# How to dynamically pick a git branch to use in Jenkins build

I'm trying to create a new project configuration for Jenkins build server. To simplify what I'm trying to do, I will use only two components to describe the problem.

#### ComponentA

- Change in this component triggers the build of this project on CI server.
- CI server has statically configured branch to monitor for changes and build. Eg. master or develop branch.
- This component contains a configuration file with required version of ComponentB it depends on.

#### ComponentB

- Changes to this component don't trigger build of this project on CI server (there will be another project to cover development of ComponentB).
- Individual versions of component are tagged
- · ComponentA has required version of ComponentB in its configuration file
- CI server does not know what branch (tag) to checkout until configuration file of ComponentA is somehow parsed.

What is the right way to achieve this on Jenkins? I was trying to find out how to add this dynamic behavior of parsing the config file and making Git Plugin to check out branch based on expected version of ComponentB but so far I have no clue.

In the next step I may even want to have wildcards (like 5.3.\*) in configuration file so I will have to find a the newest ComponentB's tag matching the wildcard.

#### **EDIT**

Now I see that I simplified my problem too much and due to the simplification, the main limitation is no longer present.

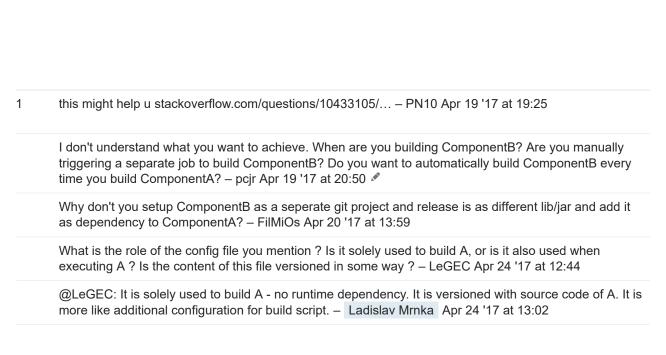
The main limitation is that Component A and B must be built together. It is not possible to build them separately as they form one executable / library and the build script needs source files from both components.

If you ask why such a strange configuration, let's give Component A and B some description:

- ComponentA: Native platform specific code
- ComponentB: Native platform independent code

continuous-integration

There may be many Component As - one for each platform, but just single Component B. Merging particular A with B produces full source code for single platform but not every platform may be updated to the latest version of B so it needs to have control over which version of B should be used for built.



edited Apr 24 '17 at 11:41



### 3 Answers

jenkins

build

One option to achieve what you want is to use the following setup:

Create two Jenkins jobs:

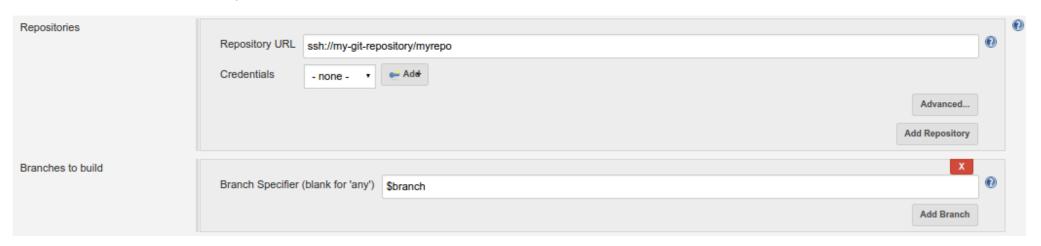
- "Component A" (automatically triggered on SCM changes)
- "Component B" ("manually" triggered)

## Step #1

Define the branch build parameter for "Component B":



Use this parameter as the "Git Plugin" branch specifier:



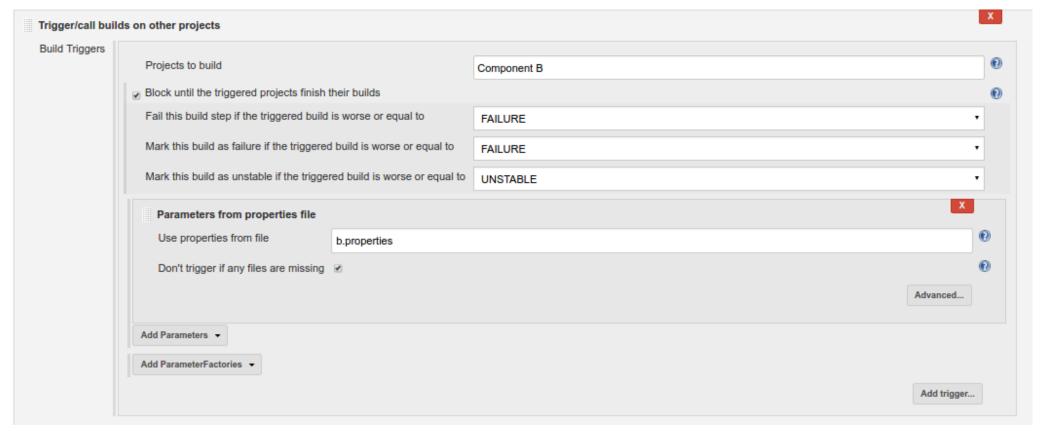
Now you should be able to manually trigger "Component B" build, by specifying a proper branch (tag) parameter to it, e.g. tags/5.3.0.

## Step #2

Add a new "Execute Shell" build step to your "Component A" build, which will extract the "Component B" version from the configuration file in the workspace, and prepare b.properties file with the "Component B" build parameters.



Install a Parameterized Trigger Jenkins plugin, and add a new "Trigger/call builds on other projects" build step to "Component A" job:



Using your b.properties file as the source of build parameters.

Now each time "Component A" is re-build, a new "Component B" build will get triggered, with the target branch/tag as a build parameter.

## Adding wildcard support

If you want to support wildcard versions, you can use git 1s-remote command to find the latest tag, like that:

```
#B=$(obtain B version from the config file in a usual way)

LATEST=$(\
    git ls-remote --tags YOUR_REPOSITORY_URL "$B"\
    |cut -d / -f3|sort -r --version-sort|head -1\)
)

cat <<EOF > b.properties
    branch=tags/$LATEST

EOF
```

This will list all the tags, matching "B" version pattern, in the remote "Component B" repository, and save the latest version number in the LATEST variable.

Add this to your "Execute Shell" step of the "Component A" job, and it should be able to handle version numbers patterns like: 5.3.\*

The catch is that the shell script will run as the Jenkins daemon user, so it must have proper credentials configured, to access the remote Git repository (e.g. via the ssh pubkey).

Alternatively you may want to look into the Credentials Binding Plugin, to reuse the Git credentials stored in Jenkins itself.

### Using Jenkins 2.0 style pipeline

You can also solve the task at hand by using a Jenkins 2.0-style Pipeline, which will allow you to checkout the code for components A and B, into a single workspace, and then apply some common build step to them.

Your pipeline could look something like this:

```
node {
  //Settings
  def credentialsId = '8fd28e34-b04e-4bc5-874a-87f4c0e05a03'
  def repositoryA = 'ssh://git@stash.com/projects/a.git'
  def repositoryB = 'ssh://git@stash.com/projects/b.git'
  stage('Checkout component A') {
     git credentialsId: credentialsId ,
     url: repositoryA , branch : "master"
   stage("Resolve and checkout component B") {
     def deps = readProperties file: 'meta.properties'
     echo "Resolved B version = ${deps['b']}"
     dir("module/b") {
           //Clone/Fetch Component B
           checkout scm:[
               $class: 'GitSCM',
                userRemoteConfigs: [[url: repositoryB, credentialsId: credentialsId]],
                branches: [[name: 'refs/tags/*']]
           ],
           changelog: false, poll: false
           //Checkout the tag, matching deps['b'] pattern
           sshagent([credentialsId]) {
                sh "git checkout \s(git tag -1 \s(b'))" |sort -r --version-sort|head -1)"
          }
   stage("Build A+B") {
        //Apply a common build step
```

Here we use the "readProperties" command, which is part of the Pipeline Utility Steps Plugin to extract the "Component B" version pattern from <code>meta.properties</code>. There are also readYaml, readJSON commands available.

Next we fetch/clone the "Component B", with the changelog: false, poll: false flags, to prevent it from being registered for the SCM poll, into the "module/b" folder of the current workspace.

Then invoke a shell command to select the tag, based on the version pattern, we have obtained above, and checkout it (5.3.\* style wildcards should also work).

The sh invocation, is wrapped in the sshagent, to make it reuse the appropriate credentials from the Jenkins credential store.

edited Apr 24 '17 at 20:35



```
Thanks, that look great but I assume it will not work for my use case - check updated question. – Ladislav Mrnka Apr 24 '17 at 11:32
```

@LadislavMrnka Yep, I see now. What version of Jenkins are you running 1.X or 2.X? – zeppelin Apr 24 '17 at 17:40

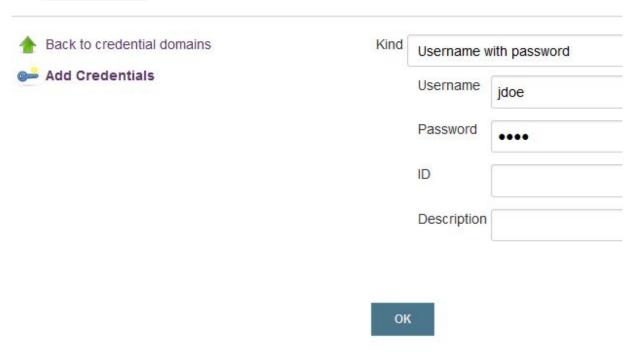
@LadislavMrnka If you are using Jenking 2.X, please check my update on using the Pipeline. – zeppelin Apr 24 '17 at 20:38

Using the Credentials Binding Plugin worked very well for me (also mentioned by @zeppelin)

### Steps:

### In the Global credentials section:

1. Add Credentials of the type: "Username with password". This should be the username and password for component B repository git server using HTTPS prote



### In your Jenkins job configuration:

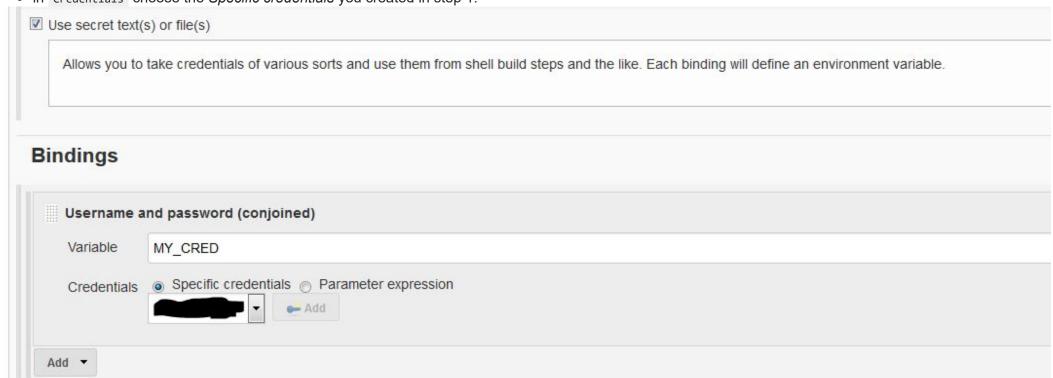
Put component A in the regular **Source Code Management** under the Git section all required fields (*Repositories*, *Branches*, etc.).

• It will clearer and cleaner to put the repository in a sub-directory: under **Additional Behaviours** choose Check out to a sub-directory and write: component\_a

Make sure also to check in **Build Triggers** the Build when a change is pushed to GitHub

In the Build Environment section tick the Use secret text(s) or file(s)

- put in Variable some name: MY\_CRED
- in Credentials choose the Specific credentials you created in step 1.



Now using the MY\_CRED in the **Execute shell** code you will have access to the component B repository:

```
DIR="component_b"
if [ "$(ls -A $DIR/.git)" ]; then
    cd $DIR
    git fetch
else
    git clone https://$MY_CRED@github.com/proj/component_b.git $DIR
    cd $DIR
fi
git show
```

```
Command

pwd
DIR="./ccc_test_clone"

if [ "$(ls -A $DIR/.git)" ]; then
cd ccc_test_clone
git fetch
else
git clone https://$MY_CRED@github
cd ccc_test_clone
fi
git show
ls -l
```

• **Note**: you will NOT see the user and password in the logs, so it should be safe. you would see: git clone 'https://\*\*\*\*@github.com/proj/component\_b.git' compo

Do all your parsing of config from component A to get the desired tag: TAG=\$(cat ./component\_a/config.cfg | grep ... | sed ...)

Checkout the desired Tag: cd component\_b; git checkout -f \$TAG

• Note: the -f force tag.

Now run the code and test as desired...

@LadislavMrnka regarding the question edit: you still can use this and during step 8 you have both codes of component A and B, and you can compile, run and tets.. – Chananel P Apr 26 '17 at 7:40

1 - would adding project B as a sub repo of project A be a possible solution?

2- (if including the full source code for B should really be avoided): would pushing builds of B to some B\_builds repo, and adding this repo as a sub-repo of A be a possible solution?

Rationale: one way to make the dependency between A and B more explicit is to represent it inside the dependencies of the repository.

This would require to add an extra step when managing the A project :

```
update `B` sub repo in `A` project, and push this to `A`
```

each time you produce a new version for B.

However, you would have a clear view, from A, about when the versions of B were integrated (e.g: "we only used B 2.0.1 starting from A 4.3.2"), and pushing to A would trigger your usual Jenkins flow.

answered Apr 24 '17 at 15:47

