Exercise – Jenkins and Git

Objective

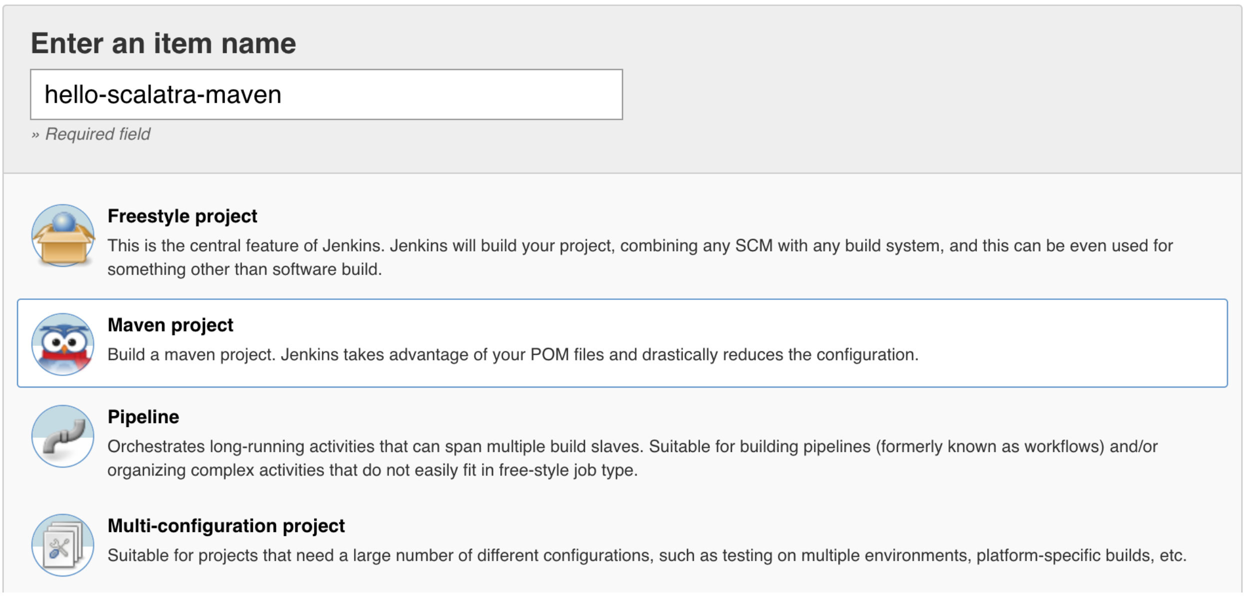
The objective of this exercise is to setup the git hooks and triggers for a project, secure Jenkins and setup source control backup for the Jenkins files.

Overview

Part 1 - Create a Maven Job

Step 1:

Click on "**Create new jobs**" or "**New Item**" to start creating a first new job. You need to give the job a name - use "**hello-scalatra-maven**" and select the "**Maven Project**" option from the list of project types below.



Enter an item name

hello-scalatra-maven

» Required field

Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Maven project
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

Pipeline
Orchestrates long-running activities that can span multiple build slaves. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Click **OK** to create the project. You will be taken to a set of configuration options.

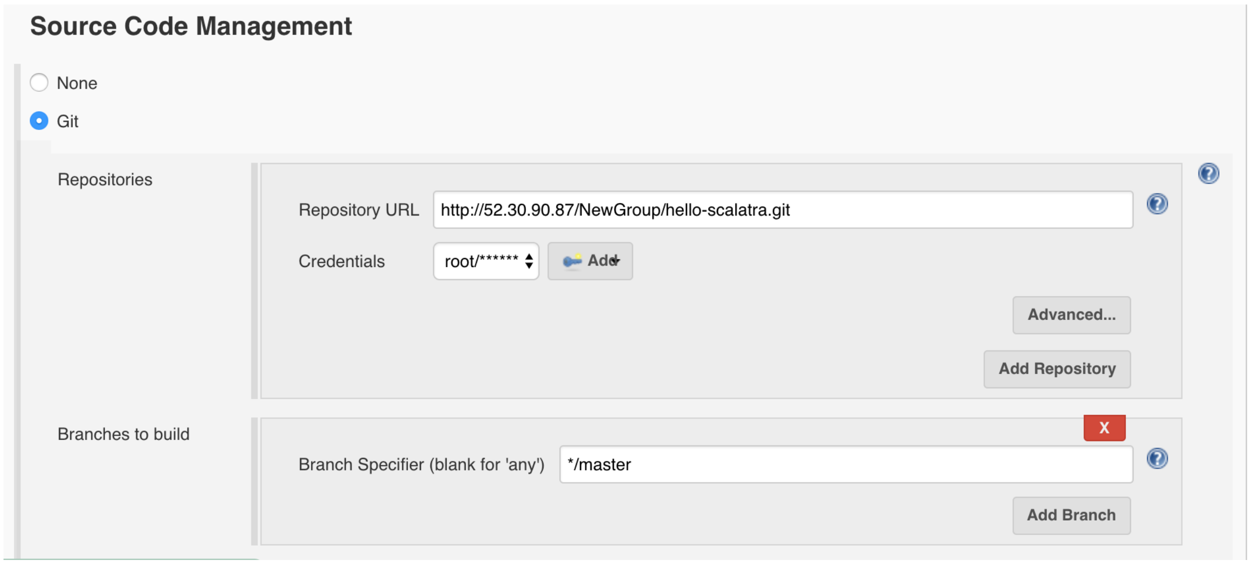
To select what source control server you use, you can choose and work further either:

1. Bitbucket
2. GitLab

Step 2:

1. IF YOU WORK WITH BITBUCKET:

Under source code management select "**Git**" and add your “**hello-scalatra**” git repository to this (not the one shown!)



Source Code Management

None

Git

Repositories

Repository URL: http://52.30.90.87/NewGroup/hello-scalatra.git

Credentials: root/******

Branches to build

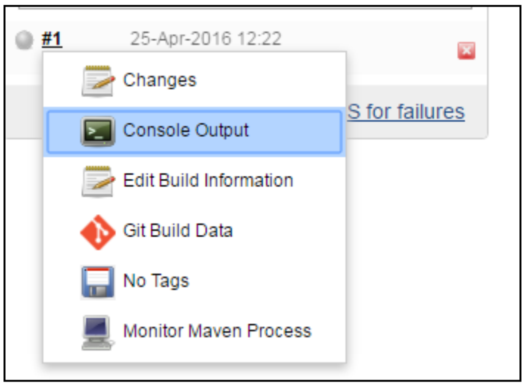
Branch Specifier (blank for 'any'): */master

Scroll down to build triggers and select the boxes for (as it is shown below):

1. Build when a change is pushed Bitbucket.

Click **save** to finish.

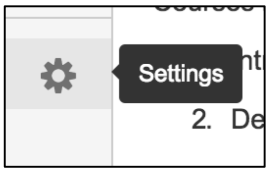
Manually build the project by clicking "**Build now**" from the left hand menu. You can see the log files by hovering over the build number and clicking the small arrow to the right, then selecting "**Console output**".



Bitbucket:

Now we have setup the listening side we need to configure Bitbucket to push a notification to Jenkins on a new commit:

1. Go to your Bitbucket repository. Click on the **settings icon** on the left of the screen.



2. Click on the **webhooks** menu option
3. Add a new **webhook**. Call it Jenkins and add the URL in the form:

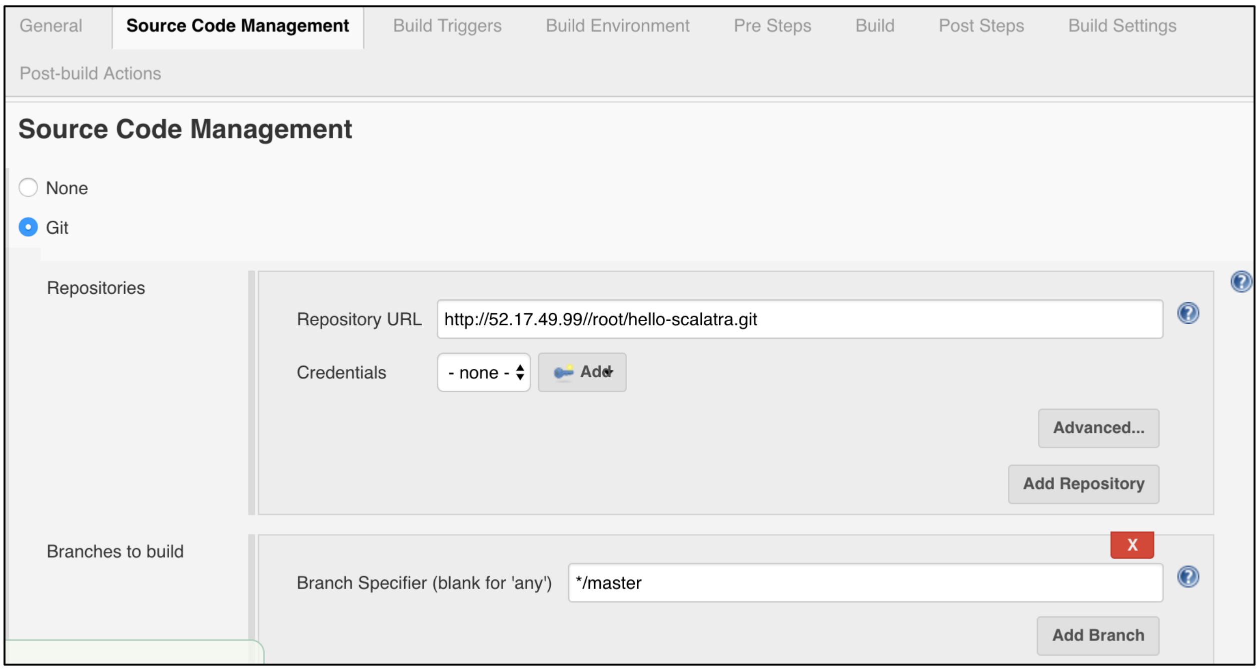
   http://[yourip]:8080/bitbucket-hook/

   Note: The end '/' is very important. If you forget this then it will not work.

2. IF YOU WORK WITH GITLAB

First you need to install "**Gitlab Hook Plugin**" first.

Under source code management select "**Git**" and add your **hello-scalatra git** repository to this from your GitLab server (not the one shown!)



Source Code Management

None
Git [✓]

Repositories

Repository URL: http://52.17.49.99//root/hello-scalatra.git

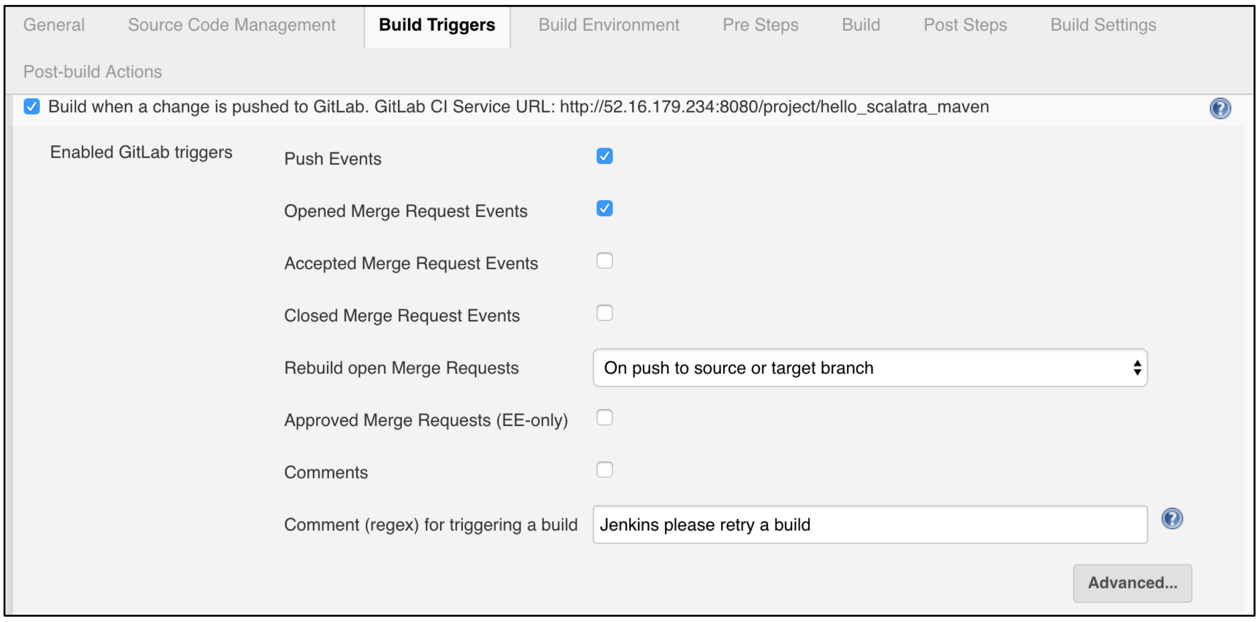
Credentials: - none -

Branches to build

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

Scroll down to build triggers and select the boxes for (as it is shown below):

1. Build when a change is pushed to GitLab. GitLab CI Service URL: *http://52.16.179.234:8080/project/hello_scalatra_maven*
    a. Select "Push Events"
    b. Choose "Rebuild open Merge Requests" to "On push to source or target branch"



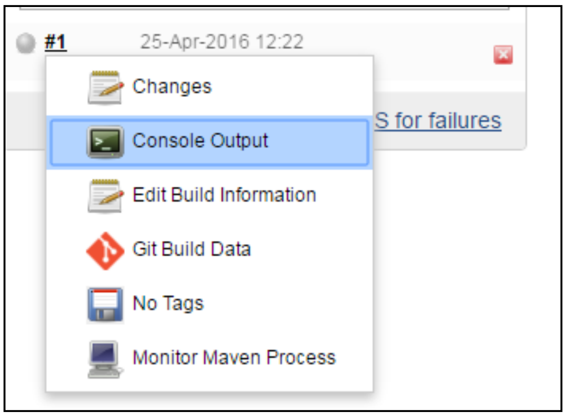
[✓] Build when a change is pushed to GitLab. GitLab CI Service URL: http://52.16.179.234:8080/project/hello_scalatra_maven

Enabled GitLab triggers

Push Events	✓
Opened Merge Request Events	✓
Accepted Merge Request Events	☐
Closed Merge Request Events	☐
Rebuild open Merge Requests	On push to source or target branch
Approved Merge Requests (EE-only)	☐
Comments	☐
Comment (regex) for triggering a build	Jenkins please retry a build

Click **save** to finish.

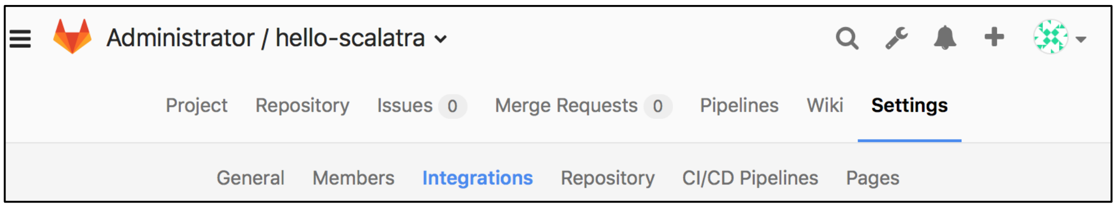
Manually build the project by clicking "**Build now**" from the left hand menu. You can see the log files by hovering over the build number and clicking the small arrow to the right, then selecting "**Console output**".



GitLab: (https://github.com/jenkinsci/gitlab-hook-plugin)

Now we have setup the listening side we need to configure GitLab to push a notification to Jenkins on a new commit.

1. Go to your bit GitLab "**hello-scalatra**" project. Click on the "**Settings**" on the right side menu and choose "**Integrations**".



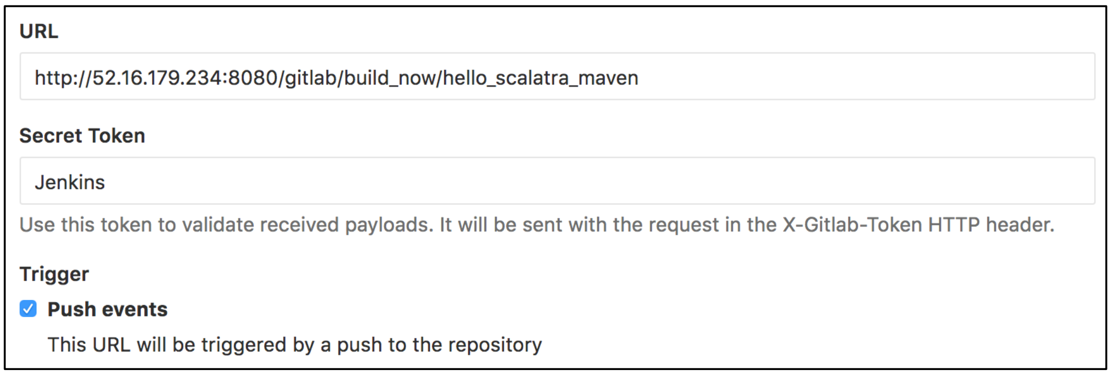
2. Call it Jenkins and add the URL in the form:
   http://<*your-jenkins-server-ip:8080*>/gitlab/build_now/Jenkins-job-name
   e.g. **http://52.16.179.234:8080/gitlab/build_now/hello_scalatra_maven**

   Give some meaningful name under "**Secret Token**". E.g. "Jenkins"

3. Select "**Push events**"

   Uncheck box for "**Enable SSL verification**"

4. Click "**Add Webhook**" button.



URL

http://52.16.179.234:8080/gitlab/build_now/hello_scalatra_maven

Secret Token

Jenkins

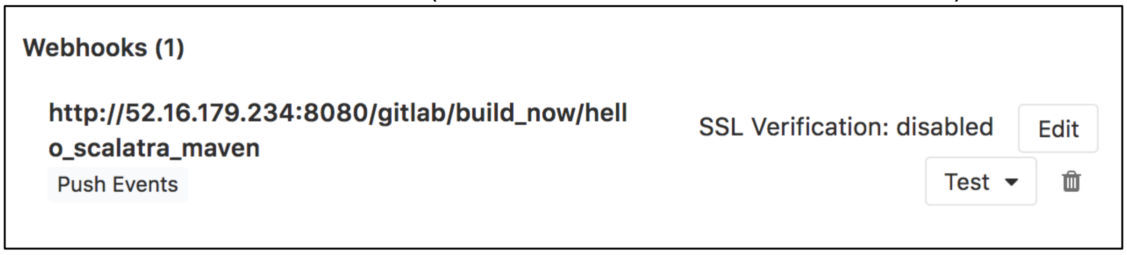
Use this token to validate received payloads. It will be sent with the request in the X-Gitlab-Token HTTP header.

Trigger

☑ Push events

This URL will be triggered by a push to the repository

5. Test it to be sure that it works (click "**Test**" button → "**Push events**")



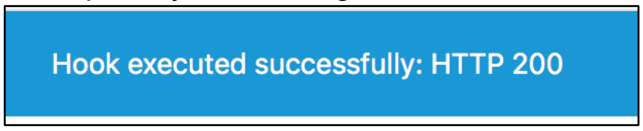
Webhooks (1)

http://52.16.179.234:8080/gitlab/build_now/hello_scalatra_maven

Push Events

SSL Verification: disabled

if test pass, you should get "**Hook executed successfully: HTTP 200**"



Hook executed successfully: HTTP 200

Step 3:

Go to your cloned copy of the "**hello-scalatra**" (BitBucket/GitLab or server CLI) project and edit one of the template files

```
hello-scalatra/ src/main/webapp/WEB-
INF/scalate/templates/views/index.ssp
```

Change the HTML code to something else (make sure it's still well formed!)

```
<h2>Hello, World!</h2>
```

6. Commit your file to git and push it to the repo. Watch to see if Jenkins picks up the change and builds your new file!

```
  $ git add .
  $ git commit -m "changes to the message"
  $ git push origin master
```

Part 2: Create a Pipeline Project

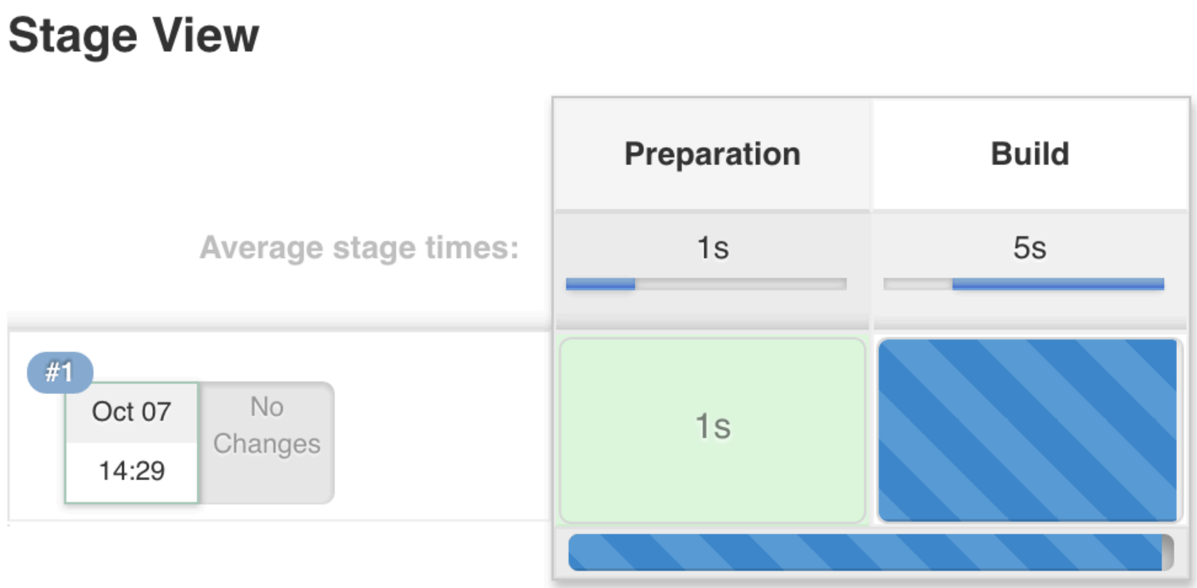
Jenkins has a set of options for pipeline projects - this is to tie in with existing DevOps pipelines.

1. From the dashboard "**create a new project**". Call this one "**hello-scalatra-pipeline**" and select the "**pipeline**" project type.
2. Under **Build Triggers** select the checkbox for *build when a change is pushed to bitbucket.*
3. In the **pipeline script** box we need to write a script.
   a. Click the "*try Sample Pipeline*" button and select "*Github + Maven*".
   b. Change the git line to use your project.
   c. Delete the stage for surefire report publishing, we have no tests so its not going to have anything to publish
   d. Then click **save**.

Your entire script should look similar to this:

```
node {
   def mvnHome
   stage('Preparation') {
      git 'YOUR-GIT-REPO-HERE'
      mvnHome = tool 'M3'
   }
   stage('Build') {

      if (isUnix()) {

         sh "'${mvnHome}/bin/mvn' -Dmaven.test.failure.ignore
clean package"

      } else {

         bat(/"${mvnHome}\bin\mvn" -Dmaven.test.failure.ignore
clean package/)

      }

   }

}
```

Click **build now** to see your project build and the pipeline stages complete.


Stage View
Preparation
Build
Average stage times:
1s
5s
#1
Oct 07
14:29
No Changes
1s