Vswitchperf Governance Model

vswitchperf Governance Model Proposal

The suggested governance model for the vSwitch Characterization project is a meritocratic model, that in terms of change/edit/review rights gives preference to committers, in other words only project committers can contribute directly to the project repository. Also only committers should be able to review suggested patches in Gerrit. However if anyone would like to become a committer, and actively contribute to the work, they will be more than welcome to formally join the project, there will be no barriers to entry in that respect.

Roles:

From TSC-Charter

- Contributor: Most Contributors work with their Committer and their component's sub-community. They contribute code or other artifacts, but do not have the right to commit to the code base. A Contributor may be promoted to a Committer by the projects' Committers.
- Committer: A person with rights to commit code or artifacts to the source code management system. The Committers will be the decision makers on design, code, packaging, and patches for their project. They must responsibly participate in the consensus decisions of the project. Committer rights are earned via code contribution and community trust. Committers select and vote for new Committers.
- Reviewer: Committer.
- Maintainer: A gatekeeper to the repository, so someone who will ensure that a patch has received the appropriate +1's and is able to give a +2 in Gerrit. The maintainer is an admin role that provides a +2, but is not a decision maker/benign dictator per se. The Maintainer enforces the agreed process. We will need to assign an overall maintainer.

How to submit a patch:

```
        Should we have our own development and discuss mailing lists? What does the community think?
```

JIRA:

JIRA will be used as a tool to allow us to achieve consensus on a feature before it's implemented as well as a bug tracking tool. All features to be developed should be entered into JIRA and approved by project committers before implementation starts.

Decision making process:

In general the process followed will be that of **lazy consensus**. Committers will be automatically notified (by email) when a new feature is added (For JIRA at least). They will be able to discuss/refine/disagree on the feature.

In general Committers have 72 hours to reject the feature otherwise they are assumed to agree. Objections/rejections should be supported by a counter proposal/feature. The alternative solution itself should then be submitted to the committers for lazy consensus. Committers will have 72 hours to reject the counter proposal, committers who don't voice an opinion are implicitly assumed to have agreed to the counter proposal.

If a resolution can't be reached after two rounds of lazy consensus (that is a proposal and a counter proposal), committers will have to cast an explicit vote whereby a majority must be achieved. The vote winning feature is the one that will be progressed to the implementation stage.

All bugs should be entered and tracked in JIRA.

Patches Review Ground rules:

A patch should contain a single logical change. For example each Test should be submitted as an individual patch for committers to review.

Each patch should be related to a particular JIRA issue/feature.

Each submitted patch will go through a Gerrit review. To move to the next stage, the patch MUST receive a +1 in Gerrit (Under Code-Review i.e. +1 Looks good to me, but someone else must approve) from at a committer from a different company.

Once a patch has received the required +1, the patch should be +2'd (Under Code-Review i.e. Looks good to me, approved) by the maintainer and pushed to the repo.
In the case of deadlock the same decision making process as specified above will be followed.

Patches **MUST** contain the following tags in the commit messages:

```
        Signed-off-by: Author Name author_name@email.address...
        JIRA: #IssueNumber
```

If there is more than one author they must all sign off a patch. Signing off a patch indicates agreement to the Developer's certificate of origin (see below).

Committers who review the patch should update the commit message with:

```
 Acked-by: Reviewer Name <reviewer_name@email.address...>
```

Patches can contain the following tags:

```
Tested-by: Tester Name <tester_name@email.address...>
Reported-by: Reporter Name <reporter_name@email.address...>
Suggested-by: Suggester Name <suggester_name@email.address...>
```

Developer's Certificate of Origin 1.1

```
By making a contribution to this project, I certify that:

(a) The contribution was created in whole or in part by me and I
    have the right to submit it under the open source license
    indicated in the file; or

(b) The contribution is based upon previous work that, to the best
    of my knowledge, is covered under an appropriate open source
    license and I have the right under that license to submit that
    work with modifications, whether created in whole or in part
    by me, under the same open source license (unless I am
    permitted to submit under a different license), as indicated
    in the file; or

(c) The contribution was provided directly to me by some other
    person who certified (a), (b) or (c) and I have not modified
    it.

(d) I understand and agree that this project and the contribution
    are public and that a record of the contribution (including all
    personal information I submit with it, including my sign-off) is
    maintained indefinitely and may be redistributed consistent with
    this project or the open source license(s) involved.
```