

Run

Table of contents

1 Preparing to run the Duke's Bank demonstration.....	2
2 Execute the application and data build and deployment via Jobcenter.....	5
3 Execute via 'ad'.....	6

Overview

This section describes how to prepare for and run a code module and data BuildAndDeploy workflow job or command and is pertinent to users responsible for releasing content changes from the source code repository and reference database to targeted deployment environments.

There are two interfaces available to execute the BuildAndDeploy commands:

- Using Jobcenter, the web-based graphical application,
- Using AntDepo's `ad` shell command

Instructions for running the BuildAndDeploy command using either interface are explained below.

1. Preparing to run the Duke's Bank demonstration

In general, there are some one-off package installation and builder/service configuration commands to be run before normal ("day-to-day" build and deployment) operations can commence. In particular (regarding the Duke's Bank demonstration that has been installed and configured in the previous tabs of this site) execute the following commands from the Antdepo client designated as the server system:

Note:

While instructions for these one-off commands are given in terms of using the Antdepo shell command it is quite possible to configure these (and any other) commands to execute from Jobcenter too.

- Checkout the Duke's Bank application source code to the build box:

Note:

By default the source is checked out from Sourceforge's Subversion repository. If you would like to be able to make updates, transfer the source into your own repository and modify the development AntBuilder's BuilderScmConnection resource setting via ProjectBuilder or Workbench.

```
$ ad -p DukesBank -t AntBuilder -o development -c scmCheckout
scmCheckout parameters:
{basedir="/home/anthony/dukesbank/development/src",
connection="https://moduleforge.svn.sourceforge.net/svnroot/moduleforge/elements/bra
module="", label="" }
.
.
.
Checked out revision 635.
[command.timer.Builder.scmCheckout: 1:22.116 sec]
```

Run

- Install and configure the Cruise Control based continuous integration environment by issuing the following command from your ControlTier server box:

```
$ ad -p DukesBank -t Site -o developmentCI -c Deploy
Start: "Run the coordinated deployment cycle across the configured
Sites." commands: dispatchCmd
begin workflow command (1/1) -> "dispatchCmd -command Deploy
-resourcename .* -resourcetype [^\.]*" ...
dispatching command: "Deploy " to: (CruiseControl) development ...
dispatching to object: development [CruiseControl] -> "Deploy "
.
.
.
command completed successfully. Execution time: 1:12.122 sec
dispatched command: Deploy completed for: (CruiseControl) development
end workflow command (1/1) -> "dispatchCmd -command Deploy -resourcename
.* -resourcetype [^\.]*"
[command.timer: 1:13.990 sec]
Completed: execution time: 1:13.990 sec
```

The CruiseControl instance should be available from build box at <http://localhost:8081> or similar (depending on your node setup). In addition you'll find a "boot.log" and "cruisecontrol.log" in the CruiseControl installation directory.

Note:

As soon as CruiseControl comes up for the first time it will kick-off an initial build and put the resultant package into the ControlTier server's package repository.

Note:

Deploying CruiseControl also installs the package dependencies (JBoss and Ant) required by the AntBuilder object.

- As a general rule, you can find an object's configuration properties (for example, in this case, the Cruisecontrol installation directory - ccDir) as follows:

```
$ ad -p DukesBank -t CruiseControl -o development -c Properties
[MULTI_LINE]
# development [CruiseControl] #

Duke's Bank continuous integration server

## Attributes ##

* basedir: "/home/anthony/dukesbank/development/src"
* ccDir: "/home/anthony/dukesbank/development/cruisecontrol-bin-2.7.1"
* interval: "60"
* javaHome: "${env.JAVA_HOME}"
* jmxport: "8001"
* path: "/bin:/usr/bin"
```

```
* rmiport: "1100"
* targetdir:
"/home/anthony/dukesbank/development/src/j2eetutorial14/examples/bank"
* webport: "8081"
.
.
.
```

- Install and configure the development and staging JBoss and HSQLDB database instances executing these commands from the ControlTier server node:

```
$ ad -p DukesBank -t Updater -o development -c Deploy
Start: "Run the coordinated deployment cycle across the configured
Sites." commands: dispatchCmd
begin workflow command (1/1) -> "dispatchCmd -command Deploy
-resourcename .* -resourcetype [^\.]*Site" ...
dispatching command: "Deploy " to: (Site) developmentDatabaseServer,
(Site) developmentApplicationServer ...
.
.
.
Completed: execution time: 33.946 sec
dispatched command: Deploy completed for: (Site)
developmentDatabaseServer, (Site) developmentApplicationServer
end workflow command (1/1) -> "dispatchCmd -command Deploy -resourcename
.* -resourcetype [^\.]*Site"
[command.timer: 33.946 sec]
Completed: execution time: 33.946 sec

$ ad -p DukesBank -t Updater -o staging -c Deploy
Start: "Run the coordinated deployment cycle across the configured
Sites." commands: dispatchCmd
begin workflow command (1/1) -> "dispatchCmd -command Deploy
-resourcename .* -resourcetype [^\.]*Site" ...
dispatching command: "Deploy " to: (Site) stagingDatabaseServer, (Site)
stagingApplicationServer ...
.
.
.
Completed: execution time: 15.775 sec
dispatched command: Deploy completed for: (Site) stagingDatabaseServer,
(Site) stagingApplicationServer
end workflow command (1/1) -> "dispatchCmd -command Deploy -resourcename
.* -resourcetype [^\.]*Site"
[command.timer: 15.775 sec]
Completed: execution time: 15.775 sec
```

The initial development and staging JBoss instances are available from the application server box(es) at <http://localhost:8180> and <http://localhost:8280>, or similar (depending on your node setup). You'll find the JBoss "boot.log" and "server.log" under the "default" server instance in the JBoss installation directory for each environment. e.g:

Run

```
$ cd ~/dukesbank/development/jboss-4.0.3SP1/server/default/log
ls *.log
boot.log  server.log
```

Note:

The JBossServer type uses the "spawn" attribute of the [Exec](#) Ant task. As a result, the console output of the JBoss "run.sh" is always lost.

To recap, execution of these deployment commands completes configuration of:

- The build box hosting both the CruiseControl continuous integration service and the ControlTier AntBuilder.
- The development application server box hosting JBoss.
- The development database server box hosting HSQLDB.
- The staging application server box hosting JBoss.
- The staging database server box hosting HSQLDB.

Along with the server box hosting the ControlTier server (Workbench, WebDAV & Jobcenter) you may have deployed software to as many as six boxes (or as few as one).

Note:

By this stage you will have seen what may appear to be several redundant installations of the same software be executed (Java, ATG, JBoss). These are necessary in order to ensure each of the application "service" instance deployments can operate completely independantly of one another.

2. Execute the application and data build and deployment via Jobcenter

In [Configure Step #3](#), a new job will have been defined for the BuildAndDeploy process. After logging into Jobcenter a list of the defined jobs will be displayed. Choosing and running a job will execute the underlying BuildAndDeploy command.

The general steps to using Jobcenter to operate code and data BuildAndDeploy and Update are listed:

1. Login to Jobcenter
2. Identify the desired job
3. Run the job
4. Customize a report

Considering the specific set of jobs loaded to Jobcenter as part of the Duke's Bank demonstration:

- Locate and select the "development.BuildAndDeploy" job and use the "Choose Options

and Run Job ..." button to set the "buildstamp" option and "Run Job Now" to run the application code build and deployment process against the development environment:

Note:

The "buildstamp" is equivalent to the "release version" of the application. You can choose whatever scheme appeals to you to ensure that each build is uniquely identified.

- You can follow the job's progress using the "Tail Output" view in Jobcenter
- Once the job has succeeded there will be two packages (JBossEar and HsqldbRdbDmp) in the package repository, and the development environment will running this new build and database dump.
- Similarly, using the same buildstamp, find the "developmentCatalog.BuildAndDeploy" job and follow the same process to acquire a database dump and reload it to the development environment.
- Lastly, find the "staging.Update" job and follow the same process to update both the staging application server and database instances with the selected buildstamp.

At this stage the development environment is up and running the Duke's Bank sample application: <http://localhost:8180/bank/main>, and the staging version of the application is at <http://localhost:8280/bank/main>, or similar (depending on your node setup). You can login to either site using the username "200" and the password "j2ee".

3. Execute via 'ad'

An alternative to executing the these commands via Jobcenter is to execute them directly via the `ad` shell command. The general usage is shown below:

```
$ ad -p project -t Updater -o name -c BuildAndDeploy -- \
  -buildstamp buildstamp
```

A typical convention is to use the date and time as the `-buildstamp` argument. For example:

```
$ ad -p project -t Updater -o name -c BuildAndDeploy -- \
  -buildstamp 200711071500
```

If you are an experienced AntDepo user, you may also know how to run individual parts of the build and update workflow by running the appropriate command from one of the subordinate commands. For example, to run just the Deploy:

```
$ ad -p project -t Updater -o name -c Deploy
```

Or to run just the build, you can execute the Build workflow separately:

```
$ ad -p project -t Updater -o name -c Build -- \
  -buildstamp buildstamp
```