Boomi CICD Implementation

Created by Srinivasulu Palem (Reference from Boomi Articles)

[1.Overview: 2](#_Toc1250518987)

[2.Introduction: 2](#_Toc1160544005)

[CICD Pipelines 2](#_Toc355247715)

[3.Prerequisites: 3](#_Toc1465944239)

[4.Installation : 3](#_Toc1916219161)

[4.1. Installing the GIT in the AWS EC2 (Amazon Linux): 3](#_Toc1866895586)

[4.2. Installation of Docker: 3](#_Toc1564458992)

[5.Configuration: 4](#_Toc628874811)

# 1.Overview

* In many organizations, the integration processes and other artifacts created in the Boomi platform need to be plugged into external CI/CD workflows and tools to track, automate, and coordinate their deployment. The Boomi platform offers a rich set of APIs that can be used by clients to automate actions such as runtime installation, component deployment, monitoring, and more.
* Leveraging these APIs, we have built numerous CI/CD implementations to enhance software delivery, code quality, and promote test-driven development.

# 2.Introduction

## CICD Pipelines

Boomi Environments allow you to organize your runtime containers such as Test, QA, UAT, etc. You can promote releases by deploying Packages to each environment using either the UI or API. Environment extensions allow you to manage credentials and settings across environments. These too can be managed by both the UI or APIs.

The APIs provide integration with your pipeline software of choice. Reference implementations include all the leading CICD platforms including Jenkins, GitHub, Azure and more. APIs enable version promotion and test automation to be executed from the pipeline platform of your choice. Pipelines can even be implemented with Boomi Integrate and the Atomsphere API connector.

You may wish to add human intervention in your release workflows. Boomi Flow and the Atomsphere APIs are a perfect solution to implement citizen-friendly release processes. The accelerator has reference implementations of implementing workflows to create and approve candidate releases.

* **Objective**: This documentation is to provide a comprehensive guide for implementing a Continuous Integration and Continuous Deployment (CI/CD) pipeline for Boomi using Jenkins, Git, and Docker on an AWS EC2 instance.

# 3.Prerequisites

* AWS EC2 Setup: Ensure you have an EC2 instance running with the necessary permissions.
* Software Requirements:
  + Jenkins
  + Git
  + Docker
  + Boomi AtomSphere API access

# 4.Installation

After launching the Ec2 server with security group settings to allow the necessary ports (e.g. 8080 for Jenkins).

Login into the server and install git and docker applications

## 4.1. Installing the GIT in the AWS EC2 (Amazon Linux)

Step 1: change the user to Root user by giving the command  
 **sudo su -**

Step2: update the YUM repositories   
 **yum update –y**

Step3: Install git by performing the below command

**yum install git –y**

Step4: verify the git is installed or not

**git –version**

## 4.2. Installation of Docker

Step 1. Update the package List

**yum update –y**

Step 2: Use YUM package manager to install Docker

**yum install docker –y**

Step3: Start Docker service

**systemctl start docker**

Step4: Enable docker to start from boot

**systemctl enable docker**

Step5: Add the ec2-user to Docker group to run the docker commands without sudo user

**usermod -aG docker ec2-user**

Step6: Apply the new group membership without logging out

**newgrp docker**

Step7: Now logout the server and do login to reflect the changes, after login you can perform the docker commands without sudo

**docker –version**

# 5.Configuration

You can find the steps to configure the Jenkins as Docker image in below link.  
[Boomi-CICD\_docker](https://github.com/OfficialBoomi/boomicicd-docker)

**Note:** Update the Dockerfile from the below provide link  
[Updated-Dockerfile](https://hathority-my.sharepoint.com/:w:/r/personal/srinivas_palem_hathority_com/_layouts/15/Doc.aspx?sourcedoc=%7B076A8A8A-4742-4416-8C86-CFA0610FA366%7D&file=Document.docx&action=editnew&mobileredirect=true&wdNewAndOpenCt=1733939644981&ct=1733939645357&wdOrigin=OFFICECOM-WEB.MAIN.NEW&wdPreviousSessionSrc=HarmonyWeb&wdPreviousSession=73ac706d-875f-489f-bf1c-977885ce70e6&cid=6f143575-9f79-4c68-a639-4ebf2ac381cd)

# Steps To run the Docker Container

After following the steps in the above links now you are succesfully build the docker image of Jenkins, we need to run the Jenkins container

To run the container, follow the steps:

docker run -p 8080:8080 -p 50000:50000 --name "boomicicd"

-e JENKINS\_USER=${JENKINS\_USER}

-e JENKINS\_PASS=${JENKINS\_PASS}

-e KEY=${KEY}

-d -t boomicicd/jenkins:latest

# Environment variables

The above docker image uses environment variables for configuration.

NOTE: you can use EXPORT command or by creating environment file to store the variables

|  |  |
| --- | --- |
| Available variables | Description |
| JENKINS\_USER | The admin variable login to Jenkins |
| JENKINS\_PASS | The password variable |
| KEY | KEY to unlock the repo. Get the key [here](https://github.com/OfficialBoomi/boomicicd-docker/blob/master/docker/secret).⁠ |

⁠

**Create a backup volume mount**

mkdir -p /var/jenkins\_home

volume="boomicicd\_vol"

docker volume create --opt type=none --opt device=/var/jenkins\_home --opt o=bind ${volume}

# Configuring Boomi Account

Configure Jenkins to connect to Boomi Account. (This assumes sfirewall/proxy b/w Jenkins and Boomi AtomSphere is set). There are two GIT repos that can be configured (both optional). One is the Git Component Repo where the component XMLs will be automatically stored. The other the GIT Release Repo where the Boomi components can be deployed using a release template. Examples

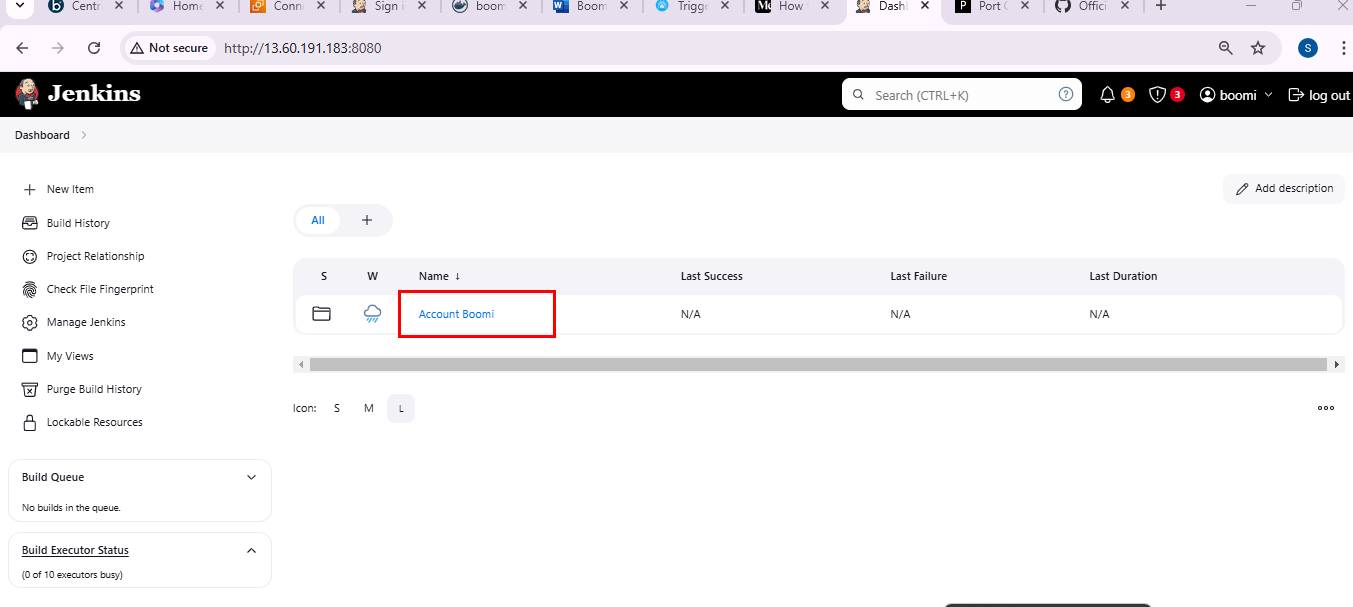
* Login to Jenkins and select the Account {Rename} folder & click configure. Review and update all the fields as required.
* Update the Display Name
* Update the folder properties Name: accountId Value: <>
* Update the folder properties Name: SCRIPTS\_HOME Value: <Full path of CLI /scripts folder>
* Update the folder properties Name: SONAR\_HOME Value: <Full path of CLI / sonar-scanner folder>
* Update the folder properties Name: LOCAL\_ATOM\_INSTALL\_DIR Value: <ATOM\_INSTALL\_DIR on Jenkins>
* Update the folder properties Name: gitComponentRepoURL Value: . If GIT requires credentials use a secretText below
* Update the folder properties Name: gitComponentRepoName Value: <Top level name of the GIT Repo; usually the part before .git>
* Update the folder properties Name: gitComponentOption Value: <CLONE| TAG> if you use CLONE, it clones the repo and pushes the content. Else creates a release tag (zip)
* Update the folder properties Name: gitComponentUserName Value: <git --config global.username>.
* Update the folder properties Name: gitComponentUserEmail Value: <git --config global.email>.
* Update the folder properties Name: gitReleaseRepoAPIURL Value: The API URI for the gitReleaseRepo
* Update the folder properties Name: sonarHostURL Value: .
* Update the folder properties Name: sonarProjectKey Value: <Name of SonarProject (if using Sonar)>

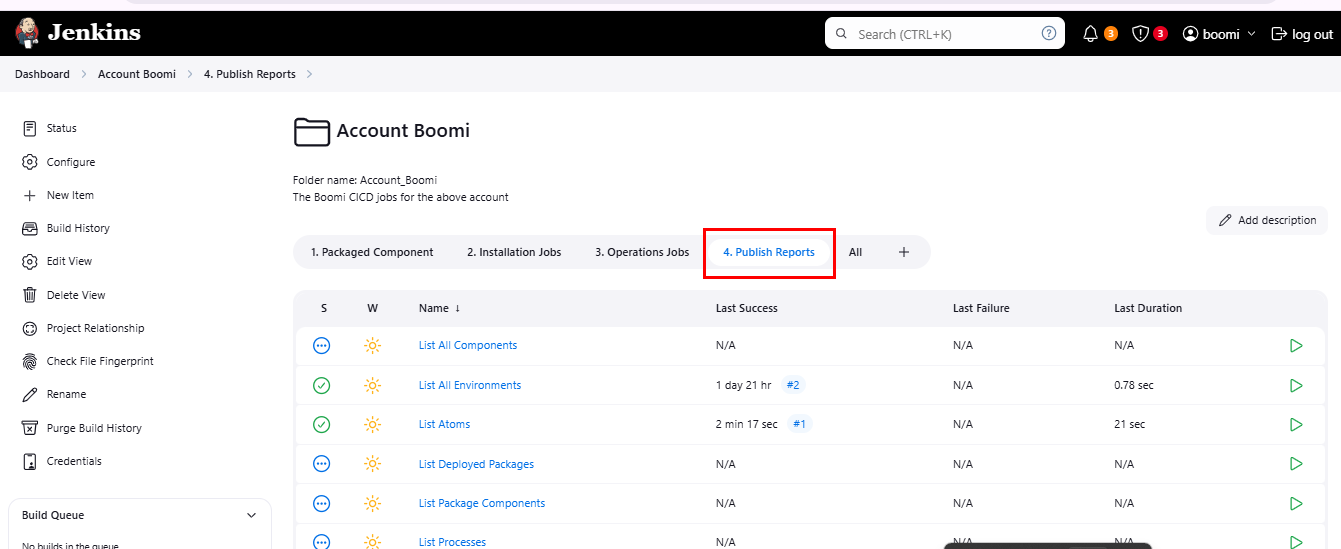
# Configure secrets

* Select the Credentials menu from the left
* Update the authToken to the Boomi API Token (Format) BOOMI\_TOKEN.user@company.com:bOomi-aPi-ToKen-
* Update the sonarToken
* Update the gitComponentRepoURL. If the gitComponentRepo has username and password
* Update the gitReleaseRepoUserPassword. To connect to gitReleaseRepo to recieve WebHooks. See
* Update the gitReleaseRepoAPIToken. To connect gitReleaseRepo to update the commit status using a username and API token.
* Update the JENKINS\_TOKEN. This is used to trigger Jobs using Jenkins API from DynamicGitJob jobs. See
* Click Rename and rename the folder name to Account\_YOUR ACCOUNT

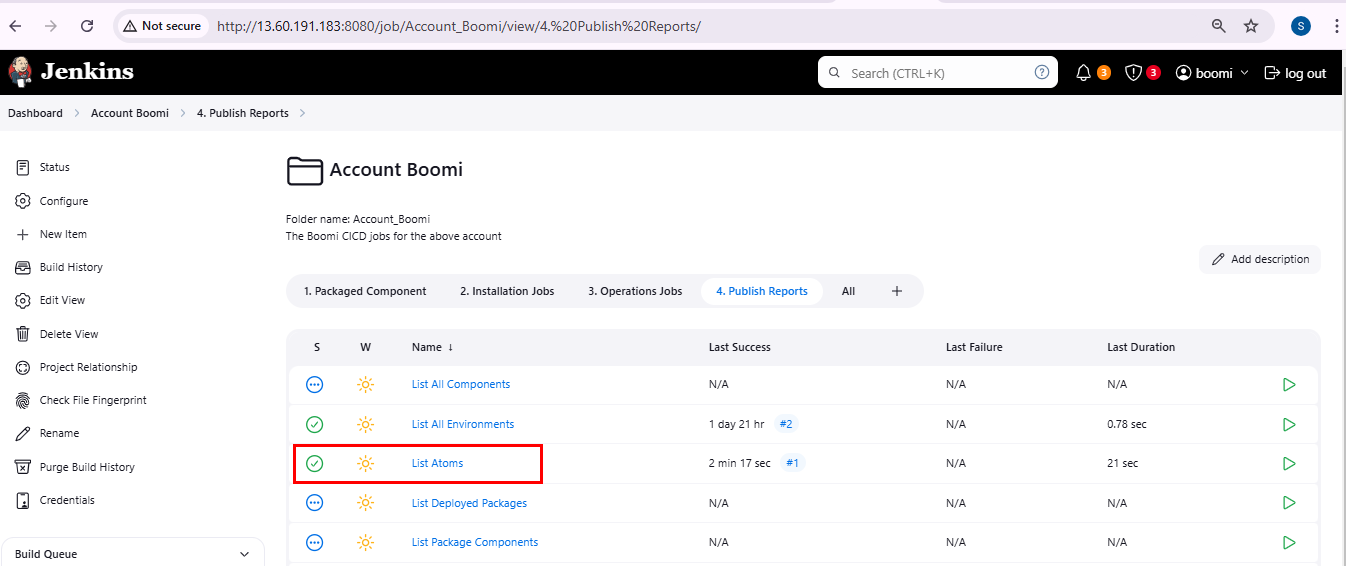
# Run your first Job

* Click on the Account Boomi folder, select the Publish Reports tab

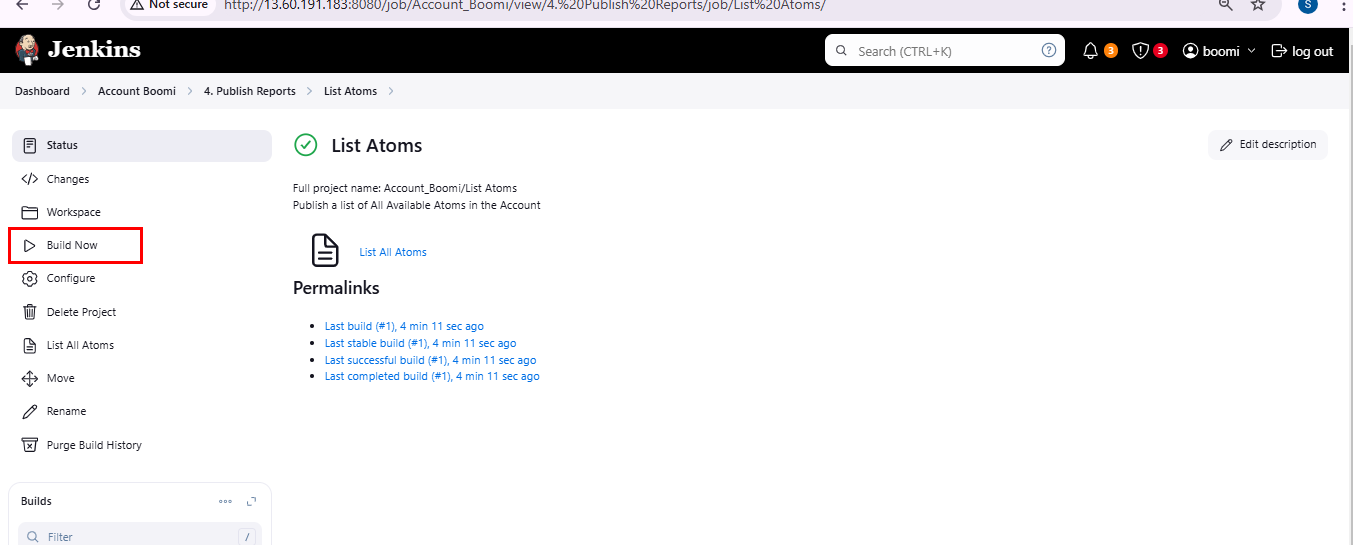




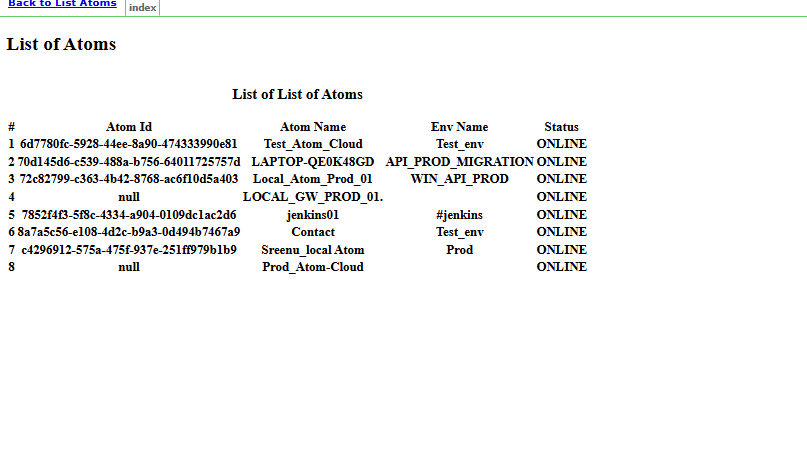
* Click on List Atoms Job



* And click on Build Now



* Once the build is complete refresh the output and select the html report



**Run Jobs**

To run any of the Jobs listed below:

* Click on the appropriate tab/job name
* Select "Build with Parameters"
* Pass the required parameters and click Build
* Review the console output or Jenkins run output for success/error notifications

|  |  |
| --- | --- |
| **Job Name** | **Notes** |
| Change All Listener Status | Changes all the Listener Status on a given Atom |
| Change Listener Status | Changes the Listener Status of a process on a given Atom |
| Continuous Package Deployment Pipeline | Executes a pipeline to create, deploy package, run separate Test process, validates process and promotes to another Env |
| Continuous Process Deployment Pipeline | Executes a pipeline to deploy a process, run separate Test process, validates process and promotes to another Env |
| Create Cloud Atom | Creates a Cloud Atom in a given Cloud |
| Create Environment | Creates an Environment in an Account |
| Create Environment and Attach Atom | Creates Environment and attaches an Atom |
| Create Packages | Creates Package Deployment for a single packages using processName or componentId. Alt will publish a report to GIT and integrate with SonarQube |
| Create Packages | Creates Package Deployment for multiple packages using processNames or componentIds. Alt will publish a report to GIT and integrate with SonarQube |
| Create Process Schedule | Creates Basic Process Schedule. For Advance schedules pleas use the AtomSphere UI |
| Delete Atom and Env | Delete Atom and Attached Environment |
| Delete Local Atom | Deletes Local Atom (Local deployed on Jenkins) |
| Deploy Local Process and Publish to GIT | Deploy a process to Local Atom and extract the component XML to GIT (Legacy Deployment) |
| Deploy Multiple Processes | Deploy multiple-processes to an Environment (Legacy) |
| Deploy Package | Creates and Deploy a single package using componentId or processName to an Environment. Alt will publish a report to GIT and integrate with SonarQube |
| Deploy Packages | Creates and Deploys multiple-packages using componentIds or processNames to an Environment. Alt will publish a report to GIT and integrate with SonarQube |
| DynamicGitJob\_Dev | This is the job points to the specific "Development" git branch. See notes below |
| DynamicGitJob\_Test | This is the job points to the specific "Test" git branch. See notes below |
| Execute Process | Executes a process on a given Atom |
| Get Installer Token | Get a InstallerToken |
| Get Cloud Installer Token | Get a InstallerToken for a Cloud Molecule. Must pass the cloudId |
| Install Local Atom | Install an Atom local to Jenkins |
| List All Environments | Publishes a report of all Environments |
| List Atoms | Publishes a report of all Atoms |
| List Deployed Packages | Publishes a report of Deployed Components in an Env |
| List Package Components | Publishes a report of all Packaged Components of a given Version |
| List Processes | Publishes a report of all Process |
| Promote Multiple Processes | Promotes Multiple Process from one Env to another (Legacy) |
| Promote Process | Promotes a single Process from one Env to Another |
| Publish Package to GIT and Sonar | Publish a package to GIT and perform Sonar validation |
| Publish Packages to GIT | Publish a package to GIT and perform GIT |
| Query Atom | Query status of a single atom |
| Query Execution Record | Query the status of a Process Execution Record within a given timespan |
| Start Local Atom | Start Local Atom |
| Stop Local Atom | Stop Local Atom |
| Undeploy Package | Undeploy a Package component |
| Undeploy Process | Undeploy a Process (Legacy) |
| Update Environment Extensions | Use the JSON file created in the Create Package process to update envirnoment extensions. Use valueFrom to replace secret value with Variables |
| Update Process Schedule Status | Update Process Schedule Status of a single process |
| Update Shared Server | Update shared webserver details of a given Atom |