September, 2020

An IBM TechDoc from

IBM Z DevOps Acceleration Program

Managing git credentials in Jenkins to access the central git provider

Dennis Behm
dennis.behm@de.ibm.com

Nicolas Dangeville
dangeville.n@fr.ibm.com

Abstract
Navigate and manage encoding issues, when managing your git credentials in Jenkins using the Jenkins Git client plugin. This TechDoc will provide the required guidance depending on the version of the Jenkins Git client plugin in use.

Release 1.1 (September 2020)

Table of Contents

Introduction

This document provides guidance on managing credentials to your central git server and the configuration of the git tools in Jenkins to successfully interact with your central git server from z/OS Unix System Services (USS).

Accessing your central git server is crucial to build up your pipeline. Starting with git version `2.14.4_zos_b09` ported by Rocket software to USS, it is possible to use HTTPS to clone a repository. Previously only SSH keys were supported for cloning the repository to Unix System Services.

In an SSH scenario, the user leverages SSH keys to authenticate against the central git provider.[1] The public key is shared with the central git provider. The key can be secured by adding a SSH passphrase, which typically needs to be provided each time the user is interacting with the remote repository, for example when cloning the repository to the build workspace or pushing changes to remote.

An HTTP(s) scenario is an attracting option in a complex network setup, when for example SSH connections are not permitted. GitHub is recommending this connection method.[2]

Several papers and documentations are available describing the configuration of a Jenkins remote agent on USS.[3] This will not be covered in this document. However, running a Jenkins remote agent on USS adds a topic to the list, which needs to be well understood: Encodings; because Unix System Services runs on codepage EBCDIC IBM-1047.

This Techdoc provides guidance in the area of managing git credentials in Jenkins and the different considerations with the different releases of the Jenkins Git client plugin[4].

You will also find in the Appendix some more details about the git credential management.

[1] https://help.github.com/en/github/authenticating-to-github/connecting-to-github-with-ssh
[2] https://help.github.com/en/github/using-git/which-remote-url-should-i-use#cloning-with-https-urls-recommended
[3] https://www.ibm.com/support/knowledgecenter/SS6T76_1.0.9/jenkinsintegration.html
[4] https://plugins.jenkins.io/git-client/

Git communication in a Jenkins context

There are several scenarios to manage the credentials when interacting with your central git provider from the build machine.

In general, you can manage the credentials on the build machine or within the CI orchestrator. The latter will enable you to centrally manage several credentials – for example one for each application repository, while using the same Jenkins remote agent. This is the one that we will explore in this chapter.

Let’s review the list of the different scenarios to manage the authentication information:

Scenarios using SSH:

1. The private SSH key is managed in Jenkins, there is no passphrase configured.
2. The private SSH key is managed in Jenkins, additionally a passphrase is set up for the key.
3. The private SSH key is managed on the build machine, there is no passphrase configured
4. The private SSH key is managed on the build machine, and there is a passphrase configured.

The SSH scenarios 3 and 4 don’t require any additional configuration, when you run the Jenkins remote agent on Unix System Services. But they often don’t meet security requirements of organizations, as the key is stored on the build machine.

SSH scenarios 1 + 2 require some additional attention, which is similar to managing the git credentials in Jenkins for HTTPS communication.

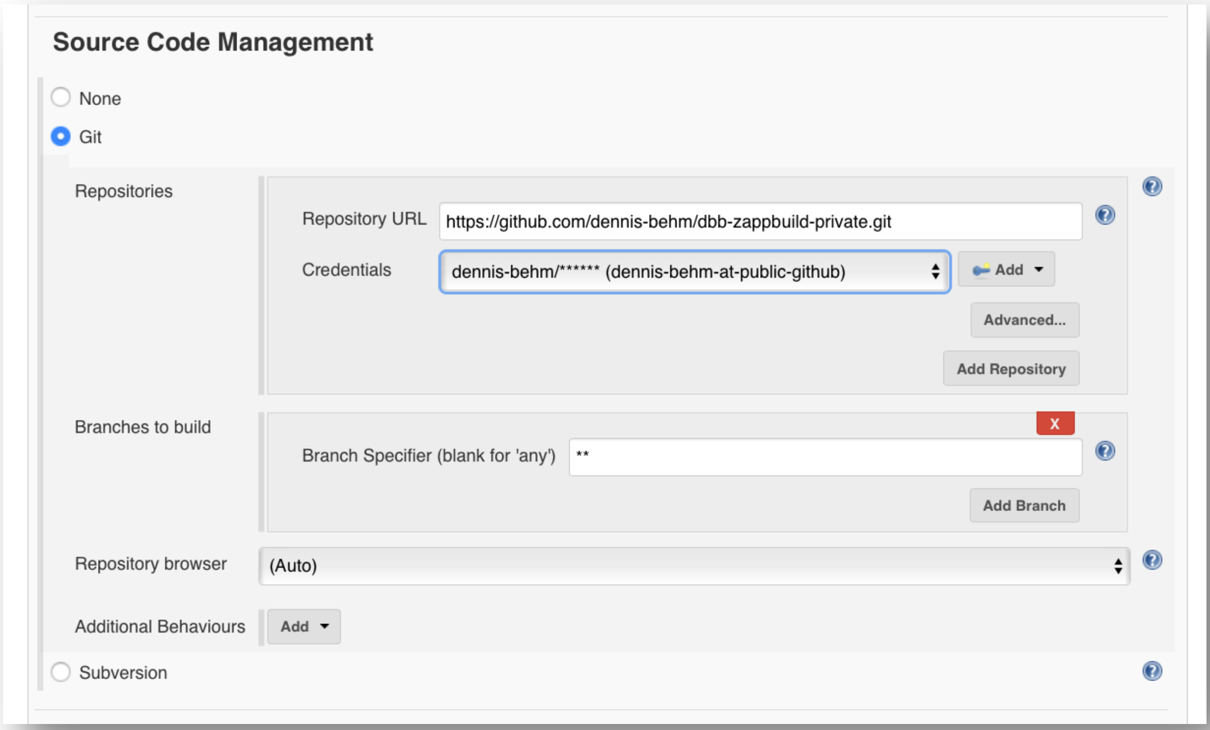
Scenarios using HTTPS

1. The credentials for accessing the central git server are managed within Jenkins
2. The credentials for accessing the central git server are managed on the build machine in the git configuration through a credential helper

The next section of this document focuses on the setup of storing git credentials within Jenkins for a HTTPS connection for your central git server.

Configuration of your Jenkins project to manage credentials

You can manage the credentials to communicate with git in the Jenkins Credential store and refer to them in your Jenkins job configuration:



Source Code Management

None
[✓ Git]

Repositories
Repository URL: https://github.com/dennis-behm/dbb-zappbuild-private.git
Credentials: dennis-behm/****** (dennis-behm-at-public-github)

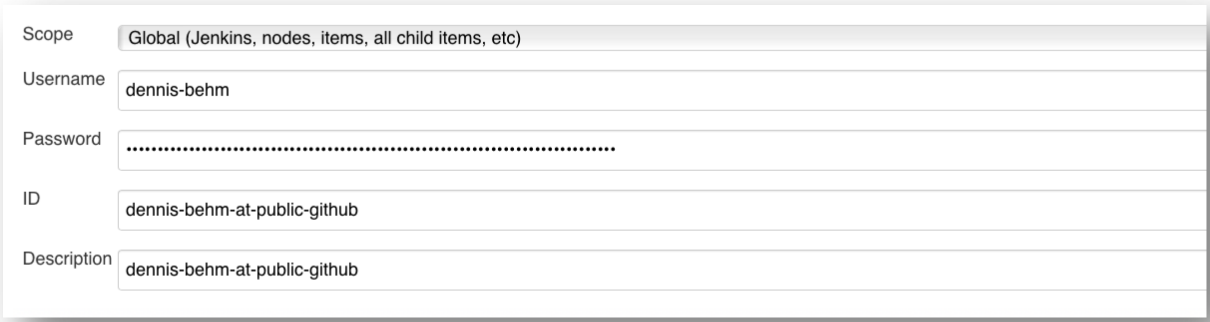
Branches to build
Branch Specifier (blank for 'any'): **

Repository browser: (Auto)

Additional Behaviours

Subversion

In this case, Jenkins stores your username and password for accessing the central git server. In addition, it can be scoped to be available only to a Credential domain.



Scope	Global (Jenkins, nodes, items, all child items, etc)
Username	dennis-behm
Password	••
ID	dennis-behm-at-public-github
Description	dennis-behm-at-public-github

As mentioned, running the Jenkins remote agent on Unix System Services, requires additional attention.

Git credential handling on the Jenkins remote agent

The Jenkins Git client plugin takes care that the credentials stored in Jenkins are correctly applied on the target build machine.

The GitClient interface provides the primary entry points for git access. It supports username / password credentials and private SSH key credentials using the Jenkins credentials plugin.

Due to the history of the plugins and the evaluation of their implementation, this area requires special attention when using the plugin in a mainframe context. One reason is the codepage of USS, where the plugins are being executed.

The below table provides an overview of the different combinations and points you to the available helpers to address known issues with the Git client plugin in the mainframe context.

Version of the **Jenkins Git Client** plugin	**Git Credential management** in Jenkins to	
	Clone via HTTP/S	Clone using SSH keys
Versions prior to v3.0.0	git-jenkins3.sh wrapper	
v3.0.0 to v3.4.0	git-jenkins2.sh wrapper	
v3.4.1 and newer	Supported through JVM properties introduced in v3.4.1	

Please note:
We used to provide a git-jenkins.sh wrapper that is now superseded with the options described in the above table.
The git-jenkins scripts can be obtained from the official dbb repository at: https://github.com/IBM/dbb/tree/master/Utilities/Jenkins

Using the Jenkins Git client plugin 3.0.0 – 3.4.0

When the git credentials are managed within Jenkins, then the Jenkins git plugin will create a set of temporary files in the build workspace to set up the git askpass environment.

By default, this temporary file is written to USS in UTF-8 (see recommended JVM property: -Djava.file.encoding=UTF-8). However, the operating system processes the file in EBCDIC. So, by default this does not work.

You might run into the below issue. We have also seen variation of this issues, where the log just reports `FATAL: Invalid id`

```
Started by user unknown or anonymous
Building      remotely      on      z-environment      (ztec-builder)      in      workspace
/var/dbb/buildhome/workspace/simpleGitClone-https-credentials-managed-by-Jenkins
using credential temporaryGitHubpwd
```

```
 > /var/dbb/config/git-jenkins-trace.sh rev-parse --is-inside-work-tree # timeout=120
Fetching changes from the remote Git repository
 >  /var/dbb/config/git-jenkins-trace.sh  config  remote.origin.url  https://github.com/git-
user/dbb-zappbuild-private.git # timeout=120
Fetching upstream changes from https://github.com/git-user/dbb-zappbuild-private.git
 > /var/dbb/config/git-jenkins-trace.sh --version # timeout=120
using GIT_ASKPASS to set credentials temporaryGitHubpwd
 >  /var/dbb/config/git-jenkins-trace.sh  fetch  --tags  --progress  --  https://github.com/git-
user/dbb-zappbuild-private.git +refs/heads/*:refs/remotes/origin/* # timeout=120
ERROR: Error fetching remote repo 'origin'
hudson.plugins.git.GitException:  Failed  to  fetch  from  https://github.com/git-user/dbb-
zappbuild-private.git
         at hudson.plugins.git.GitSCM.fetchFrom(GitSCM.java:909)
         at hudson.plugins.git.GitSCM.retrieveChanges(GitSCM.java:1131)
         at hudson.plugins.git.GitSCM.checkout(GitSCM.java:1167)
         at hudson.scm.SCM.checkout(SCM.java:504)
         at hudson.model.AbstractProject.checkout(AbstractProject.java:1208)
         at
hudson.model.AbstractBuild$AbstractBuildExecution.defaultCheckout(AbstractBuild.java:574)
         at jenkins.scm.SCMCheckoutStrategy.checkout(SCMCheckoutStrategy.java:86)
         at hudson.model.AbstractBuild$AbstractBuildExecution.run(AbstractBuild.java:499)
         at hudson.model.Run.execute(Run.java:1810)
         at hudson.model.FreeStyleBuild.run(FreeStyleBuild.java:43)
         at hudson.model.ResourceController.execute(ResourceController.java:97)
         at hudson.model.Executor.run(Executor.java:429)
Caused by: hudson.plugins.git.GitException: Command "/var/dbb/config/git-jenkins-trace.sh fetch
--tags      --progress      --      https://github.com/git-user/dbb-zappbuild-private.git
+refs/heads/*:refs/remotes/origin/*" returned status code 128:
stdout: remote: Invalid username or password.
fatal: Authentication failed for 'https://github.com/git-user/dbb-zappbuild-private.git/'

stderr:
         at
org.jenkinsci.plugins.gitclient.CliGitAPIImpl.launchCommandIn(CliGitAPIImpl.java:2430)
         at
org.jenkinsci.plugins.gitclient.CliGitAPIImpl.launchCommandWithCredentials(CliGitAPIImpl.java
:2044)
         at org.jenkinsci.plugins.gitclient.CliGitAPIImpl.access$500(CliGitAPIImpl.java:81)
         at org.jenkinsci.plugins.gitclient.CliGitAPIImpl$1.execute(CliGitAPIImpl.java:569)
         at
org.jenkinsci.plugins.gitclient.RemoteGitImpl$CommandInvocationHandler$GitCommandMasterToSlav
eCallable.call(RemoteGitImpl.java:161)
         at
org.jenkinsci.plugins.gitclient.RemoteGitImpl$CommandInvocationHandler$GitCommandMasterToSlav
eCallable.call(RemoteGitImpl.java:154)
         at hudson.remoting.UserRequest.perform(UserRequest.java:212)
         at hudson.remoting.UserRequest.perform(UserRequest.java:54)
         at hudson.remoting.Request$2.run(Request.java:369)
         at
hudson.remoting.InterceptingExecutorService$1.call(InterceptingExecutorService.java:72)
         at java.util.concurrent.FutureTask.run(FutureTask.java:277)
         at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1160)
         at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:635)
         at java.lang.Thread.run(Thread.java:820)
         Suppressed: hudson.remoting.Channel$CallSiteStackTrace: Remote call to z-environment
                  at hudson.remoting.Channel.attachCallSiteStackTrace(Channel.java:1741)
                  at
hudson.remoting.UserRequest$ExceptionResponse.retrieve(UserRequest.java:357)
                  at hudson.remoting.Channel.call(Channel.java:955)
                  at
org.jenkinsci.plugins.gitclient.RemoteGitImpl$CommandInvocationHandler.execute(RemoteGitImpl.
java:146)
                  at sun.reflect.GeneratedMethodAccessor319.invoke(Unknown Source)
                  at
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:55)
                  at java.lang.reflect.Method.invoke(Method.java:508)
                  at
org.jenkinsci.plugins.gitclient.RemoteGitImpl$CommandInvocationHandler.invoke(RemoteGitImpl.j
ava:132)
                  at com.sun.proxy.$Proxy80.execute(Unknown Source)
                  at hudson.plugins.git.GitSCM.fetchFrom(GitSCM.java:907)
                  at hudson.plugins.git.GitSCM.retrieveChanges(GitSCM.java:1131)
                  at hudson.plugins.git.GitSCM.checkout(GitSCM.java:1167)
                  at hudson.scm.SCM.checkout(SCM.java:504)
                  at hudson.model.AbstractProject.checkout(AbstractProject.java:1208)
```

```
                   at
hudson.model.AbstractBuild$AbstractBuildExecution.defaultCheckout(AbstractBuild.java:574)
                   at jenkins.scm.SCMCheckoutStrategy.checkout(SCMCheckoutStrategy.java:86)
                   at
hudson.model.AbstractBuild$AbstractBuildExecution.run(AbstractBuild.java:499)
                   at hudson.model.Run.execute(Run.java:1810)
                   at hudson.model.FreeStyleBuild.run(FreeStyleBuild.java:43)
                   at hudson.model.ResourceController.execute(ResourceController.java:97)
                   at hudson.model.Executor.run(Executor.java:429)
ERROR: Error fetching remote repo 'origin'
Finished: FAILURE
```

Unix System Services writes/reads files by default in EBCDIC codepage IBM-1047. In this particular case, the temporary files for git askpass is UTF-8 encoded, as defined in the Jenkins node configuration for the JVM Options `-Dfile.encoding=UTF-8`

The wrapper script git-jenkins2.sh[8] supports the handling of the temporary files of git_askpass and ssh_askpass of the Jenkins Git client plugin **in v3.0.0 to v3.4.1** :

```
#!/bin/sh
#
# Licensed materials - Property of IBM
# 5655-AC5 Copyright IBM Corp. 2018, 2018
# All rights reserved
# US Government users restricted rights  -  Use, duplication or
# disclosure restricted by GSA ADP schedule contract with IBM Corp.
#
# Support modified behavior in Jenkins Git client plugin > 3.0
#
###################################################################
# Set current dir to direct temp outputs to @tmp dir

currentdir=$(pwd)

# git operating via SSH
if test -n "$GIT_SSH"; then
  if test -n "$SSH_ASKPASS"; then
    passphraseFile=$(cat $SSH_ASKPASS | grep "phrase" | sed "s/.*@tmp.//" | sed "s/'//")
    if test -n "$passphraseFile"; then
      chtag  -t  -c  UTF-8  $currentdir@tmp/$passphraseFile  >>$currentdir@tmp/git-jenkins-
wrapper.log 2>&1
      chtag -p $currentdir@tmp/$passphraseFile >>$currentdir@tmp/git-jenkins-wrapper.log 2>&1
      iconv     -f     utf-8     -T     -t     IBM-1047     $currentdir@tmp/$passphraseFile
>$currentdir@tmp/$passphraseFile.tmp
      mv $currentdir@tmp/${passphraseFile}.tmp $currentdir@tmp/$passphraseFile
    fi
  fi
fi

# git operating via HTTP/S
# Setting encoding of temporary files for git askpass
# Redirect of output is mandatory
if test -n "$GIT_ASKPASS"; then
  if test -f "$GIT_ASKPASS"; then
  usernameFile=$(cat  $GIT_ASKPASS  |  grep  "Username"  |  sed  "s/.*@tmp.//"  |  sed  "s/'.*//")
>>$currentdir@tmp/git-jenkins-wrapper.log 2>&1
  passwordFile=$(cat  $GIT_ASKPASS  |  grep  "Password"  |  sed  "s/.*@tmp.//"  |  sed  "s/'.*//")
>>$currentdir@tmp/git-jenkins-wrapper.log 2>&1
  if test -n "$usernameFile"; then
    # check the current tag, if ibm-1047 don't iconv
    chtag -p $currentdir@tmp/$usernameFile | grep -q "IBM-1047" >>$currentdir@tmp/git-jenkins-
wrapper.log 2>&1
    if [ $? -eq 0 ]; then
      echo 'file already in ibm-1047' >>$currentdir@tmp/git-jenkins-wrapper.log 2>&1
    else
```

[8] https://www.ibm.com/support/knowledgecenter/SS6T76_1.0.8/jenkinsintegration.html

```
      chtag -p $currentdir@tmp/$usernameFile >>$currentdir@tmp/git-jenkins-wrapper.log 2>&1
      chtag -p $currentdir@tmp/$passwordFile >>$currentdir@tmp/git-jenkins-wrapper.log 2>&1
      chtag -t -c UTF-8 $currentdir@tmp/$usernameFile >>$currentdir@tmp/git-jenkins-wrapper.log
2>&1
      chtag -t -c UTF-8 $currentdir@tmp/$passwordFile >>$currentdir@tmp/git-jenkins-wrapper.log
2>&1
      chtag -p $currentdir@tmp/$usernameFile >>$currentdir@tmp/git-jenkins-wrapper.log 2>&1
      chtag -p $currentdir@tmp/$passwordFile >>$currentdir@tmp/git-jenkins-wrapper.log 2>&1
      iconv     -f     utf-8     -T     -t     IBM-1047     $currentdir@tmp/$usernameFile
>$currentdir@tmp/$usernameFile.tmp
      iconv     -f     utf-8     -T     -t     IBM-1047     $currentdir@tmp/$passwordFile
>$currentdir@tmp/$passwordFile.tmp
      mv $currentdir@tmp/${usernameFile}.tmp $currentdir@tmp/$usernameFile
      mv $currentdir@tmp/${passwordFile}.tmp $currentdir@tmp/$passwordFile
    fi
    fi
  fi
fi

#git "$@" | iconv -f ibm-1047 -t ibm-1047
git "$@"
```

Please be aware that the outputs of the commands need to be directed to a temporary file or to null. Otherwise the wrong parameters are returned to the Jenkins server.

Please note: The latest version of the script is stored in the public GitHub repository at: https://github.com/IBM/dbb/tree/master/Utilities/Jenkins

With the above wrapper script, you are now able to clone successfully. Verify the successful git clone process - the next build will correctly pick up the git ask pass definition and clone the repository.

```
Started by user unknown or anonymous
Building     remotely     on     z-environment     (ztec-builder)     in     workspace
/var/dbb/buildhome/workspace/simpleGitClone-CM
[WS-CLEANUP] Deleting project workspace...
[WS-CLEANUP] Done
using credential ibmuser-public-github-userid-password
Cloning the remote Git repository
Cloning repository https://github.com/git-user/dbb-zappbuild-private.git
 >  /var/dbb/config/git-jenkins.sh  init  /var/dbb/buildhome/workspace/simpleGitClone-CM  #
timeout=120
Fetching upstream changes from https://github.com/git-user/dbb-zappbuild-private.git
 > /var/dbb/config/git-jenkins.sh --version # timeout=120
using GIT_ASKPASS to set credentials
 >  /var/dbb/config/git-jenkins.sh  fetch  --tags  --progress  https://github.com/git-user/dbb-
zappbuild-private.git +refs/heads/*:refs/remotes/origin/*
 >  /var/dbb/config/git-jenkins.sh  config  remote.origin.url  https://github.com/git-user/dbb-
zappbuild-private.git # timeout=120
 >       /var/dbb/config/git-jenkins.sh       config       --add       remote.origin.fetch
+refs/heads/*:refs/remotes/origin/* # timeout=120
 >  /var/dbb/config/git-jenkins.sh  config  remote.origin.url  https://github.com/git-user/dbb-
zappbuild-private.git # timeout=120
Fetching upstream changes from https://github.com/git-user/dbb-zappbuild-private.git
using GIT_ASKPASS to set credentials
 >  /var/dbb/config/git-jenkins.sh  fetch  --tags  --progress  https://github.com/git-user/dbb-
zappbuild-private.git +refs/heads/*:refs/remotes/origin/*
Seen branch in repository origin/development
Seen 1 remote branch
 > /var/dbb/config/git-jenkins.sh show-ref --tags -d # timeout=120
Checking out Revision 22ce6fa32663e9f428b8ad0b2a69652b6d900aba (origin/development)
 > /var/dbb/config/git-jenkins.sh config core.sparsecheckout # timeout=120
 > /var/dbb/config/git-jenkins.sh checkout -f 22ce6fa32663e9f428b8ad0b2a69652b6d900aba
Commit message: "Merge pull request #14 from jbyibm/development"
 > /var/dbb/config/git-jenkins.sh rev-list --no-walk 22ce6fa32663e9f428b8ad0b2a69652b6d900aba
# timeout=120
Finished: SUCCESS
```

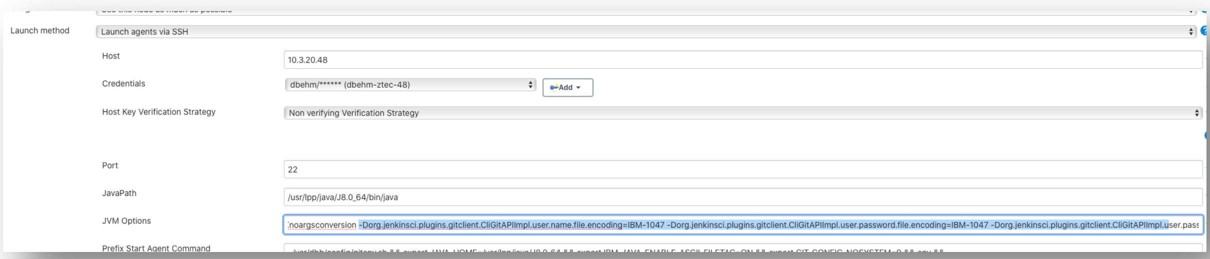
Using the Jenkins Git client plugin 3.4.1 onwards

If you are using the Jenkins Git client plugin 3.4.1 or later, you can configure the codepage for the git client plugin via a set of environment variables.

1. Open the Jenkins remote agent configuration and locate the JVM properties.

2. Add the new JVM properties:

```
-Dorg.jenkinsci.plugins.gitclient.CliGitAPIImpl.user.name.file.encoding=IBM-1047
-Dorg.jenkinsci.plugins.gitclient.CliGitAPIImpl.user.password.file.encoding=IBM-1047
-Dorg.jenkinsci.plugins.gitclient.CliGitAPIImpl.user.passphrase.file.encoding=IBM-1047
```


Launch method: Launch agents via SSH
Host: 10.3.20.48
Credentials: dbehm/****** (dbehm-ztec-48)
Host Key Verification Strategy: Non verifying Verification Strategy
Port: 22
JavaPath: /usr/lpp/java/J8.0_64/bin/java
JVM Options: :noargsconversion -Dorg.jenkinsci.plugins.gitclient.CliGitAPIImpl.user.name.file.encoding=IBM-1047 -Dorg.jenkinsci.plugins.gitclient.CliGitAPIImpl.user.password.file.encoding=IBM-1047 -Dorg.jenkinsci.plugins.gitclient.CliGitAPIImpl.user.pass
Prefix Start Agent Command

Please also see https://plugins.jenkins.io/git-client/#plugin-properties

You can now remove the git-jenkins.sh from the tool location of your node config and directly invoke rocket git.

Please note: These new JVM properties are not compatible with git-jenkins2.sh! You need to remove the git-jenkins2.sh from the tools configuration in Jenkins.

Using the Jenkins Git client plugin 2.x

In older setups of Jenkins and the Jenkins Git client plugin, the default wrapper script git-jenkins.sh addresses the communication via SSH to the central git provider.

With git-jenkins3.sh, both ways to communicate with the central git provider is possible.

Appendix

The appendix contains two additional scenarios. First, some background information about credentials required on the command line. Second, how to configure a credential helper for the build user on the build machine – which implies, that credentials don't need to be managed in Jenkins.

Cloning a private GitHub repository from the command line

Let's have a look to a simple HTTPS scenario cloning a private repository which requires to provide credentials as username and password.

Note that only private repositories require to provide credentials, public repositories don't. Therefore, the next steps apply only to private repositories using HTTPS.

For this documentation, a copy of the dbb-zappbuild repository has been setup as a private repository on GitHub. We refer to it using this URL: https://github.com/git-user/dbb-zappbuild-private

However, this document is not limited to GitHub as your central git provider. The same rules apply when using Bitbucket, GitLab or any other central git provider.

Like on all other platforms, Git will prompt the user to provide the username and password to access the repository. This is highlighted in bold:

```
IBMUSER:/u/ibmuser/git-repos: >git clone https://github.com/git-user/dbb-zappbuild-private.git
Cloning into 'dbb-zappbuild-private'...
Username for 'https://github.com':
Password for 'https://git-user@github.com':
remote: Enumerating objects: 365, done.
remote: Counting objects: 100% (365/365), done.
remote: Compressing objects: 100% (144/144), done.
remote: Total 365 (delta 213), reused 365 (delta 213), pack-reused 0
Receiving objects: 100% (365/365), 141.23 KiB | 1.70 MiB/s, done.
Resolving deltas: 100% (213/213), done.
```

If you are tired of entering your credentials each time when you interact with the remote repository, you can use a credential helper.

Configure a credential helper to manage the credentials to git

The following steps describe one of the options to store the credentials in a credential helper. Please have a look at the git scm documentation.[9] [10]

Find a helper

[9] https://git-scm.com/docs/gitcredentials
[10] https://git-scm.com/book/en/v2/Git-Tools-Credential-Storage

```
IBMUSER:/u/ibmuser/git-repos: >git help -a | grep credential-
  credential-cache          remote-ext
  credential-cache--daemon  remote-fd
  credential-store          remote-ftp
```

Set the global config to use the credential-helper and map the http url to a userid

```
IBMUSER:/u/ibmuser/git-repos: >git config --global credential.helper 'store –
file ~/.my-credentials'
IBMUSER:/u/ibmuser/git-repos: >git config --global credential.https://github.com git-user
IBMUSER:/u/ibmuser/git-repos: >git config --global -l
http.sslverify=false
user.name=Dennis Behm
user.email=dennis.behm@de.ibm.com
credential.https://github.com=git-user
credential.helper=store --file ~/.my-credentials
```

Attention: If you, use this helper store, the password is stored **unencrypted** in **`~/.my-credentials`**. 11

The first time you interact with the repository, git will prompt you for the credentials and stores them in the credential-store and will continue to use it from now on.

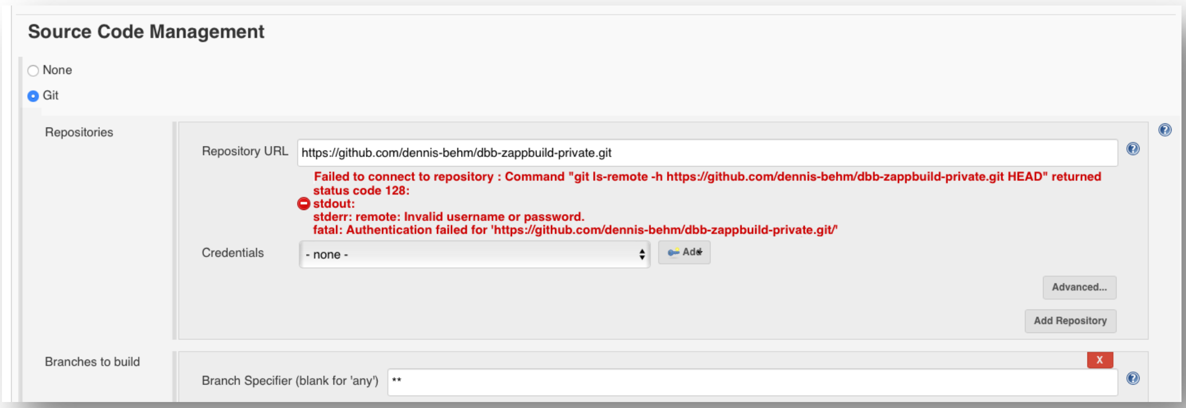
[11] https://git-scm.com/docs/git-credential-store

Use the git credential helper on the build machine to communicate with git

In the first section, we roughly have explained the configuration of a git credential helper when communicating via HTTP(s).

If you have performed the above steps for the technical user running the Jenkins agent on the build machine, you don't need to supply any credentials in Jenkins at all, as long you don't point your Jenkins job configuration to a Jenkinsfile within the repository.

You can ignore the warning displayed in the Jenkins job configuration.



Source Code Management

None [✓ Git]

Repositories

Repository URL: https://github.com/dennis-behm/dbb-zappbuild-private.git

```
Failed to connect to repository : Command "git ls-remote -h https://github.com/dennis-behm/dbb-zappbuild-private.git HEAD" returned
status code 128:
stdout:
stderr: remote: Invalid username or password.
fatal: Authentication failed for 'https://github.com/dennis-behm/dbb-zappbuild-private.git/'
```

Credentials: - none -

Branches to build

Branch Specifier (blank for 'any'): **

When executing the job, it will correctly clone the repo using the credentials stored in the Credential helper.

Attention: This is might not be a desired scenario, storing credentials on several target build machines.
Be aware that the credentials stored in this scenario are not encrypted. You should restrict read access to that file.

```
Started by user unknown or anonymous
Building remotely on z-environment (ztec-builder) in workspace
/var/dbb/buildhome/workspace/simpleGitClone
[WS-CLEANUP] Deleting project workspace...
[WS-CLEANUP] Done
No credentials specified
Cloning the remote Git repository
Cloning repository https://github.com/git-user/dbb-zappbuild-private.git
 > /var/dbb/config/git-jenkins.sh init /var/dbb/buildhome/workspace/simpleGitClone #
timeout=120
Fetching upstream changes from https://github.com/git-user/dbb-zappbuild-private.git
 > /var/dbb/config/git-jenkins.sh --version # timeout=120
 > /var/dbb/config/git-jenkins.sh fetch --tags --progress https://github.com/git-user/dbb-
zappbuild-private.git +refs/heads/*:refs/remotes/origin/*
 > /var/dbb/config/git-jenkins.sh config remote.origin.url https://github.com/git-user/dbb-
zappbuild-private.git # timeout=120
 > /var/dbb/config/git-jenkins.sh config --add remote.origin.fetch
+refs/heads/*:refs/remotes/origin/* # timeout=120
 > /var/dbb/config/git-jenkins.sh config remote.origin.url https://github.com/git-user/dbb-
zappbuild-private.git # timeout=120
```

```
Fetching upstream changes from https://github.com/git-user/dbb-zappbuild-private.git
 >  /var/dbb/config/git-jenkins.sh  fetch  --tags  --progress  https://github.com/git-user/dbb-
zappbuild-private.git +refs/heads/*:refs/remotes/origin/*
Seen branch in repository origin/development
Seen 1 remote branch
 > /var/dbb/config/git-jenkins.sh show-ref --tags -d # timeout=120
Checking out Revision 22ce6fa32663e9f428b8ad0b2a69652b6d900aba (origin/development)
 > /var/dbb/config/git-jenkins.sh config core.sparsecheckout # timeout=120
 > /var/dbb/config/git-jenkins.sh checkout -f 22ce6fa32663e9f428b8ad0b2a69652b6d900aba
Commit message: "Merge pull request #14 from jbyibm/development"
First time build. Skipping changelog.
Finished: SUCCESS
```