Stable Branch

This document describes the way of working with stable branches and doing maintenance.

The page is derived from https://wiki.openstack.org/wiki/StableBranch , simplified and adapted to OPNFV.

It was discussed on TSC on June 30, 2015 with minor comments.

At this time only Arno release maintenance is covered.

Overview

The stable branch is intended to be a safe source of fixes for high impact bugs and security issues which have been fixed on master since a given release. It allows users of release (stable) versions to benefit from the ongoing bugfix work after the release.

Official point releases for each project are published from the branch on a per need basis, as decided by the TSC. In later stages, a regular cadence for point releases may be introduced.

It's possible to check current maintained versions in the releases page. At this time only Arno is maintained.

OPNFV's stable branch policy borrows much from prior art, in particular from OpenStack.

In general all fixes should be made on the main branch and cherry picked to stable. If there is a case where the fix is not able to be merged backwards only then we would need to do any work directly on stable. The documented method for getting a fix into stable should be by a cherry-pick process.

Stable branch policy

Appropriate fixes

Only a limited class of changes are appropriate for inclusion on the stable branch.

A number of factors must be weighed when considering a change:

- **The risk of regression** - even the tiniest changes carry some risk of breaking something and we really want to avoid regressions on the stable branch
- **The user visible benefit** - are we fixing something that users might actually notice and, if so, how important is it?
- **How self-contained the fix is** - if it fixes a significant issue but also refactors a lot of code, it's probably worth thinking about what a less risky fix might look like
- Whether the fix is **already on master - a change must be a **backport** of a change already merged onto master, unless the change simply does not make sense on master (e.g. because of a change of architecture).
- If there is a suitable **work-around** for a bug, normally there won't be a fix on stable.
- Since OPNFV is using several upstream projects, typically fixes in the upstream projects will apply as fixes for OPNFV. Therefore complete **maintenance revisions of the upstream projects** (i.e. minor versions) can be used as suitable backports for OPNFV maintenance releases.
- Since OPNFV is a midstream integration effort, also test cases might be suitable backports in case they are related to critical bugs found in stable.

Rules to maintain multiple versions and exceptions will be added later.

The stable-mtc team needs to balance the risk of any given patch with the value that it will provide to users of the stable branch. A large, risky patch for a major data corruption issue might make sense. As might a trivial fix for a fairly obscure error handling case.

The stable-mtc team also will handle exceptions, of which the most common will be that a fix on stable needs to be different to the fix on master due to some other changes on master (e.g. architectural).

Some types of changes are completely forbidden:

- New features
- Changes to the external APIs
- Changes to the notification definitions
- DB schema changes
- Incompatible config file changes
- Changes including a version upgrade of an upstream component of OPNFV (since this will typically violate the above points)

Support phases

Support phases will be introduced at a later time

Review of fixes

Each backported commit proposed to gerrit should be reviewed and +2ed by two Arno-stable-maint members before it is approved. Where a stable-maint member has backported a fix, a single other +2 is sufficient for approval.

If unsure about the technical details of a given fix, stable-maint members should consult with the appropriate developers from the affected projects for a more detailed technical review.

If unsure if a fix is appropriate for the stable branch, at this time the TSC will do the final decision.

Security fixes

Fixes for embargoed security issues receive special treatment. These should be reviewed in advance of disclosure by committers and stable-maint. At the time of coordinated public disclosure, the fix is proposed simultaneously to master and the stable branches and immediately approved.

Processes

Proposing fixes

Anyone can propose a cherry-pick to the stable-maint team.

One way is that if a bugfix on master looks like a good candidate for backporting - e.g. if it's a significant bug with the revious release - then just nominating the bug for Arno maintenance will bring it to the attention of the maintainers.

If you don't have the appropriate permissions to nominate the bug, then send an email via the user list.

The best way to get the patch merged in timely manner is to send it backported by yourself. To do so, you may try to use "Cherry Pick To" button in Gerrit UI for the original patch in master. Gerrit will take care of creating a new review, modifying commit message to include 'cherry-picked from ' line etc.

If the patch you're proposing will not cherry-pick cleanly, you can help by resolving the conflicts yourself and proposing the resulting patch. Please keep Conflicts lines in the commit message to help reviewers! You can use git-review to propose a change to the stable branch with:

```
$> git log (find out the commit id of the patch that you want to backport from "git log" output)
$> git checkout stable/arno
$> git cherry-pick -x $master_commit_d
$> git review stable/arno
```

Note: cherry-pick -x option includes 'cherry-picked from ' line in the commit message which is required to avoid Gerrit bug

Failing all that, just ping one of the team and mention that you think the bug/commit is a good candidate.

Change-Ids

When cherry-picking a commit, keep the original Change-Id and gerrit will show a separate review for the stable branch while still allowing you to use the Change-Id to see all the reviews associated with it.

Hint: Change-Id line must be in the last paragraph. Conflicts in the backport: add a new paragraph, creating a new Change-Id but you can avoid that by moving conflicts above the paragraph with Change-Id line or removing empty lines to make a single paragraph.

Email Notifications

If you want to be notified of these patches you can create a watch on this screen: https://gerrit.opnfv.org/gerrit/#/settings/projects click "Watched Projects"

Project Name: All-Projects

Only If: branch:stable/arno

Then check the "Email Notifications - New Changes" checkbox. That will cause gerrit to send an email whenever a matching change is proposed, and better yet, the change shows up in your 'watched changes' list in gerrit.

Bug Tags

will be introduced when we see the need.

CI Pipeline

For Arno release the jobs will be run once per day per installer (Fuel and Foreman) on stable/arno branch. Since this is in addition to the jobs for master branch and jobs have long run time, this might need re-evaluation as we go on.

The artifacts arno/stable jobs produced are stored in the new directoies on artifacts.opnfv.org.

The artifacts produced by daily jobs would be stored

For stable/arno, the storage locations will be <project_name>/arno/<artifact_name>.iso

The docs produced by daily and merge jobs would be stored

For stable/arno, the storage locations will be <project_name>/arno/docs/<document_name>

No changes in overall functionality in merge and verify jobs: they will continue doing builds only

```
genesis-fuel-verify-master, genesis-fuel-verify-stable-arno,
genesis-fuel-merge-master, genesis-fuel-merge-stable-arno,
genesis-foreman-verify-master, genesis-foreman-verify-stable-arno,
genesis-foreman-merge-master, genesis-foreman-merge-stable-arno
```

Team organization

Project specific tasks

Each of the 5 projects that contributed to Arno will dedicate some committers which would be in charge of reviewing backports for their project, following the stable branch policy. It is in the responsibility of each project how to select those committers (e.g. vote in the team).

The group of these committers here are sometimes called the "stable branch maintenance team" or "stable-mtc" without this being necessarily a team with own organization.

Stable branch management

Stable branches are less exercised than master branches, and they may get broken by external events.

Therefore a group of committers, the "stable branch maintenance team" (stable-mtc) is tasked with specific stable branch support, making sure the branch stays in good shape and remains usable at all times. They monitor periodic jobs failures and enlist the help of others in order to fix the branches in case of breakage. They should also raise flags if for some reason they are blocked and don't receive enough support, in which case early abandon of the branch will be considered.

The stable-mtc is responsible for the enforcement of the Stable Branch policy. Initially it is composed of a release manager and at least one committer of each of the participating projects and will have similar organization and rights as OPNFV project teams. It will be granting exceptions for all questionable backports raised by projects, providing backports reviews help everywhere, and educating projects members on the stable branch policy.

The stable-mtc will propose to TSC to decide on point releases from the stable branch. Preparation of the point release will be described in a second step. The stable-mtc can also propose to TSC to decide to abandon maintenance of the release.

When a new OPNFV version is released, a stable-mtc team for that project will start. At that time, the earlier maintenance version will go to a phase with less support. Details will be defined later.

Joining the team

Existing committers are greatly encouraged to join the stable-mtc in order to help with reviewing backports, judging their appropriateness for the stable branch and approving them.

We're really keen to add more folks to the stable Arno stable-mtc to help out with reviews.

All you really need is some time and the ability to apply the "safe source of high impact fixes" and "must be fixed on master first" policies. It mostly comes down to having a good sense of the risk vs benefit of applying a backport to the branch.

If you'd like to join the team, you can start by simply [+-]1ing stable branch reviews. It's best if you can add some brief thoughts to your review on why you think the fix is suitable for stable so that we know how you're applying the policy.

Same as in normal OPNFV projects, stable-mtc team will use committer votes for some decisions like proposing a new point release to TSC.