**1) Create a workflow in GitHub Actions to run a Super-Linter, (linter.yml)**

It is a simple combination of various linters, written in bash, to help validate your source code.  
  
The objective of this tool is to,

A) Prevent broken code from being uploaded to the default branch (Usually master)  
B) Help establish coding best practices across multiple languages  
C) Build guidelines for code layout and format  
D) Automate the process to help streamline code reviews

**2) Create a Telegram workflow in GitHub Actions, (push-notify.yml & issues-notify.yml)**

Whenever there is a commit push to the repo, there will be a message send to the Telegram Bot.

**3) Create a Disabling linters and Rules workflow in GitHub Actions, (.ansible-lint.yml & .stylelintrc.json)**

Linters can often require additional configuration to ensure they work with the codebase and coding style, to avoid flagging false-positives.

The GitHub Super-Linter has set up some default configurations for each linter which should work reasonably well with common code bases, but many of the linters can be configured to disable certain rules or configure the rules to ignore certain pieces of codes.

.github/linters, the files in this folder are for the linters that will run against the code base.   
I have chosen to copy these, .ansible-lint.yml & .stylelintrc.json, into local repository in the directory: .github/linters will be used at runtime.   
If these are not present, they will be used by default in the linter run.

The files will be parsed at run time on the local branch to load all rules needed to run the Super-Linter GitHub Action.   
The GitHub Action will inform the user via the Checks API on the status and success of the process.