

Preparing and planning for the Outage Window in CI/CD Environment

1. We need to restart Jenkins regularly to upgrade Jenkins and its plugins.
2. We need to patch Linux regularly. Since our Jenkins is installed on Linux, we often need to restart Jenkins, which has jobs running.
3. Determine the maintenance upgrade you want to perform on the Jenkins server. (plugins or Jenkins server, JVM, OpenJDK)
4. Estimate how much time the maintenance of the selected plugins task will take.
5. If the maintenance will take more than 4 hours, it is advisable to do it in small increments.
6. Keep in mind that you aim to upgrade/restart the process as seamlessly as possible with very minimal interruption to the daily activities in the company.
7. Collaborate with the production, QA, and Dev team to select the best time for the outage window to allow the upgrade of the plugins.
8. The best time should be selected so that the outage window will not affect the deployment in production.
9. After determining the window shut down period, document the information and communicate with the team members throughout the update period.
10. Start the upgrading/downgrading process.

Upgrading /Downgrading Plugins

Procedures you should follow

1. Back up `JENKINS_HOME` (locally or to s3 bucket).
2. SafeRestart
The `safeRestart` function tells Jenkins what to do before restarting. It is built in Jenkins core (just navigate to `/safeRestart`); this will prevent any new builds from starting and will allow existing builds to complete before Jenkins restarts.

Tools and Actions



Reload Configuration from Disk

Discard all the loaded data in memory and reload everything from file system. Useful when you modified config files directly on disk.



Jenkins CLI

Access/manage Jenkins from your shell, or from your script.



Script Console

Executes arbitrary script for administration/trouble-shooting/diagnostics.



Update shutdown preparation

Jenkins is currently shutting down. New builds are not executing.

Add the reason for safe shutdown.

Jenkins is going to shut down
Shut down reason: upgrading the plugins



Prepare for Shutdown

Jenkins is currently shutting down. New builds are not executing.

Reason

Update reason

Cancel Shutdown

This can also be accomplished through the groovy.init call:

Go ➡

Tools and Actions



Reload Configuration from Disk

Discard all the loaded data in memory and reload everything from file system. Useful when you modified config files directly on disk.



Jenkins CLI

Access/manage Jenkins from your shell, or from your script.



Script Console

Executes arbitrary script for administration/trouble-shooting/diagnostics.



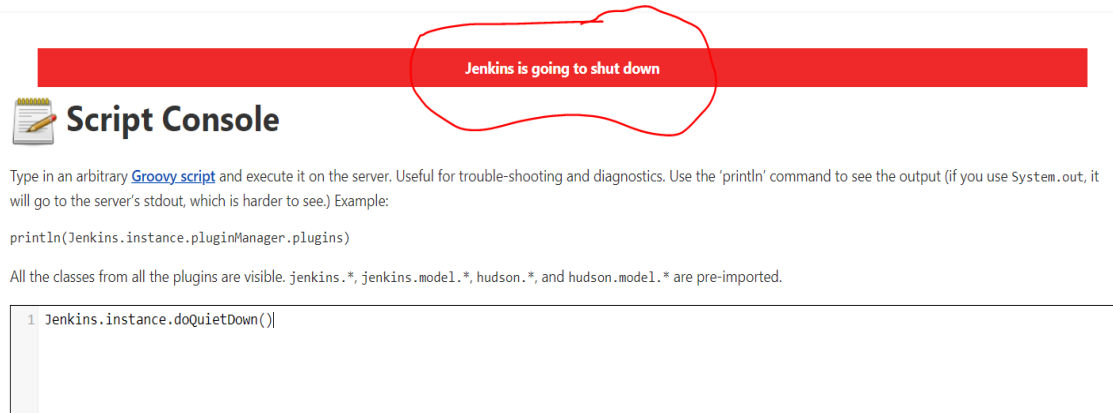
Update shutdown preparation

Jenkins is currently shutting down. New builds are not executing.

Preventing jobs from running but still leaves them in the queue.

To stop them from running, I would prefer a groovy.init call:

```
#Jenkins.instance.doQuietDown()
```



Jenkins is going to shut down

Script Console

Type in an arbitrary [Groovy script](#) and execute it on the server. Useful for trouble-shooting and diagnostics. Use the 'println' command to see the output (if you use System.out, it will go to the server's stdout, which is harder to see.) Example:

```
println(Jenkins.instance.pluginManager.plugins)
```

All the classes from all the plugins are visible. `jenkins.*`, `jenkins.model.*`, `hudson.*`, and `hudson.model.*` are pre-imported.

```
1 Jenkins.instance.doQuietDown()
```

This is not the best way to do it because it does not give a reason why we are shutting Jenkins.

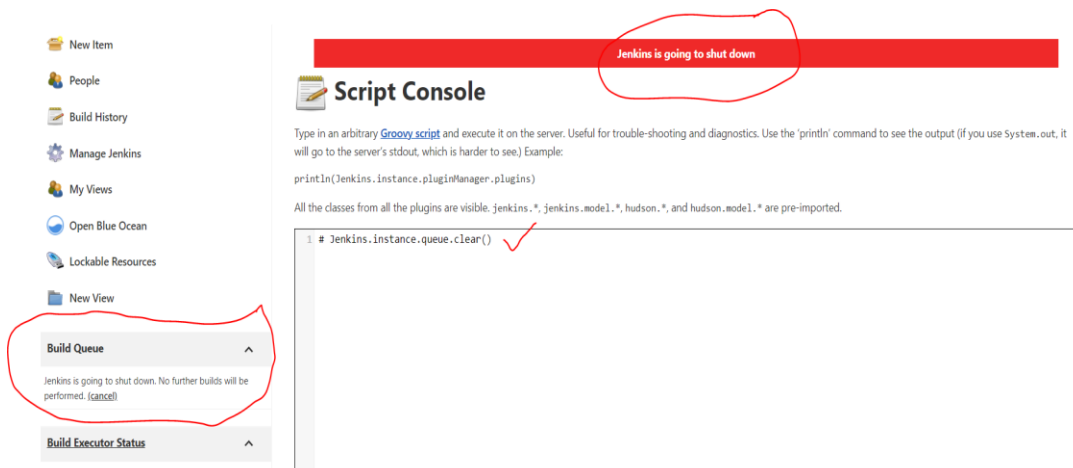
However, you can still accomplish same results as with safe shutdown.

Finally you can also use groovy.init call to clear any job that is running or queued in the pipeline.

Run

just also execute:

```
# Jenkins.instance.queue.clear()
```



Build Queue

Jenkins is going to shut down. No further builds will be performed. (cancel)

Script Console

Type in an arbitrary [Groovy script](#) and execute it on the server. Useful for trouble-shooting and diagnostics. Use the 'println' command to see the output (if you use System.out, it will go to the server's stdout, which is harder to see.) Example:

```
println(Jenkins.instance.pluginManager.plugins)
```

All the classes from all the plugins are visible. `jenkins.*`, `jenkins.model.*`, `hudson.*`, and `hudson.model.*` are pre-imported.

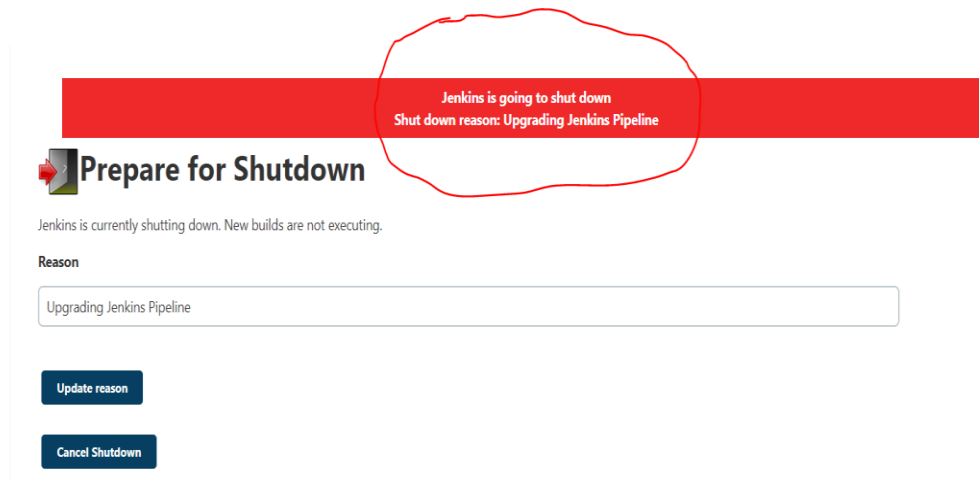
```
1 # Jenkins.instance.queue.clear()
```

Again!!

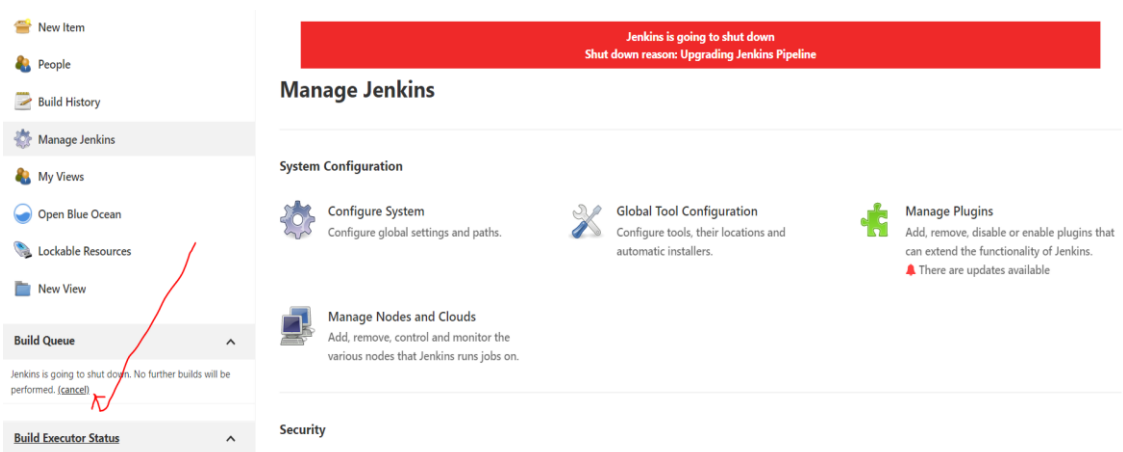
This is not the best way to do it because it does not give a reason why we shutting Jenkins.

However, you can still accomplish same results as with safe shutdown.

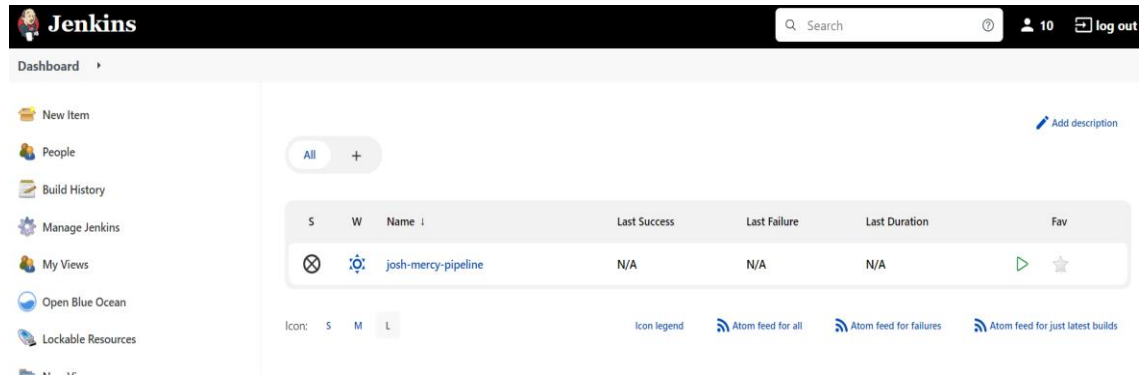
3. Start the process of upgrading and downgrading the legacy plugins
4. Upgrade and down grade the required plugins.
5. When completed, cancel the shutdown message



To do so ➔ look for Cancel as indicate in the image below and remove it



In the Jenkins server, the shut down message will disappear



6. Make a second backup of **JENKINS_HOME** after upgrading Jenkins required plugins.

Goodluck !!!