Problem

Docker container virtualization process uses Dockerfile to build an image. Dockerfile comprises of the specific instructions much like shell scripts. Example:

```
FROM cassandra
RUN sed -i 's/^start_rpc.*$/start_rpc: true/' /etc/cassandra/cassandra.yaml
```

Dockerfile is a basic text file and after it's created it's difficult to maintain and modify. In addition, Dockerfiles use base images in the FROM section, and these base images often maintained by different people.

When something changes in the base image it's difficult to trace it down. Also, in some situations the base images may lag behind (for instance with security updates) and the descendant images become vulnerable and not able to update without finding another base image.

Current solutions

People try and implement their own ways of combining Dockerfile templates. Example:

https://raw.githubusercontent.com/beekhof/pacemaker/master/extra/ansible/docker/roles/docker-host/templates/Dockerfile.j2

Every approach uses different techniques, all basically based around preprocessing Dockerfile templates into the “final” Dockerfile. That partially solves the first problem.

Second problem, which is much more serious as it could affect the security of the whole organisation, as of today stays unaddressed.

Proposal for problem 1

To completely solve the first problem provide a templating format and tool that will combine predefined parts of Dockerfile into the final format. For this separate scripts into “slices” with names representing the software being added and its version, like: jekyll-3.0.0.beta.1 or ruby-2.2.3

Each descendant slice will reference required base slices in the DEP section. All slices will be kept in a public repository (on github) with anyone able to submit a slice or a patch.

Further to this design, user can provide a separate directory of slices in the same format and version to replaces some of the public slices or to add on top. That will allow creating custom images out of mostly standard pieces with only minimal customizations.

Versioning

The slices repository and the sb tool is versioned by there rules:

1) The slices and tool must match by the major version. Example: slices 1.1.3 match the tool 1.0.34 or 1.32.4

2) The slices and tool minor version denotes the importance of changes, ex slices modified or the

tool critical updates.

3) Patch version can be freely used if the changes are not critical and backward compatibility is preserved.

4) Alpha, beta and rc versions need a special user confirmation to allow the tool to run or use the unstable slices.

Proposal for problem 2

This is the area of future work that is being currently investigated with some very promising ideas.