BluePrism Auto Deployment Cookbook

BP-GIT- Jenkins

By Ashish Easow

In the past few years, I have been involved in several ROM Assessments for Customers.

In many cases, I have observed that most of the Release management activities are done manually either by the support team or with the developer’s assistance. This kind of deployment leaves a loophole which is a red flag during audits with the possibility of unwanted problems later for large production changes like upgrades.

This got me thinking to my old SOA deployment days where we corrected this problem with Jenkins Auto deployment using GIT as the code repository. We also had a pipeline implemented for approvals to higher environments. Recalling that, I worked on a demo for a similar implementation for Blue Prism.

Background

The guide tries to give a simplistic view of the integration required for implementation of CI CD tools – GIT and Jenkins with Blue Prism. A similar approach can be used with other code repositories and deployment automation tools. Blue Prism does not recommend or support any tool and it is up to the implementor to choose the specific tools as per their requirements. This guide is an attempt to provide more visibility in the technical aspects of the set up required for these tools.

Software information

Software List and links

Term	Link
Blue Prism	Robotic Process Automation tool - https://portal.blueprism.com/products/current (v6.6)
GIT	Source code repository. For this guide a public repo was created at: https://github.com/login Organizations can choose to create their own repositories in house. The scope of this guide is to work with an existing repository. Repo used in this guide – https://github.com/*****/**********.git (Redacted for public circulation)
Jenkins	Automation Server used for deployment. Can be downloaded at: https://jenkins.io/download/ User needs to install an automation server after downloading the software. For this guide the automation server was installed and configured to run as a service (default config). The Automation server url for the demo described in the guide is: http://localhost:8080/ The url for the automation job for Blue Prism in the guide is: http://localhost:8080/job/Bp%20Demo/

Pre-requisites

GIT Repo

Prior to starting this process the below steps need to be configured for GIT repository set up

- Set up a public or private git repository by using the Git location provided in the softwares required table above.
- GIT Repo needs to be accessible by the GIT repo url in the system where Jenkins server is planned to be set up.
- GIT repo needs to be accessible for both check-out and check-in of Blue Prism process artifacts.

Jenkins

User will need to set up Jenkins server on a windows machine:

- Set up a Jenkins server by downloading and running an install, provided in the software required links table above
- Configure the Jenkins server service in services.msc, to run under an appropriate windows account (optional if everything works without this).
- Jenkins Server should be accessible at the url configured. Default url provided in the Software links table above.

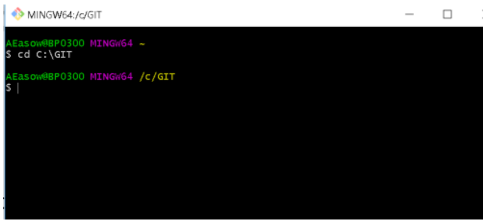
Blue Prism

User will need to install Blue Prism in Jenkins Server machine.

- The install of Blue Prism present in the system where Jenkins server is installed, will be called Jenkins BP.
- For the purpose of this guide the lower environment is called Dev BP, Upper environment is UAT BP.
- Jenkins BP should have UAT BP as a connection in its configuration file.
- The user should have access to create releases/export artefacts from Dev.
- The user should have access to deploy artefacts to UAT.

Code Repository Steps

- For this demo, create a folder GIT in C: . This folder will be used to upload any code to GIT from DEV BP.
- Open Git bash app (search in programs).
- Navigate to C:\GIT in Git Bash by running the cd command.

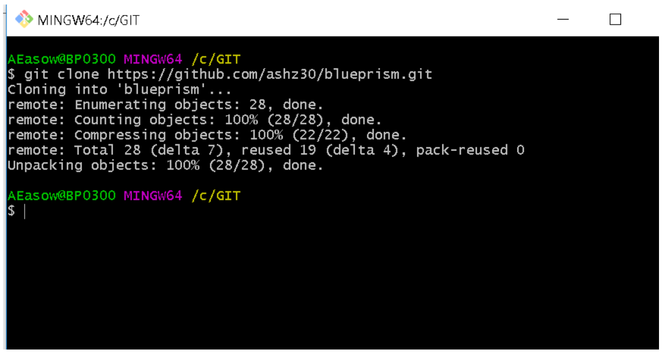
  

  MINGW64:/c/GIT

  ```
  AEasow@BP0300 MINGW64 ~
  $ cd C:\GIT

  AEasow@BP0300 MINGW64 /c/GIT
  $ |
  ```

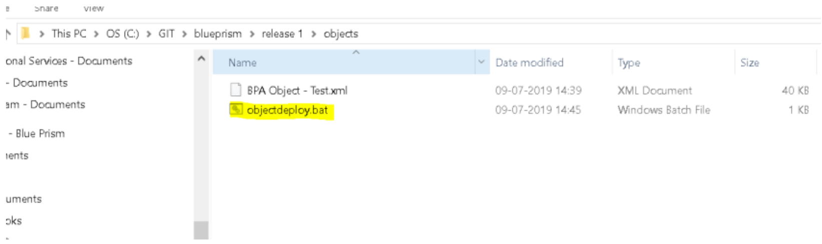
- Run command: git clone https://github.com/****/**********.git

  The command creates a clone of the remote GIT repo, when the command is run, you can see the repo getting downloaded. (should be empty at this point)



```
AEasow@BP0300 MINGW64 /c/GIT
$ git clone https://github.com/ashz30/blueprism.git
Cloning into 'blueprism'...
remote: Enumerating objects: 28, done.
remote: Counting objects: 100% (28/28), done.
remote: Compressing objects: 100% (22/22), done.
remote: Total 28 (delta 7), reused 19 (delta 4), pack-reused 0
Unpacking objects: 100% (28/28), done.

AEasow@BP0300 MINGW64 /c/GIT
$ |
```

- Navigate to blueprism folder in Git Bash by running the cd command.
- Create the required folder structure for the code repo in windows, in this case we have created release 1, inside release 1, we have 1 folder for releases, 1 folder for objects, 1 folder for processes and so on.
- In release 1/objects, create a blank bat file objectdeploy.bat. We will be adding the scripts to this bat file for deployment. (ignore the BPA Object – Test.xml file)


Share
View
This PC > OS (C:) > GIT > blueprism > release 1 > objects
onal Services - Documents
- Documents
am - Documents
- Blue Prism
ients
uments
oks

Name	Date modified	Type	Size
BPA Object - Test.xml	09-07-2019 14:39	XML Document	40 KB
objectdeploy.bat	09-07-2019 14:45	Windows Batch File	1 KB

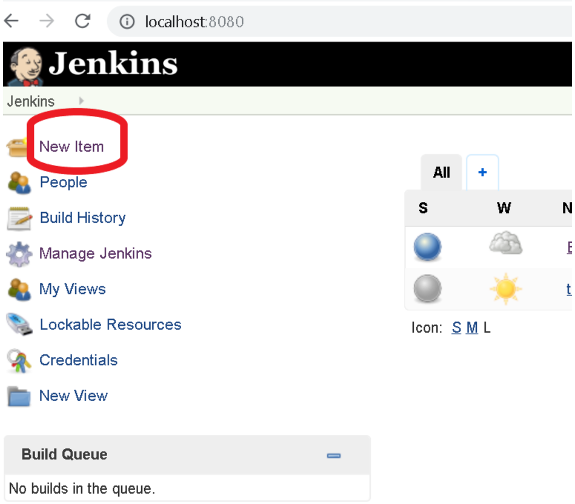
- Each folder can have its own bat file for deployment which can be called in a sequence from command line textbox or indvidually from Jenkins. So we have in objects folder objectdeploy.bat, in process folder processdeploy.bat and so on. For the purpose of this demo, we will be calling each folder's *deploy.bat files from the jenkins.
- Sample objectDeploy.bat commands (processDeploy.bat works in a similar way):

  "C:\Program Files\Blue Prism Limited\Blue Prism Automate\AutomateC.exe" /import "BPA Object - Test.xml" /overwrite /user %1 %2 /dbconname %3
- Sample releasedeploy.bat commands:

  "C:\Program Files\Blue Prism Limited\Blue Prism Automate\AutomateC.exe" /importrelease TestGit.bprelease /overwrite /user %1 %2 /dbconname %3
- User can have a single/multiple config file(s) to keep all the deployable artefacts. This point is not considered as part of this guide and is left upto the user to implement.

Jenkins Steps

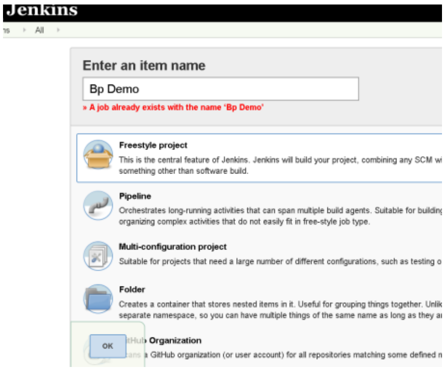
- Open Jenkins using the url localhost:8080 (or your own Jenkins url) and create a new project, by clicking on new item.



Build Queue

No builds in the queue.

- Give a name to the project (Bp Demo in this case), and Select Freestyle Project and Click on Ok.



Enter an item name

Bp Demo

» A job already exists with the name 'Bp Demo'

Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM wi
something other than software build.

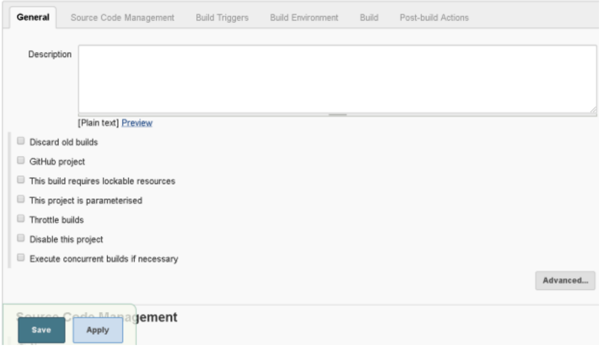
Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building
organizing complex activities that do not easily fit in free-style job type.

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing o

Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlik
separate namespace, so you can have multiple things of the same name as long as they a

GitHub Organization
a GitHub organization (or user account) for all repositories matching some defined n

- Below screen should open up after clicking on Ok.



Description

☐ Discard old builds
☐ GitHub project
☐ This build requires lockable resources
☐ This project is parameterised
☐ Throttle builds
☐ Disable this project
☐ Execute concurrent builds if necessary

- Use the below configuration for the Demo in this guide, or add your own config as required.

Git Hub Project – Checked, value - https://github.com/****/********.git

This Project is Parameterised – Checked

Add Parameters –

ReleaseNumber, Value- release 1 (or what the user provides for the build)

Username, Value- admin (or what the user provides for the build)

Password, Value- **** (or what the user provides for the build)

Environment, Value- choices DEV,UAT,PROD (or what the user provides for the build)

Source Code management – GIT Selected

Repository url populated, Value- https://github.com/****/*********.git (or what the user provides for the build)

Credentials- None (or what the user provides for the build)

Branch to build- */master (or what the user provides for the build)

Additional Behaviours- Clean before checkout

Build Environment –

Delete workspace Before build starts selected

Add timestamps to the console output selected

Build -

(1) Execute windows batch command -

```
cd %ReleaseNumber%
cd objects
objectdeploy.bat %Username% %Password% %Environment%
```

(2) Execute windows batch command -

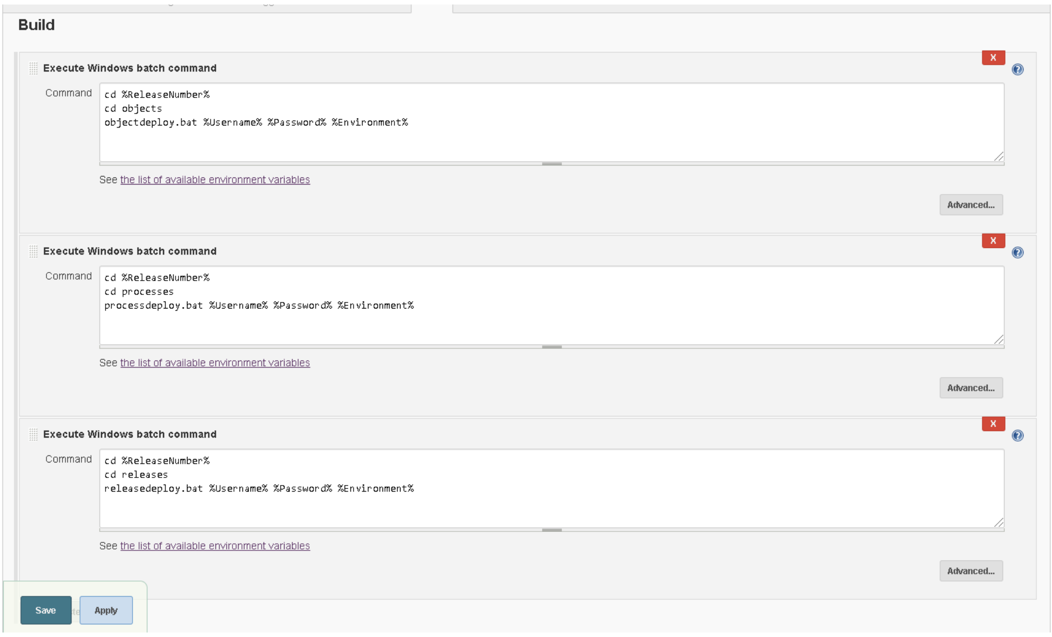
```
cd %ReleaseNumber%
cd processes
processdeploy.bat %Username% %Password% %Environment%
```

(3) Execute windows batch command -

```
cd %ReleaseNumber%
cd releases
releasedeploy.bat %Username% %Password% %Environment%
```

(user can also define custom scripts as required, or chain multiple build commands as done above).

- Click on Save button at bottom after all configuration changes are done.



Build

Execute Windows batch command

Command

```
cd %ReleaseNumber%
cd objects
objectdeploy.bat %Username% %Password% %Environment%
```

See the list of available environment variables

Execute Windows batch command

Command

```
cd %ReleaseNumber%
cd processes
processdeploy.bat %Username% %Password% %Environment%
```

See the list of available environment variables

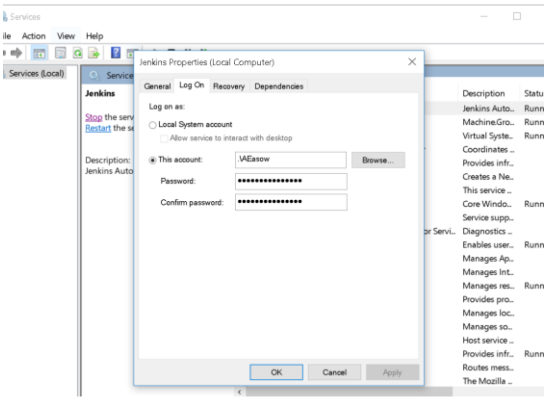
Execute Windows batch command

Command

```
cd %ReleaseNumber%
cd releases
releasedeploy.bat %Username% %Password% %Environment%
```

See the list of available environment variables

- Run Services.msc from windows and configure the Jenkins service to run from a specified user account under which BP is installed.



Jenkins Properties (Local Computer)

Log on as:

○ Local System account

□ Allow service to interact with desktop

● This account: .\AEasow

Password: ●●●●●●●●●●●●●●●

Confirm password: ●●●●●●●●●●●●●●●

Description	Statu
Jenkins Auto...	Runn
MachineGro...	Runn
Virtual Syste...	Runn
Coordinates ...	
Provides infr...	
Creates a Ne...	
This service ...	
Core Windo...	Runn
Service supp...	
Diagnostics ...	
Enables user...	Runn
Manages Ap...	
Manages Int...	
Manages res...	Runn
Provides pro...	
Manages loc...	
Manages so...	
Host service ...	
Provides infr...	Runn
Routes mess...	
The Mozilla ...	

Deployment Steps

- Add runtime deployment files to GIT

GIT.zip

Above is the GIT directory with all the files, it contains the required folder structure as well as the indvidual bat files paramterised for Jenkins deployment. The choice to add objects, process and release artefacts to a list file is left to the user. For the purpose of this guide each artefact file name is present in the bat files.

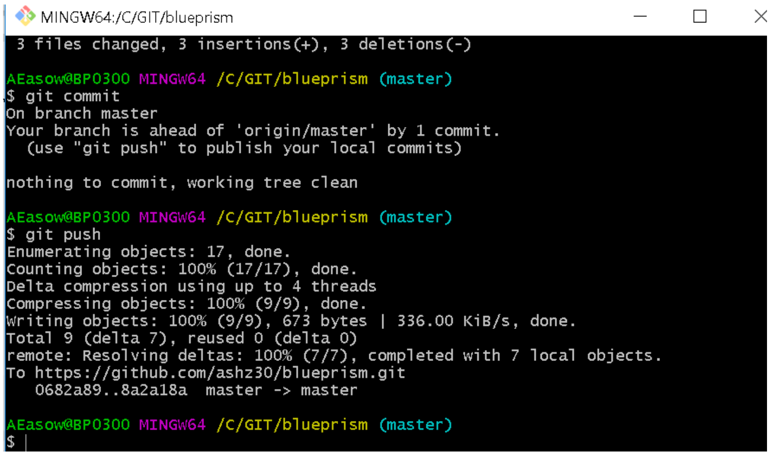
- As per the directory structure, unzip and overwrite the files in the local GIT repo created.
- Open GIT bash. And type the below commands:

```
$ cd /C/GIT/blueprism
$ git add -A
$ git commit
```

Type comment -> esc -> wq!

```
$ git push
```

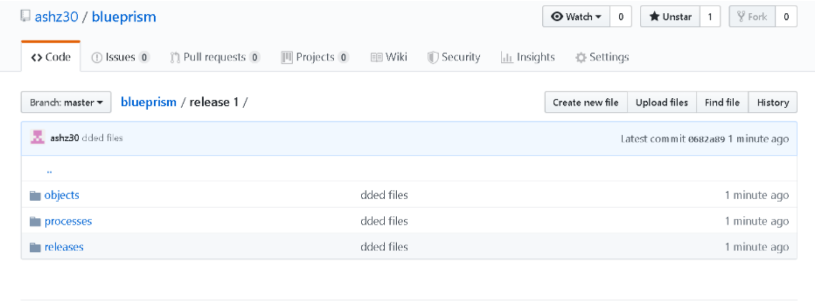
- Once the push command is issued, you should be able to see something similar to the creen below:



MINGW64:/C/GIT/blueprism

```
 3 files changed, 3 insertions(+), 3 deletions(-)

AEasow@BP0300 MINGW64 /C/GIT/blueprism (master)
$ git commit
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean

AEasow@BP0300 MINGW64 /C/GIT/blueprism (master)
$ git push
Enumerating objects: 17, done.
Counting objects: 100% (17/17), done.
Delta compression using up to 4 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (9/9), 673 bytes | 336.00 KiB/s, done.
Total 9 (delta 7), reused 0 (delta 0)
remote: Resolving deltas: 100% (7/7), completed with 7 local objects.
To https://github.com/ashz30/blueprism.git
   0682a89..8a2a18a  master -> master

AEasow@BP0300 MINGW64 /C/GIT/blueprism (master)
$ 
```

- If you refresh your GIT repository online, you should be able to see all the files added in the GIT GUI:



blueprism / release 1 /

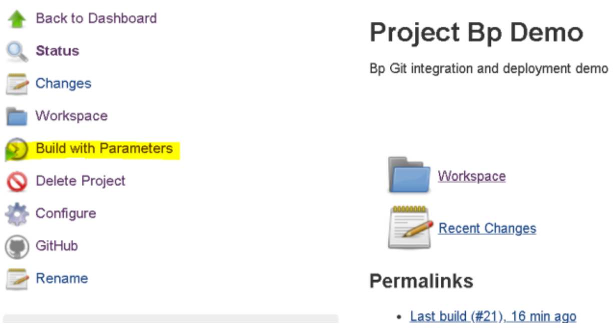
ashz30 dded files — Latest commit 0682a89 1 minute ago

..		
objects	dded files	1 minute ago
processes	dded files	1 minute ago
releases	dded files	1 minute ago

- Create a connection in Jenkins BP install, which points to DEV/UAT/PROD and the connection name should match the choice which has been provided in the Job parameters for Jenkins above. *(Environment, Value-choices DEV,UAT,PROD or what the user provides for the build)*

Job Creation and Tracking

- For creating a runtime Job, log into Jenkins, click on BP Demo (the project which has been configured). And click on Build with parameters option on the left.



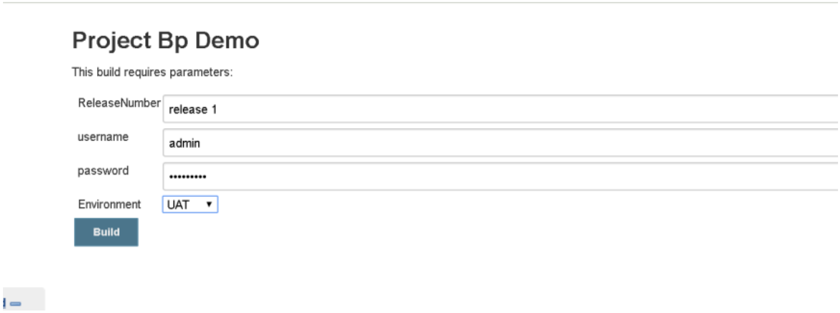
Project Bp Demo

Bp Git integration and deployment demo

Permalinks

- Last build (#21), 16 min ago

- Override any parameters required and click on build:

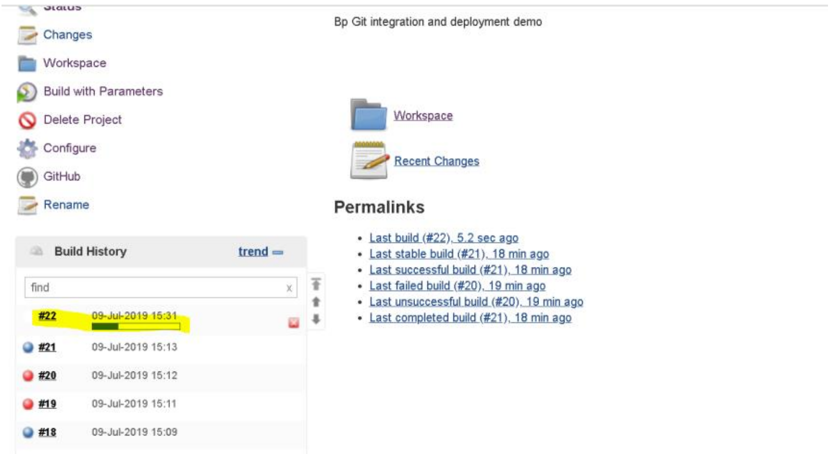


Project Bp Demo

This build requires parameters:

ReleaseNumber	release 1
username	admin
password	•••••••••
Environment	UAT

- If you refresh the page, you should be able to see your job getting executed.



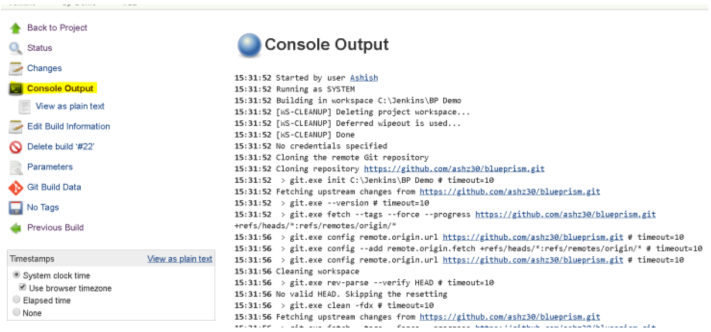
Bp Git integration and deployment demo

#22	09-Jul-2019 15:31
#21	09-Jul-2019 15:13
#20	09-Jul-2019 15:12
#19	09-Jul-2019 15:11
#18	09-Jul-2019 15:09

Permalinks

- Last build (#22), 5.2 sec ago
- Last stable build (#21), 18 min ago
- Last successful build (#21), 18 min ago
- Last failed build (#20), 19 min ago
- Last unsuccessful build (#20), 19 min ago
- Last completed build (#21), 18 min ago

- Click on the build number and console output on the next page shown:



Console Output

```
15:31:52 Started by user Ashish
15:31:52 Running as SYSTEM
15:31:52 Building in workspace C:\Jenkins\BP Demo
15:31:52 [WS-CLEANUP] Deleting project workspace...
15:31:52 [WS-CLEANUP] Deferred wipeout is used...
15:31:52 [WS-CLEANUP] Done
15:31:52 No credentials specified
15:31:52 Cloning the remote Git repository
15:31:52 Cloning repository https://github.com/ashz30/blueprism.git
15:31:52  > git.exe init C:\Jenkins\BP Demo # timeout=10
15:31:52 Fetching upstream changes from https://github.com/ashz30/blueprism.git
15:31:52  > git.exe --version # timeout=10
15:31:52  > git.exe fetch --tags --force --progress https://github.com/ashz30/blueprism.git
+refs/heads/*:refs/remotes/origin/*
15:31:56  > git.exe config remote.origin.url https://github.com/ashz30/blueprism.git # timeout=10
15:31:56  > git.exe config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
15:31:56  > git.exe config remote.origin.url https://github.com/ashz30/blueprism.git # timeout=10
15:31:56 Cleaning workspace
15:31:56  > git.exe rev-parse --verify HEAD # timeout=10
15:31:56 No valid HEAD. Skipping the resetting
15:31:56  > git.exe clean -fdx # timeout=10
15:31:56 Fetching upstream changes from https://github.com/ashz30/blueprism.git
```

- The console output gives the status of your current build. Attached is the output of the console output obtained on following this guide.

console.txt

- Any job run can be viewed and tracked to completion this way and jenkins provides a centralised deployment automation capability for this purpose.

Conclusion and Additional configuration

The user can add more variations to the build process, by adding more parameters, making the list of deployment artefacts configurable and even making the entire process of extracting the release from UAT environment, committing in GIT and deploying to production automated. Jenkins can also be integrated with other SCM tools and tests can be run (Junit, Selenium, Maven) and reports aggregated. There is considerable collateral present online for all these features for Jenkins integration. I have previously created a pipeline for user approvals for higher environments, it works great.

Users are encouraged to explore.

Today spent in worrying about travel plans for next month,

Your friendly experimental Release Manager,

Ashish Easow