HOWTO make changes in SONiC and commit

This document explains the basic points to ponder while making changes in SONiC and explains the steps to make changes.

Points to Ponder

Knowing your repositories

SONiC shall be enhanced continuosly to add new features, to enhance existing features and to fix the bugs in the existing code. To make the changes in SONiC, developers need to understand the different repositories present in SONiC. Some of the repositories are given below. For more detailed information about repositories click here.

1. sonic-buildimage
2. sonic-swss - Switch State Service
3. sonic-swss-common Common library for Switch State Service
4. sonic-sairedis - SAI objects in Redis
5. sonic-dbsyncd - SONiC Switch State Service sync daemon

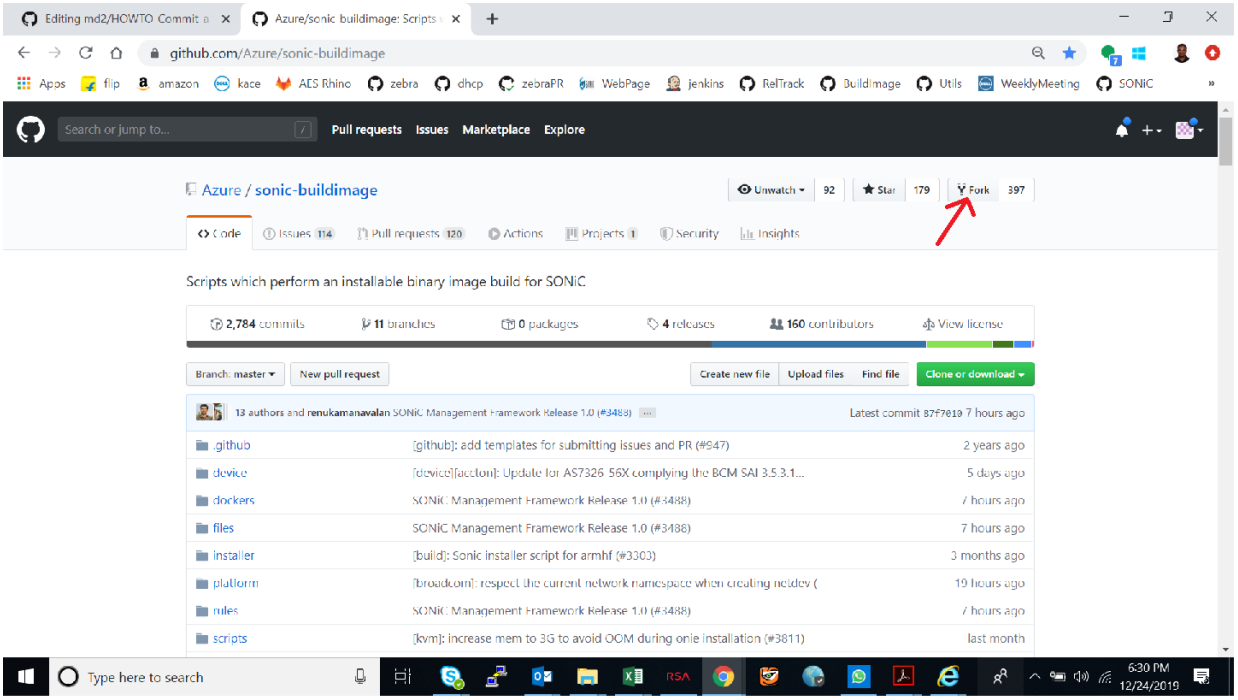
Procedure to make changes in SONiC & commit a code

Developers are requested to follow Agile development method, make changes in steps and raise separate pull request for each such test able change. Support if a feature/enhancement/bugfix involves multiple small sub-features, separate pull-request should be raised for each such test able sub-feature. Similarly, if the change involves multiple repositories and if the changes are related, do either one of the following.

1. Raise a Pull Request (PR1)on the repository (repo1) that does not have dependency on other repository (repo2) and wait for the PR to be merged. Then raise the Pull Request (PR2) on the other repository (repo2). (OR)
2. Raise both the pull requests; explicitly explain the dependency in the PRs and explain the order in which the PRs has to be merged.

Various steps related to making code changes and raising pull request is explained below.

1. **Fork the repository** Go to the master repository and click on "fork" for the repository. Refer the following picture on how to fork "sonic-buildimage".



If you had already forked the repository for previous changes, then either rebase your old forked repo or delete the old forked repo (go to your gi& create a new fork from master.

NOTE1: If you do not have an account with github, create one. All changes can be made only using your_github_username. NOTE2: To delete an old forked repo, go to https://github.com/<your_github_username>/, click repositories, look for your forked repo (example is: https://github.com/<your_github_username/sonic-buildimage will your forked sonic-buildimage repo), click settings, go to DangerZone and delete the repo.

2. **Clone the forked repository**

   Once if a repo is forked, all the changes should happen on the forked repo.

   a. Go to your list of repositories and look for the forked repository (https://github.com/<your_github_username>/, click repositorie to find it) and click the repo.

   b. Click on the green button "clone or download". Copy the git clone link from the new inline tab that opens.

c. Use the command "`git clone <clone_link>`" on your Linux server to clone the repository. Example : *git clone https://github.com/<your_github_username>/sonic-buildimage.git*

3. **Create a new branch**
   Each test-able sub-feature/enhancement/bugfix should happen in a separate branch within your repo. To create a branch, use the command "git-checkout". This command is used to switch to working tree files of existing branch or creates new branch using "-b". Specifying -b causes a new branch to be created.

`git checkout -b <branch_name>` Example : *git checkout -b SNMPAgent_cli_support*. This creates a branch named "SNMPAgent_cli_support" where you can make changes for SNMPAgent CLI.

4. **Git push origin**
   Next step is to specify to which remote "origin", the code changes need to be pushed upstream. The most common use of git push is to push your local changes to your public upstream repository. Upstream is a remote named “origin” (the default remote name if your repository is a clone) and the default branch to be updated to/from is named “master”. In order to push changes into your branch, do the following.

`git push origin <branch_name>` Example : *git push origin SNMPAgent_cli_support*. This sets the remote origin to the branch named "SNMPAgent_cli_support".

5. **Making changes** Once if the origin is set, you can make changes in the code. You can also add new files in this repo and remove unwanted files from the repo. NOTE1: In general, it is suggested to keep two directories in your environment, one for check-in process and other for compilation & testing. It is advised to make the changed in your compilation environment and complete the required testing before starting this check-in process. As part of this step, merge or copy your changes from compilation environment to this "check-in environment". This way, you can use the "git status" command to find the list of changes that you had done which will not include the unwanted generated files that are visible in compilation environment.
6. **Verify the changes made** After making changes, check the status of the files in git and verify your changes using the following commands.

`git status` - This command lists the files that are modified in the repo. `git diff` - This command shows the difference between the remote branch and the locally modified branch.

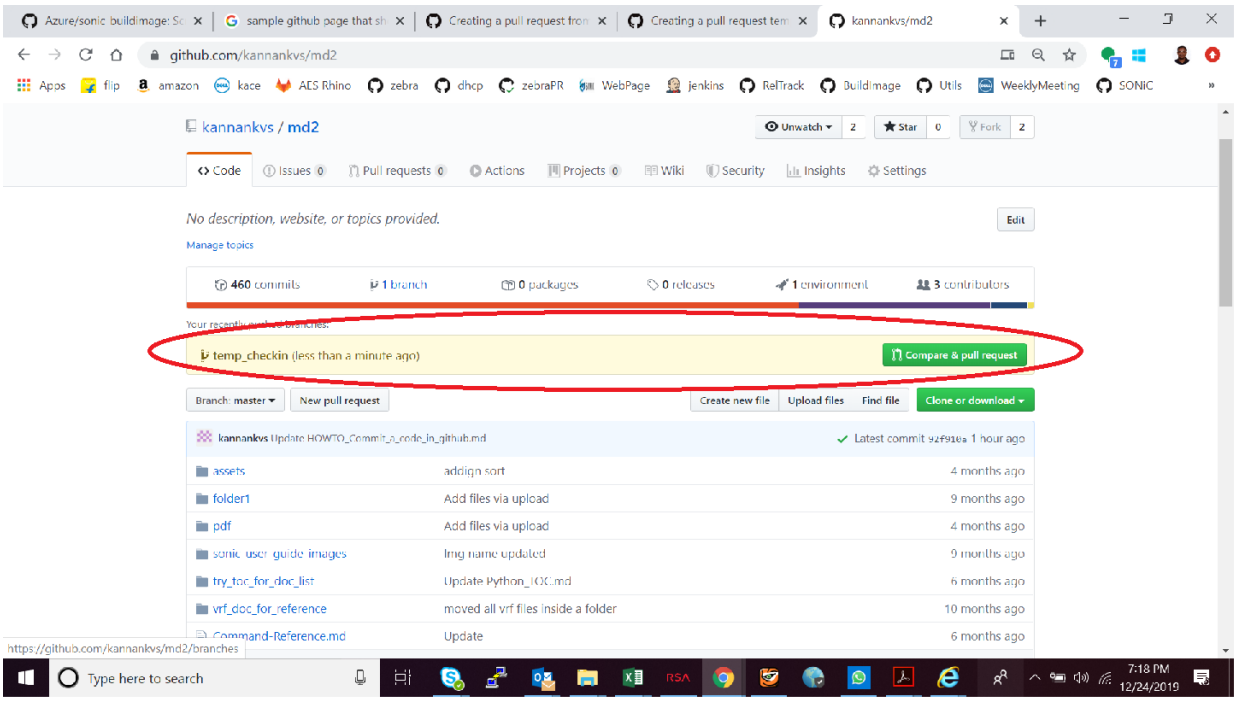
7. **Add the changes**
   Next step is to add the files using "`git add <file name>`" to add each file individually or using "`git add .`" to add all the modified files. Use "git status" to verify that all your changes got added.
8. **Commit the changes**
   Next step is to commit the changes to your branch using the command given below.

`git commit -m "<branch_name for reference>: <give your commnets here>"` Example : *git commit -m "SNMPAgent_cli_support: Added snmp_agent address CLI command to add and delete the address*"
Note : Regarding swss repository, the developer must write VS py tests for every feature that they contribute. This will run as part of every pull requests and helps to ensure that future changes by other developer is not breaking your feature.

10. **Push your changes to the main repository.** Next is to push the committed changes into upstream into the remote that was already set to "SNMPAgent_cli_support".

```
`git push --set-upstream origin <branch_name>`
Example : *git push --set-upstream origin SNMPAgent_cli_support*
```

11. **Raising the pull request.** The final step is to raise the pull request. Go to Azure github webpage https://github.com/Azure/sonic-buildimage (or your github webpage https://github.com/<your_github_username>/sonic-buildimage) where you will see a new line that says that you recently pushed some changes and it will show a link for "Compare & pull request". Refer the following image for the example.



Click on that green colour button that takes to a new page for new PullRequest (PR). It will show the branch **from** which merging has to happen, it shows the default repo "master" **to** which merging should happen. Fill the required details, explain the change in detail, explain the testing that was done to validate the changes and raise the pull request.

12. After raising the PR, continue to follow the PR for comments from reviewer, keep addressing the review comments and track it for closure. Once the PR is merged to individual repository, if you need the changes in image, the developer has to raise a pull request for “submodule” update. Ensure the description have all the commits using

“`git log --pretty=oneline --pretty=format:"%h - %ad : %s [%an]" --date=short`”.

Reference PR is at https://github.com/Azure/sonic-buildimage/pull/3675

Point to ponder to add a new feature or to enhance an existing feature

1. First step is to inform the SONiC community about the need for such enhancement and add the commitment for development for a particular release. Roadmap document should have been updated with this information.

2. Next step is to work out a development plan and update roadmap with Design Review Date, Code Review Date, etc.,
3. Next is to do the design and plan for a design review meeting with community. Present the design to community, get feedback from community, update the design and get the final approval for design from the community.
4. Do the required code changes, test it and raise pull request. Provide the test results as part of the pull request. Address the review comments and follow-up until the code is merged into the master branch and also on required release branches.
5. While fixing a bug, provide all the required reference to corresponding issue number. If the issue does not have a pull request for the issue, then raise a separate pull request with complete reproduction details and provide all relevant required details.