Pivot-Overlay Setup  
Design Document

Erik Popp

Purpose of script:

1. Automate the process of setting up a router with pivot-overlay
2. End up with a package of sufficient quality that I can publish to OpenWRT’s official repositories
3. Work with stock OpenWRT without requiring a custom build
4. Work well as a demonstration of my skills to a potential employer

Design goals:

1. Be as safe as practical
2. Provide an excellent user experience
3. Easily enable further expansion and improvement – especially support for more routers
4. Work equally well when scripted as when interactive
5. Run well on routers with too little spare flash to download all the packages needed to access the flash drive/ssd
6. Have well-architected build, deploy, and version management system

Planned methods of accomplishing these goals:

1. Be as safe as practical
   1. As much as practical, make it possible to undo each step
      1. To accommodate the scenario where the router is unreachable after a step, automatically undo each given step after N seconds unless the user confirms that the step succeeded
      2. Give the user enough information for them to determine if they succeeded
   2. At each step, offer the user the option to do a dry run to see if it looks like they expect
      1. Dry run should be default
      2. To reduce frustration, after the user does a dry run, offer to default to real runs and to re-run the previous command for real
      3. Can this be accomplished while the current command is running?
   3. At each step, explain to the user what stage they are on, why it is needed, and what options they have
2. Provide an excellent user experience
   1. Have a carefully thought out UI that is easy to navigate
   2. Always tell the user where they are in the process
      1. I think that the best way to implement this is with a breadcrumb line
3. Easily enable further expansion and improvement – especially support for more routers
   1. Have a section for router-specific code and some method of automatically identifying the router model/architecture
4. Work equally well when scripted as when interactive
   1. Split the program into the following sections:
   2. Functions (commands)
   3. Command interpreter
   4. UI
   5. CLI
5. Run well on routers with too little spare flash to download all the packages needed to access the flash drive/ssd
   1. Automatically detect when the router is likely to not have enough internal storage to hold the usb drivers, etc.
   2. Offer to download the needed packages to RAM from a local source (NOT the official source) each time the router starts up
   3. Keep a list of which packages need to be loaded each time
6. Have well-architected build, deploy, and version management system
   1. Using Git and Github
   2. I want to be able to deploy my script to the router that I’m working on from my laptop by running a deploy script on my laptop
   3. As much as practical, deploy system should minimize the risk of bricking my router
      1. Script should, as much as practical, back up the router’s internal state to RAM (probably in a .tgz file) before deploying the new version of the file
         1. This should be made performant by, if practical, pre-emptively making a backup when I start working on the project
         2. Alternatively, it could copy the files uncompressed into RAM and somehow only update what has changed (rsync temporarily installed?)
         3. It may be best to have the deploy script back up the current contents of the writable partition to a folder in the project on the laptop; probably using rsync