## Pipeline Best Practices :

1. Keep it simple: Use the minimum amount of code to connect the Pipeline steps and integrate tools. Delegate most activity to agents and reduce the load on controllers.
2. Use external scripts and tools for most tasks, especially those involving complex processing or high CPU usage.
3. Use steps from Jenkins plugins for source control, artifact management, deployment systems, and system automation.
4. Create small, variable files instead of large, global declaration files to save memory.
   1. Avoiding large global variable declaration files
   2. Avoiding very large, shared libraries
5. Reduce the number of steps in the Pipeline, Consolidate multiple sequential steps into external scripts to minimize steps.
6. Combine Pipeline steps to reduce overhead. Use a single shell step for multiple commands instead of several echo or sh steps.
7. Use Declarative syntax instead of Scripted syntax for better readability and performance.
8. Implement shared libraries to avoid the need for custom scripts inside a Declarative Pipeline.
9. Avoiding complex Groovy code in Pipelines
   1. Avoid using Groovy methods like JsonSlurper and HttpRequest to read and process large files or make external HTTP requests.
   2. Use less cyclomatic complex code , avoid too nested If, else, for , switch.
10. Use @NonCPS annotation only when necessary for methods incompatible with CPS transformation.
    1. If necessary, you can use the @NonCPS annotation to disable the CPS transformation for a specific method whose body would not execute correctly if it were CPS-transformed. Just be aware that this also means the Groovy function will have to restart completely since it is not transformed.
11. Acquire agents within parallel steps to maximize processing capacity.
12. Ensure all processing occurs within agents rather than the Jenkins controller
13. Wrap input steps in timeouts to handle potential delays gracefully.
14. Cleaning up old Jenkins builds
    1. As a Jenkins administrator, removing old or unwanted builds keeps the Jenkins controller running efficiently. When you do not remove older builds, there are less resources for more current and relevant builds. This video reviews using the [buildDiscarder](https://www.jenkins.io/doc/book/pipeline/syntax/" \l "options) directive in individual Pipeline jobs