent						Job									Run			View				SCM	Lockable Resources		
	Administer	Read	Create	Delete	ManageDomains	Update	View	Build	Configure	Connect	Create	Delete	Disconnect	Build	Cancel	Configure	Create	Delete	Discover	Move	Read	Workspace	Delete	Replay	Update	Configure	Create	Delete	Read	Tag	Reserve	Unlock	View
Anonymous Users	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
Authenticated Users	☐	☑	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
mozilla#project_admins	☑	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
mozilla#project_members	☐	☑	☑	☐	☐	☑	☐	☐	☐	☐	☐	☐	☐	☑	☑	☑	☑	☑	☐	☑	☑	☑	☑	☐	☑	☑	☑	☑	☑	☐	☐	☐	☐

At this point you are almost done with Jenkins administration. You might need to install and configure other plugins depending of the content of your pipelines, configure agent to distribute the load and, of course, update jenkins itself but you are done for the permission and user management.

Tuleap Git / Jenkins integration

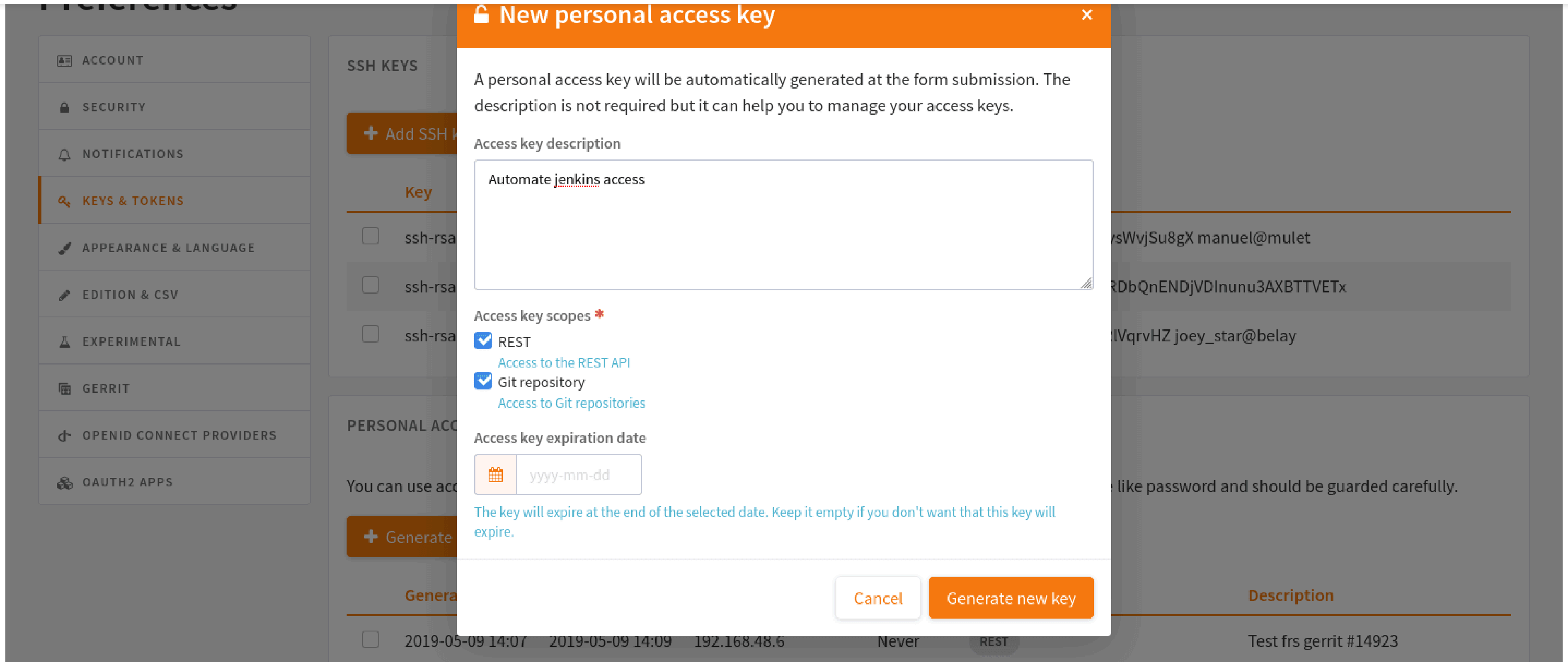
Thanks to Tuleap Git Branch Source Jenkins plugin, most of the integration between the two tools is completely streamlined.

The configuration is done once at project level, then every new git repository created in Tuleap will be automatically discovered by Jenkins, branches will be inspected to find `Jenkinsfile` and created corresponding pipelines.

Whenever a new commit will be pushed or a Pull Request created, the corresponding job will be triggered on Jenkins.

STEP 1: HAVE AN ACCESS KEY TO YOUR REPOSITORIES

In Tuleap, either with a service or personal account that have read access to the project’s repositories go in user preferences, “Keys & Tokens” section and generate a new Access Key with both `Git` and `REST` scopes.



New personal access key ×

A personal access key will be automatically generated at the form submission. The description is not required but it can help you to manage your access keys.

Access key description

Automate jenkins access

Access key scopes *

☑ REST
Access to the REST API

☑ Git repository
Access to Git repositories

Access key expiration date

yyyy-mm-dd

The key will expire at the end of the selected date. Keep it empty if you don't want that this key will expire.

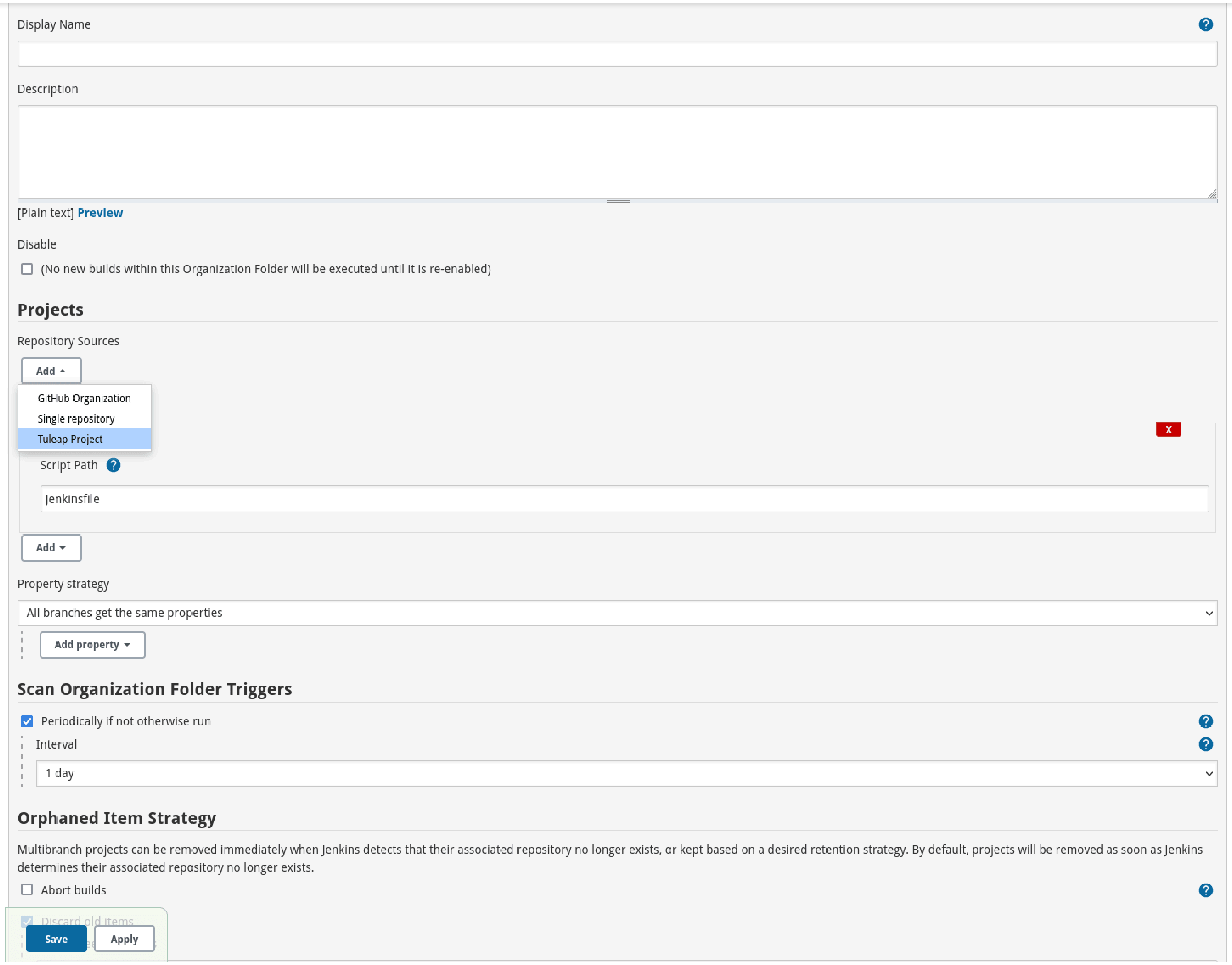
Key	
ssh-rsa	sWvjSu8gX manuel@mulet
ssh-rsa	RDbQnENDjVDInunu3AXBTTVETx
ssh-rsa	lVqrvHZ joey_star@belay

You can use acc … like password and should be guarded carefully.

Genera					Description
2019-05-09 14:07	2019-05-09 14:09	192.168.48.6	Never	REST	Test frs gerrit #14923

STEP 2: CREATE A TULEAP PROJECT

In Jenkins, create an Organization Folder and in the Projet Section of the configuration choose "Tuleap Project" on Repository sources selectbox.



Display Name

Description

[Plain text] Preview

Disable

☐ (No new builds within this Organization Folder will be executed until it is re-enabled)

Projects

Repository Sources

Script Path

Jenkinsfile

Property strategy

All branches get the same properties

Scan Organization Folder Triggers

☑ Periodically if not otherwise run

Interval

1 day

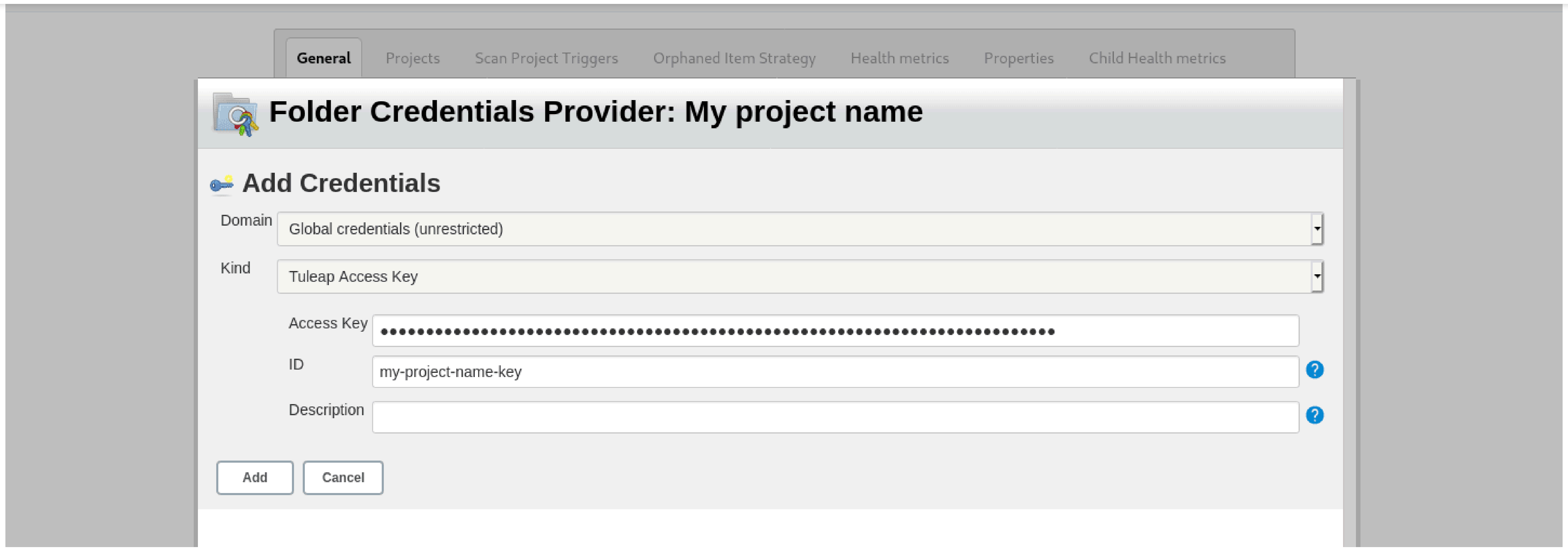
Orphaned Item Strategy

Multibranch projects can be removed immediately when Jenkins detects that their associated repository no longer exists, or kept based on a desired retention strategy. By default, projects will be removed as soon as Jenkins determines their associated repository no longer exists.

☐ Abort builds

☑ Discard old items

Once the job created you should grant it access to Tuleap with the credential you generated at Step 1. Near the credential drop down, you have a “Add” button. Create a new “Project name” credential of type “Tuleap Access Key” and give it a descriptive id so you can find it later.



Folder Credentials Provider: My project name

Add Credentials

Domain: Global credentials (unrestricted)

Kind: Tuleap Access Key

Access Key: ••

ID: my-project-name-key

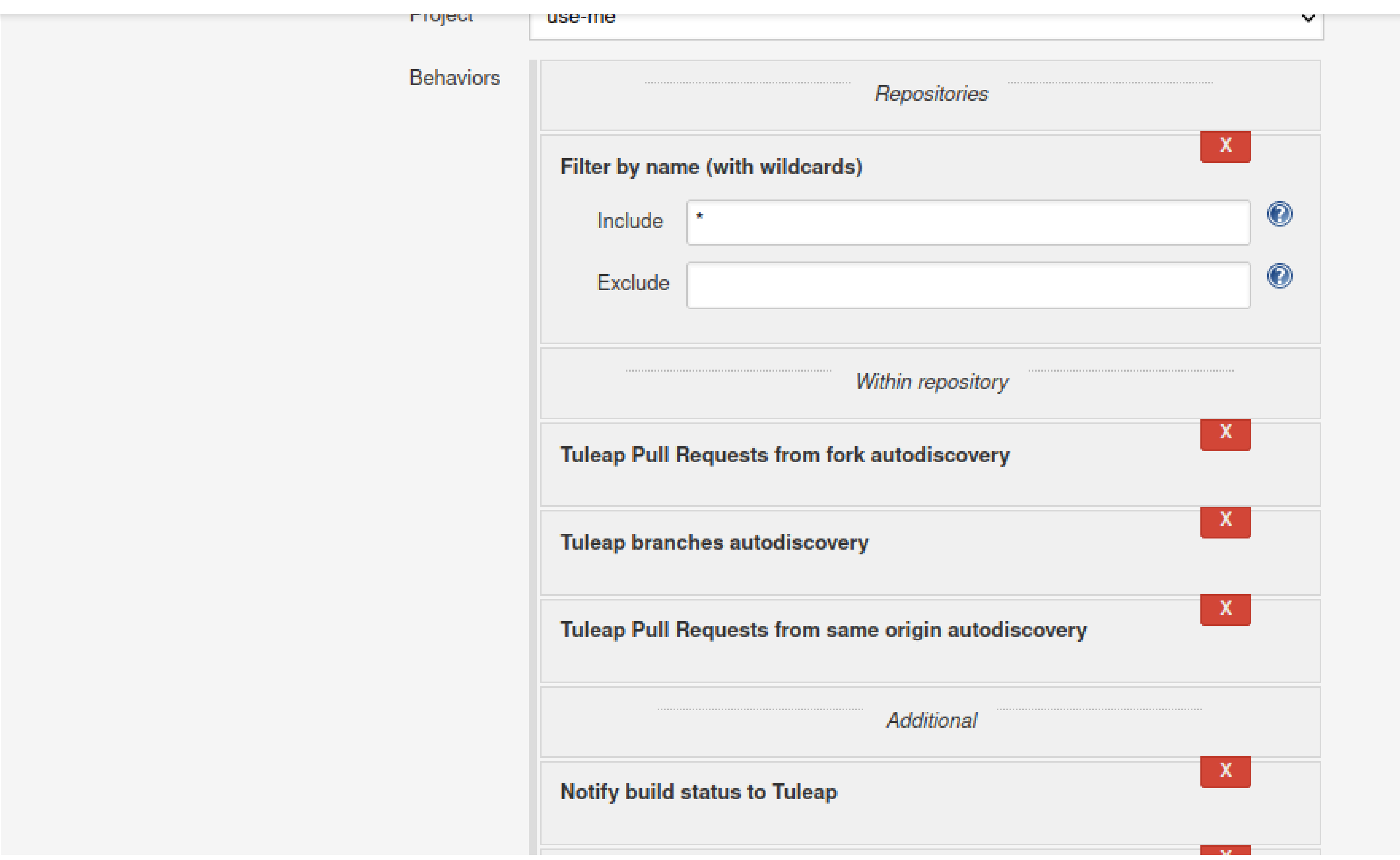
Description:

Once the credential is saved, select it in the “Credentials” dropdown.

In the “Project” dropdown right after, select the Tuleap project you want to automate.

You can adjust “Behaviours” to match your need. The `tuleap-git-branch-source` plugin proposes 4 configuration traits:

Trait Name	Category	By default	Behaviour
Tuleap branches autodiscovery	Within Repository	Yes	Discover all branches of the repository
Tuleap Pull Requests from same origin autodiscovery	Within Repository	Yes	Discover all Pull Requests created inside the repository
Tuleap Pull Requests from fork autodiscovery	Within Repository	No	Discover all Pull Requests coming from fork repositories
Notify build status to Tuleap	Additional	No	Automatically Notify Tuleap of the result of the build (will use the Access key configured in the scanner job)



Project: use-me

Behaviors

Repositories

Filter by name (with wildcards)

Include: *

Exclude:

Within repository

Tuleap Pull Requests from fork autodiscovery

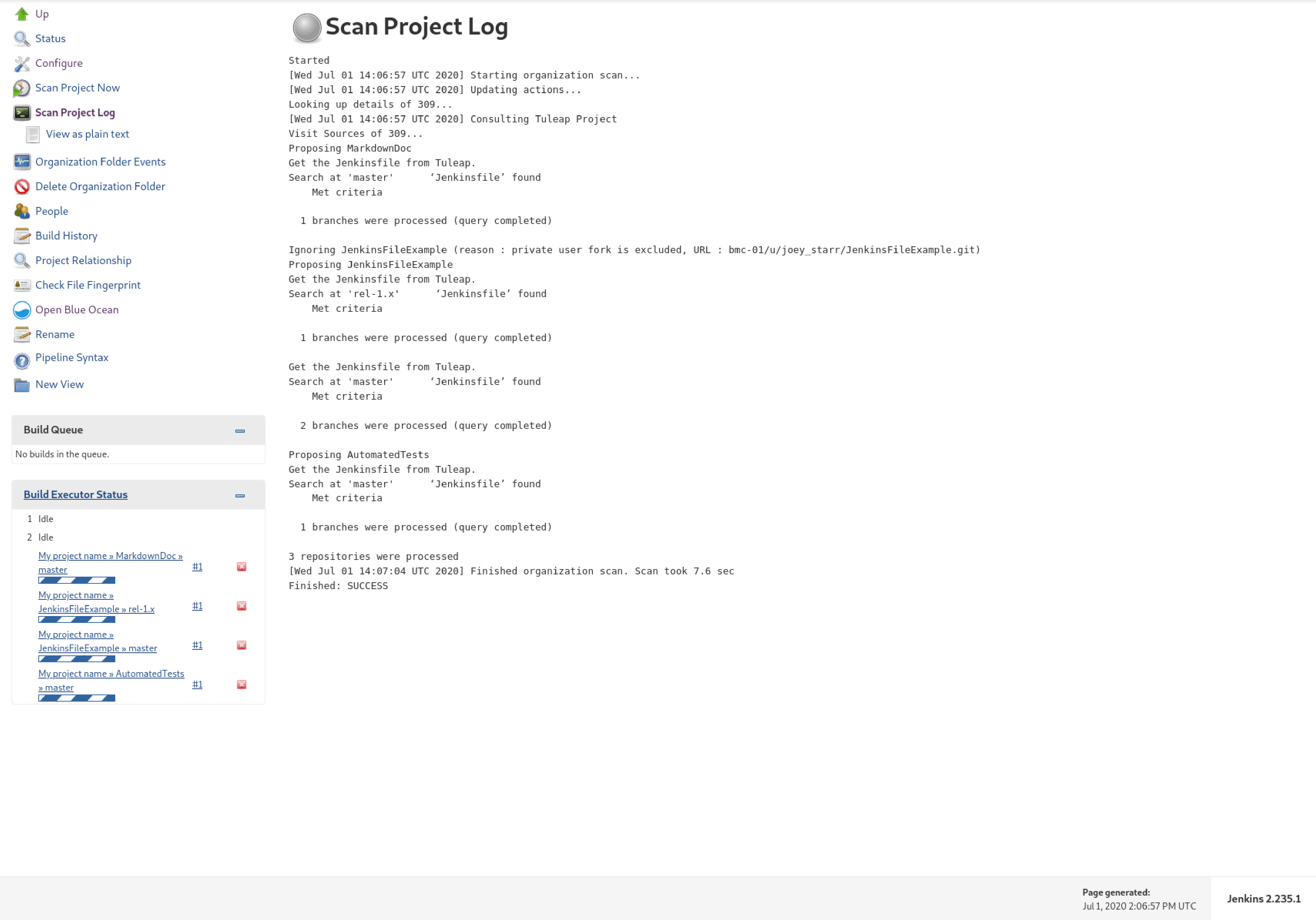
Tuleap branches autodiscovery

Tuleap Pull Requests from same origin autodiscovery

Additional

Notify build status to Tuleap

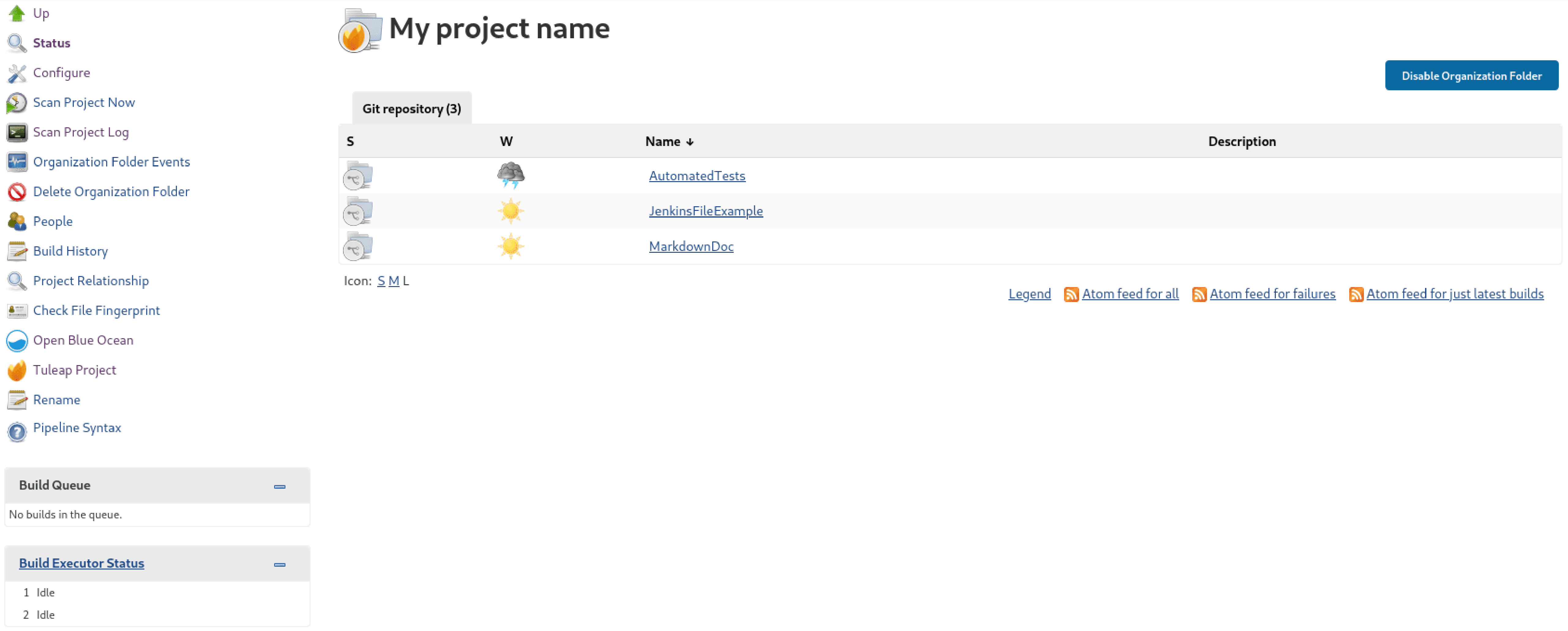
When the configuration is ready, save it. This will trigger a scan of your project to look for git repositories, their branches, their pull requests, and `Jenkinsfile` to create Jenkins jobs.


Up
Status
Configure
Scan Project Now
Scan Project Log
View as plain text
Organization Folder Events
Delete Organization Folder
People
Build History
Project Relationship
Check File Fingerprint
Open Blue Ocean
Rename
Pipeline Syntax
New View
Build Queue
No builds in the queue.
Build Executor Status
1 Idle
2 Idle
My project name » MarkdownDoc » master #1
My project name » JenkinsFileExample » rel-1.x #1
My project name » JenkinsFileExample » master #1
My project name » AutomatedTests » master #1
Page generated: Jul 1, 2020 2:06:57 PM UTC
Jenkins 2.235.1

Scan Project Log

```
Started
[Wed Jul 01 14:06:57 UTC 2020] Starting organization scan...
[Wed Jul 01 14:06:57 UTC 2020] Updating actions...
Looking up details of 309...
[Wed Jul 01 14:06:57 UTC 2020] Consulting Tuleap Project
Visit Sources of 309...
Proposing MarkdownDoc
Get the Jenkinsfile from Tuleap.
Search at 'master'      'Jenkinsfile' found
    Met criteria

  1 branches were processed (query completed)

Ignoring JenkinsFileExample (reason : private user fork is excluded, URL : bmc-01/u/joey_starr/JenkinsFileExample.git)
Proposing JenkinsFileExample
Get the Jenkinsfile from Tuleap.
Search at 'rel-1.x'      'Jenkinsfile' found
    Met criteria

  1 branches were processed (query completed)

Get the Jenkinsfile from Tuleap.
Search at 'master'      'Jenkinsfile' found
    Met criteria

  2 branches were processed (query completed)

Proposing AutomatedTests
Get the Jenkinsfile from Tuleap.
Search at 'master'      'Jenkinsfile' found
    Met criteria

  1 branches were processed (query completed)

3 repositories were processed
[Wed Jul 01 14:07:04 UTC 2020] Finished organization scan. Scan took 7.6 sec
Finished: SUCCESS
```

When the scan is completed, you will find all the git repositories where Jenkins found a `Jenkinsfile` and the status of the builds that were triggered.



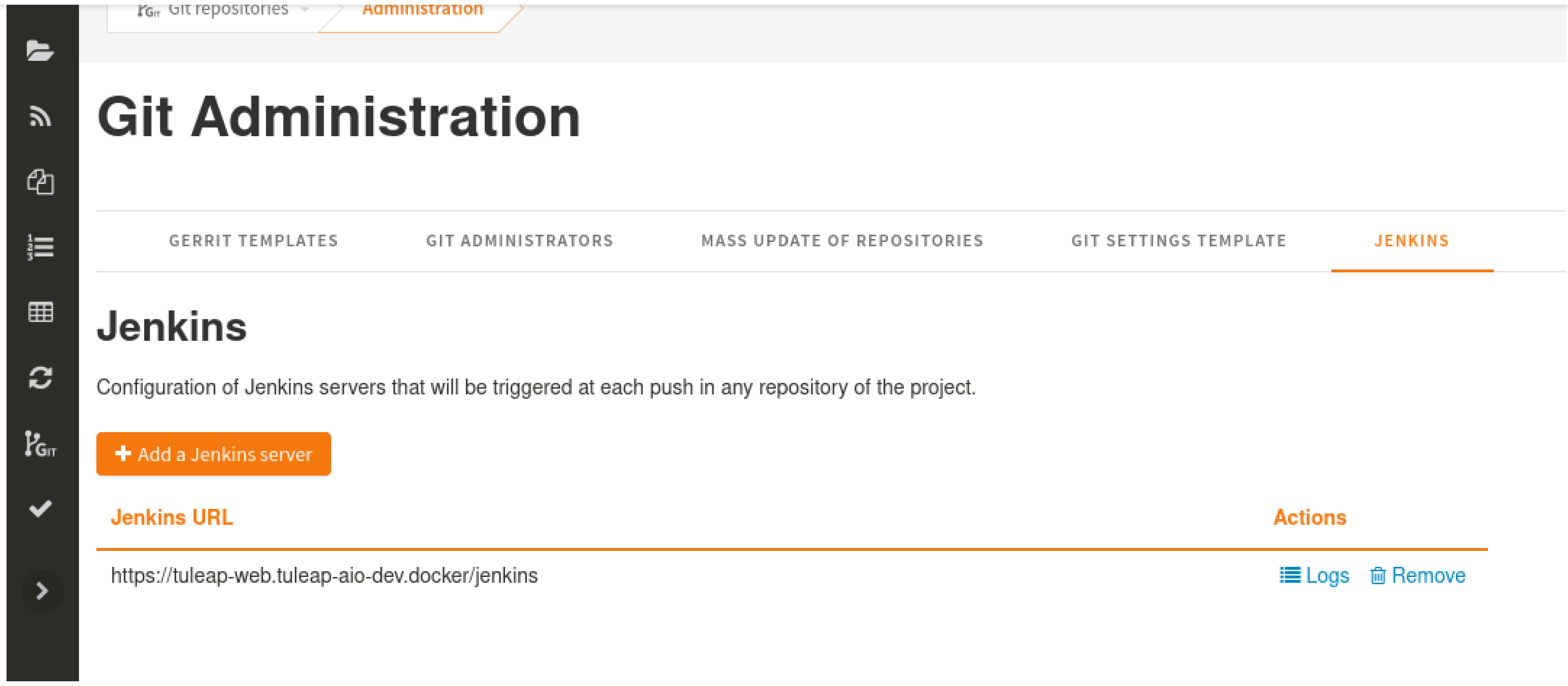
S	W	Name ↓	Description
		AutomatedTests	
		JenkinsFileExample	
		MarkdownDoc	

> Attention
>
> On Jenkins, in your project settings, you might also want to adjust “Scan Project Triggers” to a shorter period otherwise you will have to wait for 1 day between a new repository creation and jenkins to discover it.
>
> As this will trigger a full analyze of all branches and pull requests of git repositories of your Tuleap project, you need to find a balance between reactivity and Tuleap server overloading.
>
> If you don’t create a new repository every other hours, you might want to let 1 day period and trigger manually the scan whenever you create a new repository.

STEP 3: TELL TULEAP WHERE THE JENKINS SERVER IS

The final step is on Tuleap. You need to inform the git server where is the Jenkins server that must be informed about new commits that are pushed.

In the administration of the Git service of your project, there is a `Jenkins` tab where you set the Jenkins root url. For instance `https://jenkins.example.com/jenkins`.



Git Administration

Jenkins

Configuration of Jenkins servers that will be triggered at each push in any repository of the project.

Jenkins URL	Actions
https://tuleap-web.tuleap-aio-dev.docker/jenkins	Logs Remove

THAT'S IT

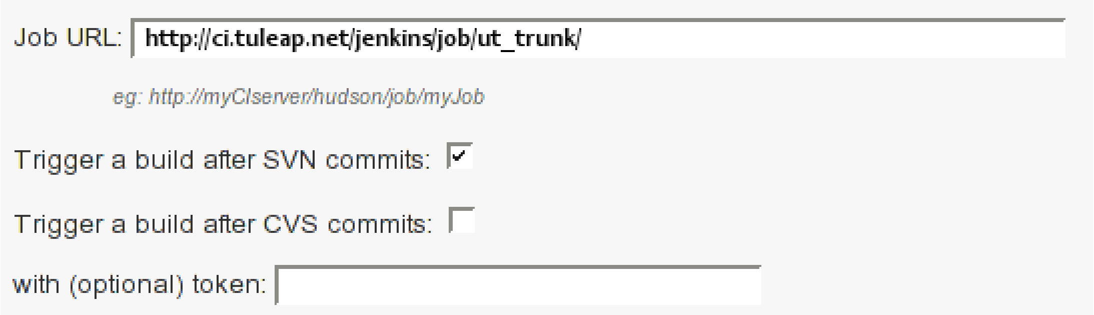
When those 3 steps are completed, you no longer have to worry about Jenkins / Tuleap integration, everything is automated.

Continuous Integration service in Tuleap

> Note
>
> The "Continuous Integration" service in Tuleap refers to an historical implementation that was mainly targeting Subversion.
>
> It also provides some widgets that can be used on Project and Personal dashboards.

REFERENCE JENKINS JOB WITH YOUR TULEAP PROJECT

In order to link Jenkins job with your project, select the Continuous Integration tab of your project, and then select the 'Add a job' link. You need then to give the URL of the Hudson job you want to associate with your project (for instance: `https://[my\_ci\_server]:8080/jenkins/job/[my\_job]`).



Job URL: http://ci.tuleap.net/jenkins/job/ut_trunk/

eg: http://myCIserver/hudson/job/myJob

Trigger a build after SVN commits: ☑

Trigger a build after CVS commits: □

with (optional) token:

You may also want to enable the auto trigger of the build for this job after each commit in your project repository (Subversion). If you have protected your build with a token, you can specify this token.

By checking this option, each commit will trigger a build of the associated job, using the pre-commit hook (you don't have anything more to do).

By the way, it is possible to link several Jenkins jobs with one Tuleap project.

JENKINS JOBS AND BUILDS

When you select the Continuous Integration tab of your project, you can see a table with all the jobs associated with your project. For every job, you can see the current status (colored bullet left to the name of the job), the name, the last successful build, the last failed build, if you have enabled SCM trigger or not.

Project admins will also see for each job some icons that let them modify the job or delete it (remove the link with Tuleap).

●	JRI	build #180	build #174				✎ ✗
●	XEclipse	build #17	build #11				✎ ✗
●	TestProjectExistingJob		build #1				✎ ✗
●	TestProjectExternalJob						✎ ✗
●	TestProjectMaven2						✎ ✗
●	trigger	build #54	build #1				✎ ✗

The name of the job is automatically detected during job creation. But you can change it if needed. This is pretty convenient if you want to make references to Jenkins items (see Make a reference to a Job). Spaces in the name of jobs are not allowed. They are replaced by (_), in order to allow references.

Jenkins References

It is possible to make references to Jenkins items in Tuleap. There are some predefined references (job, build), but you can also create your own references if needed (see Reference Overview for more details about references)

MAKE A REFERENCE TO A JOB

The keyword to make a reference to a Job is: **job**. To make a reference to a job, you can use the expressions:

- job #JobNameToReference (the job must be in the current project)
- job #project:JobNameToReference (will make a reference to the job 'JobNameToReference' of the project 'project')
- job #project_num:JobNameToReference (will make a reference to the job 'JobNameToReference' of the project with number 'project_num')

MAKE A REFERENCE TO A BUILD

The keyword to make a reference to a build is: **build**. To make a reference to a build, you can use the expressions:

- build #XXX (there must be only one job associated with the current project, and the referenced build will be the build number 'XXX' of this job)
- build #AJob/XXX (will make a reference to build number 'XXX' of job named 'AJob' of the current project)
- build #project:AJob/XXX (will make a reference to the build number 'XXX' of the job 'AJob' of project 'project')
- build #projet_num:AJob/XXX (will make a reference to the build number 'XXX' of the job 'AJob' of the project number 'project_num')

Previous Next