A new deliverable for those who want latest and greatest: GDP Master

With this blog post, the GDP Project reaches a relevant milestone. We have turned our previous release based delivery model into a rolling one by creating a central pipeline where GDP is continuously being integrated. Out of this integration point, the GDP Delivery Team can release different deliverables targeting different audiences. Currently there are two main deliverables, GDP Master, targeting those GDP contributors and Power Users who need the latest available software, and Major releases like GDP-ivi9, which targets GDP users who want to consume more stable software in the form of images ported to specific target boards.

Why did GDP switched from a traditional delivery model to a more flexible one?

Some of the main reasons are:

- Makes it easier for interested parties to know where the work / bleeding edge of GDP is being conducted, previously we've received feedback stating the branch structure / naming schema was confusing, this should ease that problem.
- Having Master not be a strictly defined 'release' or 'golden' branch also allows more flexibility for the project, allowing master to develop whilst having a stable maintenance branch for such purposes.
- genivi-dev-platform repository started by having branches per target, that during release cycles could end up out of sync due to focus being applied to a specific target (mainly qemu, for non-bsp requirements). It was intended that the '*master*' qemu branch could be used to rebase /merge all common changes across all target branches, but this proved more difficult in practice and quickly got un-manageable even for simple changes (e.g. .gitmodules + conf file conflicts).
- Moving to a singular branch for all targets also comes with its own intricacies (more explicit scripting, the assurance of common commits etc) however managing contributions and building images from scratch is more efficient.
- Merging the git history of ~5 targets into a single branch in a sane way was not possible, so the qemu branch was taken as a base. Obviously the respective histories of each target branch in GDP is important, so the branches have been archived as tags.

And how does this new model works?

The more important policies that rules the submission and management processes are described in the GDP Management wiki page. The main goal of these policies is to create a system in which it possible to keep a constantly developing master branch, as well as providing a platform (branch) that acts as a stable snapshot of a release and as such is maintained for a given period. The policies will be applied by the maintainers, but it is important that users (whether that be contributors etc) follow the guidelines where possible, as this will ensure that GDP is workable for all parties. If a contributor is aiming to upstream a new package into GDP (directly into meta-genivi-dev or via an external layer) then it is expected to be compatible with the current genivi baseline / yocto baseline component versions available in master. This ensures that the GDP continues to move forward. Generally it is expected that any patch / PR to either repository is expected to pass the go.cd integration pipelines to ensure the system is buildable. go.cd/github has been configured to report the status of the relevant pipelines in the PR GUI.

That said this is only used as an initial check, and still requires standard code review. The CI is not bulletproof, build-failures are assessed to check if the PR was the root-cause and acted upon accordingly. Please note: as it stands GDP due to the nature of being defined essentially by two repositories, has well documented / discussed PR scenarios in which PR's to both repositories have to be tested in the same integration test. There is currently no mechanism in place to test this scenario and as such has to be dealt with manually for now. GDP now makes use of selective git submodules to generate the corresponding yocto layers based on the target being built, including meta-genivi-dev which also follows the master branch policy. The remote head of the repo is set to the master branch, i.e master is the default branch. And tags will still be used for snapshots of releases.

As you can see there is still a long way to go towards having a truly continuous delivery model through a rolling release but some fundamental steps has been taken. Obviously it is still possible to contribute to GDP via patches to genivi-projects@lists.genivi.org, as stated in the updated GENIVI Contribution policy.

GDP-ivi9 gets into maintenance mode

One of the consequences of adopting this new model is that GDP-ivi9, our latest release, gets into the maintenance cycle until the next major release is published, which is expected within this year. There are currently maintenance branches of genivi-dev-platform & meta-genivi-dev.git. The maintenance branch is the name given to the currently supported 'release' branch of GDP. Once a new release is declared and branched from master, the maintenance title is transferred. Only bug/security fixes or specific backport patches / PR's should be based against this branch. It is expected that a user looking for a 'stable' GDP build (or requiring a certain version of a package) would use the maintenance branch:

- In genivi-dev-platform: GDP-ivi9 maintenance branch.
- In meta-genivi-dev.git: GDP-ivi9 maintenance branch.

The intructions to build and run GDP-ivi9 has been updated to reflect these changes and new policies has been defined for maintenance.

Other areas of the project that has been improved as a consequence of the new model

Wiki

The fact that we have now two deliverables instead of one targetting different audiences have an impact in the way our contents are structured. Now there is one central page for each one of them from which Master and major release users can get all the information required to either run or download and boot GDP. For those who want to consume the latest and greatest, GDP Master is their page. Those more interested in a more conservative and easy to consume approach should go to GDP Releases wiki page. The reference page to download images and metadata remains unchanged.

As usual with key changes, there is still some work to be done in order to update contents in several pages, check links, titles, references to Master, update diagrams, etc. Not all this work is completed but most of the relevant updates are done by now. Feel free to point us to potential improvements.

Task management

Since Master is the central integration point and it is a continuous effort, we have re-structured the GDP Epics to reflect that most of the work is done now in Master. We have extended the usage of the JIRA capabilities related with Releases (FixedVersion), simplified the interfaces to create/update tasks, improve the integration between GitHub and JIRA/Confluence and other actions that will make our everyday work a little more efficient and easy to follow.

The main GDP Delivery Epics are now:

- ~~GDP-257~~ DONE where all the tasks related with Master are being tracked.
- ~~GDP-23~~ DONE where the delivery team collects all the tasks related with the potential next release. At this point it is undetermined if we will skip GDP 10 (based on poky 2.0 and meta-ivi 10) and jump directly to GDP 11. A new epic would be created in that case.
- ~~GDP-8~~ - GDP-ivi9 DONE where the only active task is ~~GDP-259~~ DONE to track the main actions taken to maintain the current release.

In summary, GDP has adapted the traditional GENIVI delivery model to a more flexible and modern one, executing actions at different level as consequence of the changes introduced

GDP Delivery Team