08

**Fall**



Continuous Integration with Jenkins

Training Labs

Table of Contents

Introduction 3

Lab 1: Command-Line Interface 4

Lab 2: Distributed Builds 6

Lab 3: CloudBees Jenkins Operations Center (CJOC)-Client Master and Shared Slave 10

Lab 4: Security 13

Lab 5: Access Control with RBAC Plugin 15

Lab 6: High Availability (HA)?? 21

Lab 7: Backup Scheduling 21

Lab 8: Monitoring 23

Lab 9: Nodes Plus 28

Lab 10: Fast Archiver 30

Lab 11: Pull-Request Builder for GitHub 33

Lab 12: Custom Update Centers for CloudBees Jenkins Operations Center 36

Lab 13: NIO SSH Slaves 40

Lab 14: Even Scheduler 41

Lab 15: CloudBees Jenkins Analytics ?? 42

Lab 16: Cluster Operations 43

Lab 17: CloudBees Support 47

# Introduction

This workbook is designed to supplement your CloudBees Jenkins training. It consists of a sequence of lab exercises that will introduce continuous integration concepts and Jenkins best practices.

These lab exercises should be completed in sequence and in the presence of an instructor. The instructor will distribute the software and code necessary to complete these exercises.

A dedicated lab installer will install all the software required for these labs. Your instructor will tell you if this has already been done on your machine, or if you need to do so yourself.

Most of the plugins required for this course are bundled with this installation, the Lab steps have additional information on which plugins are used and how to go about installing them if they were needed.

Copy/Pasting code& commands from this Word document is not recommended as it might introduce errors

If you encounter problems with your software installation or if you do not understand any of the instructions, please ask your instructor for help.

|  |
| --- |
| Lab 1: Command-Line Interface Goal  In this lab, you’ll get used to the basics of Jenkins CLI, so that it becomes one of the tools in your toolbox.  Step 1. Download jenkins-cli.jar  Go to http://localhost:8080/, click “Manage Jenkins”, and then further click “Jenkins CLI” to get to the Jenkins CLI web page (alternatively, you can directly go to this page by navigating tohttp://localhost:8080/cli)  Download jenkins-cli.jar by clicking the link.  Step 2. Run the help command  Open command prompt by running “cmd.exe” from “Run…” in the start menu.  Copy the downloaded jenkins-cli.jar into the current directory for the convenience (tip: if you drag a downloaded file into the command prompt, it’ll paste as a full path name. This makes the copy easier.)  Run the following command and verify that it’s producing output (all in one line).  C:> java –jar jenkins-cli.jar -s http://localhost:8080/ help 2>&1 | more  Set Jenkins URL as the ‘JENKINS\_URL’ environment variable so that you don’t have to keep retyping long URL:  C:> set JENKINS\_URL=http://localhost:8080/ C:> java –jar jenkins-cli.jar help  It should produce the same output.  Step 3. Run a build  Let’s execute the “gameoflife-freestyle” project we created earlier from command line. The -s option makes the command wait for the completion of the build.  C:> java –jar jenkins-cli.jar build –s gameoflife-freestyle  For extra credit, try the command again, but this time manually abort a build while it’s going on. CLI will report a failure. Try also running with the -c option, and verify that the build is skipped.  Step 4. Create a job  Let’s programmatically create a job. Creating a job from CLI requires you to define a job in the XML format. Writing this from scratch is too hard, so we’ll do cut&paste.  Navigate to http://localhost:8080/job/gameoflife-freestyle/config.xml in the browser. You should see the XML definition of this job. From a context menu, select “view source” to reveal the raw XML, then copy&paste it into notepad.    While you are at Notepad, try making some changes to this XML. A very easy change is to add something inside the description element. Jenkins tolerates errors in XML well, so long as it’s well-formed, so try making other changes if you can. Make sure you don’t have whitespace in front of the <?xml …?> declaration, for that breaks well-formedness.  Save this document to somewhere (say “c:\job.xml”) and remember its location.  Go back to the command prompt, then execute the following command (all in one line).  C:> java –jar jenkins-cli.jar create-job gameoflife-copy < c:\job.xml  From the browser, verify that you have the new job named ‘gameoflife-copy’. |
| Lab 2: Distributed Builds Goal  The aim of this lab is to set up a Jenkins slave and run distributed builds on both the master server and the slave. We will be running the slave and the master server on the same machine, so your lab computer will be both the master and the slave.  Step 1. Setting up a slave  Go to the “Manage Jenkins → Manage Nodes” screen and add a single slave by clicking on the ‘New Node” button.  Macintosh HD:Users:valentinaarmenise:Desktop:Screen Shot 2014-03-11 at 14.28.31.png  Configure this slave as shown here:  Macintosh HD:Users:valentinaarmenise:Desktop:Screen Shot 2014-03-12 at 11.27.33.png  You should now see this slave in the Build Queue:  Macintosh HD:Users:valentinaarmenise:Desktop:Screen Shot 2014-03-11 at 14.49.16.png  Step 2. Launching a slave  Click on the slave in the Build Queue.  Macintosh HD:Users:valentinaarmenise:Desktop:Screen Shot 2014-03-11 at 15.13.11.png  For this to work, you need to have installed the Java JRE. If this is not the case, do so now using the installer provided.  Click on the “Launch icon” to start the slave agent. Step through the dialogs until the Jenkins slave agent console appears.    Step 3. Performing builds  Your Jenkins slave agent is now running. To test the distributed build process, go to the Jenkins home page and build all three build jobs in quick succession. At least one build should be delegated to the slave agent.  Macintosh HD:Users:valentinaarmenise:Desktop:Screen Shot 2014-03-12 at 15.36.17.png |
| Lab 3: CloudBees Jenkins Operations Center (CJOC)-Client Master and Shared Slave Goal  This lab will cover how to create client masters and shared slaves in the CloudBees Jenkins Operations Center product.  Pre steps  Please install and log into the CloudBees Jenkins Operations Center product. You will also need to have a network-visible machine that can be used as a slave.  Step 1. Creating client masters  Client masters are Jenkins masters which are being managed by CloudBees Jenkins Operations Center. For this management to take place, the administrator must create an object representing that Jenkins master in the CloudBees Jenkins Operations Center dashboard.  To do this, click on the “New Item” link in the left-hand menu, much like you would to create a new Jenkins job. You will now create a master, which we will name “master-1” as a “Client Master” object.  Macintosh HD:Users:tracykennedy:Desktop:Screen Shot 2015-06-05 at 11.36.21 AM.png  Once you’ve named it, you’ll have the option of configuring one or many users/emails as “master owners” who will be alerted if the client master goes offline. This parallels the “master owner” concept offered by the CloudBees Nodes Plus plugin.  You will also have the option to specify whether this client master has its own CloudBees Jenkins Enterprise license and there doesn’t need to be issued one (“No license”) or whether a license should be issued with caveats (“License, no dedicated”, “License, fixed dedicated”, “License, floating dedicated”).  For this exercise, select the “No License” option.  You will also have the ability to configure whether a client master should have a limited plugin selection using the Custom Update Center plugin. For this lab, leave this option unchecked and click “Save”.  You’ll now be prompted to specify the “Client Master URL”, which is the URL for your CloudBees Jenkins Enterprise master.  Macintosh HD:Users:tracykennedy:Desktop:Screen Shot 2015-06-05 at 11.42.15 AM.png  Once this information is copy/pasted to the field, click on the “Push Configuration” button. You will be logged into your CloudBees Jenkins Enterprise master, then prompted to confirm that you would like this master to be managed by CloudBees Jenkins Operations Center. Click “Yes” to confirm.  You will now be logged into the CloudBees Jenkins Enterprise master, but you will see a new breadcrumb in your navigation bar: Macintosh HD:Users:tracykennedy:Desktop:Screen Shot 2015-06-05 at 11.45.37 AM.png  Where “Jenkins” now represents the root level or top page of the CloudBees Jenkins Operations Center master and “master-1” represents the same in your CloudBees Jenkins Enterprise master.  Step 2. Creating shared slaves  Log back into the CloudBees Jenkins Operations Center master by clicking on the “Jenkins” breadcrumb. We will now create a shared slave object, which is a dashboard-level object like the client master we just created.  Once again, click on the “New Item” link in the left-hand menu, but select the “Shared Slave” object type and name it “shared-slave-1”. Now click “okay”.  Macintosh HD:Users:tracykennedy:Desktop:Screen Shot 2015-06-05 at 11.52.47 AM.png  You will now be brought to the shared slave configuration screen, where you will specify slave-specific settings like number of execution threads for the machine, the remote file system root for the machine, any labels that described the machine, and the credentials needed to connect to the machine.  For the purposes of this lab, specify that the slave have 3 executors, that the remote FS root is “/”, and connect over SSH to your slave machine. Now “Save” your settings.  Macintosh HD:Users:tracykennedy:Desktop:Screen Shot 2015-06-05 at 12.14.08 PM.png  Once saved, CloudBees Jenkins Operations Center will attempt to connect to the slave and launch a slave agent on the specified machine. If the connection is successfully made, you will be redirected to the top level page of CloudBees Jenkins Operations Center and you will have both a “master-1” client master and “shared-slave-1” object.  Macintosh HD:Users:tracykennedy:Desktop:Screen Shot 2015-06-05 at 12.17.30 PM.png Lab 4: Security Goal  This lab will cover how to enable security for jenkin.  Step 1. Enable security  Macintosh HD:Users:tracykennedy:Desktop:Screen Shot 2015-06-05 at 12.38.14 PM.png  We can enable security using the following options, sample screenshot for enable LDAP uthentication  Active Directory  Delegate to servlet container  Google Apps SSO (with OpenID)  Jenkins’ own user database  LDAP  OpenID SSO  Operations Center Single Sign On  Unix user/group database  **Per-Project Access Control** If your Jenkins installation hosts sensitive projects that must be visible to a restricted set of people, define permissions at the individual project level so that different people have access to different sets of projects. Or if you just want to have a single set of access rights across all your projects, Jenkins can be configured that way, too.  **LDAP Integration** Jenkins can use your existing LDAP server to authenticate and authorize users. For most typical LDAP schemas, you need no more configuration information than the LDAP server name and, even if your schema is unique, you can still integrate by specifying a few additional options.  Handling LDAP integration this way, your users need not have multiple passwords among different applications in your network.    **Zero Configuration Active Directory Integration** Jenkins comes with built-in Active Directory support. If you run ICHI on a Windows machine that participates in a domain, the configuration is truly zero-conf. You just tell Jenkins to use Active Directory and it’ll figure out everything else on its own! If you run Jenkins on a Unix machine, just tell it the Active Directory domain name and, again, Jenkins will figure everything out.  Active Directory support works correctly with a domain with forests, too.  **SSO Integration: Atlassian Crowd, CAS, SourceForge Enterprise Edition, CollabNet TeamForge** Take advantage of the plugins that the community has developed to integrate Jenkins to SSO software like Atlassian Crowd or CAS, or delegate authentication to existing software like CollabNet TeamForge so that the user maintains a single identity across your network.  This part of Jenkins is extensible, so a custom plugin would enable you to integrate to any other SSO software or other services.  **Wiki Syntax for Descriptions** In places such as project descriptions and build descriptions, you can use a common Wiki syntax (such as Conflience, TWiki, etc.) instead of letting users write raw HTML. In security-sensitive environments, this prevents users from mounting XSS attacks |
|  |
|  |
|  |
| Lab 5: Access Control with RBAC Plugin Goal  This lab will cover how to use the CloudBees Role-Based Access Control plugin with CloudBees Jenkins Operations Center.  Step 1. Configuring client master and centralized security  The first step is to enable security on the CloudBees Jenkins Operations Center master, so click on the “Manage Jenkins” link in the left-hand menu and then on the “Configure Global Security” main menu option.  Macintosh HD:Users:tracykennedy:Desktop:Screen Shot 2015-06-05 at 12.38.14 PM.png  First click on the “Enable Security” checkbox, then select the “Mock Security Realm” as the security realm and “Role-based Matrix Authorization Strategy” as the authorization policy.  For the Mock Security Realm authentication option, we will create the following users with the following text:  harry admin-ext  sally developer-ext  barry  Where the first column represents the username and the right column represents the group name, and both represent imported fields from an external authentication server like LDAP or Active Directory. Doing this will create 3 users with those usernames in Jenkins and Jenkins will recognize the admin and developer external groups and the users’ membership in them.  Now scroll down and enable client master security enforcement by picking the “Single Sign-On (security realm and authorization strategy)” option from the drop-down menu. This option will allow the 3 mock “imported” users to sign in to the managed client master with their same logins, and will enforce your configured security settings on the client master.  Macintosh HD:Users:tracykennedy:Desktop:Screen Shot 2015-06-05 at 12.46.38 PM.png  Now save your security settings by clicking the “Save” button and return to the top-level of Jenkins Operations Center. You will now see a “log in” link in the top right-hand corner, so click on that and log in as “harry” with the password “harry”.  Step 2. Creating roles  Now we will need to create roles for our mock imported users. From the top-level screen, click on the “Roles” link in the left-hand menu. This link is now available because you selected the “Role-based Matrix Authorization Strategy” option in the last section.  Now click on the “Manage” link in the left-hand menu and you will be taken to the “Manage Roles” screen.  Our goal here is to create a security set up that has the following characteristics:   * Anonymous users have no access whatsoever, and they need to login first before even seeing the CloudBees Jenkins Operations Center top page. * Once logged in, users have generic read access. * Jobs can assign some users to the “reader” role that has read-only access to jobs, their test results, build results, etc. * Jobs can assign some users to the “developer” role, who can start a new build, configure jobs, and so on. * A few people will be in the “admin” role, which gets irrevocable full access to the whole of Jenkins.   This maps **to two pre-defined system roles** (“anonymous” and “authenticated”) and **three additional roles we create** (reader, developer and admin).  To create the additional roles, type in the titles to the “Role to add” field on the “Manage Roles” page and then hit the add button:  JEBC Screenshots:addingroles.PNG  By default, “filterable” will be set to true for them all. As such, you’ll now need to edit each role’s permissions\*\*. Note that the first permissions column shows the name of the ***section*** the permission is for, while the second column shows the name of the ***checkbox*** in that section:   |  |  |  |  | | --- | --- | --- | --- | | **Role** | **Permissions (section, checkbox)** | | **Filterable** | | Anonymous | None | None | Yes | | Authenticated | Overall | Read | Yes | | Reader | Overall  Job | Read  Read | Yes | | Developer | Job  Job  Job  Job  Job | Read  Create  Configure  Build  Workspace | Yes | | Administer | Overall | Administer | No |   Once you are ready to edit the “authenticated role”, note that you’ll be able to uncheck all boxes at once by scrolling to the right and clicking on the JEBC Screenshots:unchck.PNGbutton.  JEBC Screenshots:rbacsettings.PNG  Now try to save your configuration.  Note that the “Anti-lockout” guard will kick in and change our configuration, giving “authenticated” overall Administer permissions:  JEBC Screenshots:authenticatedadmin.PNG  This is because the “admin” role is not yet associated with a group that contains at least one member. As such, saving the configuration as-is would have locked you out of the instance.  Now let’s save these altered roles by clicking the “save” button at the bottom of the screen.  Step 3. Creating groups  The next step is to create groups that connect users to the roles, and populate these groups.  Click on the “Groups” link in the left-hand menu, which will take you to the “Groups” list. We will create 3 groups here: “Developers”, “Administrators”, and “Browsers”.  Click on the “add some?” link and start creating these three groups by populating the following . We want to create groups to represent each role we have created, and we also want to add both our imported users *and* our imported groups as members of each group. The final configuration should look like this:   |  |  |  |  | | --- | --- | --- | --- | | **Group Name** | **Roles** | **Propagated** | **Members** | | Browsers | reader | Yes | barry, harry, sally | | Developers | developer | Yes | sally, developer-ext | | Administrators | administer | Yes | harry, admin-ext |   For example, “Developer” is configured like this:  Macintosh HD:Users:tracykennedy:Desktop:Screen Shot 2015-06-05 at 1.02.19 PM.png  Repeat this process until all 3 groups have been created by hitting the “Back to groups” link in the left-hand menu and then hitting the “New Group” link in the same menu while on the “Groups” page.  Macintosh HD:Users:tracykennedy:Desktop:Screen Shot 2015-06-05 at 1.07.37 PM.png  Step 4. Completing configuration of the roles  Now that we have a group with the Overall/Administer permission, we can complete the configuration of the roles.  Go back to the “Manage Roles” screen (“Manage Jenkins” -> “Manage Roles”).De-select the Overall/Admin permission for the “authenticated” role and ensure that the Overall/Read permission is still on selected.  Now save the configuration. Since the “administer” role is now associated with a populated group, you should not receive an “Anti-lockout” warning.  Step 5. Single sign-on  Now that security settings have been created and enabled, try logging into the client master Jenkins instance as different users and clicking through breadcrumbs back to the CloudBees Jenkins Operations Center master. You will see that one sign on to either master will grant you access to both with no re-sign in required. |
| Lab 6: High Availability (HA)?? |
| Lab 7: Backup Scheduling |
| Goal  This lab will cover how to use Backup jenkin System configuration, job, Build records, etc.  Step 1 : Create Backup Job  This plugin lets you create backup jobs as jobs on CloudBees Jenkins Enterprise. In this way, you can use the familiar interface for scheduling execution of backup, and monitor any failure in backup activities.  To create a backup, click *New Job* from the left and select *Backup Jenkins*. This takes you to the page where you configure the backup project. Configuration of a backup job is very similar to that of a freestyle project. In particular, you can use arbitrary build triggers to schedule backup. Once the trigger is configured, click *Add build step* and add *Take backup* builder.    Step 2 : Configuring Backup  There are three aspects to the configuration of the *Take backup* builder.  First you should decide "what to backup": the data included/excluded from the backup. This plugin divides Jenkins storage into three classifications:  Step 3 : Restoring Backup  Backup files are in the tar+gzip format, with file names relative to the Jenkins home directory. Example of restoring a backup using Linux (GNU tar) syntax:  $ cd $JENKINS\_HOME  $ tar xvfz /backups/jenkins/backup-daily-13.tar.gz  config.xml  jobs/myjob/config.xml  $JENKINS\_HOME is relocatable, meaning you don’t have to restore the backup into the same instance it was created from. This allows you to exercise the backup restore steps by creating a different Jenkins instance on your own computer and restoring things there. Lab 8: Monitoring Goal  This lab will cover how to monitoring your CloudBees Jenkins Enterprise instance with the CloudBees Monitoring plugin.  Step 1: Using the built-in dashboard  The CloudBees Monitoring plugin comes with a default view that will provide details on System Load, JVM HEAP usage, etc. This view, even is not customizable, is a great help to see what it is possible to achieve with the plugin.  The first step is to create a view in the $JENKINS\_URL/newView screen using the Jenkins Enterprise Metrics Dashboard and name it Default Metrics: |
| Once the view is create, the configuration page doesn't offer other modifications than the view name and the view description.  As said, the view offer multiple graphs to have details on the instance health and usage:  Step 2: Recreate the default view  Using a Custom Metrics View, it is possible to re-create the previous view. To do that create a new view on $JENKINS\_URL/newView, name it Custom Metrics:  Change the setting to use 4 columns and then add one by one the graphs composing the Default Metrics view.  The main advantage of this new **Custom Metrics** is that we can make the graphs or smaller by changing the *height* and the number of columns used.  Step 3: Create the dashboard you need  Beside of the pre-defined graphs, it is possible to build your own graphs.  We will create a new dashboard which will consist of a single row composed of:  a graph showing the total number of threads in the JVM as well as a breakdown by thread state  a graph showing build scheduling rate as per the standard metrics dashboard  First, you need to create a new view, as previously, named it Complex Metrix. On the configuration page, select 2 columns and set the height to Medium    On the *Add graph* button, select the **Simple graph**:    Now, we add a new Basic series using the path vm.count and the label Total threads. It will provide us the total count of threads currently running. We need this to be able to  compare the following Aggregate series that will capture each type of threads running. To do that, we will use the regexp  vm\.([a-z]+)\.count  to be able to use the thread description in the label as  $1 threads  We will end with the configuration    Those series compose the first graph of the dashboard. The second one is the built-in Build Scheduling Rate, which we add using the Add graph button.  Once selected, we can save the dashboard and see the following result:  Extra  We have created all the views on the Jenkins root level, but the views can be created in a folder. You may need this with RBAC plugin to restrict who can access and see those metrics in your organization. Lab 9: Nodes Plus Goal  This lab will cover how to use plus plugin.  Introduction  The Nodes Plus plugin provides additional functionality for managing Jenkins build nodes, including:  The ability to assign owners to nodes who will be notified when the node availability changes.  Step 1: Node owners  In a Jenkins instance that is shared by a large team (or a number of teams) there can often be a pattern whereby certain individuals are responsible for certain specific build nodes. If the build node goes off-line and fails to come back on-line then builds which are tied to that build node can end up stuck in the build queue. Most Jenkins users, eventually settle on filtering out the e-mail notification that is associated with a successful build (either by configuring the build only to email when the build fails, or by setting up rules in their mail client). Thus if a specific project’s builds are stuck in the build queue, nobody may notice until they actually browse the Jenkins instance in their web browser.  The *Node Owners* property introduced by the *Nodes Plus* plugin provides an e-mail notification to the designated owners based on a configurable set of availability triggers    Step 2 : Configuring Node owners  node owners property  **Send email when connected**  This trigger fires when the communication channel has been established with the node. If the node has been marked as Temporarily off-line then no build jobs will be accepted by the node. + NOTE: This trigger will only fire on transition from disconnected to connected.  **Send email when disconnected**  This trigger fires when the communication channel with the node has been confirmed dead. Where a node is configured to be kept on-line as much as possible Jenkins will immediately try to reconnect the node, and so in such cases the email from this trigger will be immediately followed by either the launch failure or successful connection email. As such this trigger is typically most useful where one of the other availability strategies has been selected for the node. + NOTE: This trigger will only fire on transition from connected to disconnected.  **Send email on launch failure**  This trigger fires when an attempt to establish a communication channel with the node fails. + NOTE: This trigger will fire each and every time a connection attempt fails.  **Send email when temporary off-line mark applied**  This trigger fires when the node is marked as temporarily off-line. + NOTE: This trigger will fire within the first 5 seconds of the node being marked off-line.  **Send email when temporary off-line mark removed**  This trigger fires when the node mark of being temporarily off-line has been removed from the node. + NOTE: This trigger will fire within the first 5 seconds of the node ceasing to be marked off-line.  Save the node configuration to apply the changes |
| Lab 10: Fast Archiver Goal  This lab will cover the use of Fast Archiver plugin  Introduction  The Fast Archiver plugin uses an rsync-inspired algorithm to transfer archives from slaves to a master. The result is that builds complete faster and network bandwidth usage is reduced.  Step 1 : Fast Archiver  The Fast Archiver plugin is automatically installed and enabled if you have downloaded CloudBees Jenkins Enterprise. This plugin takes over the built-in artifact archiving functionality in Jenkins. If you are already using that feature, no configuration change is needed to take advantages of this feature. This plugin also does not touch the persisted form of job configurations, so you can just uninstall this plugin and all your jobs will fall back to the plain vanilla artifact archiving    Note that fast archiving only occurs on builds run on a slave, not those run on the Jenkins master. Also there must have been previous builds which produced artifacts for the new artifacts to be compared to. Otherwise, Jenkins will perform a regular complete artifact transfer. It will do the same if the fast archiver detects that data is corrupted due to an internal bug  fast archiver console  Normally you need do nothing to configure fast archiving. As can be seen in [Figure 8.2, “Fast archiving configuration”](http://documentation.cloudbees.com/docs/cje-user-guide/fast-archiver-sect-creating.html#fast-archiver-config-img), this “artifact manager” is enabled by default when the plugin is first loaded. However you can remove this strategy and later read it in the main Jenkins configuration screen.  fast archiver config |
| Lab 11: Pull-Request Builder for GitHub Goal  This lab will cover how to Pull-Request Builder for GitHub.  Pre-steps  Login to CloudBees Jenkins Operations Center and also you need to have Github account and repository  Step 1.Generate token on Github   * Go to your GitHub profile page. * In the left sidebar, click Applications. * Click Generate new token. * Give your token a descriptive name * Select the scopes to grant to this token. Pull request tester plugin require permission to administer repository hooks and access repositories: repo, public\_repo, admin:repo\_hook, repo:status. * Click Generate token. * Copy the token to your clipboard. For security reasons, after you navigate off this page, no one will be able to see the token again     Step 2: Jenkin Global Configuration  global config  Step 3 : Using Pull-Request Builder for GitHub  Plugin introduce a new trigger option to Jenkins jobs, "Build pull requests to the repository". This option only makes sense if the job is building a project using a Github repository as SCM. Selecting this option will automatically register a web hook on Github so Jenkins get notified when pull-requests are created/updated.  Pull Request tester plugin relies on git validated merge, so this option has to be enabled as well, even if you don’t use this feature by yourself.  As a pull-request is opened on GitHub repository, Jenkins will trigger a build and report result in GitHub UI as "commit status" (a note associated to the latest commit of the pull-request). Any update to the pull request (new commit pushed, or amendment to existing commits in pull-request) will trigger another build and update the commit status. |
| Lab 12: Custom Update Centers for CloudBees Jenkins Operations Center Goal  This lab will cover how to utilize Custom Update Centers in a CloudBees Jenkins Operations Center context with client masters.  Pre-steps  Log into CloudBees Jenkins Operations Center and ensure that you have a client master configured, as well as an available internet connection.  Step 1. Creating an update center  The first step is to create a Custom Update Center, which is a dashboard-level object. Click on the “New Item” link in the left-hand menu and selected the “Update center” object type. Name the update center “cjoc-updates” and click “Ok” to continue.  You will now be sent to the configuration page for this new update center. Leave the default configuration for the plugin versioning strategy and certification signature provider. You will then need to add an upstream source to provide the update center with content, so click on the “Add” button in the Content section and select the “CloudBees Jenkins Update Center” option for this lab. Leave the stream and Jenkins version with their default settings. For the purposes of this lab, don’t configure a maintenance task and just save this initial configuration.  Now that this update center exists, we will need to populate it with plugins. Since this update center is empty, click on the “Pull everything” link in the left-hand menu.Macintosh HD:Users:tracykennedy:Desktop:Screen Shot 2015-06-05 at 1.36.03 PM.png  You will be asked to confirm the pull, so click “Yes”. CloudBees Jenkins Operations Center will then attempt to pull all plugins from the specified update center, in this case the CloudBees curated Enterprise update center.  Step 2. Connecting to client masters  While waiting for the plugin pull to finish, go back to the top-level page of CloudBees Jenkins Operations Center. Open up the configuration page for the client master by clicking on the black arrow to the right of the master’s name and selecting the “Configure” option:  Macintosh HD:Users:tracykennedy:Desktop:Screen Shot 2015-06-05 at 1.40.33 PM.png  Now check the “Specify update center for Master” option on the client master configuration page, then select the “Update Center hosted on this Jenkins instance” radio option. Since this is our first and only custom update center, the “cjoc-updates” update center should be selected by default. Now save your configuration change and you will be re-directed to the client master’s top page.  Click on the “Manage Jenkins” link in the left-hand menu and select the “Manage Plugins” item in the main menu.  Macintosh HD:Users:tracykennedy:Desktop:Screen Shot 2015-06-05 at 1.43.33 PM.png  Click on the “Available” tab in the plugin manager, and observe how there are now no plugins listed. This is because the custom update center is managing this client master’s list of available plugins, and since no plugins have been promoted, none are listed.  Step 3. Promotion  Return to the CloudBees Jenkins Operations Center top page and, if the plugin pulls are still in the build queue, wait for the plugins to finish downloading.  Now click on “cjoc-updates” and click on the “Plugins” tab. Since the pull has completed, you will see a list of all plugins pulled from the CloudBees Jenkins Enterprise update center.  Macintosh HD:Users:tracykennedy:Desktop:Screen Shot 2015-06-05 at 1.56.36 PM.png  To enable the client master to download plugins from its update center, we need to promote plugins from the list of plugins in the update center.   To do this, click on the name of any plugin and select either a specific version of the plugin to promote or the “latest” version. Many plugins are dependent on others, so if the plugin you have selected has a dependency you will see a warning about said dependency:  Macintosh HD:Users:tracykennedy:Desktop:Screen Shot 2015-06-05 at 1.58.58 PM.png  After selecting your desired version, click the “Change promoted version” button. This will promote the plugin to the downstream master’s update center and allow the master to download that plugin.  Repeat this process as desired for other plugins, then navigate to the client master and open its Plugin Manager as before. However, click on the “Advanced” tab instead of the “Available” and scroll to the bottom of the page. In the bottom right-hand corner is a “Check Now” button. Click on this button to force this Jenkins master to pull the latest plugin metadata from the upstream update center, in this case our “cjoc-updates” update center.  Now click on the “Available” tab, and you should see a list with only the plugins and versions you promoted earlier.  Macintosh HD:Users:tracykennedy:Desktop:Screen Shot 2015-06-05 at 2.05.31 PM.png |
| Lab 13: NIO SSH Slaves Goal  This lab will teach users the use of NIO SSH Slaves plugin.  Step 1 :  Enable the following plugin    Select the lunch method as below in jenkin which is installed on the linux mechine.  configure |
| Lab 14: Even Scheduler Goal  This lab will teach you how to activate even scheduler plugin  Step 1: Global preference  Enable the plugin    Go to Manage Jenkins  Select System Configuration  Select for Default Scheduler Preference the entry Prefer execution on idle nodes over previous used nodes    Step 2 : Per Job Preference   * + Picks the scheduling algorithm for a specific job, regardless of the global preference.   + Go to the configuration screen of a job   + Select Override the default scheduler preference   + Select Prefer execution on idle nodes over previous used nodes |
| Lab 15: CloudBees Jenkins Analytics ?? |
| Lab 16: Cluster Operations Goal  This lab will teach users the basics of using cluster operations with CloudBees Jenkins Operations Center to take back ups and install plugins on masters.  Pre-steps  You will need a CloudBees Jenkins Operations Center master that is pre-configured with a CloudBees Jenkins Enterprise client master and an update center called “cjoc-updates”. Log in as “harry”.  Step 1: Creating a cluster operation  A cluster operation is a new job type that is exclusive to CloudBees Jenkins Operations Center, so to begin you will need to log into CloudBees Jenkins Operations Center and click on the “New Item” button.  Name this cluster operation “cluster-operation” and select the “Cluster Operations” job type, then click “Ok”.  Macintosh HD:Users:tracykennedy:Desktop:Screen Shot 2015-06-05 at 2.48.12 PM.png  Step 2: Scheduling the operations  Like all Jenkins jobs, cluster operations can be triggered or scheduled. For the purposes of this lab, we will choose the “Build periodically” option to schedule when this operation should run. We will schedule this operation to run every 5 minutes:  5 \* \* \* \*  In a real-life use case, it would make more sense to either run the job based on a trigger or on a far less frequent basis, such as once a night for backups, once a week for plugin updates, and once every 3 months for Jenkins core updates.  Step 3: Selecting a master  Instead of having a “Build Steps” section like most jobs, cluster operations have an “Operations” section. Operations can either be performed against masters managed by CloudBees Jenkins Operations Center or an update center configured on the same.  To start, we will create an operation against our client master, so click on the “Add Operation” button and select the “Masters” option. This will cause a menu of items related to master operations to display. For this lab, we will run a backup operation on our only client master, so select “Using a specified update center” as the source criteria and the “cjoc-updates” update center should appear as the default choice.  Filters allow further targeting of specific masters, but in this case it is unnecessary so ignore that section for now.  Step 4: Creating a cluster operationbackup  Now we will configure what steps to run against our selected master. Click on the “Add Step” button under the “Steps” section and select the “Backup Master” option. The options for a cluster operation backup are the same as for a CloudBees Backup Plugin backup directly on a master. For the purposes of this lab, opt to only backup “job configurations” to a local directory outside of the $JENKINS\_HOME, like  C:\CloudBees\backups  And select the exponential decay policy.  Macintosh HD:Users:tracykennedy:Desktop:Screen Shot 2015-06-05 at 3.00.31 PM.png  Step 5: Selecting an update center  We will now add an operation to this job, but one that runs against an update center. To do this, scroll past the configured Master operations until you see the “Add Operation” button. Click the button and select “Update Centers” from the dropdown menu.  To target out one update center, it is easiest for this lab to select the “Using a specified update center” option again and taking the default “cjoc-updates” as our selection. Filters will again be unnecessary since this is the only available update center.  Step 6: Creating a plugin update cluster operation  We now want to force our custom update center to update on the schedule that we are triggering this job on. Click on the “Add Step” button and add the following steps:   * Refresh upstream sources * Pull new versions   Macintosh HD:Users:tracykennedy:Desktop:Screen Shot 2015-06-05 at 3.05.43 PM.png  This will allow our update center to refresh itself and always have the latest plugins from the CloudBees Enterprise update center already pulled.  This completes the actions needed to run against this update center, so scroll down again until you see the “Add Operation” and select the “Masters” cluster operation.  This time you will be forcing the master to upgrade its installed plugins. Select the following steps:   * Refresh update center metadata * Upgrade all plugins * Restart Now   This will cause the specified client master to get a list of the latest plugins from its upstream update center, upgrade all plugins to the newest version, and then restart the CloudBees Jenkins Enterprise master without consideration for any in-flight builds. |
| Lab 17: CloudBees Support |

Goal

This lab will teach users the basics of Cloudbees support plugin.

Pre-steps

The *Support* plugin provides the ability to generate a bundle of all the commonly requested information used by CloudBees when resolving support issues

Step 1: Creating a cluster operation

Since some CloudBees customers purchase support contracts without being licensed for CloudBees Jenkins Enterprise, this plugin is also available for download and installation into any Jenkins installation. Simply go to the plugin’s [release notes](http://release-notes.cloudbees.com/product/CloudBees+Support+Plugin), pick the latest version, and click the link in the Download section (such as to [http://jenkins-updates.cloudbees.com/download/plugins/cloudbees-support/\*latest\*/cloudbees-support.hpi](http://jenkins-updates.cloudbees.com/download/plugins/cloudbees-support/*latest*/cloudbees-support.hpi)). You can install the downloaded plugin using Manage Jenkins » Manage Plugins » Advanced » Upload Plugin

Step 2: Generate a bundle

To generate a bundle, simply go to the CloudBees Support link on the Jenkins instance.

