Continuous Delivery of PHP Applications With zs-client, Jenkins And Zend Server

Contents

Introduction

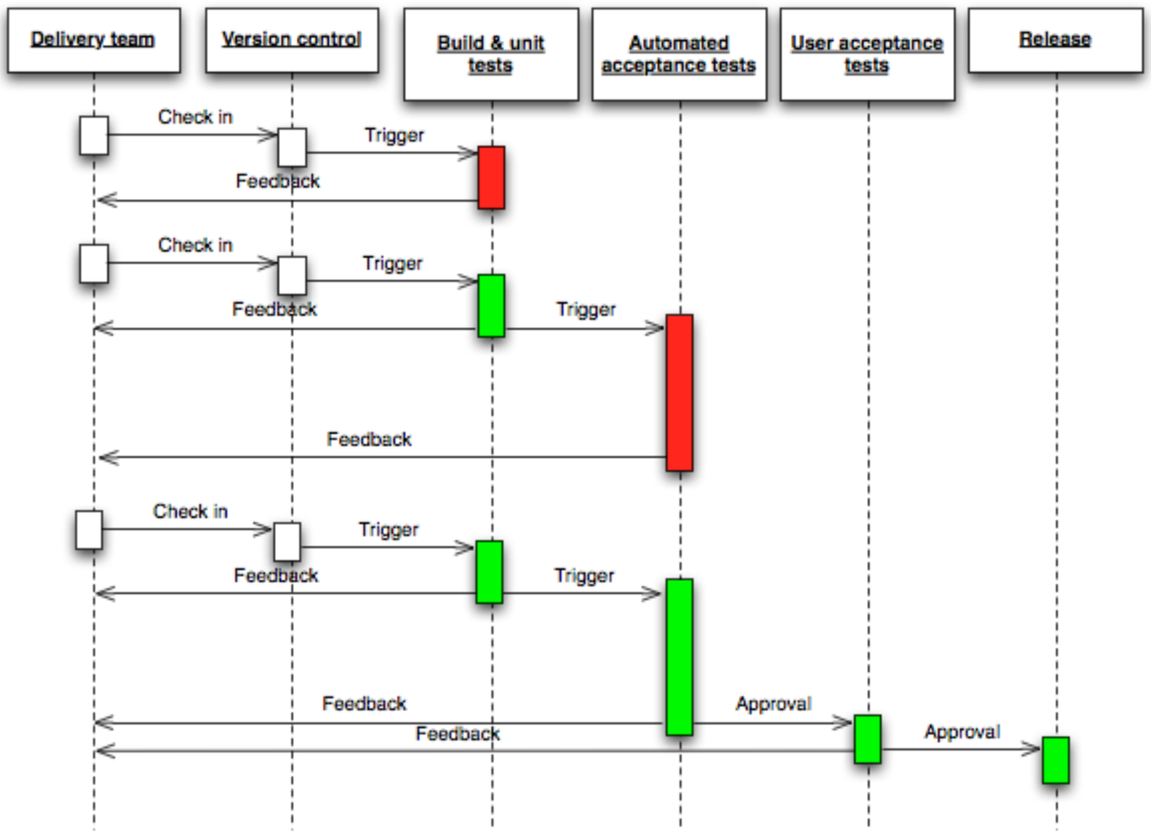
In our modern day world our products need to have fast time to market. This allows us to position our products before the other competitors, to push more often new features that keep the customers and to get fast feedback from the customers. This improves the end-customer experience, the quality and reduces the costs.

In the software world we already have existing methodologies that help the developers keep with this fast pace of development. The Agile movement created methodologies like SCRUM, eXtreme Programming, etc that are targeting this. With the rising complexity of deploying software on the production servers the speed gain that was achieved from the developers was lost in the operations side. Continuous delivery and DevOps are methodologies that tackle this problem on the operations side.

"DevOps is a methodology which helps foster collaboration between the teams that create and test applications (Dev) with those that maintain them in production environments (Ops). "[1] .

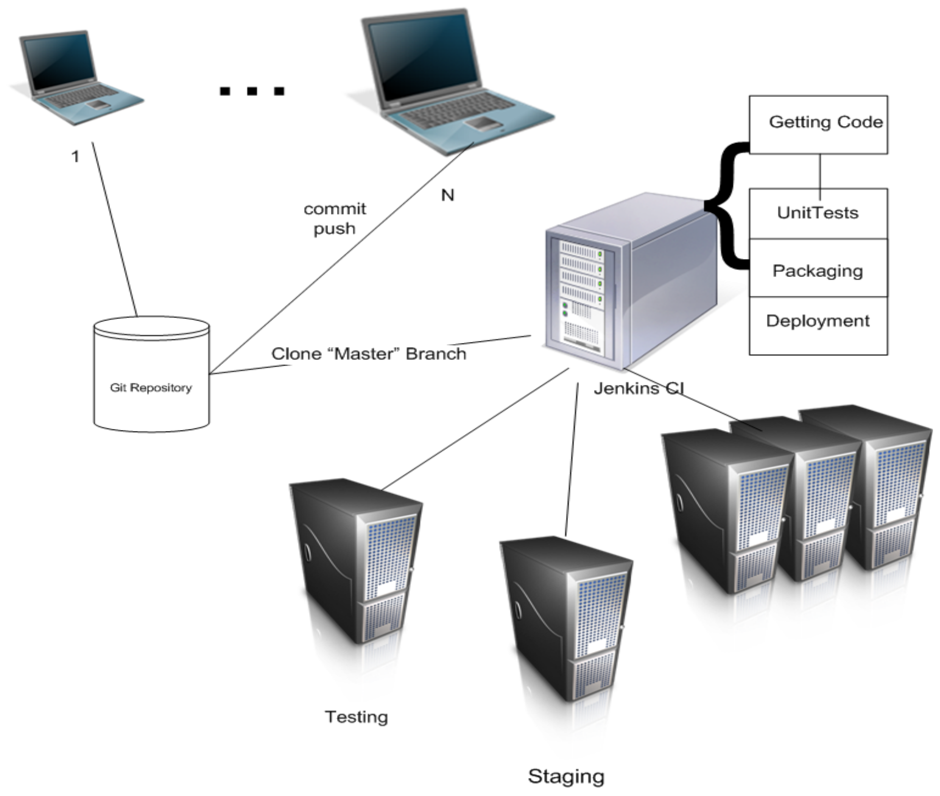
"Continuous Delivery is the natural extension of Continuous Integration: an approach in which teams ensure that **every change** to the system **is releasable**, and that we can release any version **at the push of a button**. Continuous Delivery aims to make releases boring, so we can deliver frequently and **get fast feedback** on what **users care about**."[2]

Below is a sequence diagram explaining the suggested flow of operations needed to implement continuous delivery



[1] http://www.itweb.co.za/index.php?option=com_content&view=article&id=67757:CA-global-IT-study-reveals-DevOps-driving-20-faster-time-to-market-for-new-services&catid=70
[2] http://www.thoughtworks.com/de/continuous-delivery

And if we turn this into images and terms that are coming from the developers and operations worlds we end up with the following diagram:



The diagram above shows the steps that are needed for implementing Continuous Delivery (CD). Let's explain it a bit. During the development one or more developers are working on their local machines and once they are ready with their changes they commit them to the central source code repository. In the diagram above this is a Git repository[3]. When the time comes for new release there is one Continuous Integration and Delivery machine that takes care to get the source code from the repository, run unit tests to prove the quality, package the files into distributable units and finally deploy the package on a testing or staging server for thorough testing. Once the tests are successful this same package is deployed on the production servers. And everything should be deployed on all servers with a click of a button!

What this article will show you is how to implement such a system using zs-client, Jenkins and Zend Server.

Installation

The instructions below for simplicity are targeting Debian/GNU based systems, but it should be easy for an IT person to apply these steps also to other operating systems.

[3] http://en.wikipedia.org/wiki/Git_(software)

1..n, n>=1 Zend Servers

Zend Server[4] is a complete, enterprise-ready Web Application Server for deploying, running and managing PHP applications with a high level of reliability, performance and security both on-premise and in the cloud.

In order to install Zend Server you need to execute the following commands:

```
wget http://repos.zend.com/zend.key -O- |apt-key add -
sudo sh -c \
                'echo "deb http://repos.zend.com/zend-server/6.1/deb_ssl1.0
server non-free" > \
               /etc/apt/sources.list.d/jenkins.list'
sudo apt-get update
sudo apt-get install zend-server-php-5.3 # if you want to install PHP 5.3 or
sudo apt-get install zend-server-php-5.4 # if you want to install PHP 5.4
```

For detailed instructions on installing Zend Server 6.x, see: http://files.zend.com/help/Zend-Server-6/zend-server.htm#installation_guide.htm

Important!

Before continuing, you will need to set your timezone by editing the 'date.timezone' PHP directive. You can do this using the Zend Server UI, on the Configurations | PHP page.

1 CI/CD server with Jenkins

For the continuous integration and delivery machine we will have to install the following software.

PHP Stack

PHP

```
apt-get install php5-cli
```

Pear

```
apt-get install php-pear
```

PHPUnit

PHPUnit[5] is a PHP Unit Testing framework. We will use it to run our phpunit tests and keep the high quality of our software.

```
pear install pear.phpunit.de/PHPUnit
pear config-set auto_discover 1
pear install pear.phpunit.de/PHPUnit
```

Phing

Phing[6] is a project build system, similar to Ant, but written in PHP.

```
pear channel-discover pear.phing.info
pear install phing/phing
```

[4] http://www.zend.com/en/products/server/
[5] http://phpunit.de/manual/current/en/index.html
[6] http://www.phing.info/

Jenkins

Jenkins[7] is an application that monitors executions of repeated jobs, such as building a software project or jobs run by cron. We will use Jenkins to help us build our project, test it and deploy it continuously.

```
wget -q -O - http://pkg.jenkins-ci.org/debian/jenkins-ci.org.key | \
             sudo apt-key add -
sudo sh -c 'echo deb http://pkg.jenkins-ci.org/debian binary/ > \
           /etc/apt/sources.list.d/jenkins.list'
sudo apt-get update
sudo apt-get install jenkins
```

Zs-Client

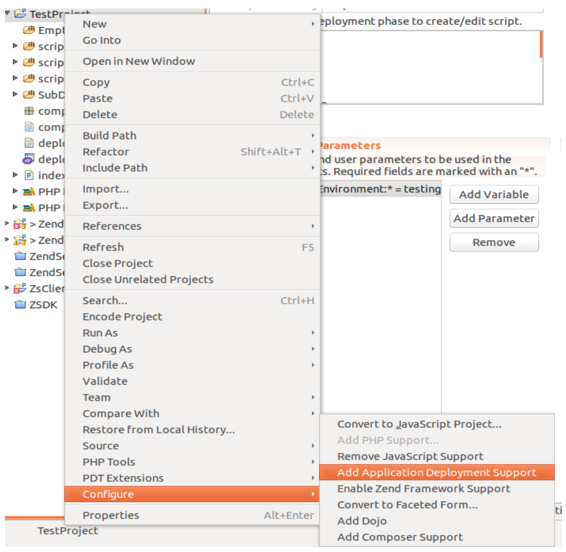
Zs-Client is a console line client that allows you remote to control Zend Server and execute commands on it without the need to go to the admin web interface or to have SSH access to the machine where Zend Server is installed.

```
cd /usr/local/bin
sudo wget https://github.com/zend-patterns/ZendServerSDK/raw/master/bin/zs-
client.phar
sudo chmod a+x zs-client.phar
```

Configuration

Add packaging support

In order to add packaging support to any existing PHP project you will have to add two files, named *deployment.xml* and *deployment.properties*, that describe how the source code should be packages and what should be executed before or after the installation of the source code on the server. These files are added once to the main directory of the project. If you have Zend Studio 9 or bigger you can enable directly deployment support. This is done by right clicking on the PHP project that you are working on, then choosing Configure and finally clicking on Add Application Deployment Support, as shown below:



[7] http://jenkins-ci.org/

If you don't have Zend Studio you can download zs-client and add deployment support to the PHP project.

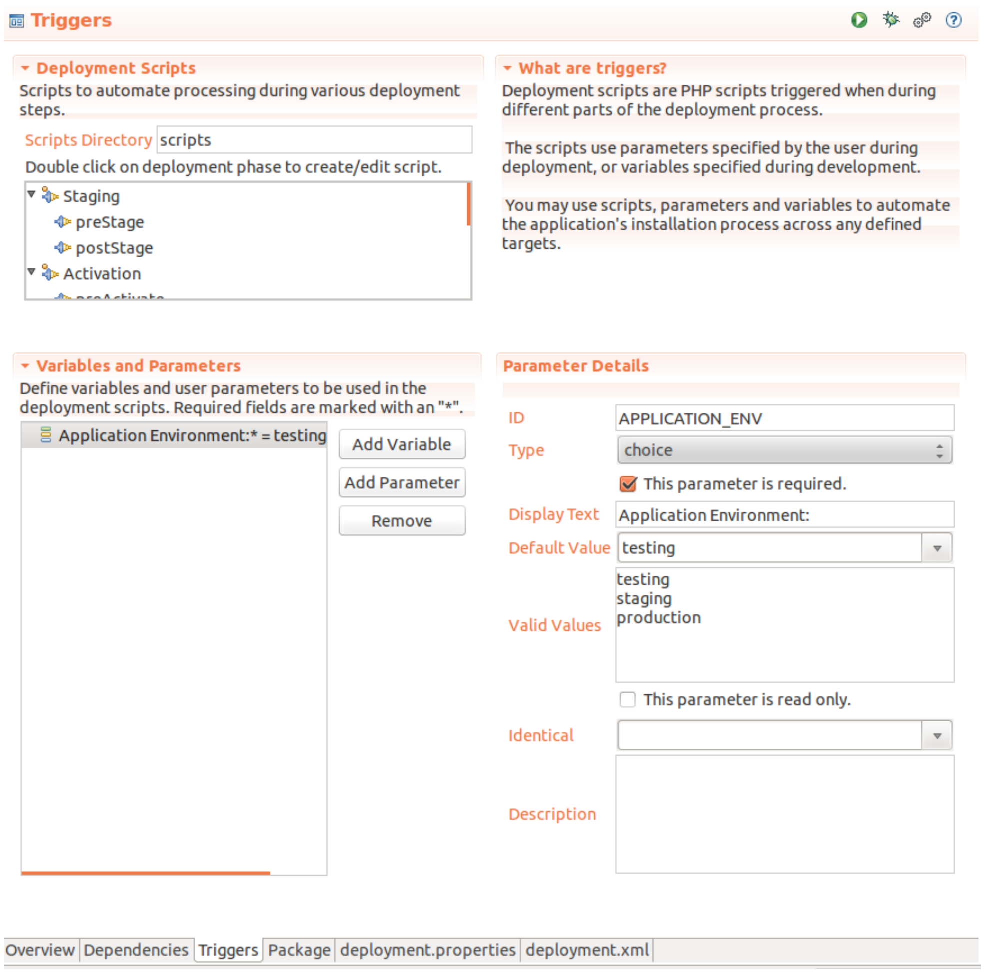
```
wget https://github.com/zendtech/ZendServerSDK/raw/master/bin/zs-client.phar
php zs-client.phar initZpk –folder=/<path-to-PHP-source-code>
```

The *deployment.properties* file contains information about the files that will be part of the package. The *appdir.includes* lists the files that are part of the application and *scriptsdir.includes* contains the files that will contain the special scripts to be executed before and after the installation.

The other file that is created is called *deployment.xml* and it contains information about the version of the application, parameters that should be passed when installing the package, required libraries, versions of PHP or PHP directives, etc.

You can find more information about the package structure and the format of the files from here: http://files.zend.com/help/Zend-Server-6/zend-server.htm#understanding_the_package_structure.htm and here: http://files.zend.com/help/Zend-Server-6/content/the_xml_descriptor_file.htm

In order to be able to deploy the same package on different application environments, like staging or production, we need to provide information during the deployment that specifies the targeted application environment. For that we need to add a new parameter of type choice to the deployment.xml as shown below:



Deployment scripts are PHP scripts triggered when during different parts of the deployment process.

The scripts use parameters specified by the user during deployment, or variables specified during development.

You may use scripts, parameters and variables to automate the application's installation process across any defined targets.

Scripts Directory: scripts

Application Environment:* = testing

ID	APPLICATION_ENV
Type	choice
	✓ This parameter is required.
Display Text	Application Environment:
Default Value	testing
Valid Values	testing staging production
	☐ This parameter is read only.
Identical	
Description	

If you don't have ZendStudio you can edit with any text editor the deployment.xml and add the needed lines. Take a look at the changes in the deployment XML related to this parameter (highlighted lines below):

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<package xmlns="http://www.zend.com/server/deployment-descriptor/1.0" version="1.0">
 <name>AddYourUniqueApplicationName</name>
 <version>
  <release>1.0.0</release>
 </version>
 <appdir>data</appdir>
 <scriptsdir>scripts</scriptsdir>
 <dependencies>
 </dependencies>
 <parameters>
  <parameter display="Application Environment:" id="APPLICATION_ENV" readonly="false"
required="true" type="choice">
    <validation>
      <enums>
        <enum>testing</enum>
        <enum>staging</enum>
        <enum>production</enum>
      </enums>
    </validation>
    <defaultvalue>testing</defaultvalue>
  </parameter>
 </parameters>
</package>
```

After that you can read the user provided information from any trigger (hook) scripts[8] that is in your package. For example before activating the package in order to set special connection settings you can do as in the example given below:

```
<?php
// code from pre_activete.php
$applicationEnv = getenv('ZS_APPLICATION_ENV');
$applicationBaseDir = getenv('ZS_APPLICATION_BASE_DIR');
$config = new Zend_Config($applicationBaseDir.'/config.ini', $applicationEnv);
```

Add Build Support

For our build we need to run different tasks that are depending on other tasks. What we need to make sure for a good quality package is to run the unit tests, after that be able to pack the application into distributable zpk file and finally deploy this file on the desired server and application environment. In order to achieve this we will add build.xml file in the root folder of our project. This build.xml will describe the steps that we need. You can directly copy the sample build.xml given below:

```
<?xml version="1.0"?>

<project name="ApplicationBuildName" basedir="." default="unittest">
```

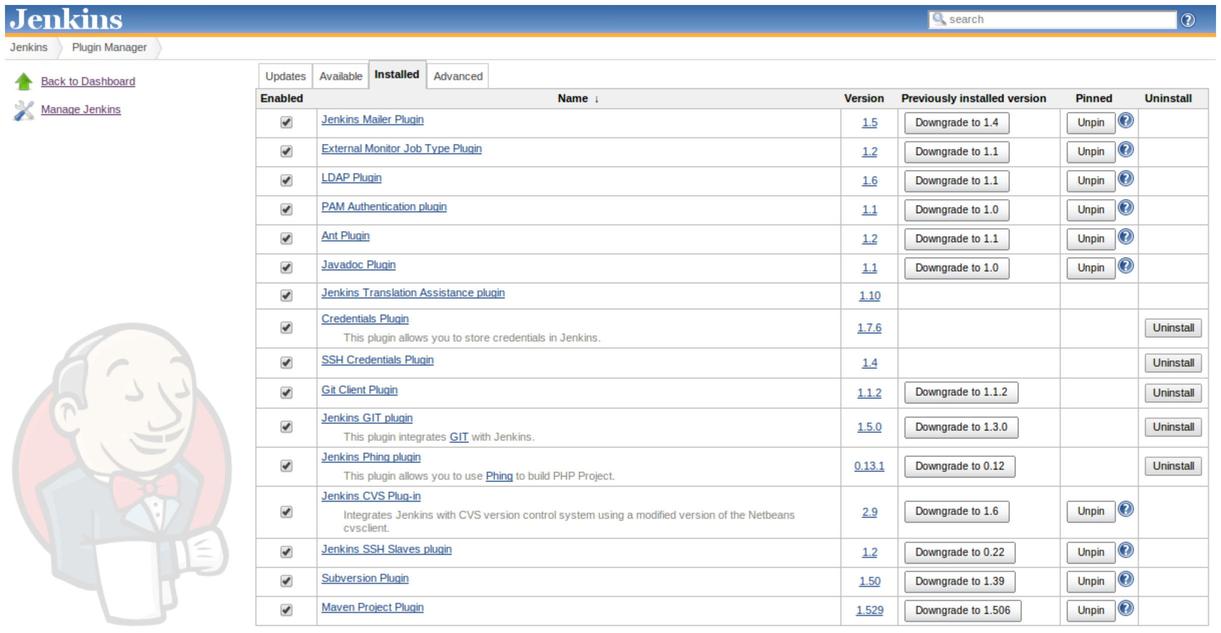
[8] http://files.zend.com/help/Zend-Server-6/content/understanding_the_package_structure.htm#Hook_Script_Constants

```
  <target name="unittest">
    <phpunit bootstrap="tests/bootstrap.php">
        <formatter type="summary" usefile="false" />
        <batchtest>
            <fileset dir="tests">
                <include name="**/*Test.php"/>
            </fileset>
        </batchtest>
    </phpunit>
  </target>

  <target name="zpk" depends="unittest">
    <exec command="php /usr/local/bin/zs-client.phar packZpk --folder='${project.basedir}' --
destination='${project.basedir}' --name='application.zpk'" checkreturn="true"/>
  </target>

  <target name="deploy" depends="zpk">
    <exec command="php /usr/local/bin/zs-client.phar installApp --
zpk='${project.basedir}/application.zpk' --zsurl='${host}' --zskey='${key}' --zssecret='${secret}' --
baseUri='${base}' --userAppName='${app}' --userParams='${params}'" checkreturn="true"/>
  </target>

</project>
```

This XML will be used from *phing* to execute the described steps on the Continuous Integration and Deployment server (CI/CD server). Also there are user parameters that can be passed to the build file. In the sample file above we have *${host}* that will be replaced with the *host* parameter given during the build process.

Jenkins

Once Jenkins is started go to its web interface and from there choose Manage Jenkins and then Manage Plugins. The list of available plugins contains Phing and Git plugins. Make sure that they are enabled. After you are done with the changes, check the list of installed plugins and see if Phing and Git are in the list, as shown below.



Jenkins | search

Jenkins › Plugin Manager

Back to Dashboard
Manage Jenkins

Updates | Available | Installed | Advanced

Enabled	Name ↓	Version	Previously installed version	Pinned	Uninstall
✓	Jenkins Mailer Plugin	1.5	Downgrade to 1.4	Unpin	
✓	External Monitor Job Type Plugin	1.2	Downgrade to 1.1	Unpin	
✓	LDAP Plugin	1.6	Downgrade to 1.1	Unpin	
✓	PAM Authentication plugin	1.1	Downgrade to 1.0	Unpin	
✓	Ant Plugin	1.2	Downgrade to 1.1	Unpin	
✓	Javadoc Plugin	1.1	Downgrade to 1.0	Unpin	
✓	Jenkins Translation Assistance plugin	1.10			
✓	Credentials Plugin This plugin allows you to store credentials in Jenkins.	1.7.6			Uninstall
✓	SSH Credentials Plugin	1.4			Uninstall
✓	Git Client Plugin	1.1.2	Downgrade to 1.1.2		Uninstall
✓	Jenkins GIT plugin This plugin integrates GIT with Jenkins.	1.5.0	Downgrade to 1.3.0		Uninstall
✓	Jenkins Phing plugin This plugin allows you to use Phing to build PHP Project.	0.13.1	Downgrade to 0.12		Uninstall
✓	Jenkins CVS Plug-in Integrates Jenkins with CVS version control system using a modified version of the Netbeans cvsclient.	2.9	Downgrade to 1.6	Unpin	
✓	Jenkins SSH Slaves plugin	1.2	Downgrade to 0.22	Unpin	
✓	Subversion Plugin	1.50	Downgrade to 1.39	Unpin	
✓	Maven Project Plugin	1.529	Downgrade to 1.506	Unpin	

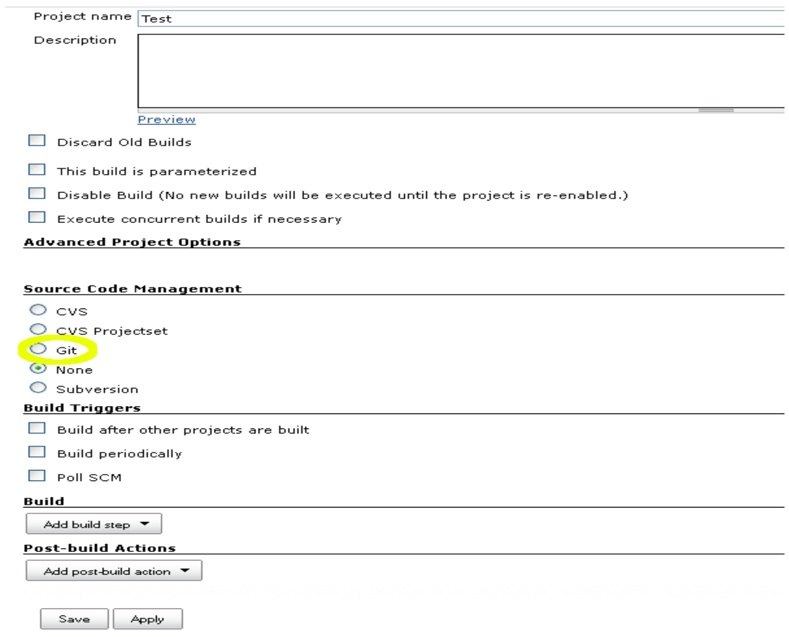
Usage

Create New Jenkins Job

Create a new Job from the Jenkins web interface. Go to *New Job* and in the form provide unique name for the new job, for example *StagingBuild*, and choose *"Build a free-style software project".*

Choose repository

Now edit the new job configuration. You should be able to specify what versioning system you want to use. Specify git and as show in the image below:



Project name: Test

Description:

Preview

☐ Discard Old Builds

☐ This build is parameterized

☐ Disable Build (No new builds will be executed until the project is re-enabled.)

☐ Execute concurrent builds if necessary

Advanced Project Options

Source Code Management

○ CVS
○ CVS Projectset
○ Git
● None
○ Subversion

Build Triggers

☐ Build after other projects are built

☐ Build periodically

☐ Poll SCM

Build

Add build step ▾

Post-build Actions

Add post-build action ▾

Save | Apply

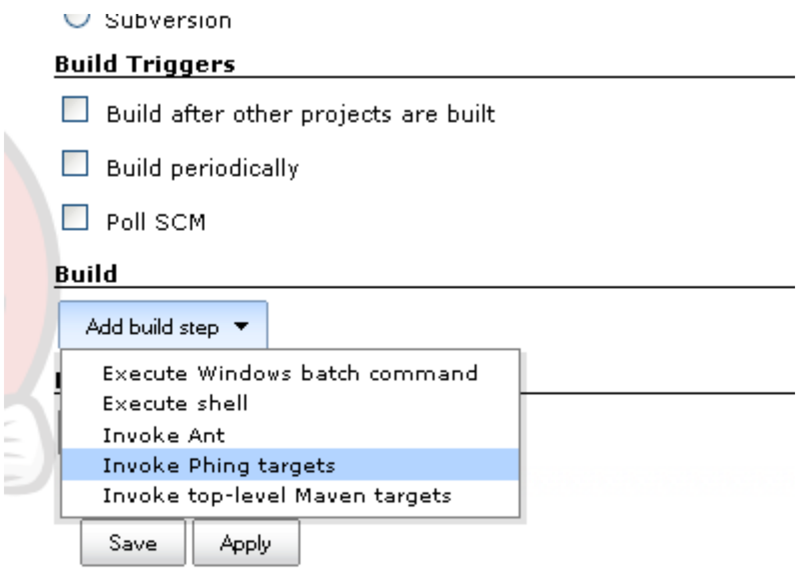
Then be sure to specify the *Repository URL, Branch Specifier, Config user.name Value* and *Config user.email Value.*

Example:
Repository URL : https://zendtechnologies@bitbucket.org/zendtechnologies/deploymentapp.git
Branch Specifier: master
Config user.name Value: Slavey Karadzhov
Config user.email: slavey@zend.com

For every new build Jenkins will try to set a tag to your repository to specify exactly which files were used to build the package. If you do not want this option you have to check the *Skip internal tag* option.

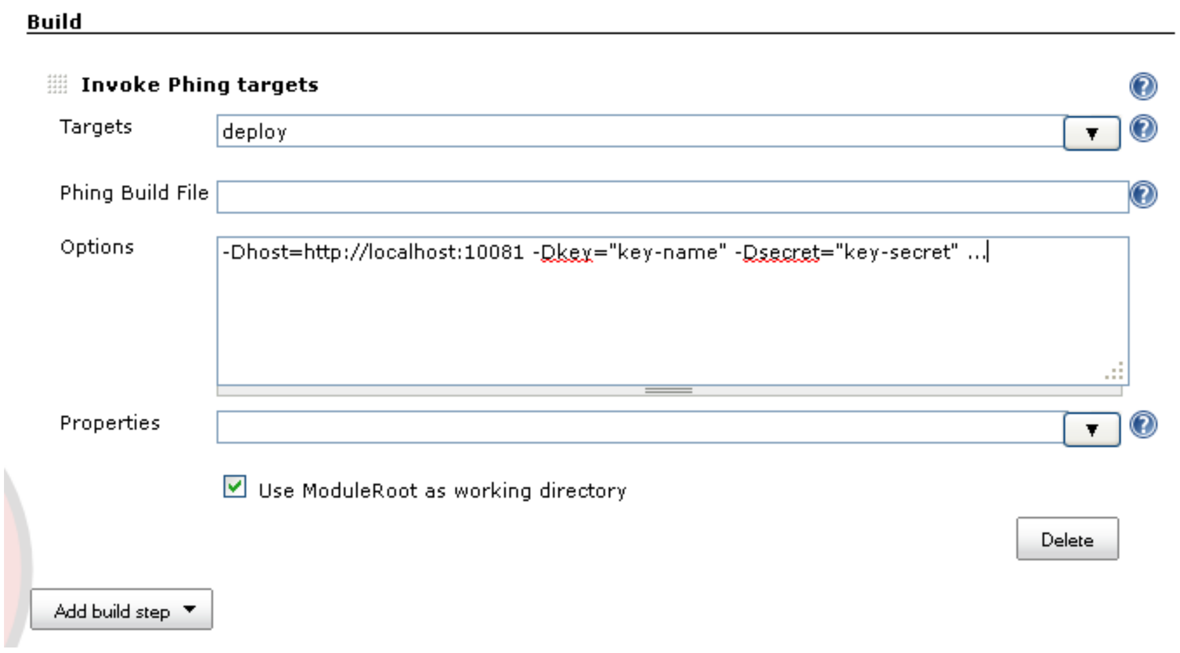
Pass parameters to the phing build

Next you have to choose what build system do you want to use. Select *"invoke Phing Targets"* as shown below.



If that option does not exist in your Jenkins web interface make sure that you have installed the Phing Jenkins Plugin.

After that make sure to specify the target name that you want to call. In our example build.xml we have *unittest*, *zpk* and *deploy* as targets. I will select *deploy* which will call zpk and unittest to be executed, before the actual deploy task. Now click on the *Advanced* button and you will be able to see text area where you can pass options. In the text area for every option that you want to type you have to add –D<name>=<value>. For example –Dhost=http://localhost:10081/ will pass a *host* parameter to the build.xml and the value will be *http://localhost:10081/.* If you need to add more options separate them with a space. See the example below.



```
-Dhost=http://localhost:10081 -Dkey="key-name" -Dsecret="key-secret" ...
```

☑ Use ModuleRoot as working directory

In our sample build.xml we expect the following parameters to be passed during the build process:

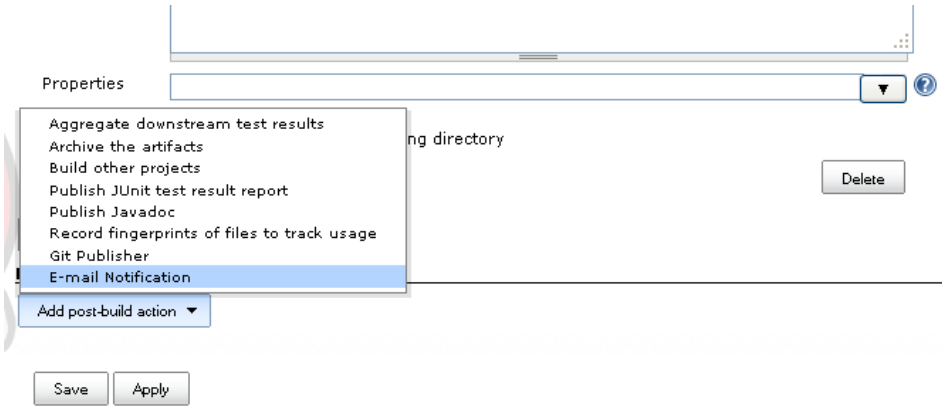
- *host* – pointing to an URL where the Zend Server web interface can be access
- *key* – valid Zend Server WebAPI name
- *secret* – valid Zend Server WebAPI secret for the provided name
- *base* – base URL where the application should be installed

In addition to these mandatory parameters there are also two optional parameters

- app – display name of the application, if different then the one specified in the deployment.xml
- params – the value should be a string formatted as valid HTTP query string that contains the custom parameters that we need to pass during our build. For our case we can add in the options text area the following string:
 -Dparams="APPLICATION_ENV=staging"

Email someone after successful built.

If we want to inform someone about a successful build then we can add post-build step. Choose from the available post-build actions "E-mail Notification" as shown below:



And in the *Recipients* fill the space separated email addresses of your testers release manager and your boss, if needed.

Build Now

When you are ready with the settings just click on the *Build now* button for your Jenkins project and you will be able to enjoy a completely automated delivery process that will deploy the package on one Zend Server. If this server is part of a cluster, then all the other Zend Servers in the cluster will have this application also being deployed to them. This way you can fast time to market with minimal efforts from your side.

Happy Continuous Delivery!