Triggering Jenkins jobs remotely via git post-commit hooks

This document explains how we can trigger a Jenkins job from outside (such as GIT commit hooks or from a script). We will make use of "curl" command to trigger the job on Jenkins from command line and then on every new commit on source GIT repository.

Problem Scenario:

We want a Jenkins job to build automatically once there is a "commit" on GIT repository. This is normally done through "post-commit hooks" in GIT.

Prerequisites:

1. A user should be created/existing which will be used by remote curl commands
2. A Jenkins job with a GIT repository configured as SCM source.

Let's assume the following:

1. Jenkins job name – mytestjob
2. Username – testuser
3. Password – p@ssword
4. GIT repo name – mygitrepo

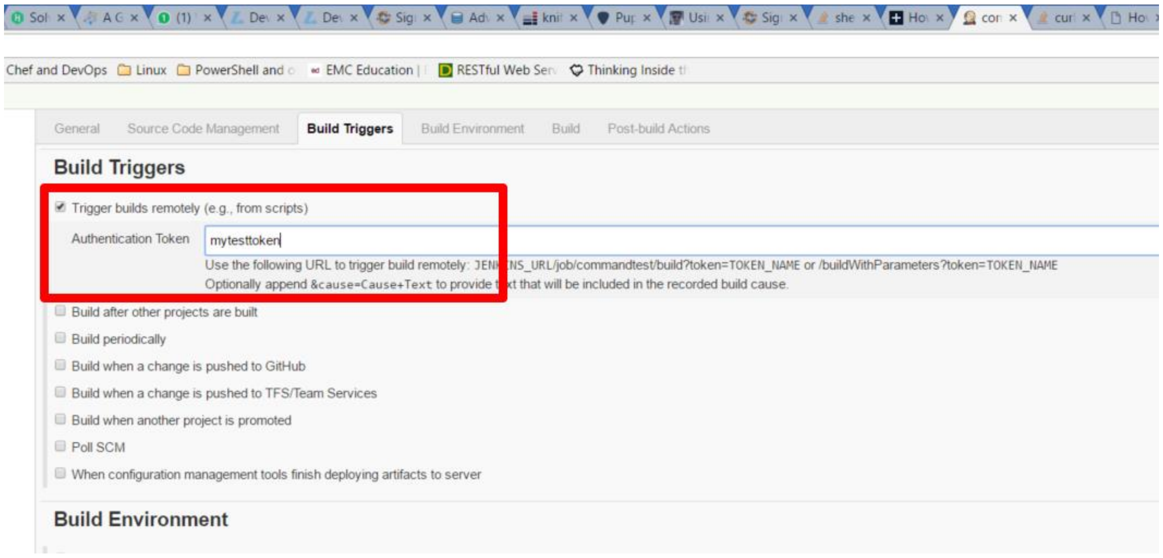
Standard Build:

1. Configure Jenkins job "**mytestjob**" to use "**mygitrepo**" to be used as SCM source.

2. Put a check against the following option under Build Trigger section of Jenkins job:

   "Trigger builds remotely (e.g. from scripts)"

3. Specify a random string as the authentication Token (let's assume "**mytesttoken**")



Build Triggers

- [✓] Trigger builds remotely (e.g., from scripts)

  Authentication Token: mytesttoken

  Use the following URL to trigger build remotely: `JENKINS_URL/job/commandtest/build?token=TOKEN_NAME` or `/buildWithParameters?token=TOKEN_NAME`
  Optionally append `&cause=Cause+Text` to provide text that will be included in the recorded build cause.
- [ ] Build after other projects are built
- [ ] Build periodically
- [ ] Build when a change is pushed to GitHub
- [ ] Build when a change is pushed to TFS/Team Services
- [ ] Build when another project is promoted
- [ ] Poll SCM
- [ ] When configuration management tools finish deploying artifacts to server

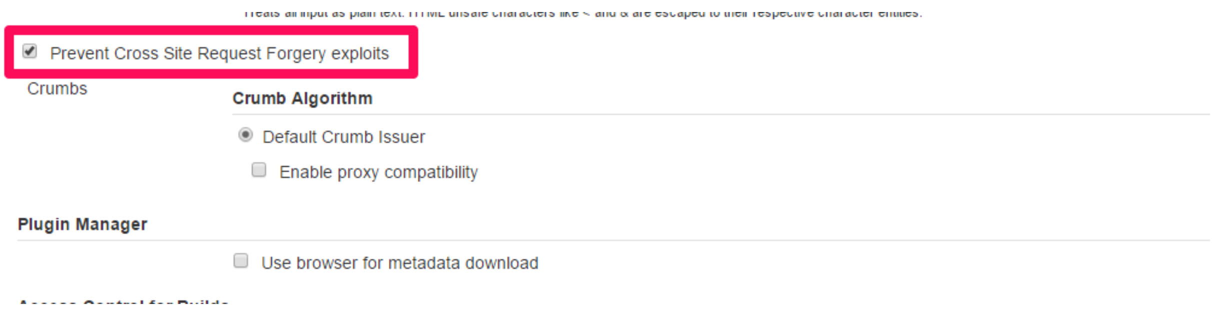
Build Environment

4. Save the job and issue the following command from command line to trigger the job from outside:

```
curl --user 'testuser:p@ssword' -X POST
"http://jenkinsurl.com:8080/job/mytestjob/build" -data
token=mytesttoken --data delay=0sec
```

Check the Jenkins WebUI. You should be able to see a new build already triggered for “mytestjob”

Note: if you get an error similar to `"HTTP ERROR 403 No valid crumb was included in the request"` it means that you have "Prevent Cross Site Request Forgery exploits" checked on the Jenkins "Configure Global Security" page. You need to uncheck that option. Since you're most probably not exposing your Jenkins instance to the world so that should be fine. (Refer screenshot below):



☑ Prevent Cross Site Request Forgery exploits

Crumbs

Crumb Algorithm

● Default Crumb Issuer

☐ Enable proxy compatibility

Plugin Manager

☐ Use browser for metadata download

Parameterized Build:

In case, Jenkins build is parameterized, then you need to specify each parameter in the curl command. Let's assume **mytestjob** has 2 parameters, ‘**node_name**’ and “**app_pool**’, then the curl command should look like following:

```
curl --user 'user1:password1' -X POST
"http://jenkins.mycompany.com:8080/job/myjob/build" -data
token=mytoken1 --data delay=0sec --data-urlencode json='{"parameter":
[{"name":"node_name", "value":"webserver1.com"},
{"name":"app_pool", "value":"myapppool"}]}'
```

You should be able to see “mytestjob” being triggered instantly on Jenkins webUI once you run above curl commands.

Note: You need to mention the parameters and values in above mentioned format even if those parameters have default values specified in the Jenkins job definition.

What are Git hooks?

Git hooks are scripts that Git executes before or after events such as: commit, push, and receive. Git hooks are a built-in feature - no need to download anything. Git hooks are run locally.

These hook scripts are only limited by a developer's imagination. Some example hook scripts include:

pre-commit	: Check the commit message for spelling errors.
pre-receive	: Enforce project coding standards.
post-commit	: Email/SMS team members of a new commit.
post-receive	: Push the code to production.

GIT Post-commit hook to trigger Jenkins Job:

Create a file called post-commit in your local `.git/hooks` directory under the repository from which you want to trigger the Jenkins job. The post-commit file is a regular bash script:

```
#!/bin/bash

curl --user 'testuser:p@ssword' -X POST
"http://jenkinsurl.com:8080/job/mytestjob/build" -data  token=mytesttoken   --
data delay=0sec
```

At this point, whenever you commit code in this repository, you should see the Jenkins job being triggered instantly.

Note: You might need to provide execute permission to your post-commit file in case it’s not there already.