A real example of Jenkins active choices and reactive parameter

BART.JAKUBOWSKI.IN
30 JUN 2019 • 5 MIN READ

I've struggled with this for a while but finally made it!

Main goal was to read a list of docker images tags from a private docker registry, each time a Jenkins job is executed.

Decided to use a **Active Choice plug-in.**

This plug-in provides several additional Jenkins parameter types that can be rendered as user interface (UI) controls in job forms. It supports using system environment variables, global node properties, and you also have at least the Jenkins project in the Groovy context for Freestyle jobs.

For my use case Active Choice Reactive Parameter will be a best fit as it dynamically generate value options for a build parameter using a Groovy script. In addition it can be dynamically updated when the value of other job UI control change.

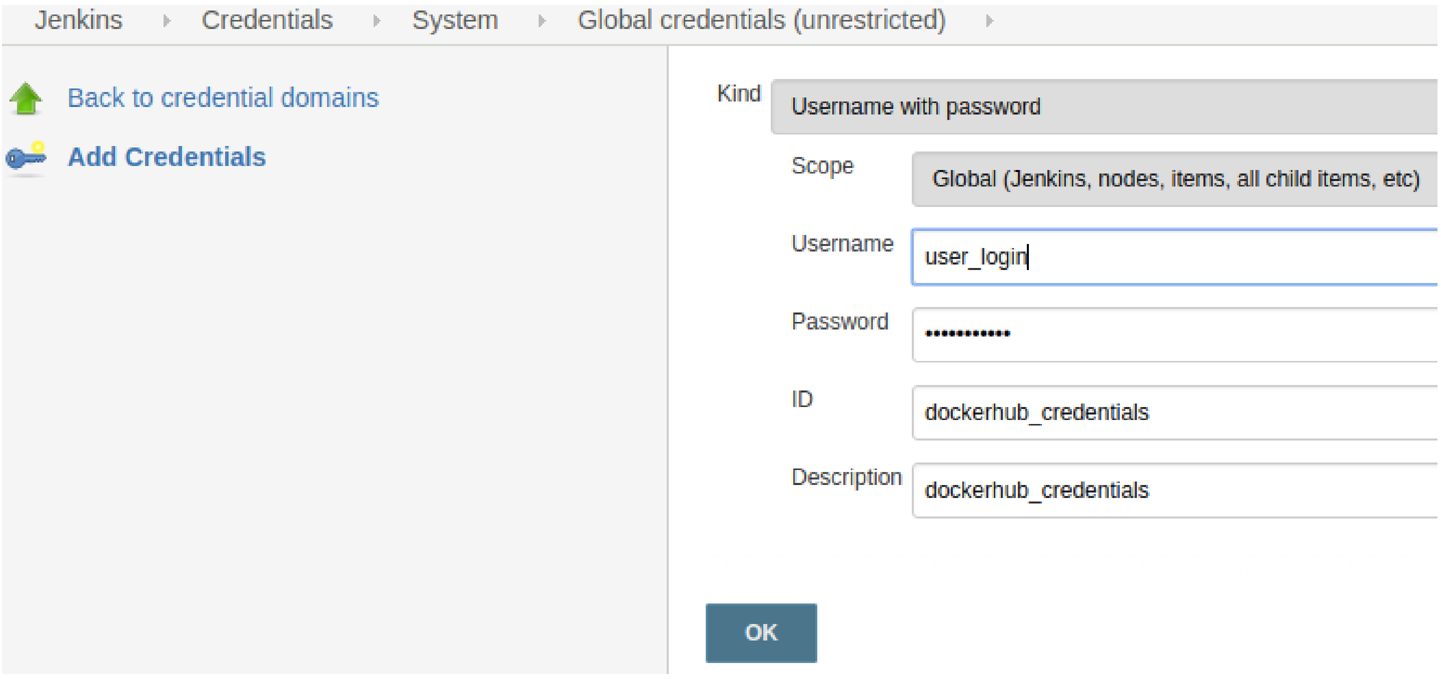
Requirements

Project type
According to the plugin description it requires our job will be a freestyle job.

Parameters to define
Lets assume the following params will be needed:

- Service - our selected service
- Tag - list of tags from registry
- Environment - place where its gonna be deployed

Docker credentials
Because registry is private and I don't want to keep credentials in any kind of scripts or jobs my idea is to use credentials store in Jenkins. Adding my docker username and password:

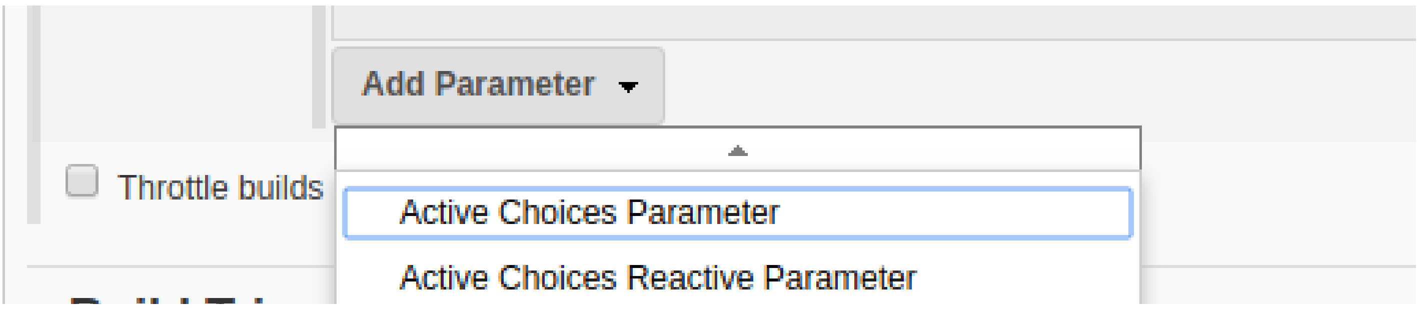


Kind: Username with password
Scope: Global (Jenkins, nodes, items, all child items, etc)
Username: user_login
Password: •••••••••••
ID: dockerhub_credentials
Description: dockerhub_credentials

Now I will be able to use it inside a script just by using a credential ID.

Creating the Job

On the beginning let's tick `This project is parameterized` and start adding params.
From the UI side it will look like this:



Build Trigg

Active Choices Reactive Reference Parameter

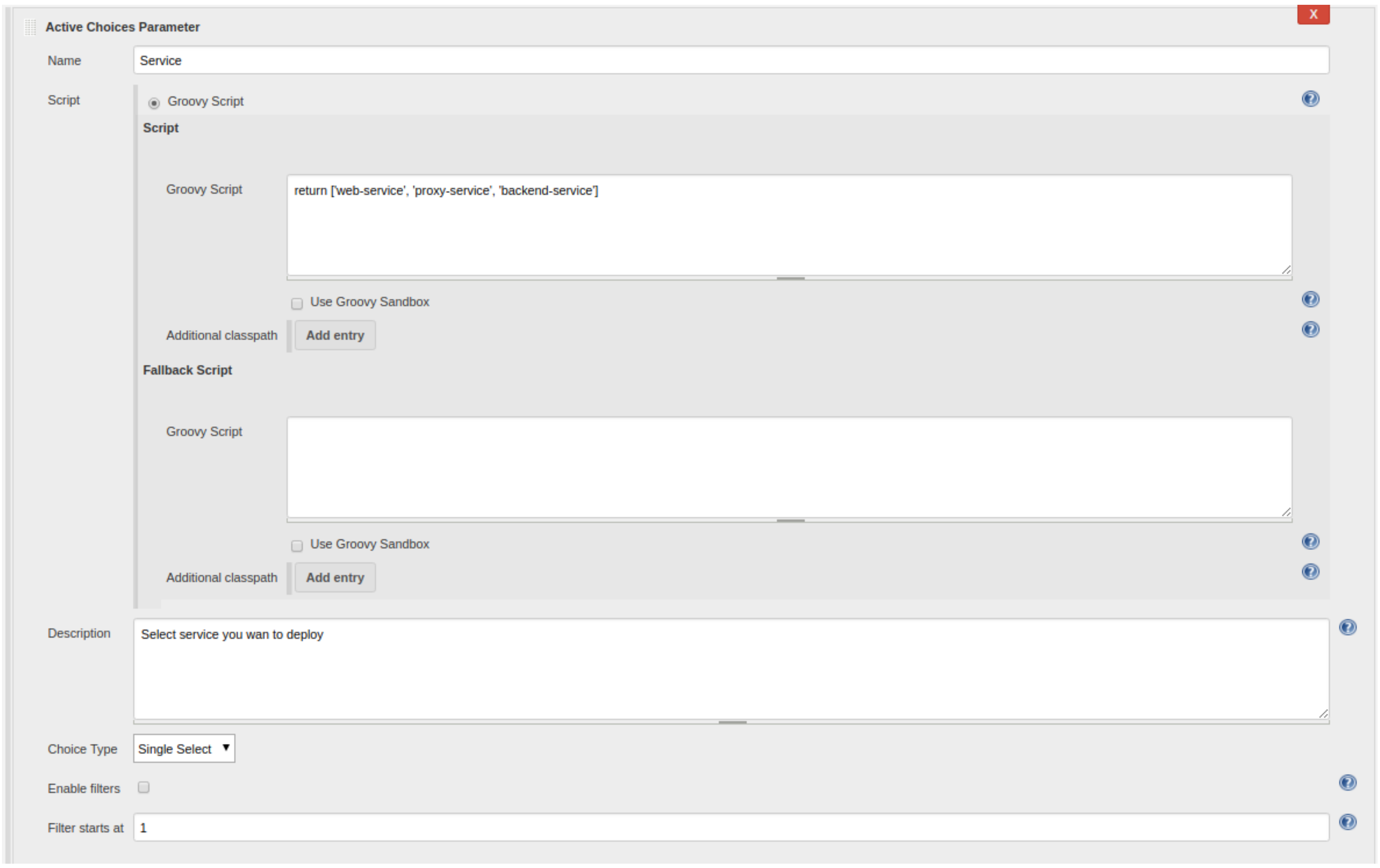
1. Service parameter
First parameter will allow to select the service. Based on this choice we will be getting a list of docker image tags in the next parameter.

This should be a `Active Choices Parameter` as we will use the name in the next one.
So the name for the param will be `Service` and we will use a `Groovy script` to return a list of values, like this:

```
return ['web-service', 'proxy-service', 'backend-service']
```

Fallback choice is something that can be skipped as its not possible that something will go wrong while returning a list :)
Choice type I'm setting to `Single Select`.

At the end it should look like this:


Active Choices Parameter

Name: Service

Script: (●) Groovy Script

Script

Groovy Script:

```
return ['web-service', 'proxy-service', 'backend-service']
```

☐ Use Groovy Sandbox

Additional classpath: Add entry

Fallback Script

Groovy Script:

☐ Use Groovy Sandbox

Additional classpath: Add entry

Description: Select service you wan to deploy

Choice Type: Single Select

Enable filters: ☐

Filter starts at: 1

2. Tag parameter
Second parameter will be the tag that we want to choose. Its slightly more complicated to get the data form a remote api with authentication so there will be move groovy involved.
This time we adding `Active Choices Reactive Parameter`
Name of this one is set to `Tag` and we will use another `Groovy script` like before to return a list of image tags:

```
import groovy.json.JsonSlurperClassic
import jenkins.model.Jenkins

image = imageName(Service)
token = getAuthTokenDockerHub()
tags = getTagFromDockerHub(image, token)
return tags

def getTagFromDockerHub(imgName, authToken) {
    def url = new URL("https://hub.docker.com/v2/repositories/${imgName}/tags?page=1&page_size=50")
    def parsedJSON = parseJSON(url.getText(requestProperties:["Authorization":"JWT ${authToken}"]))
    def regexp = "^\\d{1,2}.\\d{1,2}\$"
    parsedJSON.results.findResults {
    it.name =~ /$regexp/ ? "${it.name}".toString() : null
    }
}

def getAuthTokenDockerHub() {
    def creds = com.cloudbees.plugins.credentials.CredentialsProvider.lookupCredentials(
    com.cloudbees.plugins.credentials.common.StandardUsernameCredentials.class,
    Jenkins.instance,
    null,
    null)
    for (c in creds) {
    if (c.id == "dockerhub_credentials") {
        user = c.username
        pass = c.password
    }
    }
    def url = new URL("https://hub.docker.com/v2/users/login/")
    def conn = url.openConnection()
    conn.setRequestMethod("POST")
    conn.setRequestProperty("Content-Type", "application/json")
    conn.doOutput = true

    def authString = "{\"username\": \"${user}\", \"password\": \"${pass}\"}"
    def writer = new OutputStreamWriter(conn.outputStream)
    writer.write(authString)
    writer.flush()
    writer.close()
    conn.connect()

    def result = parseJSON(conn.content.text)
    return result.token
}
def parseJSON(json) {
    return new groovy.json.JsonSlurperClassic().parseText(json)
}
def imageName(name){
return "bartekj/${name}".toString()
}
```

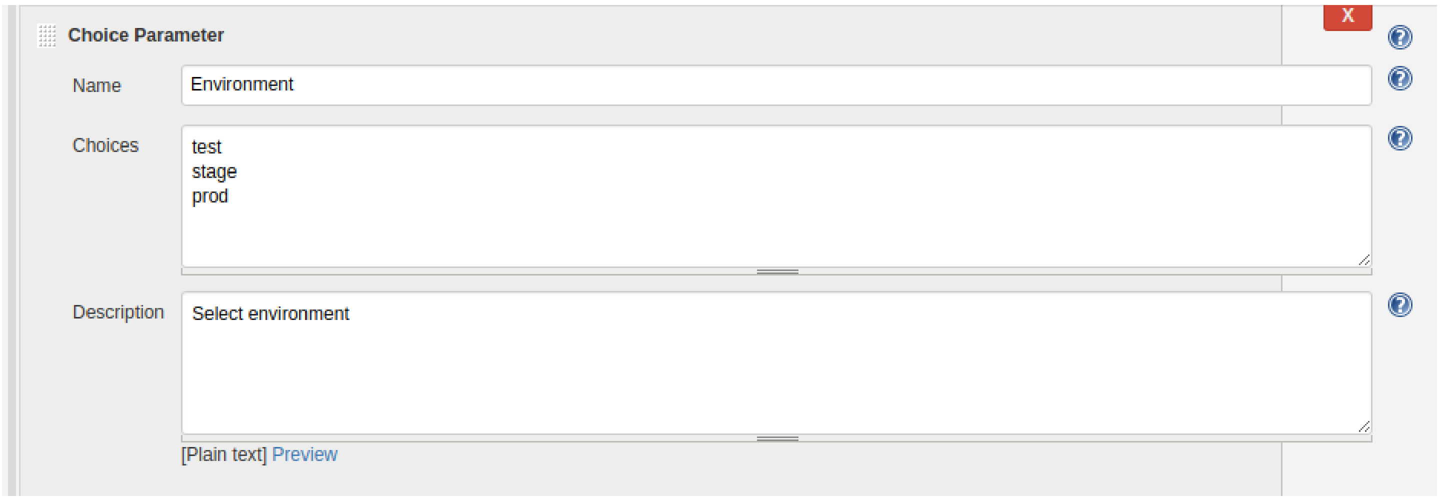
Another thing that need to be done before the above code will work is to define a `Referenced parameters` as the image variable depends on it.
I will not get into details that much as this script is pretty straight forward.
It puts an image name together based on `Service` parameter, obtaining a token from dockerhub using Jenkins credentials ID and parse json output with selected regex returning a list of docker image tags.

Same as before setting up `choice type` to Single Select.

3. Environment parameter

Last element that will be needed is a simple `Choice parameter` that together with previous params will be used inside Jenkinsfile after the job will be executed.

Name: `Environment` and choices (one per line): `test stage prod`.
Should look like this:


Choice Parameter

Name: Environment

Choices:
test
stage
prod

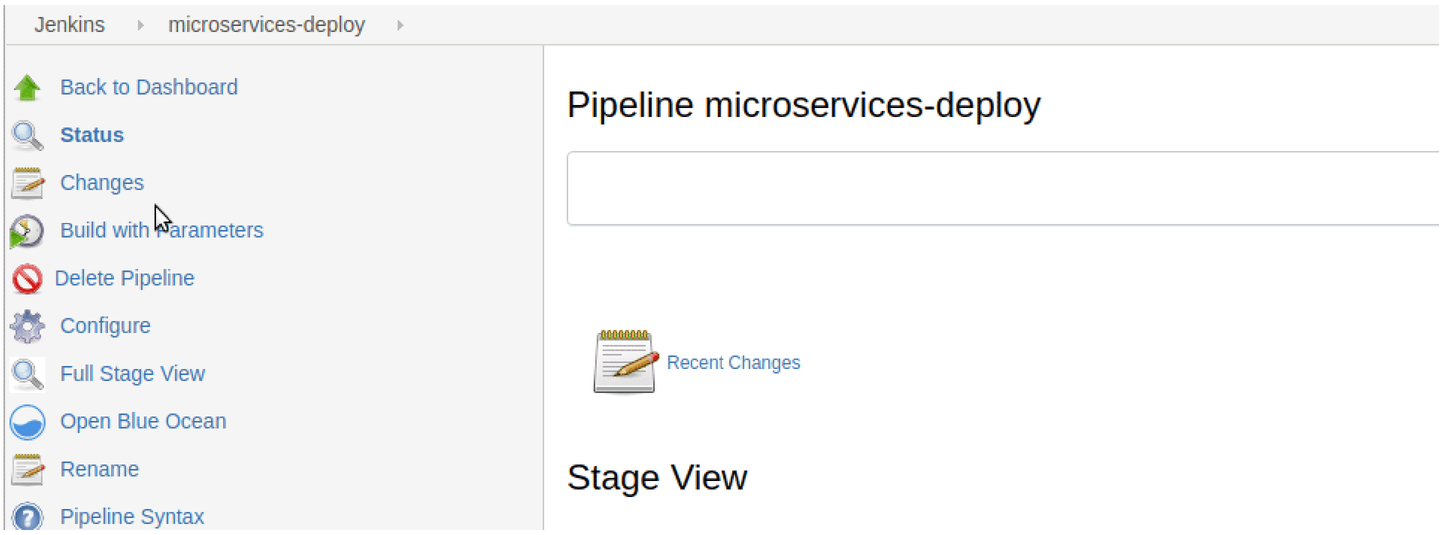
Description: Select environment

[Plain text] Preview

Putting everything together

Putting everything together there is one case to remember (depending on your Jenkins security config). We are using groovy scripts in a job so after first run go to `Manage Jenkins >> In-process Script Approval` and approve the code that was used here (possibly scripts and fallback).

When everything is ready it should look like this:



Extra: job DSL definition

For those who use here is a snippet for DSL. (without scm definition and other stuff)

```
job('microservices-deploy') {
    parameters {
        activeChoiceParam('Service') {
            description('Select service you wan to deploy')
            choiceType('SINGLE_SELECT')
            groovyScript {
                script('return ['web-service', 'proxy-service', 'backend-service'])
                fallbackScript('"fallback choice"')
            }
        }
        activeChoiceReactiveParam('Tag') {
            description('Select tag from dockerhub')
            choiceType('SINGLE_SELECT')
            groovyScript {
                script('''
                    import groovy.json.JsonSlurperClassic
                    import jenkins.model.Jenkins

                    image = imageName(Service)
                    token = getAuthTokenDockerHub()
                    tags = getTagFromDockerHub(image, token)
                    return tags

                    def getTagFromDockerHub(imgName, authToken) {
                        def url = new URL("https://hub.docker.com/v2/repositories/${imgName}/tags?page=1&page_size=50")
                        def parsedJSON = parseJSON(url.getText(requestProperties:["Authorization":"JWT ${authToken}"]))
                        def regexp = "^\\d{1,2}.\\d{1,2}\$"
                        parsedJSON.results.findResults {
                        it.name =~ /$regexp/ ? "${it.name}".toString() : null
                        }
                    }

                    def getAuthTokenDockerHub() {
                        def creds = com.cloudbees.plugins.credentials.CredentialsProvider.lookupCredentials(
                        com.cloudbees.plugins.credentials.common.StandardUsernameCredentials.class,
                        Jenkins.instance,
                        null,
                        null)
                        for (c in creds) {
                        if (c.id == "dockerhub_credentials") {
                            user = c.username
                            pass = c.password
                        }
                        }
                        def url = new URL("https://hub.docker.com/v2/users/login/")
                        def conn = url.openConnection()
                        conn.setRequestMethod("POST")
                        conn.setRequestProperty("Content-Type", "application/json")
                        conn.doOutput = true

                        def authString = "{\"username\": \"${user}\", \"password\": \"${pass}\"}"
                        def writer = new OutputStreamWriter(conn.outputStream)
                        writer.write(authString)
                        writer.flush()
                        writer.close()
                        conn.connect()

                        def result = parseJSON(conn.content.text)
                        return result.token
                    }
                    def parseJSON(json) {
                        return new groovy.json.JsonSlurperClassic().parseText(json)
                    }
                    def imageName(name){
                    return "bartekj/${name}".toString()
                    }
                ''')
                fallbackScript('"fallback choice"')
            }
            referencedParameter('Service')
        }
        choiceParam('Environment', ['test', 'stage', 'prod'], 'Select environment')
    }
}
```

Enjoy!

Subscribe to bart.jakubowski.in

Get the latest posts delivered right to your inbox

youremail@example.com

Subscribe

MORE IN JENKINS

Jenkins security and script approvals
15 Apr 2019 – 2 min read

Remote git branches with groovy
18 Feb 2019 – 1 min read

See all 2 posts →

PYTHON

Find and delete slack messages

There will be future post about creating simple slack bots in python and this quick post is kind of related ;) Case is quite simple, we have a bot and it produced lot of crap on several channels. Here is a simple python snippet to

BART.JAKUBOWSKI.IN 28 OCT 2019 • 1 MIN READ

NVIDIA

Nvidia and AMD together on Linux

For a longer while I'm having multiple GPU's on a Linux desktop. Reason behind it is quite simple - I'm using it for crypto mining! And I must say it

BART.JAKUBOWSKI.IN 3 MAY 2019 • 3 MIN READ

[redacted]