DevOps – IIB continuous build and deploy automation using GitHub, Jenkins and JFrog Artifactory.

Surendrakumar Jebakani
Perficient Inc.
OVERVIEW .................................................................................................................................3
1.1 BASIC FUNCTIONALITIES .......................................................................................................4
1.2 BUILD ENVIRONMENTS .........................................................................................................4

DEVELOPMENT ..................................................................................................................................5
2.1 GITHUB CONFIGURATION ........................................................................................................5
2.2 IIB TOOLKIT CONFIGURATION (Using eGit plugin, you can clone GitHub repositories into your IIB toolkit workspace.) .....................................................13

JENKINS ..............................................................................................................................................25
3.1 ANT BUILD SCRIPT ....................................................................................................................25
3.2 JENKINS CONFIGURATION .......................................................................................................26
3.3 SCM Configuration .....................................................................................................................31

JFrog Artifactory Configuration ......................................................................................................38

CONTINUOUS BUILD AND DEPLOYMENT ....................................................................................43
1 OVERVIEW

This document demonstrates how continuous build and deploy automation is achieved for IBM Integration Bus v9 deployment using open source GitHub, Jenkins and JFrog Artifactory.
1.1 BASIC FUNCTIONALITIES

- **Full Deployment** – Latest code is pulled from GitHub repositories, and each IIB applications are assembled, packed, override properties applied then deployed to respective Integration Server (Execution Group) on IIB nodes.
- **Assemble and Deploy by Application Name** – Latest code for the given Application is pulled from GitHub repository. It is then assembled, packed, override properties applied then deployed to respective Integration Server (Execution Group) on IIB nodes
- **Deploy Only** – Latest bar file is taken from Git repository and after the environment specific bar override applied, the overridden bar file is deployed to respective Integration Server (Execution Group) on IIB nodes
- **Assemble and Build** – Latest code for the given Application is pulled from GitHub repository. It is then assembled, packed, and compiled. The newly created bar file is published to the Artifactory. In this option, deployment will not happen.
- **Deploy from Artifactory** – Works only with “Deploy Only” option, when this option is selected bar file is downloaded from Artifactory (by default latest bar file for the given application is downloaded. Bar file can also be downloaded by specifying the target build number)
- **Continuous Build and Deployment**
  - Scheduled Build Trigger – the build job gets triggered automatically by a scheduler at the scheduled time.
  - Build when a change is pushed to GitHub – the build job gets triggered automatically every time when there is a change pushed by someone to IIB GitHub repositories.

1.2 BUILD ENVIRONMENTS

- **IBM Integration Bus v9.0.0.6** – Dev platform
- **Jenkins v2.0** – for continuous integrations
• **GitHub** - for version control (eGit plugin can be installed on top of IIB Toolkit for source code management. i.e pull/push changes from/to Git repository)

• **Apache ANT v1.9.7** – for build automation

• **JFrog Artifactory 4.13.2** – Artifact Repository Manager for archiving the build artifacts

## 2 DEVELOPMENT

### 2.1 GITHUB CONFIGURATION

- Create multiple repositories, one per IIB application and one for all the common components (error handling, common esql procedure/functions, logging etc).
Create a dedicated repository for IIB build process to store and manage the build related artifacts.

- Click the Green New button

- Have a naming standard for naming the repositories.
- Make the repo Public and Initialize the README file.
Create a new repository

A repository contains all the files for your project, including the revision history.

Owner  Repository name

Great repository names are short and memorable. Need inspiration? How about curly-palm-tree.

Description (optional)

RESTful API service for getting/updating info

Public
Any logged in user can see this repository. You choose who can commit.

Private
You choose who can see and commit to this repository.

Initialize this repository with a README
This will let you immediately clone the repository to your computer. Skip this step if you’re importing an existing repository.

Add .gitignore: None

Create repository
- Click the README.md file to edit.
Fill in the “Edit File” tab with info using the text below as a template.

# IIB Account Service

RESTful API service for getting/updating account info

- AccountService_APP - Application containing Account Service API

*The following are used by the service but located in iib-common repository*
- Common_CICS_LIB - Library containing CICS HTTP caller subflow
- Common_LIB - Library containing error handler subflows
- CommonUtils_LIB - Library containing common util code

*An application is a container for all the resources that are required to create a solution. An application can contain IBM® Integration Bus resources, such as flows, message definitions, libraries, and JAR files.*
A library is a logical grouping of related code, data, or both. A library typically contains reusable helper routines and resources such as subflows, ESQL modules, message definitions, maps, and Java utilities. A library is useful to group together resources of the same type or function, for reuse or ease of management. Libraries are optional.*

Integration_IIB_TestService

RESTful API service for getting/updated info

- Integration_IIB_TestService - Application containing Account Service API

*The following are used by the service but located in iib-common repository*

- Common_CICS_IIB - Library containing CICS HTTP caller subflow
- Common_LIB - Library containing error handler subflows
- CommonUtils_IIB - Library containing common util code
- [%] - Java project folder containing java code for logging
- [%] - Library containing logging subflow

*An application is a container for all the resources that are required to create a solution. An application can contain IBM® Integration Bus resources, such as flows, message definitions, libraries, and JAR files.*

*A library is a logical grouping of related code, data, or both. A library typically contains reusable helper routines and resources such as subflows, ESQL modules, message definitions, maps, and Java utilities. A library is useful to group together resources of the same type or function, for reuse or ease of management. Libraries are optional.*

- Click Commit Changes
Repo should be successfully created. Click the “Clone or download” button and then the “Copy to Clipboard” button to copy the HTTPs URL.
2.2 IIB TOOLKIT CONFIGURATION (Using eGit plugin, you can clone GitHub repositories into your IIB toolkit workspace.)
   o Switch to IIB Toolkit and go to the Git Repository Exploring Perspective
- Click the Clone Repo Button

- Paste the URL in the URI box and put in your Git Username and Password
- Click the Next Button
Branch Selection

Select branches to clone from remote repository. Remote tracking branches will be created to track updates for these branches in the remote repository.

Branches of https://git.example.com/Integration_IB_TestService.git:

Select All Deselect All

master

< Back Next > Finish Cancel
• Click Browse and select a local folder that you have chosen to store all your Git Repositories. Click Finish.

The repository should now show in your Git Perspective.
To share the IIB project (application, library etc) to Git repository:

- Right click on the project and select “Team” and “Share application”.
Right click on the project and select “Team” and “Share Project”.
Select “Git” and click Next.
o Select the repository from the drop-down list and click Finish.
To push the changes to GitHub:

- Make the changes in the message flow code, after the changes are unit-tested locally and ready for DEV/QA deployment:

  ![Image of Git Staging view](image)

  - You will see all the changed artifacts in the “Git Staging” view.
  - Select the changed files under “Unstaged Changes” in “Git Staging” and move to “Staged Changes” by drag-and-drop.

  ![Image of Git Staging view after selection](image)

  - Add a comment in “Commit Message” and click the “Commit (Ctrl + Enter)”
As you soon as you committed the change, you will see a change pending to be pushed (↑) in corresponding Git repository. At this time the changes are only committed locally and not pushed to the Github server repository.

Right click on the repository and select “Push to Upstream” to push the changes to Github server repository.
Confirm the change in GitHub by logging into https://github.com/<yourcompanyname>/<repository name>

3 JENKINS

3.1 ANT BUILD SCRIPT

- Ant needs to be installed on the Jenkins server.
- An ant build.xml is created with multiple ant target tasks to perform the following tasks
  - GetApplicationList – For “Full Deployment”, IIB application list is taken from the Jenkins property file
  - assembleApp – Get the IIB code from Jenkins workspace (pulled from Git repositories)
- **mqsipackagebar** – To create deployable BAR files. (mqsicreatebar could not be used as IIB Toolkit is not installed on build Jenkins/IIB servers)
- **mqsiapplybaroverride** – To replace the environment specific configurable values taken from the bar override property files
- **mqsideploy.deploybar** – To deploy the overridden bar file to respective Execution Group (Integration Server) on multiple IIB nodes
- **copyFiles** – To copy the deployable bar file from one IIB server to other IIB servers for deployment
- **cleanupFiles** – To cleanup the files/directories from IIB server after the deployment

### 3.2 JENKINS CONFIGURATION

- Jenkins is used to perform the typical build server work, such as doing continuous/official/nightly and on-demand builds.
- Login to Jenkins server to create a new build job for IIB. Click “New Item”, enter the name of the build project in “Enter an item Name”, and select “Freestyle project” and then click “ok”. (There is a build project “iib-build-job” created in Jenkins for continuous IIB build deployment purpose)
You can parameterize the project with parameters that you want to input into the build. These parameters will be passed to the build job when you start the build.

Select the PROJECT_NAME in which the Application resides.

- APPLICATION_NAME
  - Select the Application you wish to deploy.
## Project iib-build-job-NonProd

This build requires parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROJECT_NAME</td>
<td>ETDA</td>
</tr>
<tr>
<td>APPLICATION_NAME</td>
<td>ETDA_AccountService_APP</td>
</tr>
<tr>
<td>DEPLOYMENT_OPTION</td>
<td>DEPLOY_ONLY</td>
</tr>
<tr>
<td>DEPLOY_FROM_ARTIFACTORY</td>
<td>NO</td>
</tr>
<tr>
<td>ARTIFACTORY_BUILD_NO</td>
<td>ENTER THE BUILD NUMBER</td>
</tr>
<tr>
<td>ENV_NAME</td>
<td>DEV</td>
</tr>
<tr>
<td>BRANCH_NAME</td>
<td>master</td>
</tr>
<tr>
<td>USER_NAME</td>
<td>1310586</td>
</tr>
<tr>
<td>PASSWORD</td>
<td><strong>vision login user name</strong></td>
</tr>
</tbody>
</table>

**Build**
There are three deployment options given.

- **ASSEMBLE_AND_DEPLOY** - Pulls the sourcecode from GitHub repository, builds, applies bar overrides and then deploys. Newly built bar file will be published to Artifactory.
- **ASSEMBLE_AND_BUILD** - Pulls the sourcecode from GitHub repository, builds but DOESN’T deploy. Newly built bar file will be published to Artifactory.
- **DEPLOY_ONLY** - Gets the bar file from GitHub, applies bar overrides and then deploys.

When the “DEPLOY_ONLY” option is chosen, also choose whether to deploy from Github or Artifactory.

- **YES** - To deploy the bar file from Artifactory.
- **NO** - To deploy the bar file from GitHub.

Enter the Artifactory build No to retrieve the bar file. (Ex: 89). If you enter LATEST, it will download the latest bar file from Artifactory and deploy the same.

---

**ARTIFACTORY_BUILD_NO**

**ENV_NAME**

**DEV**
If you choose YES for DEPLOY_FROM_ARTIFACTORY?, then enter either the Artifactory Build Number of the bar file or “LATEST” to get the latest bar file.

- ENV_NAME: Name of the IIB environment, DEV or QA
- BRANCH_NAME: GitHub repository branch name
- USER_NAME: IIB server login username (AD username)
- PASS: IIB server login password (AD password)
3.3 SCM Configuration

- Make sure that Multiple SCM plugin in installed in Jenkins.
- Select “Multiple SCMs” option under “Source Code Management” and add all the IIB Git repositories.
- **Build Configuration**
  - In the “Build” section, click “Add build step” and select “Invoke Ant” from the drop-down list.
  - Fill-up the information as below
    - Ant Version – Name of the Ant installation.
- Targets – Not needed if you are using custom build file. If not, specify the ant target tasks here.
- Build File – The location of custom build file. It is pulled from Git “iib-build-process” repository
- Properties – You can add the properties with key-value pair in this section.

- **FULL_DEPLOYMENT**
  - It is used for deploying all the IIB applications to respective EG/IS running on multiple IIB nodes.
- Make sure to add the application name in the APPLICATION_NAME list first.
- Make sure to add the Git repositories in Jenkins “Multiple SCMs” under “Source Control Management”.
- Click the “iib-build-job” project from the list of Jenkins projects

- Click the “Build with Parameters” from the listed options at the left side.
- Select the options like above and input your user name and password in USER_NAME and PASSWORD
- Click “Build” to trigger the build deployment.
- You will see a new build job in progress under “Build History”, click on the build job to see the progress
You can also click the “workspace” to see the components pulled from Git repository into Jenkins workspace.
“Console Output” shows the build logs step-by-step. At the end it shows whether the build is success or not.
**ASSEMBLE_AND_DEPLOY**

- This option is used for build and deploying a particular IIB application. (Source code for that given application is pulled from Git repository)
  - Select the IIB application name from the drop-down list in the APPLICATION_NAME input parameter.
  - Select “ASSEMBLE_AND_DEPLOY” in the DEPLOYMENT_OPTION input parameter
  - Select the environment in ENV_NAME
  - After inputting the username/password, click build.

**DEPLOY_ONLY**

- This option is used just for deploying previously deployed bar file. (Bar file is taken from Git repository `iib-build-process/build_process/barfileBaseLocation/`)
  - The bar file is overridden with environment specific configurable values taken from property files (`iib-build-process/build_process/BarFileProperties/DEV/`).
  - Overridden bar file is then deployed.

4 **JFrog Artifactory Configuration**

- Make sure that the JFrog Artifactory plugin installed in Jenkins.
  - In the Jenkins job configuration page, under “Build Environment” select “Generic-Artifactory Integration”
Make the Artifactory configuration as below:

**Upload:**
- Make sure that “Capture and publish build info” is selected under “More Details”

- **In Action:**

  **Upon completion of deployment** – See the Jenkins job logs to get the URL of the Artificatory to see and download the newly built and deployed bar file.

```plaintext
Deploying build link to: http://artifactory:8081/artifactory/job/build

Warning: you have no plugins providing access control for builds, so falling back to legacy behavior of permitting any downstream builds to be triggered

Triggering a new build of jib-tester
Finished: SUCCESS
```

- **To deploy the bar file from Artifactory:**
5 CONTINUOUS BUILD AND DEPLOYMENT

- "Build Triggers" with "Poll SCM" option
  - Using this option, the build job gets triggered automatically by a scheduler at the scheduled time.
  - In the schedule box, put a schedule timing (Ex: H 22 1 1-12 * - This will trigger the build job 1st day of every month at 10:00 PM.)
  - By default, "FULL_DEPLOYMENT" option will be executed (configurable so can be changed)

- "Build when a change is pushed to GitHub" option
  - Using this option, the build job gets triggered automatically every time when there is a change pushed by someone to IIB GitHub repositories.
  - Get the Jenkins hook URL
    - In Jenkins, go to Manage Jenkins → Configure System, look for “Manually manage hook URL’s in the “GitHub Web Hook” section. The hook URL will be shown when you’re clicking on the question mark icon on the right
side. Usually, it consists of your hostname and the /web-hook/ postfix. Example: https://<your-domain-name>/web-hook/

- Add the hook URL in GitHub repository
  - Go to the IIB repository for which you want to enable “Build when a change is pushed to GitHub”.
  - Under “Settings” and “Webhook & Services”, click “Add service” in the “Services” section.
  - Select Jenkins (GitHub plugin) from the drop-down list.
  - And click “Add service”

- You can click “Test Service” to confirm everything is correctly configured.