Installing The Codename One Build Server (beta)

Getting Started

Modified: February 26, 14

Prerequisites

Codename One server setup requires decent system familiarity with Linux/Mac & Windows installs.
You will need two Linux machines (can be virtualized) at least one Mac and at least one Windows 8 machine (both cannot be virtualized). It is expected that machined are reasonably modern and have at least 100gb of storage ideally over SSD. 8gb of RAM is recommended although less could be used.

You will need to install JDK 1.7 (not the JRE) on all of the machines. Both 64 bit and 32 bit versions of the JDK should work.

You will need to setup a shared folder between the machines that you can update and that will be mirrored across the machines over the network e.g. using Samba/Windows share.

The Codename One corporate server setup is divided into the following distinct stages:

- Server setup
- Shared Folder
- Linux build server setup
- Windows 8 build server setup
- Mac build server setup
- Client configuration
- Updates

It is recommended that all servers reside within the corporate network to avoid potential security risks. However, this would require that the mobile devices have access to the corporate network for some functionality to work.
The servers need access to the Internet only in order to provide push notification support and can work without an external network otherwise.

Notice that the files in the distribution are password protected to provide another layer of security.

Your distribution should include the following files:

- cloud_server.zip – the cloud server
- base_system1.zip & base_system2.zip – the base image for the shared folder split to two zips to keep the size down
- source.properties – placed in the base system shared folder separately. Seethe updates section to understand its significance.
- linux_build_image.zip – image file for the Linux build server
- mac_build_image.zip – image file for the Mac build server

SUPPORT@CODENAMEONE.COM
Lev Avraham 1
Tel Aviv, Israel, 64284

WEB
http://www.codenameone.com/

- windows_build_image.zip – Windows 8 build server image
- codenameone_ios_cert.zip – certificates for iOS development

Server Setup

The Codename One server is the web/application server to which developers connect in order to send their builds and retrieve completed builds. It can also be used in runtime for app configuration.

All servers and clients must be able to access the server via a specific URL; this URL can be a name or an IP address for the server.

Setting up the server is relatively simple, just unzip the cloud_server.zip file into a destination within the server and make sure Java 7 is properly configured in the server (including JAVA_HOME and PATH variables) the command java –version should return that this is indeed a Java 7 VM.

Within the hierarch of the server locate the file server_config.properties (it should be under server/web/WEB-INF/classes/).

Uncomment and fill out the variables serverURL and fromEmailAddress to indicate the URL of the server based on the IP/DNS address of the server (including http and closing slash). Notice that that URL must end with :8080/ to indicate the proper port and closing slash.
The fromEmailAddress should indicate an address used as the “from” when emails are dispatched from the Codename One server.

Also update email.properties within the same hierarchy with the proper settings for SMTP email.

Add the run.sh file from the root of the unzipped file into the startup script for your server, launch the run.sh script then access the URL you typed above into the server_config.properties file.

You should see a login/signup dialog. To test this is indeed working you can signup, activate your account and login.

Shared Folder

The shared folder is where the build_server.zip file should be expanded. The shared folder should be mapped into a directory on every operating system e.g. /mnt/shared or x: for Windows etc.

Within the shared folder you will need to unzip base_system1.zip and base_system2.zip then copy into place the source.properties file. The order becomes important with updates.

Linux Build Server Setup

The Linux build server needs to have a user named ec2-user under which all the work should be done.

Besides the standard installation of JDK 1.7 you will also need to install xvfb so the command line xvfb-run will work as expected.

Android SDK for Linux must be downloaded from the Google site and extracted to the home directory. Rename the directory “android-sdk” then within android-sdk/tools issue the command:
./android update sdk --no-ui –all

This should download the entire Android development environment into place, which should take a while to complete.

You will need to extract the file linux_build_image.zip to /home/ec2-user once that is performed you should see the files android.sh, rim.sh & j2me.sh

Edit all of them and make sure they point to your shared folder directory correctly. E.g.:

java -jar AutoDeploy.jar /full_path_to_shared_folder .

Don’t miss the dot at the end of the command! Now enter the respective directories for rim, android & j2me. Edit the deployment.properties files to point at your cloud server URL instead of the placeholder URL.

Once this is done bind the commands to run on startup so restarting the server will relaunch these processes.

Mac Build Image

Make sure to install the latest JDK 1.7 from Oracle onto the Mac. Download xcode 5 (not the latest version) from Apple’s developer site and install it.

Rename /Applications/xcode.app to /Applications/xcode5.app

Launch xcode5 and create/run the hello world iOS project to initialize all elements.

Disable the computer from entering full sleep mode in the power saving section in the settings.

Open Keychain Access (click the magnifying glass at the top left and type Keychain to find this). You should see an unlocked icon with the word login selected in bold. Enter preferences for the app and make sure that within the first aid

section all options are checked.

Import the certificates from the Codename One iOS certificates zip by double clicking the p12 files and the provisioning profile files. The password for the certificates is “password”.

Unzip the mac_build_image.zip file onto the desktop edit run.sh. Fix it to point to the shared folder.

Now enter the directory created on the desktop and edit the deployment.properties file to point at your cloud server URL instead of the placeholder URL.

Run run.sh

Edit runDesktop.sh and make a similar set of fixes to it.

Launch it as well.

It is possible that a dialog for permissions will popup on the Mac during a build. Select Always Allow to prevent this from repeating and from that point on it should work smoothly. Its possible this will recur when rebooting the machine.

Windows Server Desktop

Install Visual Studio Express 2012 for Windows Phone.

Install inno setup:
http://www.jrsoftware.org/isinfo.php

Install wix:
http://wixtoolset.org/

Unzip windows_build_image.zip onto the desktop. Edit win.bat and windesk.bat to point them at the shared folder you mapped to the machine.

Now enter the directories created on the desktop and edit the deployment.properties files to point at your cloud server URL instead of the placeholder URL.

Bind the win.bat and windesk.bat files to the start up process of the machine.

Client Configuration

This process is currently only supported on NetBeans. Once NetBeans is configured other IDE’s will work with the configuration defined from NetBeans.

On the developer machines install NetBeans, go to the global NetBeans settings. Select “Codename One” then “Advanced” and type in the URL for the cloud server.

All builds sent from this machine should now arrive to the cloud server.

You will need to register and activate the user on the build server for builds to actually reach it.

Updates

Updates come in two flavors; cloud updates can be unzipped on top of the current deployment assuming the server was stopped using server/bin/shutdown.sh then started again as usual.

Other updates should be unzipped into shared folder. Special care needs to be given to the extra file source.properties, which is shipped separately. The source.properties file must be copied into the shared folder only after its completely synced. If the shared folder uses an immediate protocol such as NFS or SMB this shouldn’t be a problem the file can be copied immediately after the copy of the other files. However, if the protocol isn’t instant e.g. dropbox, box etc. the file must by copied into place only after all other files were synchronized to all the servers. This is important since that file marks that an update is pending and the data might be partially synchronized if it is copied on top.