The Mini-Maint Experiment

Duncan White,
Department of Computing

Nov 2018

Introduction

- Our current **maint** system is powerful and flexible, but only works in DoC (because that's where it's evolved from first principles), and relies heavily on NFS, and our 146.169.0.0/17 network.
- We are potentially interested in being able to provide a CSG-like Ubuntu environment for machines not in DoC, not on 146.169.
- Giuseppe is looking at Ansible, which offers one possible route.
- Over summer 2018, I decided to try an experiment based on **maint**:

The mini-maint Experiment

- I wondered if one could pull out a small neat non-DoC specific subset of maint out of the 41000 (!) line behemoth that **maint** has become, and end up with a small and manageable system, that can still do useful work, in order to:
  - Demonstrate it to others.
  - Talk about it in future CSG Tech Seminars, and
  - Perhaps even use it for DoC purposes outside of 146.169..
  - Take the opportunity to redesign a few longstanding irritations (such as **maint dist**'s inability to **remove files** when a host **leaves a hostclass**, which necessitated the hideous **maint undist**) in a playpen, for later porting back to full blown **maint**.
- Let's call this concept **minimaint**.

- My idea was to have 3 core components:
  1. A public git repo containing all the source code - including **/sbin/maint**, the libraries and all the maint scripts.
  2. A separate git repo containing all DoC specific configuration, including per-maint script runon and runwhen data, plus per-maint config data eg. the dist tree and the apt package lists etc.
  3. We would also need to replace db-admin entirely, as all the information in it, and even the schema, is DoC specific. However, one could abstract out the information that all sites would need:
     - A set of users.
     - A set of groups.
     - A set of (user,group) membership information.
     - Machine similarity relationships (ie. hostclasses).

     Of course, this data changes more rapidly than most config data (that's why it's in a database after all:-)). More thought is needed about this.

- The source code git repo would have to be reengineered to have NO DoC specific knowledge in it, no use of NFS, rsynclinux.doc.ic.ac.uk, db-admin database, and no casual assumptions like *every hostname is in .doc.ic.ac.uk*.
- The config repo would not so public, but would be accessible to DoC machines outside of 146.169. This config information ends up in an **/etc/minimaint** directory on a client.
- Other sites could use a *different* config repo, while still using the common source repo.
- Replacing db-admin: I wondered whether all this data could be downloaded in bulk from a RESTful API, in JSON format.
- Alternatively, it could be generated (in our case FROM db-admin) as a small number of JSON files in a standardised site-neutral format, every N minutes, added to the config repo and pushed.

- For now, my demonstration config repo contains a hostclass file containing some made up example hostclass info, a users file listing the users etc.
- The basic strategy for developing **mini-maint** was to pick particular goals (maint scripts are the obvious source of such goals), and COPY only those bits of maint and it's libraries that are needed to implement the current goal into the **mini-maint** source repo.
- While incorporating each new chunk of code, we will separate out the config from the source code, delete as much historically accumulated nastyness as possible, and refactor it as much as possible.
- Then we will simply see what emerges over time.

Initial Goals for mini-maint on 16.04

1. **minimaint hello**:
   Get **minimaint** to run a "hello world" maint script, with the minimum amount of infrastructure up and running. Achieved Monday 2nd July, with approx 4200 lines of Perl code.

Initial Goals for mini-maint on 16.04

1. **minimaint hello**:
   Get **minimaint** to run a "hello world" maint script, with the minimum amount of infrastructure up and running. Achieved Monday 2nd July, with approx 4200 lines of Perl code.
2. **minimaint dist**:
   Achieved Tuesday 3rd July, now approx 4700 lines of code.

Initial Goals for mini-maint on 16.04

1. **minimaint hello**:
   Get **minimaint** to run a "hello world" maint script, with the minimum amount of infrastructure up and running. Achieved Monday 2nd July, with approx 4200 lines of Perl code.
2. **minimaint dist**:
   Achieved Tuesday 3rd July, now approx 4700 lines of code.
3. Reengineered **minimaint dist**:
   fully integrated undist logic into dist, so that can remove host-class specific files when a host *leaves a host class*. Also reengineered the dist hierarchy to allow us to set properties (ownership, mode etc) in a dist directory, to affect all alternative versions of that file for different host classes. Achieved Thurs 5th, now approx 5000 lines of Perl code, but MUCH prettier:-).

4. **minimaint aptitude**:
   Ported maint_compose() and did initial **016findpkgs maint script** to produce the list of desired pkgs, Thurs 5th July.
5. Then ported the rest of **017aptitude**, got working at home (yes, from offsite!) the evening of Fri 6th July. approx 5900 lines of Perl code now.

4. **minimaint aptitude**:
   Ported maint_compose() and did initial **016findpkgs maint script** to produce the list of desired pkgs, Thurs 5th July.
5. Then ported the rest of **017aptitude**, got working at home (yes, from offsite!) the evening of Fri 6th July. approx 5900 lines of Perl code now.
6. Bootstrapping code: Thought about how to best get minimaint to start on a new fresh Ubuntu 16.04 install not necessarily in DoC. Set up a fresh minimaint config for "this site" as a git repo. Extracted the minimaint BEGIN code out, tweaked it a bit to give the bootstrap script, added some code to it to install some prereq packages, then read a config.url file saying where to find the site config git repo, clone that into /etc/minimaint, the read the /etc/minimaint/phase1 config file, determining where to fetch the minimaint source tree itself from [a second git repo] etc. Did this mainly on Fri 6th July too. Probably only 50 lines of fresh code, as most was a copy of parts of **sbin/minimaint**.

7. **minimaint useraccounts** Ported the massive UserAccounts module, pulled out all the config (especially the tight linkage with the db-admin database, replaced for now with config files, which I autogenerated via generators from db-admin), and then implemented the 010useraccounts maint script. Got it generating /etc/passwd and /etc/group and /etc/auto.homes on Tues 10th, with approx 6700 lines of Perl code.
8. But of course ran into the user-id and group-id problems given that the base install was not a CSG-managed system. Added an offset capability to change the numeric uids and gids of all DoC users on such a machine as a temporary fix.

7. **minimaint useraccounts** Ported the massive UserAccounts module, pulled out all the config (especially the tight linkage with the db-admin database, replaced for now with config files, which I autogenerated via generators from db-admin), and then implemented the 010useraccounts maint script. Got it generating /etc/passwd and /etc/group and /etc/auto.homes on Tues 10th, with approx 6700 lines of Perl code.
8. But of course ran into the user-id and group-id problems given that the base install was not a CSG-managed system. Added an offset capability to change the numeric uids and gids of all DoC users on such a machine as a temporary fix.
9. Imported our user-id and group-id filesystem renumbering tool (which we use when preparing a new linuxroot in full maint), gave users the option of running it when bootstrapping minimaint onto the machine, so that when in DoC, and generating /etc/auto.homes as well, NFS actually works.

10. Most of the rest of the maint scripts - next few days. Reengineered 4 auditing maint scripts that previously (yuck!) wrote direct into /vol/linux/autofiles, to use a new client/server I wrote (adapting the Raspberry Pi and cloud VM "phone home" daemons), that sent key/value pairs, with the values encoded as JSON, to a listener that received them and updated files in the new /vol/auditinfo volume.

- NB: the new audit framework was developed in real maint, but with minimaint structure in mind, because we have wanted to stop writing direct into /vol/linux for ages, and this solved the problem! Hence **095newaudit**.
- I also implemented, in the newaudit scripts, a mechanism to place site-specific additional Perl modules in the config repo.
- At this point, in late July, I informally demonstrated mini-maint to most of CSG around at the time. Then I stopped work on mini-maint, as other start of term preparations rose in priority.

mini-maint on Ubuntu 18.04

- These last few days, in early November, I thought I'd revisit making **minimaint** work on Ubuntu 18.04 - leaving the minimaint source repo entirely unchanged.
- I copied the original Ubuntu 16.04 minimaint-in-DoC config git repo (**csg-mini-maint-dcw-config**) to a fresh repo: **git@github.com:ImperialCollegeLondon/csg-mini-maint-dcw-config-18.04.git**
- Then I tweaked various disted files (for example, the apt/sources.list.d files that named 16.04), and grabbed a small number of hostclass-specific package lists from 18.04 maint.
- A few hours work and mini-maint was running happily on 18.04, subject to the same limits as Ubuntu 16.04 earlier.

Time for a demo! mini-maint on Ubuntu 18.04

- Ok. Enough talk. Let's see it in action. We start with a machine with a **completely blank hard disk**.
- Then we install non-CSG Ubuntu 18.04 on it, which takes about 10 minutes, and freshen up all the packages, which takes as long again.
- Then, in a root terminal, we:

```
cd /tmp
wget https://www.doc.ic.ac.uk/~dcw/bootstrap18.04.tgz
tar xzf bootstrap18.04.tgz
cd bootstrap
```

- Now, check that the config.url file contains the correct URL of your site repo, now's the time to change this to another URL if you want.

```
./bootstrap
```

- During the bootstrap, you are asked if you want to renumber uids and gids on the disk (for compatibility for DoC uids and gids from 100 up) - if you're setting up a machine on a DoC vlan where you might want NFS to work later, say yes.
- Later, you're asked to either set the machine's hostname or accept it's current one. Whatever the machine's hostname ends up as (currently the SHORT hostname, i.e. without the domain), it will need to be in at least one hostclass in the 18.04 config repo for **minimaint** to work.
- The last thing that **bootstrap** does is to run **minimaint hello**. If everything's ok, it'll say hello in typical **maint** style.
- You should now run:

```
minimaint dist
minimaint aptitude
```

- This installs approx 4500 packages. For some reason, doing so takes out Gnome - and even the ttys. But a reboot fixes that!

What Does minimaint do, and not do, and Where Next?

- **minimaint** is just an experiment. We don't have to do any more work on it, if we don't want to. I created it for various reasons, including enabling me to give a talk about **maint**'s structure in a future CSG Technical Seminar.
- If nothing else, I fully intend to backport the new improved **dist/undist** logic, and the new improved **perms** maint script, into **maint**.
- But we could go further, for a modest investment of time/energy. Currently:
- **minimaint** has no installation mechanism. I didn't tackle that in the summer, but there's no reason to think that we couldn't port the installation part of **maint** into **minimaint** land. Focusssing solely on Lloyd's REXX based partitioning might allow a much simpler and smaller configuration.

- We'd need to replace the **NFS root** delivery mechanism with some other mechanism, that could work readonly around the world. Of course we'd have to tackle GPT/UEFI at some stage.
- **minimaint** does not run at boot, but that's just a matter of disting a few files (to add a systemd job to run minimaint).
- **minimaint** does not run from cron, but that should be straightforward.
- **minimaint** doesn't, when operating on a PC outside 146.169, magically solve the *how could we support DoC users outside of DoC?* problem. We could reuse Guiseppe's work on setting up LDAP/Samba/LightDM configuration, dist out some extra files, and if necessary add an LDAP-and-Samba maint script to do additional configuration on **minimaint** hosts not in the **DOCHOSTS** class.
- **Collaborators welcome - indeed essential - on any of the above.**