Blue Prism Auto Deployment Cookbook (v2)

BP-GIT- Jenkins

Trademarks and copyrights

Blue Prism Limited, 2 Cinnamon Park, Crab Lane, Warrington, WA2 0XP, United Kingdom
Registered in England: Reg. No. 4260035. Tel: +44 870 879 3000. Web: www.blueprism.com

Contents

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 (v6.7)
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/ashz30/blueprism.git
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 jobs for Blue Prism in the guide is: http://localhost:8080/job/BpImport/ http://localhost:8080/job/BPExport/ http://localhost:8080/job/BuildApproval/

Pre-requisites

GIT Repo

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

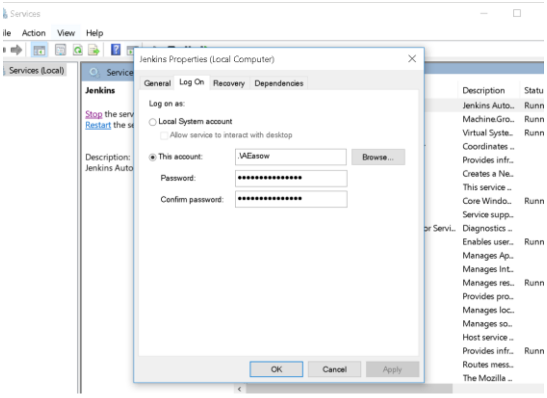
- Set up a public or private git repository by using the Git location provided in the softwares required table above.
- GIT Repo needs to be accessible by the GIT repo url in the system where Jenkins server is planned to be set up.
- GIT repo needs to be accessible for both check-out and check-in of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

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 ...	

- Jenkins Server should be accessible at the url configured. Default url provided in the Software links table above.

Blue Prism

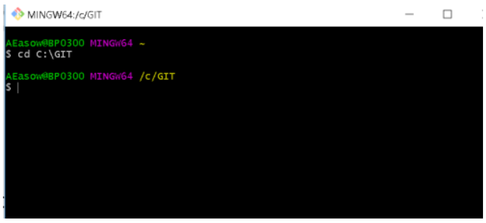
User will need to install Blue Prism in Jenkins Server machine.

- 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.

BP Import project:

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.

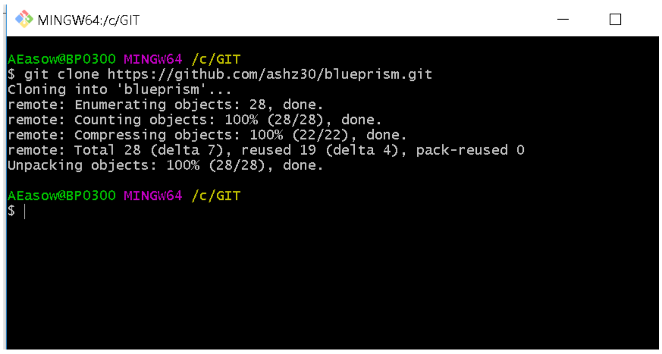


MINGW64:/c/GIT

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

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

- Run command: git clone https://github.com/ashz30/blueprism.git

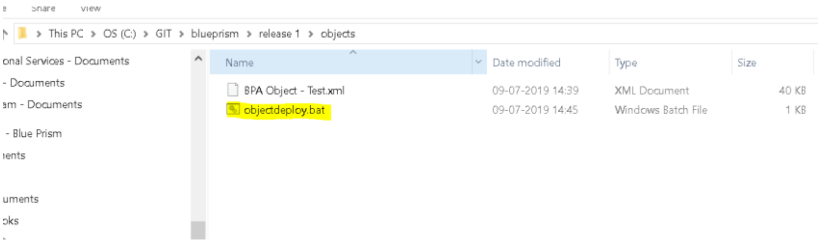
  The command creates a clone of the remote GIT repo, when the command is run, you can see the repo getting downloaded. (should be empty at this point)



MINGW64:/c/GIT

```
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)



This PC > OS (C:) > GIT > blueprism > release 1 > objects

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

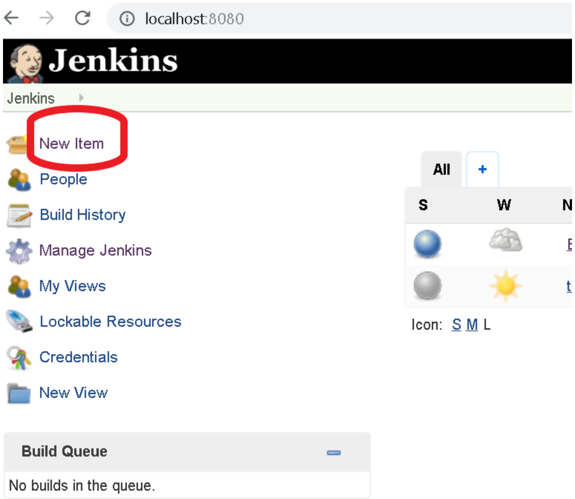
- Sample releasedeploy.bat commands:

```
"C:\Program Files\Blue Prism Limited\Blue Prism Automate\AutomateC.exe" /importrelease
TestGit.bprelease /overwrite /user %1 %2 /dbconname %3
```

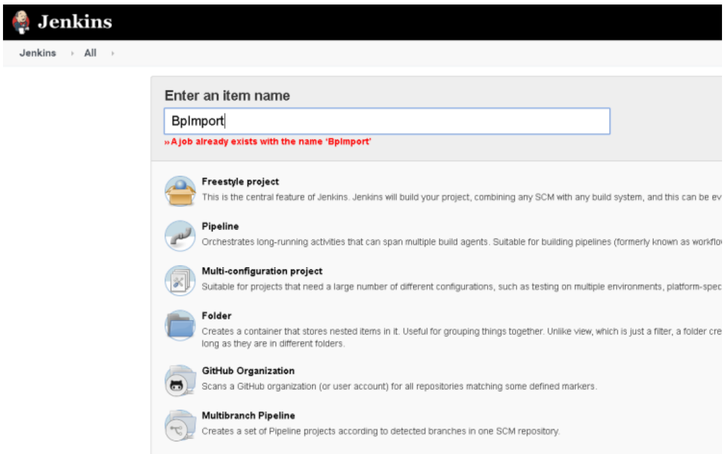
- User can have a single/multiple config file(s) to keep all the deployable artefacts. This point is not considered as part of this guide and is left upto the user to implement.

Jenkins Steps

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



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



Enter an item name

BpImport

» A job already exists with the name 'BpImport'

Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be ev

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflo

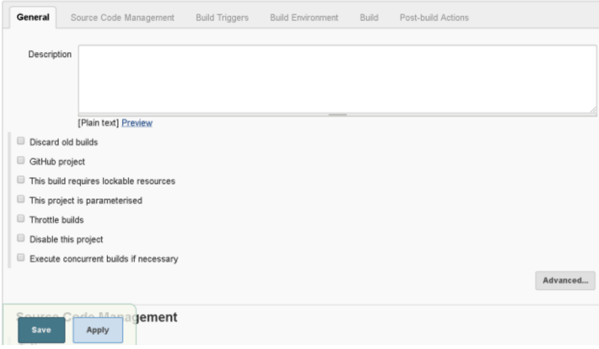
Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-spec

Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder cre long as they are in different folders.

GitHub Organization
Scans a GitHub organization (or user account) for all repositories matching some defined markers.

Multibranch Pipeline
Creates a set of Pipeline projects according to detected branches in one SCM repository.

- Below screen should open up after clicking on Ok.



☐ 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/ashz30/blueprism.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/ashz30/blueprism.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 -

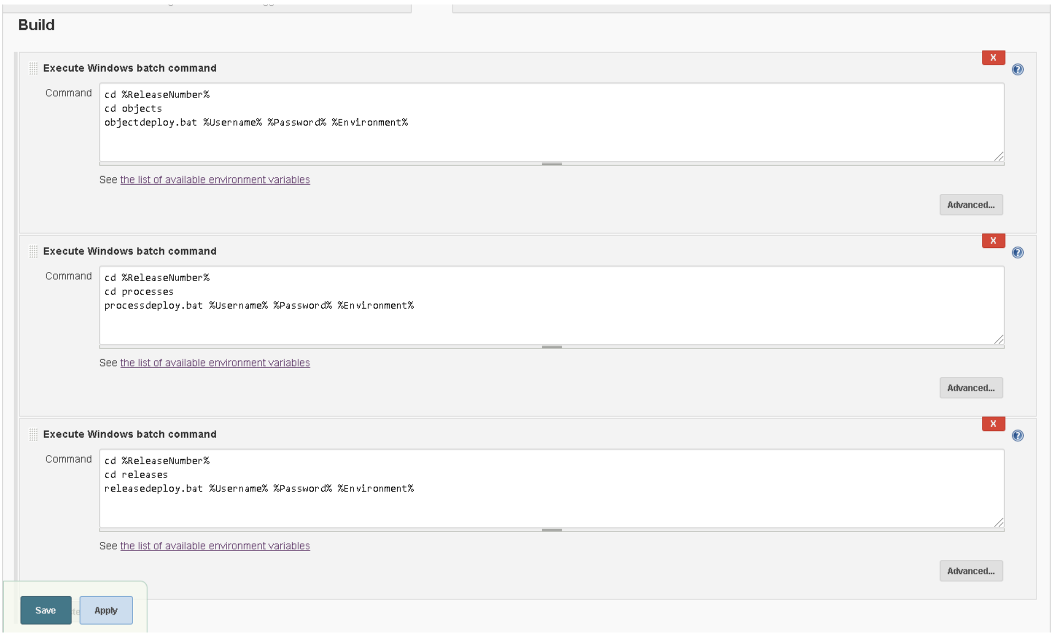
```
cd %ReleaseNumber%
cd processes
processdeploy.bat %Username% %Password% %Environment%
```

(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

Advanced...

Execute Windows batch command

Command

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

See the list of available environment variables

Advanced...

Execute Windows batch command

Command

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

See the list of available environment variables

Advanced...

Deployment Steps

- Add runtime deployment files to GIT (optional, needed if you are setting up your own repo)

  This step needs to be done when your building a repostory from scratch, for this guide the checkout from Git should set up all the files required from ashz30's repo. (you can copy them for your own use)

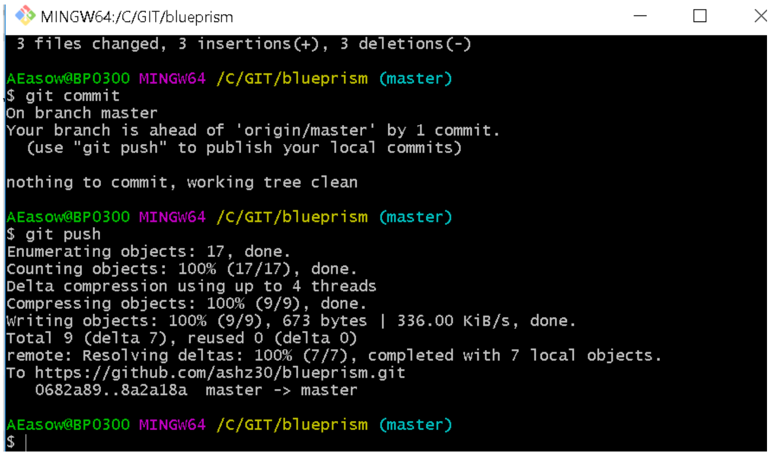
- Open GIT bash. And type the below commands (optional, needed if you are setting up your own repo):

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

  Type comment -> esc -> wq!

  ```
  $ git push
  ```

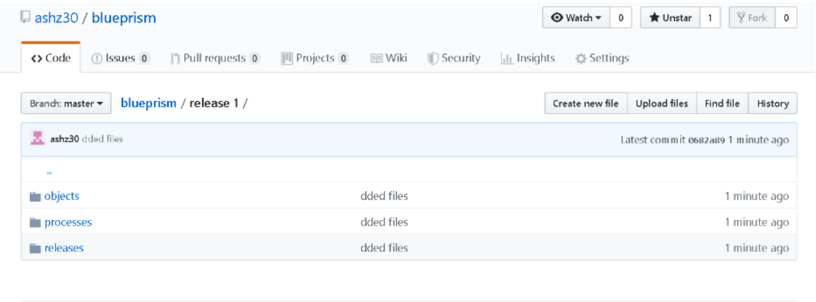
- Once the push command is issued, you should be able to see something similar to the screen below:



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 /

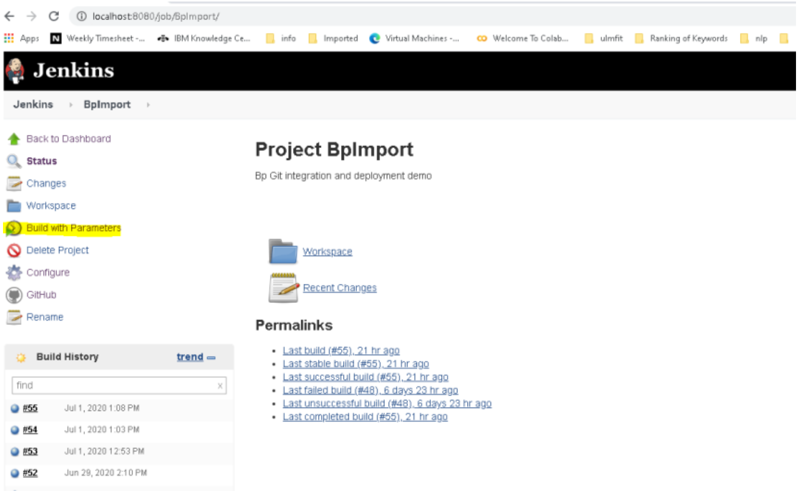
ashz30 dded files — Latest commit 0682a89 1 minute ago

..		
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 BpImport

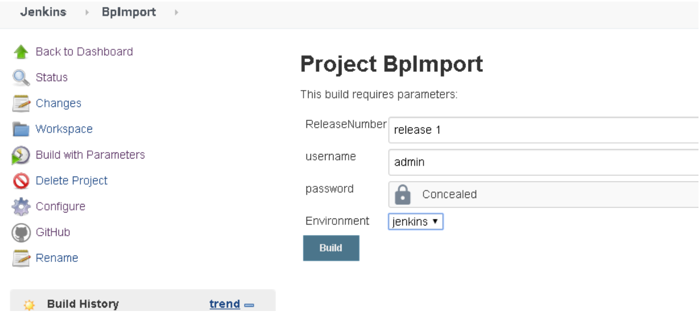
Bp Git integration and deployment demo

Permalinks

- Last build (#55), 21 hr ago
- Last stable build (#55), 21 hr ago
- Last successful build (#55), 21 hr ago
- Last failed build (#48), 6 days 23 hr ago
- Last unsuccessful build (#48), 6 days 23 hr ago
- Last completed build (#55), 21 hr ago

#55	Jul 1, 2020 1:08 PM
#54	Jul 1, 2020 1:03 PM
#53	Jul 1, 2020 12:53 PM
#52	Jun 29, 2020 2:10 PM

- Override any parameters required and click on build:



Project BpImport

This build requires parameters:

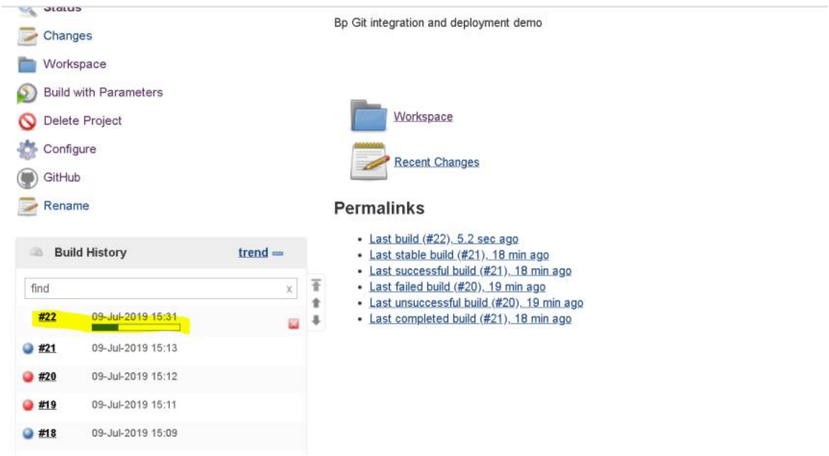
ReleaseNumber: release 1

username: admin

password: Concealed

Environment: jenkins

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



Bp Git integration and deployment demo

Build History trend

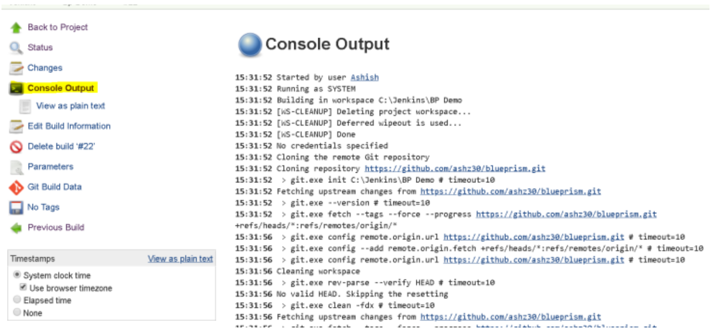
find

#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:



Timestamps View as plain text

● System clock time
☑ Use browser timezone
○ Elapsed time
○ None

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
```

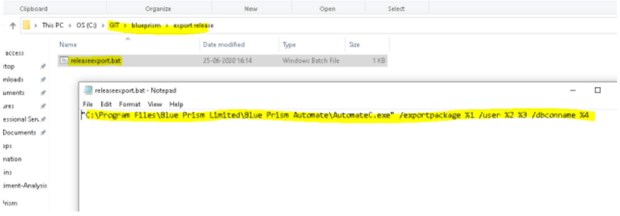
- The console output gives the status of your current build.
- Log into Blue Prism and check the assets deployed.
- Jenkins Job checked out the Blue Prism assets from the source code respository and deployed the assets into the environment chose without any human intervention.

Build Export project:

Code Repository Steps

- For this project, you would need to have your own GIT Repo in which you have commit, create branch and push rights.
- Export release bat file and folder contains the command required to do the export.

  Please take note of the below folder structure as this gets accessed by Jenkins during build.



Name	Date modified	Type	Size
releaseexport.bat	25-06-2020 16:14	Windows Batch File	1 KB

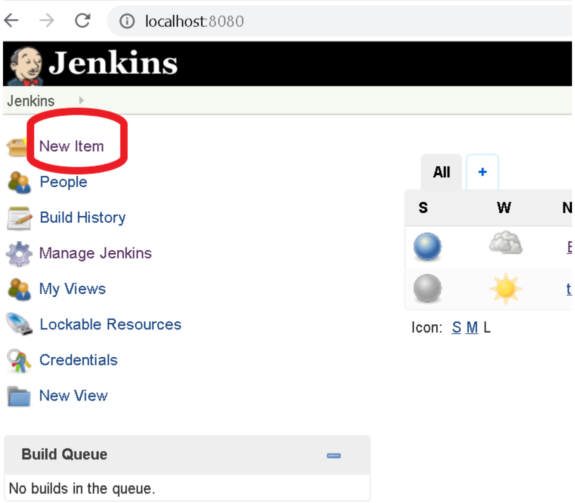
```
"C:\Program Files\Blue Prism Limited\Blue Prism Automate\AutomateC.exe" /exportpackage %1 /user %2 %3 /dbconname %4
```

"C:\Program Files\Blue Prism Limited\Blue Prism Automate\AutomateC.exe" /exportpackage %1 /user %2 %3 /dbconname %4

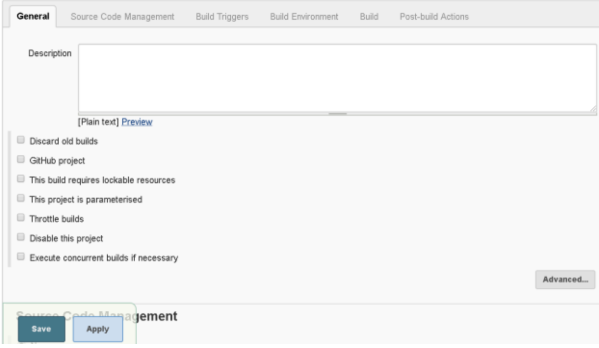
Users should get the required files when they pull the master branch when running the job for BPImport.

Jenkins Steps

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



- Give a name to the project (BPExport in this case), and Select Freestyle Project and Click on Ok.
- Below screen should open up after clicking on Ok.



☐ 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/ashz30/blueprism.git

This Project is Parameterised – Checked

Add Parameters –

ReleaseNumber, Value- release 2 (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)

PackageName – TestExport (needs to match the package created in Blue Prism)

Git_Branch – new_branch (this branch will be created in GIT)

Source Code management – GIT Selected

Repository url populated, Value- https://github.com/ashz30/blueprism.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 -

(4) Execute windows batch command -

```
git branch %Git_Branch%
git checkout %Git_Branch%
```

(5) Execute windows batch command -

```
mkdir "%ReleaseNumber%"
cd "%ReleaseNumber%"

Echo Starting
START CMD /c "../export release/releaseexport.bat" %PackageName% %username%
%password% %Environment%
Echo Done
```

Note: START CMD is used to call the releaseexport.bat file in GIT because on calling the plain export command or the bat file as 'CALL', it results in a strange context error. Blue Prism command line commands are tested in plain command prompt, so I couldn't solve the context issue, but running the command in a separate command window created through START works fine.

(6) Execute windows batch command -

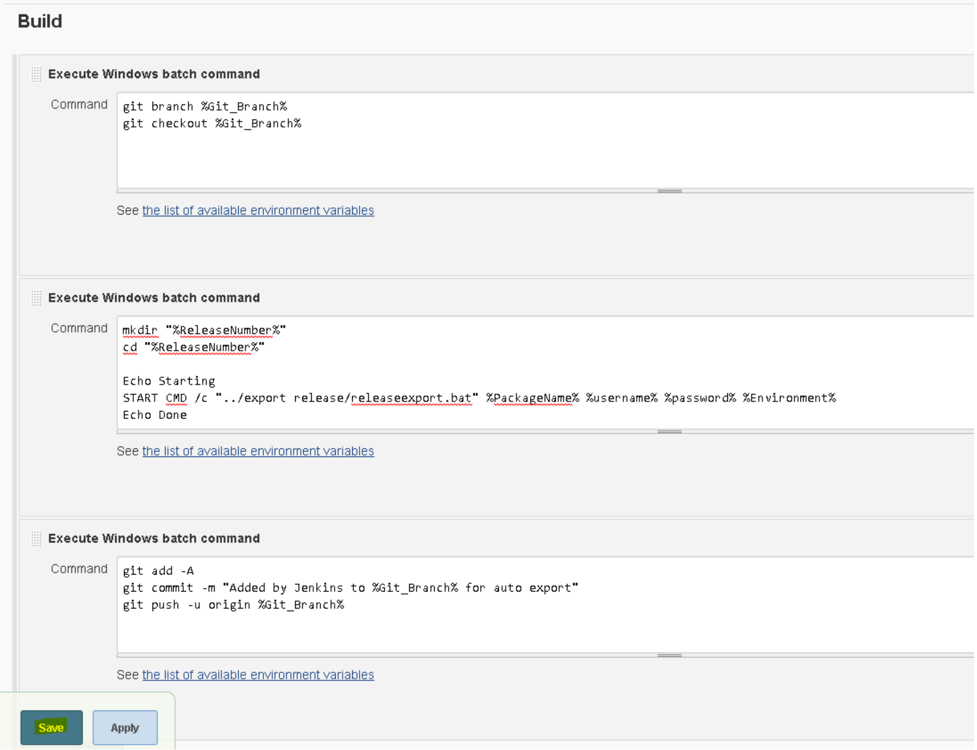
git add -A

git commit -m "Added by Jenkins to %Git_Branch% for auto export"

git push -u origin %Git_Branch%

(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
See the list of available environment variables
Execute Windows batch command
Command
See the list of available environment variables
Execute Windows batch command
Command
See the list of available environment variables
Save
Apply

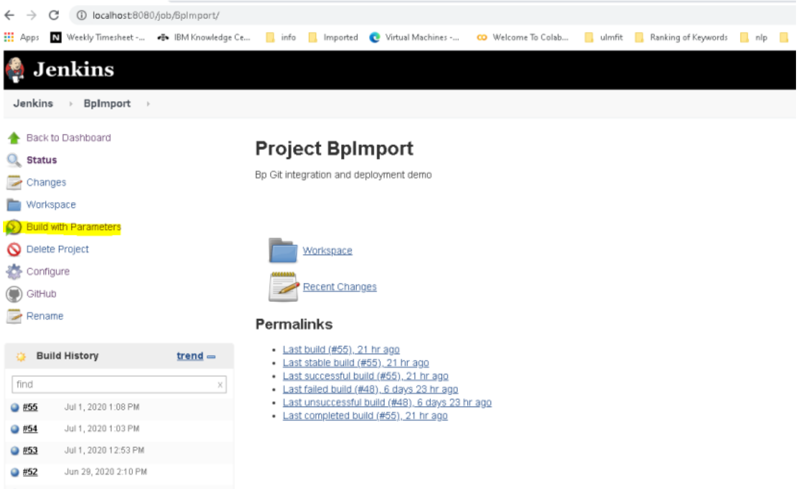
```
git branch %Git_Branch%
git checkout %Git_Branch%
```

```
mkdir "%ReleaseNumber%"
cd "%ReleaseNumber%"

Echo Starting
START CMD /c "../export release/releaseexport.bat" %PackageName% %username% %password% %Environment%
Echo Done
```

```
git add -A
git commit -m "Added by Jenkins to %Git_Branch% for auto export"
git push -u origin %Git_Branch%
```

Job Creation and Tracking

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



Project BpImport

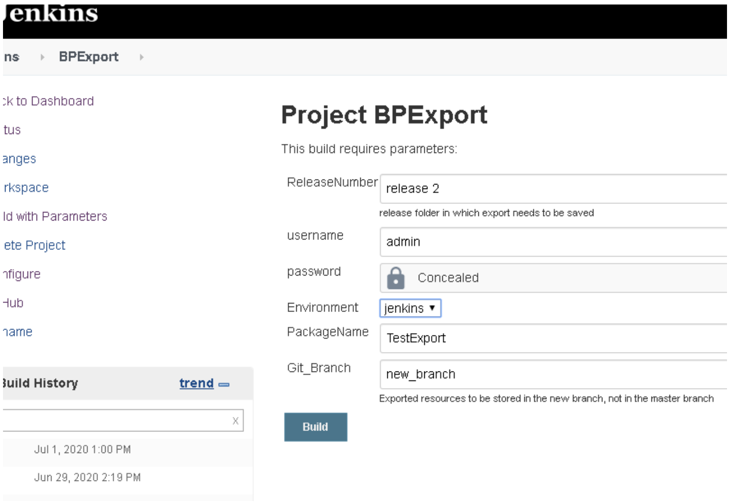
Bp Git integration and deployment demo

Permalinks

- Last build (#55), 21 hr ago
- Last stable build (#55), 21 hr ago
- Last successful build (#55), 21 hr ago
- Last failed build (#48), 6 days 23 hr ago
- Last unsuccessful build (#48), 6 days 23 hr ago
- Last completed build (#55), 21 hr ago

#55	Jul 1, 2020 1:08 PM
#54	Jul 1, 2020 1:03 PM
#53	Jul 1, 2020 12:53 PM
#52	Jun 29, 2020 2:10 PM

- Override any parameters required and click on build:



Project BPExport

This build requires parameters:

ReleaseNumber: release 2
release folder in which export needs to be saved

username: admin

password: Concealed

Environment: jenkins ▾

PackageName: TestExport

Git_Branch: new_branch
Exported resources to be stored in the new branch, not in the master branch

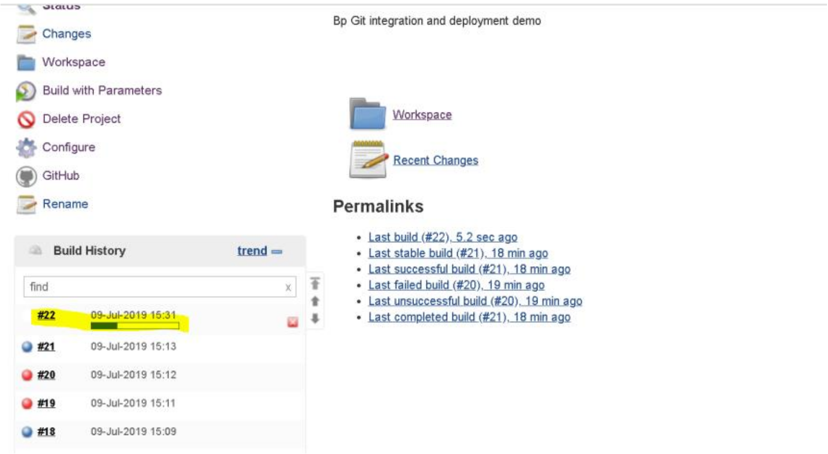
Build

Build History — trend

Jul 1, 2020 1:00 PM

Jun 29, 2020 2:19 PM

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



Bp Git integration and deployment demo

Build History — trend

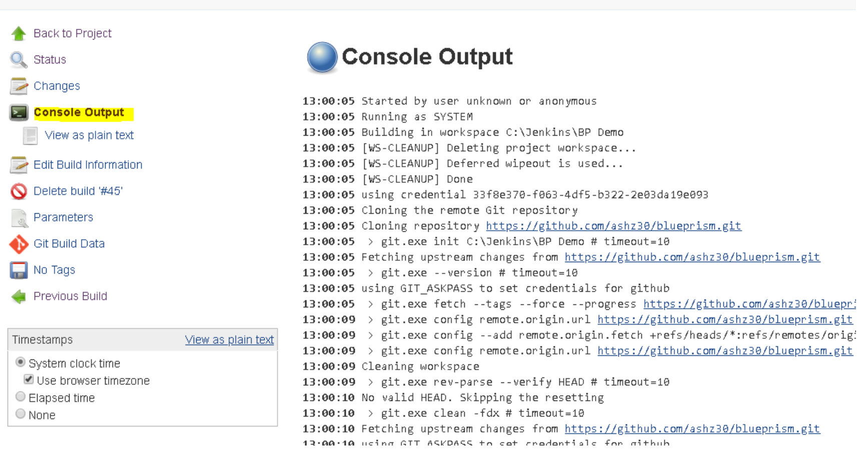
find

#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:



Timestamps — View as plain text

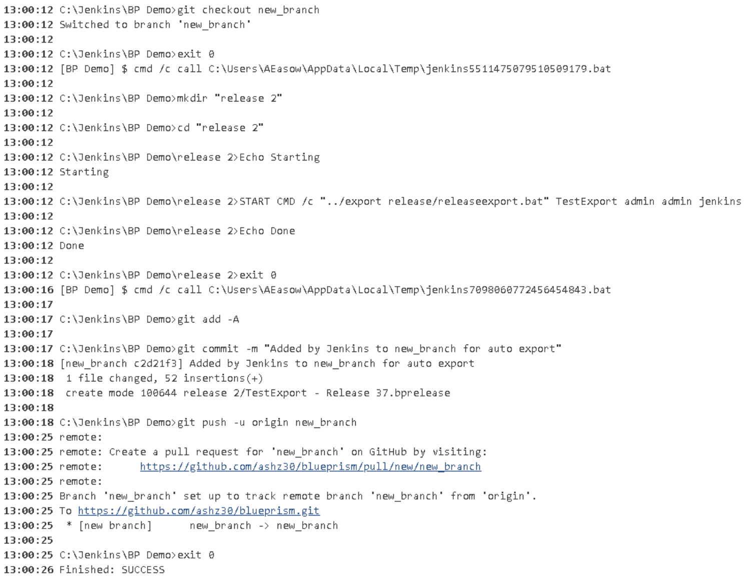
- ○ System clock time
- ☑ Use browser timezone
- ○ Elapsed time
- ○ None

Console Output

```
13:00:05 Started by user unknown or anonymous
13:00:05 Running as SYSTEM
13:00:05 Building in workspace C:\Jenkins\BP Demo
13:00:05 [WS-CLEANUP] Deleting project workspace...
13:00:05 [WS-CLEANUP] Deferred wipeout is used...
13:00:05 [WS-CLEANUP] Done
13:00:05 using credential 33f8e370-f063-4df5-b322-2e03da19e093
13:00:05 Cloning the remote Git repository
13:00:05 Cloning repository https://github.com/ashz30/blueprism.git
13:00:05  > git.exe init C:\Jenkins\BP Demo # timeout=10
13:00:05 Fetching upstream changes from https://github.com/ashz30/blueprism.git
13:00:05  > git.exe --version # timeout=10
13:00:05 using GIT_ASKPASS to set credentials for github
13:00:05  > git.exe fetch --tags --force --progress https://github.com/ashz30/bluepr
13:00:09  > git.exe config remote.origin.url https://github.com/ashz30/blueprism.git
13:00:09  > git.exe config --add remote.origin.fetch +refs/heads/*:refs/remotes/orig
13:00:09  > git.exe config remote.origin.url https://github.com/ashz30/blueprism.git
13:00:09 Cleaning workspace
13:00:09  > git.exe rev-parse --verify HEAD # timeout=10
13:00:10 No valid HEAD. Skipping the resetting
13:00:10  > git.exe clean -fdx # timeout=10
13:00:10 Fetching upstream changes from https://github.com/ashz30/blueprism.git
13:00:10 using GIT_ASKPASS to set credentials for github
```

The console output gives the status of your current build.

- Make sure in the console you can see the files being committed and the new branch getting created.



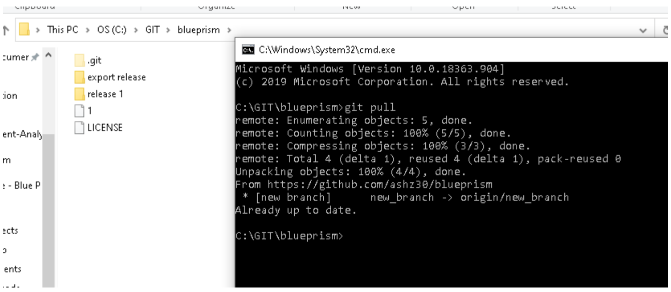
```
13:00:12 C:\Jenkins\BP Demo>git checkout new_branch
13:00:12 Switched to branch 'new_branch'
13:00:12
13:00:12 C:\Jenkins\BP Demo>exit 0
13:00:12 [BP Demo] $ cmd /c call C:\Users\AEasow\AppData\Local\Temp\jenkins5511475079510509179.bat
13:00:12
13:00:12 C:\Jenkins\BP Demo>mkdir "release 2"
13:00:12
13:00:12 C:\Jenkins\BP Demo>cd "release 2"
13:00:12
13:00:12 C:\Jenkins\BP Demo\release 2>Echo Starting
13:00:12 Starting
13:00:12
13:00:12 C:\Jenkins\BP Demo\release 2>START CMD /c "../export release/releaseexport.bat" TestExport admin admin jenkins
13:00:12
13:00:12 C:\Jenkins\BP Demo\release 2>Echo Done
13:00:12 Done
13:00:12
13:00:12 C:\Jenkins\BP Demo\release 2>exit 0
13:00:16 [BP Demo] $ cmd /c call C:\Users\AEasow\AppData\Local\Temp\jenkins7098060772456454843.bat
13:00:17
13:00:17 C:\Jenkins\BP Demo>git add -A
13:00:17
13:00:17 C:\Jenkins\BP Demo>git commit -m "Added by Jenkins to new_branch for auto export"
13:00:18 [new_branch c2d21f3] Added by Jenkins to new_branch for auto export
13:00:18  1 file changed, 52 insertions(+)
13:00:18  create mode 100644 release 2/TestExport - Release 37.bprelease
13:00:18
13:00:18 C:\Jenkins\BP Demo>git push -u origin new_branch
13:00:25 remote:
13:00:25 remote: Create a pull request for 'new_branch' on GitHub by visiting:
13:00:25 remote:      https://github.com/ashz30/blueprism/pull/new/new_branch
13:00:25 remote:
13:00:25 Branch 'new_branch' set up to track remote branch 'new_branch' from 'origin'.
13:00:25 To https://github.com/ashz30/blueprism.git
13:00:25  * [new branch]      new_branch -> new_branch
13:00:25
13:00:25 C:\Jenkins\BP Demo>exit 0
13:00:26 Finished: SUCCESS
```

- Goto your local GIT Repo in a command line, and run command

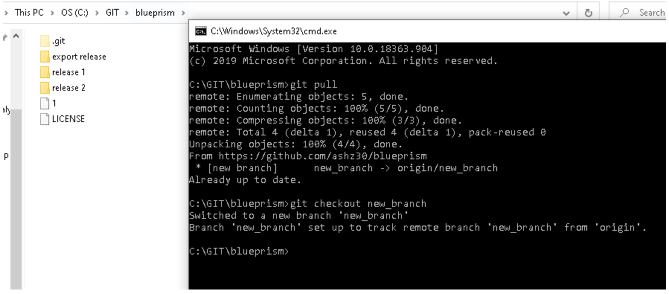
  >git pull

  >git checkout new_branch

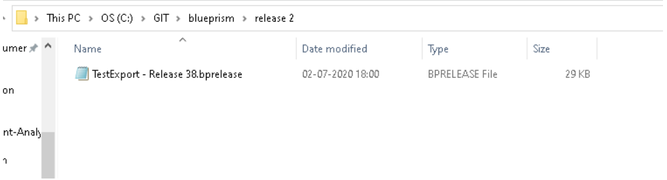
  You should be able to view your new branch comitted and the released which was checked in.


This PC › OS (C:) › GIT › blueprism ›
.git
export release
release 1
1
LICENSE
C:\Windows\System32\cmd.exe

```
Microsoft Windows [Version 10.0.18363.904]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\GIT\blueprism>git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 4 (delta 1), reused 4 (delta 1), pack-reused 0
Unpacking objects: 100% (4/4), done.
From https://github.com/ashz30/blueprism
 * [new branch]      new_branch -> origin/new_branch
Already up to date.

C:\GIT\blueprism>
```


This PC › OS (C:) › GIT › blueprism ›
Search
.git
export release
release 1
release 2
1
LICENSE
C:\Windows\System32\cmd.exe

```
Microsoft Windows [Version 10.0.18363.904]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\GIT\blueprism>git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 4 (delta 1), reused 4 (delta 1), pack-reused 0
Unpacking objects: 100% (4/4), done.
From https://github.com/ashz30/blueprism
 * [new branch]      new_branch -> origin/new_branch
Already up to date.

C:\GIT\blueprism>git checkout new_branch
Switched to a new branch 'new_branch'
Branch 'new_branch' set up to track remote branch 'new_branch' from 'origin'.

C:\GIT\blueprism>
```



Name	Date modified	Type	Size
TestExport - Release 38.bprelease	02-07-2020 18:00	BPRELEASE File	29 KB

- Jenkins job ran the autoexport scripts and checked in the release without any human intervention into into the source code repository.

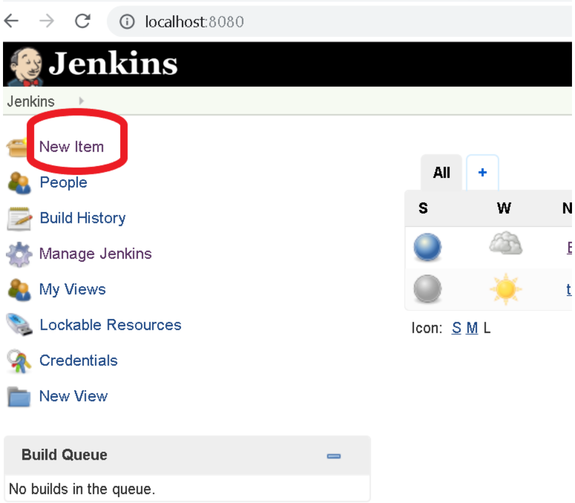
Build Approval project:

Pre-Requisite steps

- For this demo, you would need to have set up BPImport Project, and pipeline plugin needs to be installed in Jenkins.

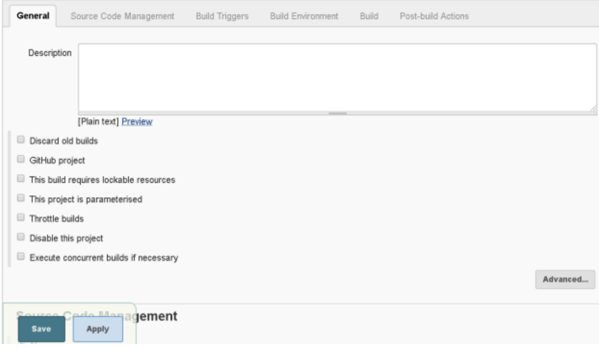
Jenkins Steps

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



- Give a name to the project (BuildApproval in this case), and Select **Pipeline** Project and Click on Ok.

- Below screen should open up after clicking on Ok.



Description

[Plain text] Preview

□ 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/ashz30/blueprism.git

This Project is Parameterised – Checked

Add Parameters (these need to be passed) –

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

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

JobName, Value- BPImport (or what the user provides for the build)

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

*Password is not given here, so that password can be picked up from the job which needs to be run i.e. BpImport (same can be done for username)

Source Code management – GIT Selected

Repository url populated, Value- https://github.com/ashz30/blueprism.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 –

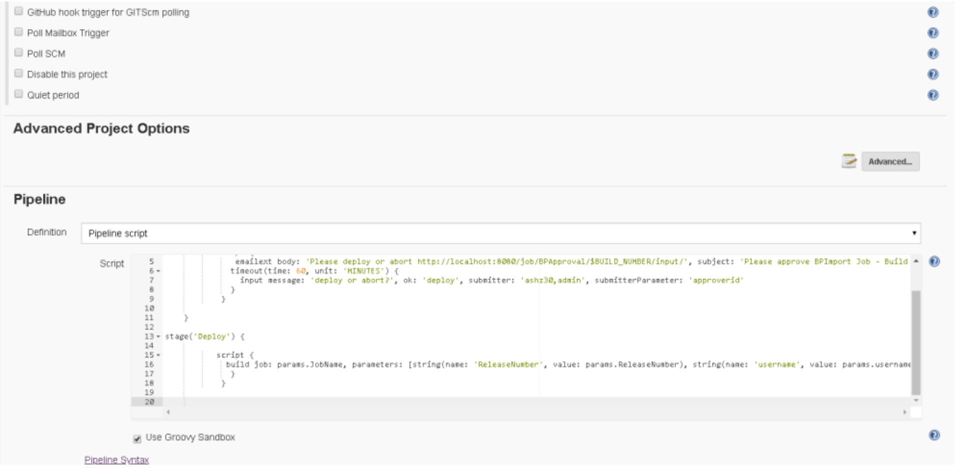
Delete workspace Before build starts selected

Add timestamps to the console output selected

Pipeline -

(7) Pipeline script -

```
stage('build') {
      script {
            emailext body: 'Please deploy or abort http://localhost:8080/job/BPApproval/$BUILD_NUMBER/input/', subject:
  'Please approve BPImport Job - Build no: $BUILD_NUMBER', to: '<email here for approval>'
          timeout(time: 60, unit: 'MINUTES') {
            input message: 'deploy or abort?', ok: 'deploy', submitter: 'ashz30,admin', submitterParameter: 'approverid'
          }
        }
  }
stage('Deploy') {
      script {
           build job: params.JobName, parameters: [string(name: 'ReleaseNumber', value: params.ReleaseNumber),
  string(name: 'username', value: params.username), string(name: 'Environment', value: params.Environment)]
          }
        }
```



Definition: Pipeline script

Script:

```
             emailext body: 'Please deploy or abort http://localhost:8080/job/BPApproval/$BUILD_NUMBER/input/', subject: 'Please approve BPImport Job - Build
            timeout(time: 60, unit: 'MINUTES') {
              input message: 'deploy or abort?', ok: 'deploy', submitter: 'ashz30,admin', submitterParameter: 'approverid'
            }
          }

    }

stage('Deploy') {

          script {
            build job: params.JobName, parameters: [string(name: 'ReleaseNumber', value: params.ReleaseNumber), string(name: 'username', value: params.username
            }
          }

```

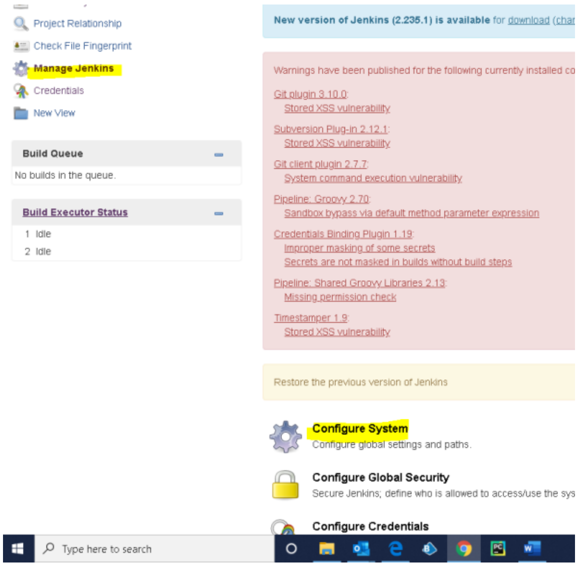
☑ Use Groovy Sandbox

Pipeline Syntax

- Save the Job once done.

- **Jenkins System set up** :

  Navigate to – Home Page –> Manage Jenkins - > Configure System.



New version of Jenkins (2.235.1) is available for download (char

Warnings have been published for the following currently installed co

Git plugin 3.10.0:
  Stored XSS vulnerability
Subversion Plug-in 2.12.1:
  Stored XSS vulnerability
Git client plugin 2.7.7:
  System command execution vulnerability
Pipeline: Groovy 2.70:
  Sandbox bypass via default method parameter expression
Credentials Binding Plugin 1.19:
  Improper masking of some secrets
  Secrets are not masked in builds without build steps
Pipeline: Shared Groovy Libraries 2.13:
  Missing permission check
Timestamper 1.9:
  Stored XSS vulnerability

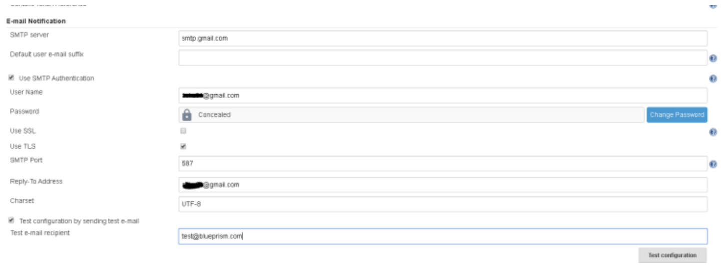
Restore the previous version of Jenkins

Configure System
Configure global settings and paths.

Configure Global Security
Secure Jenkins; define who is allowed to access/use the sys

Configure Credentials

- Scroll down all the way to Email Notification.



E-mail Notification

SMTP server	smtp.gmail.com
Default user e-mail suffix	
☑ Use SMTP Authentication	
User Name	[redacted]@gmail.com
Password	Concealed
Use SSL	□
Use TLS	☑
SMTP Port	587
Reply-To Address	[redacted]@gmail.com
Charset	UTF-8
☑ Test configuration by sending test e-mail	
Test e-mail recipient	test@blueprism.com

- Settings:
  - Smtp server – smtp.gmail.com
  - User smtp authentication – checked
  - Username – outgoing email id
  - Password - *****

- User TLS – checked
- Smtp port – 587 (for gmail)
- Reply to address – Outgoing email id
- Charset – UTF-8
- Test the configuration if required (once full set up is done)

- For tls to work, I had to add certificate for smtp.gmail.com to C:\Program Files (x86)\Jenkins\jre\lib\security\cacerts by keytool (for gmail, the steps differ for each mail domain)

  Links - https://support.google.com/a/answer/6180220?hl=en

  Copy and save the lines between "-----BEGIN CERTIFICATE-----" and "-----END CERTIFICATE-----" into a file, say, gmail.cert, copy to location - C:\Program Files (x86)\Jenkins\jre\lib\security

  Goto location C:\Program Files (x86)\Jenkins\jre\lib\security in command line. And run below command (or equivalent)

  “C:\Program Files (x86)\Jenkins\jre\bin\keytool” -import -alias gmail -keystore “C:\Program Files (x86)\Jenkins\jre\lib\security\cacerts" -file gmail.cert

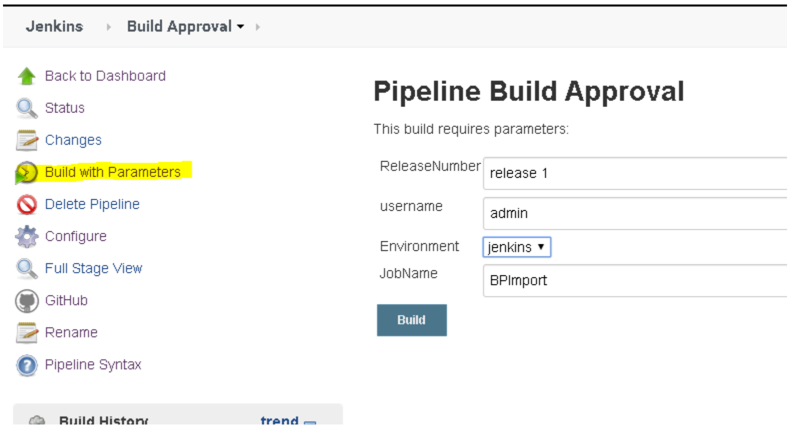
  Note: your need to provide the password to access the keystore. The password for the default java keystore is changeit

  Answer Yes when it ask "Trust this certificate? [no]: yes"

  Test configuration earlier should now work.

Job Creation and Tracking

- For creating a runtime Job, log into Jenkins, click on BuildApproval (the project which has been configured). And click on Build with parameters option on the left.
- Override any parameters required and click on build:

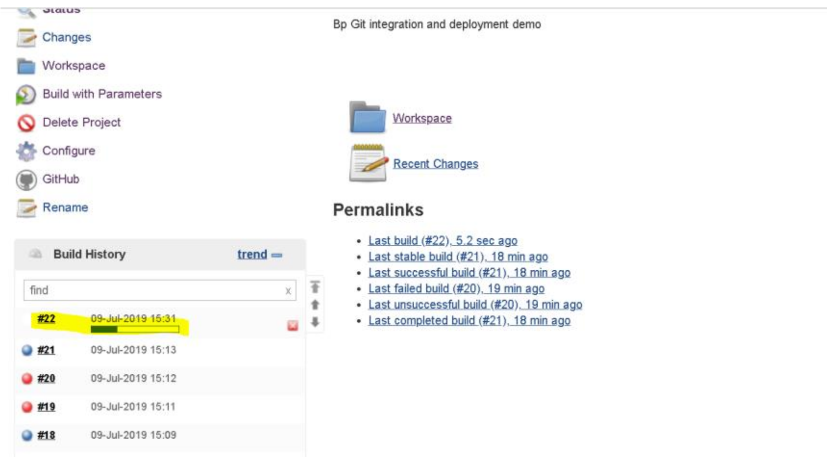


Pipeline Build Approval

This build requires parameters:

ReleaseNumber	release 1
username	admin
Environment	jenkins ▾
JobName	BPImport

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



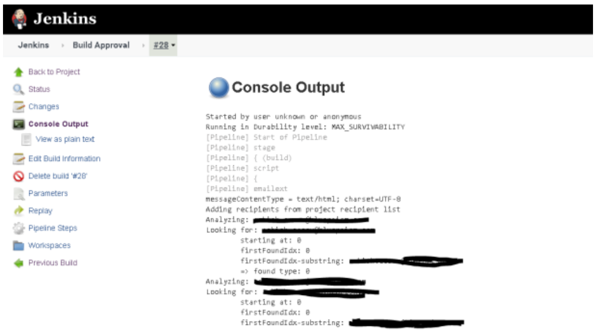
Bp Git integration and deployment demo

Build	Date
#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:

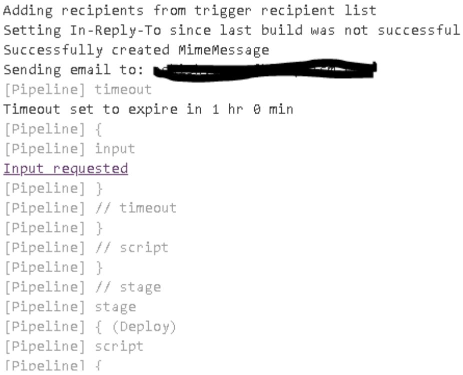


Console Output

```
Started by user unknown or anonymous
Running in Durability level: MAX_SURVIVABILITY
[Pipeline] Start of Pipeline
[Pipeline] stage
[Pipeline] { (build)
[Pipeline] script
[Pipeline] {
[Pipeline] emailext
messageContentType = text/html; charset=UTF-8
Adding recipients from project recipient list
Analyzing: [redacted]
Looking for: [redacted]
        starting at: 0
        firstFoundIdx: 0
        firstFoundIdx-substring: [redacted]
        => found type: 0
Analyzing: [redacted]
Looking for: [redacted]
        starting at: 0
        firstFoundIdx: 0
        firstFoundIdx-substring: [redacted]
```

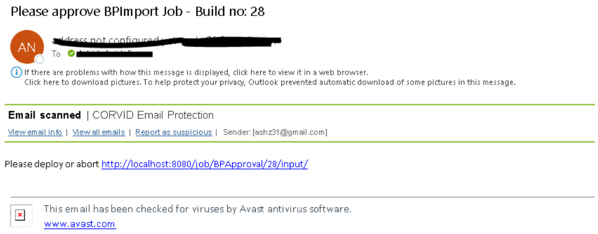
The console output gives the status of your current build.

- Once it reaches the stage, you should get an email in the id you configured.



```
Adding recipients from trigger recipient list
Setting In-Reply-To since last build was not successful
Successfully created MimeMessage
Sending email to: [redacted]
[Pipeline] timeout
Timeout set to expire in 1 hr 0 min
[Pipeline] {
[Pipeline] input
Input requested
[Pipeline] }
[Pipeline] // timeout
[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Deploy)
[Pipeline] script
[Pipeline] {
```

- Check the email, and navigate to the link to approved.



Please approve BPImport Job - Build no: 28

AN [redacted]

To [redacted]

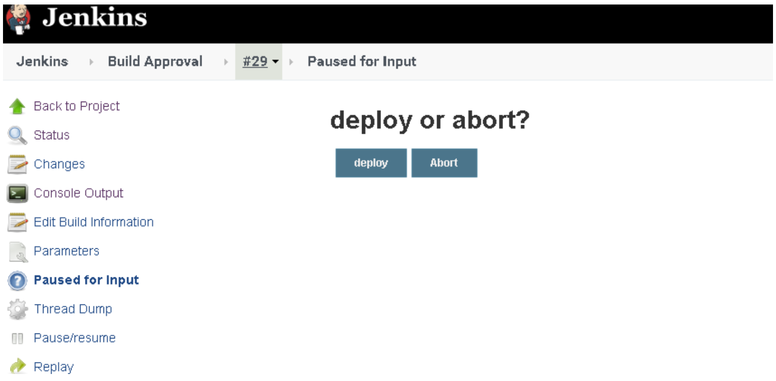
If there are problems with how this message is displayed, click here to view it in a web browser.
Click here to download pictures. To help protect your privacy, Outlook prevented automatic download of some pictures in this message.

Email scanned | CORVID Email Protection

View email info | View all emails | Report as suspicious | Sender: [ashz31@gmail.com]

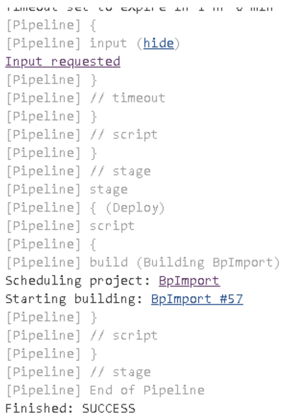
Please deploy or abort http://localhost:8080/job/BPApproval/28/input/

This email has been checked for viruses by Avast antivirus software.
www.avast.com



deploy or abort?

If approved (deploy), the build should call BPImport and succesfully complete the called Job. You can view the status of Blue Prism studio, to check status of Assets in BPImport which was deployed.



```
Timeout set to expire in 1 hr 0 min
[Pipeline] {
[Pipeline] input (hide)
Input requested
[Pipeline] }
[Pipeline] // timeout
[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Deploy)
[Pipeline] script
[Pipeline] {
[Pipeline] build (Building BpImport)
Scheduling project: BpImport
Starting building: BpImport #57
[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] // stage
[Pipeline] End of Pipeline
Finished: SUCCESS
```

- Jenkins job asked for Approval from a pre-configured email id, and once the approval was obtained, ran the job automatically without any human intervention.

Conclusion and Additional configuration

The current guide provides modular pieces of functionality and not a full use case in one go as I believe each customer requirement is different with different use cases, so its best to use these modular functionalities provided to build a larger use case as per those requirements.

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.

There are multiple plugins available which also provide static scanning functionality (SonarQube, Fortify etc), these can be incorporated in the build process for further automation of deployments.

Users are encouraged to explore.