****

**MDW Cloud Foundry Cookbook – Table of Contents**

**Prerequisites:**

[**Eclipse with the MDW Plugin**](#eclipse)

Get set up for MDW cloud development.

**Running in the Cloud:**

[**1: Clone the Demo Project from Git**](#clone_git)

Clone mdwdemo from its Git repository as a starting point.

[**2: Build and Deploy to Cloud Foundry**](#deploy_cloud_foundry)

Use the Cloud Foundry Eclipse plugin or Command Line Interface to deploy.

**Running Locally:**

[**1. Tomcat Container**](#tomcat_container)

Set up Tomcat for local deployment.

[**2. Deploy Locally**](#deploy_locally)

Make changes and test them on the fly.

**MDW Cloud Foundry Cookbook – Prerequisites**

[Contents](#contents) [Next](#clone_git)

**Eclipse with the MDW Plugin**

**Set Up Eclipse on Your PC:**

* Make sure you have Java Runtime 7 or 8 installed on your computer:
  + [**http://www.oracle.com/technetwork/java/javase/downloads**](http://www.oracle.com/technetwork/java/javase/downloads)
* Install Eclipse Mars (4.5.x) for JavaEE Developers:
  + [**https://www.eclipse.org/downloads**](https://www.eclipse.org/downloads)
* Launch Eclipse. Here’s an example command line that includes appropriate JVM memory settings for running and debugging:

**C:\eclipse\_4.5.2\eclipse.exe -vm C:\jdk1.7.0\_45\bin\javaw.exe -vmargs -Xms512m -Xmx1024m**

* Install the latest version of the MDW Plug-In via Eclipse Software Updates (Help > Install New Software > Add > **http://lxdenvmtc143.dev.qintra.com:6101/MdwPlugin** > Install).
* Make sure Eclipse has a Java 7 or Java 8 Installed JRE available in your workspace (Window > Preferences > Java > Installed JREs).

**Optional Plug-Ins:**

* Cloud Foundry (Deploy and manage Java/Spring apps in Cloud Foundry)

**http://dist.springsource.com/release/TOOLS/cloudfoundry**

* Groovy (Groovy script syntax highlighting, auto-complete, etc.)

**http://dist.springsource.org/snapshot/GRECLIPSE/e4.5**

* BIRT (Business Intelligence and Reporting)

**http://download.eclipse.org/birt/update-site/4.5**

**MDW Cloud Foundry Cookbook – Running in the Cloud**

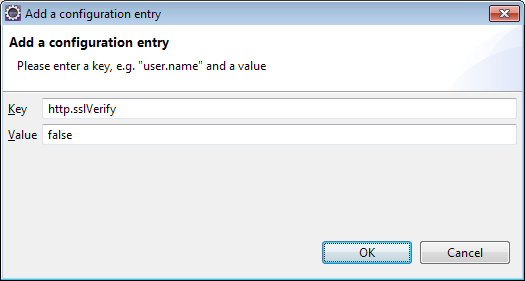
[Previous](#eclipse) [Contents](#contents) [Next](#deploy_cloud_foundry)

**1. Clone the Demo Project from Git**

A quick way to get familiar with the layout of an MDW workflow cloud project is to start with the mdwdemo project. This project is available from the MDW GitLab instance in the CenturyLink Labs cloud. You can clone this easily using Eclipse as described below. If you prefer to use command line Git, clone the repository from **https://8.22.8.164/mdw/mdwdemo.git** using credentials **mdw/ldap\_0123**.

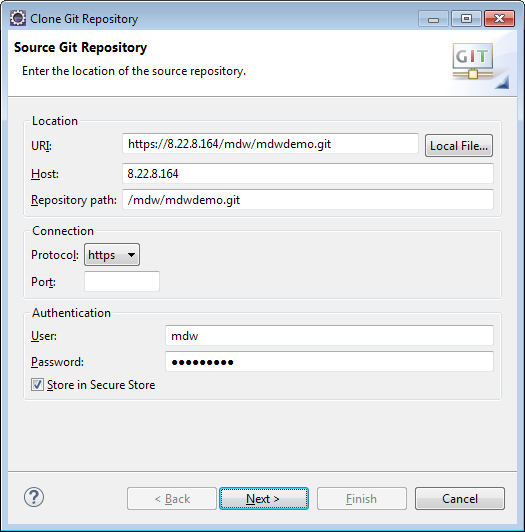
**Disable Git SSL Verification in Eclipse:**

* Our SSL certificate is self-signed and is not trusted by the Git client. Therefore, the first step is to disable (at least temporarily) SSL verification. From the Eclipse menu, select Window > Preferences > Team > Git > Configuration > Add Entry…
* Create an entry like this:

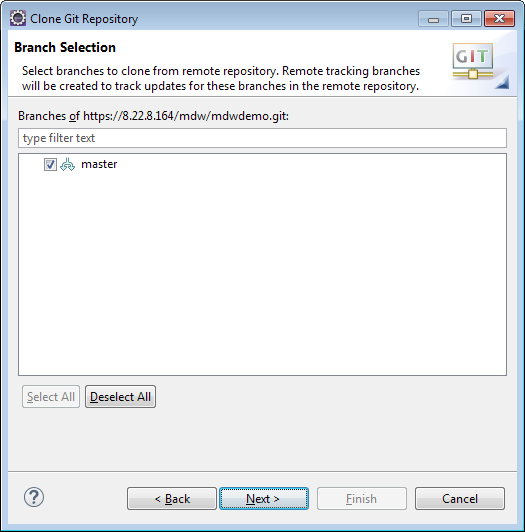


**Clone the mdwdemo Project into Your Workspace:**

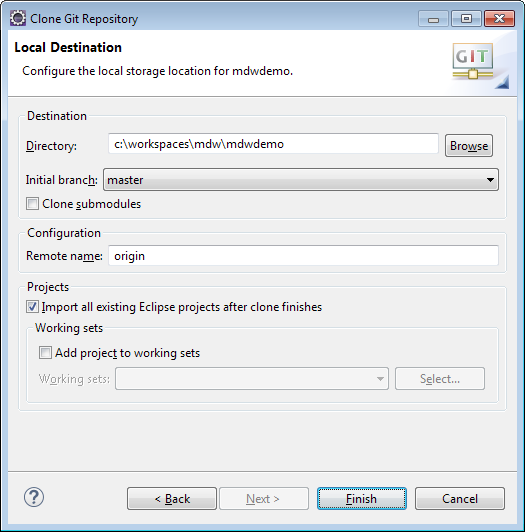
* Access the Git Repositories view by selecting Window > Show View > Other > Git > Git Repositories:
* Copy the repository URL into your clipboard: **https://8.22.8.164/mdw/mdwdemo.git**
* Right-click in Git Repositories view and select Paste Repository Path or URI.
* Enter **mdw** for the User and **ldap\_0123** for the Password:



* Click Next and make sure the **master** branch is selected:



* Click Next again. Point to your workspace/mdwdemo for the Destination Directory. Check "Import all existing Eclipse projects after clone finishes". Then click Finish.



**MDW Designer Perspective:**

* To best view mdwdemo's workflow assets, switch to Designer Perspective (Window > Perspective > Open Perspective > Other > MDW Designer).
* In Process Explorer view expand the mdwdemo project to see the included workflow packages. These contain the assets that you'll deploy to Cloud Foundry in the following steps.

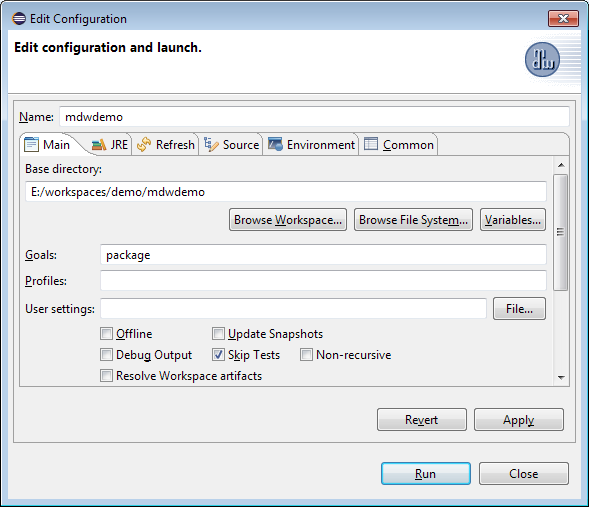
**MDW Cloud Foundry Cookbook – Running in the Cloud**

[Previous](#clone_git) [Contents](#contents) [Next](#tomcat_container)

**2. Build and Deploy to Cloud Foundry**

**Build mdwdemo:**

* Switch to Java perspective in Eclipse so that the pom.xml Maven build file is visible in Package Explorer.
* If you have the Eclipse Groovy plug-in installed, edit pom.xml to set the groovy.jar so that it points to the groovy-all jar file in your Eclipse installation. This is needed so the Cucumber test cases that are executed during the build will not cause exceptions due to a Groovy version mismatch.
* Right-click on the pom.xml file and select Run As > Maven Build. Enter "package" for the Goal. Also on the JRE tab make sure a Java 7 runtime is selected as a minimum.



**Getting a Cloud Foundry Space:**

* TODO: Describe the steps for getting set up with a Cloud Foundry space in the CenturyLink IT PCF dev environment.

**Ensure Permissions:**

* MDW uses a relational database for storing runtime values and user information like workgroups, roles and permissions. By default this is an embedded MariaDB instance. This arrangement is suitable for development purposes, and is the most convenient way to get started. With this option, during initial deployment the database will automatically be populated with the users designated in config/seed\_users.json. If you don't want to log in as the mdwapp default user, edit this file to include yourself with Process Designer and Execute permissions.
* Optionally, for longer-term persistence, you can point instead to any of the supported external databases identified in the MDW [Compatibility Matrix](http://cshare.ad.qintra.com/sites/MDW/Releases/Compatibility). With this option, you'll use the SQL scripts at the following URL to install the MDW db schema and reference data, and the initial user set is inserted via seed\_users.sql:
  + [**http://cshare.ad.qintra.com/sites/MDW/Developer%20Resources/MDW%20Database**](http://cshare.ad.qintra.com/sites/MDW/Developer%20Resources/MDW%20Database)

**Cloud Foundry Client Options:**

* This tutorial uses the Cloud Foundry Command Line Interface:
  + [**https://cli.run.pivotal.io/stable?release=windows64&source=github**](https://cli.run.pivotal.io/stable?release=windows64&source=github)

Pivotal, the company behind Cloud Foundry, publishes an Eclipse plugin that provides some of the same capabilities through the Eclipse WTP Servers view:

* + [**https://github.com/cloudfoundry/eclipse-integration-cloudfoundry**](https://github.com/cloudfoundry/eclipse-integration-cloudfoundry)

However, as noted in the readme markdown on their GitHub site, this plugin is transitioning over to the Eclipse Community:

* + [**https://projects.eclipse.org/projects/ecd.cft**](https://projects.eclipse.org/projects/ecd.cft)

The new Eclipse Tools for Cloud Foundry plugin is currently only available in the form of nightly builds, and is not compatible with the old Pivotal version. So until this situation stabilizes, the preferred approach for deploying MDW apps on Cloud Foundry is the CLI. Feel free to experiment with either of the Eclipse plugins, but in any case you'll want to install the CLI binary for access to certain capabilities.

**Perform a "cf push" to deploy mdwdemo:**

* Note: The following examples use the MDW demo deployment in [CenturyLink's AppFog](https://www.ctl.io/appfog/) cloud environment. See the [Cloud Foundry Dev Guide](http://docs.cloudfoundry.org/devguide/#cf), or type "cf help" for details about the available commands.
* A Cloud Foundry deployment is configured primarily through a manifest.yml file in the project root directory. Take a look at the manifest.yml in mdwdemo, and note that it stipulates the MDW buildpack through its Git repository URL. The settings you'll likely need to configure in this file are as follows:
  + MDW\_VERSION - Whichever version of the MDW framework you're using (must be available as a buildpack).
  + mdw.database.url - Only if you're using an external database rather than embedded.
  + mdw.hub.url - This is the user-access app endpoint URL as reported after a successful push.
  + mdw.services.url - The services endpoint. In development this is the same as mdw.hub.url.
* After you've installed the Cloud Foundry CLI, open a command-line window in the mdwdemo root directory. Use the "cf login" command, then enter your credentials to log in to your cloud space:

**cf login -a https://api.useast.appfog.ctl.io -o MDWF**

(The -a parameter designates the API endpoint, and the -o parameter is the organization -- type "cf login -h" for details).

* Now, to deploy the mdwdemo app simply type cf push:

**cf push**

* In another command-line session, you can tail the application logs by typing:

**cf logs mdwdemo**

* Once the push is complete, verify successful deployment by accessing MDWHub in your browser with a URL something like this:

**https://mdwdemo.useast.appfog.ctl.io/mdw**

**Make a Change and Push Again:**

* In Eclipse switch to MDW Designer perspective and expand the MyServices package. Open the Employees.java Dynamic Java asset and edit the get() method of this REST service to expect your ID and return your employee information:

@Override

@Path("/{id}")

@ApiOperation(value="Retrieve an employee by their ID",

notes="Currently only retrieves a single employee, and only dxoakes.",

response=Employee.**class**)

**public** JSONObject get(String path, Map<String,String> headers)

**throws** ServiceException, JSONException {

String id = getSegment(path, 2);

**if** ("dxoakes".equals(id)) {

UserVO emp = **new** UserVO(id);

emp.setName("Donald Oakes");

emp.setAttribute("Email", "donald.oakes@centurylink.com");

emp.setAttribute("Phone", "303 992 9747");

**return** emp.getJson();

}

**else** {

**return** **null**;

}

}

Note: This example service comes from the MDW [Services Cookbook](ServicesCookbook.html).

* Save the file, incrementing its version, and type the "cf push" command again.
* Once the push has completed, access the service in your browser through a URL like the following:

[**http://mdwdemo.useast.appfog.ctl.io/mdw/Services/MyServices/Employees/<your-cuid**](http://mdwdemo.useast.appfog.ctl.io/mdw/Services/MyServices/Employees/%3cyour-cuid)**>**

**MDW Cloud Foundry Cookbook – Running Locally**

[Previous](#deploy_cloud_foundry) [Contents](#contents) [Next](#deploy_locally)

**1. Tomcat Container**

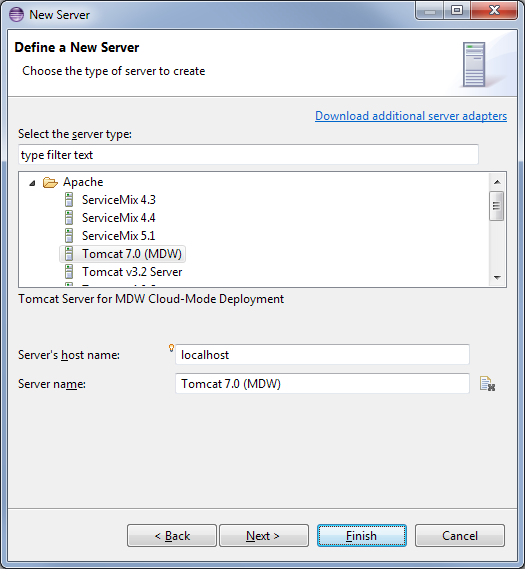
It can quickly become tedious to build and push to Cloud Foundry every time you want to test a code change. To deploy mdwdemo locally, you can run on Tomcat. With a Tomcat server running locally, your changes can be hot-deployed so that pushes and server restarts are not required.

**Supported Tomcat Containers:**

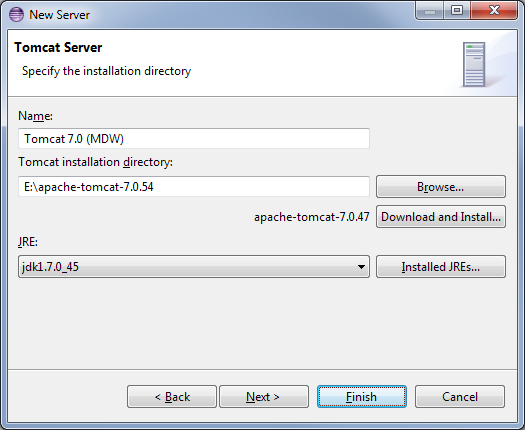
* Apache Tomcat 7:
  + [**http://tomcat.apache.org/download-70.cgi**](http://tomcat.apache.org/download-70.cgi)

**Create a Tomcat Server:**

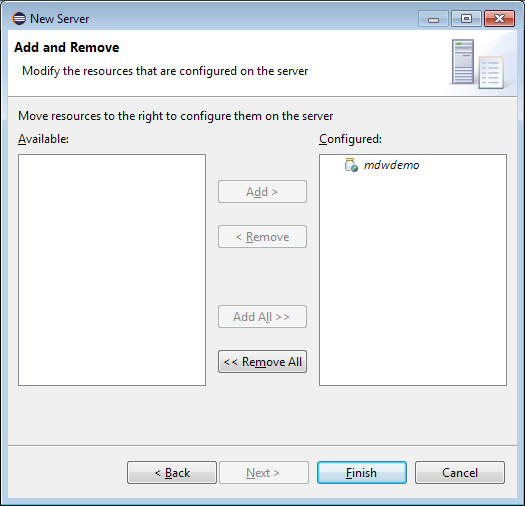
* To execute a workflow process you need a server running with MDW deployed. For debugging in Eclipse the easiest way to set this up is through a Web Tools Platform server instance. From the menu select File > New > Other > Server > Server. Click Next and select Apache > Tomcat 7.0 (MDW) from the options. To be able to debug your Dynamic Java it's important that you select ***Tomcat 7.0 (MDW)*** instead of the standard ***Tomcat 7.0 Server***. The server name is arbitrary, so you can make is something friendlier than the default.



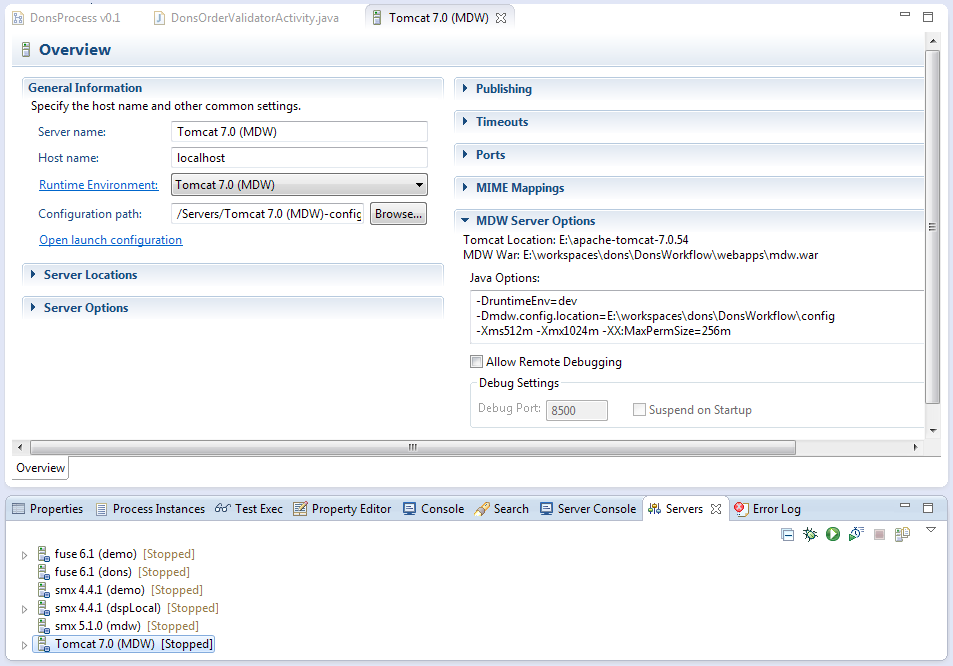
* If you've not previously used a Tomcat 7 runtime in Eclipse, clicking Next takes you to a page where you specify your Tomcat location. Make sure that the selected JRE is Java 1.7.



* The final page of the New Server wizard is where you designate your workflow project to be deployed on the server. After that, click Finish to create the server instance.



* If the Servers view is not visible in your current perspective, from the menu select Window > Show View > Other > Server > Servers. You should see your Tomcat 7 server in this view. You can double-click the server to edit its configuration. Expand the Timeouts section, change the start timeout value to 300 seconds, and hit Ctrl-S to save your changes. Then close the editor.



**MDW Cloud Foundry Cookbook – Running Locally**

[Previous](#tomcat_container) [Contents](#contents)

**2. Deploy Locally**

**Run Tomcat:**

* Edit the following properties in config/mdw.properties:
  + mdw.asset.location - This is the directory path on your hard-drive where your assets are located.
  + mdw.git.local.path - The root directory of your mdwdemo Git project.
  + mdw.hub.user - You may have noticed the Java system property runtimeEnv in your server config, which is preset to "dev". This property allows you to bypass authentication locally.
* Now that you've created the WTP server instance, the Servers view gives you a handy way to start and stop Tomcat. And output is directed to the Eclipse Console view, where you can click on stack traces to open associated source code (including MDW Framework code and Dynamic Java). Start your server in debug mode by right-clicking on it and selecting Debug (or use the icon in the Servers view toolbar).
* The first time you start your server Tomcat explodes the mdw.war file in your deploy/webapps directory and caches the deployable content. This can sometimes take a minute. With the server running you should see MDW output in the Eclipse Console view.

Tip: When you upgrade to a new MDW build version in Eclipse, Designer automatically downloads the corresponding mdw.war file into your deploy/webapps directory. If at any time you want to clean out the MDW deployment and start fresh, you can delete mdw.war and the exploded mdw directory (and for a very thorough cleansing you can even delete the Tomcat cache under deploy/work/Catalina/localhost/mdw). Then you can deploy from scratch from Package Explorer view by right-clicking on your workflow project and selecting MDW Update > Update Framework Libraries.

* You can confirm that MDW was successfully deployed by accessing MDWHub in your browser:

<http://localhost:8080/mdw>

**Make and Test a Change:**

* Make another change to the Employees service. After saving the asset in Designer, you should be able to access the service right away and see your changes at:

[**http://localhost:8080/mdw/Services/MyServices/Employees/<your-cuid**](http://localhost:8080/mdw/Services/MyServices/Employees/%3cyour-cuid)**>**

* Once you've implemented and tested a feature, you can deploy to the cloud using "cf push".

**Next Steps:**

* Check out some of the other MDW developer cookbooks:
  + <http://cshare.ad.qintra.com/sites/MDW/Developer%20Resources/Tutorials>
* Browse through the online help docs, which are the same as those in Eclipse:
  + http://lxdenvmtc143.dev.qintra.com:8989/mdw/doc/index.html
* The MDW JavaDocs:
  + <http://lxdenvmtc143.dev.qintra.com:8989/mdw/javadoc/index.html>