DIMS Developer Guide

Release 1.0.0

David Dittrich

Nov 27, 2017

Contents

This document (version 1.0.0) covers issues related to development of DIMS components from a developer's perspective.

CHAPTER 1

Introduction

This chapter introduces the software development policies, methodology for source code and configuration file management, documentation, and use of continuous integration mechanisms used for deployment and testing of DIMS components.

1.1 Overview

- Section *Development and Core Tool Policy* discusses the policies that developers should understand in order to operate in a manner consistent with the rest of the team.
- All documentation for the DIMS project is written using restructured text (reST) and Sphinx. Section ref:*documentation* covers how to use these tools for producing professional looking and cross-referenced on-line (HTML) and off-line (PDF) documentation.
- DIMS software is maintained under version control using Git and the HubFlow methodology and tool set. Section *Source Code Management with Git* covers how these tools are used for source code, documentation, and system configuration files.
- Changes to source code that are pushed to Git repositories trigger build processes using the Jenkins continuous integration environment. These triggers build and/or deploy software to specified locations, run tests, and/or configure service components. In most cases, Ansible is used as part of the process driven by Jenkins. Section *Continuous Integration* provides an overview of how this works and how to use it in development and testing DIMS components.
- System software installation and configuration of DIMS components are managed using Ansible playbooks that are in turn maintained in Git repositories. Only a bare minimum of manual steps are required to bootstrap a DIMS deployment. After that, configuration changes are made to Git repositories and those changes trigger continuous integration processes to get these changes into the running system. Section *Deployment and Configuration* covers how to use this framework for adding or managing the open source components that are used in a DIMS deployment.
- For more information about the Lifecycle of DIMS Machines reference dimspacker:lifecycle. For more information about software development in general, reference dittrich:swdev.

CHAPTER 2

Referenced documents

1. ansibleinventory:ansibleinventory
2. dimsdockerfiles:usingdockerindims
3. dimspacker:dimspacker
4. dimssr:dimssystemrequirements
5. dimsad:dimsarchitecturedesign
6. dittrich:homepage home page.

CHAPTER 3

Development and Core Tool Policy

This section contains policy statements regarding software development that all developers working on the DIMS project are expected to adhere to.

In order to prevent core tools being used by developers being incompatible, rendering installation instructions buggy and/or causing random failures in a complicated build environment, everyone on the DIMS project **must** use the same core tools, and use the same workflow processes. This will allow controlled updates and provide stability in the tools we are using within the project.

Without some discipline and adherence to documented policies, far too much time ends up being wasted when one person can do something, but another can't, or something runs fine on one system, but fails on another system. In either case, team members get blocked from making forward progress and the project suffers as a result. These policies are not being imposed to stifle anyone's creativity, but to help *everyone* on the team be more productive.

Attention: The requirement to adhere to the policies stated here is partly to keep the project moving forward smoothly, but also to ensure that the sofware products developed by this project are suitable for public open source release as required by the contract (see dimssr:opensourcerelease in dimssr:dimssystemrequirements) and in conformance with University of Washington policy.

3.1 General Software Development Philosophy

This section covers some very high-level philosophical points that DIMS software developers should keep in mind.

There are a huge List of software development philosophies on Wikipedia. One of the most relevant to the DIMS project, based on a contractual requirement (see dimssr:agileDevelopment) is the Agile Manifesto. This manifesto is based on twelve principles:

1. Customer satisfaction by early and continuous delivery of valuable software
2. Welcome changing requirements, even in late development
3. Working software is delivered frequently (weeks rather than months)
4. Close, daily cooperation between business people and developers

5. Projects are built around motivated individuals, who should be trusted
6. Face-to-face conversation is the best form of communication (co-location)
7. Working software is the principal measure of progress
8. Sustainable development, able to maintain a constant pace
9. Continuous attention to technical excellence and good design
10. Simplicity—the art of maximizing the amount of work not done—is essential
11. Self-organizing teams
12. Regular adaptation to changing circumstance

3.1.1 DIMS Development Credo

The following are a series of guiding development principles that have become self-evident over time as the project progressed. It echoes some of the concepts expressed in the Agile Manifeto above.

- **Avoid friction** - "Friction" is anything that slows down an otherwise smoothly running process. Little things that are broken, missing facts, new programs that were written but don't yet have any documentation, all make it harder for someone to get work done because something causes friction. Everything grinds to a halt until the little roadblock can be removed and then it takes more time to ramp back up to speed.
- **Know your tools** - It is tempting to just start using a program without first reading the fine manual (RTFM!?). While this may seem like a quick way to get up and running, it can end up costing much more in the long run. Take the time (a couple hours, perhaps) to read through as much of a new tool's documentation to familiarize yourself with the tool's capabilities and where to find answers to "how the heck to do I do X?"

  Note: The PI has, for decades, maintained an extensive web page with links to resources on tools and how to use them. Check there first, and feel free to pass along any good resources you find that are not yet included there.

- **Take control** - Relying on the default behaviors that are programmed into an open source product that we use within the DIMS project, without fully understanding them, can cause problems. When possible, being explicit about how programs are configured and how they are invoked can make these opaque default behaviors less of a problem.
- **Make it simple** - It may take a little effort, but being focused on finding a simple solution that can be applied uniformly makes it easier to intergrate a large set of components. The more differences there are the way a subsystem or service is configured on multiple hosts (like DNS, for example) means the behavior is random and unpredictable from one computer system to another, causing friction
- **Make it work first, then make it better** - Trying to engineer a complete solution to some need can mean delays in getting something working, which delays getting that component integrated with other components. Or worrying about how slow something might be during initial development and trying to optimize the solution before it is even working and tested by someone else. Make it work first, doing something simple, then deal with optimization and a comprehensive feature set later.
- **Avoid hard coding!!!** - When ever possible, avoid using hard-coded values in programs, configuration files, or other places where a simple change of plans or naming conventions results in having to go find and edit dozens of files. A complete system made up of multiple services and software components that must be replicated as a whole *cannot* possibly be replicated if someone has to hunt down and change hundreds of values in files spread all over the place.
- **Ansible-ize all the things** - All configuration, package installation, or entity creation on a computer system should be looked at in terms of how it can be automated with Ansible. Whenever you are tempted to run a

command to change something, or fire up an editor to set a variable, *put it in Ansible and use Ansible to apply it*. Manual processes are not well documented, are not managed under version control, are not programatically repeatable, and make it harder to scale or replicate a system of systems because they cause *huge* amounts of friction.

- **Template and version control all configuration** - Adding a new service (e.g., Nginx, or RabbitMQ) that may have several configuration files is not just a one-time task. It will be repeated many times, for testing, for alternate deployments, or when hardware fails or virtual machines get upgraded. Don't think that cutting corners to get something up and running fast by just hand-configuration is the right way to go, because doing it again will take as much time (or maybe even longer, if someone unfamiliar with the process has to do it the next time). Take the time when adding a new service to learn how it is configured, put all of its configuration files under Ansible control, and use Ansible playbooks or other scripts to do the configuration at deployment time and at runtime.
- **Done means someone else can do it, not just you.** A program that compiles, but nobody else can run, is not done. A bug that was fixed, but hasn't been tested by someone other than the person who wrote the code or fixed the bug, is not done. Something that doesn't have documentation, or test steps that explain how to replicate the results, are not done.
- **You can't grade your own exam** Tickets should not be closed until someone else on the team has been able to validate the results.
- **Document early, document often** - A program that has no documentation, or a process someone learns that has no documentation to record that knowledge and how to use it, doesn't contribute much to moving the project forward. We are a team who mostly works independently, across multiple timezones and on different daily schedules.

3.2 Source Code Control

As pointed out by Jeff Knupp in his blog post Open Sourcing a Python Project the Right Way, "git and GitHub have become the de-facto standard for Open Source projects." Just as Knupp's post suggests, the DIMS project has been following the same *git-flow* model described by Vincent Driesen in his A successful Git branching model blog post, using *Sphinx* and *RST* (see the section *Documenting DIMS Components*), and using *continuous integration* via Jenkins (see *Continuous Integration*).

3.3 Copyright

All source code should include a copyright statement with the year the project started (2013) and the current year, as shown here:

```
#!/usr/bin/env python
#
# Copyright (C) 2013-2017 University of Washington. All rights reserved.
#
# ...
```

Note: Where possible, include the actual copyright symbol. For example, in Sphinx documents, follow the instructions in Section *Insertion of text using direct substitution*.

3.4 Python Development and Debugging

Several component subsystems used in and developed the DIMS project are written in Python. We have chosen to use the PyCharm Community Edition integrated development environment and debugger.

Note: Updating PyCharm may involve exporting and re-importing settings at the time the program. This is covered in Section updatingpycharm.

3.5 License

All source code repositories shall include the following license statement to accompany the Copyright statement in the previous section.

```
Berkeley Three Clause License
=============================

Copyright (c) 2013-2017 University of Washington. All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this
list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice,
this list of conditions and the following disclaimer in the documentation
and/or other materials provided with the distribution.

3. Neither the name of the copyright holder nor the names of its contributors
may be used to endorse or promote products derived from this software without
specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

3.6 Developing on a fork from GitHub

In this section, we will go through the steps for using Hub Flow for developing on a branch forked from GitHub, publishing the results back to GitHub for others to share.

For this example, there has already been a fork made on GitHub. Start by cloning it to your local workstation:

```
[dittrich@localhost git (master)]$ git clone https://github.com/uw-dims/sphinx-
↪autobuild.git
Cloning into 'sphinx-autobuild'...
remote: Counting objects: 366, done.
remote: Total 366 (delta 0), reused 0 (delta 0)
Receiving objects: 100% (366/366), 62.23 KiB | 0 bytes/s, done.
Resolving deltas: 100% (180/180), done.
Checking connectivity... done.
[dittrich@localhost git (master)]$ cd sphinx-autobuild/
[dittrich@localhost sphinx-autobuild (develop)]$ git branch -a
* develop
  remotes/origin/HEAD -> origin/develop
  remotes/origin/develop
  remotes/origin/feature/1-arbitrary-watch
  remotes/origin/feature/tests
  remotes/origin/master
[dittrich@localhost sphinx-autobuild (develop)]$ git checkout master
Branch master set up to track remote branch master from origin by rebasing.
Switched to a new branch 'master'
[dittrich@localhost sphinx-autobuild (master)]$ git branch -a
  develop
* master
  remotes/origin/HEAD -> origin/develop
  remotes/origin/develop
  remotes/origin/feature/1-arbitrary-watch
  remotes/origin/feature/tests
  remotes/origin/master
[dittrich@localhost sphinx-autobuild (develop)]$ ls
AUTHORS                          NEWS.rst                        fabfile.py
↪              requirements-testing.txt
CONTRIBUTING.rst            README.rst                      fabtasks
↪      requirements.txt
LICENSE                          docs                            pytest.ini
↪              setup.py
MANIFEST.in                 entry-points.ini                requirements-dev.txt
↪      sphinx_autobuild
```

CHAPTER 4

Communication and coordination

This section discusses the communication mechanisms and how to use them for distributed development and working remotely.

Attention: Also refer to the *Development and Core Tool Policy* section for the rules of the road for working with the communication and coordination tools described in this section.

4.1 Daily Scrum

The project used a daily scrum meeting to try to keep everyone on track and moving forward. As much as possible (without an officially trained "scrum master" to lead every meeting) team members were encouraged to answer three questions:

1. What ticket(s) have you been working on?
2. What ticket(s) will you be working on next?
3. What is blocking your progress?

The goal is to keep the meeting to 15 minutes, so any longer discussions were deferred to a "tailgate" session after everyone has made their initial scrum contribution. The "tailgate" session may last much longer, sometimes running to over an hour (since some team members on the DIMS project were working remotely and it wasn't possible to just "drop into to your office" for side-bar conversations).

Note: It is *really hard* to avoid these 7 Mistakes During the Daily Stand-up Meeting.

4.2 Remote Coordination

This section describes the tools used for coordinating work remotely, such as team coding, daily scrum, and weekly "standup" meetings.

We initially used Adobe Connect, though that was phased out because it was so unstable and difficult to use. Various other coordination tools were tested, with Skype and Google Hangout being the most compatible and useful (though each has its own issues.)

4.2.1 Using Google Hangout

Requirements

- Firefox (at least when running Ubuntu on developer laptops)
- A Gmail address
- Plugin for Google Hangouts and Google Talk

Note: For now, DIMS developers should keep their name and email address in the file `$NAS/users/user-data.txt` for this purpose.

Plugin Installation

1. Go to http://www.google.com/tools/dlpage/hangout/download.html?hl=en
2. Choose the "64 bit .deb (For Debian/Ubuntu)" option.
3. Click the "Install Plugin" button.
4. When the software installation dialog box opens, choose to open the file with the Ubuntu Software Center.
5. Click the "Install" button on the upper right hand side of the next window that opens.
6. You will be prompted for your password.
7. The install should finish on its own.

When you make or join your first call

1. In your Gmail window, all contacts you may chat with will be on the left hand side, unless you've changed your settings.
2. Since we all don't have each other's addresses for Gmail right now, you won't actually have any contacts.
3. Whomever starts the Hangouts session can send a link via email to each member of a meeting.
4. If you get an invitation via an email, click the link in the email, and a new window will open for the Hangout session.
5. If whomever starts the Hangout session knows your email and adds you to the call, a box will pop up in the bottom right hand of your Gmail window. The top of this box will be highlighted green.
6. Click the part that says you've been invited to a Hangouts video chat.
7. There will be a pop up in the top left hand corner, pointing to a gray box to the left of the address bar.

8. Click the gray box and choose “Allow” or “Allow and Remember” for the plugin to work during just the current call or for all calls, forever.

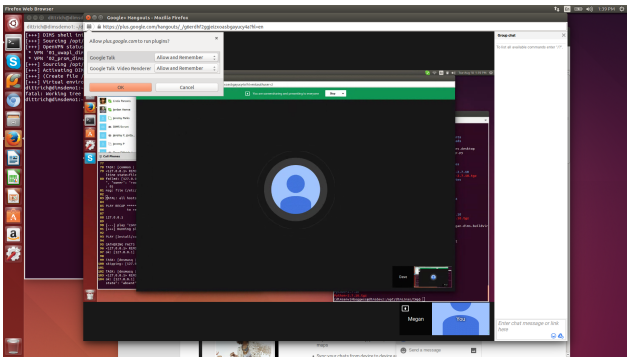


Fig. 4.1: Google Hangout Plug-in Settings

Note: If your sound or microphone settings are sounding funny:

- Check Ubuntu’s sound settings
- Make sure the speakers and microphone settings are using the appropiate option (i.e., the builtin hardware if you aren’t using headphones and vice versa).

Screensharing in Google Hangouts

These docs have been update to reflect the changes Google has made to Hangouts. There is an ‘original version’, and sometimes the ‘new version’ is finicky, and we must resort to the ‘original version’. As such, the instructions for the ‘original version’ will be preserved in the *Google Hangouts ‘Original Version’ Screenshare Instructions*

- Everyone can do it, at the same time!
- Hover your cursor in the top right hand corner of your Hangouts window.
- Click the option with the three vertically-aligned dots.
- A menu will drop down, choose `Screen Share`.
- You can choose to share your whole desktop or individual windows of other applications you have open on your desktop. It doesn’t appear you can share all windows of an application, such as Terminal. If you have 5 Terminal windows open, you can only share 1 of them, unless you share your entire desktop. You can open multiple tabs, and those will be shared.
- Resizing of windows works just fine when screensharing also.

> **Caution:** If two people are sharing the screen at the same time, and one of them puts the focus on the other person’s shared screen, you will put Google Hangout into a feedback loop that will eventually bring the internet to its knees. Don’t say I didn’t warn you!

4.2.2 Ops-Trust email lists

We use an instance of the Ops-Trust portal system for managing accounts and project email lists. Once an account is active, there are a number of mailing lists that DIMS project members should join to get various email communications

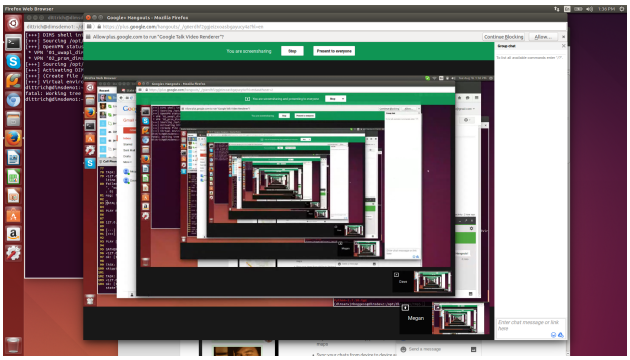


Fig. 4.2: The internet melting in an infinite Google Hangout loop...

(some ad-hoc and sent by humans, and some automatically generated for situational awareness, alerts, testing, etc.)

List	Purpose
`dims-datafeeds`	Automated data feeds
`dims-demo`	DIMS demo mailing list
`dims-devops`	Continuous integration and syadmin notifications
`dims-general`	General Discussion
`dims-reports`	Periodic generated reports
`dims-test`	DIMS test mailing list
`dims-testeval`	Testing and Evaluation reports
`dims-vetting`	Vetting and Vouching

CHAPTER 5

Source Code Management with Git

Daily development work on DIMS source code is done using a local server accessed via SSH to `git.devops.develop`. The public release of DIMS software will be from github.com/uw-dims with public documentation delivered on ReadTheDocs. (DIMS documentation is covered in Section *Documenting DIMS Components*.)

Note: At this point github.com/uw-dims primarily contains forked repositories of the software described in Section installingtools.

Team members need to have familiarity with a few general task sets, which are covered in the sections below. These tasks include things like:

- Cloning repositories and initializing them for use of the `hub-flow` Git addon scripts.
- On a daily basis, updating repositories, creating feature or hotfix branches to work on projects, and finishing those branches after testing is complete to merge them back into the `develop` branch.
- Creating new repositories, setting triggers for post-commit actions, and monitoring continuous integration results.
- Keeping up to date with new repositories (or starting fresh with a new development system by cloning all DIMS repositories a new.)

Attention: Every now and then, you may do something with Git and immediately think, "Oh, no! I did *not* want to do *that*..." :(

There are resources on Dave Dittrich's home page in the dittrich:usinggit section. Two good resources for learning how things work with Git (and how to undo them) are:

- How to undo (almost) anything with Git, GitHub blog post by jaw6, June 8, 2015
- Undo Almost Anything with Git webinar, YouTube video by Peter Bell and Michael Smith, February 11, 2014

5.1 Foundational Git Resources

- Yan Pritzker's Git Workflows book
- The Thing About Git
- Commit Often, Perfect Later, Publish Once: Git Best Practices
- Git Tips
- git-flow utilities to follow Vincent Dreisen branching workflow
- HubFlow (GitFlow for GitHub)

5.2 The need for policy and discipline

Git is a great tool for source management, but can be a little tricky to use when there is a team of programmers all using Git in slightly different ways. Bad habits are easy to form, like the short-cut of working on the `develop` branch in a multi-branch workflow.

Figure *Vincent Driessen Git branching model* comes from Vincent Driessen's "A successful Git branching model". The DIMS project is following this model as best we can to maintain consistency in how we create and use branches. The general policy is to derive branch names from Jira tickets, in order to keep information about why the branch exists, who is responsible for working on it, and what is supposed to be done on the branch, in a system that can track progress and prioritization of activities within sprints.

Because public release of source code will be through GitHub, the `hubflow` tool was chosen for use within the project. Take a moment to read through the following Gist (original source: bevanhunt/hubflow_workflow), just to get an overview of `hubflow` concepts. This Gist provides an overview of `hubflow` branch concepts and some other things about Git that are good to keep in mind, but this is *not* the totality of information in this guide about using `hubflow` (keep reading further down for more DIMS-specific examples of using `hubflow` commands).

```
Git Hubflow Workflow:

Sync Branch:
git hf update - this will update master and develop and sync remote branches␣
↪withlocal ones (be sure not to put commits into develop or master as it will push␣
↪these up)
git hf push - this will push your commits in your local branch to the matching remote␣
↪branch
git hf pull - this will pull the remote commits into your local branch (don't use if␣
↪the remote branch has been rebased - use git pull origin "your-branch" instead)

Feature Branch:
gif hf feature start "my-feature" - this will create a feature branch on origin and␣
↪local will be based off the latest develop branch (make sure to git hf update␣
↪before or you will get an error if local develop and remote develop have divereged)
git hf feature finish "my-feature" - this will delete the local and remote branches␣
↪(only do this after a PR has been merged)
git hf feature cancel -f "my-feature" - this will delete the local and remote␣
↪branches (only do this if the feature branch was created in error)
git hf feature checkout "my-feature" - this will checkout the feature branch

Hotfix Branch:
git hf hotfix start "release-version" - this will create a hotfix branch on origin␣
↪and local will be based off the latest develop branch (make sure to git hf update␣
↪before or you get an error if local develop and remote devleop have divereged)
```

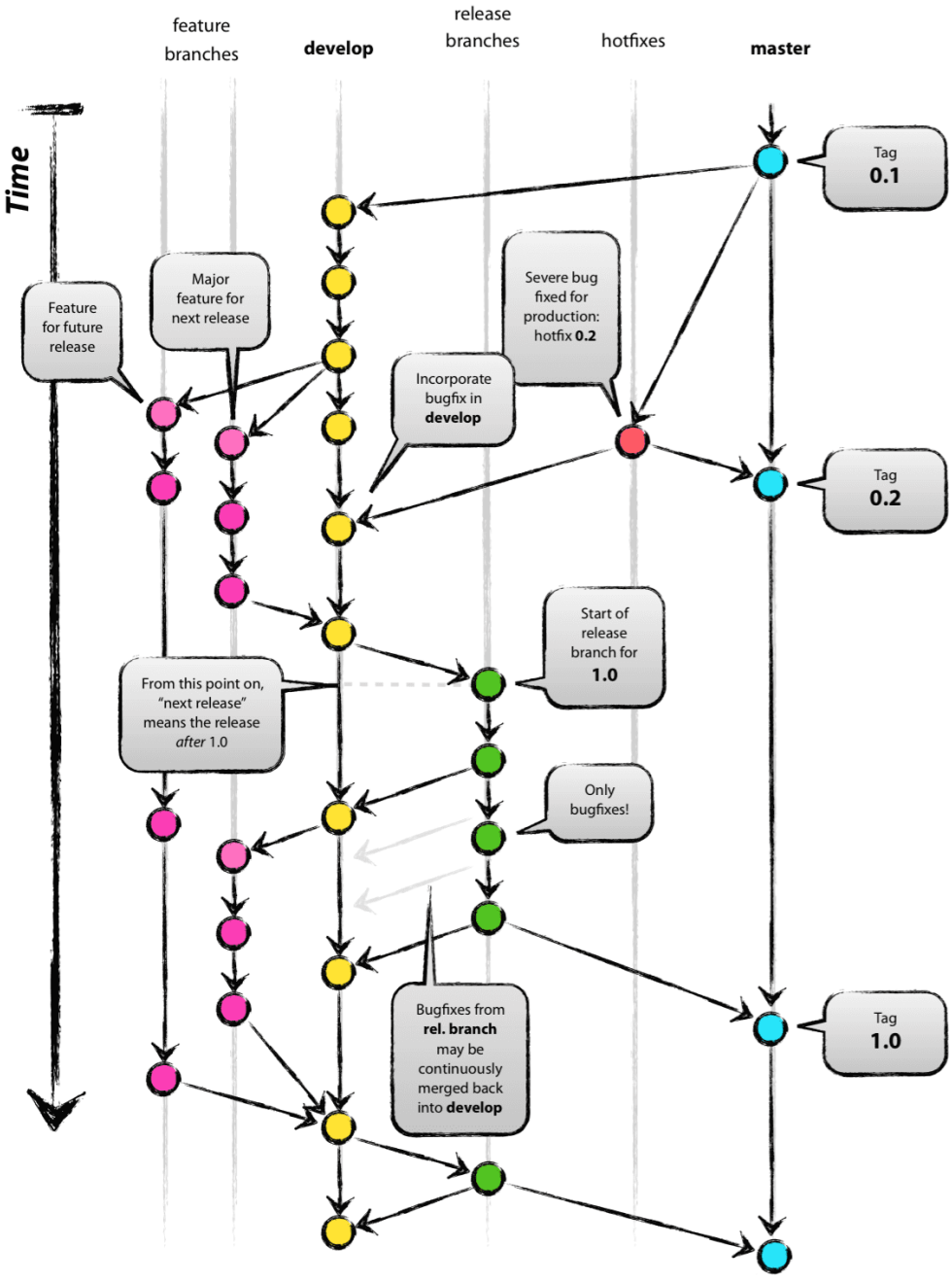


Fig. 5.1: Vincent Driessen Git branching model

```
git hf hotfix finish "release-version" - this will delete the local and remote␣
↪branches and merge the commits of the hotfix branch into master and develop␣
↪branches - it will also create a release tag that matches the release version on␣
↪master
git hf hotfix cancel -f "release-version" - this will delete the remote and local␣
↪branch (only do this if the hotfix was created in error)
git checkout hotfix/"release-version" - this will checkout the hotfix branch (make␣
↪sure to git hf update first)

Release Branch:
git hf release start "release-version" - this will create a release branch on origin␣
↪and local will be based off the latest develop branch (make sure to git hf update␣
↪before or you get an error if local develop and remote devleop have divereged)
git hf release finish "release-version" - this will delete the local and remote␣
↪branches and merge the commits of the release branch both into develop and master -␣
↪it will also create a release tag that matches the release version on master
git hf release cancel -f "release-version" - this will delete the local and remote␣
↪branch (only do this if the release was created in error)
git checkout release/"release-version" - this will checkout the release branch (make␣
↪sure to git hf update first)

Preparing a PR:
- put the Aha! Ticket # in PR title with a description
- assign to the proper reviewer
- don't squash the commits until after reviewed
- after review - squash the commits

Squashing Commits:
- checkout the branch you want to squash
- git merge-base "my-branch" develop (returns merge-base-hash)
- git rebase -i "merge-base-hash"
- change all commit types to "squash" from "pick" in the text file (except first) &␣
↪save file
- if you get a no-op message in the text file and still have multiple commits then␣
↪use the command git rebase -i (without the hash)
- fix any merge conflicts
- you should have one commit
- force update your remote branch: git push origin "my-branch" -f

Resolving merge conflicts with the develop branch that are not squashing related␣
↪(generally on PRs - auto-merge will show as disabled):
- git hf update
- git rebase develop (while in your branch)
- resolve any merge conflicts

Rules to remember:
- don't ever git merge branches together manually (should never run command - git␣
↪merge)
- squash only after review and before merging PR into develop
```

Note: There is a large body of references on Git that are constantly being updated in the Software Development>Git section of Dave Dittrich's web page.

Caution: Mac OS X (by default) uses a *case insensitive* HFS file system. Unlike Ubuntu and other Linux/Unix distributions using case-sensitive file systems like `ext2`, `reiserfs`, etc., the default OS X file system does not care if you name a file `THISFILE` or `ThisFile` or `thisfile`. All of those refer to the same file on a Mac. This can cause problems when you use Git to share a source repository between computers running OS X, Windows, and/or Linux, because what Linux thinks are two files, the Mac only thinks is one (and that really causes problems for Git).

See Git on Mac OS X: Don't ignore case! and How do I commit case-sensitive only filename changes in Git?. A solution for Mac OS X, posted in Case sensitivity in Git, is documented in Section *Using a Case-Sensitive sparse image on Mac OS X*.

5.3 Global Git Configuration

As we learn about best practices, the following set of global configuration settings will be updated. Refer back to this page, or look in the `dims-git` repo, for the latest configuration examples.

The following are user-specific settings that you should alter for your own account and preferences of editor/merge method:

```
$ git config --global user.name "Dave Dittrich"
$ git config --global user.email "dittrich@u.washington.edu"
$ git config --global merge.tool vimdiff
$ git config --global core.editor vim
```

Caution: There is a bad side-effect of the way the initial common Git configuration were managed using Ansible. Whenever the `dims-users-create` role was played, a fresh copy of the user's global Git configuration file (`~/.gitconfig`) is created, over-writing whatever the user had created by issuing the commands above and forcing the user to have to re-issue those commands every time the play was run. (See the file `$GIT/ansible-playbooks/dims-users-create/templates/gitconfig.j2`). That is a bug in that it is not idempotent.

One quick hack that restores these values is to add those commands to your `$HOME/.bash_aliases` file, which is run every time a new interactive Bash shell is started.

A better long-term solution, which we are working towards, is to have the `user.name` and `user.email` configuration settings come from the Trident portal user attributes table, so they can be set by the user and stored in one central location, which can then be retreived from the Trident user database and applied consistently by Ansible when it sets up user accounts.

The following are general and can be applied to anyone's configuration (included here without a prompt so you can cut/paste to a command line):

```
git config --global push.default tracking
git config --global core.excludesfile ~/.gitignore_global
git config --global core.autocrlf false
git config --global color.diff auto
git config --global color.status auto
git config --global color.branch auto
git config --global color.interactive auto
git config --global color.ui auto
git config --global branch.autosetuprebase always
```

The following are convenience aliases that help with certain tasks:

```
git config --global alias.find 'log --color -p -S'
git config --global alias.stat 'status -s'
git config --global alias.unstage "reset HEAD --"
git config --global alias.uncommit "reset --soft HEAD^"
git config --global alias.gr 'log --full-history --decorate=short --all --color --
↪graph'
git config --global alias.lg 'log --oneline --decorate=short --abbrev-commit --all --
↪color --graph'
git config --global alias.log1 'log --oneline --decorate=short'
```

5.4 Daily tasks with Git

This section covers regular tasks that are performed to work with source code using Git. This section assumes you are using the `hub flow` tool described in Section installingtools.

Warning: These tools are being installed in the `dimsenv` Python virtual environment to make it easier for everyone on the team to access them and to stay up to date with instructions in this document. If you have *any* problems, file a Jira ticket or talk to Dave immediately upon encountering a problem. Do not let yourself get blocked on something and block everyone else as a result!

5.4.1 Updating local repos

The most common task you need to do is keep your local Git repos up to date with the code that others have pushed to remote repositories for sharing. With several dozen individual Git repos, keeping your system up to date with all of these frequently changing repos using `git` commands alone is difficult.

To make things easier, helper programs like the `hubflow` scripts and `mr` can be used, but even those programs have their limits.

The preferred method of updating the larger set of DIMS Git repos is to use `dims.git.syncrepos`, which in turn calls `hubflow` via `mr` as part of its processing. This convenience script (described in Section *Updating with dims.git.syncrepos*) works on many repos at once, saving time and effort.

You should still learn how `hubflow` and `mr` work, since you will need to use them to update individual Git repos when you are working within those repos, so we will start with those tools.

Updating using `hubflow`

The following command ensures that a local repo you are working on is up to date:

Note: The list of actions that are performed is provided at the end of the command output. This will remind you of what all is happening under the hood of Hub Flow and is well worth taking a few seconds of your attention.

```
$ git hf update
Fetching origin
remote: Counting objects: 187, done.
remote: Compressing objects: 100% (143/143), done.
remote: Total 165 (delta 56), reused 1 (delta 0)
Receiving objects: 100% (165/165), 31.78 KiB | 0 bytes/s, done.
```

```
Resolving deltas: 100% (56/56), completed with 13 local objects.
From git.devops.develop:/opt/git/ansible-playbooks
   001ba47..0e12ec3  develop    -> origin/develop
 * [new branch]      feature/dims-334 -> origin/feature/dims-334
Updating 001ba47..0e12ec3
Fast-forward
 docs/source/conf.py                        | 2 +-
 roles/dims-ci-utils-deploy/tasks/main.yml | 5 +++++
 2 files changed, 6 insertions(+), 1 deletion(-)

Summary of actions:
- Any changes to branches at origin have been downloaded to your local repository
- Any branches that have been deleted at origin have also been deleted from your␣
↪local repository
- Any changes from origin/master have been merged into branch 'master'
- Any changes from origin/develop have been merged into branch 'develop'
- Any resolved merge conflicts have been pushed back to origin
- You are now on branch 'develop'
```

If a branch existed on the remote repo (e.g., the `feature/eliot` branch used in testing), it would be deleted:

```
$ git branch -a
* develop
  master
  remotes/origin/develop
  remotes/origin/feature/eliot
  remotes/origin/master
$ git hf update
Fetching origin
From git.devops.develop:/opt/git/dims-asbuilt
 x [deleted]         (none)     -> origin/feature/eliot

Summary of actions:
- Any changes to branches at origin have been downloaded to your local repository
- Any branches that have been deleted at origin have also been deleted from your␣
↪local repository
- Any changes from origin/master have been merged into branch 'master'
- Any changes from origin/develop have been merged into branch 'develop'
- Any resolved merge conflicts have been pushed back to origin
- You are now on branch 'develop'
$ git branch -a
* develop
  master
  remotes/origin/develop
  remotes/origin/master
```

While using `git hf update && git hf pull` seems like it is simple enough, the DIMS project has several dozen repos, many of which are inter-related. Keeping them all up to date is not simple, and because of this developers often get far out of sync with the rest of the team.

Updating using the `mr` command

A useful tool for managing multiple Git repositories and keeping them in sync with the master branches is to use the program mr.

`mr` uses a configuration file that can be added to using `mr register` within a repo, or by editing/writing the `.mrconfig` file directly.

Attention: These instructions assume the reader is *not already using* `mr` on a regular basis. Additionally, all DIMS Git repos are assumed to be segrated into their own directory tree apart from any other Git repos that the developer may be using.

This assumption allows for use of a `.mrconfig` file specifically for just DIMS source code that can be overwritten entirely with DIMS-specific settings.

Cloning all of the DIMS source repos at once, or getting the contents of what should be an up-to-date `.mrconfig` file, is covered in the Section *Cloning multiple repos from git.devops.develop*.

After all repos have been cloned, they can be kept up to date on a daily basis. Start your work session with the following commands:

```
$ cd $GIT
$ mr update
```

Caution: If you do not update a repo before attempting to `git hf push` or `git hf update` with commited changes, Git will do a `pull` and potentially you will end up with at best a `merge`, and at worst a merge conflict that you must resolve before the `push` can complete. If you are not comfortable handling a merge conflict, talk to another team member to get help.

Updating with `dims.git.syncrepos`

A script that combines several of the above steps into one single command is `dims.git.synrepos`.

```
[dimsenv] dittrich@dimsdemo1:~ () $ dims.git.syncrepos --version
dims.git.syncrepos version 1.6.97
```

In the example here, highlighted lines show where repos are *dirty* (Repo[9], Repo[13], and Repo[33]), meaning they have tracked files that are not committed yet and cannot be updated, *clean* and requiring updates from the remote repo (Repo[12]), and new repositories from the remote server (Repo[28] and Repo[30]) that are being cloned and initialized for use with *hub-flow* tools. At the end, `dims.git.syncrepos` reports how many repos were updated out of the available repos on the remote server, how many new repos it added, and/or how many repos could not be updated because they are dirty. Lastly, it reports how long it took (so you can be aware of how long you have to go get coffee after starting a sync.)

```
 1  [dimsenv] dittrich@dimsdemo1:~ () $ dims.git.syncrepos
 2  [+++] Found 46 available repos at git@git.devops.develop
 3  [+++] Repo[1] "/home/dittrich/dims/git/ansible-inventory" clean:
 4  [+++] Repo[2] "/home/dittrich/dims/git/ansible-playbooks" clean:
 5  [+++] Repo[3] "/home/dittrich/dims/git/cif-client" clean:
 6  [+++] Repo[4] "/home/dittrich/dims/git/cif-java" clean:
 7  [+++] Repo[5] "/home/dittrich/dims/git/configs" clean:
 8  [+++] Repo[6] "/home/dittrich/dims/git/dims" clean:
 9  [+++] Repo[7] "/home/dittrich/dims/git/dims-ad" clean:
10  [+++] Repo[8] "/home/dittrich/dims/git/dims-asbuilt" clean:
11  [---] Repo[9] "/home/dittrich/dims/git/dims-ci-utils" is dirty:
12  ?? dims/diffs.1
13  ?? dims/manifest.dat
14  ?? ubuntu-14.04.2/ubuntu-14.04.3-install.dd.bz2
15  4bb5516 (feature/dims-406) Merge branch 'develop' into feature/dims-406
16
17  [+++] Repo[10] "/home/dittrich/dims/git/dims-dashboard" clean:
```

```
[+++] Repo[11] "/home/dittrich/dims/git/dims-db-recovery" clean:
[+++] Repo[12] "/home/dittrich/dims/git/dims-devguide" clean:
remote: Counting objects: 29, done.
remote: Compressing objects: 100% (22/22), done.
remote: Total 22 (delta 13), reused 0 (delta 0)
Unpacking objects: 100% (22/22), done.
From git.devops.develop:/opt/git/dims-devguide
   daffa68..4b2462b  develop    -> origin/develop
Updating daffa68..4b2462b
Fast-forward
 .bumpversion.cfg                |  2 +-
 docs/source/conf.py             |  4 ++--
 docs/source/deployconfigure.rst |  2 +-
 docs/source/referenceddocs.rst  | 13 ++++++++++++
 4 files changed, 17 insertions(+), 4 deletions(-)
[---] Repo[13] "/home/dittrich/dims/git/dims-dockerfiles" is dirty:
8a47fca (HEAD -> develop) Bump version: 1.1.11 → 1.1.12

[+++] Repo[14] "/home/dittrich/dims/git/dims-dsdd" clean:
[+++] Repo[15] "/home/dittrich/dims/git/dims-jds" clean:
[+++] Repo[16] "/home/dittrich/dims/git/dims-keys" clean:
[+++] Repo[17] "/home/dittrich/dims/git/dims-ocd" clean:
[+++] Repo[18] "/home/dittrich/dims/git/dims-packer" clean:
[+++] Repo[19] "/home/dittrich/dims/git/dims-sample-data" clean:
[+++] Repo[20] "/home/dittrich/dims/git/dims-sr" clean:
[+++] Repo[21] "/home/dittrich/dims/git/dims-supervisor" clean:
[+++] Repo[22] "/home/dittrich/dims/git/dims-svd" clean:
[+++] Repo[23] "/home/dittrich/dims/git/dimssysconfig" clean:
[+++] Repo[24] "/home/dittrich/dims/git/dims-tp" clean:
[+++] Repo[25] "/home/dittrich/dims/git/dims-tr" clean:
[+++] Repo[26] "/home/dittrich/dims/git/dims-vagrant" clean:
[+++] Repo[27] "/home/dittrich/dims/git/ELK" clean:
[+++] Adding Repo[28] fuse4j to /home/dittrich/dims/.mrconfig and checking it out.
mr checkout: /home/dittrich/dims/git/fuse4j
Cloning into 'fuse4j'...
remote: Counting objects: 523, done.
remote: Compressing objects: 100% (240/240), done.
remote: Total 523 (delta 186), reused 523 (delta 186)
Receiving objects: 100% (523/523), 180.86 KiB | 0 bytes/s, done.
Resolving deltas: 100% (186/186), done.
Checking connectivity... done.
Using default branch names.

Which branch should be used for tracking production releases?
   - master
Branch name for production releases: [master]
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []

mr checkout: finished (1 ok; 43 skipped)
[+++] Repo[29] "/home/dittrich/dims/git/ipgrep" clean:
[+++] Adding Repo[30] java-native-loader to /home/dittrich/dims/.mrconfig and␣
↪checking it out.
```

```
mr checkout: /home/dittrich/dims/git/java-native-loader
Cloning into 'java-native-loader'...
remote: Counting objects: 329, done.
remote: Compressing objects: 100% (143/143), done.
remote: Total 329 (delta 62), reused 329 (delta 62)
Receiving objects: 100% (329/329), 178.36 KiB | 0 bytes/s, done.
Resolving deltas: 100% (62/62), done.
Checking connectivity... done.
Using default branch names.

Which branch should be used for tracking production releases?
   - master
Branch name for production releases: [master]
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []

mr checkout: finished (1 ok; 44 skipped)
[+++] Repo[31] "/home/dittrich/dims/git/java-stix-v1.1.1" clean:
[+++] Repo[32] "/home/dittrich/dims/git/mal4s" clean:
[---] Repo[33] "/home/dittrich/dims/git/MozDef" is dirty:
 M docker/Dockerfile
 M docker/Makefile

[+++] Repo[34] "/home/dittrich/dims/git/ops-trust-openid" clean:
[+++] Repo[35] "/home/dittrich/dims/git/ops-trust-portal" clean:
[+++] Repo[36] "/home/dittrich/dims/git/poster-deck-2014-noflow" clean:
[+++] Repo[37] "/home/dittrich/dims/git/prisem" clean:
[+++] Repo[38] "/home/dittrich/dims/git/prisem-replacement" clean:
[+++] Repo[39] "/home/dittrich/dims/git/pygraph" clean:
[+++] Repo[40] "/home/dittrich/dims/git/rwfind" clean:
[---] Repo[41] "/home/dittrich/dims/git/sphinx-autobuild" is clean:
[+++] Repo[42] "/home/dittrich/dims/git/stix-java" clean:
[+++] Repo[43] "/home/dittrich/dims/git/ticketing-redis" clean:
[+++] Repo[44] "/home/dittrich/dims/git/tsk4j" clean:
[+++] Repo[45] "/home/dittrich/dims/git/tupelo" clean:
[+++] Repo[46] "/home/dittrich/dims/git/umich-botnets" clean:
[+++] Updated 40 of 46 available repos.
[+++] Summary of actions for repos that were updated:
- Any changes to branches at origin have been downloaded to your local repository
- Any branches that have been deleted at origin have also been deleted from your␣
↪local repository
- Any changes from origin/master have been merged into branch 'master'
- Any changes from origin/develop have been merged into branch 'develop'
- Any resolved merge conflicts have been pushed back to origin
[+++] Added 3 new repos: fuse4j java-native-loader tsk4j
[+++] Could not update 3 repos: dims-ci-utils dims-dockerfiles MozDef
[+++] Updating repos took 00:04:12
```

5.4.2 Finding Changes and Changed Files

When working with lots of branches, especially branches that last for a long time, it becomes difficult to find changes and changed files (and the commits that contain them), in order to ensure required changes are present on the branch you are working on.

Git has a command `whatchanged` that assists with this. (See the git-whatchanged documentation.)

Say you are on a branch, and running a program that relies on an Ansible inventory file for obtaining a list of hosts. When the program is run, it appears to iterate over the *group names*, not host names as you expect:

```
$ test.vagrant.list --status
production: not_created
development: not_created
```

The reason for this is a missing `:children` modifier on the names of a group that has sub-groups. You know this bug was fixed, but which branch (or which commit) contains the fix? Use `git whatchanged` and pass it the name of the file that has the problem.

```
 $ git whatchanged -- inventory/deployment/all
commit 963b006a7aceee21eb35da41546ae5da7596382e
Author: Dave Dittrich <dittrich@u.washington.edu>
Date:   Wed Dec 14 23:07:37 2016 -0800

    Add missing ":children" modifier

:100644 100644 d9918d0... 9ce596b... M  v2/inventory/deployment/all

commit 00afbb5bfadc46ef9b5f253a13a6212cb3fca178
Author: Dave Dittrich <dittrich@u.washington.edu>
Date:   Sun Dec 11 21:31:04 2016 -0800

    Update and normalize inventory 'all' files with children groups

:100644 100644 99bb8d9... d9918d0... M  v2/inventory/deployment/all

commit 777cce71f944650c0ff5cf47723ee6b9f322c987
Author: Dave Dittrich <dittrich@u.washington.edu>
Date:   Sat Dec 3 21:33:38 2016 -0800

    Refactor inventory directories to use group vars properly

:100644 100644 98376da... 99bb8d9... M  v2/inventory/deployment/all

commit 3cdb37d04c9d8bedb5277ad4cfbeafdec55f69b0
Author: Dave Dittrich <dittrich@u.washington.edu>
Date:   Tue Nov 22 20:00:19 2016 -0800

    Add vagrants to local and deployment groups

:100644 100644 b199ae1... 98376da... M  v2/inventory/deployment/all

commit 92eec6c03c28824725b9fc0c4560b4fdccfa880e
Author: Dave Dittrich <dittrich@u.washington.edu>
Date:   Fri Nov 18 16:53:04 2016 -0800

    Add initial inventory for deployment to get dynamic inventory working

:000000 100644 0000000... b199ae1... A  v2/inventory/deployment/all
```

If you add the `--patch` option, you also see the changes themselves and can identify the initial problem file as well as the commit that contains the fix (output edited for brevity):

```
 $ git whatchanged --patch -- inventory/deployment/all
commit 963b006a7aceee21eb35da41546ae5da7596382e
Author: Dave Dittrich <dittrich@u.washington.edu>
Date:   Wed Dec 14 23:07:37 2016 -0800

    Add missing ":children" modifier

diff --git a/v2/inventory/deployment/all b/v2/inventory/ectf/all
index d9918d0..9ce596b 100644
--- a/v2/inventory/deployment/all
+++ b/v2/inventory/deployment/all
@@ -27,7 +27,7 @@ red.devops.deployment
 yellow.devops.deployment

 # Hosts are Vagrant virtual machines
-[vagrants]
+[vagrants:children]
 production
 development

commit 00afbb5bfadc46ef9b5f253a13a6212cb3fca178
Author: Dave Dittrich <dittrich@u.washington.edu>
Date:   Sun Dec 11 21:31:04 2016 -0800

    Update and normalize inventory 'all' files with children groups

diff --git a/v2/inventory/deployment/all b/v2/inventory/ectf/all
index 99bb8d9..d9918d0 100644
--- a/v2/inventory/deployment/all
+++ b/v2/inventory/deployment/all
  . . .
 [manager]
 core-[01:03].devops.deployment

+[worker]
+red.devops.deployment
+yellow.devops.deployment
+
 # Hosts are Vagrant virtual machines
 [vagrants]
+production
+development
+
+[production]
 red.devops.deployment
 core-[01:03].devops.deployment
 yellow.devops.deployment
-blue16.devops.deployment
+
+[development]
 blue14.devops.deployment
-green.devops.deployment
 . . .
```

You can now `git cherry-pick` the commit with the fix (`963b006a7aceee21eb35da41546ae5da7596382e`) and move on:

```
$ test.vagrant.list --status
red: saved
core-01: not_created
core-02: not_created
core-03: not_created
yellow: saved
blue14: not_created
```

5.4.3 Managing Version Numbers

The DIMS project uses the Python program bumpversion to update version numbers in Git repositories, following PEP 440 – Version Identification and Dependency Specification. You can learn how bumpversion works from these resources:

- bumpversion screencast showing bumpversion in action
- A Python Versioning Workflow With Bumpversion

Note: You can find examples of using bumpversion (including its configuration file `.bumpversion.cfg` and how it is used to manage version numbers in files) in this document in Section *Cherry-picking a commit from one branch to another*.

The program `bumpversion` is included in the Python virtual environment `dimsenv` that is created for use in DIMS development.

```
[dimsenv] dittrich@27b:~/git/homepage (develop*) $ which bumpversion
/Users/dittrich/dims/envs/dimsenv/bin/bumpversion
```

Caution: Because you must be in the same directory as the `.bumpversion.cfg` file when you invoke `bumpversion`, it is sometimes problematic when using it to work in a sub-directory one or more levels below the configuration file. You may see example command lines like `(cd ..; bumpversion patch)` that use sub-shells to temporarily change to the right directory, do the `bumpversion patch`, then exit leaving you in the same directory where you are editing files. That is a little more work than is desirable, but doing a bunch of `cd ..`, `bumpersion patch`, `cd backagain` is even more work.

To make it easier to increment version numbers, a helper script `dims.bumpversion` is available as well:

```
[dimsenv] dittrich@27b:~/git/homepage (develop*) $ which dims.bumpversion
/Users/dittrich/dims/envs/dimsenv/bin/dims.bumpversion
[dimsenv] dittrich@27b:~/git/homepage (develop*) $ dims.bumpversion --help
Usage:
/Users/dittrich/dims/envs/dimsenv/bin/dims.bumpversion [options] [args]

Use "/Users/dittrich/dims/envs/dimsenv/bin/dims.bumpversion --help" to see options.
Use "/Users/dittrich/dims/envs/dimsenv/bin/dims.bumpversion --usage" to see help on
↪"bumpversion" itself.

/Users/dittrich/dims/envs/dimsenv/bin/dims.bumpversion -- [bumpversion_options]␣
↪[bumpversion_args]
```

```
Follow this second usage example and put -- before any bumpversion
options and arguments to pass them on bumpversion (rather than
process them as though they were /Users/dittrich/dims/envs/dimsenv/bin/dims.
↪bumpversion arguments.) After
all, /Users/dittrich/dims/envs/dimsenv/bin/dims.bumpversion is just a shell wrapping␣
↪bumpversion.


Options:
  -h, --help     show this help message and exit
  -d, --debug    Enable debugging
  -u, --usage    Print usage information.
  -v, --verbose  Be verbose (on stdout) about what is happening.
```

The default when you just invoke `dims.bumpversion` is to do `bumpversion patch`, the most frequent version increment. To use a different increment, just add it as an argument on the command line (e.g., `dims.bumpvesion minor`).

Here is an example of how this section edit was done, showing the version number increment in the workflow:

```
 [dimsenv] dittrich@localhost:~/dims/git/dims-devguide/docs (develop*) $ git add␣
↪source/sourcemanagement.rst
 [dimsenv] dittrich@localhost:~/dims/git/dims-devguide/docs (develop*) $ git stat
 M  docs/source/sourcemanagement.rst
 [dimsenv] dittrich@localhost:~/dims/git/dims-devguide/docs (develop*) $ git commit -
↪m "Add subsection on version numbers and bumpversion/dims.bumpversion"
 [develop b433bae] Add subsection on version numbers and bumpversion/dims.bumpversion
  1 file changed, 92 insertions(+)
 [dimsenv] dittrich@localhost:~/dims/git/dims-devguide/docs (develop*) $ dims.
↪bumpversion
 [dimsenv] dittrich@localhost:~/dims/git/dims-devguide/docs (develop*) $ git hf push
 Fetching origin
 Already up-to-date.
 Counting objects: 11, done.
 Delta compression using up to 8 threads.
 Compressing objects: 100% (11/11), done.
 Writing objects: 100% (11/11), 2.53 KiB | 0 bytes/s, done.
 Total 11 (delta 7), reused 0 (delta 0)
 remote: Running post-receive hook: Thu Oct 22 22:31:50 PDT 2015
 remote: [+++] post-receive-06jenkinsalldocs started
 remote: [+++] REPONAME=dims-devguide
 remote: [+++] BRANCH=develop
 remote: [+++] newrev=00727d53dbc8130cdbdbe35be80f1f4c2d2ee7fa
 remote: [+++] oldrev=e8e7d4db40dd852a044525fdfbada1fe80d81739
 remote: [+++] Branch was updated.
 remote: [+++] This repo has a documentation directory.
 remote:   % Total    % Received % Xferd  Average Speed   Time    Time     Time  ␣
↪Current
 remote:                                  Dload  Upload   Total   Spent    Left  Speed
 remote: 100    79    0     0  100    79      0   2613 --:--:-- --:--:-- --:--:--  ␣
↪3761
 remote:   % Total    % Received % Xferd  Average Speed   Time    Time     Time  ␣
↪Current
 remote:                                  Dload  Upload   Total   Spent    Left  Speed
 remote: 100    78    0     0  100    78      0   2524 --:--:-- --:--:-- --:--:--  ␣
↪3250
 remote: [+++] post-receive-06jenkinsalldocs finished
 To git@git.devops.develop:/opt/git/dims-devguide.git
```

```
   e8e7d4d..00727d5  develop -> develop

Summary of actions:
- The remote branch 'origin/develop' was updated with your changes
```

5.4.4 Initializing a repo for `hub-flow`

Every time you clone a new DIMS repo, it must be initialized with `hub-flow` so that `hub-flow` commands work properly. Initialize your repo this way:

```
$ git clone git@git.devops.develop:/opt/git/dims-ad.git
Cloning into 'dims-ad'...
remote: Counting objects: 236, done.
remote: Compressing objects: 100% (155/155), done.
remote: Total 236 (delta 117), reused 159 (delta 76)
Receiving objects: 100% (236/236), 26.20 MiB | 5.89 MiB/s, done.
Resolving deltas: 100% (117/117), done.
Checking connectivity... done.
$ cd dims-ad
$ git hf init
Using default branch names.

Which branch should be used for tracking production releases?
   - master
Branch name for production releases: [master]
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
```

After initializing `hub-flow`, there will be two new sections in your `.git/config` file starting with `hubflow`:

```
$ cat .git/config
[core]
        repositoryformatversion = 0
        filemode = true
        bare = false
        logallrefupdates = true
        ignorecase = true
        precomposeunicode = true
[remote "origin"]
        url = git@git.devops.develop:/opt/git/dims-ad.git
        fetch = +refs/heads/*:refs/remotes/origin/*
[branch "master"]
        remote = origin
        merge = refs/heads/master
        rebase = true
[hubflow "branch"]
        master = master
        develop = develop
[branch "develop"]
        remote = origin
```

```
        merge = refs/heads/develop
        rebase = true
[hubflow "prefix"]
        feature = feature/
        release = release/
        hotfix = hotfix/
        support = support/
        versiontag =
```

Note: A possible test for inclusion in the `dims-ci-utils` test suite would be to check for the existance of the `hubflow "branch"` and `hubflow "prefix"` sections.

These are automatically created when repos are checked out using the `dims.git.syncrepos` script and/or methods involving `mr` described in the following sections.

5.5 Infrequent tasks with Git

5.5.1 Cloning multiple repos from `git.devops.develop`

There are several dozen repositories on `git.devops.develop` that contain DIMS-generated code, configuration files, and/or documentation, but also local copies of Git repositories from other sources (some with DIMS-related customizations).

To get a list of all repositories on `git.devops.develop`, use the Git shell command `list`:

```
$ ssh git@git.devops.develop list
prisem-replacement.git
ELK.git
cif-java.git
cif-client.git
dims-ad.git
supervisor.git
dims-tr.git
lemonldap-ng.git
pygraph.git
parsons-docker.git
configs.git
poster-deck-2014-noflow.git
dims-keys.git
dims.git
dims-tp.git
ops-trust-portal.git
dimssysconfig.git
dims-dockerfiles.git
stix-java.git
ansible-playbooks.git
dims-dashboard.git
mal4s.git
dims-ocd.git
sphinx-autobuild.git
dims-devguide.git
dims-asbuilt.git
ticketing-redis.git
```

```
dims-sr.git
prisem.git
umich-botnets.git
dims-dsdd.git
dims-sample-data.git
packer.git
java-stix-v1.1.1.git
vagrant.git
dims-jds.git
ansible-inventory.git
ops-trust-openid.git
dims-coreos-vagrant.git
configstest.git
poster-deck-2014.git
rwfind.git
dims-ci-utils.git
ipgrep.git
tupelo.git
dims-opst-portal.git
lemonldap-dims.git
MozDef.git
tsk4j.git
dims-svd.git
```

To clone all of these repositories in a single step, there is another Git shell command `mrconfig` that returns the contents of a `.mrconfig` file (see `man mr` for more information).

Caution: To use a `.mrconfig` file in a an arbitrary directory, you will need to add the directory path to this file to the `~/.mrtrust` file. In this example, we will clone repos into `~/dims/git` by placing the `.mrconfig` file in the `~/dims` directory.

```
$ cat ~/.mrtrust
/Users/dittrich/dims/.mrconfig
/Users/dittrich/git/.mrconfig
```

If you are building a documentation set (i.e., a limited set of documentation-only repositories that are cross-linked using the `intersphinx` extension to Sphinx as described in Section *Cross-referencing between documents with the sphinx.ext.intersphinx extension*.

```
$ cd ~/dims
$ ssh git@git.devops.develop mrconfig dims-ad dims-sr dims-ocd
[git/dims-ad]
checkout = git clone 'git@git.devops.develop:/opt/git/dims-ad.git' 'dims-ad' &&
        (cd dims-ad; git hf init)
show = git remote show origin
update = git hf update
pull = git hf update &&
        git hf pull
stat = git status -s

[git/dims-sr]
checkout = git clone 'git@git.devops.develop:/opt/git/dims-sr.git' 'dims-sr' &&
        (cd dims-sr; git hf init)
show = git remote show origin
update = git hf update
pull = git hf update &&
```

```
        git hf pull
stat = git status -s

[git/dims-ocd]
checkout = git clone 'git@git.devops.develop:/opt/git/dims-ocd.git' 'dims-ocd' &&
        (cd dims-ocd; git hf init)
show = git remote show origin
update = git hf update
pull = git hf update &&
        git hf pull
stat = git status -s
$ ssh git@git.devops.develop mrconfig dims-ad dims-sr dims-ocd > .mrconfig
$ mr checkout
mr checkout: /Users/dittrich/dims/git/dims-ad
Cloning into 'dims-ad'...
remote: Counting objects: 518, done.
remote: Compressing objects: 100% (437/437), done.
remote: Total 518 (delta 308), reused 155 (delta 76)
Receiving objects: 100% (518/518), 27.88 MiB | 5.88 MiB/s, done.
Resolving deltas: 100% (308/308), done.
Checking connectivity... done.
Using default branch names.

Which branch should be used for tracking production releases?
   - master
Branch name for production releases: [master]
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []

mr checkout: /Users/dittrich/dims/git/dims-ocd
Cloning into 'dims-ocd'...
remote: Counting objects: 474, done.
remote: Compressing objects: 100% (472/472), done.
remote: Total 474 (delta 288), reused 0 (delta 0)
Receiving objects: 100% (474/474), 14.51 MiB | 4.26 MiB/s, done.
Resolving deltas: 100% (288/288), done.
Checking connectivity... done.
Using default branch names.

Which branch should be used for tracking production releases?
   - master
Branch name for production releases: [master]
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []

mr checkout: /Users/dittrich/dims/git/dims-sr
```

```
Cloning into 'dims-sr'...
remote: Counting objects: 450, done.
remote: Compressing objects: 100% (445/445), done.
remote: Total 450 (delta 285), reused 0 (delta 0)
Receiving objects: 100% (450/450), 498.20 KiB | 0 bytes/s, done.
Resolving deltas: 100% (285/285), done.
Checking connectivity... done.
Using default branch names.

Which branch should be used for tracking production releases?
   - master
Branch name for production releases: [master]
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []

mr checkout: finished (3 ok)
$ mr stat
mr stat: /Users/dittrich/tmp/dims/git/dims-ad

mr stat: /Users/dittrich/tmp/dims/git/dims-ocd

mr stat: /Users/dittrich/tmp/dims/git/dims-sr

mr stat: finished (3 ok)
```

Note: The example just shown uses only three repos. If you do not specify any repo names on the `mrconfig` Git shell command, it will return the settings for all 50+ DIMS repos. You can then clone the entire set of DIMS repositories with the same `mr checkout` command, and update all of them at once with `mr update`.

5.5.2 Adding a newly-created repository

Until the `dims.git.syncrepos` script has a new feature added to it to detect when a new repo exists on `git.devops.develop` that does not have a local repo associated with it, you must do this yourself.

When someone uses the `newrepo` script to create a new repo on `git.devops.develop`, you will need to get new `.mrconfig` settings for that repo in order for `dims.git.syncrepo` to synchronize it. If you have your `$GIT` environment variable pointing to a directory that *only* has DIMS Git repos in it, you just need to create an updated `.mrconfig` file.

Note: It is safest to get a new copy of the `.mrconfig` file contents and save them to a temporary file that you can compare with the current file to ensure you are getting just what you expect, and only then over-writing the `.mrconfig` file with the new contents. The steps are shown here:

```
$ cd $GIT/..
$ ssh git@git.devops.develop mrconfig > .mrconfig.new
$ diff .mrconfig .mrconfig.new
```

```
324a325,333
> [git/dims-db-recovery]
> checkout = git clone 'git@git.devops.develop:/opt/git/dims-db-recovery.git' 'dims-
↪db-recovery' &&
>     (cd dims-db-recovery; git hf init)
> show = git remote show origin
> update = git hf update
> pull = git hf update &&
>     git hf pull
> stat = git status -s
>
$ mv .mrconfig.new .mrconfig
$ mr checkout
mr checkout: /Users/dittrich/dims/git/dims-db-recovery
Cloning into 'dims-db-recovery'...
remote: Counting objects: 351, done.
remote: Compressing objects: 100% (254/254), done.
remote: Total 351 (delta 63), reused 350 (delta 63)
Receiving objects: 100% (351/351), 7.60 MiB | 5.62 MiB/s, done.
Resolving deltas: 100% (63/63), done.
Checking connectivity... done.
Using default branch names.

Which branch should be used for tracking production releases?
   - master
Branch name for production releases: [master]
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []

mr checkout: finished (1 ok; 43 skipped)
```

5.5.3 Creating Git repositories

As discussed in the introduction to this section, DIMS software will be hosted on both a local server `git.devops.develop` and from github.com/uw-dims. This section covers creation of new repositories on both systems.

Creating repositories on GitHub

Setting up remote Git repositories on `git.devops.develop`

Before a repository can be shared between DIMS team members, a remote repository must be set up on `git.devops.develop` for sharing. The following is an example session creating a new repository named `dims-ocd` for *operational concept description* (a.k.a., *Concept of Operations*).

```
$ slogin git.devops.develop
Welcome to Ubuntu 12.04.5 LTS (GNU/Linux 3.13.0-43-generic x86_64)
[ ... ]
Last login: Sun Jan 11 12:04:36 2015 from lancaster.devops.develop
dittrich@jira:~$ sudo su - gituser
```

```
[sudo] password for dittrich:
git@jira:~$ cd /opt/git
git@jira:/opt/git$ newrepo dims-ocd.git
Initialized empty Git repository in /opt/git/dims-ocd.git/
git@jira:/opt/git$ echo "DIMS Operational Concept Description" > dims-ocd.git/
↪description
git@jira:/opt/git$ tree dims-ocd.git
dims-ocd.git
+- branches
+- config
+- description
+- HEAD
+- hooks
|   +- post-receive -> /opt/git/bin/post-receive
|   +- post-receive-00logamqp -> /opt/git/bin/post-receive-00logamqp
|   +- post-receive-01email -> /opt/git/bin/post-receive-01email
+- info
|   +- exclude
+- objects
|   +- info
|   +- pack
+- refs
    +- heads
    +- tags

9 directories, 7 files
```

As can be seen in the output of `tree` at the end, the steps above only create `post-receive` hooks for logging to AMQP and sending email when a `git push` is done. To add a Jenkins build hook, do the following command as well:

```
git@jira:/opt/git$ ln -s /opt/git/bin/post-receive-02jenkins dims-ocd.git/hooks/post-
↪receive-02jenkins
git@jira:/opt/git$ tree dims-ocd.git/hooks/
dims-ocd.git/hooks/
+- post-receive -> /opt/git/bin/post-receive
+- post-receive-00logamqp -> /opt/git/bin/post-receive-00logamqp
+- post-receive-01email -> /opt/git/bin/post-receive-01email
+- post-receive-02jenkins -> /opt/git/bin/post-receive-02jenkins

0 directories, 4 files
```

Setting up a local Git repository before pushing to remote

After setting up the remote repository, you should create the initial local repository. The basic steps are as follows:

1. Create the new local repo directory.
2. Populate the directory with the files you want in the repo.
3. Add them to the repo.
4. Commit the files with a comment
5. Create an initial version tag.
6. Set `remote.origin.url` to point to the remote repo.
7. Push the new repo to the remote repo.

8. Push the tags to the remote repo.

Here is an edited transcript of performing the above tasks.

```
$ cd $GIT
$ mkdir dims-ocd
$ git init
Initialized empty Git repository in /Users/dittrich/git/.git/
[ ... prepare files ... ]
$ ls
MIL-STD-498-templates.pdf   UW-logo.png                     conf.py
 ↪      newsystem.rst
Makefile                    _build                          currentsystem.rst
 ↪      notes.rst
OCD-DID.pdf                 _static                         impacts.rst
 ↪      operationalscenarios.rst
OCD.html                    _templates                      index.rst
 ↪      referenceddocs.rst
OCD.rst                     analysis.rst                    justifications.rst
 ↪      scope.rst
UW-logo-32x32.ico           appendices.rst                  license.txt
$ rm OCD.rst
$ ls
MIL-STD-498-templates.pdf   _build                          currentsystem.rst
 ↪      notes.rst
Makefile                    _static                         impacts.rst
 ↪      operationalscenarios.rst
OCD-DID.pdf                 _templates                      index.rst
 ↪      referenceddocs.rst
OCD.html                    analysis.rst                    justifications.rst
 ↪      scope.rst
UW-logo-32x32.ico           appendices.rst                  license.txt
UW-logo.png                 conf.py                         newsystem.rst
$ git add .
$ git commit -m "Initial load of MIL-STD-498 template"
[master (root-commit) 39816fa] Initial load of MIL-STD-498 template
 22 files changed, 1119 insertions(+)
 create mode 100644 dims-ocd/MIL-STD-498-templates.pdf
 create mode 100644 dims-ocd/Makefile
 create mode 100644 dims-ocd/OCD-DID.pdf
 create mode 100755 dims-ocd/OCD.html
 create mode 100644 dims-ocd/UW-logo-32x32.ico
 create mode 100644 dims-ocd/UW-logo.png
 create mode 100644 dims-ocd/_build/.gitignore
 create mode 100644 dims-ocd/_static/.gitignore
 create mode 100644 dims-ocd/_templates/.gitignore
 create mode 100644 dims-ocd/analysis.rst
 create mode 100644 dims-ocd/appendices.rst
 create mode 100644 dims-ocd/conf.py
 create mode 100644 dims-ocd/currentsystem.rst
 create mode 100644 dims-ocd/impacts.rst
 create mode 100644 dims-ocd/index.rst
 create mode 100644 dims-ocd/justifications.rst
 create mode 100644 dims-ocd/license.txt
 create mode 100644 dims-ocd/newsystem.rst
 create mode 100644 dims-ocd/notes.rst
 create mode 100644 dims-ocd/operationalscenarios.rst
 create mode 100644 dims-ocd/referenceddocs.rst
 create mode 100644 dims-ocd/scope.rst
```

```
$ git tag -a "2.0.0" -m "Initial template release"
$ git remote add origin git@git.devops.develop:/opt/git/dims-ocd.git
$ git push -u origin master
Counting objects: 24, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (22/22), done.
Writing objects: 100% (24/24), 251.34 KiB | 0 bytes/s, done.
Total 24 (delta 1), reused 0 (delta 0)
remote: Running post-receive hook: Thu Jan 15 20:46:33 PST 2015
To git@git.devops.develop:/opt/git/dims-ocd.git
 * [new branch]      master -> master
Branch master set up to track remote branch master from origin by rebasing.
$ git push origin --tags
Counting objects: 1, done.
Writing objects: 100% (1/1), 173 bytes | 0 bytes/s, done.
Total 1 (delta 0), reused 0 (delta 0)
remote: Running post-receive hook: Thu Jan 15 20:46:45 PST 2015
To git@git.devops.develop:/opt/git/dims-ocd.git
 * [new tag]         2.0.0 -> 2.0.0
```

5.5.4 Cherry-picking a commit from one branch to another

There are times when you are working on one branch (e.g., `feature/coreos`) and find that there is a bug due to a missing file. This file should be on the `develop` branch from which this feature branch was forked, so the solution is to fix the bug on the `develop` branch and also get the fix on the feature branch.

As long as that change (e.g., an added file that does not exist on the branch) has no chance of a conflict, a simple `cherry-pick` of the commit will get things synchronized. Here is an example of the steps:

Let's say the bug was discovered by noticing this error message shows up when rendering a Sphinx document using `sphinx-autobuild`:

```
+--------- source/index.rst changed --------------------------------------------
/Users/dittrich/git/dims-ci-utils/docs/source/lifecycle.rst:306: WARNING: External␣
↪Graphviz file u'/Users/dittrich/git/dims-ci-utils/Makefile.dot' not found or␣
↪reading it failed
+------------------------------------------------------------------------------
```

The file `Makefile.dot` is not found. (The reason is that the `lifecycle.rst` file was moved from a different place, but the file it included was not.) We first stash our work (if necessary) and check out the develop branch. Next, locate the missing file:

```
$ git checkout develop
Switched to branch 'develop'
Your branch is up-to-date with 'origin/develop'.
$ find ../.. -name 'Makefile.dot'
../../packer/Makefile.dot
```

We now copy the file to where we believe it should reside, and to trigger a new `sphinx-autobuild`, we touch the file that includes it:

```
$ cp ../../packer/Makefile.dot ..
$ touch source/lifecycle.rst
```

Switching to the `sphinx-autobuild` status window, we see the error message has gone away.

```
+--------- source/lifecycle.rst changed ----------------------------------------
+-------------------------------------------------------------------------------

[I 150331 16:40:04 handlers:74] Reload 1 waiters: None
[I 150331 16:40:04 web:1825] 200 GET /lifecycle.html (127.0.0.1) 0.87ms
[I 150331 16:40:04 web:1825] 200 GET /_static/css/theme.css (127.0.0.1) 1.87ms
[I 150331 16:40:04 web:1825] 304 GET /livereload.js (127.0.0.1) 0.50ms
[I 150331 16:40:04 web:1825] 200 GET /_static/doctools.js (127.0.0.1) 0.43ms
[I 150331 16:40:04 web:1825] 200 GET /_static/jquery.js (127.0.0.1) 0.67ms
[I 150331 16:40:04 web:1825] 200 GET /_static/underscore.js (127.0.0.1) 0.48ms
[I 150331 16:40:04 web:1825] 200 GET /_static/js/theme.js (127.0.0.1) 0.40ms
[I 150331 16:40:04 web:1825] 200 GET /_images/virtual_machine_lifecycle_v2.jpeg (127.
↪0.0.1) 4.61ms
[I 150331 16:40:04 web:1825] 200 GET /_images/whiteboard-lifecycle.png (127.0.0.1) 2.
↪02ms
[I 150331 16:40:04 web:1825] 200 GET /_images/packer_diagram.png (127.0.0.1) 1.65ms
[I 150331 16:40:04 web:1825] 200 GET /_images/screenshot-lifecycle.png (127.0.0.1) 1.
↪37ms
[I 150331 16:40:04 web:1825] 200 GET /_images/vm_org_chart.png (127.0.0.1) 0.70ms
[I 150331 16:40:04 web:1825] 200 GET /_images/graphviz-
↪f8dca63773d709e39ae45240fc6b7ed94229eb74.png (127.0.0.1) 0.92ms
[I 150331 16:40:04 web:1825] 200 GET /_static/fonts/fontawesome-webfont.woff?v=4.0.3␣
↪(127.0.0.1) 0.55ms
[I 150331 16:40:05 handlers:109] Browser Connected: http://127.0.0.1:41013/lifecycle.
↪html
```

Now we double-check to make sure we have the change we expect, add, and commit the fix:

```
$ git stat
?? Makefile.dot
$ git add ../Makefile.dot
$ git commit -m "Add Makefile.dot from packer repo for lifecycle.rst"
[develop d5a948e] Add Makefile.dot from packer repo for lifecycle.rst
 1 file changed, 83 insertions(+)
 create mode 100644 Makefile.dot
```

Make note of the commit that includes just the new file: commit `d5a948e` in this case. Now you could bump the version if necessary before pushing.

```
$ (cd ..; bumpversion patch)
$ git hf push
Fetching origin
Already up-to-date.
Counting objects: 10, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (10/10), done.
Writing objects: 100% (10/10), 783 bytes | 0 bytes/s, done.
Total 10 (delta 8), reused 0 (delta 0)
remote: Running post-receive hook: Tue Mar 31 17:02:43 PDT 2015
remote: Scheduled polling of dims-ci-utils-deploy-develop
remote: Scheduled polling of dims-ci-utils-deploy-master
remote: Scheduled polling of dims-seed-jobs
remote: No git consumers for URI git@git.devops.develop:/opt/git/dims-ci-utils.git
remote: [+++] post-receive-06jenkinsalldocs started
remote: [+++] REPONAME=dims-ci-utils
remote: [+++] BRANCH=develop
remote: [+++] newrev=a95c9e1356ff7c6aaed5bcdbe7b533ffc74b6cc1
remote: [+++] oldrev=d5a948ebef61da98b7849416ee340e0a4ba45a3a
```

```
remote: [+++] Branch was updated.
remote: [+++] This repo has a documentation directory.
remote: [+++] post-receive-06jenkinsalldocs finished
To git@git.devops.develop:/opt/git/dims-ci-utils.git
   d5a948e..a95c9e1  develop -> develop

Summary of actions:
- The remote branch 'origin/develop' was updated with your changes
```

Now you can go back to the feature branch you were working on, and `cherry-pick` the commit with the missing file.

```
$ git checkout feature/coreos
Switched to branch 'feature/coreos'
Your branch is ahead of 'origin/feature/coreos' by 1 commit.
  (use "git push" to publish your local commits)
$ git cherry-pick d5a948e
[feature/coreos 14dbf59] Add Makefile.dot from packer repo for lifecycle.rst
 Date: Tue Mar 31 16:38:03 2015 -0700
 1 file changed, 83 insertions(+)
 create mode 100644 Makefile.dot
$ git log
commit 14dbf59dff5d6fce51c899b32fef87276dbddef7
Author: Dave Dittrich <dave.dittrich@gmail.com>
Date:   Tue Mar 31 16:38:03 2015 -0700

    Add Makefile.dot from packer repo for lifecycle.rst
...
```

Note: Note that this results in a new commit hash on this branch (in this case, `14dbf59dff5d6fce51c899b32fef87276dbddef7`).

5.5.5 Synchronizing with an *upstream* repository

Note: The DIMS project is using forks of several source repositories, some for the sake of local customization, and some for adding features necessary for DIMS purposes. The MozDef project is one of these (see the dimsad:dimsarchitecturedesign document, Section dimsad:conceptofexecution).

To track another project's Git repository, syncing it with a fork that you use locally, it is necessary to do the following:

- Configuring a remote for a fork
- Syncing a fork

1. Make sure that you have defined *upstream* properly, e.g.,

   ```
   [dimsenv] ~/dims/git/MozDef (master) $ git remote -v
   origin      git@git.devops.develop:/opt/git/MozDef.git (fetch)
   origin      git@git.devops.develop:/opt/git/MozDef.git (push)
   upstream    git@github.com:jeffbryner/MozDef.git (fetch)
   upstream    git@github.com:jeffbryner/MozDef.git (push)
   ```

2. Fetch the contents of the `upstream` remote repository:

```
[dimsenv] ~/dims/git/MozDef (master) $ git fetch upstream
remote: Counting objects: 6, done.
remote: Total 6 (delta 2), reused 2 (delta 2), pack-reused 4
Unpacking objects: 100% (6/6), done.
From github.com:jeffbryner/MozDef
   700c1be..4575c0f  master     -> upstream/master
 * [new tag]         v1.12      -> v1.12
```

3. Checkout the branch to sync (e.g., `master`) and then `merge` any changes:

```
[dimsenv] ~/dims/git/MozDef (master) $ git checkout master
Already on 'master'
Your branch is up-to-date with 'origin/master'.
[dimsenv] ~/dims/git/MozDef (master) $ git merge upstream/master
Merge made by the 'recursive' strategy.
 alerts/unauth_ssh_pyes.conf |  4 ++++
 alerts/unauth_ssh_pyes.py   | 78␣
↪++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 2 files changed, 82 insertions(+)
 create mode 100644 alerts/unauth_ssh_pyes.conf
 create mode 100644 alerts/unauth_ssh_pyes.py
[dimsenv] ~/dims/git/MozDef (master) $ git push origin master
Counting objects: 8, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (8/8), done.
Writing objects: 100% (8/8), 2.11 KiB | 0 bytes/s, done.
Total 8 (delta 3), reused 0 (delta 0)
remote: Running post-receive hook: Thu Sep 17 20:52:14 PDT 2015
To git@git.devops.develop:/opt/git/MozDef.git
   180484a..766da56  master -> master
```

4. Now push the updated repository to the "local" *remote repository* (i.e, `git.devops.develop` for the DIMS project):

```
[dimsenv] ~/dims/git/MozDef (master) $ git push origin master
Counting objects: 8, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (8/8), done.
Writing objects: 100% (8/8), 2.11 KiB | 0 bytes/s, done.
Total 8 (delta 3), reused 0 (delta 0)
remote: Running post-receive hook: Thu Sep 17 20:52:14 PDT 2015
To git@git.devops.develop:/opt/git/MozDef.git
   180484a..766da56  master -> master
```

5. If the *remote* repository is itself the fork (e.g., if you fork a repository on GitHub, then want to maintain a "local" *remote repository* on-site for your project, you may wish to use a label other than `upstream` to connote the fork differently):

```
[dimsenv] ~/git/ansible (release1.8.4*) $ git remote -v
davedittrich        git@github.com:davedittrich/ansible.git (fetch)
davedittrich        git@github.com:davedittrich/ansible.git (push)
origin      https://github.com/ansible/ansible.git (fetch)
origin      https://github.com/ansible/ansible.git (push)
```

5.5.6 Starting a "release"

By convention, DIMS repositories have at least one file, named `VERSION`, that contains the release version number. You can see the current release by looking at the contents of this file.

```
$ cat VERSION
1.1.4
```

Note: There may be other files, such as the Sphinx documentation configuration file, `docs/source/conf.py` usually, or other source files for Python or Java builds. Each of the files that has a version/release number in it **must** use the same string and be included in the `.bumpversion.cfg` file in order for `bumpversion` to properly manage release numbers.

Now that you know what the current version number is, you can initiate a release branch with `hub-flow`, knowing that the new numbr will be. In this case, we will create a release branch `1.2.0` to increment the minor version number component.

```
$ git hf release start 1.2.0
Fetching origin
Switched to a new branch 'release/1.2.0'
Total 0 (delta 0), reused 0 (delta 0)
remote: Running post-receive hook: Thu Jan 22 18:33:54 PST 2015
To git@git.devops.develop:/opt/git/ansible-playbooks.git
 * [new branch]      release/1.2.0 -> release/1.2.0

Summary of actions:
- A new branch 'release/1.2.0' was created, based on 'dev'
- The branch 'release/1.2.0' has been pushed up to 'origin/release/1.2.0'
- You are now on branch 'release/1.2.0'

Follow-up actions:
- Bump the version number now!
- Start committing last-minute fixes in preparing your release
- When done, run:

     git hf release finish '1.2.0'
```

You should now be on the new release branch:

```
$
```

After making any textual changes, bump the version number to match the new release number:

```
$ bumpversion minor
```

Now the release can be finished. You will be placed in an editor to create comments for actions like merges and tags.

```
$ bumpversion minor
$ cat VERSION
1.2.0
$ git hf release finish '1.2.0'
Fetching origin
Fetching origin
Counting objects: 9, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (8/8), done.
```

```
Writing objects: 100% (9/9), 690 bytes | 0 bytes/s, done.
Total 9 (delta 7), reused 0 (delta 0)
remote: Running post-receive hook: Thu Jan 22 18:37:24 PST 2015
To git@git.devops.develop:/opt/git/ansible-playbooks.git
   3ac28a2..5ca145b  release/1.2.0 -> release/1.2.0
Switched to branch 'master'
Your branch is up-to-date with 'origin/master'.
Removing roles/tomcat/tasks/main.yml
Removing roles/tomcat/handlers/main.yml
Removing roles/tomcat/defaults/main.yml
Removing roles/postgres/templates/pg_hba.conf.j2
Removing roles/postgres/files/schema.psql
Removing roles/ozone/files/postgresql-9.3-1102.jdbc41.jar
Auto-merging roles/logstash/files/demo.logstash.deleteESDB
Auto-merging roles/logstash/files/demo.logstash.addwebsense
Auto-merging roles/logstash/files/demo.logstash.addufw
Auto-merging roles/logstash/files/demo.logstash.addrpcflow
Auto-merging roles/logstash/files/demo.logstash.addcymru

[ ... ]

~
".git/MERGE_MSG" 7L, 280C written
Merge made by the 'recursive' strategy.
 .bumpversion.cfg                                                   |   11 +
 Makefile                                                           |   61 +-
 VERSION                                                            |    1 +
 configure-all.yml                                                  |    5 +-
 dims-all-desktop.yml                                               |   56 +
 dims-all-server.yml                                                |  125 ++
 dims-cifv1-server.yml                                              |   50 +

[...]


Release 1.2.0.
#
# Write a message for tag:
#   1.2.0
# Lines starting with '#' will be ignored.

[...]

~
".git/TAG_EDITMSG" 5L, 97C written
Switched to branch 'dev'
Your branch is up-to-date with 'origin/dev'.

Merge tag '1.2.0' into dev for
Merge tag '1.2.0' into dev for
Merge tag '1.2.0' into dev for Release 1.2.0.

# Please enter a commit message to explain why this merge is necessary,
# especially if it merges an updated upstream into a topic branch.
#
# Lines starting with '#' will be ignored, and an empty message aborts
# the commit.
```

```
[...]

".git/MERGE_MSG" 7L, 273C written
Merge made by the 'recursive' strategy.
 .bumpversion.cfg    | 2 +-
 VERSION             | 2 +-
 docs/source/conf.py | 4 ++--
 group_vars/all      | 2 +-
 4 files changed, 5 insertions(+), 5 deletions(-)
Deleted branch release/1.2.0 (was 5ca145b).
Counting objects: 2, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 447 bytes | 0 bytes/s, done.
Total 2 (delta 0), reused 0 (delta 0)
remote: Running post-receive hook: Thu Jan 22 18:38:17 PST 2015
To git@git.devops.develop:/opt/git/ansible-playbooks.git
   3ac28a2..aec921c  dev -> dev
Total 0 (delta 0), reused 0 (delta 0)
remote: Running post-receive hook: Thu Jan 22 18:38:19 PST 2015
To git@git.devops.develop:/opt/git/ansible-playbooks.git
   2afb58f..2482d07  master -> master
Counting objects: 1, done.
Writing objects: 100% (1/1), 166 bytes | 0 bytes/s, done.
Total 1 (delta 0), reused 0 (delta 0)
remote: Running post-receive hook: Thu Jan 22 18:38:25 PST 2015
To git@git.devops.develop:/opt/git/ansible-playbooks.git
 * [new tag]         1.2.0 -> 1.2.0
remote: Running post-receive hook: Thu Jan 22 18:38:28 PST 2015
To git@git.devops.develop:/opt/git/ansible-playbooks.git
 - [deleted]         release/1.2.0

Summary of actions:
- Latest objects have been fetched from 'origin'
- Release branch has been merged into 'master'
- The release was tagged '1.2.0'
- Tag '1.2.0' has been back-merged into 'dev'
- Branch 'master' has been back-merged into 'dev'
- Release branch 'release/1.2.0' has been deleted
- 'dev', 'master' and tags have been pushed to 'origin'
- Release branch 'release/1.2.0' in 'origin' has been deleted.
```

Lastly, bump the patch version number in the `dev` branch to make sure that when something reports the version in developmental code builds, it doesn't look like you are using code from the *last tagged* `master` branch. That completely defeats the purpose of using version numbers for dependency checks or debugging.

```
$ bumpversion patch
$ git push
Counting objects: 9, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (8/8), done.
Writing objects: 100% (9/9), 683 bytes | 0 bytes/s, done.
Total 9 (delta 7), reused 0 (delta 0)
remote: Running post-receive hook: Thu Jan 22 18:51:00 PST 2015
To git@git.devops.develop:/opt/git/ansible-playbooks.git
   aec921c..d4fe053  dev -> dev
```

Figure *New 1.2.0 release on master, dev now on 1.2.1.* shows what the branches look like with GitX.app on a Mac:

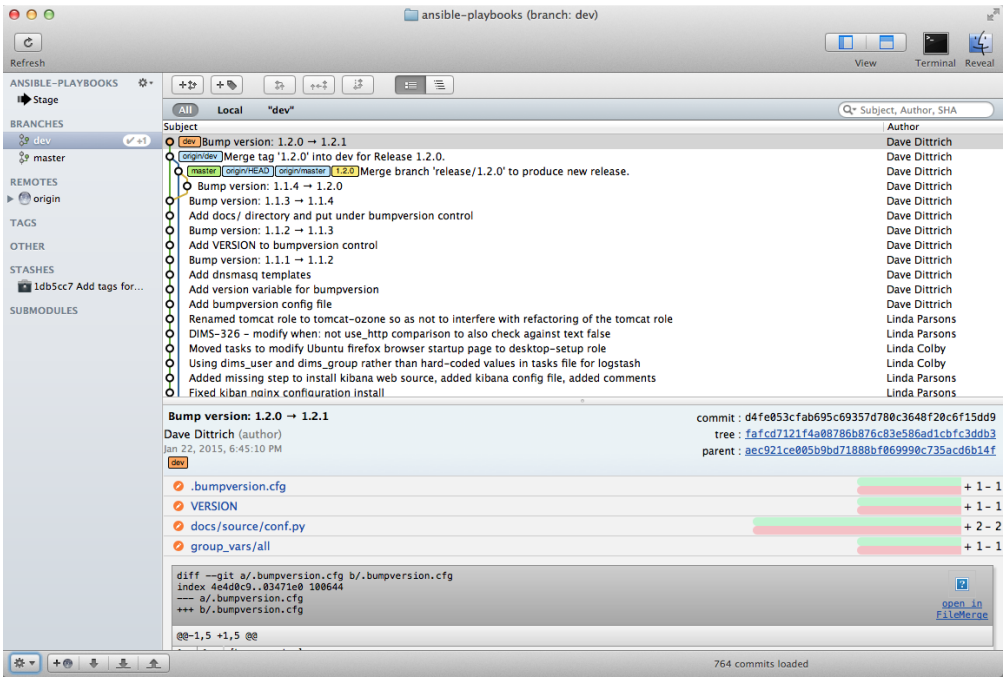


Fig. 5.2: New 1.2.0 release on master, dev now on 1.2.1.

5.5.7 Branch Renaming

Several of the git repos comprising the DIMS source code management system are using the name `dev` for the main development branch. The (somewhat) accepted name for the development branch is `develop`, as detailed in e.g. *http://nvie.com/posts/a-successful-git-branching-model/*.

We would therefore like to rename any dev branch to develop throughout our git repo set. This will of course impact team members who use the central repos to share work. Research online suggests that branch renaming can be done. The best source found was https://gist.github.com/lttlrck/9628955, who suggested a three-part operation

```
# Rename branch locally
git branch -m old_branch new_branch
# Delete the old branch
git push origin :old_branch
# Push the new branch, set local branch to track the new remote
git push --set-upstream origin new_branch
```

To test this recipe out without impacting any existing repos and therefore avoiding any possible loss of real work, we constructed a test situation with a central repo and two fake 'users' who both push and pull from that repo. A branch rename is then done, following the recipe above. The impact on each of the two users is noted.

First, we create a bare repo. This will mimic our authoratitive repos on `git.devops.develop`. We'll call this repo `dims-328.git`, named after the DIMS Jira ticket created to study the branch rename issue:

```
$ cd
$ mkdir depot
$ cd depot
$ git init --bare dims-328.git
```

Next, we clone this repo a first time, which simulates the first 'user' (replace /home/stuart/ with your local path):

```
$ cd
$ mkdir scratch
$ cd scratch
$ git clone file:///home/stuart/depot/dims-328.git
```

Next, we dd some content in master branch

```
$ cd dims-328
$ echo content > foo
$ git add foo
$ git commit -m "msg"
$ git push origin master
```

We now clone the 'depot' repo a second time, to simulate the second user. Both users are then developing using the authoratitive repo as the avenue to share work. Notice how the second user clones into the specified directory `dims-328-2`, so as not to tread on the first user's work:

```
$ cd ~/scratch
$ git clone file:///home/stuart/depot/dims-328.git dims-328-2
```

user1 (first clone) then creates a `dev` branch and adds some content to it:

```
$ cd ~/scratch/dims-328
$ git branch dev
$ git checkout dev
$ echo content > devbranch
$ git add devbranch
$ git commit -m "added content to dev branch"
$ git push origin dev
```

This will create a `dev` branch in the origin repo, i.e the depot.

Next, as the second user, pull the changes, checkout `dev` and edit:

```
$ cd ~scratch/dims-328-2
$ git pull
$ git checkout dev
$ echo foo >> devbranch
```

At this point we have two 'users' with local repos, both of which share a common upstream repo. Both users have got the dev branch checked out, and may have local changes on that branch.

Now, we wish to rename the branch `dev` to `develop` throughout, i.e. at the depot and in users' repos.

Using instructions from https://gist.github.com/lttlrck/9628955, and noting the impacts to each user, we first act as *user1*, who will be deemed 'in charge' of the renaming process:

```
$ cd ~scratch/dims-328
$ git branch -m dev develop
$ git push origin :dev
To file:///home/stuart/depot/dims-328.git
 - [deleted]         dev
$ git push --set-upstream origin develop
Counting objects: 2, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 259 bytes | 0 bytes/s, done.
Total 2 (delta 0), reused 0 (delta 0)
```

```
To file:///home/stuart/depot/dims-328.git
 * [new branch]      develop -> develop
Branch develop set up to track remote branch develop from origin.
```

Warning: (This reads like a ..warning block. Is that how it was meant?)

The `git push` output message implies a deletion of the `dev` branch in the depot. If *user2* were to interact with `origin/dev` now, what would happen??

Here are the contents of *user1*'s `.git/config` after the 3-operation rename:

```
$ cat .git/config
[core]
    repositoryformatversion = 0
    filemode = true
    bare = false
    logallrefupdates = true
[remote "origin"]
    url = file:///home/stuart/depot/dims-328.git
    fetch = +refs/heads/*:refs/remotes/origin/*
[branch "master"]
    remote = origin
    merge = refs/heads/master
[branch "develop"]
    remote = origin
    merge = refs/heads/develop
```

Note how there are references to `develop` but none to `dev`. As far as *user1* is concerned, the branch rename appears to have worked and is complete.

Now, what does *user2* see? With `dev` branch checked out, *and* with a local mod, we do a `pull`:

```
$ cd ~scratch/dims-328-2
$ git pull
From file:///home/stuart/depot/dims-328
 * [new branch]      develop    -> origin/develop
Your configuration specifies to merge with the ref 'dev'
from the remote, but no such ref was fetched.
```

This is some form of error message. *user2*'s `.git/config` at this point is this:

```
$ cat .git/config
[core]
    repositoryformatversion = 0
    filemode = true
    bare = false
    logallrefupdates = true
[remote "origin"]
    url = file:///home/stuart/depot/dims-328.git
    fetch = +refs/heads/*:refs/remotes/origin/*
[branch "master"]
    remote = origin
    merge = refs/heads/master
[branch "dev"]
    remote = origin
    merge = refs/heads/dev
```

Perhaps just the branch rename will work for *user2*? As *user2*, we do the first part of the *rename recipe*:

```
$ git branch -m dev develop
```

No errors from this, but *user2*'s `.git/config` still refers to a `dev` branch:

```
$ cat .git/config
[core]
    repositoryformatversion = 0
    filemode = true
    bare = false
    logallrefupdates = true
[remote "origin"]
    url = file:///home/stuart/depot/dims-328.git
    fetch = +refs/heads/*:refs/remotes/origin/*
[branch "master"]
    remote = origin
    merge = refs/heads/master
[branch "develop"]
    remote = origin
    merge = refs/heads/dev
```

Next, as *user2*, we issued the third part of the *rename recipe* (but skipped the second part):

```
$ git push --set-upstream origin develop
Branch develop set up to track remote branch develop from origin.
Everything up-to-date.
```

Note that this is a `push`, but since `user2` had no committed changes locally, no content was actually pushed.

Now *user2*'s `.git/config` looks better, the token `dev` has changed to `develop`:

```
$ cat .git/config
[core]
        repositoryformatversion = 0
        filemode = true
        bare = false
        logallrefupdates = true
[remote "origin"]
        url = file:///home/stuart/depot/dims-328.git
        fetch = +refs/heads/*:refs/remotes/origin/*
[branch "master"]
        remote = origin
        merge = refs/heads/master
[branch "develop"]
        remote = origin
        merge = refs/heads/develop
```

Next, as *user2*, commit the local change, and push to depot:

```
$ git add devbranch
$ git commit -m "msg"
$ git push
```

So it appears that *user2* can issue just the branch rename and upstream operation, and skip the second component of the 3-part recipe (`git push origin :old_branch`), likely since this is an operation on the remote (depot) itself and was already done by *user1*.

Finally, we switch back to *user1* and pull changes made by *user2*:

```
$ cd ~scratch/dims-328
$ git pull
```

Warning: This has addressed *only* git changes. The wider implications of a git branch rename on systems such as Jenkins has yet to be addressed. Since systems like Jenkins generally just clone or pull from depots, it is expected that only git URLs need to change from including `dev` to `develop`.

5.5.8 Deleting accidentally created tags

When trying to finish a release, you may accidentally create a tag named `finish`. It may even get propagated automatically to `origin`, in which case it could propagate to others' repos:

```
mr update: /Users/dittrich/dims/git/dims-keys
Fetching origin
From git.devops.develop:/opt/git/dims-keys
 * [new tag]         finish     -> finish
```

You can delete them locally and remotely with the following commands:

```
 $ git tag -d finish
 Deleted tag 'finish' (was 516d9d2)
 $ git push origin :refs/tags/finish
 remote: Running post-receive hook: Thu Aug  6 16:07:17 PDT 2015
 To git@git.devops.develop:/opt/git/dims-keys.git
  - [deleted]         finish
```

5.5.9 Recovering deleted files

Files that have been deleted in the past, and the deletions commited, can be recovered by searching the Git history of deletions to identify the commit that included the deletion. The file can then be checked out using the predecessor to that commit. See Find and restore a deleted file in a Git repository

5.5.10 Fixing comments in unpublished commits

Note: This section was derived from http://makandracards.com/makandra/868-change-commit-messages-of-past-git-commits

Warning: Only do this if you have **not already pushed** the changes!! As noted in the `git-commit` man page for the `--amend` option:

```
You should understand the implications of rewriting history if you
amend a commit that has already been published. (See the "RECOVERING
FROM UPSTREAM REBASE" section in git-rebase(1).)
```

There may be times when you accidentally make multiple commits, one at a time, using the same comment (but the changes are not related to the comment).

Here is an example of three commits all made with `git commit -am` using the same message:

```
$ git log
commit 08b888b9dd33f53f0e26d8ff8aab7309765ad0eb
Author: Dave Dittrich <dave.dittrich@gmail.com>
Date:   Thu Apr 30 18:35:08 2015 -0700

    Fix intersphinx links to use DOCSURL env variable

commit 7f3d0d8134c000a787aad83f2690808008ed1d96
Author: Dave Dittrich <dave.dittrich@gmail.com>
Date:   Thu Apr 30 18:34:40 2015 -0700

    Fix intersphinx links to use DOCSURL env variable

commit f6f5d868c8ddd12018ca662a54d1f58c150e6364
Author: Dave Dittrich <dave.dittrich@gmail.com>
Date:   Thu Apr 30 18:33:59 2015 -0700

    Fix intersphinx links to use DOCSURL env variable

commit 96575c967f606e2161033de92dd2dc580ad60a8b
Merge: 1253ea2 dae5aca
Author: Linda Parsons <lparsonstech@gmail.com>
Date:   Thu Apr 30 14:00:49 2015 -0400

    Merge remote-tracking branch 'origin/develop' into develop

commit 1253ea20bc553759c43d3a999b81be009851d195
Author: Linda Parsons <lparsonstech@gmail.com>
Date:   Thu Apr 30 14:00:19 2015 -0400

    Added information for deploying to infrastructure
```

Note: Make note that the commit immediately prior to the three erroneously commented commits is `96575c96`. We will use that commit number in a moment...

Looking at the patch information shows these are clearly not all correctly commented:

```
$ git log --patch
commit 08b888b9dd33f53f0e26d8ff8aab7309765ad0eb
Author: Dave Dittrich <dave.dittrich@gmail.com>
Date:   Thu Apr 30 18:35:08 2015 -0700

    Fix intersphinx links to use DOCSURL env variable

diff --git a/docs/makedocset b/docs/makedocset
index dafbedb..9adb954 100644
--- a/docs/makedocset
+++ b/docs/makedocset
@@ -7,7 +7,14 @@
 # This is useful for building a set of documents that employ
 # intersphinx linking, obtaining the links from the co-local
 # repositories instead of specified remote locations.
+#
+# To build the docs for a specific server (e.g., when building
+# using a local docker container running Nginx), set the
```

```
+# environment variable DOCSURL to point to the server:
+#
+# $ export DOCSURL=http://192.168.99.100:49153

+DOCSURL=${DOCSURL:-http://app.devops.develop:8080/docs/devel}

 # Activate dimsenv virtual environment for Sphinx
 . $HOME/dims/envs/dimsenv/bin/activate

commit 7f3d0d8134c000a787aad83f2690808008ed1d96
Author: Dave Dittrich <dave.dittrich@gmail.com>
Date:   Thu Apr 30 18:34:40 2015 -0700

    Fix intersphinx links to use DOCSURL env variable

diff --git a/docs/source/conf.py b/docs/source/conf.py
index 9fdc100..b3cd483 100644
--- a/docs/source/conf.py
+++ b/docs/source/conf.py
@@ -351,13 +351,16 @@ epub_exclude_files = ['search.html']
 # If false, no index is generated.
 #epub_use_index = True

+os.environ['GITBRANCH'] = "develop"
+
+if os.environ.get('DOCSURL') is None:
+    #os.environ['DOCSURL'] = "file://{}".format(os.environ.get('GIT'))
+    os.environ['DOCSURL'] = "http://app.devops.develop:8080/docs/{}/html/".format(
+        os.environ['GITBRANCH'])

 intersphinx_cache_limit = -1   # days to keep the cached inventories (0 == forever)
 intersphinx_mapping = {
-        'dimsocd': ("%s/dims/docs/dims-ocd" % os.environ['HOME'],
-                    ('http://app.devops.develop:8080/docs/develop/html/dims-ocd/
↪objects.inv', None)),
-        'dimsad': ("%s/dims/docs/dims-ad" % os.environ['HOME'],
-                    ('http://app.devops.develop:8080/docs/develop/html/dims-ad/
↪objects.inv', None)),
-        'dimssr': ("%s/dims/docs/dims-sr" % os.environ['HOME'],
-                    ('http://app.devops.develop:8080/docs/develop/html/dims-sr/
↪objects.inv', None))
+        'dimsocd': ("{}/dims-ocd".format(os.environ['DOCSURL']), None),
+        'dimsad': ("{}/dims-ad".format(os.environ['DOCSURL']), None),
+        'dimssr': ("{}/dims-sr".format(os.environ['DOCSURL']), None)
 }

commit f6f5d868c8ddd12018ca662a54d1f58c150e6364
Author: Dave Dittrich <dave.dittrich@gmail.com>
Date:   Thu Apr 30 18:33:59 2015 -0700

    Fix intersphinx links to use DOCSURL env variable

diff --git a/docs/makedocs b/docs/makedocs
deleted file mode 100644
index dafbedb..0000000
--- a/docs/makedocs
+++ /dev/null
@@ -1,66 +0,0 @@
```

```
-#!/bin/bash -x
-#
-# This script builds multiple Sphinx documents in repos
-# residing (in their current checkout branch/state) in
-# the directory specified by the $GIT environment variable.
-#
-# This is useful for building a set of documents that employ
-# intersphinx linking, obtaining the links from the co-local
-# repositories instead of specified remote locations.
...
```

The last commit is easy to fix. Just use `git commit --amend` and edit the message:

```
$ git commit --amend

Add DOCSURL selection of where docs reside for intersphinx links

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# Date:      Thu Apr 30 18:35:08 2015 -0700
#
# On branch develop
# Your branch is ahead of 'origin/develop' by 3 commits.
#   (use "git push" to publish your local commits)
#
# Changes to be committed:
#       modified:   makedocset
```

Now we can see the message has been changed, but so has the commit hash!

```
$ git log --patch
commit 654cb34378cb0a4140725a37e3724b6dcee7aebd
Author: Dave Dittrich <dave.dittrich@gmail.com>
Date:   Thu Apr 30 18:35:08 2015 -0700

    Add DOCSURL selection of where docs reside for intersphinx links

diff --git a/docs/makedocset b/docs/makedocset
index dafbedb..9adb954 100644
--- a/docs/makedocset
+++ b/docs/makedocset
@@ -7,7 +7,14 @@
 # This is useful for building a set of documents that employ
 # intersphinx linking, obtaining the links from the co-local
 # repositories instead of specified remote locations.
+#
+# To build the docs for a specific server (e.g., when building
+# using a local docker container running Nginx), set the
+# environment variable DOCSURL to point to the server:
+#
+# $ export DOCSURL=http://192.168.99.100:49153

+DOCSURL=${DOCSURL:-http://app.devops.develop:8080/docs/devel}

 # Activate dimsenv virtual environment for Sphinx
 . $HOME/dims/envs/dimsenv/bin/activate
```

```
commit 7f3d0d8134c000a787aad83f2690808008ed1d96
Author: Dave Dittrich <dave.dittrich@gmail.com>
Date:   Thu Apr 30 18:34:40 2015 -0700

    Fix intersphinx links to use DOCSURL env variable

diff --git a/docs/source/conf.py b/docs/source/conf.py
...
```

The second commit has the correct comment, but commit `f6f5d868c` was simply renaming a file. It got caught up as a commit when the `-a` option was given when committing the changed file, not realizing the renamed file had already been added to the cache.

To change the message for *only* commit `f6f5d86`, start an interactive rebase at the commit immediately prior to that commit (in this case, commit `96575c9`). Change `pick` to `edit` for that commit.

```
$ git rebase -i 96575c9

edit f6f5d86 Fix intersphinx links to use DOCSURL env variable
pick 7f3d0d8 Fix intersphinx links to use DOCSURL env variable
pick 654cb34 Add DOCSURL selection of where docs reside for intersphinx links

# Rebase 96575c9..654cb34 onto 96575c9 (       3 TODO item(s))
#
# Commands:
# p, pick = use commit
# r, reword = use commit, but edit the commit message
# e, edit = use commit, but stop for amending
# s, squash = use commit, but meld into previous commit
# f, fixup = like "squash", but discard this commit's log message
# x, exec = run command (the rest of the line) using shell
#
# These lines can be re-ordered; they are executed from top to bottom.
#
# If you remove a line here THAT COMMIT WILL BE LOST.
#
# However, if you remove everything, the rebase will be aborted.
#
# Note that empty commits are commented out
```

As soon as you exit the editor, Git will begin the rebase and tell you what to do next:

```
Stopped at f6f5d868c8ddd12018ca662a54d1f58c150e6364... Fix intersphinx links to use␣
↪DOCSURL env variable
You can amend the commit now, with

        git commit --amend

Once you are satisfied with your changes, run

        git rebase --continue
```

Now use `git commit --amend` to edit the comment:

```
$ git commit --amend

Rename makedocs -> makedocset

```

```
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# Date:      Thu Apr 30 18:33:59 2015 -0700
#
# rebase in progress; onto 96575c9
# You are currently editing a commit while rebasing branch 'develop' on '96575c9'.
#
# Changes to be committed:
#       renamed:    makedocs -> makedocset
#
```

Finish off by continuing the rebase for the remaining commits.

```
$ git rebase --continue
Successfully rebased and updated refs/heads/develop.
```

Now `git log` shows the correct comments, as well as new commit hashes:

```
$ git log
commit 89af6d9fda07276d3cb06dfd2977f1392fb03b25
Author: Dave Dittrich <dave.dittrich@gmail.com>
Date:   Thu Apr 30 18:35:08 2015 -0700

    Add DOCSURL selection of where docs reside for intersphinx links

commit c2c55ff3dcbf10739c5d86ce8a6192e930ccd265
Author: Dave Dittrich <dave.dittrich@gmail.com>
Date:   Thu Apr 30 18:34:40 2015 -0700

    Fix intersphinx links to use DOCSURL env variable

commit 2155936ad7e3ae71ef5775b2036a4b6c21a9a86d
Author: Dave Dittrich <dave.dittrich@gmail.com>
Date:   Thu Apr 30 18:33:59 2015 -0700

    Rename makedocs -> makedocset

commit 96575c967f606e2161033de92dd2dc580ad60a8b
Merge: 1253ea2 dae5aca
Author: Linda Parsons <lparsonstech@gmail.com>
Date:   Thu Apr 30 14:00:49 2015 -0400

    Merge remote-tracking branch 'origin/develop' into develop
```

5.5.11 Squashing Commits Before Merging

Working on a feature branch for a long time can mean many changes are made. The idea of "commit often" to push code so other team members can review it means that sometimes multiple edits are made (or reverted commits) while debugging something. Or you may make a number of changes that are unrelated to the topic of the feature branch that would be best kept together in a single commit.

It is possible to combine multiple commits into a single commit using an interactive Git rebase (`git rebase -i`). The idea is to interactively select a starting point for the rebase operation, then using `pick` and `squash` to select the proper commits to keep, and those subsequent commits that should be merged into the previous commit. If there are

dozens of commits, this can get very complicated, but the idea can be demonstrated with three simple changes that we wish to turn into just one merge commit.

Note: This example is being done with a temporary repository that we will make for this purpose, allowing experimentation in a way that will not result in harm to an actual repo.

Start by initializing a temporary repo in `/tmp/testing`.

```
[dimsenv] dittrich@ren:/tmp/testing () $ git init .
Initialized empty Git repository in /private/tmp/testing/.git/
```

We now create three files, each with a numeric name and the corresponding number as the contents of the file, and add each file to the staging area.

```
[dimsenv] dittrich@ren:/tmp/testing () $ for i in 1 2 3; do echo $i > $i.txt; git add
↪$i.txt; done
[dimsenv] dittrich@ren:/tmp/testing () $ git stat
A  1.txt
A  2.txt
A  3.txt
```

We now make our initial commit.

```
[dimsenv] dittrich@ren:/tmp/testing () $ git commit -m 'Initial commit'
[master (root-commit) 3ee79c4] Initial commit
 3 files changed, 3 insertions(+)
 create mode 100644 1.txt
 create mode 100644 2.txt
 create mode 100644 3.txt
```

Now we check out a branch were we will make our changes, before merging them back into `master`.

```
[dimsenv] dittrich@ren:/tmp/testing (master) $ git checkout -b foo
Switched to a new branch 'foo'
```

We now make three changes ("edits" to two files, and one file deletion).

```
[dimsenv] dittrich@ren:/tmp/testing (foo) $ echo "22222" > 2.txt
[dimsenv] dittrich@ren:/tmp/testing (foo*) $ git add 2.txt
[dimsenv] dittrich@ren:/tmp/testing (foo*) $ git commit -m "First edit"
[foo 71738c7] First edit
 1 file changed, 1 insertion(+), 1 deletion(-)
```

```
[dimsenv] dittrich@ren:/tmp/testing (foo) $ echo "1111" > 1.txt
[dimsenv] dittrich@ren:/tmp/testing (foo*) $ git commit -am "Second edit"
[foo 0b0e0a9] Second edit
 1 file changed, 1 insertion(+), 1 deletion(-)
```

```
[dimsenv] dittrich@ren:/tmp/testing (foo) $ git rm 3.txt
rm '3.txt'
[dimsenv] dittrich@ren:/tmp/testing (foo*) $ git commit -m "Removed file"
[foo 0a522af] Removed file
 1 file changed, 1 deletion(-)
 delete mode 100644 3.txt
```

If we now look at the commit history, we see the initial commit where we branched off, and the three change commits.

```
[dimsenv] dittrich@ren:/tmp/testing (foo) $ git gr
* commit 0a522af0d8c09d206f37b647014628a89070fe94 (HEAD -> foo)
| Author: Dave Dittrich <dittrich@u.washington.edu>
| Date:   Mon Sep 5 17:59:36 2016 -0700
|
|     Removed file
|
* commit 0b0e0a986e228b177e8775900198c99af80ef5f2
| Author: Dave Dittrich <dittrich@u.washington.edu>
| Date:   Mon Sep 5 17:58:34 2016 -0700
|
|     Second edit
|
* commit 71738c7b1d2f504110190eaca3c71461e7090cc6
| Author: Dave Dittrich <dittrich@u.washington.edu>
| Date:   Mon Sep 5 17:58:19 2016 -0700
|
|     First edit
|
* commit 3ee79c4d4455a5517a93ce7e02db88d3db7934f4 (master)
  Author: Dave Dittrich <dittrich@u.washington.edu>
  Date:   Mon Sep 5 17:57:51 2016 -0700

      Initial commit
```

We now start an interactive rebase, referencing the hash of the initial commit (the one right before all of the changes on the branch).

```
[dimsenv] dittrich@ren:/tmp/testing (foo) $ git rebase -i 3ee79c4

pick   71738c7 First edit
squash 0b0e0a9 Second edit
squash 0a522af Removed file

# Rebase 3ee79c4..0a522af onto 3ee79c4 (       3 TODO item(s))
#
# Commands:
# p, pick = use commit
# r, reword = use commit, but edit the commit message
# e, edit = use commit, but stop for amending
# s, squash = use commit, but meld into previous commit
# f, fixup = like "squash", but discard this commit's log message
# x, exec = run command (the rest of the line) using shell
#
# These lines can be re-ordered; they are executed from top to bottom.
#
# If you remove a line here THAT COMMIT WILL BE LOST.
#
# However, if you remove everything, the rebase will be aborted.
#
# Note that empty commits are commented out
```

When you save the file out of the editor, Git will perform the rebase operation:

```
[detached HEAD 5f90a71] First edit
 Date: Mon Sep 5 17:58:19 2016 -0700
 3 files changed, 2 insertions(+), 3 deletions(-)
 delete mode 100644 3.txt
```

```
Successfully rebased and updated refs/heads/foo.
```

You can now see the commit history has been changed to reflect only one commit (with a combined comment indicating all of the actions from each commit, since we didn't alter any of the comments while we did the squashing).

```
[dimsenv] dittrich@ren:/tmp/testing (foo) $ git gr
* commit 5f90a717f96501ba6526a83c107302e0fbc30f10 (HEAD -> foo)
| Author: Dave Dittrich <dittrich@u.washington.edu>
| Date:   Mon Sep 5 17:58:19 2016 -0700
|
|     First edit
|
|     Second edit
|
|     Removed file
|
* commit 3ee79c4d4455a5517a93ce7e02db88d3db7934f4 (master)
  Author: Dave Dittrich <dittrich@u.washington.edu>
  Date:   Mon Sep 5 17:57:51 2016 -0700

      Initial commit
```

We can now merge the single resulting commit back into the `master` branch.

```
[dimsenv] dittrich@ren:/tmp/testing (foo) $ git checkout master
Switched to branch 'master'
[dimsenv] dittrich@ren:/tmp/testing (master) $ git merge foo
Updating 3ee79c4..5f90a71
Fast-forward
 1.txt | 2 +-
 2.txt | 2 +-
 3.txt | 1 -
 3 files changed, 2 insertions(+), 3 deletions(-)
 delete mode 100644 3.txt
```

Our changes have taken effect:

```
[dimsenv] dittrich@ren:/tmp/testing (master) $ ls
1.txt   2.txt
[dimsenv] dittrich@ren:/tmp/testing (master) $ cat *
1111
22222
```

The `HEAD` has now moved and `master` and `foo` are at the same point. (We can now delete the `foo` branch.)

```
[dimsenv] dittrich@ren:/tmp/testing (master) $ git gr
* commit 5f90a717f96501ba6526a83c107302e0fbc30f10 (HEAD -> master, foo)
| Author: Dave Dittrich <dittrich@u.washington.edu>
| Date:   Mon Sep 5 17:58:19 2016 -0700
|
|     First edit
|
|     Second edit
|
|     Removed file
|
* commit 3ee79c4d4455a5517a93ce7e02db88d3db7934f4
  Author: Dave Dittrich <dittrich@u.washington.edu>
```

```
  Date:   Mon Sep 5 17:57:51 2016 -0700

      Initial commit
```

5.5.12 Merging changes from `develop` into feature branches to keep up to date

When branches live for a long time, and changes occur to what should be the stable `develop` branch, those branches start to drift and can become "broken" because things like hotfixes and new features are not present on the feature branch. To avoid this, first try to *not have feature branches live for a long time*, and second, merge the develop branch into the feature branches when needed.

Start by reading these pages:

- Commit Often, Perfect Later, Publish Once: Git Best Practices
- Git Tips

```
$ git checkout pyes
Switched to branch 'pyes'
Your branch is up-to-date with 'origin/pyes'.
[dittrich@localhost ansible-playbooks (pyes)]$ git rebase dev
First, rewinding head to replay your work on top of it...
Applying: Add pyes pip install to ELK role
Applying: Add marker to prevent re-loading of existing data and don't add Team Cymru␣
↪data
Applying: Fix bug in demo.logstash.deleteESDB script
Applying: Add feedback message if data already loaded
Applying: Remove debug flag from data pre-load play
Applying: Add demo.firefox.setup changes
Applying: Make default for Kibana be the UFW dataset
Applying: Add curl to base role
Using index info to reconstruct a base tree...
M   roles/base-os/tasks/main.yml
Falling back to patching base and 3-way merge...
Auto-merging roles/base-os/tasks/main.yml
Applying: Add elasticutils Python library to ELK role
Applying: Add play to pin Gnome Terminal to panel
Applying: Fix Ansible syntax error re: Gnome Terminal fix
Applying: Fix typo in Gnome Terminal script setup
Applying: Add dconf-tools package for dconf program
Applying: Run Gnome Terminal pin script with sudo
Applying: Fix for dbus-launch failure
Applying: Fix bug in pin-gnome-terminal script re: dconf write operation
Applying: Fix bug in pin-gnome-terminal script
Applying: Fix bug in pin-gnome-terminal re: empty 'value' variable</verbatim>
```

This is what the `pyes` branch looked like before:

This is what the `pyes` branch looked like after the rebase:

Notice the numbers `+23-19` in the after image? We just rebased commits from the local repo branch `develop` onto the local repo branch `pyes`. We haven't done anything yet with the remote repo. The numbers mean that after rebasing to get the missing commits, there are commits that exist on the local repo that do not exist on the remote repo, and vice-versa.

If we now try to push the `pyes` branch, Git complains that it can't because there are remote changes that are not in the local repo that need to be merged and checked for possible conflict before the push can proceed.

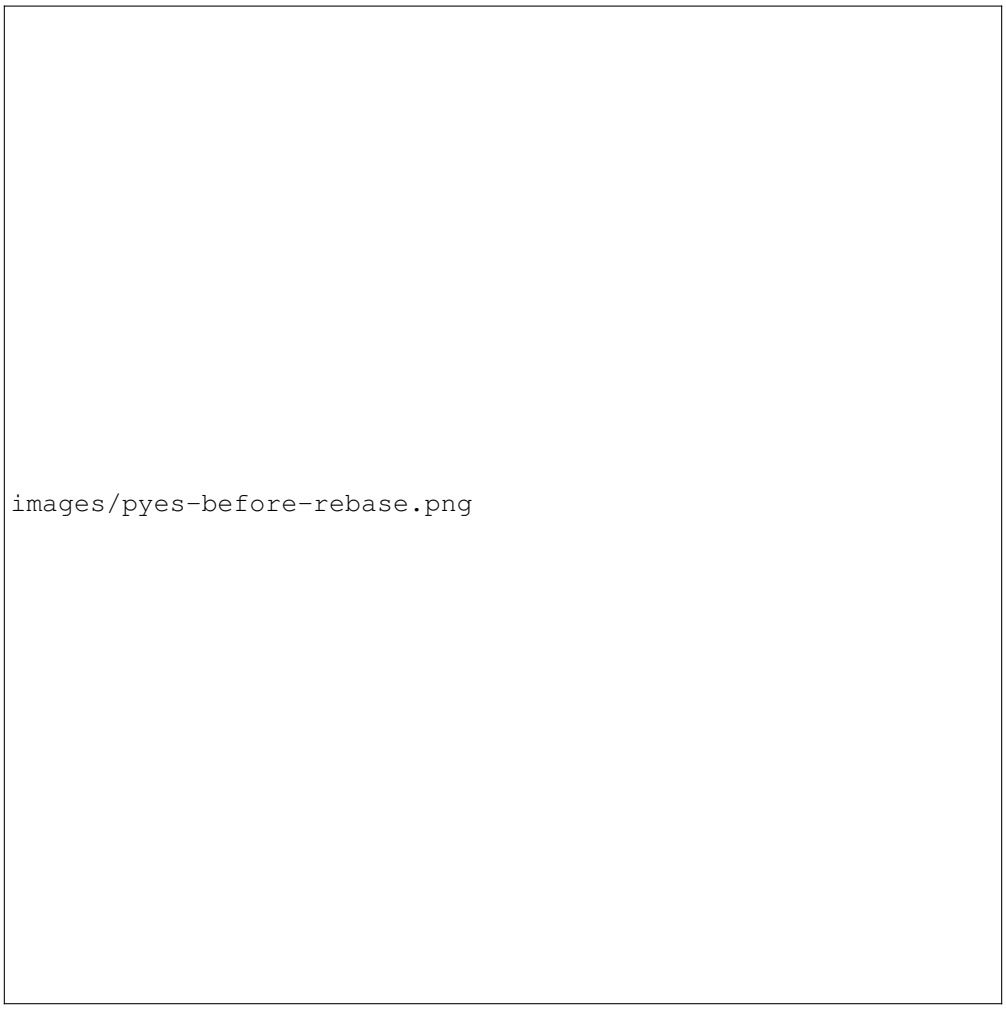


Fig. 5.3: Branch `pyes` before rebase

Fig. 5.4: Branch `pyes` after rebase

```
$ git push
To git@git.devops.develop:/opt/git/ansible-playbooks.git
 ! [rejected]        pyes -> pyes (non-fast-forward)
error: failed to push some refs to 'git@git.devops.develop:/opt/git/ansible-playbooks.
↪git'
hint: Updates were rejected because the tip of your current branch is behind
hint: its remote counterpart. Integrate the remote changes (e.g.
hint: 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.</verbatim>
```

Doing a `git pull` first, then a `git push` results in a clean rebase of the remote commits with the local commits (which are now up to date on the `pyes` feature branch in relation to the `develop` branch.)

```
$ git pull
First, rewinding head to replay your work on top of it...
Applying: Added schema.psl to populate dims database from ops-trust
Applying: added postgres-dims role and files
Using index info to reconstruct a base tree...
M   dims-global-server.yml
<stdin>:94: trailing whitespace.
-- Name: plpgsql; Type: EXTENSION; Schema: -; Owner:
<stdin>:101: trailing whitespace.
-- Name: EXTENSION plpgsql; Type: COMMENT; Schema: -; Owner:
<stdin>:210: trailing whitespace.
-- Name: attestations; Type: TABLE; Schema: public; Owner: postgres; Tablespace:
<stdin>:224: trailing whitespace.
-- Name: audit_history; Type: TABLE; Schema: public; Owner: postgres; Tablespace:
<stdin>:237: trailing whitespace.
-- Name: language_skill; Type: TABLE; Schema: public; Owner: postgres; Tablespace:
warning: squelched 50 whitespace errors
warning: 55 lines add whitespace errors.
Falling back to patching base and 3-way merge...
Auto-merging dims-global-server.yml
Applying: Add curl to all hosts in base-os role
Using index info to reconstruct a base tree...
M   roles/base-os/tasks/main.yml
Falling back to patching base and 3-way merge...
Auto-merging roles/base-os/tasks/main.yml
Applying: Add curl to base role
Using index info to reconstruct a base tree...
M   roles/base-os/tasks/main.yml
Falling back to patching base and 3-way merge...
No changes -- Patch already applied.
[dittrich@localhost ansible-playbooks (pyes)]$ git push
Counting objects: 15, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (14/14), done.
Writing objects: 100% (15/15), 392.72 KiB | 0 bytes/s, done.
Total 15 (delta 9), reused 3 (delta 0)
remote: Running post-receive hook: Tue Nov  4 18:12:01 PST 2014
To git@git.devops.develop:/opt/git/ansible-playbooks.git
   9b23575..2166e16  pyes -> pyes</verbatim>
```

5.5.13 Permanently Removing Files from a Git Repo

Before publishing once private repositories on an internal Git repo server to a public server like GitHub requires making sure that **all** sensitive data is premanantly removed from the local repository's history **before** first pushing it to GitHub. Just going a `git rm file` on a file will remove it from the working directory, but not from Git's history. It still exists in a previous commit in the repo.

A common reason for purging files from the Git history is when someone commits many large binary archive files (e.g., some source packages, operating system installation ISOs, etc). Those files exist in their original distribution servers and mirrors, so there isn't a need to put them under revision control. They just make the Git repo larger for no good reason.

Danger: Realize that if you are trying to permanently remove secrets, such as passwords or encryption private keys, even doing these steps is not enough. Read what GitHub has to say in the Danger block at the top of their Remove sensitive data page. In short, any time extremely sensitive data (like a password or private key) is published to GitHub, it should **be considered compromised**, reported to the project lead, and changed as soon as possible.

- How to delete files permanently from your local and remote git repositories, by Anoopjohn, February 20, 2014
- GitHub aaronzirbes/shrink-git-repo.sh ("This script will help you remove large files from your git repo history and shrink the size of your repository.")
- How to Shrink a Git Repository, by Steve Lorek, May 11, 2012

The page How to Shrink a Git Repository was used successfully to perform cleanup of a large number of archives that were committed to the `ansible-playbooks` repo. The string `filename` needed to be substituted with the paths of the files to delete, which were identified by the script `git-find-largest` and edited with `vi` and `awk` to strip out just the paths. The following command was then used on the list:

```
for f in $(cat largest.txt); do \
  git filter-branch --tag-name-filter cat \
    --index-filter "git rm -r --cached --ignore-unmatch $f" \
    --prune-empty -f -- --all; \
done
```

After that, the steps to clear the cache, do garbage collection and pruning, etc. were followed.

See also GitHub's Remove sensitive data page to use either `git filter-branch` or the BFG Repo-Cleaner to remove files from a clone of the repo and then push the clean version to GitHub.

```
bfg 1.12.15
Usage: bfg [options] [<repo>]

  -b, --strip-blobs-bigger-than <size>
                           strip blobs bigger than X (eg '128K', '1M', etc)
  -B, --strip-biggest-blobs NUM
                           strip the top NUM biggest blobs
  -bi, --strip-blobs-with-ids <blob-ids-file>
                           strip blobs with the specified Git object ids
  -D, --delete-files <glob>
                           delete files with the specified names (eg '*.class', '*.
↪{txt,log}' - matches on file name, not path within repo)
  --delete-folders <glob>  delete folders with the specified names (eg '.svn', '*-tmp
↪' - matches on folder name, not path within repo)
  --convert-to-git-lfs <value>
                           extract files with the specified names (eg '*.zip' or '*.
↪mp4') into Git LFS
  -rt, --replace-text <expressions-file>
```

```
                          filter content of files, replacing matched text. Match␣
↪expressions should be listed in the file, one expression per line - by default,␣
↪each expression is treated as a literal, but 'regex:' & 'glob:' prefixes are␣
↪supported, with '==>' to specify a replacement string other than the default of
↪'***REMOVED***'.
  -fi, --filter-content-including <glob>
                          do file-content filtering on files that match the␣
↪specified expression (eg '*.{txt,properties}')
  -fe, --filter-content-excluding <glob>
                          don't do file-content filtering on files that match the␣
↪specified expression (eg '*.{xml,pdf}')
  -fs, --filter-content-size-threshold <size>
                          only do file-content filtering on files smaller than <size>
↪ (default is 1048576 bytes)
  -p, --protect-blobs-from <refs>
                          protect blobs that appear in the most recent versions of␣
↪the specified refs (default is 'HEAD')
  --no-blob-protection    allow the BFG to modify even your *latest* commit. Not␣
↪recommended: you should have already ensured your latest commit is clean.
  --private               treat this repo-rewrite as removing private data (for␣
↪example: omit old commit ids from commit messages)
  --massive-non-file-objects-sized-up-to <size>
                          increase memory usage to handle over-size Commits, Tags,␣
↪and Trees that are up to X in size (eg '10M')
  <repo>                  file path for Git repository to clean
```

5.6 Git and Secrets

There are a plethora of ways to deal with secrets in relation to source code in public Git repos.

Some groups chose to separate out the secrets into repos that are not made public (i.e., one public repo without secrets, and one private repo with only the secrets). This adds some complexity and requires multiple Git repository locations that clearly separate the private and public repos.

Other groups may prefer to keep their repositories small in number and simple, using a single directory tree with the secrets being encrypted within that tree. At one extreme of the "secrets in the repo" mechanism require encrypting the entire repo, while at the other end only a limited number of specific secrets are encrypted, leaving the majority of the repository in clear text form.

The DIMS project started out with lots of Git repos that were narrowly focused on specific system components to try to modularize in a way that facilitated integrating open source tools written by other groups. The primary repository that needed secrets was the Ansible playbooks repository.

Attention: Regardless of which mechanism for managing secrets you chose, *everyone* with Git commit rights *must* have discipline when it comes to handling secrets. It only takes a few mistakes to cause a lot of cleanup headaches, or for an accidental commit followed by a missed review and unintended push to result in a password or key exposure crisis.

The DIMS project ran into this problem many times, with accidental commits including private keys, passwords, and unredacted sample data from production systems. It wasn't until the repos were going to be made public that reviews identified several of these mistakes, causing long delays while cleanup activities were added to code completion tasks.

There is a cost/benefit tradeoff that must be made between using more than just one shared "development" reposi-

tory location (to more closely vet and control commits to the "official" repository location) vs. the time and effort required to sanitize accidentally committed secrets and simultaneously delete all clones at the same time to prevent the secrets being accidentally pushed back into the repo.

The two mechanisms first tested by the DIMS project were:

- Ansible Vault
- git-crypt

5.6.1 Ansible Vault

Ansible Vault is a command-line tool provided by Ansible. It allows for application of encryption at a very granular level.

Vault is a password-based encryption system. This password can be stored in a file, but it must be shared to every user who is allowed access to the secret data files that are encrypted. This means there is still one step of having to figure out how to share the vault password.

Once the password is known by all parties, the process is pretty simple. To create a file you want to be encrypted,

```
$ ansible-vault create --vault-password-file=$PASS_FILE vaultfile.yml
```

To view the file, without being able to edit it,

```
$ ansible-vault view --vault-password-file=$PASS_FILE vaultfile.yml
```

To edit the file,

```
$ ansible-vault edit --vault-password-file=$PASS_FILE vaultfile.yml
```

To encrypt a file,

```
$ ansible-vault encrypt --vault-password-file=$PASS_FILE newvaultfile.yml
```

To decrypt a file,

```
$ ansible-vault decrypt --vault-password-file=$PASS_FILE newvaultfile.yml
```

When you commit a vault-protected file, it will be the encrypted file that is committed to Git. Thus, if you `decrypt` a file to view it, you'll have to `encrypt` it again, and the file will change, so you'll have to commit it again.

If you need to rekey a file,

```
$ ansible-vault rekey --new-vault-password-file=$NEW_PASS_FILE rekeyvaultfile.yml
```

To use Ansible to share secret information, one way is by copying an entire file of secrets. First, you must load a file with the secret information. This is generally done by creating a dictionary of information, including the destination of the secret file, the owner and group and mode of the file, as well as the actual contents of the file. You then run a task to load the secret file, which is decrypted with the Vault password and read by Ansible, then the information from that file is used to create the new file on the target machine.

Warning: One important thing to note is that you must use the `no_log` module in these types of tasks to keep Ansible from printing the secret information in the output of running plays.

```
- name: Load secret password file
  include_vars: "vault.yml"
  no_log: true
  when: ansible_os_family == "Debian"
  tags: [ ansible-server ]

- name: Copy secret password file
  copy:
    dest: "{{ item.key }}"
    content: "{{ item.value.content }}"
    owner: "{{ item.value.owner }}"
    group: "{{ item.value.group }}"
    mode: "{{ item.value.mode }}"
  with_dict: "{{ vault_password }}"
  when: ansible_os_family == "Debian"
  no_log: true
  tags: [ ansible-server ]
```

The following is an example of the `vault.yml` file:

```
---

# File: unencrypted version of vault.yml

password:
  /home/ansible/pass.txt
    owner: "ansible"
    group: "ansible"
    mode: "u=r,go="
    content: |
      Secretsecretsecret

# eof
```

Additionally, you can use vault to keep variables secret that may not be used to create a whole file, like an SSH key. For example, a username and password might be needed in a service's configuration file. All you need to do is create a Vault-encrypted file with those secrets, include the secret vars file in a task before needing to use those secret variables (say, in a template), and then the secret will be populated on the target machine.

However, unless using the `ansible-vault view` command, the secret variables file is encrypted, so it isn't searchable. A solution to this problem is to include a secret variable in a "normal" vars file, but don't include the actual secret there–set that variable to another variable.

Let's say we need a username and password for Service X that is going to run on several machines in deployment `local`. In our group_vars file for this deployment, `deployment-local.yml`, we'd define the username and password variables for Service X as follows:

```
...

serviceXUsername: "{{ vault_serviceXUsername }}"
serviceXPassword: "{{ vault_serviceXPassword }}"

...
```

We would use the `ansible-vault create` command to then define the `vault_*` variables with their actual secret data, as follows:

```
---

# File: unencrypted version of deployment-local-vault.yml

vault_serviceXUsername: "secretUsername"
vault_serviceXPassword: "secretPassword"

# eof
```

Now, the secret variables file has to be included before any variable defined within it is used:

```
- name: Load deployment secret variables file
  include_vars: "../../../inventory/group_vars/deployment-local-vault.yml"
  no_log: true
  when: ansible_os_family == "Debian"
  tags: [ ansible-server ]

- name: Use secret variables
  template: "src=test-secret.yml dest=/etc/test-secret.yml owner=root group=root mode=
↪{{ mode_0644 }}"
  no_log: true
  sudo: yes
  when: ansible_os_family == "Debian"
  tags: [ ansible-server ]
```

Helpful Links:

- http://docs.ansible.com/ansible/playbooks_best_practices.html#variables-and-vaults
- https://www.onwebsecurity.com/devops/safely-storing-ansible-playbook-secrets
- https://dantehranian.wordpress.com/2015/07/24/managing-secrets-with-ansible-vault-the-missing-guide-part-1-of-2/
- https://dantehranian.wordpress.com/2015/07/24/managing-secrets-with-ansible-vault-the-missing-guide-part-2-of-2/
- https://serversforhackers.com/video/ansible-using-vault

5.6.2 git-crypt

Another granular approach we looked at was `git-crypt`. This integrates encryption using GPG keys within the `git` ecosystem. It allows for encryption by file, rather than repo. In addition, since the encryption integrated in the `git` ecosystem, you don't have to use special commands to view or edit encrypted files; encrypted files are decrypted when a repo is checked out and encrypted when committed.

The trouble we have found is a problem many projects will have, especially open source ones. The files are encrypted with the GPG keys available at the time of a given file's encryption. This means that if other project members join after a file is encrypted, that file will have to be re-encrypted once the new member's GPG key has been integrated. All encrypted files would ahve to be ID'd and then re-encrypted. Additionally, this also compounds the key revocation problem as whoever has a private key to decrypt the files will still be able to see the old commits. The files can't just be re-encrypted, all the secrets they contain must be changed.

At this time, we have chosen to stick with `Vault` over `git-crypt` for the reasons listed above.

5.6.3 Git and Unix permissions

A final issue that must be known and kept in mind is the limitation in Git's ability to properly handle Unix permission masks. Unix permission masks (or `umask`) are bit masks that handle multiple permissions (i.e., `read`, `write`, and

execute) for multiple groups (i.e., owner, group, and other), along with some other permission bits (setgid, setuid, and sticky bits).

Git only pays attention to the execute permissions, and does so in a limited way. That means a file may have a mode 0755 (write permission only for user, but universal read and execute permission), or it may have a mode 0644 (same as above, but no execute permission for anyone).

To see this problem in action, we will use the keys.host.create script to generate some SSH key pairs. SSH only allows keys to be used if they have *no* read permission for other, so it generates keys with permission mask 0600 (see highlighted lines):

```
$ keys.host.create -d . -p EXAMPLE
$ ls -l
total 40
-rw-rw-r-- 1 dittrich dittrich  358 Mar 13 12:17 key_fingerprints.txt
-rw-rw-r-- 1 dittrich dittrich 1304 Mar 13 12:17 known_hosts.add
-rw------- 1 dittrich dittrich  668 Mar 13 12:17 ssh_host_dsa_key
-rw-r--r-- 1 dittrich dittrich  616 Mar 13 12:17 ssh_host_dsa_key.pub
-rw------- 1 dittrich dittrich  365 Mar 13 12:17 ssh_host_ecdsa_key
-rw-r--r-- 1 dittrich dittrich  282 Mar 13 12:17 ssh_host_ecdsa_key.pub
-rw------- 1 dittrich dittrich  432 Mar 13 12:17 ssh_host_ed25519_key
-rw-r--r-- 1 dittrich dittrich  112 Mar 13 12:17 ssh_host_ed25519_key.pub
-rw------- 1 dittrich dittrich 1675 Mar 13 12:17 ssh_host_rsa_key
-rw-r--r-- 1 dittrich dittrich  408 Mar 13 12:17 ssh_host_rsa_key.pub
```

Now, add the keys to Git and watch the permissions that Git gives to those files:

```
$ git add .
$ git commit -m 'Add keys'
[master (root-commit) 47f872b] Add keys
 10 files changed, 66 insertions(+)
 create mode 100644 keys/key_fingerprints.txt
 create mode 100644 keys/known_hosts.add
 create mode 100644 keys/ssh_host_dsa_key
 create mode 100644 keys/ssh_host_dsa_key.pub
 create mode 100644 keys/ssh_host_ecdsa_key
 create mode 100644 keys/ssh_host_ecdsa_key.pub
 create mode 100644 keys/ssh_host_ed25519_key
 create mode 100644 keys/ssh_host_ed25519_key.pub
 create mode 100644 keys/ssh_host_rsa_key
 create mode 100644 keys/ssh_host_rsa_key.pub
```

As you can see, all files that did not have the execute bit got a permissions mask of 100644, so all files that were 0600 will end up being pulled with a permissions mask of 0644. SSH will not allow a permission mask of 0644 on private keys, so if these were user keys SSH would not allow them to be used. To ensure that checking out or pulling these files will still work requires an extra step to fix these permissions, which is a little complicated and involves the use of Git hooks.

The simplest method is to use the public/private pairing to identify all files that end in .pub and change the mode to 0600 on the file with the .pub extension stripped (i.e., the associated private key). A script to do this may look something like this:

```
#!/bin/bash
echo "[+] Verifying private key permissions and correcting if necessary"
find * -type f -name '*.pub' |
  while read pubkey; do
    privkey=$(dirname $pubkey)/$(basename $pubkey .pub)
    if [[ -f $privkey ]]; then
      mode=$(stat -c %a $privkey)
```

```
      if [[ $? -ne 0 ]]; then
        echo "[-] Failed to get mode for $privkey"
      elif [[ "$mode" != "600" ]]; then
        echo "[+] Changing mode $mode to 600 on file $privkey"
        chmod 600 $privkey
      fi
    fi
  done
exit 0
```

Attention: You cannot add files in the `.git/hooks` directory, where hooks are found for execution by Git, to Git tracking. The `.git` directory is _not_ part of the Git repository commit structure. You can add a directory named `hooks/` at the top level of the tracked repo and create links into the `.git/hooks` directory. This has to be done at least once per clone of the repo, which is up to the person doing the clone to perform manually (or to use wrapper scripts around Git that do this on the initial clone operation).

Here is a `Makefile` that automates this process:

```
.PHONY: all
all: install-hooks

.PHONY: install-hooks
install-hooks:
        @(cd hooks; find * -type f -o -type l 2>/dev/null) | \
                while read hook; do \
                        echo "[+] Installing .git/hooks/$$hook"; \
                        ln -sf ../../hooks/$$hook .git/hooks/$$hook; \
                done
```

To preserve the actual permission masks in a general way for all files being committed to Git is much more complicated and goes beyond the needs of this particular issue. Examples of how to do this are found in the following references, or search for other options.

- githooks - Hooks used by Git
- How To Use Git Hooks To Automate Development and Deployment Tasks
- Retaining file permissions with Git

CHAPTER 6

Documenting DIMS Components

This chapter covers Sphinx and ReStructured Text (reST), and how they are used with ReadTheDocs (a hosted documentation site) and GitHub (a hosted Git source repository site) to document open source project repositories. It includes specifics of how Sphinx is used for documentation within the DIMS project.

6.1 Required Background Reading

Before trying to use Sphinx, it is important to understand how it works and what basic things you can do with it to produce organized and structured documentation that includes things like headings, tables, figures, images, links, cross-references to labelled items, and callout notes.

Start by taking less than five minutes and reading **all** of the *very short* Sphinx Style Guide. It will give you some insight into high-level concepts of Sphinx and reST.

Next, spend another 10-15 minutes and read through **all** of the slightly longer Documenting Your Project Using Sphinx document to see the full range of markup and directives supported by reST.

A short tutorial that includes an example is IBM's Easy and beautiful documentation with Sphinx.

A much longer (2+hours when delivered live) Sphinx Tutorial v0.1 by Brandon Rhodes from PyCon 2013 walks through the full range of tasks necessary to document a Python code project.

Lastly, read Problems with StructuredText to learn about limitations in reST and some ways to deal with them.

6.2 Why Sphinx?

Just to illustrate how widely Sphinx is used in the open source community, here is a list of project repos in Dave Dittrich's $GIT directory that use Sphinx (by virtue of their containing a Sphinx configuration file `conf.py` under a documentation directory):

```
[dittrich@localhost git]$ find . -name conf.py
./ansible/docsite/conf.py
./celery/docs/conf.py
```

```
./crits/documentation/src/conf.py
./cuckoo/docs/book/src/conf.py
./CybOXProject/python-cybox/docs/conf.py
./elasticsearch-dsl-py/docs/conf.py
./MAECProject/python-maec/docs/conf.py
./MozDef/docs/source/conf.py
./pika/docs/conf.py
./pyxb/doc/conf.py
./redis-py/docs/conf.py
./robotframework/doc/api/conf.py
./sphinx_rtd_theme/demo_docs/source/conf.py
./STIXProject/python-stix/docs/conf.py
./TAXIIProject/libtaxii/docs/conf.py
./thug/doc/source/conf.py
```

Sphinx, since it is a Python project, is effectively programmable and highly configurable and flexible. You can do parameterized creation of documents to make them unique to a site using an open source software product, can exercise tests in code, can produce HTML and LaTeX-derived PDF, all from the same source documentation files. That is just the start. Sphinx also produces search indexes, dynamic tables of contents, forward and back buttons in HTML pages, and many other helpful features for documenting a project. Because it effectively compiles the documentation, things like unit tests, functional tests, software version descriptions, insertion of Graphviz directed and undirected graphs to illustrate relationships between system components... The list goes on.

6.3 Manually Initiating a `docs` directory with `sphinx-quickstart`

The program `sphinx-quickstart` can be used to initiate a Sphinx document directory. It is important to understand the ramifications of the first three questions in the context of how other Sphinx tools (e.g., `sphinx-autobuild`) work. Use of `sphinx-autobuild` is covered later. Here are the first two questions you are faced with after running `sphinx-quickstart` and what results from the choice.

```
[dittrich@localhost tmp]$ sphinx-quickstart
Welcome to the Sphinx 1.2.3 quickstart utility.

Please enter values for the following settings (just press Enter to
accept a default value, if one is given in brackets).

Enter the root path for documentation.
> Root path for the documentation [.]:

You have two options for placing the build directory for Sphinx output.
Either, you use a directory "_build" within the root path, or you separate
"source" and "build" directories within the root path.
> Separate source and build directories (y/n) [n]: y
```

6.3.1 Separated source and build directories

Answering **y** to the second question (as shown above) results a having separate `source` and `build` directories, with the following structure:

```
.
+- Makefile
+- build
+- source
```

```
    +- _static
    +- _templates
    +- conf.py
    +- index.rst

4 directories, 3 files
```

When you initiate a build with `make html`, here is what the resulting directory contents will include:

```
.
+- Makefile
+- build
|   +- doctrees
|   |   +- environment.pickle
|   |   +- index.doctree
|   +- html
|       +- _sources
|       |   +- index.txt
|       +- _static
|       |   +- ajax-loader.gif
|       |   +- basic.css
|       |   +- comment-bright.png
|       |   +- comment-close.png
|       |   +- comment.png
|       |   +- default.css
|       |   +- doctools.js
|       |   +- down-pressed.png
|       |   +- down.png
|       |   +- file.png
|       |   +- jquery.js
|       |   +- minus.png
|       |   +- plus.png
|       |   +- pygments.css
|       |   +- searchtools.js
|       |   +- sidebar.js
|       |   +- underscore.js
|       |   +- up-pressed.png
|       |   +- up.png
|       |   +- websupport.js
|       +- genindex.html
|       +- index.html
|       +- objects.inv
|       +- search.html
|       +- searchindex.js
+- source
    +- _static
    +- _templates
    +- conf.py
    +- index.rst

8 directories, 31 files
```

Note: Notice how the `build/` directory now contains subdirectories `html/` and `doctrees/` directories. There were no files created or changed in `source/` directory by the make operation.

Warning: You should answer **y** to the second question. DIMS project repositories should have **separated** `source/` and `build/` directories.

6.3.2 Mixed source and build

Had the second and third questions above been answered with a **n**, this is what the resulting directory structure would look like:

```
.
|- Makefile
|- _build
|- _static
|- _templates
|- conf.py
+- index.rst

3 directories, 3 files
```

Notice the `conf.py` and `index.rst` files are located in the same directory root as `_build`. When you build this document with `make html`, the resulting directory structure now looks like this:

```
.
+- Makefile
+- _build
|   +- doctrees
|   |   +- environment.pickle
|   |   +- index.doctree
|   +- html
|       +- _sources
|       |   +- index.txt
|       +- _static
|       |   +- ajax-loader.gif
|       |   +- basic.css
|       |   +- comment-bright.png
|       |   +- comment-close.png
|       |   +- comment.png
|       |   +- default.css
|       |   +- doctools.js
|       |   +- down-pressed.png
|       |   +- down.png
|       |   +- file.png
|       |   +- jquery.js
|       |   +- minus.png
|       |   +- plus.png
|       |   +- pygments.css
|       |   +- searchtools.js
|       |   +- sidebar.js
|       |   +- underscore.js
|       |   +- up-pressed.png
|       |   +- up.png
|       |   +- websupport.js
|       +- genindex.html
|       +- index.html
|       +- objects.inv
|       +- search.html
|       +- searchindex.js
```

```
+- _static
+- _templates
+- conf.py
+- index.rst

7 directories, 31 files
```

Note: In this second example, the source files `index.rst` and the `conf.py` file are at the same directory level as the `_build/` directory (and all of its contents). Doing a `make html` or `make latexpdf` both cause the source directory `.` to change, because new files and directores were created within the `.` directory.

The `sphinx-quickstart` program gives you an option of separating the source directory from other directories. When this option is chosen, the result is a directory structure that has the `Makefile` at the top level with a `build` and `source` directory at the same directory level, which looks like this:

```
.
|- Makefile
|- build
+- source
    +- README.rst
    +- _static
    +- _templates
    +- conf.py
    +- developing.rst
    +- index.rst
    +- intro.rst
    +- license.rst
    +- quickstart.rst

4 directories, 8 files
```

6.4 Building Sphinx Documentation

You can build HTML manually with the `Makefile`, build PDF output with the `Makefile`, or automatically build HTML whenever files change on disk using `sphinx-autobuild`.

When you are ready to try building your documentation, start with manually building HTML output (which you can test locally with a browser). Once you understand how building HTML works, and know what to look for in terms of error messages and warnings, you will find it is faster and easier to create Sphinx documents using `sphinx-autobuild` and a browser in a second window.

6.4.1 Manually Building HTML

The most simple way to render Sphinx documents is to use the `Makefile` created by `sphinx-quickstart` using `make` as shown here:

```
[dittrich@localhost docs (dev)]$ make html
sphinx-build -b html -d build/doctrees   source build/html
Making output directory...
Running Sphinx v1.2.3
loading pickled environment... not yet created
loading intersphinx inventory from http://docs.python.org/objects.inv...
```

```
building [html]: targets for 8 source files that are out of date
updating environment: 8 added, 0 changed, 0 removed
reading sources... [ 12%] README
reading sources... [ 25%] continuousintegration
reading sources... [ 37%] deployconfigure
reading sources... [ 50%] developing
reading sources... [ 62%] documentation
reading sources... [ 75%] index
reading sources... [ 87%] introduction
reading sources... [100%] quickstart

looking for now-outdated files... none found
pickling environment... done
checking consistency... /Users/dittrich/git/dims-ci-utils/docs/source/README.rst::␣
↪WARNING: document isn't included in any toctree
done
preparing documents... done
writing output... [ 12%] README
writing output... [ 25%] continuousintegration
writing output... [ 37%] deployconfigure
writing output... [ 50%] developing
writing output... [ 62%] documentation
writing output... [ 75%] index
writing output... [ 87%] introduction
writing output... [100%] quickstart

writing additional files... genindex search
copying images... [100%] images/DD_home_page_small.jpg

copying downloadable files... [100%] /Users/dittrich/git/dims-ci-utils/docs/source/
↪images/DD_home_page.png

copying static files... done
copying extra files... done
dumping search index... done
dumping object inventory... done
build succeeded, 1 warning.

Build finished. The HTML pages are in build/html.
```

You can now load the page with a browser:

```
[dittrich@localhost docs (dev)]$ open -a Opera.app build/html/index.html
```

6.4.2 Manually Building PDF using LaTeX

Now, render the same document as a PDF file using LaTeX:

```
[dittrich@localhost docs (dev)]$ make latexpdf
sphinx-build -b latex -d build/doctrees   source build/latex
Making output directory...
Running Sphinx v1.2.3
loading pickled environment... done
building [latex]: all documents
updating environment: 0 added, 0 changed, 0 removed
looking for now-outdated files... none found
```

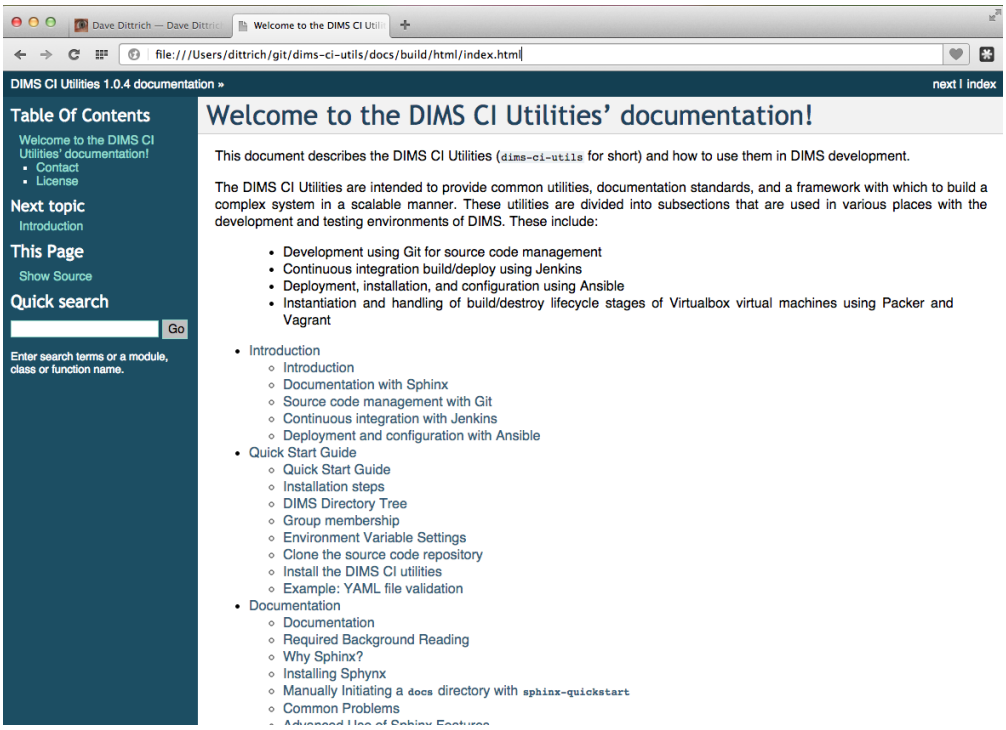


Fig. 6.1: This documentation, rendered on a Mac using Opera.

```
processing DIMSCIUtilities.tex... index introduction quickstart documentation␣
↪developing continuousintegration deployconfigure
resolving references...
writing... done
copying images... dims-ci-utils-doc.png DD_home_page_small.jpg
copying TeX support files...
done
build succeeded.
Running LaTeX files through pdflatex...
/Applications/Xcode.app/Contents/Developer/usr/bin/make -C build/latex all-pdf
pdflatex  'DIMSCIUtilities.tex'
This is pdfTeX, Version 3.14159265-2.6-1.40.15 (TeX Live 2014/MacPorts 2014_4)␣
↪(preloaded format=pdflatex)
 restricted \write18 enabled.
entering extended mode
(./DIMSCIUtilities.tex
LaTeX2e <2014/05/01>
Babel <3.9k> and hyphenation patterns for 43 languages loaded.
(./sphinxmanual.cls
Document Class: sphinxmanual 2009/06/02 Document class (Sphinx manual)
(/opt/local/share/texmf-texlive/tex/latex/base/report.cls
Document Class: report 2007/10/19 v1.4h Standard LaTeX document class
(/opt/local/share/texmf-texlive/tex/latex/base/size10.clo)))
(/opt/local/share/texmf-texlive/tex/latex/base/inputenc.sty

[ ...pages of output removed... ]

[25] [26]
Chapter 6.
```

```
[27] [28]
Chapter 7.
[29] [30]
Chapter 8.
(./DIMSCIUtilities.ind) [31] (./DIMSCIUtilities.aux) )
(see the transcript file for additional information){/opt/local/share/texmf-tex
live/fonts/enc/dvips/base/8r.enc}</opt/local/share/texmf-texlive/fonts/type1/ur
w/courier/ucrb8a.pfb></opt/local/share/texmf-texlive/fonts/type1/urw/courier/uc
rr8a.pfb></opt/local/share/texmf-texlive/fonts/type1/urw/courier/ucrro8a.pfb></
opt/local/share/texmf-texlive/fonts/type1/urw/helvetic/uhvb8a.pfb></opt/local/s
hare/texmf-texlive/fonts/type1/urw/helvetic/uhvbo8a.pfb></opt/local/share/texmf
-texlive/fonts/type1/urw/times/utmb8a.pfb></opt/local/share/texmf-texlive/fonts
/type1/urw/times/utmr8a.pfb></opt/local/share/texmf-texlive/fonts/type1/urw/tim
es/utmri8a.pfb>
Output written on DIMSCIUtilities.pdf (35 pages, 381656 bytes).
Transcript written on DIMSCIUtilities.log.
pdflatex finished; the PDF files are in build/latex.
```

Now open the PDF file (this example uses Mac OS X Preview.app, but you can also use `evince` on some Linux systems):

```
[dittrich@localhost docs (dev)]$ open build/latex/DIMSCIUtilities.pdf
```

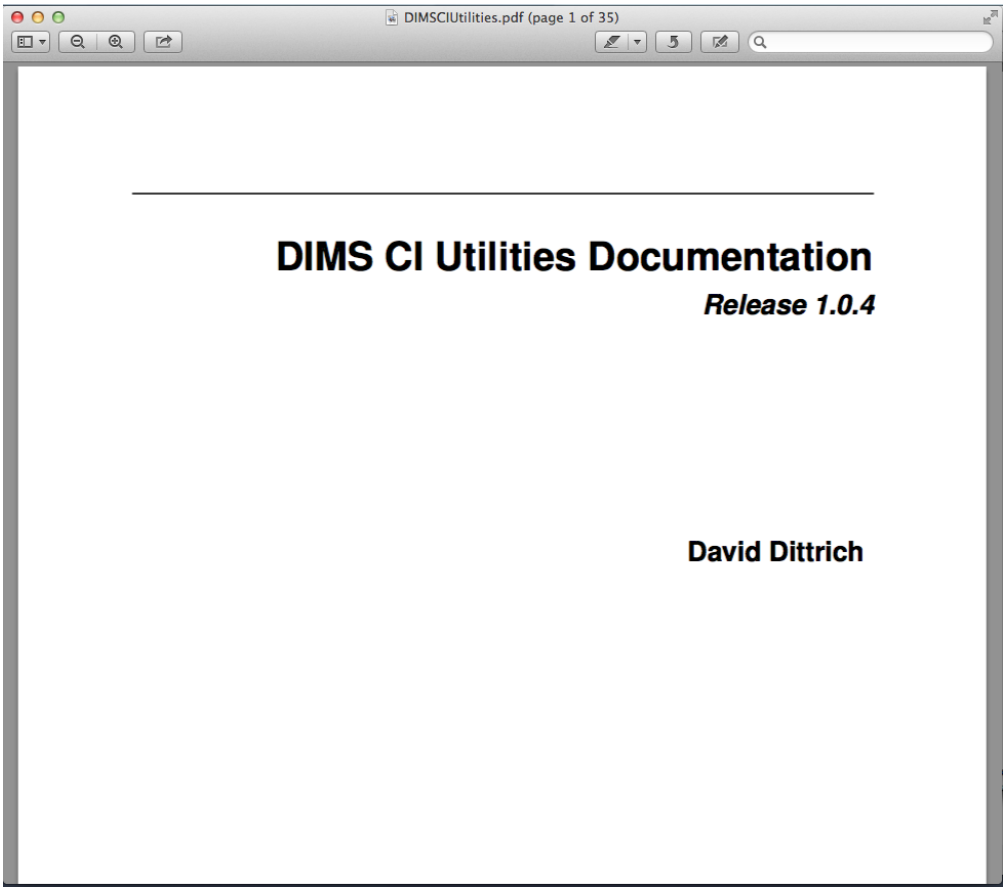


Fig. 6.2: This documentation, rendered using LaTeX on a Mac, viewed with Preview.

6.4.3 Automatically building HTML

Sphinx has a program called `sphinx-autobuild` that can monitor a directory for any file changes in that directory and below, re-building the document immediately upong detecting changes. When used to build HTML content, it makes the pages available on a local TCP port using a simple HTTP service (just like if the docs were put up on GitHub, readthedocs, etc.)

Note: You may need to install `sphinx-autobuild` using `pip` separately. Refer to section installingsphinx.

Here is where the importance of splitting the `source/` directory from `build/` directory becomes evident.

Invoke `sphinx-autobuild` from the command line in a separate terminal window, so you can watch the output for error messages. By default, `sphinx-autobuild` listens on 8000/tcp. (This can be changed with the `-p` flag on the command line). After starting `sphinx-autobuild` you then enter the URL that is produced (in this case, the URL is `http://127.0.0.1:8000`). Now edit files in another terminal or editor application window.

```
[dittrich@localhost docs (dev)]$ sphinx-autobuild --ignore '*.swp' source build/html
Serving on http://127.0.0.1:8000
[I 150105 18:50:45 handlers:109] Browser Connected: http://127.0.0.1:8000/
↪documentation.html
[I 150105 18:50:45 handlers:118] Start watching changes
[I 150105 18:50:48 handlers:74] Reload 1 waiters: None
[I 150105 18:50:48 web:1811] 200 GET /documentation.html (127.0.0.1) 16.57ms
[I 150105 18:50:48 web:1811] 304 GET /livereload.js (127.0.0.1) 1.08ms
[I 150105 18:50:48 web:1811] 200 GET /_static/pygments.css (127.0.0.1) 0.83ms
[I 150105 18:50:48 web:1811] 200 GET /_static/default.css (127.0.0.1) 0.62ms
[I 150105 18:50:48 web:1811] 200 GET /_static/jquery.js (127.0.0.1) 1.24ms
[I 150105 18:50:48 web:1811] 200 GET /_static/underscore.js (127.0.0.1) 1.09ms
[I 150105 18:50:48 web:1811] 200 GET /_static/doctools.js (127.0.0.1) 0.68ms
[I 150105 18:50:48 web:1811] 200 GET /_images/DD_home_page_small.jpg (127.0.0.1) 0.
↪86ms
[I 150105 18:50:48 web:1811] 200 GET /_static/basic.css (127.0.0.1) 0.46ms
[I 150105 18:50:48 web:1811] 200 GET /_images/dims-ci-utils-doc-html.png (127.0.0.1)␣
↪1.59ms
[I 150105 18:50:48 web:1811] 200 GET /_images/dims-ci-utils-doc-pdf.png (127.0.0.1) 0.
↪72ms
[I 150105 18:50:48 handlers:109] Browser Connected: http://127.0.0.1:8000/
↪documentation.html

+--------- source/documentation.rst changed ------------------------------------
| Running Sphinx v1.2.3
| loading pickled environment... not yet created
| No builder selected, using default: html
| loading intersphinx inventory from http://docs.python.org/objects.inv...
| building [html]: targets for 8 source files that are out of date
| updating environment: 8 added, 0 changed, 0 removed
| reading sources... [ 12%] README
| reading sources... [ 25%] continuousintegration
| reading sources... [ 37%] deployconfigure
| reading sources... [ 50%] developing
| reading sources... [ 62%] documentation
| reading sources... [ 75%] index
| reading sources... [ 87%] introduction
| reading sources... [100%] quickstart
/Users/dittrich/git/dims-ci-utils/docs/source/documentation.rst:281: WARNING: Literal␣
↪block ends without a blank line; unexpected unindent.
/Users/dittrich/git/dims-ci-utils/docs/source/documentation.rst:519: WARNING: Literal␣
↪block ends without a blank line; unexpected unindent.
```

```
|
| looking for now-outdated files... none found
| pickling environment... done
/Users/dittrich/git/dims-ci-utils/docs/source/README.rst:: WARNING: document isn't␣
↪included in any toctree
| checking consistency... done
| preparing documents... done
| writing output... [ 12%] README
| writing output... [ 25%] continuousintegration
| writing output... [ 37%] deployconfigure
| writing output... [ 50%] developing
| writing output... [ 62%] documentation
| writing output... [ 75%] index
| writing output... [ 87%] introduction
| writing output... [100%] quickstart
|
| writing additional files... genindex search
| copying images... [ 33%] dims-ci-utils-doc-pdf.png
| copying images... [ 66%] DD_home_page_small.jpg
| copying images... [100%] dims-ci-utils-doc-html.png
|
| copying downloadable files... [100%] /Users/dittrich/git/dims-ci-utils/docs/source/
↪images/DD_home_page.png
|
| copying static files... done
| copying extra files... done
| dumping search index... done
| dumping object inventory... done
| build succeeded, 3 warnings.
+------------------------------------------------------------------------------


+--------- source/documentation.rst changed ------------------------------------
| Running Sphinx v1.2.3
| loading pickled environment... done
| No builder selected, using default: html
| building [html]: targets for 0 source files that are out of date
| updating environment: 0 added, 0 changed, 0 removed
| looking for now-outdated files... none found
| no targets are out of date.
+------------------------------------------------------------------------------

[I 150105 18:51:17 handlers:74] Reload 1 waiters: None
[I 150105 18:51:17 web:1811] 200 GET /documentation.html (127.0.0.1) 1.70ms
[I 150105 18:51:17 web:1811] 200 GET /_static/default.css (127.0.0.1) 0.70ms
[I 150105 18:51:17 web:1811] 200 GET /_static/doctools.js (127.0.0.1) 0.76ms
[I 150105 18:51:17 web:1811] 200 GET /_static/underscore.js (127.0.0.1) 0.88ms
[I 150105 18:51:17 web:1811] 200 GET /_static/jquery.js (127.0.0.1) 1.26ms
[I 150105 18:51:17 web:1811] 200 GET /_static/pygments.css (127.0.0.1) 0.71ms
[I 150105 18:51:17 web:1811] 304 GET /livereload.js (127.0.0.1) 0.83ms
[I 150105 18:51:17 web:1811] 200 GET /_images/DD_home_page_small.jpg (127.0.0.1) 1.
↪04ms
[I 150105 18:51:17 web:1811] 200 GET /_static/basic.css (127.0.0.1) 0.54ms
[I 150105 18:51:17 web:1811] 200 GET /_images/dims-ci-utils-doc-html.png (127.0.0.1)␣
↪1.86ms
[I 150105 18:51:17 web:1811] 200 GET /_images/dims-ci-utils-doc-pdf.png (127.0.0.1) 0.
↪96ms
[I 150105 18:51:17 handlers:109] Browser Connected: http://127.0.0.1:8000/
↪documentation.html
```

Every time you change a file, `sphinx-autobuild` will rebuild it and your brower will be informed that it needs to reload the page so you can immediately see the results. This helps in developing Sphinx documentation quickly, as all you need to do is edit files and watch for error messages in the `sphinx-autobuild` window and see if the browser page shows what you want it to show.

Warning: The above example uses `--ignore '*.swp'` to avoid temporary swap files created by the `vim` editor. If you use an editor that creates temporary files using a different file extension, you should use that name instead. Otherwise, every time you open a file with the editor it will appear to `sphinx-autobuild` as though a source file changed and it will regenerate the document.

Warning: If you restart the `sphinx-autobuild` process, you will need to reconnect the browser to the `sphinx-autobuild` listening port, otherwise the browser will stop updating the page automatically at the end of each automatic build. Refreshing the page can fix this.

If you start the browser and attempt to re-open a previously used URL *before* you start `sphinx-autobuild`, you may experience a similar problem. Try to use `touch` to update a file, or edit a file and force a write operation. Either of these will trigger a rebuild and refresh of the browser, which should then keep it in sync.

The example above produces **a lot** of output in the `sphinx-autobuild` terminal output, which in practice makes it a little harder to see the error messages. To decrease the amount of output, you may want to add the `-q` flag (see also `sphinx-build -h` for how to control the underlying build process, and `sphinx-autobuild --help` for more autobuild options).

```
[dittrich@localhost docs (dev)]$ sphinx-autobuild -q --ignore '*.swp' source build/
↪html
```

Warning: By default, `sphinx-autobuild` will attempt to bind to port 8000/tcp. If that port is in use by another instance of `sphinx-autobuild`, you will get an error message. Use the `-p` flag to change the listening port number to something else (e.g., `-p 8001`).

6.5 Fixing errors

If there are any problems, Sphinx will call them out with warnings. Pay attention to the build output.

```
rm -rf build/*
sphinx-build -b html -d build/doctrees   source build/html
Making output directory...
Running Sphinx v1.2.3
loading pickled environment... not yet created
loading intersphinx inventory from http://docs.python.org/objects.inv...
building [html]: targets for 7 source files that are out of date
updating environment: 7 added, 0 changed, 0 removed
reading sources... [ 14%] README
reading sources... [ 28%] advanced
reading sources... [ 42%] developing
reading sources... [ 57%] index
reading sources... [ 71%] intro
reading sources... [ 85%] license
```

```
reading sources... [100%] quickstart

/Users/dittrich/git/dims-ci-utils/docs/source/intro.rst:26: WARNING: Inline literal␣
↪start-string without end-string.
/Users/dittrich/git/dims-ci-utils/docs/source/intro.rst:95: WARNING: Literal block␣
↪ends without a blank line; unexpected unindent.
looking for now-outdated files... none found
pickling environment... done
checking consistency...
/Users/dittrich/git/dims-ci-utils/docs/source/README.rst:: WARNING: document isn't␣
↪included in any toctree
/Users/dittrich/git/dims-ci-utils/docs/source/advanced.rst:: WARNING: document isn't␣
↪included in any toctree
/Users/dittrich/git/dims-ci-utils/docs/source/license.rst:: WARNING: document isn't␣
↪included in any toctree
done
preparing documents... done
writing output... [ 14%] README
writing output... [ 28%] advanced
writing output... [ 42%] developing
writing output... [ 57%] index
writing output... [ 71%] intro
writing output... [ 85%] license
writing output... [100%] quickstart

writing additional files... genindex search
copying static files... done
copying extra files... done
dumping search index... done
dumping object inventory... done
build succeeded, 23 warnings.

Build finished. The HTML pages are in build/html.
```

6.5.1 Typographic errors

Both of the errors seen in this first example above are simple typographical errors in the `intro.rst` file.

The first one, as it says, involves an improper literal on line 25:

```
25   A much longer (2+hours when delivered live) ``Sphinx Tutorial v0.1`_ by Brandon
26   Rhodes from PyCon 2013 walks through the full range of tasks necessary to
27   document a Python code project.
```

Here is the context for the second error message, regarding line 95:

```
73   Manually Initiating a ``docs`` directory with ``sphinx-quickstart``
74   ---------------------------------------------------------------------
75
76   The ``sphinx-quickstart`` program gives you an option of separating the soruce
77   directory from other directories. The result is a directory structure that
78   looks like this: ::
79
80       .
81       +- Makefile
82       +- build
83       +- source
```

```
84            +- README.rst
85            +- _static
86            +- _templates
87            +- conf.py
88            +- developing.rst
89            +- index.rst
90            +- intro.rst
91            +- license.rst
92            +- quickstart.rst
93
94        4 directories, 8 files
95    ..
96
```

As you can see, there is no blank line before the end of the literal block that ends on line 94 and before the reST comment tag (`..`) on line 25 (the one identified in the error message).

This is a simple error, but it happens quite frequently when inserting literal text examples. If need be, go back and re-read Sphinx Style Guide and Documenting Your Project Using Sphinx every now and then when you are starting out to get a refresher, and also have a browser window up with the The reStructuredText_ Cheat Sheet: Syntax Reminders or Quick reStructuredText quick reference guide to help while writing reST documents.

6.5.2 Link errors

A more subtle problem that comes up frequently when creating links to reference material in Sphinx documents is this error:

```
/Users/dittrich/git/dims-ci-utils/docs/source/intro.rst:274: ERROR: Unknown
target name: "the restructuredtext_ cheat sheet: syntax reminders".
```

See if you can spot the reason why by looking very closely at lines 274 and 323 before reading the explanation that follows:

```
...
273     are starting out to get a refresher, and also have a browser window
274     up with the `The reStructuredText_ Cheat Sheet: Syntax Reminders`_ or
275     `Quick reStructuredText`_ quick reference guide to help while
276     writing reST documents.
...
321    .. _Sphinx Tutorial v0.1: http://brandons-sphinx-tutorial.readthedocs.org/en/v0.
↪1/
323    .. _The reStructuredText_ Cheat Sheet: Syntax Reminders: http://docutils.
↪sourceforge.net/docs/user/rst/cheatsheet.txt
324    .. _Quick reStructuredText: http://docutils.sourceforge.net/docs/user/rst/
↪quickref.html
```

Unlike the links on lines 321 and 324, the target string specified on line 323 has *two* colons in it. This causes Sphinx to parse the line incorrectly (which in turn causes the `Unknown target name` error to be triggered). The error is not really on line 274, but is actually on line 323! It just presents itself as a missing target error on line 274. The solution is to make sure that all colons in targets for links are escaped, **except** the one before the URL, like this:

```
323    .. _The reStructuredText_ Cheat Sheet\: Syntax Reminders: http://docutils.
↪sourceforge.net/docs/user/rst/cheatsheet.txt
```

6.5.3 LaTeX image errors

You may get errors rendering LaTeX PDF documents that include image files. Such an error may look like this:

```
[dittrich@localhost docs (feature/docs)]$ make latexpdf
sphinx-build -b latex -d build/doctrees   source build/latex
Running Sphinx v1.2.3

...

Running LaTeX files through pdflatex...
/Applications/Xcode.app/Contents/Developer/usr/bin/make -C build/latex all-pdf
pdflatex  'DIMSCIUtilities.tex'
This is pdfTeX, Version 3.14159265-2.6-1.40.15 (TeX Live 2014/MacPorts 2014_4)␣
↪(preloaded format=pdflatex)

...

Chapter 1.
[3] [4] (/opt/local/share/texmf-texlive/tex/latex/psnfss/ts1pcr.fd) [5]

pdfTeX warning: pdflatex: arithmetic: number too big
! Dimension too large.
<argument> \ht \@tempboxa

l.348 ...=0.800\linewidth]{images/DD_home_page_small.png}

? q
OK, entering \batchmodemake[1]: *** [DIMSCIUtilities.pdf] Error 1
make: *** [latexpdf] Error 2
```

The solution to this is to use `mogrify -density 90 DD_home_page_small.png` to fix the image resolution metadata in the PNG file.

6.5.4 LaTeX Unicode rendering errors

Another error message that could occur when rendering the kind of text in the `code-block` seen in Section *Typographic errors* relates to the Unicode characters produced by the `tree` program to show the indentation levels.

Here is an error message (with the specific lines highlighted) that can show up in a Jenkins build process FAILURE message:

```
Running LaTeX files through pdflatex...
make -C build/latex all-pdf
make[1]: Entering directory `/var/lib/jenkins/jobs/dims-docs-deploy/workspace/ansible-
↪playbooks/docs/build/latex'
pdflatex  'AnsiblePlaybooksRepository.tex'
This is pdfTeX, Version 3.1415926-1.40.10 (TeX Live 2009/Debian)
entering extended mode
(./AnsiblePlaybooksRepository.tex

...

Underfull \hbox (badness 10000) in paragraph at lines 819--822
[]\T1/ptm/m/n/10 While it is not re-quired to in-stall dims-ci-utils, you prob-
a-bly will want to run the play-book
[11] [12] [13]
```

```

! Package inputenc Error: Unicode char \u8:â”œ not set up for use with LaTeX.

See the inputenc package documentation for explanation.
Type  H <return>  for immediate help.
 ...

l.1040 â”œ-- defaults

?
! Emergency stop.
 ...

l.1040 â”œ-- defaults

!  ==> Fatal error occurred, no output PDF file produced!
Transcript written on AnsiblePlaybooksRepository.log.
make[1]: *** [AnsiblePlaybooksRepository.pdf] Error 1
make[1]: Leaving directory `/var/lib/jenkins/jobs/dims-docs-deploy/workspace/ansible-
↪playbooks/docs/build/latex'
make: *** [latexpdf] Error 2
Build step 'Custom Python Builder' marked build as failure
Warning: you have no plugins providing access control for builds, so falling back to␣
↪legacy behavior of permitting any downstream builds to be triggered
Finished: FAILURE
```

Here is the specific block of text that triggered the rendering error message:

```
.. code-block:: bash

    - defaults
        - main.yml
    - files
        - base-requirements.txt
        - debian-virtualenv-prereqs.sh
        - dimsenv-requirements.txt
        - macos-virtualenv-prereqs.sh
    - meta
        - main.yml
    - tasks
        - main.yml
        - post_tasks.yml -> ../../../dims/post_tasks.yml
        - pre_tasks.yml -> ../../../dims/pre_tasks.yml
    - templates
        - bashrc.dims.virtualenv.j2
        - builddimsenvmod.sh.j2


..
```

The problem is that the long-dash character is not defined to LaTeX. This is done in the Sphinx `conf.py` file, and all DIMS documents should include these definitions because we frequently embed output of `tree`, which uses Unicode characters for line drawing. (Not all do, which causes random failures when adding text to Sphinx documents.)

```
latex_elements = {
 ...
# Additional stuff for the LaTeX preamble.
#
# The following comes from
```

```
# https://github.com/rtfd/readthedocs.org/issues/416
#
'preamble': "".join((
    '\DeclareUnicodeCharacter{00A0}{ }',     # NO-BREAK SPACE
    '\DeclareUnicodeCharacter{2014}{\dash}', # LONG DASH
    '\DeclareUnicodeCharacter{251C}{+}',     # BOX DRAWINGS LIGHT VERTICAL AND RIGHT
    '\DeclareUnicodeCharacter{2514}{+}',     # BOX DRAWINGS LIGHT UP AND RIGHT
)),
}
```

Note: See http://tex.stackexchange.com/questions/34604/entering-unicode-characters-in-latex

6.5.5 "LaTeX is not a TTY" errors

Another variation of errors during LaTeX rendering presents itself similarly to the previous error, but the problem is due to inability to map a Unicode character to a LaTeX macro: the problem is due to directly (or indirectly) sending output saved from Unix command line programs that do fancy things like coloring characters, etc, using ANSI escape sequences. While a terminal program that uses the Unix TTY subsystem may handle the ANSI escape sequences, and HTML renderers may know how to handle the ANSI escape sequences, LaTeX does not. Here is an example of this problem, excerpted from a Jenkins build job email message:

```
Started by user anonymous
[EnvInject] - Loading node environment variables.
Building in workspace /var/lib/jenkins/jobs/dims-docs-deploy/workspace

Deleting project workspace... done

[workspace] $ /bin/bash -xe /tmp/shiningpanda5607640542889107840.sh
+ jenkins.logmon
[workspace] $ /bin/bash -xe /tmp/shiningpanda5535708223044870299.sh
+ jenkins.dims-docs-deploy
[+++] jenkins.dims-docs-deploy: Deploying documentation
[+++] jenkins.dims-docs-deploy: Get global vars from jenkins.dims-defaults.
[+++] jenkins.dims-defaults Default variables
[+++]    PLAYBOOKSREPO=ansible-playbooks
[+++]    INVENTORYREPO=ansible-inventory
[+++]    GITURLPREFIX=git@git.devops.develop:/opt/git/
[+++]    MASTERBRANCH=master
[+++]    DEVBRANCH=develop
[+++]    DEVHOSTS=development
[+++]    MASTERHOSTS=production
[+++]    DEFAULTHOSTFILE=development
[+++]    DEFAULTANSIBLEBRANCH=develop
[+++]    DEFAULTINVENTORYBRANCH=develop
[+++]    DEFAULTREMOTEUSER=ansible

...
! Package inputenc Error: Keyboard character used is undefined
(inputenc)                in inputencoding `utf8'.

See the inputenc package documentation for explanation.
Type  H <return>  for immediate help.
 ...

```

```
l.5790 ...dl{}GIT/dims\PYGZhy{}dockerfiles/configu

!  ==> Fatal error occurred, no output PDF file produced!
Transcript written on UsingDockerinDIMS.log.
make[1]: *** [UsingDockerinDIMS.pdf] Error 1
make[1]: Leaving directory `/var/lib/jenkins/jobs/dims-docs-deploy/workspace/dims-
↪dockerfiles/docs/build/latex'
make: *** [latexpdf] Error 2
Build step 'Custom Python Builder' marked build as failure
Warning: you have no plugins providing access control for builds, so falling back to␣
↪legacy behavior of permitting any downstream builds to be triggered
Finished: FAILURE
```

To find the line in question (5790, in this case, called out in output line 34 above), manually trigger a LaTeX PDF build from the Sphinx document and then look for the LaTeX source file that corresponds with the PDF file name (seen in output line 38 above) in the `build/latex` subdirectory (in this case, it would be `$GIT/dims-dockerfiles/docs/build/latex/UsingDockerinDIMS.tex`) to find the character that causes the error:

```
[dimsenv] ~/dims/git/dims-dockerfiles/docs (develop) $ make latexpdf

 ...

! Package inputenc Error: Keyboard character used is undefined
(inputenc)                in inputencoding `utf8'.

See the inputenc package documentation for explanation.
Type  H <return>  for immediate help.
 ...

l.5789 ...dl{}GIT/dims\PYGZhy{}dockerfiles/configu

? ^Cmake[1]: *** Deleting file `UsingDockerinDIMS.pdf'
^Z
[1]+  Stopped                 make latexpdf
[dimsenv] ~/dims/git/dims-dockerfiles/docs (develop) $ kill -9 %1
[1]+  Killed: 9               make latexpdf
[dimsenv] ~/dims/git/dims-dockerfiles/docs (develop) $ pr -n build/latex/
↪UsingDockerinDIMS.tex | less

  ...

 5780    * VPN \PYGZsq{}01\PYGZus{}uwapl\PYGZus{}dimsdev2\PYGZsq{} is running
 5781    * VPN \PYGZsq{}02\PYGZus{}prsm\PYGZus{}dimsdev2\PYGZsq{} is running
 5782   [+++] Sourcing /opt/dims/etc/bashrc.dims.d/bashrc.dims.virtualenv ...
 5783   [+++] Activating DIMS virtual environment (dimsenv) [ansible\PYGZhy{}
↪playbooks v1.2.93]
 5784   [+++] (Create file /home/mboggess/.DIMS\PYGZus{}NO\PYGZus{}DIMSENV\PYGZus{}
↪ACTIVATE to disable)
 5785   [+++] Virtual environment \PYGZsq{}dimsenv\PYGZsq{} activated [ansible\PYGZhy
↪{}playbooks v1.2.93]
 5786   [+++] /opt/dims/bin/dims.install.dimscommands: won\PYGZsq{}t try to install␣
↪scripts in /opt/dims
 5787   [+++] Sourcing /opt/dims/etc/bashrc.dims.d/git\PYGZhy{}prompt.sh ...
 5788   [+++] Sourcing /opt/dims/etc/bashrc.dims.d/hub.bash\PYGZus{}completion.sh ...
 5789  ESC[1;34m[dimsenv]ESC[0m ESC[1;33mmboggess@dimsdev2:\PYGZti{}/core\PYGZhy{}
↪localESC[0m () \PYGZdl{} bash \PYGZdl{}GIT/dims\PYGZhy{}dockerfiles/configu
 5790   rations/elasticsearch/setup\PYGZus{}cluster.sh
 5791
```

```
35   5792   elasticsearch@.service                                0\PYGZpc{}    0      0.0KB/s
↪ \PYGZhy{}\PYGZhy{}:\PYGZhy{}\PYGZhy{} ETA
36   5793   elasticsearch@.service                              100\PYGZpc{} 1680      1.6KB/s
↪ 00:00
37   5794
38   5795   start\PYGZus{}elasticsearch\PYGZus{}cluster.sh                    0\PYGZpc{}
↪ 0      0.0KB/s   \PYGZhy{}\PYGZhy{}:\PYGZhy{}\PYGZhy{} ETA
39   5796   start\PYGZus{}elasticsearch\PYGZus{}cluster.sh                  100\PYGZpc{}
↪75      0.1KB/s   00:00
40   5797   ESC[1;34m[dimsenv]ESC[0m ESC[1;33mmboggess@dimsdev2:\PYGZti{}/core\PYGZhy{}
↪localESC[0m () \PYGZdl{} vagrant ssh core\PYGZhy{}01 \PYGZhy{}\PYGZhy{} \PYGZhy{}A
41   5798   VM name: core\PYGZhy{}01 \PYGZhy{} IP: 172.17.8.101
42   5799   Last login: Wed Sep  9 13:50:22 2015 from 10.0.2.2
43   5800
44   5801   CoreESC[38;5;206mOESC[38;5;45mSESC[39m alpha (794.0.0)
45   5802   ESC]0;core@core\PYGZhy{}01:\PYGZti{}^GESC[?1034hESC[01;32mcore@core\PYGZhy{}
↪01ESC[01;34m \PYGZti{} \PYGZdl{}ESC[00m ls
46   5803   ESC[0mESC[01;34minstancesESC[0m  start\PYGZus{}elasticsearch\PYGZus{}cluster.
↪sh  ESC[01;34mstaticESC[0m  ESC[01;34mtemplatesESC[0m
47   5804   ESC]0;core@core\PYGZhy{}01:\PYGZti{}^GESC[01;32mcore@core\PYGZhy{}01ESC[01;
↪34m \PYGZti{} \PYGZdl{}ESC[00m bash start\PYGZus{}elasticsearch\PYGZus{}cluster.sh
48   5805   ESC]0;core@core\PYGZhy{}01:\PYGZti{}^GESC[01;32mcore@core\PYGZhy{}01ESC[01;
↪34m \PYGZti{} \PYGZdl{}ESC[00m ESC[Ketcdctl cluster\PYGZhy{}hea
49    ...
```

Note: Pay close attention to the commands used to reproduce the error that Jenkins encountered from the command line. LaTeX, which is being invoked by Sphinx (a Python program that invokes `pdflatex` as a subprocess) has some problems getting the **CTRL-C** character (see line 14). To work around this, do the following:

1. Suspend the process with **CTRL-Z** (see line 15).
2. Identify the suspended job's number found within the square brackets on line 16: (`[1]+ Stopped ...`, in this case, job `1`).
3. Use the `kill` command (see `man kill` and `man signal`) to send the `-9` (non-maskable interrupt) signal to the suspended job (see line 17).
4. Use `pr -n` to add line numbers to the file and pass the output to a pager like `less` to find the line number called out by LaTeX (see lines 19 and 32).

As can be seen in line 32 above, the escape sequence **ESC[1;34m** (set foreground color 'Blue': see Bash Prompt HOWTO: Chapter 6. ANSI Escape Sequences: Colours and Cursor Movement) is causing LaTeX to fail.

The moral of the story is, only send properly mapped Unicode and/or UTF-8/ASCII text to Sphinx, so that when it does not fail when it invokes LaTeX.

Note: You can strip ANSI escape sequences in many ways. Google "strip ANSI escape sequences" to find some. Another way to handle this is to disable colorizing, or cut/paste command output as simple text rather than capturing terminal output with programs like `script`.

6.6 Common Tasks

6.6.1 Creating figures with thumbnails with links to larger images

Dave Dittrich's home page has a section with images, which uses low-resolution version for the main page and keeps the high-resolution image in a separate `_download` directory to be used as targets in the captions of those thumbnails. Here is a partial directory listing:

```
[dittrich@localhost sphinx (master)]$ tree
.
+- Makefile
+- _build
+- _download
+- . . .
+- images
|   +- Black_Mamba_Vienna-small.jpg
|   +- Black_Mamba_Vienna.jpg
|   +- Climbing_Gym_Manchester-small.jpg
|   +- Climbing_Gym_Manchester.jpg
|   +- DCA_Sunset-small.jpg
|   +- DCA_Sunset.jpg
|   +- QR-code-security-QR-code.gif
|   +- QR-code-security-QR-code.png
|   +- Sagrada_Familia_Barcelona-small.jpg
|   +- Sagrada_Familia_Barcelona.jpg
|   +- Screen-Shot-2014-12-31-at-1.15.34-PM.png
|   +- Seattle_Sunset_1-small.jpg
|   +- Seattle_Sunset_1.jpg
|   +- Seattle_Sunset_2-small.jpg
|   +- Seattle_Sunset_2.jpg
|   +- T-Rex-Chicago-small.jpg
|   +- T-Rex-Chicago.jpg
|   +- UW-Memorial-Way-Northbound-small.jpg
|   +- UW-Memorial-Way-Northbound.jpg
|   +- WA_OR_Volcanoes-small.jpg
|   +- WA_OR_Volcanoes.jpg
|   +- . . .
|   +- weber_guy.png
+- images.rst
+- . . .
+- www.rst
```

Here is how the figure with link works:

```
.. figure:: images/DD_home_page_small.jpg
   :alt: Dave Dittrich's home page
   :width: 80%
   :align: center

   A screen shot of Dave Dittrich's home page.
   :download:`Full size image <images/DD_home_page.png>
```

Note: Mouse over the image and right click and you will get the small image. Mouse over the words **Full size image** in the caption and right click and you get the... well, yes... full size image.

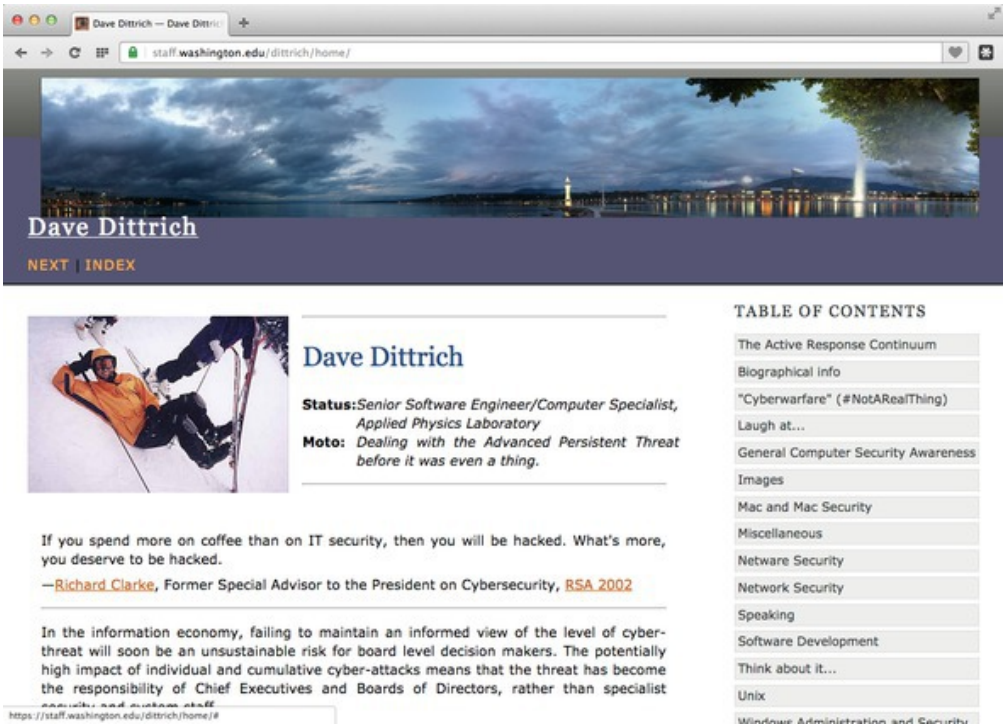


Fig. 6.3: A screen shot of Dave Dittrich's home page. `Full size image`

Note: The small version of an image can be created using ImageMagick's `convert` program like this:

```
$ convert DD_home_page.png -resize 50% DD_home_page_small.jpg
```

6.6.2 Section numbering

Sphinx does not render HTML documents with section numbers by default, but it will render LaTeX PDF documents with section numbers. To make these both consistent, add the `:numbered:` option to the `toctree` directive:

```
Contents:

.. toctree::
   :numbered:
   :maxdepth: 2
```

See http://sphinx-doc.org/markup/toctree.html and http://stackoverflow.com/questions/20061577/sphinx-section-numbering-for-more-than-level-3-headings-sectnum

6.6.3 Converting HTML content to Sphinx reST files

Many of the DIMS project documents are created using templates described in A forgotten military standard that saves weeks of work (by providing free project management templates), by Kristof Kovacs. Kovacs' web site has HTML versions of each of these templates in a ZIP archive. Download the archive and unpack it. Using the program `html2rest`, you can convert these documents to a reST format document.

Warning: The format of these HTML files is not parsed properly by `html2rest`, at least not without some pre-processing. Strip out the HTML break tags using the following commands:

```
$ sed 's|<br/>||g' ~/MIL-STD-498-templates-html-master/SRS.html > SRS.html
$ html2rest SRS.html > SRS.rst-orig
```

Once converted, you can now split the file `SRS.rst-orig` into separate sections, enable section numbering with the `:numbered:` option to the `toctree` directive and strip off the hard-coded numbers from sections, and add labels for sections (for cross-referencing). Here is an example of before and after for one such section from `SRS.html`:

Section 5 from the original HTML:

```
<h1>5.  Requirements traceability.</h1>
<p>This paragraph shall contain:
<ol type="a">
        <li>Traceability from each CSCI requirement in this specification to the␣
↪system (or subsystem, if applicable) requirements it addresses.  (Alternatively,␣
↪this traceability may be provided by annotating each requirement in Section 3.)

    Note:  Each level of system refinement may result in requirements not directly␣
↪traceable to higher-level requirements.  For example, a system architectural design␣
↪that creates multiple CSCIs may result in requirements about how the CSCIs will␣
↪interface, even though these interfaces are not covered in system requirements. ␣
↪Such requirements may be traced to a general requirement such as "system␣
↪implementation" or to the system design decisions that resulted in their generation.
↪ </li>
    <li>Traceability from each system (or subsystem, if applicable) requirement␣
↪allocated to this CSCI to the CSCI requirements that address it.  All system␣
↪(subsystem) requirements allocated to this CSCI shall be accounted for.  Those that␣
↪trace to CSCI requirements contained in IRSs shall reference those IRSs.</li>
</ol>
</p>
```

Section 5 from `SRS.rst-orig` after conversion with `html2rest`:

```
5. Requirements traceability.
=============================

This paragraph shall contain:

#. Traceability from each CSCI requirement in this specification to
   the system (or subsystem, if applicable) requirements it addresses.
   (Alternatively, this traceability may be provided by annotating each
   requirement in Section 3.) Note: Each level of system refinement may
   result in requirements not directly traceable to higher-level
   requirements. For example, a system architectural design that creates
   multiple CSCIs may result in requirements about how the CSCIs will
   interface, even though these interfaces are not covered in system
   requirements. Such requirements may be traced to a general requirement
   such as "system implementation" or to the system design decisions that
   resulted in their generation.
#. Traceability from each system (or subsystem, if applicable)
   requirement allocated to this CSCI to the CSCI requirements that
   address it. All system (subsystem) requirements allocated to this CSCI
   shall be accounted for. Those that trace to CSCI requirements
   contained in IRSs shall reference those IRSs.
```

Section 5 in a separate file `traceability.rst`:

```
.. _traceability:

Requirements traceability
=========================

This paragraph shall contain:

#. Traceability from each CSCI requirement in this specification to
   the system (or subsystem, if applicable) requirements it addresses.
   (Alternatively, this traceability may be provided by annotating each
   requirement in Section :ref:`requirements`.)

   .. note::

       Each level of system refinement may result in requirements not directly
       traceable to higher-level requirements. For example, a system
       architectural design that creates multiple CSCIs may result in
       requirements about how the CSCIs will interface, even though these
       interfaces are not covered in system requirements. Such requirements may
       be traced to a general requirement such as "system implementation" or to
       the system design decisions that resulted in their generation.

#. Traceability from each system (or subsystem, if applicable)
   requirement allocated to this CSCI to the CSCI requirements that
   address it. All system (subsystem) requirements allocated to this CSCI
   shall be accounted for. Those that trace to CSCI requirements
   contained in IRSs shall reference those IRSs.
```

The rendered HTML for section 5 can be seen in the figure *Correct rendering of note within list*.

6.6.4 Referencing subsections or figures

In the last example covered in section *Converting HTML content to Sphinx reST files*, note the label definition `.. _traceability:` right before the section heading *Requirements traceability*. A reference to this label will result in the section heading being used as the text for the hyperlink. This section itself is preceded by the label `referencinglabels`, which is rendered on reference as *Referencing subsections or figures*. This is the way to reference a sub-section (or figure, table, etc.) of a document.

Note: The section *Cross-referencing between documents with the sphinx.ext.intersphinx extension* builds on this concept of linking to arbitrary locations in a file by label.

6.7 Common Problems

6.7.1 Improperly referencing links to external documents

Sphinx documents are used to produce HTML, but reST itself is not like HTML in terms of links to external references. An HTML document may have many `HREF` elements that all have the same text to represent the hyperlink, but links in reST documents produce targets that can be cross-referenced from multiple places and need to be unique.

Here is output of `sphinx-autobuild` showing this problem:

```
+--------- source/.vmprovisioning.rst.swp changed ----------------------------
/Users/dittrich/git/dims-ci-utils/docs/source/vmprovisioning.rst:3: WARNING:␣
↪Duplicate explicit target name: "here".
/Users/dittrich/git/dims-ci-utils/docs/source/vmprovisioning.rst:3: WARNING:␣
↪Duplicate explicit target name: "here".
```

This message reports that the target name *help* has been duplicated twice within the text (meaning it occurs three times in definitions).

Note: The line number reported is not accurate for some reason. It is not actually on line 3 in `vmprovisioning.rst`.

Here are the three occurrances of the target *help* in the file:

```
 [...]

 When a new VM is created from base.ovf via import, it will of course
 inherit the complete hard drive contents from the OVF.  If the import
 is done without the ``keepnatmacs`` option, the new VM will have a new
 MAC address, which will then *not* match the details in the udev file,
 at which point the VM's network configuration will appear broken.
 This issue is a known one, see e.g. `here
 <http://kb.vmware.com/selfservice/microsites/search.do?language=en_US&cmd=displayKC&
↪externalId=1032790>`_
 or simply Google **ubuntu udev vm**.

 [...]

 Again ``packer`` is used to build the ``base-keyed`` OVF from the base OVF.
 We also perform an extra step (see ``base-keyed/base-keyed.json``) in
 converting the OVF to a Vagrant box file.  Further, we use the merged
 ``Vagrantfile`` idiom (Packer instructions `here
 <https://www.packer.io/docs/post-processors/vagrant.html>`_ and
 Vagrant description `here
 <https://docs.vagrantup.com/v2/vagrantfile/>`_) to embed SSH
 credentials into the box file itself.  These are then available to
 every Vagrant-managed VM created from that box file.  This eliminates
 the need to manage (via manual or automated edits) SSH user name and
 private key name in each/every ``Vagrantfile`` spawned from this box.

 [...]
```

These three links can all be made unique like this:

```
 [...]

 When a new VM is created from base.ovf via import, it will of course
 inherit the complete hard drive contents from the OVF.  If the import
 is done without the ``keepnatmacs`` option, the new VM will have a new
 MAC address, which will then *not* match the details in the udev file,
 at which point the VM's network configuration will appear broken.
 This issue is a known one, as seen in the VMWare Knowledge base article
 `Networking fails after cloning an Ubuntu virtual machine (1032790)`_
 or simply Google **ubuntu udev vm**.

 [...]
```

```
Again ``packer`` is used to build the ``base-keyed`` OVF from the base OVF.
We also perform an extra step (see ``base-keyed/base-keyed.json``) in
converting the OVF to a Vagrant box file.  Further, we use the merged
``Vagrantfile`` idiom to embed SSH credentials into the box file itself.
(See the Packer documentation on the `Vagrant Post-Processor`_ and
Vagrant documentation on the `Vagrantfile`_.)
These are then available to
every Vagrant-managed VM created from that box file.  This eliminates
the need to manage (via manual or automated edits) SSH user name and
private key name in each/every ``Vagrantfile`` spawned from this box.

[...]

.. _Networking fails after cloning an Ubuntu virtual machine (1032790): http://kb.
↪vmware.com/selfservice/microsites/search.do?language=en_US&cmd=displayKC&
↪externalId=1032790
.. _Vagrant Post-Processor: https://www.packer.io/docs/post-processors/vagrant.html
.. _Vagrantfile: https://docs.vagrantup.com/v2/vagrantfile/
```

6.7.2 Not having the proper white space around literal blocks

We saw this example, and how to fix it, in Section *Fixing errors*.

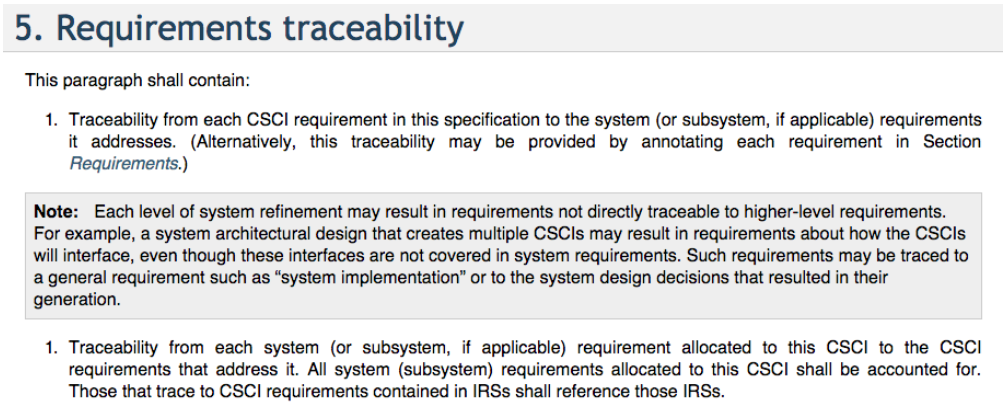
6.7.3 Using inconsistent indentation in literal blocks and directives

Say you are trying to create a list, and you want to include notes or warnings in one of the list items. Here are examples of the wrong and right way to do this.

Source code (incorrect indentation within list):

```
.. _traceability:

Requirements traceability
=========================

This paragraph shall contain:

#. Traceability from each CSCI requirement in this specification to
   the system (or subsystem, if applicable) requirements it addresses.
   (Alternatively, this traceability may be provided by annotating each
   requirement in Section :ref:`requirements`.)

.. note::

   Each level of system refinement may result in requirements not directly
   traceable to higher-level requirements. For example, a system
   architectural design that creates multiple CSCIs may result in
   requirements about how the CSCIs will interface, even though these
   interfaces are not covered in system requirements. Such requirements may
   be traced to a general requirement such as "system implementation" or to
   the system design decisions that resulted in their generation.

#. Traceability from each system (or subsystem, if applicable)
   requirement allocated to this CSCI to the CSCI requirements that
```

```
   address it. All system (subsystem) requirements allocated to this CSCI
   shall be accounted for. Those that trace to CSCI requirements
   contained in IRSs shall reference those IRSs.
```



5. Requirements traceability

This paragraph shall contain:

1. Traceability from each CSCI requirement in this specification to the system (or subsystem, if applicable) requirements it addresses. (Alternatively, this traceability may be provided by annotating each requirement in Section *Requirements*.)

Note: Each level of system refinement may result in requirements not directly traceable to higher-level requirements. For example, a system architectural design that creates multiple CSCIs may result in requirements about how the CSCIs will interface, even though these interfaces are not covered in system requirements. Such requirements may be traced to a general requirement such as "system implementation" or to the system design decisions that resulted in their generation.

1. Traceability from each system (or subsystem, if applicable) requirement allocated to this CSCI to the CSCI requirements that address it. All system (subsystem) requirements allocated to this CSCI shall be accounted for. Those that trace to CSCI requirements contained in IRSs shall reference those IRSs.

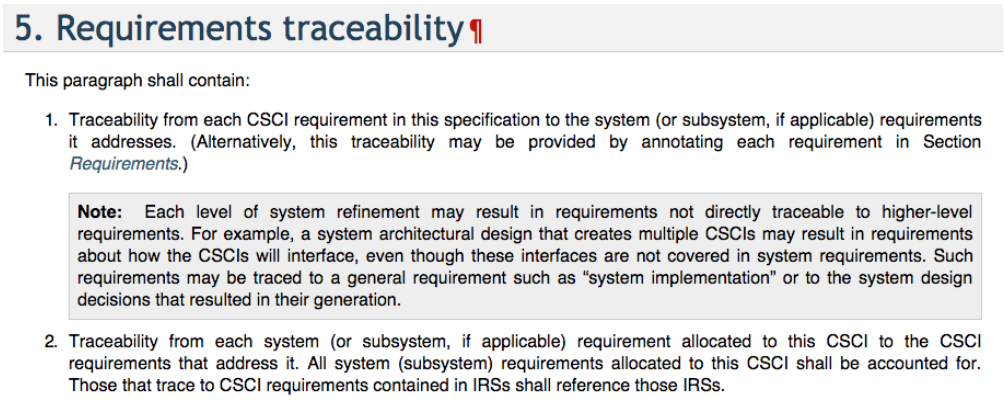
Fig. 6.4: Incorrect rendering of note within list

Source code (correct indentation within list):

```
.. _traceability:

Requirements traceability
=========================

This paragraph shall contain:

#. Traceability from each CSCI requirement in this specification to
   the system (or subsystem, if applicable) requirements it addresses.
   (Alternatively, this traceability may be provided by annotating each
   requirement in Section :ref:`requirements`.)

   .. note::

       Each level of system refinement may result in requirements not directly
       traceable to higher-level requirements. For example, a system
       architectural design that creates multiple CSCIs may result in
       requirements about how the CSCIs will interface, even though these
       interfaces are not covered in system requirements. Such requirements may
       be traced to a general requirement such as "system implementation" or to
       the system design decisions that resulted in their generation.

#. Traceability from each system (or subsystem, if applicable)
   requirement allocated to this CSCI to the CSCI requirements that
   address it. All system (subsystem) requirements allocated to this CSCI
   shall be accounted for. Those that trace to CSCI requirements
   contained in IRSs shall reference those IRSs.
```

6.7.4 Having multiple colons in link target labels

It is easy to get used to using directives like `.. figure::` or `.. note::` and placing the double-colon after them. Labels look similar, but are not directives. They also have the underscore in front of them and should look like:



5. Requirements traceability ¶

This paragraph shall contain:

1. Traceability from each CSCI requirement in this specification to the system (or subsystem, if applicable) requirements it addresses. (Alternatively, this traceability may be provided by annotating each requirement in Section *Requirements*.)

   Note: Each level of system refinement may result in requirements not directly traceable to higher-level requirements. For example, a system architectural design that creates multiple CSCIs may result in requirements about how the CSCIs will interface, even though these interfaces are not covered in system requirements. Such requirements may be traced to a general requirement such as "system implementation" or to the system design decisions that resulted in their generation.

2. Traceability from each system (or subsystem, if applicable) requirement allocated to this CSCI to the CSCI requirements that address it. All system (subsystem) requirements allocated to this CSCI shall be accounted for. Those that trace to CSCI requirements contained in IRSs shall reference those IRSs.

Fig. 6.5: Correct rendering of note within list

```
.. _label:

This is a reference to :ref:`label`.
```

6.8 Advanced Use of Sphinx Features

This section discusses more advanced features of Sphinx to accomplish particular tasks.

To illustrate two cases, consider the following:

- During DIMS development, there will be (1) a DIMS instance for developers to use, (2) another DIMS instance for test and evaluation prior to release, and (3) yet another instance for user acceptance and functional testing that will be used by the PRISEM user base.
- In production, there will be an instance of DIMS deployed for different groups in multiple parts of the country, each with their own name, organizational structure and policies, etc.

In order to produce documentation that provides a sufficient level of precise detail so as to be immediately useful, documents for DIMS will need to be build by doing parameterized construction of documentation based on case-specific parameters.

Put yourself in a DIMS user's shoes. Which of the following two examples would be more useful to you?

Note: To access the DIMS front end, connect your browser to: `https://dims.example.com:12345/dimsapi/`

Note: To access the DIMS front end, connect your browser to the specific host and port configured for the login portal server, followed by the string "/dimsapi/". Ask your site administrator for the details.

Every instantiation of the full DIMS system (comprised of many separate service components) will be unique in several run-time aspects. Each will have its own IP address block, its own top level Domain Name System name, its own organizational name, its own policies and its own membership. That is just a start. One set of documentation cannot possibly be generalized in a way that it can be used by everyone, without reading like the second example above. Each instantiation needs its own uniquely produced documentation, which means **documenation must be**

configured just like the system itself is configured. If the documentation must be hand-edited for each user, that places a huge burden on those wanting to implement DIMS and the system will not be used widely enough to have the intended impact.

6.8.1 Cross-referencing between documents with the `sphinx.ext.intersphinx` extension

ReST supports Cross-referencing arbitrary locations within a document using `:ref:`. To reference arbitrary locations (by their label) in other documents requires the Sphinx extension `sphinx.ext.intersphinx`. (See the documentation for sphinx.ext.intersphinx and Section *Referencing subsections or figures* for more on labels.)

Intersphinx links allow, for example, cross referencing a test in the Test Plan document to a requirement or user story in the Requirements document to provide requirements traceability in testing.

Mapping URLs to documents

The first step is to enable the extension by making sure it is included in the `conf.py` file:

```
# Add any Sphinx extension module names here, as strings. They can be
# extensions coming with Sphinx (named 'sphinx.ext.*') or your custom
# ones.
extensions = [
    'sphinx.ext.autodoc',
    'sphinx.ext.doctest',
    'sphinx.ext.todo',
    'sphinx.ext.intersphinx',
    'sphinx.ext.graphviz',
    'sphinx.ext.ifconfig',
]
```

When you build HTML output, any labels that are defined in your reST files e recorded in an *inventory* file. By default, the inventory is named `objects.inv`.

To cross-reference the `objects.inv` files from other documents requires a mapping of these inventories to symbolic name to define a label namespace for use in `:ref:` directives.

Note: You may use multiple targets for each inventory file, which is necessary when you are building multiple documents locally before they have been published in their final internet-accessible web site (e.g, the Read the Docs site). Obviously, if the remote inventory does not exist, it cannot be used (but when it does exist, you may want to use it instead of a local copy). Documents built automatically for publication with Jenkins would not have local copies, so they automatically would link with remote versions.

Warning: Because of the *chicken/egg* problem just described, document sets that are to be cross-linked would need to be rendered *twice* in order to first generate the inventory file that is used by other documents that reference it, and to get those inventories from the other documents in order to reference them. This is similar to how LaTeX works, where the recommendation is to run `pdflatex` twice, then run `bibtex` for bibliographies, then run `pdflatex` one last time to get cross-references and citations set up properly.

In the example below, both *local* and *remote* locations are specified.

Warning: You cannot use '-' in the symbol that maps the inventory files, so the following examples simply remove that character from the Git repo names.

```
intersphinx_cache_limit = -1   # days to keep the cached inventories (0 == forever)
intersphinx_mapping = {
        'dimsocd': ('http://app.devops.develop:8080/docs/develop/html/dims-ocd',
                    ('../../dims-ocd/build/html/objects.inv',None)),
        'dimsad': ('http://app.devops.develop:8080/docs/develop/html/dims-ad',
                    ('../../dims-ad/build/html/objects.inv',None)),
        'dimstp': ('http://app.devops.develop:8080/docs/develop/html/dims-tp',
                    ('../../dims-tp/build/html/objects.inv',None))
}
```

Linking to the label

In the reST document (in this case, the `referenceddocs.rst` file), normal `:ref:` directives are used, but the target of the `:ref:` includes the name of the inventory prepended to the label so as to map to the proper URL. The first reference in this example maps to the Operational Concept Description document:

```
.. _referenceddocs:

Referenced Documents
====================

The following documents describe the DIMS project and provide background
material related to tasking.

#. :ref:`dimsocd:dimsoperationalconceptdescription`

#. :ref:`dimsad:dimsarchitecturedesign`

#. :ref:`dimstp:dimstestplan`

#. HSHQDC-13-C-B0013, "From Local to Gobal Awareness: A Distributed Incident␣
↪Management System," Draft contract, Section C - Statement of Work (marked up␣
↪version)

#. MIL-STD-498, Military Standard Software Development and Documentation,
   AMSC No. N7069, Dec. 1994.
```

The label `dimsoperationalconceptdescription` occurs in the `index.rst` file on line 3, immediately preceding the title on line 6 (which has the release number inserted into it).

```
1  .. DIMS Operational Concept Description documentation master file.
2
3  .. _dimsoperationalconceptdescription:
4
5  ================================================
6  DIMS Operational Concept Description v |release|
7  ================================================
8
9  .. topic:: Executive Summary
10
11     Since HSPD-7 was released in 2003, the Department of Homeland Security has
12     had a core mission of working to protect the nationâ€™s critical
```

```
13      infrastructure. In 2008, the *National Response Framework* was released, and
14      ...
```

The final rendered *DIMS System Requirements* document has links to the related *DIMS Operational Concept Description*, *DIMS Architecture Design*, and *DIMS Test Plan* documents, all with their current release number visible for precise cross-referencing.

Note: Documents released from the `master` branch, all at once, will be easier to trace back to the code base for which they apply.

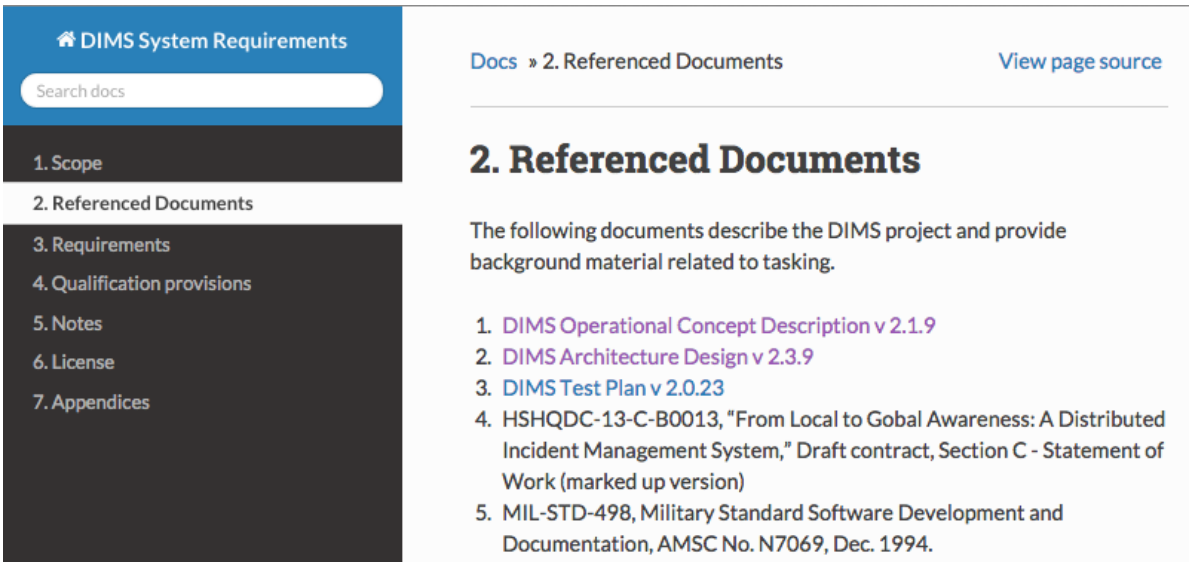


Fig. 6.6: Rendered intersphinx links

When you build the document, you will see the `objects.inv` files being loaded:

```
(dimsenv)[dittrich@localhost dims-sr (develop)]$ make html
Makefile:27: warning: overriding commands for target `help'
/opt/dims/etc/Makefile.dims.global:48: warning: ignoring old commands for target `help
↪'
sphinx-build -b html -d build/doctrees   source build/html
Running Sphinx v1.3.1+
loading pickled environment... done
loading intersphinx inventory from http://app.devops.develop:8080/docs/develop/html/
↪dims-ocd/objects.inv...
loading intersphinx inventory from http://app.devops.develop:8080/docs/develop/html/
↪dims-ad/objects.inv...
loading intersphinx inventory from http://app.devops.develop:8080/docs/develop/html/
↪dims-tp/objects.inv...
building [mo]: targets for 0 po files that are out of date
building [html]: targets for 0 source files that are out of date
updating environment: [config changed] 8 added, 0 changed, 0 removed
reading sources... [ 12%] appendices
reading sources... [ 25%] index
...
```

6.8.2 Insertion of text using direct substitution

Sphinx has ways of producing customized output when documents are built using direct textual substitution, and through execution of programs from within Sphinx. The simplest method is direct substitution.

Say you want a copyright symbol in a document. You start by selecting or creating a file that maps strings surrounded by pipe characters to some other string. There is a file called `isonum.txt` that does this for many Unicode characters, like the copyright symbol. The first 20 lines of this file look like this:

```
1  .. This data file has been placed in the public domain.
2  .. Derived from the Unicode character mappings available from
3     <http://www.w3.org/2003/entities/xml/>.
4     Processed by unicode2rstsubs.py, part of Docutils:
5     <http://docutils.sourceforge.net>.
6
7  .. |amp|    unicode:: U+00026 .. AMPERSAND
8  .. |apos|   unicode:: U+00027 .. APOSTROPHE
9  .. |ast|    unicode:: U+0002A .. ASTERISK
10 .. |brvbar| unicode:: U+000A6 .. BROKEN BAR
11 .. |bsol|   unicode:: U+0005C .. REVERSE SOLIDUS
12 .. |cent|   unicode:: U+000A2 .. CENT SIGN
13 .. |colon|  unicode:: U+0003A .. COLON
14 .. |comma|  unicode:: U+0002C .. COMMA
15 .. |commat| unicode:: U+00040 .. COMMERCIAL AT
16 .. |copy|   unicode:: U+000A9 .. COPYRIGHT SIGN
17 .. |curren| unicode:: U+000A4 .. CURRENCY SIGN
18 .. |darr|   unicode:: U+02193 .. DOWNWARDS ARROW
19 .. |deg|    unicode:: U+000B0 .. DEGREE SIGN
20 .. |divide| unicode:: U+000F7 .. DIVISION SIGN
```

Note: This is how to visually parse line 16: The `..` at the start to indicate a reST directive is being used, `|copy|` as the string to match, `unicode::  U+000A9` as a reST directive for a Unicode character, and `.. COPYRIGHT SIGN` as a comment that explains this is the copyright sign. The comment is unecessary, but helps explain what is being mapped.

You must first include the map before any substitutions will be recognized, then wherever the string `|copy|` occurs, the Unicode character `U+000A9` will be inserted. Here is a simple example of how to do this:

```
.. include:: <isonum.txt>

Copyright |copy| 2014-2017 University of Washington. All rights reserved.
```

This code renders as follows:

Copyright © 2014-2017 University of Washington. All rights reserved.

6.8.3 Insertion of text programmatically

A more complicated way of text substitution is by using the fact that Sphinx is a Python program, which can include and execute Python code at run time.

Let's start by creating a minimial Sphinx doc set using `sphinx-quickstart`.

We then modify the `conf.py` file by uncommenting the path modification line as follows:

```
# add these directories to sys.path here. If the directory is relative to the
# documentation root, use os.path.abspath to make it absolute, like shown here.
sys.path.insert(0, '/opt/dims/etc')
```

Next, put this line at the very end of the file:

```
from rst_prolog import *
```

Create the file `/opt/dims/etc/rst_prolog.py` and insert an `rst_prolog` string that is used by Sphinx before generating any output files:

```
rst_prolog = """
.. |dims_ftw|    replace:: for the win
"""
```

Here is a minimal Sphinx file that includes the variable that we will substitute at compile-time:

```
.. Sphinx Demo repository documentation master file, created by
   sphinx-quickstart on Tue Dec 30 12:43:11 2014.
   You can adapt this file completely to your liking, but it should at least
   contain the root `toctree` directive.

Welcome to Sphinx Demo repository's documentation!
==================================================

.. toctree::
   :maxdepth: 2

.. include: <rst_prolog>

This is |dims_ftw|!!!

Indices and tables
==================

* :ref:`genindex`
* :ref:`modindex`
* :ref:`search`
```

When you render this file with `make html` and then load it with a browser (in this case, `lynx`), you get the following:

```
#Welcome to Sphinx Demo repository's documentation! -- Sphinx.. (p1 of 2)
   #Sphinx Demo Repository 1.0 documentation

Navigation

     * index
     * Sphinx Demo Repository 1.0 documentation

Welcome to Sphinx Demo repository's documentation!

This is for the win!!!

Indices and tables

     * Index
     * Module Index
     * Search Page
```

```
Table Of Contents
-- press space for next page --
  Arrow keys: Up and Down to move.  Right to follow a link; Left to go back.
 H)elp O)ptions P)rint G)o M)ain screen Q)uit /=search [delete]=history list
```

Warning: When you render this document, Python includes the `rst_prolog.py` file (which is actually Python code) and will produce a `.pyc` file. You may need to delete it, if and when you remove the associated `.py` file.

```
[dittrich@localhost docs (dev)]$ ls -l /opt/dims/etc/
total 24
-rw-r--r--  1 dims      dims  3046 Dec 30 10:11 Makefile.dims.global
-rw-r--r--  1 dittrich  dims    58 Dec 30 15:13 rst_prolog.py
-rw-r--r--  1 dittrich  dims   177 Dec 30 15:14 rst_prolog.pyc
```

6.8.4 Inserting a graph using Graphviz

Sphinx uses Graphviz to render directed and undirected graphs inline in a document. To insert a graph, create a DOT language file to describe the graph, then reference the file using the `graphviz::` directive.

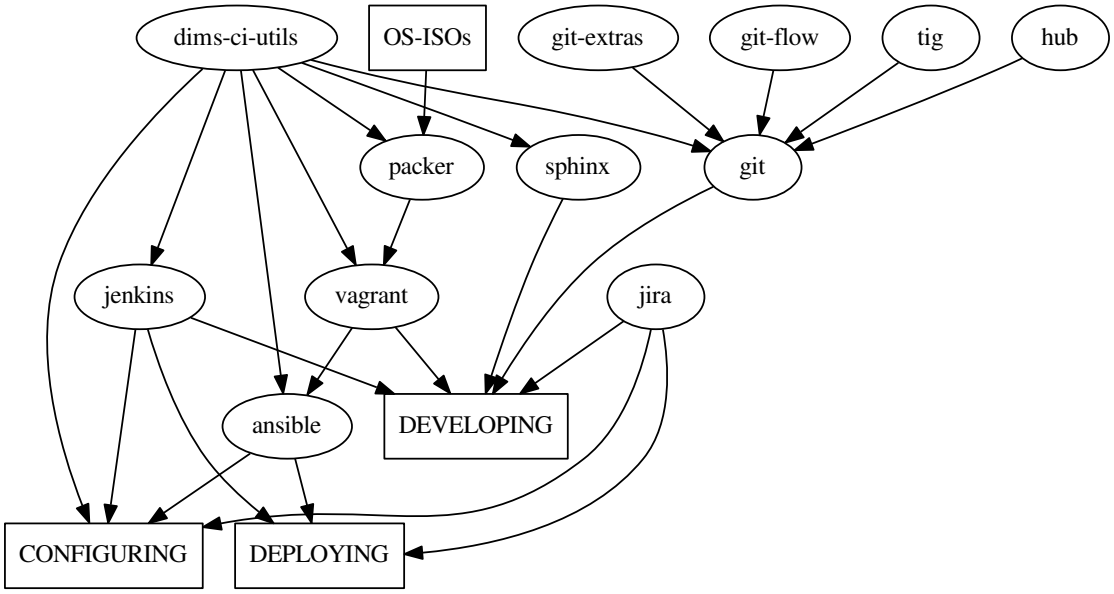


Fig. 6.7: Relationships between tools and processes in DIMS

THe DOT file for the graph above looks like this:

```
digraph tools {
    "DEPLOYING" [shape=rectangle];
        "DEVELOPING" [shape=rectangle];
        "CONFIGURING" [shape=rectangle];
```

```
        "OS-ISOs" [shape=rectangle];
        "dims-ci-utils" -> "jenkins";
        "dims-ci-utils" -> "ansible";
        "dims-ci-utils" -> "packer";
        "dims-ci-utils" -> "vagrant";
        "dims-ci-utils" -> "sphinx";
        "dims-ci-utils" -> "git";
        "git-extras" -> "git";
        "git-flow" -> "git";
        "tig" -> "git";
        "hub" -> "git";
        "OS-ISOs" -> "packer";
        "packer" -> "vagrant";
    "vagrant" -> "ansible";
    "vagrant" -> "DEVELOPING";
        "git" -> "DEVELOPING";
        "sphinx" -> "DEVELOPING";
        "jira" -> "DEVELOPING";
        "jira" -> "DEPLOYING";
        "jira" -> "CONFIGURING";
        "jenkins" -> "DEVELOPING";
        "ansible" -> "DEPLOYING";
        "jenkins" -> "DEPLOYING";
        "dims-ci-utils" -> "CONFIGURING";
        "ansible" -> "CONFIGURING";
        "jenkins" -> "CONFIGURING";
}
```

Note: You can find a Gallery of example DOT files at the Graphviz web site that shows how to do more advanced things, such as labelled edges.

CHAPTER 7

Continuous Integration

7.1 Continuous Integration

Continuous Integration is a software engineering process where multiple developers merge their working code into a coherent system on a regular basis, allowing for easier testing of code changes and integration of disparate parts of the system. Using a combination of a build system (Jenkins, in this case) and triggers invoked by the source code management system (Git, in this case), a change to source code results in that code being compiled, bundled, and installed (as necessary) onto the hosts where it needs to run to serve its function within the system as a whole.

Continuous Integration works well in a software engineering environment using Agile/Scrum.

7.2 How source changes are propagated

This section summarizes how changes in source repos are propagated using Jenkins and Ansible. You can find more information in the documentation for the `ansible-inventory` and `ansible-playbooks` repositories.

Git repos containing DIMS software under development contain "post-receive" hooks which notify the Jenkins server when changes are pushed to a repository. We are currently using two kinds of hooks: 1) A general hook which notifies Jenkins that a push has occured, and 2) A hook which calls a parameterized Jenkins job when a push has occured.

For the general hook, Jenkins jobs essentially "listen" for the notifications. A Jenkins job specifies the repository and branch it wishes to be notified about, as well as optionally specifying particular directory locations it is monitoring. When a notification is received that matches, the job will determine if any actually source changes occurred. If so, the job is run.

The "parameterized" hook is used to call a parameterized Jenkins documentation job when a push is received in a system documentation repository. The Jenkins job builds the documentation in the repo and deploys it (using Ansible) to any documentation servers in the system inventory that correspond to the branch that was updated.

Attention: In general, each repository with DIMS software under development will have a Jenkins job "listening" for each branch of the repository that we want to build and deploy continuously. Note that Jenkins jobs can be

triggered by changes to more than one branch, but we found it is unreliable. When using hubflow to do releases, for example, a job that was supposed to be triggered by changes in both the `master` and `develop` branch only built the `develop` branch even though changes had been pushed to both the `master` and `develop` branches. Since we can programmatically create jobs via the Jenkins Job DSL plugin, it is trivial to create (and modify) jobs for both the `master` and `develop` branches (and other branches as needed - release branches for testing, for example).

A Jenkins job that builds and deploys updated software from a Git repository uses Ansible to do the deployment.

Note: We are currently using flat files to define the inventory for a deployment (a "host" file), although we hope to move to using dynamic inventories. Either way, we need to define the hosts in a system and group them in ways that make deployments easy and scalable. (More on this subject can be found in the ansibleinventory:ansibleinventory documentation.)

Ideally, software in a "develop" branch would be deployed in one or more development and/or test systems, each defined by a single host file (inventory). Software in the "master" branch would be deployed to one or more production or operational systems. One could set up a workflow where release branches were automatically deployed to a release-test system - where the software could be tested before final release. (When the code in the release branch was fully tested and accepted, it would be merged into master according to the hubflow workflow, which would cause it to be automatically deployed to production/operational systems).

Figure *How software in repositories flows to machines in inventories* illustrates this. At the current time, however, we essentially only have one "system" - a "development" system that has grown ad hoc and was not created from scratch using our proposed workflows. The figure shows how we have develop branches of (some) repos also installed in what we've named "prisem", "project", and "infrastructure" inventories. Ideally we would want to consolidate machines under the "development" inventory if we truly wish to install "develop" branch software automatically on all these machines. This would make defining jobs in the Jenkins DSL simpler as well. See the ansibleinventory:ansibleinventory documentation for a description of our current inventory.

We define "groups" for machines in inventories. The groups are used by Ansible to determine whether or not plays should be run on machines in an inventory. The following figure illustrates this. Machines can be found in more than one group. The group "all" contains all machines in the inventory. A playbook that specifies a host of "all" will run on all machines in the inventory (unless further limited by other means, such as flags passed to the `ansible-playbook` command or conditional expressions in a role task). The `dims-ci-utils` code, for example, is to be installed on all machines in the inventory. However, the role that deploys `dims-ci-utils` restricts a couple tasks to specific groups of machines. One of those groups is the "git" group.

7.3 Continuous deployment of documentation

For our documentation, we currently deploy all docs from all repository branches to a single VM to make retrieval efficient and to aid in development of the documentation. Ansible is not used for deployment. We simply use `rsync` over SSH to deploy the docs.

The following figure shows the flows involved in documentation deployment.

The workflow runs something like this:

1. Push to remote repository runs a post-receive hook.
2. Post-receive hook calls the parameterized Jenkins job dims-docs-deploy if either a branch is deleted or if a branch is updated in a repo that contains documentation. The job is called twice - once to build html and once to build PDF.

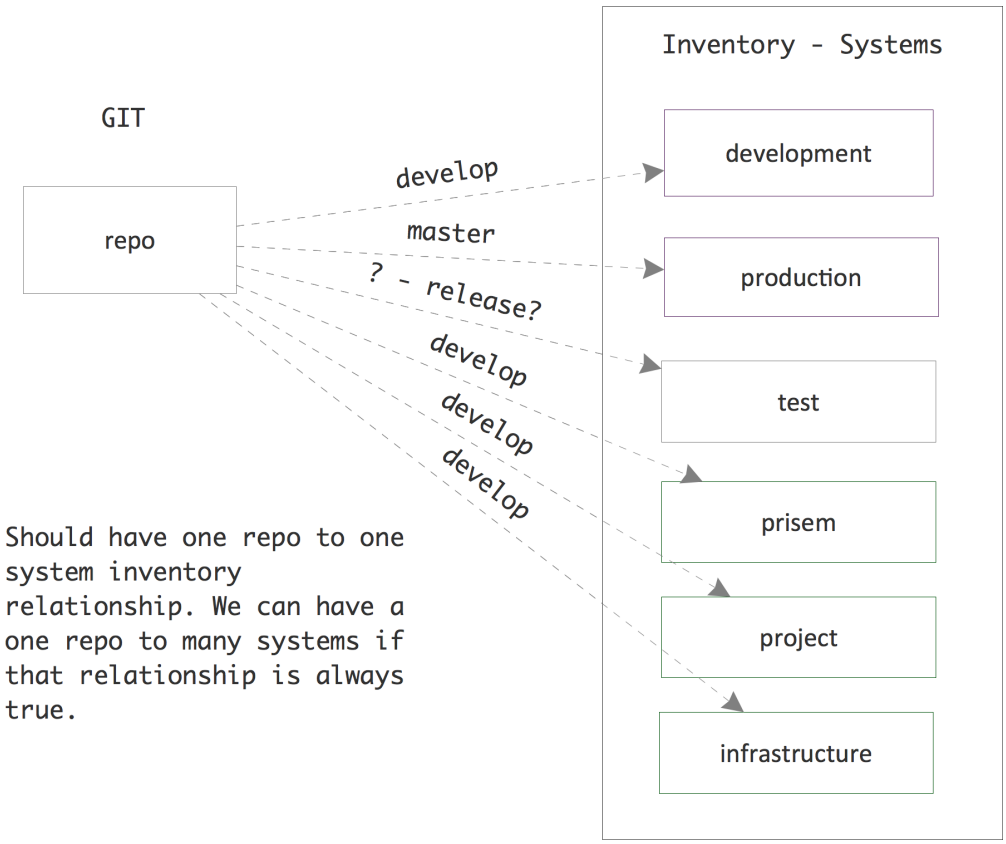


Fig. 7.1: How software in repositories flows to machines in inventories

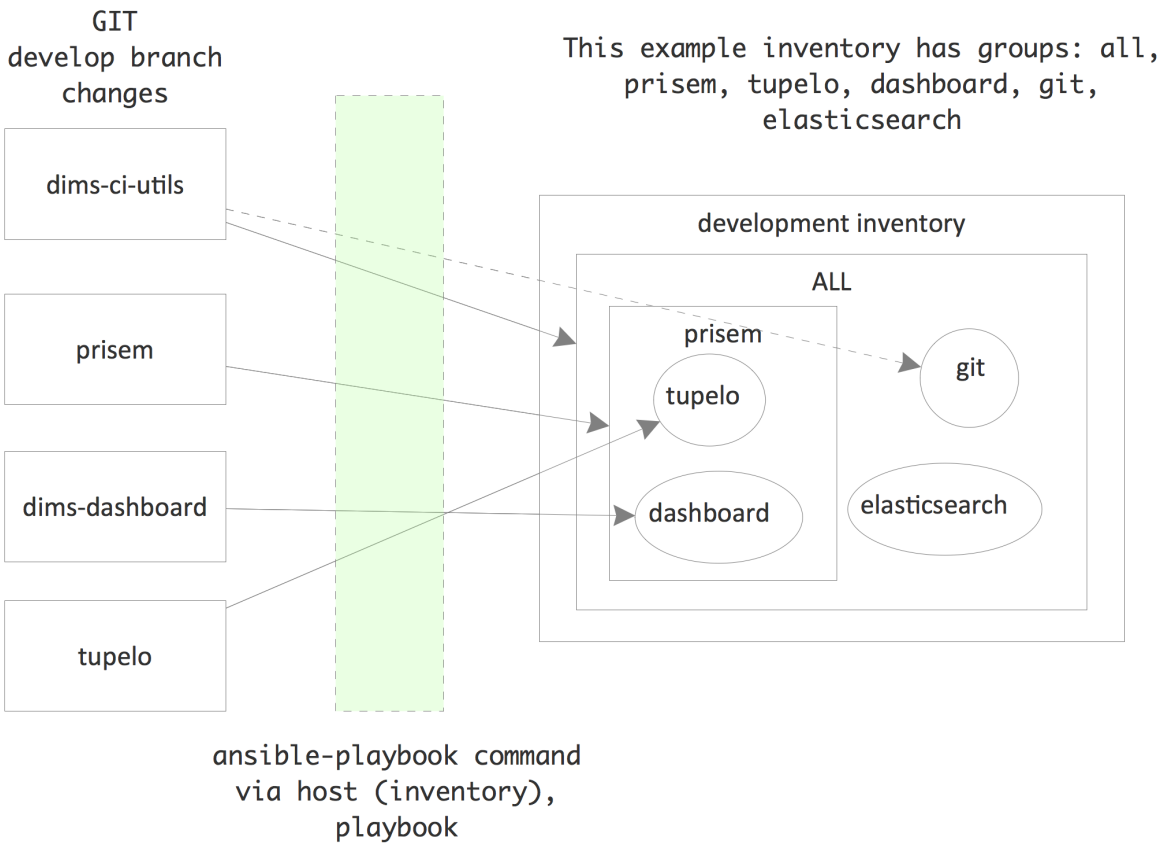


Fig. 7.2: Machines belong to different groups in an inventory

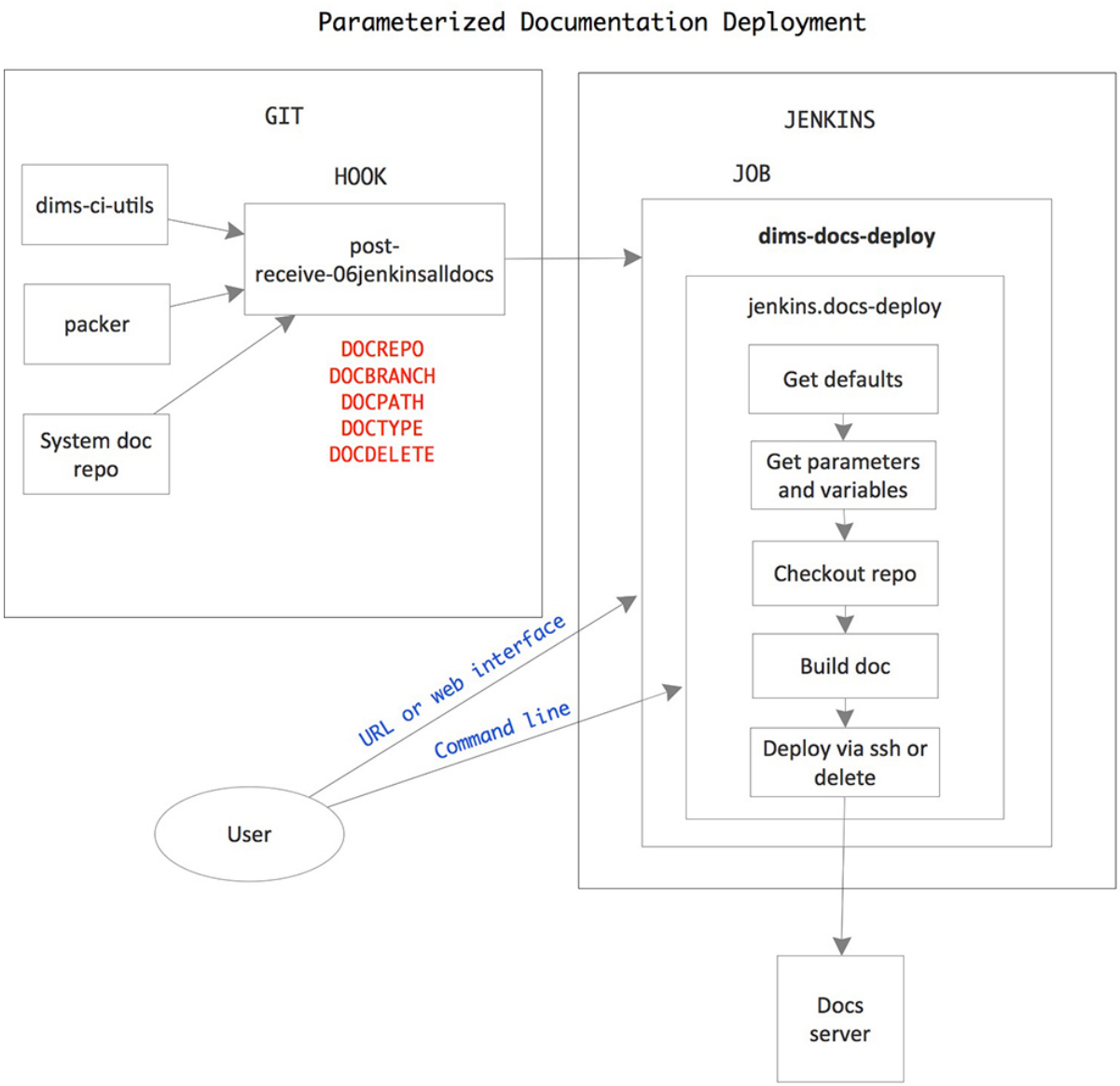


Fig. 7.3: Diagram of documentation deployment flow.

3. Jenkins job dims-deploy-docs runs the script jenkins.docs-deploy
4. Script jenkins.docs-deploy clones and checks out the documentation, builds the documentation, and rsyncs the documentation to the target server.

Documentation is deployed on the target documentation server with the following directory structure:

```
/opt/dims/docs/$BRANCH/html/$REPONAME
/opt/dims/docs/$BRANCH/pdf/$REPONAME
```

Note: `$BRANCH` only includes the last part of a branch name with the `/` delimiter. Therefore, since we use the hubflow branching model, branch `feature/dims-313` is deployed to `/opt/dims/docs/dims-313/html/$REPONAME` and `/opt/dims/docs/dims-313/pdf/$REPONAME`

To view the documentation, you go to https://\protect\T1\textdollarHOST:\protect\T1\textdollarPORT/docs/\protect\T1\textdollarBRANCH/\protect\T1\textdollarTYPE/\protect\T1\textdollarREPONAME or go to https://\protect\T1\textdollarHOST:\protect\T1\textdollarPORT/docs/ and browse the directory tree.

Currently the Jenkins job defaults to deploying the documentation on https://app.devops.develop:8443/docs

The following paragraphs describe this workflow in more detail.

7.3.1 Post-receive hook

The post-receive hook, `post-receive-jenkins06alldocs`, calls a parameterized Jenkins job, `dims-docs-deploy`, when the repository receives a push. The hook code follows:

The hook determines if the repo contains documentation based upon the existence of the file `$REPO/docs/source/conf.py`. This determines the value of `DOCPATH`, which is the path in the repository to the `Makefile` that will build the docs.

Attention: *All* DIMS source repositories *must* have their documentation in a subdirectory named `docs/` in order to simplify the logic of finding and processing Sphinx documentation.

Once the `DOCPATH` is determined, two `curl` commands are sent to Jenkins server to call the job `dims-docs-deploy` - once for HTML, and once for PDF.

The hook source is located in `dims-ci-utils/git/` and is deployed by the `dims-ci-utils-deploy-$BRANCH` jobs.

Note: Currently, we do not have an automated way to add the symlink to the appropriate repos. The command to do that is:

```
$ ln -s /opt/git/bin/post-receive-06jenkinsalldocs /opt/git/${REPO}.git/hooks/post-
↪receive-06jenkinsalldocs
```

7.3.2 Jenkins parameterized job dims-docs-deploy

The parameterized job dims-docs-deploy accepts the following parameters, with the defaults shown. All parameters are string parameters.

NAME	Default Value	Description
REPO		Repository to build
BRANCH		Branch to build
DOCPATH	.	Path to Makefile
DOCTYPE	html	Type of doc to build, html or pdf
DOCDELETE	false	True to delete docs for this branch
DOCHOST	app.devops.develop	Host to receive the docs
DOCDEST	/opt/dims/docs	Root path on host to receive the docs
DOCURL	http://app.devops.develop:8443/docs	URL of docs index

Defaults are given to make it easier to run the job via curl or via the Jenkins web interface - you don't need to include all of the parameters unless they are different than the defaults shown. The post-receive hooks sends the parameters `REPO`, `BRANCH`, `DOCPATH`, `DOCTYPE`, and `DOCDELETE` when it calls the job.

The `dims-docs-deploy` job is created via Jenkins DSL, so it is easy to modify if needed. The Jenkins DSL is located in the file `jenkins/DSL/jenkins-dsl.groovy`, in the `dims-ci-utils` repo. It is automatically run by the Jenkins seed job `dims-seed-job` whenever a change is pushed to the `jenkins/DSL` directory. In this way, the jobs are always up-to-date.

The portion of `jenkins-dsl.groovy` that builds the parameterized documentation job is shown below:

```
// Parameterized job to build and deploy DIMS documentation
job {
    name 'dims-docs-deploy'
    description ('Job to build and deploy DIMS documenation')
    logRotator(-1, 15, -1, -5)
    parameters {
        stringParam('REPO', '', 'Repository to build')
        stringParam('BRANCH', '', 'Branch of the repo to use')
        stringParam('DOCPATH', '.', 'Path to the doc Makefile from repo root')
        stringParam('DOCTYPE', 'html', 'Type of document to build - html or pdf')
        stringParam('DOCDELETE', 'false', 'True if the documentation is to be deleted
↪')
        stringParam('DOCHOST', docHost, 'Host to receive the docs')
        stringParam('DOCDEST', docDest, 'Root destination on host to deploy the docs')
        stringParam('DOCURL', docUrl, 'URL to documentation root directory')
    }
    wrappers {
        preBuildCleanup()
    }
    // This job runs a script
    steps {
        shell ( "jenkins.dims-docs-deploy")
    }
    publishers {
      downstreamParameterized postNotify
    }
}
```

The post-receive hook calls `dims-deploy-docs` via `curl`. You can also do this manually. For example:

```
$ curl --data-urlencode "REPO=${REPONAME}" --data-urlencode "BRANCH=${BRANCH}" --data-
↪urlencode "DOCPATH=${DOCPATH}" --data-urlencode "DOCTYPE=${DOCTYPE}" $JENKINSURL/
↪job/$JOB/buildWithParameters
```

where you have defined the variables shown and `JOB="dims-docs-deploy"` and `JENKINSURL="http://jenkins.devops.develop"`

You can also run the job via the Jenkins UI. Go to http://jenkins.devops.develop/view/Current/job/dims-docs-deploy/

and click the `Build with Parameters` link on the left.

7.3.3 Deployment script jenkins.dims-docs-deploy

As you can see in the previous section, the build step of the `dims-docs-deploy` job calls the `jenkins.dims-docs-deploy` script. The script has access to the job's parameters as environment variables, so they don't need to be passed explicitly when the script is called from the Jenkins job. The script, `jenkins.dims-docs-deploy`, along with other scripts used to build and deploy software by Jenkins, has its source located in `dims-ci-utils/jenkins/job-scripts`. It is deployed on Jenkins in the `/opt/dims/bin` directory.

The `jenkins.dims-docs-deploy` script follows the pattern used by other deploy job scripts:

1. Get default variables
2. Get parameters and massage as needed
3. Checkout the docs repo and branch as specified by parameters
4. Build the docs
5. Deploy the docs

Since we are deploying all documentation to one server irrespective of branch, we do not use the Ansible infrastructure for final deployment. Instead we simply use ssh to make the modifications on the target machine as necessary. A variable, `REMOTEUSER`, is used for the user making the SSH calls. On Jenkins, this user is `ansible`. If you are running the script manually (while testing, for example), you can provide a different user by calling the script with `REMOTEUSER`, as in:

```
$ REPO=dims-sr BRANCH=develop DOCPATH=. DOCTYPE=html REMOTEUSER=$USER jenkins.dims-
↪docs-deploy
```

Of course, `$USER` must be a DIMS user on the target machine (one of the default users installed by Ansible when a DIMS machine is provisioned) and have the appropriate private key.

For your reference, the `jenkins.dims-docs-deploy` source follows:

CHAPTER 8

Deployment and Configuration

8.1 Deployment and Configuration

8.2 Type of Systems

This section will outline the general types of DIMS systems we will deploy. For example:

1. Developer boxes
2. Integration environment
3. Dedicated test environment
4. Production environment

These names were borrowed from the Hubflow documentation at http://datasift.github.io/gitflow/Versioning.html

8.2.1 Developer boxes

This refers to one of two types of systems:

1. Virtual machines (or a system of virtual machines), managed by Vagrant, that a developer runs on his developer workstation to develop, debug and test code.
2. A bare-metal system, such as a laptop, used for development, field testing, demonstrations, etc.

8.2.2 Integration environment

CHAPTER 9

Programming Conventions

This chapter includes examples of programming practices, pointers to code that can serve as a starting point for new programs, and other references to help DIMS developers (and their products) be consistent, maximally functional, robust, and professional.

9.1 Use of `Makefile` helpers

The Unix make utility is used heavily in DIMS development as a means of scripting command sequences in a way that simplifies tasks and makes them more repeatable and consistent. Rather than producing a `README` file that lists generic examples of long list of command lines that must be altered slightly for a specific purpose (e.g., naming things uniquely), a `Makefile` may be used that has a single target that identified by a meaningful word like "target" that can be invoked with one easy to remember command:

```
$ make target
```

A DIMS `Makefile` helper library is available (and encouraged to be used) to consolidate frequently needed variables, macros, etc., into a single file that can be included at the top of every DIMS `Makefile`. One of the things this library includes is a mechanism to embed self-documenting text that can be output when you type `make help`. Here is an example from the `dims-ci-utils/dims/` directory:

```
[dittrich@27b dims (develop)]$ make help
/Users/dittrich/dims/git/dims-ci-utils/dims
[Using Makefile.dims.global v1.6.22 rev 1.4-295-g38b22c8]
-------------------------------------------------------------------------------
Usage: make [something]

Where "something" is one of the targets listed in the sections below.

 ------------------------------------------------------------------------------
                 Targets from Makefile.dims.global

 help - Show this help information (usually the default rule)
 dimsdefaults - show default variables included from Makefile.dims.global
```

```
version - show the Git revision for this repo
envcheck - perform checks of requirements for DIMS development
------------------------------------------------------------------------
                  Targets from Makefile

bootstrap - install foundation items (i.e., test_functions.sh,
            test.dims-ci-utils.envcheck, and test.dims-ci-utils.installcheck)
install - install programs/files
manifest - create expected results (a manifest, for lack of
           a better term) for installcheck to use.
           (See also: README.installcheck)
uninstall - remove programs/files
installdirs - create required directories
installcheck - test to see if everything installed properly
clean - Remove manifest.txt file.
test - run test.dims-ci-utils.envcheck to test test_functions.sh
------------------------------------------------------------------------
```

The first few lines of `$GIT/dims-ci-utils/dims/Makefile` are shown here, with the `include` statement and help text for the target `bootstrap`:

```
# Makefile for DIMS test scripts and other common
# continuous integration aids.

# The following include line is special for
# this one Makefile, since it is used to manage
# Makefile.dims.global.  Other Makefile invocations
# should use:
# include $(DIMS)/etc/Makefile.dims.global
include Makefile.dims.global

#HELP bootstrap - install foundation items (i.e., test_functions.sh,
#HELP             test.dims-ci-utils.envcheck, and test.dims-ci-utils.installcheck)
.PHONY: bootstrap
bootstrap: installdirs \
    $(DIMSETC)/Makefile.dims.global \
    $(DIMSBIN)/test_functions.sh \
    $(DIMSBIN)/test.dims-ci-utils.envcheck \
    $(DIMSBIN)/test.dims-ci-utils.installcheck

...
```

9.2 Variable Naming Conventions

Because the DIMS project uses several programming languages, and several open source tools that may have their own naming policies, variable naming is sometimes quiet hard to normalize into a single form. Programmers also came to the project with their own preferences and past practices, combined with a desire to make their own mark and exercise their own creativity. The result is a mixture of naming conventions and variable name choices that are not entirely consistent.

On top of this, the goal of *no hard coded values* means that variables are often used to create other variables in order to break down something as complicated as a Uniform Resource Locator (URL) that has protocol, host name or address, port number, directory path, and file name. In order to refer to (a) the complete URL, (b) just the file name, and (c) the specific version number, requires that many (but not *too many*) variables be defined.

Some of these variables will also be specific to the program being used (e.g., "consul" for the Consul distributed key/value store program), and some are shared with the operating system and/or hardware architecture (e.g., "ubuntu" and "amd64"). It is desireable to not have multiple variables that hold the same value, but sometimes necessary to have two variables that have slightly different (but related) values (e.g., `ansible_architecture` may hold `x86_64`, but a program like Consul may use `amd64` as the architecture identifier. The following is the full target URL for the Consul binary:

```
https://releases.hashicorp.com/consul/0.6.4/consul_0.6.4_linux_amd64.zip
```

This can be broken down into the following variables:

Variable	Example Value(s)
`consul_version`	`0.6.4`
`consul_os`	`linux` (for Ubuntu, Debian), `darwin` (for Mac OS X)
`consul_arch`	`amd64` (equivalent to `ansible_architecture` value `x86_64`)
`consul_artifact`	`consul_{{ consul_version }}_{{ consul_os }}_{{ consul_arch }}.zip`
`consul_dist_url`	`https://releases.hashicorp.com/consul/{{ consul_version }}/`

The templated version of an Ansible play that downloads the distribution artifact for Consul would thus look like:

```
- name: Make Consul distribution artifact present
  get_url:
    url={{ consul_dist_url }}/{{ consul_artifact }}
    dest={{ dims_deploy }}/{{ role_name }}
    sha256sum={{ consul_sha256sum }}
  sudo: yes
  tags: [ consul ]
```

Using variable names that begin with a consistent string identifying the role or program that they correspond with also helps in two ways.

First, it groups the names (when sorted) and differentiates variables with the same function (e.g., identifying a version number) between Ansible roles. Since Ansible lumps all variables into the same name space, and may process roles in unpredicatable order, there is a chance that a variable with the same exact name in two or more roles will *not* have the value that you think it will have when the play is actually performed. Names must be either prefixed with a string that differentiates them (e.g., `consul_version` vs. `docker_version`), or a more complex nested data structure must be used. Since the latter is more complex, harder to read and harder for those who are new to Ansible to remember how to use, prefixing and separation by way of underscore characters is the preferable naming scheme.

Secondly, this naming style matches that already used by other Ansible facts, which makes the Ansible debugging output a little more consistent and easier to read, again primarily because of the lexical grouping that results.

```
[dimsenv] dittrich@dimsdemo1:~/dims/git/dims-devguide/docs (develop*) $ ls /bin
bash                    dmesg           loginctl          ntfsls                    stty
bunzip2                 dnsdomainname   lowntfs-3g        ntfsmftalloc              su
busybox                 domainname      ls                ntfsmove                  sync
bzcat                   dumpkeys        lsblk             ntfstruncate              tailf
bzcmp                   echo            lsmod             ntfswipe                  tar
bzdiff                  ed              mkdir             open                      ␣
↪tempfile
bzegrep                 egrep           mknod             openvt                    touch
bzexe                   false           mktemp            pidof                     true
bzfgrep                 fgconsole       more              ping                      ␣
↪udevadm
bzgrep                  fgrep           mount             ping6                     ␣
↪ulockmgr_server
bzip2                   findmnt         mountpoint        plymouth                  umount
```

```
bzip2recover          fuser          mt                plymouth-upstart-bridge  uname
bzless                fusermount     mt-gnu            ps                       ␣
↪uncompress
bzmore                getfacl        mv                pwd                      ␣
↪unicode_start
cat                   grep           nano              rbash                     vdir
cgroups-mount         gunzip         nc                readlink                 ␣
↪vmmouse_detect
cgroups-umount        gzexe          nc.openbsd        red                       which
chacl                 gzip           nc.traditional    rm                       ␣
↪whiptail
chgrp                 hostname       netcat            rmdir                    ␣
↪ypdomainname
chmod                 ip             netstat           rnano                     zcat
chown                 kbd_mode       nisdomainname     running-in-container      zcmp
chvt                  keyctl         ntfs-3g           run-parts                 zdiff
cp                    kill           ntfs-3g.probe     rzsh                      zegrep
cpio                  kmod           ntfs-3g.secaudit  sed                       zfgrep
dash                  less           ntfs-3g.usermap   setfacl                   zforce
date                  lessecho       ntfscat           setfont                   zgrep
dbus-cleanup-sockets  lessfile       ntfsck            setupcon                  zless
dbus-daemon           lesskey        ntfscluster       sh                        zmore
dbus-uuidgen          lesspipe       ntfscmp           sh.distrib                znew
dd                    ln             ntfsdump_logfile  sleep                     zsh
df                    loadkeys       ntfsfix           ss                        zsh5
dir                   login          ntfsinfo          static-sh
```

You can see that the Zip and Bzip2 related programs all start with `z` and `bz` respectively. But just because a programs starts with "z" does **not** mean that it is a Zip related program (e.g., `zsh` is a shell, completely unrelated to Zip.) You can use tab completion in Bash or other shells to find all commands that start with "z" and guess which ones are part of the Zip program set, but you can't use a `help` sub-command to get the actual list. (See `man zip` to see how the documentation lists other options, though not exactly all of them in the main `zip` man page.)

Another way of organizing programs (or more precisely, sub-programs) is to use a plugin-style mechanism for organization. Git is an example of a multi-component program suite that does this. It has many sub-programs, each starting with `git-` in the name. The Git Hubflow tools integrate themselves into Git because their names start with `git-`:

```
[dimsenv] dittrich@dimsdemo1:~ () $ (cd $(dirname $(which git)); ls git*)
git              git-hf          git-hf-pull     git-hf-update     git-shell
git-big-picture  git-hf-feature  git-hf-push     git-hf-upgrade    git-upload-archive
git-crypt        git-hf-hotfix   git-hf-release  git-hf-version    git-upload-pack
git-cvsserver    git-hf-init     git-hf-support  git-receive-pack
```

You can use the tab command completion functionality of Bash to help show you the grouped commands, or use the `git help` sub-command, which not only lists them, but also provides even more hints on how to get help and usage information. Here is the list of first-level sub-commands related to Git Hubflow:

```
[dimsenv] dittrich@dimsdemo1:~ () $ git hf help
usage: git hf <subcommand>

Available subcommands are:
   init      Initialize a new git repo with support for the branching model.
   feature   Manage your feature branches.
   release   Manage your release branches.
   hotfix    Manage your hotfix branches.
   push      Push the changes from your current branch (plus any new tags) back␣
↪upstream.
```

```
   pull      Pull upstream changes down into your master, develop, and current␣
↪branches.
   update    Pull upstream changes down into your master and develop branches.
   version   Shows version information.

Try 'git hf <subcommand> help' for details.
```

This downside to this mechanism is that it requires that the top- or mid-level command (in this case, `git`) have complex logic and funcationality to support this plugin model.

An easier way to acheive the same effect of allowing tab command completion to group commands. The DIMS project uses a convention of multi-component names separated by periods, kind of like DNS Domain names are constructed. The first component is supposed to be the high-level grouping, followed by sub-groups, and lastly sub-functions.

Note: These scripts are installed into either `/opt/dims/bin`, or Python virtual environments, as necessary to support development and testing without breaking the basic system functionality by accidentally installing a script with a syntax error that fails out when called from another system script. The tab command completion feature of Bash will find them, regardless of which directory from the `$PATH` they are installed in.

The main grouping used within DIMS is `dims`, and another related to Packer/Vagrant virtual machines and test/validation functionality is `test`. Here is what you get when you use tab command completion with these two prefixes:

```
[dimsenv] dittrich@dimsdemo1:~ () $ test.<TAB><TAB>
test.ansible.yaml                test.runner.orig
test.dims-ci-utils.envcheck      test.supervisor
test.dims-ci-utils.installcheck  test.vagrant.ansible-current
test.md5.output                  test.vagrant.factory
test.packer.factory              test.vagrant.list
test.packer.list                 test.vagrant.listvms
test.runner                      test.yaml.validate
[dimsenv] dittrich@dimsdemo1:~ () $ dims.<TAB><TAB>
dims.ansible-playbook          dims.elasticsearch.service    dims.localcluster.start
dims.ansible-playbook.orig     dims.git.repoversion          dims.localcluster.status
dims.boot2docker               dims.git.syncrepos            dims.localcluster.stop
dims.buildvirtualenv           dims.help                     dims.makedocset
dims.bumpversion               dims.install.createusb        dims.nas.mount
dims.cleanup                   dims.install.dimscommands     dims.nas.umount
dims.clusterconfig.list        dims.jj2                      dims.remote.
↪setupworkstation
dims.clusterconfig.local       dims.localcluster.create      dims.shutdown
dims.clusterconfig.nas         dims.localcluster.destroy     dims.sphinx-autobuild
dims.cluster.runscript         dims.localcluster.runscript   dims.swapcapslockctrl
```

This helps show or remind you what scripts are available and how they may be related to each other.

Note: While this naming scheme is the general policy, it is not universally true for all DIMS programs. Some legacy programs integrated into DIMS, scripts that were written for specific applications (like `list` as one of the Git-shell commands), or programs written by DIMS team members as prototypes that were not categorized at the time they were written, have non-standard names.

Also, the organizational taxonomy for naming scripts has grown organically over time, since this kind of organization is sometimes not readily obvious at the time someone needs to write a new script. Efforts to refactor these scripts have taken a lower priority to simply implenting those that are needed to complete the project.

An effort was begun to bring all of these scripts underneath a main CLI the way that Git works, in order to better integrate the use of variables that are used for differentiating between deployments. This high-level CLI is `dimscli` (see *Developing modules for the DIMS CLI app (dimscli)*). This is still a work-in-progress.

Attention: Here are some questions to ask when confronted with variable or program naming:

- I want to put some value into a variable. Is there already a variable that has been defined to hold this same value? (If so, is that the appropriate variable to use? Do we need another variable with the same value, and how will the next person know which one to use?)
- Is there already a dictionary or other more complex structure that holds related variables? (If so, is it appropriate to add another field or mapping to that structure? Is this data structure the place where someone else will look to find this variable?)
- What naming convention is used elsewhere? Is `camelCaseStyle`, `ALLCAPS`, or `all_lower_case_with_underscores` used? Think about how you or someone else will search the source files to find this variable, and ask yourself if it will be necessary to use a complicated regular expressions or simple one to find it?
- Is the new name a translation of an existing name, and if so, why is it being translated instead of used with the exact same spelling? For example, if `Makefile` has something like `VARIABLE_NAME={{ varName }}` in it, why was `varName` translated to `VARIABLE_NAME`? (Look at the last bullet point to see why this matters.)
- How will the script name group with other scripts when sorted by the `ls` program? Will the script be grouped *with* other simple scripts that are commonly related, or will it end up being mixed up?

  A really good example for helping understanding this point is naming files with dates in the file names. The strings using the `MM_DD_YYYY` style (or the European equivalent `DD_MM_YYYY` style), or spelled out `Jan_01_2015`, `Feb_15_2016`, etc., will **not sort properly**. If instead you use a naming structure that puts the things that change least frequently to the left, and the things that change most frequently to the right (e.g., `2015_01_01`, `2016_02_15`) not only will the names **always** sort properly, but they also group better by year and month!

There are sometimes good reasons to deviate from convention, to translate from one variable name to another, or to create a new variable that holds the same value as another variable. If you are able to explicitly describe why a specific new variable name was chosen, put in a comment to explain the deviation. If you are in doubt, or can't really give a good reason for doing something that may have negative consequences later on for someone else, log a ticket or bring up the issue at a scrum meeting to discuss the issue and get resolution.

CHAPTER 10

Ops-trust-db VM Creation

These are instructions for running Ansible playbooks to install a clone of the current ops-trust database. You first need a VM to install it on. The directions for creating a VM are located at dimspacker:vmquickstart.

If you have already created a server .box file, then you can skip the first part of the instructions. You can either go to dimspacker:vmquickstartinstantiate and create a new server, or if you've already created one and want to re-use it, you can destroy it (via vagrant destroy) and bring it back up again fresh via vagrant up.

Section dimspacker:vmquickstartrunansible talks about running playbooks against the VM.

For using the VM as a postgresql server, you don't need to install the python virtual environment. Just run the following to prepare the VM:

```
$ ./run_playbook base-os.yml
$ ./run_playbook common.yml
$ ./run_playbook provision-dims-users.yml
$ ./run_playbook dims-ci-utils-deploy.yml -vv -e artifact_branch=develop
```

Then, run the ops-trust-testdb-configure.yml playbook:

```
$ ./run_playbook -g ops-trust-testdb-servers ops-trust-testdb-configure.yml
-vv
```

This will install postgresql, create the database, and populate it (if it doesn't already exist). To re-initialize the database (like if you've been working with it and want to start over) you need to specify initialize_postgres_dims as true:

```
$ ./run_playbook -g ops-trust-testdb-servers ops-trust-testdb-configure.yml
-vv -e initialize_postgres_dims=true
```

Note: The following is an example of the data that exists in the ops-trust table.

The data for ops-trust has been redacted and truncated. Sensitive information has been replaced with * and only the first record of each table has been shown. In addition some tables are not shown.

```
[('openid_source_cache',), ('second_factors',), ('web_sessions',), ('trustgroup',), (
↪'attestations',), ('member_trustgroup',), ('language_skill',), ('audit_history',), (
↪'languages',), ('member_mailinglist',), ('mailinglist',), ('member',), ('member_
↪email',), ('member_detail_types',), ('member_details',), ('member_language_skill',),
↪ ('member_state',), ('member_vouch',), ('message_catalog',), ('message_types',), (
↪'openid_associations',), ('second_factor_types',)]

Executing SELECT on openid_source_cache
Column names: ['src_ip', 'last_try', 'attempt_count']
Number of records: 7
First record: ('*.*.*.*', datetime.datetime(yyyy, mm, dd, hh, mm, ss, 159691), *)

Executing SELECT on second_factors
Column names: ['uuid', 'member', 'type', 'entered', 'active', 'counter', 'key', 'descr
↪']
Number of records: 0

Executing SELECT on web_sessions
Column names: ['id', 'a_session']
Number of records: 59
First record: ('******', "$D = {'form_id' => '******','test' => '******','uuid' =>
↪'******','sysadmin' => ******,'_SESSION_ETIME' => ******,'_SESSION_ID' => '******',
↪'admin' => ******,'ntg' => ******,'member' => '******','change_pw' => 0,'can_see' =>
↪ 1,'~logged-in' => 't','trustgroup' => 'dims','_SESSION_REMOTE_ADDR' => '******.
↪******.******.******','_SESSION_CTIME' => ******,'_catalyst_session' => {},'_
↪SESSION_ATIME' => ******};;$D")

Executing SELECT on trustgroup
Column names: ['ident', 'descr', 'shortname', 'min_invouch', 'target_invouch', 'pgp_
↪required', 'please_vouch', 'vouch_adminonly', 'nom_enabled', 'min_outvouch', 'max_
↪inactivity', 'can_time_out', 'max_vouchdays', 'idle_guard', 'has_wiki']
Number of records: 2
First record: ('dims', 'DIMS Trust Group', 'dims', 1, 1, False, False, False, True, 0,
↪ datetime.timedelta(30), True, 15, datetime.timedelta(25), True)

Executing SELECT on attestations
Column names: ['ident', 'descr', 'trustgroup']
Number of records: 6
First record: ('met', 'I have met them in person more than once.', 'main')

Executing SELECT on member_trustgroup
Column names: ['member', 'trustgroup', 'admin', 'entered', 'activity', 'state', 'email
↪']
Number of records: 13
First record: ('******', 'main', False, datetime.datetime(yyyy, mm, dd, hh, mm, ss,␣
↪275385), datetime.datetime(yyyy, mm, dd, hh, mm, ss, 944053), 'active', '******@uw.
↪edu')

Executing SELECT on language_skill
Column names: ['skill', 'seq']
Number of records: 4
First record: ('native', 4)

Executing SELECT on audit_history
Column names: ['member', 'what', 'entered']
Number of records: 259
First record: ('******', "SQL: UPDATE member_trustgroup SET state = 'approved' WHERE␣
↪ROW(member, trustgroup, state) = ROW('******', 'dims', 'vetted'); ", datetime.
↪datetime(yyyy, mm, dd, hh, mm, ss, 135660))
```

```
Executing SELECT on languages
Column names: ['name', 'iso_639_1']
Number of records: 184
First record: ('Afar', 'aa')

Executing SELECT on member_mailinglist
Column names: ['member', 'lhs', 'trustgroup', 'virtual']
Number of records: 70
First record: ('******', 'general', 'dims', False)

Executing SELECT on mailinglist
Column names: ['lhs', 'trustgroup', 'descr', 'members_only', 'can_add_self',
 ↪'automatic', 'always_crypt', 'virtual', 'activity', 'email_footer', 'pubkey', 'key_
 ↪update_at', 'seckey']
Number of records: 12
First record: ('vetting', 'dims', 'Vetting and Vouching', True, True, True, False,␣
 ↪False, None, None, '-----BEGIN PGP PUBLIC KEY BLOCK-----\n******\n-----END PGP␣
 ↪PUBLIC KEY BLOCK-----\n', datetime.datetime(yyyy, mm, dd, hh, mm, ss, 252368), '----
 ↪-BEGIN PGP PRIVATE KEY BLOCK-----\n******\n-----END PGP PRIVATE KEY BLOCK-----\n')

Executing SELECT on member
Column names: ['ident', 'descr', 'affiliation', 'password', 'passwd_chat', 'tz_info',
 ↪'im_info', 'tel_info', 'sms_info', 'post_info', 'bio_info', 'airport', 'no_email',
 ↪'hide_email', 'furlough', 'change_pw', 'entered', 'activity', 'uuid', 'sysadmin',
 ↪'login_attempts', 'login_try_begin', 'image']
Number of records: 8
First record: ('******', '****** ******(Full name)', '@uw.edu', '******', None, '', '
 ↪', '', None, '', '******', None, False, False, False, False, datetime.datetime(yyyy,
 ↪ mm, dd, hh, mm, ss, 488278), datetime.datetime(yyyy, mm, dd, hh, mm, ss, 288486),
 ↪'0878ca30-c16a-435d-8991-39dd366fa4d4', False, 0, None, None)

Executing SELECT on member_email
Column names: ['member', 'email', 'pgpkey_id', 'verified', 'pgpkey_expire', 'keyring',
 ↪ 'keyring_update_at']
Number of records: 8
First record: ('*****', '*****@uw.edu', '8E01820D', True, datetime.datetime(yyyy, mm,␣
 ↪dd, hh, mm, ss, 0), '-----BEGIN PGP PUBLIC KEY BLOCK-----\******\n-----END PGP␣
 ↪PUBLIC KEY BLOCK-----\n', datetime.datetime(yyyy, mm, dd, hh, mm, ss, 554641))

Executing SELECT on member_detail_types
Column names: ['type', 'display_name']
Number of records: 1
First record: ('callsign', 'Amateur radio callsign')

Executing SELECT on member_details
Column names: ['member', 'type', 'entered', 'value']
Number of records: 1
First record: ('*****', 'callsign', datetime.datetime(yyyy, mm, dd, hh, mm, ss,␣
 ↪76232), 'none')

Executing SELECT on member_language_skill
Column names: ['member', 'language', 'skill', 'entered']
Number of records: 3
First record: ('*****', 'en', 'native', datetime.datetime(yyyy, mm, dd, hh, mm, ss,␣
 ↪914698))

Executing SELECT on member_state
```

```
Column names: ['ident', 'can_login', 'can_see', 'can_send', 'can_recv', 'blocked',
↪'hidden']
Number of records: 9
First record: ('nominated', False, False, False, False, False, False)

Executing SELECT on member_vouch
Column names: ['vouchor', 'vouchee', 'trustgroup', 'comment', 'entered', 'positive']
Number of records: 19
First record: ('*****', '*****', 'dims', '', datetime.datetime(yyyy, mm, dd, hh, mm,␣
↪ss, 266320), True)

Executing SELECT on message_catalog
Column names: ['trustgroup', 'message_type', 'message_template']
Number of records: 0

Executing SELECT on message_types
Column names: ['ident', 'descr']
Number of records: 3
First record: ('web_global_hello', 'Global public about Ops-t page')

Executing SELECT on openid_associations
Column names: ['uuid', 'assoc_type', 'session_type', 'mac_key', 'timestamp']
Number of records: 0

Executing SELECT on second_factor_types
Column names: ['type', 'descr']
Number of records: 3
First record: ('TOTP', 'Time based One Time Password - TOPT')
```

CHAPTER 11

Developing Bash scripts

The DIMS project uses many scripts (and GNU `make` files) to structure, organize, and serve as an abstraction layer wrapping other applications that require a large number of variables or command line options to function. Because the DIMS project attempts to support multiple independent instances of a set of open source tools on a small-scale distributed system, the ability to generalize operations across a set of a dozen or more servers is important. This is not necessary to scale to one large system with hundreds or thousands of servers, but rather to support a single small group operating multiple replicated instances of the distributed system for *development*, *testing*, *production for group A* and *production for group B*, etc.

Many of the basic scripts used in the DIMS project are written in Bourne Again Shell (BASH, a.k.a. Bash and `bash`). Bash is a relatively simple scripting language for basic things, but quickly can get very complicated when attempting to perform advanced tasks that are required for scalable distributed systems.

Advanced Bash programming requires understanding of some fairly low-level Unix/Linux process and file system concepts, as well as learning how to do run-time debugging of a scripting shell language that can be far more opaque than languages like Python executed within Integrated Development Environments (IDEs) and run-time debuggers. (Bash does have a runtime debugger, but there is a steep learning curve for those diving into advanced Bash scripting.) DIMs programmers are advised to read, understand, and follow (to the best of their ability) the Google Shell Style Guide. Other references on advanced Bash scripting can be found in Section *Bash programming references*.

11.1 Command line processing using Google's `shFlags`

One of the most powerful features of Unix/Linux commands is the ability to inheret default behaviors (e.g., through setting environment variables that are passed from a process to its children and children's children as scripts invoke scripts), and to be able to alter or select options on the command line to change the scripts' behavior.

The DIMS project uses Google's **'shFlags'_** library to standardize command line processing. Programers are required to study the Documentation shFlags 1.2.x. page to learn how **'shFlags'_** works and how to use it.

All scripts should support a common set of basic functions such as the following:

- `--help` provides help on arguments and options supported by the script:

```
$ test.vagrant.list --help
usage: test.vagrant.list [options] [ACTION] FQDN
flags:
  -d,--[no]debug:  enable debug mode (default: false)
  -C,--category:  category identifier (default: 'devops')
  -D,--deployment:  deployment identifier (default: 'develop')
  -g,--group:  inventory group to list (default: 'vagrants')
  -r,--[no]running:  list only running vagrants (default: false)
  -s,--[no]status:  return Vagrant status and exit (default: false)
  -H,--[no]shortnames:  return short hostnames (default: false)
  -u,--[no]usage:  show usage information (default: false)
  -i,--[no]vagrants:  list vagrants from inventory (default: false)
  -m,--[no]vms:  list Virtualbox VMs (default: false)
  -v,--[no]verbose:  be verbose (default: false)
  -V,--[no]version:  print version number and exit (default: false)
  -h,--help:  show this help (default: false)
```

- `--usage` provides more detailed information and examples of how to invoke the script to perform required functions:

```
$ test.vagrant.list --usage
   usage: test.vagrant.factory [options] [ACTION] FQDN
flags:
  -d,--[no]debug:  enable debug mode (default: false)
  -b,--[no]list-boxes:  list Vagrant boxes (default: false)
  -o,--[no]list-ovfs:  list Packer base OVF files (default: false)
  -D,--[no]list-vms:  list Virtualbox VMs (default: false)
  -s,--[no]vagrant-status:  return Vagrant status and exit (default: false)
  -P,--[no]remove-packer-box:  remove Packer box file (default: false)
  -u,--[no]usage:  show usage information (default: false)
  -v,--[no]verbose:  be verbose (default: false)
  -V,--[no]version:  print version number and exit (default: false)
  -h,--help:  show this help (default: false)

where 'ACTION' is one of:
    build         - build a virtual machine NAME for DEPLOYMENT-CATEGORY from
                    locally checked out ansible-playbooks repo
    build-develop - build a virtual machine NAME for DEPLOYMENT-CATEGORY from
                    origin, branch dev, of ansible-playbooks (old default packer␣
↪behavior)
    clean         - clean up directory for virtual machine NAME
    destroy       - destroy virtual machine NAME
    spotless      - completely remove all traces of virtual machine NAME
    get-args      - get args for test.vagrant.ansible-current
    show          - show parameters used to build virtual machine NAME
    usage         - show this help text

... and 'FQDN' is the fully-qualified domain name of a host defined
in the directory /home/dittrich/dims/git/ansible-playbooks/v2/inventory/
↪$DEPLOYMENT/host_vars/

To get the status of a vagrant, use the --vagrant-status option:

  $ test.vagrant.factory --vagrant-status red.devops.local
  Vagrant "red" status: running


To see status of all potential Vagrants, do:
```

```
$ test.vagrant.factory --vagrant-status
Vagrant "blue14" status: not created
Vagrant "blue16" status: not created
Vagrant "core-01" status: running
Vagrant "core-02" status: running
Vagrant "core-03" status: running
Vagrant "green" status: running
Vagrant "red" status: running
Vagrant "yellow" status: not created
```

- `--version` produces the version number, which is typically maintained using `bumpversion` (see Section *Managing Version Numbers* for how `bumpversion` is used.)

```
$ test.vagrant.list --version
test.vagrant.list 1.2.28
```

11.2 Script naming conventions

11.3 Bash programming references

- Basic grammar rules of Bash, BashHackersWiki
- **Commandlinefu.com**
  - Find Duplicate Files (based on size first, then MD5 hash)
  - Recursively remove all empty directories
  - **GitHub google/styleguide ("Style guides for Google-originated open-source projects")**
    - Shell Style Guide
- **Command line option parsing**
  - GitHub kward/shflags (Automatically exported from https://code.google.com/p/shflags)
  - Easy Bash Scripting With Shflags, by Steve Francia, July 8, 2011
  - Using getopts in bash shell script to get long and short command line options, Stackoverflow post
- **Advanced Bash scripting**
  - How "Exit Traps" Can Make Your Bash Scripts Way More Robust And Reliable, by Aaron Maxwell
  - I set variables in a loop that's in a pipeline. Why do they disappear after the loop terminates? Or, why can't I pipe data to read?, BasFAQ/024
  - The Ultimate Bash Array Tutorial with 15 Examples, by Sasikala on June 3, 2010
  - BashGuide/Arrays
- **Debugging Bash scripts**
  - Debug your shell scripts with bashdb, by Ben Martin, November 24, 2008
  - Debugging a script, Bash Hackers Wiki
  - Why does my shell script choke on whitespace or other special characters?, StackExchange post by Gilles, May 24 2014

CHAPTER 12

Developing modules for the DIMS CLI app (`dimscli`)

12.1 Bootstrapping the `dimscli` app for development

1. Clone the repo `python-dimscli` from `git.devops.develop`. This can be done by running `dims.git.syncrepos`:

2. Prepare a new Python virtual environment with all of the DIMS pre-requisite tools necessary for DIMS software development:

```
[dimsenv] dittrich@dimsdemo1:~/dims/git/python-dimscli (develop) $ VENV=dimscli␣
↪dimsenv.install.user
sudo password:

PLAY [Install python virtual environment] ***********************************

...

PLAY RECAP *********************************************************************
localhost                  : ok=30   changed=19   unreachable=0    failed=0
```

   The new `dimscli` virtual environment should show up as an option for `workon`:

```
[dimsenv] dittrich@dimsdemo1:~/dims/git/python-dimscli (develop) $ workon
dimscli
dimsenv
```

3. Invoke the new `dimscli` Python virtual environment.

```
[dimsenv] dittrich@dimsdemo1:~/dims/git/python-dimscli (develop) $ workon dimscli
[+++] Virtual environment 'dimscli' activated [ansible-playbooks v1.2.113]
```

4. Because this is a new Python virtual environment created with the DIMS build tools, it only has those Python packages defined in Ansible playbooks role `python-virtualenv`.

The first time you try to run `dimscli`, or any time that you change any of the pre-requisites used for programming `dimscli` modules, you must use `pip` to update and/or install the required packages. These will eventually be added to the defaults for the `dimsenv` standard virtual environment.

```
[dimscli] dittrich@dimsdemo1:~/dims/git/python-dimscli (develop) $ pip install -U␣
↪-r requirements.txt
Collecting pbr<2.0,>=1.4 (from -r requirements.txt (line 1))
Using cached pbr-1.8.1-py2.py3-none-any.whl
Collecting six>=1.9.0 (from -r requirements.txt (line 2))
Using cached six-1.10.0-py2.py3-none-any.whl
Requirement already up-to-date: Babel>=1.3 in /home/dittrich/dims/envs/dimscli/
↪lib/python2.7/site-packages (from -r requirements.txt (line 3))
Collecting cliff>=1.14.0 (from -r requirements.txt (line 4))
Downloading cliff-1.15.0-py2-none-any.whl
Collecting keystoneauth1>=1.0.0 (from -r requirements.txt (line 5))
Downloading keystoneauth1-1.2.0-py2.py3-none-any.whl (149kB)
100% | extblock extblock extblock extblock extblock extblock extblock extblock
extblock extblock extblock extblock extblock extblock extblock extblock extblock
extblock extblock extblock extblock extblock extblock extblock extblock extblock
extblock extblock extblock extblock extblock extblock| 151kB 2.7MB/s
Collecting os-client-config!=1.6.2,>=1.4.0 (from -r requirements.txt (line 6))
Downloading os_client_config-1.10.1-py2.py3-none-any.whl (42kB)
100% | extblock extblock extblock extblock extblock extblock extblock extblock
extblock extblock extblock extblock extblock extblock extblock extblock extblock
extblock extblock extblock extblock extblock extblock extblock extblock extblock
extblock extblock extblock extblock extblock extblock| 45kB 6.0MB/s

...

Running setup.py bdist_wheel for msgpack-python
Stored in directory: /home/dittrich/.cache/pip/wheels/f3/97/a5/
↪dd6e3b680de10b689464c44bc211239d1fe54bd296ff860897
Running setup.py bdist_wheel for functools32
Stored in directory: /home/dittrich/.cache/pip/wheels/38/c6/c7/
↪ee17acd621120c302e25c2fa8b3a8b235d5d1137c6ab4c9728
Successfully built simplejson warlock msgpack-python functools32
Installing collected packages: msgpack-python, oslo.serialization, python-
↪keystoneclient, simplejson,
python-neutronclient, functools32, jsonschema, jsonpointer, jsonpatch, warlock,␣
↪python-glanceclient,
python-novaclient, python-cinderclient, python-openstackclient

Successfully installed functools32-3.2.3.post2 jsonpatch-1.12 jsonpointer-1.10␣
↪jsonschema-2.5.1 msgpack-python-0.4.6
oslo.serialization-1.11.0 python-cinderclient-1.4.0 python-glanceclient-1.1.0␣
↪python-keystoneclient-1.8.1
python-neutronclient-3.1.0 python-novaclient-2.34.0 python-openstackclient-1.8.0␣
↪simplejson-3.8.1 warlock-1.2.0
PrettyTable-0.7.2 appdirs-1.4.0 cliff-1.15.0 cliff-tablib-1.1 cmd2-0.6.8␣
↪debtcollector-0.10.0 iso8601-0.1.11
keystoneauth1-1.2.0 monotonic-0.4 netaddr-0.7.18 netifaces-0.10.4 os-client-
↪config-1.10.1 oslo.config-2.6.0
oslo.i18n-2.7.0 oslo.utils-2.7.0 oslosphinx-3.3.1 pbr-1.8.1 pyparsing-2.0.5 pytz-
↪2015.7 requests-2.8.1
six-1.10.0 stevedore-1.9.0 tablib-0.10.0 unicodecsv-0.14.1 wrapt-1.10.5
```

5. Once all the pre-requisite packages are installed in the virtual environment, install the `dimscli` app and its modules as well using `python setup.py install` or `pip install -e .` (either will work):

```
[dimscli] dittrich@dimsdemo1:~/dims/git/python-dimscli (develop) $ python setup.
↪py install
running install
[pbr] Writing ChangeLog
[pbr] Generating ChangeLog
[pbr] ChangeLog complete (0.0s)
[pbr] Generating AUTHORS
[pbr] AUTHORS complete (0.0s)
running build
running build_py
creating build
creating build/lib
creating build/lib/dimscli
creating build/lib/dimscli/common

...

byte-compiling /home/dittrich/dims/envs/dimscli/lib/python2.7/site-packages/
↪dimscli/common/timing.py to timing.pyc
byte-compiling /home/dittrich/dims/envs/dimscli/lib/python2.7/site-packages/
↪dimscli/common/context.py to context.pyc
byte-compiling /home/dittrich/dims/envs/dimscli/lib/python2.7/site-packages/
↪dimscli/common/clientmanager.py to clientmanager.pyc
byte-compiling /home/dittrich/dims/envs/dimscli/lib/python2.7/site-packages/
↪dimscli/common/logs.py to logs.pyc
byte-compiling /home/dittrich/dims/envs/dimscli/lib/python2.7/site-packages/
↪dimscli/common/utils.py to utils.pyc
running install_egg_info
Copying python_dimscli.egg-info to /home/dittrich/dims/envs/dimscli/lib/python2.7/
↪site-packages/python_dimscli-0.0.1.dev391-py2.7.egg-info
running install_scripts
Installing dimscli script to /home/dittrich/dims/envs/dimscli/bin
```

6. Run the `dimscli` app like any other program, directly from the command line.

   There are two ways to use `dimscli`.

   - As a single command with command line options like other Linux commands

     ```
     [dimscli] dittrich@dimsdemo1:~/dims/git/python-dimscli (develop) $ dimscli --
     ↪version
     dimscli 0.0.1
     [dimscli] dittrich@dimsdemo1:~/dims/git/python-dimscli (develop) $
     ```

   - As an interactive shell that allows you to run multiple commands in sequence within the same context (i.e., the same state, or runtime settings you invoke while in the shell) by just just the program name and no arguments or options.

     ```
     [dimscli] dittrich@dimsdemo1:~/dims/git/python-dimscli (develop) $ dimscli
     defaults: {u'auth_type': 'password', u'compute_api_version': u'2', 'key':␣
     ↪None, u'database_api_version': u'1.0',
     'api_timeout': None, u'baremetal_api_version': u'1', 'cacert': None, u'image_
     ↪api_use_tasks': False,
     u'floating_ip_source': u'neutron', u'orchestration_api_version': u'1', u
     ↪'interface': None, u'network_api_version':
     u'2.0', u'image_format': u'qcow2', u'object_api_version': u'1', u'image_api_
     ↪version': u'2', 'verify': True,
     u'identity_api_version': u'2.0', u'volume_api_version': u'1', 'cert': None, u
     ↪'secgroup_source': u'neutron',
     ```

```
u'dns_api_version': u'2', u'disable_vendor_agent': {}}
cloud cfg: {'auth_type': 'password', u'compute_api_version': u'2', u
↪'orchestration_api_version': u'1',
u'database_api_version': u'1.0', 'cacert': None, u'network_api_version': u'2.0
↪', u'image_format': u'qcow2',
u'object_api_version': u'1', u'image_api_version': u'2', 'verify': True, u
↪'dns_api_version': u'2',
'verbose_level': '1', 'region_name': '', 'api_timeout': None, u'baremetal_api_
↪version': u'1', 'auth': {},
'default_domain': 'default', u'image_api_use_tasks': False, u'floating_ip_
↪source': u'neutron', 'key': None,
'timing': False, 'deferred_help': False, u'identity_api_version': u'2.0', u
↪'volume_api_version': u'1',
'cert': None, u'secgroup_source': u'neutron', u'interface': None, u'disable_
↪vendor_agent': {}}
compute API version 2, cmd group dims.compute.v2
network version 2.0 is not in supported versions 2
network API version 2.0, cmd group dims.network.v2
image API version 2, cmd group dims.image.v2
volume API version 1, cmd group dims.volume.v1
identity API version 2.0, cmd group dims.identity.v2
object_store API version 1, cmd group dims.object_store.v1
(dimscli) help

Shell commands (type help <topic>):
===================================
cmdenvironment  edit  hi       l   list  pause  r    save  shell      show
ed              help  history  li  load  py     run  set   shortcuts

Undocumented commands:
======================
EOF  eof  exit  q  quit

Application commands (type help <topic>):
=========================================
aggregate add host     host show              role list
aggregate create       ip fixed add           role remove
aggregate delete       ip fixed remove        role show
aggregate list         ip floating add        security group create
aggregate remove host  ip floating create     security group delete
aggregate set          ip floating delete     security group list
aggregate show         ip floating list       security group rule create
catalog list           ip floating pool list  security group rule delete
catalog show           ip floating remove     security group rule list
command list           keypair create         security group set
complete               keypair delete         security group show
configuration show     keypair list           server create
console log show       keypair show           server delete
console url show       module list            server image create
container create       network create         server list
container delete       network delete         server reboot
container list         network list           server rebuild
container save         network set            server set
container show         network show           server show
endpoint create        object create          server ssh
endpoint delete        object delete          service create
endpoint list          object list            service delete
endpoint show          object save            service list
```

```
extension list          object show             service show
flavor create           project create          token issue
flavor delete           project delete          token revoke
flavor list             project list            user create
flavor set              project set             user delete
flavor show             project show            user list
flavor unset            role add                user role list
help                    role create             user set
host list               role delete             user show

(dimscli) exit
END return value: 0
[dimscli] dittrich@dimsdemo1:~/dims/git/python-dimscli (develop) $
```

12.2 Command Structure

The `dimscli` shell follows the `openstack` client in the manner in which commands are to be constructed. See the Openstack Command Structure page for details. To quote:

> *Commands consist of an object described by one or more words followed by an action. Commands that require two objects have the primary object ahead of the action and the secondary object after the action. Any positional arguments identifying the objects shall appear in the same order as the objects. In badly formed English it is expressed as "(Take) object1 (and perform) action (using) object2 (to it).".*

```
<object-1> <action> <object-2>
```

Examples:

```
$ group add user <group> <user>

$ volume type list   # 'volume type' is a two-word single object
```

12.3 Completing commands in `dimscli`

The initial implementation of `dimscli` ported from the `openstackclient` code base does not have much actual code underlying it, though the scaffolding of `openstackclient` and many of its defined modules are currently configured in the code. You can see the modules that are not there by simply asking for `dimscli --help` and noting the errors (and what they point to, which indicates which code you need to seek out to use and/or replace.)

```
[dimscli] dittrich@dimsdemo1:~/dims/git/python-dimscli (develop) $ dimscli --help
defaults: {u'auth_type': 'password', u'compute_api_version': u'2', 'key': None, u
↪'database_api_version': u'1.0', 'api_timeout': None, u'baremetal_api_version': u'1',
↪ 'cacert': None, u'image_api_use_tasks
': False, u'floating_ip_source': u'neutron', u'orchestration_api_version': u'1', u
↪'interface': None, u'network_api_version': u'2.0', u'image_format': u'qcow2', u
↪'object_api_version': u'1', u'image_api_ve
rsion': u'2', 'verify': True, u'identity_api_version': u'2.0', u'volume_api_version':␣
↪u'1', 'cert': None, u'secgroup_source': u'neutron', u'dns_api_version': u'2', u
↪'disable_vendor_agent': {}}
cloud cfg: {'auth_type': 'password', u'compute_api_version': u'2', u'orchestration_
↪api_version': u'1', u'database_api_version': u'1.0', 'cacert': None, u'network_api_
↪version': u'2.0', u'image_format': u'
qcow2', u'object_api_version': u'1', u'image_api_version': u'2', 'verify': True, u
↪'dns_api_version': u'2', 'verbose_level': '1', 'region_name': '', 'api_timeout':␣
↪None, u'baremetal_api_version': u'1', 'a
```

```
uth': {}, 'default_domain': 'default', u'image_api_use_tasks': False, u'floating_ip_
↪source': u'neutron', 'key': None, 'timing': False, 'deferred_help': True, u
↪'identity_api_version': u'2.0', u'volume_api
_version': u'1', 'cert': None, u'secgroup_source': u'neutron', u'interface': None, u
↪'disable_vendor_agent': {}}
compute API version 2, cmd group dims.compute.v2
network version 2.0 is not in supported versions 2
network API version 2.0, cmd group dims.network.v2
image API version 2, cmd group dims.image.v2
volume API version 1, cmd group dims.volume.v1
identity API version 2.0, cmd group dims.identity.v2
object_store API version 1, cmd group dims.object_store.v1
usage: dimscli [--version] [-v] [--log-file LOG_FILE] [-q] [-h] [--debug]
               [--os-cloud <cloud-config-name>]
               [--os-region-name <auth-region-name>]
               [--os-cacert <ca-bundle-file>] [--verify | --insecure]
               [--os-default-domain <auth-domain>]
 ...

  --os-object-api-version <object-api-version>
                        Object API version, default=1 (Env:
                        OS_OBJECT_API_VERSION)

Commands:
Could not load EntryPoint.parse('aggregate_add_host = dimscli.compute.v2.
↪aggregate:AddAggregateHost')
Could not load EntryPoint.parse('aggregate_create = dimscli.compute.v2.
↪aggregate:CreateAggregate')
Could not load EntryPoint.parse('aggregate_delete = dimscli.compute.v2.
↪aggregate:DeleteAggregate')
Could not load EntryPoint.parse('aggregate_list = dimscli.compute.v2.
↪aggregate:ListAggregate')
Could not load EntryPoint.parse('aggregate_remove_host = dimscli.compute.v2.
↪aggregate:RemoveAggregateHost')
Could not load EntryPoint.parse('aggregate_set = dimscli.compute.v2.
↪aggregate:SetAggregate')
Could not load EntryPoint.parse('aggregate_show = dimscli.compute.v2.
↪aggregate:ShowAggregate')
Could not load EntryPoint.parse('catalog_list = dimscli.identity.v2_0.
↪catalog:ListCatalog')
Could not load EntryPoint.parse('catalog_show = dimscli.identity.v2_0.
↪catalog:ShowCatalog')
Could not load EntryPoint.parse('command_list = dimscli.common.module:ListCommand')
  complete       print bash completion command
Could not load EntryPoint.parse('configuration_show = dimscli.common.
↪configuration:ShowConfiguration')
Could not load EntryPoint.parse('console_log_show = dimscli.compute.v2.
↪console:ShowConsoleLog')
Could not load EntryPoint.parse('console_url_show = dimscli.compute.v2.
↪console:ShowConsoleURL')
Could not load EntryPoint.parse('container_create = dimscli.object.v1.
↪container:CreateContainer')
Could not load EntryPoint.parse('container_delete = dimscli.object.v1.
↪container:DeleteContainer')
Could not load EntryPoint.parse('container_list = dimscli.object.v1.
↪container:ListContainer')
Could not load EntryPoint.parse('container_save = dimscli.object.v1.
↪container:SaveContainer')
```

```
Could not load EntryPoint.parse('container_show = dimscli.object.v1.
↪container:ShowContainer')
Could not load EntryPoint.parse('endpoint_create = dimscli.identity.v2_0.
↪endpoint:CreateEndpoint')
Could not load EntryPoint.parse('endpoint_delete = dimscli.identity.v2_0.
↪endpoint:DeleteEndpoint')
Could not load EntryPoint.parse('endpoint_list = dimscli.identity.v2_0.
↪endpoint:ListEndpoint')
Could not load EntryPoint.parse('endpoint_show = dimscli.identity.v2_0.
↪endpoint:ShowEndpoint')
Could not load EntryPoint.parse('extension_list = dimscli.common.
↪extension:ListExtension')
Could not load EntryPoint.parse('flavor_create = dimscli.compute.v2.
↪flavor:CreateFlavor')
Could not load EntryPoint.parse('flavor_delete = dimscli.compute.v2.
↪flavor:DeleteFlavor')
Could not load EntryPoint.parse('flavor_list = dimscli.compute.v2.flavor:ListFlavor')
Could not load EntryPoint.parse('flavor_set = dimscli.compute.v2.flavor:SetFlavor')
Could not load EntryPoint.parse('flavor_show = dimscli.compute.v2.flavor:ShowFlavor')
Could not load EntryPoint.parse('flavor_unset = dimscli.compute.v2.flavor:UnsetFlavor
↪')
  help           print detailed help for another command
Could not load EntryPoint.parse('host_list = dimscli.compute.v2.host:ListHost')
Could not load EntryPoint.parse('host_show = dimscli.compute.v2.host:ShowHost')
Could not load EntryPoint.parse('ip_fixed_add = dimscli.compute.v2.fixedip:AddFixedIP
↪')
Could not load EntryPoint.parse('ip_fixed_remove = dimscli.compute.v2.
↪fixedip:RemoveFixedIP')
Could not load EntryPoint.parse('ip_floating_add = dimscli.compute.v2.
↪floatingip:AddFloatingIP')
Could not load EntryPoint.parse('ip_floating_create = dimscli.compute.v2.
↪floatingip:CreateFloatingIP')
Could not load EntryPoint.parse('ip_floating_delete = dimscli.compute.v2.
↪floatingip:DeleteFloatingIP')
Could not load EntryPoint.parse('ip_floating_list = dimscli.compute.v2.
↪floatingip:ListFloatingIP')
Could not load EntryPoint.parse('ip_floating_pool_list = dimscli.compute.v2.
↪floatingippool:ListFloatingIPPool')
Could not load EntryPoint.parse('ip_floating_remove = dimscli.compute.v2.
↪floatingip:RemoveFloatingIP')
Could not load EntryPoint.parse('keypair_create = dimscli.compute.v2.
↪keypair:CreateKeypair')
Could not load EntryPoint.parse('keypair_delete = dimscli.compute.v2.
↪keypair:DeleteKeypair')
Could not load EntryPoint.parse('keypair_list = dimscli.compute.v2.keypair:ListKeypair
↪')
Could not load EntryPoint.parse('keypair_show = dimscli.compute.v2.keypair:ShowKeypair
↪')
Could not load EntryPoint.parse('module_list = dimscli.common.module:ListModule')
Could not load EntryPoint.parse('network_create = dimscli.network.v2.
↪network:CreateNetwork')
Could not load EntryPoint.parse('network_delete = dimscli.network.v2.
↪network:DeleteNetwork')
Could not load EntryPoint.parse('network_list = dimscli.network.v2.network:ListNetwork
↪')
Could not load EntryPoint.parse('network_set = dimscli.network.v2.network:SetNetwork')
Could not load EntryPoint.parse('network_show = dimscli.network.v2.network:ShowNetwork
↪')
```

```
Could not load EntryPoint.parse('object_create = dimscli.object.v1.object:CreateObject
↪')
Could not load EntryPoint.parse('object_delete = dimscli.object.v1.object:DeleteObject
↪')
Could not load EntryPoint.parse('object_list = dimscli.object.v1.object:ListObject')
Could not load EntryPoint.parse('object_save = dimscli.object.v1.object:SaveObject')
Could not load EntryPoint.parse('object_show = dimscli.object.v1.object:ShowObject')
Could not load EntryPoint.parse('project_create = dimscli.identity.v2_0.
↪project:CreateProject')
Could not load EntryPoint.parse('project_delete = dimscli.identity.v2_0.
↪project:DeleteProject')
Could not load EntryPoint.parse('project_list = dimscli.identity.v2_0.
↪project:ListProject')
Could not load EntryPoint.parse('project_set = dimscli.identity.v2_0.
↪project:SetProject')
Could not load EntryPoint.parse('project_show = dimscli.identity.v2_0.
↪project:ShowProject')
Could not load EntryPoint.parse('role_add = dimscli.identity.v2_0.role:AddRole')
Could not load EntryPoint.parse('role_create = dimscli.identity.v2_0.role:CreateRole')
Could not load EntryPoint.parse('role_delete = dimscli.identity.v2_0.role:DeleteRole')
Could not load EntryPoint.parse('role_list = dimscli.identity.v2_0.role:ListRole')
Could not load EntryPoint.parse('role_remove = dimscli.identity.v2_0.role:RemoveRole')
Could not load EntryPoint.parse('role_show = dimscli.identity.v2_0.role:ShowRole')
Could not load EntryPoint.parse('security_group_create = dimscli.compute.v2.security_
↪group:CreateSecurityGroup')
Could not load EntryPoint.parse('security_group_delete = dimscli.compute.v2.security_
↪group:DeleteSecurityGroup')
Could not load EntryPoint.parse('security_group_list = dimscli.compute.v2.security_
↪group:ListSecurityGroup')
Could not load EntryPoint.parse('security_group_rule_create = dimscli.compute.v2.
↪security_group:CreateSecurityGroupRule')
Could not load EntryPoint.parse('security_group_rule_delete = dimscli.compute.v2.
↪security_group:DeleteSecurityGroupRule')
Could not load EntryPoint.parse('security_group_rule_list = dimscli.compute.v2.
↪security_group:ListSecurityGroupRule')
Could not load EntryPoint.parse('security_group_set = dimscli.compute.v2.security_
↪group:SetSecurityGroup')
Could not load EntryPoint.parse('security_group_show = dimscli.compute.v2.security_
↪group:ShowSecurityGroup')
Could not load EntryPoint.parse('server_create = dimscli.compute.v2.
↪server:CreateServer')
Could not load EntryPoint.parse('server_delete = dimscli.compute.v2.
↪server:DeleteServer')
Could not load EntryPoint.parse('server_image_create = dimscli.compute.v2.
↪server:CreateServerImage')
Could not load EntryPoint.parse('server_list = dimscli.compute.v2.server:ListServer')
Could not load EntryPoint.parse('server_reboot = dimscli.compute.v2.
↪server:RebootServer')
Could not load EntryPoint.parse('server_rebuild = dimscli.compute.v2.
↪server:RebuildServer')
Could not load EntryPoint.parse('server_set = dimscli.compute.v2.server:SetServer')
Could not load EntryPoint.parse('server_show = dimscli.compute.v2.server:ShowServer')
Could not load EntryPoint.parse('server_ssh = dimscli.compute.v2.server:SshServer')
Could not load EntryPoint.parse('service_create = dimscli.identity.v2_0.
↪service:CreateService')
Could not load EntryPoint.parse('service_delete = dimscli.identity.v2_0.
↪service:DeleteService')
Could not load EntryPoint.parse('service_list = dimscli.identity.v2_0.
↪service:ListService')
```

```
Could not load EntryPoint.parse('service_show = dimscli.identity.v2_0.
↪service:ShowService')
Could not load EntryPoint.parse('token_issue = dimscli.identity.v2_0.token:IssueToken
↪')
Could not load EntryPoint.parse('token_revoke = dimscli.identity.v2_0.
↪token:RevokeToken')
Could not load EntryPoint.parse('user_create = dimscli.identity.v2_0.user:CreateUser')
Could not load EntryPoint.parse('user_delete = dimscli.identity.v2_0.user:DeleteUser')
Could not load EntryPoint.parse('user_list = dimscli.identity.v2_0.user:ListUser')
Could not load EntryPoint.parse('user_role_list = dimscli.identity.v2_0.
↪role:ListUserRole')
Could not load EntryPoint.parse('user_set = dimscli.identity.v2_0.user:SetUser')
Could not load EntryPoint.parse('user_show = dimscli.identity.v2_0.user:ShowUser')
END return value: 1
[dimscli] dittrich@dimsdemo1:~/dims/git/python-dimscli (develop) $
```

Using the last error message above as an example, there needs to be a module named `$GIT/python-dimscli/dimscli/identity/v2_0/user.py` with a class `ShowUser`. Look in the `python-openstack/openstack/identity/v2_0/` directory for their `user.py` and build off that example.

Attention: Clone the `python-openstackclient` repo using `git clone https://git.openstack.org/openstack/python-openstackclient` and see the `cliff` documentation, Section Exploring the Demo App, for how this works.

Attention: See the file `$GIT/python-dimscli/README.rst` for more documentation produced during initial creation of the openstackclient fork of `dimscli`.

`cliff` supports list formatting in tables, CSV, JSON, etc., but not in shell format. That is only supported by the `ShowOne` class, which is not what we want for producing a set of variables for insertion into shell environments.

```
[dimsenv] dittrich@dimsdemo1:~/dims/git/python-dimscli (develop*) $ dimscli list nodes
+----------------+---------------+
| Node           | Address       |
+----------------+---------------+
| b52            | 10.86.86.7    |
| consul-breathe | 10.142.29.117 |
| consul-echoes  | 10.142.29.116 |
| consul-seamus  | 10.142.29.120 |
| dimsdemo1      | 10.86.86.2    |
| dimsdev1       | 10.86.86.5    |
| dimsdev2       | 10.86.86.5    |
| four           | 192.168.0.101 |
+----------------+---------------+
```

```
[dimsenv] dittrich@dimsdemo1:~/dims/git/python-dimscli (develop*) $ dimscli list␣
↪nodes -f csv
"Node","Address"
"b52","10.86.86.7"
"consul-breathe","10.142.29.117"
"consul-echoes","10.142.29.116"
"consul-seamus","10.142.29.120"
"dimsdemo1","10.86.86.2"
"dimsdev1","10.86.86.5"
```

```
"dimsdev2","10.86.86.5"
"four","192.168.0.101"
```

```
[dimsenv] dittrich@dimsdemo1:~/dims/git/python-dimscli (develop*) $ dimscli list␣
↪nodes -f json
[{"Node": "b52", "Address": "10.86.86.7"}, {"Node": "consul-breathe", "Address": "10.
↪142.29.117"}, {"Node": "consul-echoes", "Address": "10.142.29.116"}, {"Node":
↪"consul-seamus", "Address": "10.142.29.120"}, {"Node": "dimsdemo1", "Address": "10.
↪86.86.2"}, {"Node": "dimsdev1", "Address": "10.86.86.5"}, {"Node": "dimsdev2",
↪"Address": "10.86.86.5"}, {"Node": "four", "Address": "192.168.0.101"}]
[dimsenv] dittrich@dimsdemo1:~/dims/git/python-dimscli (develop*) $ dimscli list␣
↪nodes -f json | python -m json.tool
[
    {
        "Address": "10.86.86.7",
        "Node": "b52"
    },
    {
        "Address": "10.142.29.117",
        "Node": "consul-breathe"
    },
    {
        "Address": "10.142.29.116",
        "Node": "consul-echoes"
    },
    {
        "Address": "10.142.29.120",
        "Node": "consul-seamus"
    },
    {
        "Address": "10.86.86.2",
        "Node": "dimsdemo1"
    },
    {
        "Address": "10.86.86.5",
        "Node": "dimsdev1"
    },
    {
        "Address": "10.86.86.5",
        "Node": "dimsdev2"
    },
    {
        "Address": "192.168.0.101",
        "Node": "four"
    }
]
```

To produce the list in the form of shell variables, we need to create a custom formatter and load it into the `dimscli` shell via Stevedore.

After adding the new formatter, it is possible to extract the list of nodes registered with Consul and produce a set of variable declarations from the list.

```
[dimsenv] dittrich@dimsdemo1:~/dims/git/python-dimscli (develop*) $ dimscli list␣
↪nodes -f shell
b52="10.86.86.7"
consul_breathe="10.142.29.117"
consul_echoes="10.142.29.116"
```

```
consul_seamus="10.142.29.120"
dimsdemo1="10.86.86.2"
dimsdev1="10.86.86.5"
dimsdev2="10.86.86.5"
four="192.168.0.101"
```

In practice, you may wish to insert these as variables in the shell's `set` using the `eval` statement for use when invoking shell commands:

```
[dimsenv] dittrich@dimsdemo1:~/dims/git/python-dimscli (develop*) $ eval $(dimscli␣
↪list nodes -f shell --prefix=DIMS_)
[dimsenv] dittrich@dimsdemo1:~/dims/git/python-dimscli (develop*) $ set | grep DIMS_
DIMS_REV=unspecified
DIMS_VERSION='1.6.124 (dims-ci-utils)'
DIMS_b52=10.86.86.7
DIMS_consul_breathe=10.142.29.117
DIMS_consul_echoes=10.142.29.116
DIMS_consul_seamus=10.142.29.120
DIMS_dimsdemo1=10.86.86.2
DIMS_dimsdev2=10.86.86.5
DIMS_four=192.168.0.101
    echo "REV:      $DIMS_REV";
    echo "[dims-ci-utils version $(version) (rev $DIMS_REV)]";
        echo "$PROGRAM $DIMS_VERSION";
        echo "$BASE $DIMS_VERSION";
```

12.4 Adding New Columns to Output

Say we want to also include the Consul status, to help determine which node is currently the *Leader* in a cluster, which are a *Peer* in the cluster, and which are simply an *Agent* that is proxying to the cluster.

The changes to existing code to affect this new feature are shown here:

```
commit caab2d05274898878e1123bd337b431c8d2f2a8e
Author: Dave Dittrich <dittrich@u.washington.edu>
Date:   Sat Jan 2 12:53:56 2016 -0800

    Add Consul node status to 'nodes list' output

diff --git a/dimscli/list.py b/dimscli/list.py
index 45acdda..3893b10 100644
--- a/dimscli/list.py
+++ b/dimscli/list.py
@@ -26,9 +26,35 @@ class Nodes(Lister):

     log = logging.getLogger(__name__)

+    def get_node_status(self):
+        """
+        Determine the status from Consul
+
+        :return: None
+        """
+        self.leaderDict = dict(zip(['Address', 'Port'],
+                                   self.consul.status.leader().split(":")))
+        self.peersDictList = [dict(zip(['Address', 'Port'], p.split(":")))
```

```
+                               for p in self.consul.status.peers()]
+
+    def status(self, address):
+        """
+        Determine node status as returned from Consul.
+
+        :param address: IP address to check
+        :return: One of: "Leader", "Peer", or "Agent"
+        """
+        if address in self.leaderDict.values():
+            return "Leader"
+        elif address in [p['Address'] for p in self.peersDictList]:
+            return "Peer"
+        else:
+            return "Agent"
+
     def take_action(self, parsed_args):
-        consul = consulate.Consul()
-        nodes = consul.catalog.nodes()
-        columns = ('Node', 'Address')
-        data = ((node['Node'], node['Address']) for node in nodes)
+        self.consul = consulate.Consul()
+        nodes = self.consul.catalog.nodes()
+        self.get_node_status()
+        columns = ('Node', 'Address', 'Status')
+        data = ((node['Node'], node['Address'], self.status(node['Address'])) for␣
↪node in nodes)
         return (columns, data)
```

```
[dimsenv] dittrich@dimsdemo1:~/dims/git/python-dimscli (develop*) $ dimscli nodes list
+-----------+---------------+--------+
| Node      | Address       | Status |
+-----------+---------------+--------+
| b52       | 10.86.86.2    | Agent  |
| breathe   | 10.142.29.117 | Leader |
| dimsdemo1 | 10.86.86.3    | Agent  |
| echoes    | 10.142.29.116 | Peer   |
| seamus    | 10.142.29.120 | Peer   |
+-----------+---------------+--------+
[dimsenv] dittrich@dimsdemo1:~/dims/git/python-dimscli (develop*) $ dimscli nodes␣
↪list -f csv
"Node","Address","Status"
"b52","10.86.86.2","Agent"
"breathe","10.142.29.117","Leader"
"dimsdemo1","10.86.86.3","Agent"
"echoes","10.142.29.116","Peer"
"seamus","10.142.29.120","Peer"
[dimsenv] dittrich@dimsdemo1:~/dims/git/python-dimscli (develop*) $ dimscli nodes␣
↪list -f json | python -mjson.tool
[
    {
        "Address": "10.86.86.2",
        "Node": "b52",
        "Status": "Agent"
    },
    {
        "Address": "10.142.29.117",
        "Node": "breathe",
```

```
        "Status": "Leader"
    },
    {
        "Address": "10.86.86.3",
        "Node": "dimsdemo1",
        "Status": "Agent"
    },
    {
        "Address": "10.142.29.116",
        "Node": "echoes",
        "Status": "Peer"
    },
    {
        "Address": "10.142.29.120",
        "Node": "seamus",
        "Status": "Peer"
    }
]
```

If we wish to turn a subset of this table into variables, using the `shell` output feature added above, we need to select a pair of columns (to map to *Variable=Value* in the output). The results could then be used in Ansible playbooks, shell scripts, selecting color for nodes in a graph, or any number of other purposes.

```
[dimsenv] dittrich@dimsdemo1:~/dims/git/python-dimscli (develop*) $ dimscli nodes␣
↪list --column Node --column Status -f shell
b52="Agent"
breathe="Leader"
dimsdemo1="Agent"
echoes="Peer"
seamus="Peer"
```

12.5 Adding New Commands

In this example, we will add a new command `ansible` with a subcommand `execute` that will use Ansible's Python API (specifically the `ansible.runner.Runner` class) to execute arbitrary commands on hosts via Ansible.

Note: What is being demonstrated here is adding a new subcommand to the `dimscli` repo directly. It is also possible to add a new command from a module in another repo using Stevedore.

Here are the changes that implement this new command:

```
commit eccf3af707aac5a13144580bfbf548b45616d49f
Author: Dave Dittrich <dittrich@u.washington.edu>
Date:   Fri Jan 1 20:34:42 2016 -0800

    Add 'ansible execute' command

diff --git a/dimscli/dimsansible/__init__.py b/dimscli/dimsansible/__init__.py
new file mode 100644
index 0000000..e69de29
diff --git a/dimscli/dimsansible/ansiblerunner.py b/dimscli/dimsansible/ansiblerunner.
↪py
new file mode 100644
index 0000000..68cd3ea
```

```
--- /dev/null
+++ b/dimscli/dimsansible/ansiblerunner.py
@@ -0,0 +1,61 @@
+#!/usr/bin/python
+
+import sys
+import logging
+
+from cliff.lister import Lister
+from ansible.runner import Runner
+
+HOST_LIST = "/etc/ansible/hosts"
+CMD = "/usr/bin/uptime"
+
+class Execute(Lister):
+    """Execute a command via Ansible and return a list of results.
+
+    """
+
+    log = logging.getLogger(__name__)
+
+    def get_parser(self, prog_name):
+        parser = super(Execute, self).get_parser(prog_name)
+        parser.add_argument(
+            "--host-list",
+            metavar="<host-list>",
+            default=HOST_LIST,
+            help="Hosts file (default: {})".format(HOST_LIST),
+        )
+        parser.add_argument(
+            "--program",
+            metavar="<program>",
+            default=CMD,
+            help="Program to run (default: {})".format(CMD),
+        )
+        return parser
+
+    def take_action(self, parsed_args):
+
+        results = Runner(
+                host_list=parsed_args.host_list,
+                pattern='*',
+                forks=10,
+                module_name='command',
+                module_args=parsed_args.program,
+        ).run()
+
+        if results is None:
+           print "No hosts found"
+           sys.exit(1)
+
+        outtable = []
+
+        for (hostname, result) in results['contacted'].items():
+            if not 'failed' in result:
+                outtable.append((hostname, 'GOOD', result['stdout']))
+            elif 'failed' in result:
+                outtable.append((hostname, 'FAIL', result['msg']))
```

```
+        for (hostname, result) in results['dark'].items():
+            outtable.append((hostname, 'DARK', result['msg']))
+
+        column_names = ('Host', 'Status', 'Results')
+
+        return column_names, outtable
diff --git a/setup.cfg b/setup.cfg
index 14f6ce7..9571d4f 100644
--- a/setup.cfg
+++ b/setup.cfg
@@ -37,6 +37,7 @@ dims.cli =
     files_list = dimscli.list:Files
     nodes_list = dimscli.list:Nodes
     show_file = dimscli.show:File
+    ansible_execute = dimscli.dimsansible.ansiblerunner:Execute

 cliff.formatter.list =
     shell = dimscli.formatters.shell:DIMSShellFormatter
```

Here is what the command can do (as seen in the `--help` output).

```
[dimscli] dittrich@dimsdemo1:ims/git/python-dimscli/dimscli (develop*) $ dimscli␣
↪ansible execute --help
usage: dimscli ansible execute [-h]
                               [-f {csv,html,json,json,shell,table,value,yaml,yaml}]
                               [-c COLUMN] [--prefix PREFIX]
                               [--max-width <integer>] [--noindent]
                               [--quote {all,minimal,none,nonnumeric}]
                               [--host-list <host-list>] [--program <program>]

Execute a command via Ansible and return a list of results.

optional arguments:
  -h, --help            show this help message and exit
  --host-list <host-list>
                        Hosts file (default: /etc/ansible/hosts)
  --program <program>   Program to run (default: /usr/bin/uptime)

output formatters:
  output formatter options

  -f {csv,html,json,json,shell,table,value,yaml,yaml}, --format {csv,html,json,json,
↪shell,table,value,yaml,yaml}
                        the output format, defaults to table
  -c COLUMN, --column COLUMN
                        specify the column(s) to include, can be repeated

shell formatter:
  a format a UNIX shell can parse (variable="value")

  --prefix PREFIX       add a prefix to all variable names

table formatter:
  --max-width <integer>
                        Maximum display width, 0 to disable

json formatter:
  --noindent            whether to disable indenting the JSON
```

```
CSV Formatter:
  --quote {all,minimal,none,nonnumeric}
                        when to include quotes, defaults to nonnumeric
```

The script defaults to using the standard Ansible `/etc/ansible/hosts` file to get its inventory. In this case, the DIMS `$GIT/ansible-inventory/development` file was copied to the default location. Using this file to execute the default command `/usr/bin/uptime` on the defined `development` hosts results in the following:

```
[dimscli] dittrich@dimsdemo1:ims/git/python-dimscli/dimscli (develop*) $ dimscli␣
↪ansible execute
+-------------------------------+--------+-----------------------------------------
↪------------------------------+
| Host                          | Status | Results                                 ␣
↪                              |
+-------------------------------+--------+-----------------------------------------
↪------------------------------+
| linda-vm1.devops.develop      | GOOD   |  18:31:22 up 146 days,  8:27,  1 user, ␣
↪load average: 0.00, 0.01, 0.05 |
| u12-dev-ws-1.devops.develop   | GOOD   |  18:31:21 up 146 days,  8:27,  1 user, ␣
↪load average: 0.00, 0.01, 0.05 |
| hub.devops.develop            | GOOD   |  02:31:22 up 128 days,  8:42,  1 user, ␣
↪load average: 0.00, 0.01, 0.05 |
| floyd2-p.devops.develop       | GOOD   |  18:31:21 up 20 days, 56 min,  1 user, ␣
↪load average: 0.02, 0.04, 0.05 |
| u12-dev-svr-1.devops.develop  | GOOD   |  18:31:22 up 142 days, 11:22,  1 user, ␣
↪load average: 0.00, 0.01, 0.05 |
+-------------------------------+--------+-----------------------------------------
↪------------------------------+
```

Using the `--program` command line option, a different command can be run:

```
[dimscli] dittrich@dimsdemo1:ims/git/python-dimscli/dimscli (develop*) $ dimscli␣
↪ansible execute --program "ip addr"
+-------------------------------+--------+-----------------------------------------
↪-------------------------------------------------------------------+
| Host                          | Status | Results                                 ␣
↪                                                                   |
+-------------------------------+--------+-----------------------------------------
↪-------------------------------------------------------------------+
| linda-vm1.devops.develop      | GOOD   | 1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536␣
↪qdisc noqueue state UNKNOWN                                         |
|                               |        |     link/loopback 00:00:00:00:00:00 brd␣
↪00:00:00:00:00:00                                                   |
|                               |        |     inet 127.0.0.1/8 scope host lo      ␣
↪                                                                    |
|                               |        |        valid_lft forever preferred_lft␣
↪forever                                                             |
|                               |        | 2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP>␣
↪mtu 1500 qdisc pfifo_fast state UP qlen 1000                        |
|                               |        |     link/ether 08:00:27:3b:3a:65 brd␣
↪ff:ff:ff:ff:ff:ff                                                   |
|                               |        |     inet 10.0.2.15/24 brd 10.0.2.255 scope␣
↪global eth0                                                         |
|                               |        |        valid_lft forever preferred_lft␣
↪forever                                                             |
|                               |        | 3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP>␣
↪mtu 1500 qdisc pfifo_fast state UP qlen 1000                        |
```

```
|                                 |        |     link/ether 08:00:27:36:2b:2c brd␣
↪ff:ff:ff:ff:ff:ff                                                              |
|                                 |        |     inet 192.168.88.11/24 brd 192.168.88.
↪255 scope global eth1                                                          |
|                                 |        |        valid_lft forever preferred_lft␣
↪forever                                                                        |
| u12-dev-svr-1.devops.develop | GOOD   | 1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536␣
↪qdisc noqueue state UNKNOWN                                                    |
|                                 |        |     link/loopback 00:00:00:00:00:00 brd␣
↪00:00:00:00:00:00                                                              |
|                                 |        |     inet 127.0.0.1/8 scope host lo      ␣
↪                                                                               |
|                                 |        |        valid_lft forever preferred_lft␣
↪forever                                                                        |
|                                 |        |     inet6 ::1/128 scope host            ␣
↪                                                                               |
|                                 |        |        valid_lft forever preferred_lft␣
↪forever                                                                        |
|                                 |        | 2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP>␣
↪mtu 1500 qdisc pfifo_fast state UP qlen 1000                                   |
|                                 |        |     link/ether 08:00:27:38:db:8c brd␣
↪ff:ff:ff:ff:ff:ff                                                              |
|                                 |        |     inet 10.0.2.15/24 brd 10.0.2.255 scope␣
↪global eth0                                                                    |
|                                 |        |        valid_lft forever preferred_lft␣
↪forever                                                                        |
|                                 |        |     inet6 fe80::a00:27ff:fe38:db8c/64 scope␣
↪link                                                                           |
|                                 |        |        valid_lft forever preferred_lft␣
↪forever                                                                        |
|                                 |        | 3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP>␣
↪mtu 1500 qdisc pfifo_fast state UP qlen 1000                                   |
|                                 |        |     link/ether 08:00:27:e7:80:52 brd␣
↪ff:ff:ff:ff:ff:ff                                                              |
|                                 |        |     inet 192.168.88.13/24 brd 192.168.88.
↪255 scope global eth1                                                          |
|                                 |        |        valid_lft forever preferred_lft␣
↪forever                                                                        |
|                                 |        |     inet6 fe80::a00:27ff:fee7:8052/64 scope␣
↪link                                                                           |
|                                 |        |        valid_lft forever preferred_lft␣
↪forever                                                                        |
| hub.devops.develop           | GOOD   | 1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536␣
↪qdisc noqueue state UNKNOWN group default                                      |
|                                 |        |     link/loopback 00:00:00:00:00:00 brd␣
↪00:00:00:00:00:00                                                              |
|                                 |        |     inet 127.0.0.1/8 scope host lo      ␣
↪                                                                               |
|                                 |        |        valid_lft forever preferred_lft␣
↪forever                                                                        |
|                                 |        |     inet6 ::1/128 scope host            ␣
↪                                                                               |
|                                 |        |        valid_lft forever preferred_lft␣
↪forever                                                                        |
|                                 |        | 2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP>␣
↪mtu 1500 qdisc pfifo_fast state UP group default qlen 1000                     |
|                                 |        |     link/ether 08:00:27:9c:f8:95 brd␣
↪ff:ff:ff:ff:ff:ff                                                              |
```

```
|                                 |        |     inet 10.0.2.15/24 brd 10.0.2.255 scope␣
↪global eth0                                                                          |
|                                 |        |        valid_lft forever preferred_lft␣
↪forever                                                                              |
|                                 |        |     inet6 fe80::a00:27ff:fe9c:f895/64 scope␣
↪link                                                                                 |
|                                 |        |        valid_lft forever preferred_lft␣
↪forever                                                                              |
|                                 |        | 3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP>␣
↪mtu 1500 qdisc pfifo_fast state UP group default qlen 1000                           |
|                                 |        |     link/ether 08:00:27:28:63:2a brd␣
↪ff:ff:ff:ff:ff:ff                                                                    |
|                                 |        |     inet 192.168.88.14/24 brd 192.168.88.
↪255 scope global eth1                                                                |
|                                 |        |        valid_lft forever preferred_lft␣
↪forever                                                                              |
|                                 |        |     inet6 fe80::a00:27ff:fe28:632a/64 scope␣
↪link                                                                                 |
|                                 |        |        valid_lft forever preferred_lft␣
↪forever                                                                              |
|                                 |        | 4: docker0: <BROADCAST,MULTICAST,UP,LOWER_
↪UP> mtu 1500 qdisc noqueue state UP group default                                    |
|                                 |        |     link/ether 56:84:7a:fe:97:99 brd␣
↪ff:ff:ff:ff:ff:ff                                                                    |
|                                 |        |     inet 172.17.42.1/16 scope global␣
↪docker0                                                                              |
|                                 |        |        valid_lft forever preferred_lft␣
↪forever                                                                              |
|                                 |        |     inet6 fe80::5484:7aff:fefe:9799/64␣
↪scope link                                                                           |
|                                 |        |        valid_lft forever preferred_lft␣
↪forever                                                                              |
|                                 |        | 22: veth6dc6dd5: <BROADCAST,MULTICAST,UP,
↪LOWER_UP> mtu 1500 qdisc noqueue master docker0 state UP group default               |
|                                 |        |     link/ether 8e:d6:f5:66:fb:88 brd␣
↪ff:ff:ff:ff:ff:ff                                                                    |
|                                 |        |     inet6 fe80::8cd6:f5ff:fe66:fb88/64␣
↪scope link                                                                           |
|                                 |        |        valid_lft forever preferred_lft␣
↪forever                                                                              |
|                                 |        | 42: vethdc35259: <BROADCAST,MULTICAST,UP,
↪LOWER_UP> mtu 1500 qdisc noqueue master docker0 state UP group default               |
|                                 |        |     link/ether 46:c3:87:32:83:a1 brd␣
↪ff:ff:ff:ff:ff:ff                                                                    |
|                                 |        |     inet6 fe80::44c3:87ff:fe32:83a1/64␣
↪scope link                                                                           |
|                                 |        |        valid_lft forever preferred_lft␣
↪forever                                                                              |
| floyd2-p.devops.develop         | GOOD   | 1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536␣
↪qdisc noqueue state UNKNOWN                                                          |
|                                 |        |     link/loopback 00:00:00:00:00:00 brd␣
↪00:00:00:00:00:00                                                                    |
|                                 |        |     inet 127.0.0.1/8 scope host lo␣
↪                                                                                     |
|                                 |        |        valid_lft forever preferred_lft␣
↪forever                                                                              |
|                                 |        | 2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP>␣
↪mtu 1500 qdisc pfifo_fast state UP qlen 1000                                         |
```

```
|                                 |        |     link/ether 52:54:00:17:19:9a brd␣
↪ff:ff:ff:ff:ff:ff                                                              |
|                                 |        |     inet 172.22.29.175/24 brd 172.22.29.255␣
↪scope global eth0                                                             |
|                                 |        |        valid_lft forever preferred_lft␣
↪forever                                                                       |
|                                 |        | 3: eth1: <BROADCAST,MULTICAST> mtu 1500␣
↪qdisc noop state DOWN qlen 1000                                               |
|                                 |        |     link/ether 52:54:00:85:34:b7 brd␣
↪ff:ff:ff:ff:ff:ff                                                             |
| u12-dev-ws-1.devops.develop  | GOOD   | 1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536␣
↪qdisc noqueue state UNKNOWN                                                   |
|                                 |        |     link/loopback 00:00:00:00:00:00 brd␣
↪00:00:00:00:00:00                                                             |
|                                 |        |     inet 127.0.0.1/8 scope host lo          ␣
↪                                                                              |
|                                 |        |        valid_lft forever preferred_lft␣
↪forever                                                                       |
|                                 |        | 2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP>␣
↪mtu 1500 qdisc pfifo_fast state UP qlen 1000                                  |
|                                 |        |     link/ether 08:00:27:07:6b:00 brd␣
↪ff:ff:ff:ff:ff:ff                                                             |
|                                 |        |     inet 10.0.2.15/24 brd 10.0.2.255 scope␣
↪global eth0                                                                   |
|                                 |        |        valid_lft forever preferred_lft␣
↪forever                                                                       |
|                                 |        | 3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP>␣
↪mtu 1500 qdisc pfifo_fast state UP qlen 1000                                  |
|                                 |        |     link/ether 08:00:27:75:a0:25 brd␣
↪ff:ff:ff:ff:ff:ff                                                             |
|                                 |        |     inet 192.168.88.12/24 brd 192.168.88.
↪255 scope global eth1                                                         |
|                                 |        |        valid_lft forever preferred_lft␣
↪forever                                                                       |
|                                 |        | 4: docker0: <BROADCAST,MULTICAST,UP,LOWER_
↪UP> mtu 1500 qdisc noqueue state UNKNOWN                                      |
|                                 |        |     link/ether 5a:cb:bd:c2:f5:82 brd␣
↪ff:ff:ff:ff:ff:ff                                                             |
|                                 |        |     inet 172.17.42.1/16 scope global␣
↪docker0                                                                       |
|                                 |        |        valid_lft forever preferred_lft␣
↪forever                                                                       |
+---------------------------------+--------+-----------------------------------------
↪-------------------------------------------------------------------+
```

```
[dimscli] dittrich@dimsdemo1:ims/git/python-dimscli/dimscli (develop*) $ dimscli␣
↪ansible execute --program "cat /etc/hosts"
+---------------------------------+--------+-----------------------------------------
↪-----------------------------------------------------------------------------------
↪------------+
| Host                            | Status | Results                                 ␣
↪                                                                                  ␣
↪            |
+---------------------------------+--------+-----------------------------------------
↪-----------------------------------------------------------------------------------
↪------------+
| linda-vm1.devops.develop     | GOOD   | 127.0.0.1      localhost                ␣
↪                                                                                  ␣
↪            |
```

```
|                              |         | 127.0.1.1      ubu12-generic                    ␣
↪                                                                                         ␣
↪             |
|                              |         |                                                ␣
↪                                                                                         ␣
↪             |
|                              |         | # The following lines are desirable for␣
↪IPv6 capable hosts                                                                       ␣
↪                 |
|                              |         | ::1     ip6-localhost ip6-loopback             ␣
↪                                                                                         ␣
↪             |
|                              |         | fe00::0 ip6-localnet                           ␣
↪                                                                                         ␣
↪             |
|                              |         | ff00::0 ip6-mcastprefix                        ␣
↪                                                                                         ␣
↪             |
|                              |         | ff02::1 ip6-allnodes                           ␣
↪                                                                                         ␣
↪             |
|                              |         | ff02::2 ip6-allrouters                         ␣
↪                                                                                         ␣
↪             |
|                              |         |                                                ␣
↪                                                                                         ␣
↪             |
|                              |         | 127.0.0.1      auth-test.devops.develop␣
↪manager-test.devops.develop  reload-test.devops.develop test5.prisem.washingto |
|                              |         | 127.0.0.1      auth-test.devops.develop␣
↪manager-test.devops.develop  reload-test.devops.develop test5.prisem.washingto |
|                              |         | 127.0.0.1      auth-test.devops.develop␣
↪manager-test.devops.develop  reload-test.devops.develop test5.prisem.washingto |
|                              |         | 127.0.0.1      auth-test.devops.develop␣
↪manager-test.devops.develop  reload-test.devops.develop test5.prisem.washingto |
|                              |         | 127.0.0.1      auth-test.devops.develop␣
↪manager-test.devops.develop  reload-test.devops.develop test5.prisem.washingto |
|                              |         | 127.0.0.1      auth-test.devops.develop␣
↪manager-test.devops.develop  reload-test.devops.develop test5.prisem.washingto |
|                              |         | 127.0.0.1      auth-test.devops.develop␣
↪manager-test.devops.develop  reload-test.devops.develop test5.prisem.washingto |
|                              |         | 127.0.0.1      auth-test.devops.develop␣
↪manager-test.devops.develop  reload-test.devops.develop test5.prisem.washingto |
|                              |         | 127.0.0.1      auth-test.devops.develop␣
↪manager-test.devops.develop  reload-test.devops.develop test5.prisem.washingto |
| u12-dev-svr-1.devops.develop | GOOD    | 127.0.0.1      localhost                        ␣
↪                                                                                         ␣
↪             |
|                              |         | 127.0.1.1      u12-dev-svr-1                    ␣
↪                                                                                         ␣
↪             |
|                              |         |                                                ␣
↪                                                                                         ␣
↪             |
|                              |         | # The following lines are desirable for␣
↪IPv6 capable hosts                                                                       ␣
↪                 |
|                              |         | ::1     ip6-localhost ip6-loopback             ␣
↪                                                                                         ␣
↪             |
```

```
|                               |         | fe00::0 ip6-localnet                ␣
↪                                                                             ␣
↪             |
|                               |         | ff00::0 ip6-mcastprefix             ␣
↪                                                                             ␣
↪             |
|                               |         | ff02::1 ip6-allnodes                ␣
↪                                                                             ␣
↪             |
|                               |         | ff02::2 ip6-allrouters              ␣
↪                                                                             ␣
↪             |
| hub.devops.develop            | GOOD    | 127.0.0.1      localhost            ␣
↪                                                                             ␣
↪             |
|                               |         | 127.0.1.1      hub                  ␣
↪                                                                             ␣
↪             |
|                               |         |                                     ␣
↪                                                                             ␣
↪             |
|                               |         | # The following lines are desirable for␣
↪IPv6 capable hosts                                                           ␣
↪                 |
|                               |         | ::1     localhost ip6-localhost ip6-
↪loopback                                                                     ␣
↪                     |
|                               |         | ff02::1 ip6-allnodes                ␣
↪                                                                             ␣
↪             |
|                               |         | ff02::2 ip6-allrouters              ␣
↪                                                                             ␣
↪             |
|                               |         | 127.0.1.1 hub                       ␣
↪                                                                             ␣
↪             |
| floyd2-p.devops.develop       | GOOD    | 127.0.0.1      localhost            ␣
↪                                                                             ␣
↪             |
|                               |         | 127.0.0.1      floyd2-p floyd2-p.devops.
↪develop                                                                      ␣
↪          |
| u12-dev-ws-1.devops.develop   | GOOD    | 127.0.0.1      localhost            ␣
↪                                                                             ␣
↪             |
|                               |         | 127.0.1.1      u12-dev-1            ␣
↪                                                                             ␣
↪             |
|                               |         |                                     ␣
↪                                                                             ␣
↪             |
|                               |         | # The following lines are desirable for␣
↪IPv6 capable hosts                                                           ␣
↪                 |
|                               |         | ::1     ip6-localhost ip6-loopback  ␣
↪                                                                             ␣
↪             |
|                               |         | fe00::0 ip6-localnet                ␣
↪                                                                             ␣
↪             |
```

```
|                              |        | ff00::0 ip6-mcastprefix
↪                                                                                ␣
↪            |                                                                   ␣
|                              |        | ff02::1 ip6-allnodes
↪                                                                                ␣
↪            |                                                                   ␣
|                              |        | ff02::2 ip6-allrouters
↪                                                                                ␣
↪            |                                                                   ␣
+------------------------------+--------+-----------------------------------------
↪----------------------------------------------------------------------------------
↪------------+
```

To run a command across the full set of ansible-compatible hosts, we can use the helper `Makefile` in the `$GIT/ansible-inventory` repo to extract a list of *all* hosts specified in *any* inventory file to form a complete set.

Note: This helper `Makefile` was originally written to take a set of static inventory files and generate a set, rather than forcing someone to manually edit a file and manually combine all hosts from any file (which is error prone, tedious, difficult to remember how to do... basically impractical for a scalable solution.)

```
[dimscli] dittrich@dimsdemo1:ims/git/python-dimscli/dimscli (develop*) $ (cd $GIT/
↪ansible-inventory; make help)
usage: make [something]

Where 'something' is one of:

 help - Show this help information
 all -           Default is create complete_inventory file.

 inventory -     Create file 'complete_inventory' with all hosts
                 from any file with an '[all]' section in it.

 tree -          Produce a tree listing of everything except
                 'older-*' directories.

 clean -         Clean up files.


[dimscli] dittrich@dimsdemo1:ims/git/python-dimscli/dimscli (develop*) $ (cd $GIT/
↪ansible-inventory; make inventory)
echo '[all]' > complete_inventory
cat development hosts-old infrastructure Makefile prisem project | awk '\
            /^\[all\]/ { echo = 1; next; }\
            /^$/       { echo = 0; }\
                         { if (echo == 1) { print; } }' |\
                sort | uniq >> complete_inventory
```

Now this list can be used to run the command across the full set of hosts under Ansible control.

```
[dimscli] dittrich@dimsdemo1:ims/git/python-dimscli/dimscli (develop*) $ dimscli␣
↪ansible execute --host-list /home/dittrich/dims/git/ansible-inventory/complete_
↪inventory
+------------------------------+--------+-----------------------------------------
↪----------------------------------------------------------------------------------
↪-----------------------------------------+
| Host                         | Status | Results
↪                                                                                ␣
↪                                                |
```

```
+-----------------------------+--------+-----------------------------------------
↪---------------------------------------------------------------------------------
↪----------------------------------------+
| rabbitmq.devops.develop      | GOOD   |  18:35:04 up 20 days,  1:00,  1 user,  load␣
↪average: 0.00, 0.04, 0.05                                                       ␣
↪                                        |
| wellington.devops.develop    | GOOD   |  18:35:06 up 146 days,  8:43,  1 user, ␣
↪load average: 0.43, 0.64, 0.43                                                  ␣
↪                                          |
| hub.devops.develop           | GOOD   |  02:35:02 up 128 days,  8:46,  1 user, ␣
↪load average: 0.11, 0.06, 0.05                                                  ␣
↪                                          |
| git.devops.develop           | GOOD   |  18:35:03 up 146 days,  8:30,  2 users, ␣
↪load average: 0.18, 0.07, 0.06                                                  ␣
↪                                         |
| time.devops.develop          | GOOD   |  18:35:04 up 20 days,  1:00,  2 users, ␣
↪load average: 0.06, 0.13, 0.13                                                  ␣
↪                                          |
| jira-int.devops.develop      | GOOD   |  18:35:03 up 146 days,  8:30,  2 users, ␣
↪load average: 0.18, 0.07, 0.06                                                  ␣
↪                                         |
| u12-dev-ws-1.devops.develop  | GOOD   |  18:35:05 up 146 days,  8:30,  1 user, ␣
↪load average: 0.01, 0.02, 0.05                                                  ␣
↪                                          |
| sso.devops.develop           | GOOD   |  18:35:05 up 146 days,  8:30,  1 user, ␣
↪load average: 0.00, 0.02, 0.05                                                  ␣
↪                                          |
| lapp-int.devops.develop      | GOOD   |  18:35:02 up 146 days,  8:31,  2 users, ␣
↪load average: 0.16, 0.05, 0.06                                                  ␣
↪                                         |
| foswiki-int.devops.develop   | GOOD   |  18:35:03 up 146 days,  8:31,  1 user, ␣
↪load average: 0.00, 0.01, 0.05                                                  ␣
↪                                          |
| u12-dev-svr-1.devops.develop | GOOD   |  18:35:03 up 142 days, 11:26,  1 user, ␣
↪load average: 0.03, 0.04, 0.05                                                  ␣
↪                                          |
| linda-vm1.devops.develop     | GOOD   |  18:35:05 up 146 days,  8:31,  1 user, ␣
↪load average: 0.13, 0.04, 0.05                                                  ␣
↪                                          |
| floyd2-p.devops.develop      | GOOD   |  18:35:02 up 20 days, 59 min,  1 user, ␣
↪load average: 0.08, 0.04, 0.05                                                  ␣
↪                                          |
| jenkins-int.devops.develop   | GOOD   |  18:35:03 up 146 days,  8:31,  1 user, ␣
↪load average: 0.01, 0.02, 0.05                                                  ␣
↪                                          |
| lapp.devops.develop          | GOOD   |  18:35:02 up 146 days,  8:31,  1 user, ␣
↪load average: 0.16, 0.05, 0.06                                                  ␣
↪                                          |
| eclipse.devops.develop       | DARK   | SSH encountered an unknown error during the␣
↪connection. We recommend you re-run the command using -vvvv, which will enable SSH␣
↪debugging output to help diagnose the issue |
| lancaster.devops.develop     | DARK   | SSH encountered an unknown error during the␣
↪connection. We recommend you re-run the command using -vvvv, which will enable SSH␣
↪debugging output to help diagnose the issue |
+-----------------------------+--------+-----------------------------------------
↪---------------------------------------------------------------------------------
↪----------------------------------------+
```

Note: As can be seen here, the hosts `eclipse.devops.develop` and `lancaster.devops.develop` do not conform with the standard use of Ansible via SSH. These kind of *one-off* or manually-configured hosts limit the scalability and consistent use of Ansible as a system configuration and management tool.

12.6 Adding a Module in Another Repo

```
[dimsenv] dittrich@dimsdemo1:~/git/ansible-playbooks () $ cookiecutter https://git.
↪openstack.org/openstack-dev/cookiecutter.git
Cloning into 'cookiecutter'...
remote: Counting objects: 602, done.
remote: Compressing objects: 100% (265/265), done.
remote: Total 602 (delta 345), reused 563 (delta 310)
Receiving objects: 100% (602/602), 81.17 KiB | 0 bytes/s, done.
Resolving deltas: 100% (345/345), done.
Checking connectivity... done.
module_name [replace with the name of the python module]: dims_ansible_playbook
repo_group [openstack]: dims
repo_name [replace with the name for the git repo]: ansible-playbooks
launchpad_project [replace with the name of the project on launchpad]:
project_short_description [OpenStack Boilerplate contains all the boilerplate you␣
↪need to create an OpenStack package.]: Python ansible-playbook module for dimscli
Initialized empty Git repository in /home/dittrich/git/ansible-playbooks/ansible-
↪playbooks/.git/
[master (root-commit) 7d01bbe] Initial Cookiecutter Commit.
 26 files changed, 647 insertions(+)
 create mode 100644 .coveragerc
 create mode 100644 .gitignore
 create mode 100644 .gitreview
 create mode 100644 .mailmap
 create mode 100644 .testr.conf
 create mode 100644 CONTRIBUTING.rst
 create mode 100644 HACKING.rst
 create mode 100644 LICENSE
 create mode 100644 MANIFEST.in
 create mode 100644 README.rst
 create mode 100644 babel.cfg
 create mode 100644 dims_ansible_playbook/__init__.py
 create mode 100644 dims_ansible_playbook/tests/__init__.py
 create mode 100644 dims_ansible_playbook/tests/base.py
 create mode 100644 dims_ansible_playbook/tests/test_dims_ansible_playbook.py
 create mode 100755 doc/source/conf.py
 create mode 100644 doc/source/contributing.rst
 create mode 100644 doc/source/index.rst
 create mode 100644 doc/source/installation.rst
 create mode 100644 doc/source/readme.rst
 create mode 100644 doc/source/usage.rst
 create mode 100644 requirements.txt
 create mode 100644 setup.cfg
 create mode 100644 setup.py
 create mode 100644 test-requirements.txt
 create mode 100644 tox.ini
[dimsenv] dittrich@dimsdemo1:~/git/ansible-playbooks () $ ls -l
total 4
drwxrwxr-x 5 dittrich dittrich 4096 Jan  1 16:17 ansible-playbooks
```

CHAPTER 13

Service Discovery Using Consul

Consul provides many services that are used by DIMS components, including a key/value store and DNS service. DIMS takes advantage of the DNS service by having `dnsmasq` on each host direct certain queries to the Consul cluster for resolution, which can be used for service discovery (as opposed to hard-coding IP addresses or specific host names and port numbers in source code or configuration files.) The chapter *Developing modules for the DIMS CLI app (dimscli)* discusses some of the ways Consul is accessed by `dimscli` (e.g., see Section *Adding New Columns to Output*)

A program named `ianitor` (GitHub ClearcodeHQ/ianitor) facilitates using this Consul DNS capability by wrapping services so they are registered in Consul's DNS and monitored by Consul's health checking features. This would allow a monitoring application to notify someone when a DIMS service component (such as something in the backend data store) becomes unavailable.

Note: The `ianitor` package from PyPi is installed in the DIMS Python Virtual Environment, so it should be available on all DIMS components that would need it.

This registration and service discovery process be illustrated using the `netcat` (`nc`) program to create a listening process that will demonstrate how this works.

First, we start `nc` on a specific listening port

```
[dimsenv] dittrich@dimsdemo1:~ () $ ianitor --port 9999 netcat -- nc -l 9999
```

There is no output at this point, since `nc` is now running in the foreground (under the watch of `ianitor`, also running in the foreground) patiently listening on port `9999` for something to connect to it. You can prove to yourself that it is running by looking in the process tree:

```
 init(1)-+-ModemManager(1000)-+-{ModemManager}(1032)
         |                    `-{ModemManager}(1036)
         | ...
         |-lightdm(1662)-+-Xorg(1673)
         |               |-lightdm(1738)-+-init(2060)-+-GoogleTalkPlugi(3880)-+-
↪{GoogleTalkPlugi}(3881)
         |               |               |            | ...
```

```
        |                |               |             |-tmux(3066)-+-bash(4512)---
↪ianitor(680)---nc(683)
        |                |               |             | ...
        | ...
```

Now that the service is running, we can validate that `iainitor` has registered it in Consul. Figure *Consul Service Listing* shows Consul's view of **Services** showing `service:netcat` has been registered and is alive and healthy.

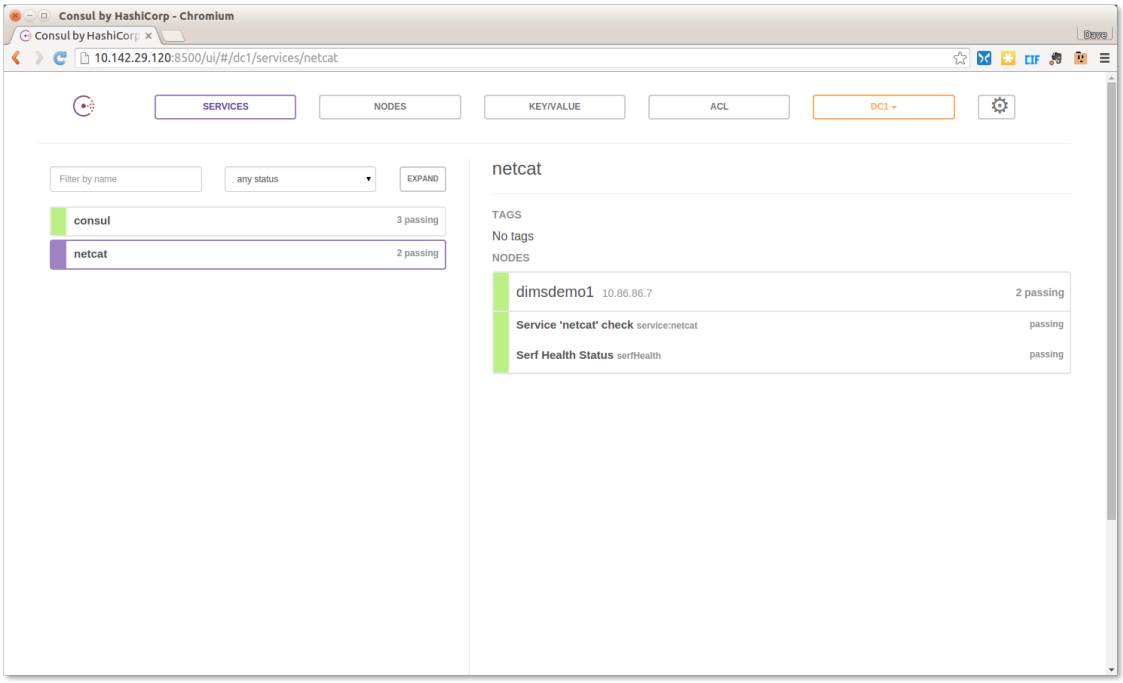


Fig. 13.1: Consul Service Listing

Using `dig`, the host on which this service was registered can be obtained by a simple **A** record lookup for `netcat.service.consul`, as seen here:

```
[dimsenv] dittrich@dimsdemo1:~ () $ dig netcat.service.consul

; <<>> DiG 9.9.5-3ubuntu0.7-Ubuntu <<>> netcat.service.consul
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 16448
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;netcat.service.consul.         IN      A

;; ANSWER SECTION:
netcat.service.consul.  0       IN      A       10.86.86.7

;; Query time: 26 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Sun Jan 24 12:19:58 PST 2016
;; MSG SIZE  rcvd: 76
```

Now switch to Consul's **Nodes** tab. Figure *Consul service registration for netcat* shows that node `dimsdemo1` is

running the service `netcat`, and this time the service port is also shown to the right ("`:9999`"):

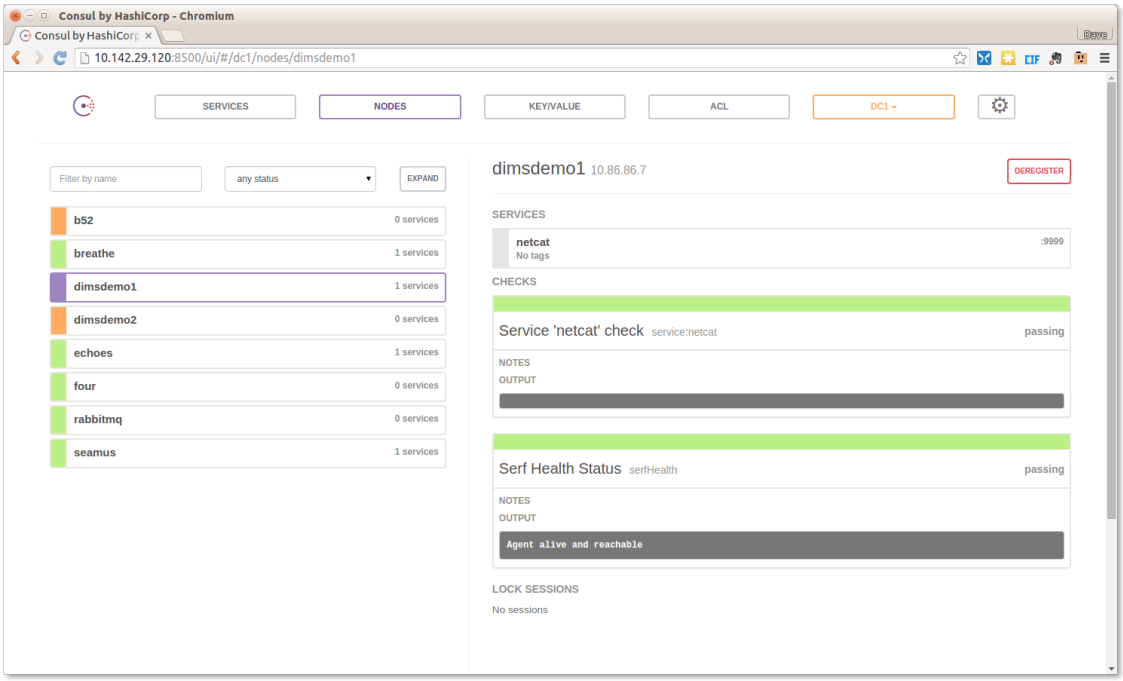


Fig. 13.2: Consul service registration for netcat

The service's port number can also be obtained from Consul via `dnsmasq` by asking for the DNS **SRV** record for `netcat.service.consul`:

```
[dimsenv] dittrich@dimsdemo1:~ () $ dig netcat.service.consul SRV

; <<>> DiG 9.9.5-3ubuntu0.7-Ubuntu <<>> netcat.service.consul SRV
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 8464
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; QUESTION SECTION:
;netcat.service.consul.          IN      SRV

;; ANSWER SECTION:
netcat.service.consul.  0       IN      SRV     1 1 9999 dimsdemo1.node.dc1.consul.

;; ADDITIONAL SECTION:
dimsdemo1.node.dc1.consul. 0    IN      A       10.86.86.7

;; Query time: 13 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Sun Jan 24 12:48:44 PST 2016
;; MSG SIZE  rcvd: 146
```

Now we can test connecting to the `netcat` listener (which will show anything that gets sent to it after the TCP connection is established.)

Attention: When attempting to duplicate this example, keep in mind that you must have already enabled `iptables` access to the port on which `nc` is listening, otherwise any connection attempt will be blocked and this won't work as shown here. **Always** keep `iptables` in mind when trying to expose network services and test them.

The first test will be using `curl` from the command line:

```
[dimsenv] dittrich@dimsdemo1:~ () $ curl --data Hello http://dimsdemo1.node.dc1.
↪consul:9999/areyouthere
```

Going back to the window where we ran `ianitor`, the result is the following:

```
[dimsenv] dittrich@dimsdemo1:~ () $ ianitor --port 9999 netcat -- netcat -l 9999
POST /areyouthere HTTP/1.1
User-Agent: curl/7.35.0
Host: dimsdemo1.node.dc1.consul:9999
Accept: */*
Content-Length: 5
Content-Type: application/x-www-form-urlencoded

Hello
```

Note: Because `netcat` simply listens on a port and then prints out what it receives (never sending anything back), both windows will hang. Just **CTRL-C** to kill them. This is just a proof-of-concept, not a real service. If you kill the `ianitor`/`nc` command first, the `curl` response will make this very clear with this message:

```
curl: (52) Empty reply from server
```

If you connect directly using `http://dimsdemo1.node.dc1.consul:9999` from a browser, you would get a slightly different result.

```
[dimsenv] dittrich@dimsdemo1:~ () $ ianitor --port 9999 netcat -- netcat -l 9999
GET / HTTP/1.1
Host: dimsdemo1.node.dc1.consul:9999
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:43.0) Gecko/20100101 Firefox/
↪43.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

In practice, `ianitor` would be used to wrap a service that is being started by some process manager, such as `supervisord`. See the Example supervisord config on the `ianitor` GitHub page.

CHAPTER 14

Docker Datacenter

This chapter documents a walk thru for running a development instance of Docker Universal Control Plane, part of Docker Datacenter.

Watch a UCP demo.

14.1 Datacenter Walk-thru

The following output walks thru these items:

- starting 3 VMs with Docker Machine
- installing UCP on one node as a controller; joining 2 other nodes
- setting up container networking on each node
- creating one overlay network
- starting a Consul container on each node

```
####################################
# DESTROY PREV DOCKER-MACHINE VMS: #
# 1. STOP NODES                    #
# 2. REMOVE NODES                  #
# 3. REMOVE VIRTUALBOX HOST-ONLY-  #
#    INTERFACE                     #
####################################

[dimsenv] mboggess@dimsdev2:~ () $ docker-machine ls
NAME    ACTIVE   DRIVER       STATE     URL                         SWARM   DOCKER    ␣
↪ERRORS
node1   -        virtualbox   Running   tcp://192.168.99.100:2376           v1.10.2
node2   -        virtualbox   Running   tcp://192.168.99.101:2376           v1.10.2
node3   -        virtualbox   Running   tcp://192.168.99.102:2376           v1.10.2
[dimsenv] mboggess@dimsdev2:~ () $ docker-machine stop node1
Stopping "node1"...
```

```
Machine "node1" was stopped.
[dimsenv] mboggess@dimsdev2:~ () $ docker-machine stop node2
Stopping "node2"...
Machine "node2" was stopped.
[dimsenv] mboggess@dimsdev2:~ () $ docker-machine stop node3
Stopping "node3"...
Machine "node3" was stopped.
[dimsenv] mboggess@dimsdev2:~ () $ docker-machine rm  node1
About to remove node1
Are you sure? (y/n): y
Successfully removed node1
[dimsenv] mboggess@dimsdev2:~ () $ docker-machine rm node2
About to remove node2
Are you sure? (y/n): y
Successfully removed node2
[dimsenv] mboggess@dimsdev2:~ () $ docker-machine rm node3
About to remove node3
Are you sure? (y/n): y
Successfully removed node3
[dimsenv] mboggess@dimsdev2:~ () $ docker-machine ls
NAME   ACTIVE   DRIVER   STATE   URL   SWARM   DOCKER   ERRORS
[dimsenv] mboggess@dimsdev2:~ () $ vboxmanage list hostonlyifs
Name:            vboxnet0
GUID:            786f6276-656e-4074-8000-0a0027000000
DHCP:            Disabled
IPAddress:       192.168.99.1
NetworkMask:     255.255.255.0
IPV6Address:     fe80:0000:0000:0000:0800:27ff:fe00:0000
IPV6NetworkMaskPrefixLength: 64
HardwareAddress: 0a:00:27:00:00:00
MediumType:      Ethernet
Status:          Up
VBoxNetworkName: HostInterfaceNetworking-vboxnet0

[dimsenv] mboggess@dimsdev2:~ () $ vboxmanage hostonlyif
Usage:

VBoxManage hostonlyif       ipconfig <name>
                            [--dhcp |
                            --ip<ipv4> [--netmask<ipv4> (def: 255.255.255.0)] |
                            --ipv6<ipv6> [--netmasklengthv6<length> (def: 64)]]
                            create |
                            remove <name>

[dimsenv] mboggess@dimsdev2:~ () $ vboxmanage hostonlyif remove vboxnet0
0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%
[dimsenv] mboggess@dimsdev2:~ () $ vboxmanage list hostonlyifs


#################################
# START NEW DOCKER MACHINE VMS: #
#################################

[dimsenv] mboggess@dimsdev2:~ () $ docker-machine create -d virtualbox \
> --virtualbox-memory "2000" \
> --virtualbox-disk-size "5000" node0
Running pre-create checks...
(node0) You are using version 4.3.28r100309 of VirtualBox. If you encounter issues,␣
↪you might want to upgrade to version 5 at https://www.virtualbox.org
```

```
Creating machine...
(node0) Copying /home/mboggess/.docker/machine/cache/boot2docker.iso to /home/
↪mboggess/.docker/machine/machines/node0/boot2docker.iso...
(node0) Creating VirtualBox VM...
(node0) Creating SSH key...
(node0) Starting the VM...
(node0) Check network to re-create if needed...
(node0) Found a new host-only adapter: "vboxnet0"
(node0) Waiting for an IP...
Waiting for machine to be running, this may take a few minutes...
Detecting operating system of created instance...
Waiting for SSH to be available...
Detecting the provisioner...
Provisioning with boot2docker...
Copying certs to the local machine directory...
Copying certs to the remote machine...
Setting Docker configuration on the remote daemon...
Checking connection to Docker...
Docker is up and running!
To see how to connect your Docker Client to the Docker Engine running on this virtual␣
↪machine, run: docker-machine env node0
[dimsenv] mboggess@dimsdev2:~ () $ docker-machine create -d virtualbox \
> --virtualbox-memory "2000" node1
Running pre-create checks...
(node1) You are using version 4.3.28r100309 of VirtualBox. If you encounter issues,␣
↪you might want to upgrade to version 5 at https://www.virtualbox.org
Creating machine...
(node1) Copying /home/mboggess/.docker/machine/cache/boot2docker.iso to /home/
↪mboggess/.docker/machine/machines/node1/boot2docker.iso...
(node1) Creating VirtualBox VM...
(node1) Creating SSH key...
(node1) Starting the VM...
(node1) Check network to re-create if needed...
(node1) Waiting for an IP...
Waiting for machine to be running, this may take a few minutes...
Detecting operating system of created instance...
Waiting for SSH to be available...
Detecting the provisioner...
Provisioning with boot2docker...
Copying certs to the local machine directory...
Copying certs to the remote machine...
Setting Docker configuration on the remote daemon...
Checking connection to Docker...
Docker is up and running!
To see how to connect your Docker Client to the Docker Engine running on this virtual␣
↪machine, run: docker-machine env node1
[dimsenv] mboggess@dimsdev2:~ () $ docker-machine create -d virtualbox --virtualbox-
↪memory "2000" node2
Running pre-create checks...
(node2) You are using version 4.3.28r100309 of VirtualBox. If you encounter issues,␣
↪you might want to upgrade to version 5 at https://www.virtualbox.org
Creating machine...
(node2) Copying /home/mboggess/.docker/machine/cache/boot2docker.iso to /home/
↪mboggess/.docker/machine/machines/node2/boot2docker.iso...
(node2) Creating VirtualBox VM...
(node2) Creating SSH key...
(node2) Starting the VM...
(node2) Check network to re-create if needed...
```

```
(node2) Waiting for an IP...
Waiting for machine to be running, this may take a few minutes...
Detecting operating system of created instance...
Waiting for SSH to be available...
Detecting the provisioner...
Provisioning with boot2docker...
Copying certs to the local machine directory...
Copying certs to the remote machine...
Setting Docker configuration on the remote daemon...
Checking connection to Docker...
Docker is up and running!
To see how to connect your Docker Client to the Docker Engine running on this virtual␣
↪machine, run: docker-machine env node2
[dimsenv] mboggess@dimsdev2:~ () $ docker-machine ls
NAME    ACTIVE   DRIVER       STATE     URL                         SWARM   DOCKER    ␣
↪ERRORS
node0   -        virtualbox   Running   tcp://192.168.99.100:2376           v1.10.2
node1   -        virtualbox   Running   tcp://192.168.99.101:2376           v1.10.2
node2   -        virtualbox   Running   tcp://192.168.99.102:2376           v1.10.2


##############################
# INSTALL UCP ON CONTROLLER: #
##############################

# The following command is important.
# It sets up the shell environment to interact
#  with the Docker engine on that node.
[dimsenv] mboggess@dimsdev2:~ () $ eval $(docker-machine env node0)
[dimsenv] mboggess@dimsdev2:~ () $ docker run --rm -it \
> -v /var/run/docker.sock:/var/run/docker.sock \
> --name ucp docker/ucp install -i \
> --swarm-port 3376 --host-address $(docker-machine ip node0)
Unable to find image 'docker/ucp:latest' locally
latest: Pulling from docker/ucp
9ba637b863b8: Pull complete
a3ed95caeb02: Pull complete
Digest: sha256:1016db92f68ef6f9b58053e10eec8465efc7344a3f1e4cbb8fc3e446c42e89d4
Status: Downloaded newer image for docker/ucp:latest
INFO[0000] Verifying your system is compatible with UCP

# Remember your password. The username is "admin".
Please choose your initial Orca admin password:
Confirm your initial password:
INFO[0016] Pulling required images... (this may take a while)
WARN[0044] None of the hostnames we'll be using in the UCP certificates [node0 127.0.␣
↪0.1 172.17.0.1 192.168.99.100] contain a domain component.  Your generated certs␣
↪may fail TLS validation unless you only use one of these shortnames or IPs to␣
↪connect.  You can use the --san flag to add more aliases
You may enter additional aliases (SANs) now or press enter to proceed with the above␣
↪list.
Additional aliases:
INFO[0047] Installing UCP with host address 192.168.99.100 - If this is incorrect,␣
↪please specify an alternative address with the '--host-address' flag
INFO[0000] Generating UCP Cluster Root CA
INFO[0018] Generating UCP Client Root CA
INFO[0021] Deploying UCP Containers
INFO[0026] UCP instance ID:␣
↪3RSF:3X4K:YW6L:I6E2:XFT3:3INP:JHBF:AFEB:PUXE:FNDU:SELW:43GE
```

```
INFO[0026] UCP Server SSL: SHA1␣
↪Fingerprint=13:C0:19:5A:98:92:34:18:9A:CB:3E:4F:EB:A3:0E:D6:E3:8E:4E:C6
INFO[0026] Login as "admin"/(your admin password) to UCP at https://192.168.99.100:443

#######################################
# JOIN OTHER NODES TO UCP CONTROLLER: #
#######################################

# Make sure to run the eval command for each node
#  before the docker run command!
[dimsenv] mboggess@dimsdev2:~ () $ eval $(docker-machine env node1)
[dimsenv] mboggess@dimsdev2:~ () $ docker run --rm -it -v /var/run/docker.sock:/var/
↪run/docker.sock --name ucp docker/ucp join -i --host-address $(docker-machine ip␣
↪node1)
Please enter the URL to your UCP server: https://192.168.99.100
UCP server https://192.168.99.100
Subject: ucp
Issuer: UCP Client Root CA
SHA1 Fingerprint=13:C0:19:5A:98:92:34:18:9A:CB:3E:4F:EB:A3:0E:D6:E3:8E:4E:C6
Do you want to trust this server and proceed with the join? (y/n): y
Please enter your UCP Admin username: admin
Please enter your UCP Admin password:
INFO[0017] Pulling required images... (this may take a while)
WARN[0048] None of the hostnames we'll be using in the UCP certificates [node1 127.0.
↪0.1 172.17.0.1 192.168.99.101] contain a domain component.  Your generated certs␣
↪may fail TLS validation unless you only use one of these shortnames or IPs to␣
↪connect.  You can use the --san flag to add more aliases
You may enter additional aliases (SANs) now or press enter to proceed with the above␣
↪list.
Additional aliases:
INFO[0000] This engine will join UCP and advertise itself with host address 192.168.
↪99.101 - If this is incorrect, please specify an alternative address with the '--
↪host-address' flag
INFO[0000] Verifying your system is compatible with UCP
INFO[0006] Starting local swarm containers
[dimsenv] mboggess@dimsdev2:~ () $ eval $(docker-machine env node2)
[dimsenv] mboggess@dimsdev2:~ () $ docker run --rm -it -v /var/run/docker.sock:/var/
↪run/docker.sock --name ucp docker/ucp join -i --host-address $(docker-machine ip␣
↪node2)
Please enter the URL to your UCP server: https://192.168.99.100
UCP server https://192.168.99.100
Subject: ucp
Issuer: UCP Client Root CA
SHA1 Fingerprint=13:C0:19:5A:98:92:34:18:9A:CB:3E:4F:EB:A3:0E:D6:E3:8E:4E:C6
Do you want to trust this server and proceed with the join? (y/n): y
Please enter your UCP Admin username: admin
Please enter your UCP Admin password:
INFO[0013] Pulling required images... (this may take a while)
WARN[0043] None of the hostnames we'll be using in the UCP certificates [node2 127.0.
↪0.1 172.17.0.1 192.168.99.102] contain a domain component.  Your generated certs␣
↪may fail TLS validation unless you only use one of these shortnames or IPs to␣
↪connect.  You can use the --san flag to add more aliases
You may enter additional aliases (SANs) now or press enter to proceed with the above␣
↪list.
Additional aliases:
INFO[0000] This engine will join UCP and advertise itself with host address 192.168.
↪99.102 - If this is incorrect, please specify an alternative address with the '--
↪host-address' flag
```

```
INFO[0000] Verifying your system is compatible with UCP
INFO[0003] Starting local swarm containers


################################
# SETUP CONTAINER NETWORKING: #
################################

[dimsenv] mboggess@dimsdev2:~ () $ eval $(docker-machine env node0)
[dimsenv] mboggess@dimsdev2:~ () $ docker-machine ssh node0
                        ##         .
                  ## ## ##        ==
               ## ## ## ## ##    ===
           /"""""""""""""""""\___/ ===
      ~~~ {~~ ~~~~ ~~~ ~~~~ ~~~ ~ /  ===- ~~~
           \_______ o           __/
             \    \         __/
              \____\_______/
 _                 _   _____      _             _
| |__   ___   ___ | |_|___ \ __| | ___   ___| | _____ _ __
| '_ \ / _ \ / _ \| __| __) / _` |/ _ \ / __| |/ / _ \ '__|
| |_) | (_) | (_) | |_ / __/ (_| | (_) | (__|   <  __/ |
|_.__/ \___/ \___/ \__|_____\__,_|\___/ \___|_|\_\___|_|
Boot2Docker version 1.10.2, build master : 611be10 - Mon Feb 22 22:47:06 UTC 2016
Docker version 1.10.2, build c3959b1

# Run the engine-discovery command using the
#  --update flag to force the update
docker@node0:~$ docker run --rm -it --name ucp \
> -v /var/run/docker.sock:/var/run/docker.sock \
> docker/ucp engine-discovery \
> --controller 192.168.99.100 \
> --host-address 192.168.99.100 --update
INFO[0000] This cluster does not have high availability because you only have a␣
↪single controller.
INFO[0000] If you plan to add additional replica nodes for HA, you should do this␣
↪before running this tool to reduce downtime.
WARN[0002] Configuration updated.  You will have to manually restart the docker␣
↪daemon for the changes to take effect.

# Restart docker. Since this is a boot2docker node,
#  it's via init.d rather than the service command.
docker@node0:~$ sudo /etc/init.d/docker restart
Need TLS certs for node0,127.0.0.1,10.0.2.15,192.168.99.100
-------------------

# Check to see if docker is running.
# If this is the first or second nodes, the etcd cluster
#  will complain because it hasn't achieved a quorum yet.
docker@node0:~$ sudo tail -f /var/log/docker.log
time="2016-03-07T22:11:44.070183580Z" level=debug msg="/usr/local/sbin/iptables, [--
↪wait -t nat -A POSTROUTING -p tcp -s 172.17.0.6 -d 172.17.0.6 --dport 12382 -j␣
↪MASQUERADE]"
time="2016-03-07T22:11:44.093786359Z" level=debug msg="Assigning addresses for␣
↪endpoint ucp-client-root-ca's interface on network bridge"
time="2016-03-07T22:11:44.094136165Z" level=debug msg="/usr/local/sbin/iptables, [--
↪wait -t nat -A DOCKER -p tcp -d 0/0 --dport 12376 -j DNAT --to-destination 172.17.0.
↪4:2376 ! -i docker0]"
```

```
time="2016-03-07T22:11:44.097042994Z" level=debug msg="/usr/local/sbin/iptables, [--
↪wait -t filter -A DOCKER ! -i docker0 -o docker0 -p tcp -d 172.17.0.4 --dport 2376 -
↪j ACCEPT]"
time="2016-03-07T22:11:44.097919349Z" level=debug msg="/usr/local/sbin/iptables, [--
↪wait -t nat -A POSTROUTING -p tcp -s 172.17.0.4 -d 172.17.0.4 --dport 2376 -j␣
↪MASQUERADE]"
time="2016-03-07T22:11:44.120796586Z" level=debug msg="Assigning addresses for␣
↪endpoint ucp-proxy's interface on network bridge"
time="2016-03-07T22:11:47.002871426Z" level=debug msg="could not find network␣
↪1faf51793e49dbfb42520d09598c45431e0a4791ca280866d2d451965ef88b05: client: etcd␣
↪cluster is unavailable or misconfigured"
time="2016-03-07T22:11:50.003202887Z" level=debug msg="could not find endpoint␣
↪b605cdd78addde5cfde07aa525d67ce9855fc106b8477a6effdf1cc22a185254 in global: client:␣
↪etcd cluster is unavailable or misconfigured"
time="2016-03-07T22:11:53.005044110Z" level=debug msg="could not find network␣
↪1faf51793e49dbfb42520d09598c45431e0a4791ca280866d2d451965ef88b05: client: etcd␣
↪cluster is unavailable or misconfigured"
time="2016-03-07T22:11:55.240581226Z" level=debug msg="could not find endpoint␣
↪8e2956581574300e02f4efb367eb88e042f82a80985ed50f282f79d0529b0e23 in global: client:␣
↪etcd cluster is unavailable or misconfigured"
time="2016-03-07T22:11:58.240348551Z" level=debug msg="failed to get endpoints for␣
↪network bridge scope global: client: etcd cluster is unavailable or misconfigured"
time="2016-03-07T22:12:01.240514251Z" level=debug msg="could not find network␣
↪1faf51793e49dbfb42520d09598c45431e0a4791ca280866d2d451965ef88b05: client: etcd␣
↪cluster is unavailable or misconfigured"
time="2016-03-07T22:12:04.240443463Z" level=error msg="discovery error: client: etcd␣
↪cluster is unavailable or misconfigured"
time="2016-03-07T22:12:04.240519734Z" level=warning msg="Registering as \"192.168.99.
↪100:12376\" in discovery failed: client: etcd cluster is unavailable or␣
↪misconfigured"
time="2016-03-07T22:12:04.240552300Z" level=debug msg="could not find endpoint␣
↪87e7556fe7ffe84c25abe1a0838f51ef54389eb6c87468f8431eec495fcf4c5c in global: client:␣
↪etcd cluster is unavailable or misconfigured"
time="2016-03-07T22:12:07.240511596Z" level=debug msg="failed to get endpoints for␣
↪network bridge scope global: client: etcd cluster is unavailable or misconfigured"
time="2016-03-07T22:12:07.241604653Z" level=error msg="discovery error: client: etcd␣
↪cluster is unavailable or misconfigured"
time="2016-03-07T22:12:10.240455936Z" level=error msg="discovery error: Unexpected␣
↪watch error"
time="2016-03-07T22:12:10.240512021Z" level=debug msg="could not find network␣
↪1faf51793e49dbfb42520d09598c45431e0a4791ca280866d2d451965ef88b05: client: etcd␣
↪cluster is unavailable or misconfigured"
time="2016-03-07T22:12:13.240302992Z" level=debug msg="could not find endpoint␣
↪95bb73a58f52adcf1ca98ba9a1298eb6447684b5716a59c4b43481b13795bcc2 in global: client:␣
↪etcd cluster is unavailable or misconfigured"
time="2016-03-07T22:12:16.240504890Z" level=debug msg="failed to get endpoints for␣
↪network bridge scope global: client: etcd cluster is unavailable or misconfigured"
time="2016-03-07T22:12:19.240514865Z" level=debug msg="could not find network␣
↪1faf51793e49dbfb42520d09598c45431e0a4791ca280866d2d451965ef88b05: client: etcd␣
↪cluster is unavailable or misconfigured"
time="2016-03-07T22:12:19.240580469Z" level=debug msg="Assigning addresses for␣
↪endpoint ucp-cluster-root-ca's interface on network bridge"
time="2016-03-07T22:12:19.240598629Z" level=debug msg="RequestAddress(LocalDefault/
↪172.17.0.0/16, <nil>, map[])"
time="2016-03-07T22:12:19.243498664Z" level=debug msg="/usr/local/sbin/iptables, [--
↪wait -t nat -A DOCKER -p tcp -d 0/0 --dport 12381 -j DNAT --to-destination 172.17.0.
↪8:12381 ! -i docker0]"
time="2016-03-07T22:12:19.246101114Z" level=debug msg="/usr/local/sbin/iptables, [--
↪wait -t filter -A DOCKER ! -i docker0 -o docker0 -p tcp -d 172.17.0.8 --dport 12381␣
↪-j ACCEPT]"
```

```
time="2016-03-07T22:12:19.248576494Z" level=debug msg="/usr/local/sbin/iptables, [--
↪wait -t nat -A POSTROUTING -p tcp -s 172.17.0.8 -d 172.17.0.8 --dport 12381 -j␣
↪MASQUERADE]"
time="2016-03-07T22:12:19.276450231Z" level=debug msg="Assigning addresses for␣
↪endpoint ucp-cluster-root-ca's interface on network bridge"
time="2016-03-07T22:12:22.240591811Z" level=debug msg="could not find network␣
↪1faf51793e49dbfb42520d09598c45431e0a4791ca280866d2d451965ef88b05: client: etcd␣
↪cluster is unavailable or misconfigured"
time="2016-03-07T22:12:25.240425959Z" level=warning msg="Registering as \"192.168.99.
↪100:12376\" in discovery failed: client: etcd cluster is unavailable or␣
↪misconfigured"
time="2016-03-07T22:12:25.240490015Z" level=debug msg="could not find endpoint␣
↪b605cdd78addde5cfde07aa525d67ce9855fc106b8477a6effdf1cc22a185254 in global: client:␣
↪etcd cluster is unavailable or misconfigured"
time="2016-03-07T22:12:28.240338660Z" level=debug msg="could not find network␣
↪1faf51793e49dbfb42520d09598c45431e0a4791ca280866d2d451965ef88b05: client: etcd␣
↪cluster is unavailable or misconfigured"
time="2016-03-07T22:12:31.240448754Z" level=error msg="discovery error: client: etcd␣
↪cluster is unavailable or misconfigured"
time="2016-03-07T22:12:31.240505330Z" level=debug msg="could not find network␣
↪1faf51793e49dbfb42520d09598c45431e0a4791ca280866d2d451965ef88b05: client: etcd␣
↪cluster is unavailable or misconfigured"
^C
docker@node0:~$ exit
[dimsenv] mboggess@dimsdev2:~ () $ docker-machine ssh node1
                        ##         .
                  ## ## ##        ==
               ## ## ## ## ##    ===
           /"""""""""""""""""\___/ ===
      ~~~ {~~ ~~~~ ~~~ ~~~~ ~~~ ~ /  ===- ~~~
           \_______ o           __/
             \     \         __/
              \____\_______/
 _                 _   ____     _            _
| |__   ___   ___ | |_|___ \ __| | ___   ___| | _____ _ __
| '_ \ / _ \ / _ \| __| __) / _` |/ _ \ / __| |/ / _ \ '__|
| |_) | (_) | (_) | |_ / __/ (_| | (_) | (__|   <  __/ |
|_.__/ \___/ \___/ \__|_____\__,_|\___/ \___|_|\_\___|_|
Boot2Docker version 1.10.2, build master : 611be10 - Mon Feb 22 22:47:06 UTC 2016
Docker version 1.10.2, build c3959b1

# Run the engine-discovery command using the
#  --update flag to force the update
docker@node1:~$ docker run --rm -it --name ucp \
> -v /var/run/docker.sock:/var/run/docker.sock \
> docker/ucp engine-discovery \
> --controller 192.168.99.100 \
> --host-address 192.168.99.101 --update
INFO[0000] This cluster does not have high availability because you only have a␣
↪single controller.
INFO[0000] If you plan to add additional replica nodes for HA, you should do this␣
↪before running this tool to reduce downtime.
WARN[0002] Configuration updated.  You will have to manually restart the docker␣
↪daemon for the changes to take effect.

# Restart docker. Since this is a boot2docker node,
#  it's via init.d rather than the service command.
docker@node1:~$ sudo /etc/init.d/docker restart
```

```
Need TLS certs for node1,127.0.0.1,10.0.2.15,192.168.99.101
-------------------

# Check to see if docker is running.
# If this is the first or second nodes, the etcd cluster
#  will complain because it hasn't achieved a quorum yet.
docker@node1:~$ sudo tail -f /var/log/docker.log
time="2016-03-07T22:13:46.271986099Z" level=debug msg="/usr/local/sbin/iptables, [--
↪wait -t nat -n -L DOCKER]"
time="2016-03-07T22:13:46.272800845Z" level=debug msg="/usr/local/sbin/iptables, [--
↪wait -t nat -N DOCKER]"
time="2016-03-07T22:13:46.273586600Z" level=debug msg="/usr/local/sbin/iptables, [--
↪wait -t filter -n -L DOCKER]"
time="2016-03-07T22:13:46.274353619Z" level=debug msg="/usr/local/sbin/iptables, [--
↪wait -t filter -n -L DOCKER-ISOLATION]"
time="2016-03-07T22:13:46.275113441Z" level=debug msg="/usr/local/sbin/iptables, [--
↪wait -t filter -C DOCKER-ISOLATION -j RETURN]"
time="2016-03-07T22:13:46.276694579Z" level=debug msg="/usr/local/sbin/iptables, [--
↪wait -I DOCKER-ISOLATION -j RETURN]"
time="2016-03-07T22:13:49.257937305Z" level=debug msg="Registering ipam driver: \
↪"default\""
time="2016-03-07T22:13:49.258517657Z" level=error msg="discovery error: client: etcd␣
↪cluster is unavailable or misconfigured"
time="2016-03-07T22:13:52.258733627Z" level=error msg="discovery error: client: etcd␣
↪cluster is unavailable or misconfigured"
time="2016-03-07T22:13:52.259818156Z" level=debug msg="failed to get endpoints for␣
↪network bridge scope global: client: etcd cluster is unavailable or misconfigured"
time="2016-03-07T22:13:55.257366820Z" level=debug msg="failed to get networks for␣
↪scope global: client: etcd cluster is unavailable or misconfigured"
time="2016-03-07T22:13:55.257706784Z" level=error msg="discovery error: Unexpected␣
↪watch error"
time="2016-03-07T22:13:55.258525392Z" level=debug msg="releasing IPv4 pools from␣
↪network bridge (1000394495e2c6bd577004f17a75ed038c7f1696448275552fd1d7219e38f02e)"
time="2016-03-07T22:13:55.258554539Z" level=debug msg="ReleaseAddress(LocalDefault/
↪172.17.0.0/16, 172.17.0.1)"
time="2016-03-07T22:13:55.258957934Z" level=debug msg="ReleasePool(LocalDefault/172.
↪17.0.0/16)"
time="2016-03-07T22:13:55.260916658Z" level=info msg="Default bridge (docker0) is␣
↪assigned with an IP address 172.17.0.0/16. Daemon option --bip can be used to set a␣
↪preferred IP address"
time="2016-03-07T22:13:55.260932331Z" level=debug msg="Allocating IPv4 pools for␣
↪network bridge (b88f0323c65d45add2d7b24c36a9206d006b420c6030273192d24721948d0e02)"
time="2016-03-07T22:13:55.260946821Z" level=debug msg="RequestPool(LocalDefault, 172.
↪17.0.0/16, , map[], false)"
time="2016-03-07T22:13:55.263397489Z" level=debug msg="RequestAddress(LocalDefault/
↪172.17.0.0/16, 172.17.0.1, map[RequestAddressType:com.docker.network.gateway])"
time="2016-03-07T22:13:55.263939490Z" level=debug msg="/usr/local/sbin/iptables, [--
↪wait -t nat -C POSTROUTING -s 172.17.0.0/16 ! -o docker0 -j MASQUERADE]"
time="2016-03-07T22:13:55.265768161Z" level=debug msg="/usr/local/sbin/iptables, [--
↪wait -t nat -C DOCKER -i docker0 -j RETURN]"
time="2016-03-07T22:13:55.269104645Z" level=debug msg="/usr/local/sbin/iptables, [--
↪wait -t nat -I DOCKER -i docker0 -j RETURN]"
time="2016-03-07T22:13:55.279625706Z" level=debug msg="/usr/local/sbin/iptables, [--
↪wait -D FORWARD -i docker0 -o docker0 -j DROP]"
time="2016-03-07T22:13:55.281677039Z" level=debug msg="/usr/local/sbin/iptables, [--
↪wait -t filter -C FORWARD -i docker0 -o docker0 -j ACCEPT]"
time="2016-03-07T22:13:55.283548456Z" level=debug msg="/usr/local/sbin/iptables, [--
↪wait -t filter -C FORWARD -i docker0 ! -o docker0 -j ACCEPT]"
```

```
time="2016-03-07T22:13:55.285372426Z" level=debug msg="/usr/local/sbin/iptables, [--
↪wait -t filter -C FORWARD -o docker0 -m conntrack --ctstate RELATED,ESTABLISHED -j␣
↪ACCEPT]"
time="2016-03-07T22:13:55.286692461Z" level=debug msg="/usr/local/sbin/iptables, [--
↪wait -t nat -C PREROUTING -m addrtype --dst-type LOCAL -j DOCKER]"
time="2016-03-07T22:13:55.288968983Z" level=debug msg="/usr/local/sbin/iptables, [--
↪wait -t nat -A PREROUTING -m addrtype --dst-type LOCAL -j DOCKER]"
time="2016-03-07T22:13:55.292775137Z" level=debug msg="/usr/local/sbin/iptables, [--
↪wait -t nat -C OUTPUT -m addrtype --dst-type LOCAL -j DOCKER ! --dst 127.0.0.0/8]"
time="2016-03-07T22:13:55.295829670Z" level=debug msg="/usr/local/sbin/iptables, [--
↪wait -t nat -A OUTPUT -m addrtype --dst-type LOCAL -j DOCKER ! --dst 127.0.0.0/8]"
time="2016-03-07T22:13:55.297141841Z" level=debug msg="/usr/local/sbin/iptables, [--
↪wait -t filter -C FORWARD -o docker0 -j DOCKER]"
time="2016-03-07T22:13:55.298264112Z" level=debug msg="/usr/local/sbin/iptables, [--
↪wait -t filter -C FORWARD -o docker0 -j DOCKER]"
time="2016-03-07T22:13:55.299305220Z" level=debug msg="/usr/local/sbin/iptables, [--
↪wait -t filter -C FORWARD -j DOCKER-ISOLATION]"
time="2016-03-07T22:13:55.301557053Z" level=debug msg="/usr/local/sbin/iptables, [--
↪wait -D FORWARD -j DOCKER-ISOLATION]"
time="2016-03-07T22:13:55.302973335Z" level=debug msg="/usr/local/sbin/iptables, [--
↪wait -I FORWARD -j DOCKER-ISOLATION]"
time="2016-03-07T22:13:55.306115140Z" level=warning msg="Your kernel does not support␣
↪cgroup blkio weight"
time="2016-03-07T22:13:55.306137240Z" level=warning msg="Your kernel does not support␣
↪cgroup blkio weight_device"
time="2016-03-07T22:13:55.306368634Z" level=debug msg="Cleaning up old shm/mqueue␣
↪mounts: start."
time="2016-03-07T22:13:55.306436591Z" level=debug msg="Cleaning up old shm/mqueue␣
↪mounts: done."
time="2016-03-07T22:13:55.306847988Z" level=debug msg="Loaded container␣
↪21d865de28e79c10a80125d1fb13d72f3e6b3d44accf8264b3e285d3b5ddbe2d"
time="2016-03-07T22:13:55.307060932Z" level=debug msg="Loaded container␣
↪d5d7a3b2009769571f35029711c2564b7ee6231cbadfa858c9bad471a933f676"
time="2016-03-07T22:13:55.307357391Z" level=debug msg="Starting container␣
↪d5d7a3b2009769571f35029711c2564b7ee6231cbadfa858c9bad471a933f676"
time="2016-03-07T22:13:55.309267051Z" level=debug msg="Starting container␣
↪21d865de28e79c10a80125d1fb13d72f3e6b3d44accf8264b3e285d3b5ddbe2d"
time="2016-03-07T22:13:55.310005643Z" level=debug msg="container mounted via␣
↪layerStore: /mnt/sda1/var/lib/docker/aufs/mnt/
↪b8c1dc4410caf9f4fab300b45a9f0fbff089d70294540060f76f1f2f9f572ac1"
time="2016-03-07T22:13:55.310584317Z" level=debug msg="container mounted via␣
↪layerStore: /mnt/sda1/var/lib/docker/aufs/mnt/
↪129e034a6b4e86ec6f8f294a1d900575fd811ac461509a392543bcf63a81e269"
time="2016-03-07T22:13:58.257050034Z" level=debug msg="failed to get networks for␣
↪scope global: client: etcd cluster is unavailable or misconfigured"
time="2016-03-07T22:14:01.256474377Z" level=debug msg="failed to get networks for␣
↪scope global: client: etcd cluster is unavailable or misconfigured"
time="2016-03-07T22:14:04.256797930Z" level=debug msg="failed to get endpoints for␣
↪network bridge scope global: client: etcd cluster is unavailable or misconfigured"
time="2016-03-07T22:14:04.257003090Z" level=debug msg="Assigning addresses for␣
↪endpoint ucp-proxy's interface on network bridge"
time="2016-03-07T22:14:04.257028352Z" level=debug msg="RequestAddress(LocalDefault/
↪172.17.0.0/16, <nil>, map[])"
time="2016-03-07T22:14:04.259816002Z" level=debug msg="/usr/local/sbin/iptables, [--
↪wait -t nat -A DOCKER -p tcp -d 0/0 --dport 12376 -j DNAT --to-destination 172.17.0.
↪2:2376 ! -i docker0]"
time="2016-03-07T22:14:04.263040050Z" level=debug msg="/usr/local/sbin/iptables, [--
↪wait -t filter -A DOCKER ! -i docker0 -o docker0 -p tcp -d 172.17.0.2 --dport 2376 -
↪j ACCEPT]"
```

```
time="2016-03-07T22:14:04.265142604Z" level=debug msg="/usr/local/sbin/iptables, [--
↪wait -t nat -A POSTROUTING -p tcp -s 172.17.0.2 -d 172.17.0.2 --dport 2376 -j␣
↪MASQUERADE]"
time="2016-03-07T22:14:04.292801112Z" level=debug msg="Assigning addresses for␣
↪endpoint ucp-proxy's interface on network bridge"
time="2016-03-07T22:14:07.256722098Z" level=debug msg="failed to get endpoints for␣
↪network bridge scope global: client: etcd cluster is unavailable or misconfigured"
time="2016-03-07T22:14:07.256809339Z" level=debug msg="Assigning addresses for␣
↪endpoint ucp-swarm-join's interface on network bridge"
time="2016-03-07T22:14:07.256832041Z" level=debug msg="RequestAddress(LocalDefault/
↪172.17.0.0/16, <nil>, map[])"
time="2016-03-07T22:14:07.259553952Z" level=debug msg="Assigning addresses for␣
↪endpoint ucp-swarm-join's interface on network bridge"
time="2016-03-07T22:14:08.636576527Z" level=warning msg="Registering as \"192.168.99.
↪101:12376\" in discovery failed: client: etcd cluster is unavailable or␣
↪misconfigured"
time="2016-03-07T22:14:08.845887701Z" level=info msg="Daemon has completed␣
↪initialization"
time="2016-03-07T22:14:08.845913783Z" level=info msg="Docker daemon" commit=c3959b1␣
↪execdriver=native-0.2 graphdriver=aufs version=1.10.2
time="2016-03-07T22:14:08.846256409Z" level=debug msg="Registering routers"
time="2016-03-07T22:14:08.846270306Z" level=debug msg="Registering HEAD, /containers/
↪{name:.*}/archive"
time="2016-03-07T22:14:08.847061019Z" level=debug msg="Registering GET, /containers/
↪json"
time="2016-03-07T22:14:08.847153618Z" level=debug msg="Registering GET, /containers/
↪{name:.*}/export"
time="2016-03-07T22:14:08.847238114Z" level=debug msg="Registering GET, /containers/
↪{name:.*}/changes"
time="2016-03-07T22:14:08.847326718Z" level=debug msg="Registering GET, /containers/
↪{name:.*}/json"
time="2016-03-07T22:14:08.847399636Z" level=debug msg="Registering GET, /containers/
↪{name:.*}/top"
time="2016-03-07T22:14:08.847474180Z" level=debug msg="Registering GET, /containers/
↪{name:.*}/logs"
time="2016-03-07T22:14:08.847546361Z" level=debug msg="Registering GET, /containers/
↪{name:.*}/stats"
time="2016-03-07T22:14:08.847617874Z" level=debug msg="Registering GET, /containers/
↪{name:.*}/attach/ws"
time="2016-03-07T22:14:08.847760982Z" level=debug msg="Registering GET, /exec/{id:.*}/
↪json"
time="2016-03-07T22:14:08.847832158Z" level=debug msg="Registering GET, /containers/
↪{name:.*}/archive"
time="2016-03-07T22:14:08.847912683Z" level=debug msg="Registering POST, /containers/
↪create"
time="2016-03-07T22:14:08.847974072Z" level=debug msg="Registering POST, /containers/
↪{name:.*}/kill"
time="2016-03-07T22:14:08.848047654Z" level=debug msg="Registering POST, /containers/
↪{name:.*}/pause"
time="2016-03-07T22:14:08.848126808Z" level=debug msg="Registering POST, /containers/
↪{name:.*}/unpause"
time="2016-03-07T22:14:08.848204817Z" level=debug msg="Registering POST, /containers/
↪{name:.*}/restart"
time="2016-03-07T22:14:08.848278681Z" level=debug msg="Registering POST, /containers/
↪{name:.*}/start"
time="2016-03-07T22:14:08.848348615Z" level=debug msg="Registering POST, /containers/
↪{name:.*}/stop"
time="2016-03-07T22:14:08.848420618Z" level=debug msg="Registering POST, /containers/
↪{name:.*}/wait"
```

```
time="2016-03-07T22:14:08.848491251Z" level=debug msg="Registering POST, /containers/
↪{name:.*}/resize"
time="2016-03-07T22:14:08.848590568Z" level=debug msg="Registering POST, /containers/
↪{name:.*}/attach"
time="2016-03-07T22:14:08.848669165Z" level=debug msg="Registering POST, /containers/
↪{name:.*}/copy"
time="2016-03-07T22:14:08.848742896Z" level=debug msg="Registering POST, /containers/
↪{name:.*}/exec"
time="2016-03-07T22:14:08.848819642Z" level=debug msg="Registering POST, /exec/{name:.
↪*}/start"
time="2016-03-07T22:14:08.848889548Z" level=debug msg="Registering POST, /exec/{name:.
↪*}/resize"
time="2016-03-07T22:14:08.848958077Z" level=debug msg="Registering POST, /containers/
↪{name:.*}/rename"
time="2016-03-07T22:14:08.849037834Z" level=debug msg="Registering POST, /containers/
↪{name:.*}/update"
time="2016-03-07T22:14:08.849109277Z" level=debug msg="Registering PUT, /containers/
↪{name:.*}/archive"
time="2016-03-07T22:14:08.849190550Z" level=debug msg="Registering DELETE, /
↪containers/{name:.*}"
time="2016-03-07T22:14:08.849216845Z" level=debug msg="Registering GET, /images/json"
time="2016-03-07T22:14:08.849318873Z" level=debug msg="Registering GET, /images/search
↪"
time="2016-03-07T22:14:08.849373401Z" level=debug msg="Registering GET, /images/get"
time="2016-03-07T22:14:08.849428402Z" level=debug msg="Registering GET, /images/
↪{name:.*}/get"
time="2016-03-07T22:14:08.849509958Z" level=debug msg="Registering GET, /images/
↪{name:.*}/history"
time="2016-03-07T22:14:08.849580238Z" level=debug msg="Registering GET, /images/
↪{name:.*}/json"
time="2016-03-07T22:14:08.849658289Z" level=debug msg="Registering POST, /commit"
time="2016-03-07T22:14:08.849710575Z" level=debug msg="Registering POST, /images/
↪create"
time="2016-03-07T22:14:08.849765617Z" level=debug msg="Registering POST, /images/load"
time="2016-03-07T22:14:08.849832560Z" level=debug msg="Registering POST, /images/
↪{name:.*}/push"
time="2016-03-07T22:14:08.849910171Z" level=debug msg="Registering POST, /images/
↪{name:.*}/tag"
time="2016-03-07T22:14:08.849985573Z" level=debug msg="Registering DELETE, /images/
↪{name:.*}"
time="2016-03-07T22:14:08.850055502Z" level=debug msg="Registering GET, /networks"
time="2016-03-07T22:14:08.850105467Z" level=debug msg="Registering GET, /networks/
↪{id:.*}"
time="2016-03-07T22:14:08.850177977Z" level=debug msg="Registering POST, /networks/
↪create"
time="2016-03-07T22:14:08.850235246Z" level=debug msg="Registering POST, /networks/
↪{id:.*}/connect"
time="2016-03-07T22:14:08.850306649Z" level=debug msg="Registering POST, /networks/
↪{id:.*}/disconnect"
time="2016-03-07T22:14:08.850389979Z" level=debug msg="Registering DELETE, /networks/
↪{id:.*}"
time="2016-03-07T22:14:08.850473015Z" level=debug msg="Registering OPTIONS, /
↪{anyroute:.*}"
time="2016-03-07T22:14:08.850537923Z" level=debug msg="Registering GET, /_ping"
time="2016-03-07T22:14:08.850603214Z" level=debug msg="Registering GET, /events"
time="2016-03-07T22:14:08.850654396Z" level=debug msg="Registering GET, /info"
time="2016-03-07T22:14:08.850697470Z" level=debug msg="Registering GET, /version"
time="2016-03-07T22:14:08.850746844Z" level=debug msg="Registering POST, /auth"
```

```
time="2016-03-07T22:14:08.850784258Z" level=debug msg="Registering GET, /volumes"
time="2016-03-07T22:14:08.850834588Z" level=debug msg="Registering GET, /volumes/
↪{name:.*}"
time="2016-03-07T22:14:08.850936508Z" level=debug msg="Registering POST, /volumes/
↪create"
time="2016-03-07T22:14:08.850995395Z" level=debug msg="Registering DELETE, /volumes/
↪{name:.*}"
time="2016-03-07T22:14:08.851065688Z" level=debug msg="Registering POST, /build"
time="2016-03-07T22:14:08.851130299Z" level=info msg="API listen on [::]:2376"
time="2016-03-07T22:14:08.851169803Z" level=info msg="API listen on /var/run/docker.
↪sock"
time="2016-03-07T22:14:09.137038800Z" level=debug msg="Calling GET /v1.15/info"
time="2016-03-07T22:14:09.137075352Z" level=debug msg="GET /v1.15/info"
time="2016-03-07T22:14:09.142123681Z" level=debug msg="Calling GET /v1.15/version"
time="2016-03-07T22:14:09.142158720Z" level=debug msg="GET /v1.15/version"
time="2016-03-07T22:14:09.144484526Z" level=debug msg="Calling GET /v1.15/containers/
↪json"
time="2016-03-07T22:14:09.144504295Z" level=debug msg="GET /v1.15/containers/json?
↪all=1&size=0"
time="2016-03-07T22:14:09.212279532Z" level=debug msg="Calling GET /v1.15/events"
time="2016-03-07T22:14:09.212316010Z" level=debug msg="GET /v1.15/events"
time="2016-03-07T22:14:09.222183110Z" level=debug msg="Calling GET /v1.15/containers/
↪d5d7a3b2009769571f35029711c2564b7ee6231cbadfa858c9bad471a933f676/json"
time="2016-03-07T22:14:09.222216653Z" level=debug msg="GET /v1.15/containers/
↪d5d7a3b2009769571f35029711c2564b7ee6231cbadfa858c9bad471a933f676/json"
time="2016-03-07T22:14:09.247701879Z" level=debug msg="Calling GET /v1.15/containers/
↪21d865de28e79c10a80125d1fb13d72f3e6b3d44accf8264b3e285d3b5ddbe2d/json"
time="2016-03-07T22:14:09.247740095Z" level=debug msg="GET /v1.15/containers/
↪21d865de28e79c10a80125d1fb13d72f3e6b3d44accf8264b3e285d3b5ddbe2d/json"
time="2016-03-07T22:14:09.271798719Z" level=debug msg="Calling GET /v1.15/images/json"
time="2016-03-07T22:14:09.271838761Z" level=debug msg="GET /v1.15/images/json?all=1"
time="2016-03-07T22:14:09.280243977Z" level=debug msg="Calling GET /v1.15/volumes"
time="2016-03-07T22:14:09.280284281Z" level=debug msg="GET /v1.15/volumes"
time="2016-03-07T22:14:09.295893747Z" level=debug msg="Calling GET /v1.15/networks"
time="2016-03-07T22:14:09.295919607Z" level=debug msg="GET /v1.15/networks"
^C
docker@node1:~$ exit
[dimsenv] mboggess@dimsdev2:~ () $ docker-machine ssh node2
                        ##         .
                  ## ## ##        ==
               ## ## ## ## ##    ===
           /"""""""""""""""""\___/ ===
      ~~~ {~~ ~~~~ ~~~ ~~~~ ~~~ ~ /  ===- ~~~
           \_______ o           __/
             \     \         __/
              \____\_______/
 _                 _   ____     _            _
| |__   ___   ___ | |_|___ \ __| | ___   ___| | _____ _ __
| '_ \ / _ \ / _ \| __| __) / _` |/ _ \ / __| |/ / _ \ '__|
| |_) | (_) | (_) | |_ / __/ (_| | (_) | (__|   <  __/ |
|_.__/ \___/ \___/ \__|_____\__,_|\___/ \___|_|\_\___|_|
Boot2Docker version 1.10.2, build master : 611be10 - Mon Feb 22 22:47:06 UTC 2016
Docker version 1.10.2, build c3959b1

# Run the engine-discovery command using the
#  --update flag to force the update
docker@node2:~$ docker run --rm -it --name ucp \
> -v /var/run/docker.sock:/var/run/docker.sock \
```

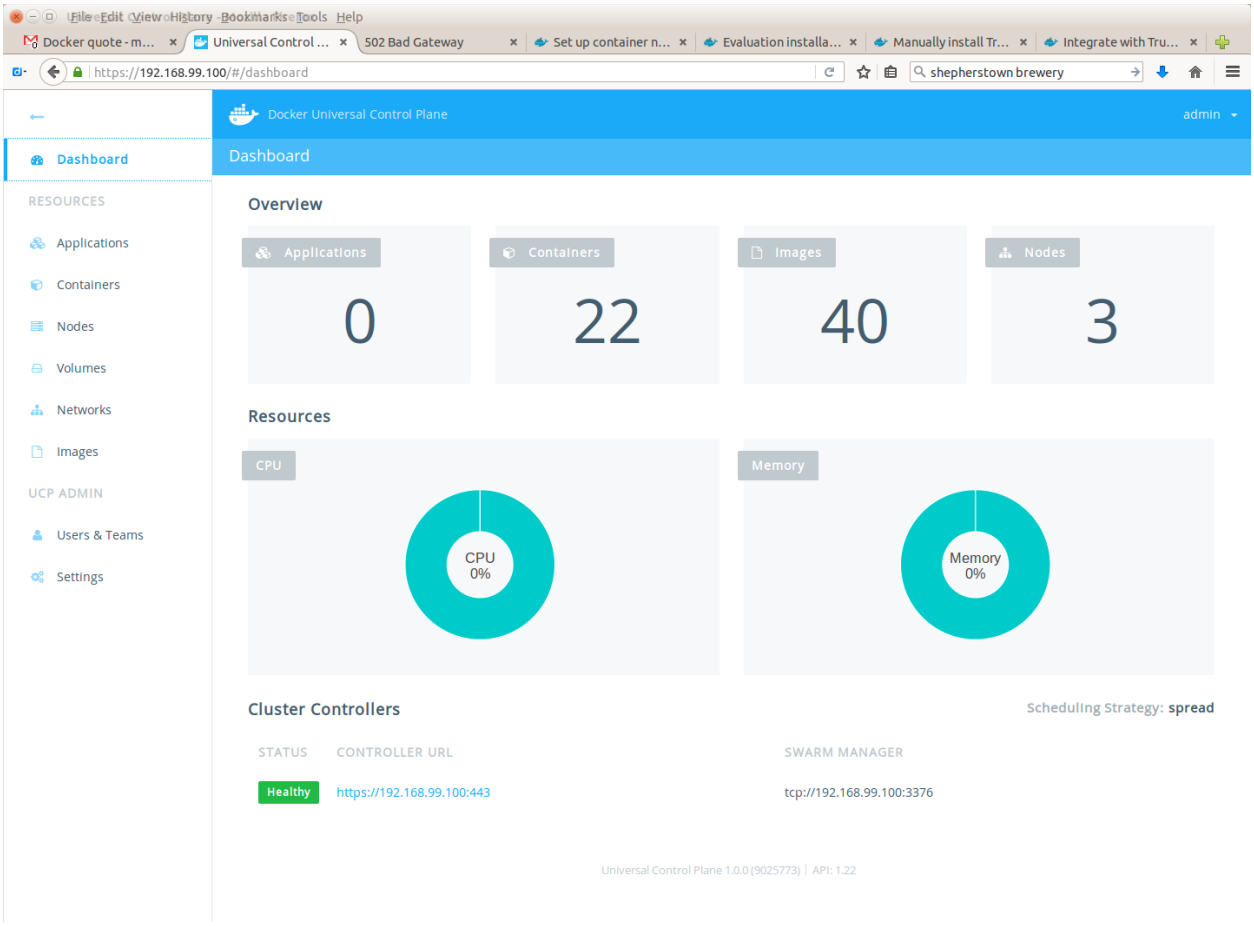
```
> docker/ucp engine-discovery \
> --controller 192.168.99.100 \
> --host-address 192.168.99.102 --update
INFO[0000] This cluster does not have high availability because you only have a␣
↪single controller.
INFO[0000] If you plan to add additional replica nodes for HA, you should do this␣
↪before running this tool to reduce downtime.
WARN[0002] Configuration updated.  You will have to manually restart the docker␣
↪daemon for the changes to take effect.

# Restart docker. Since this is a boot2docker node,
#  it's via init.d rather than the service command.
docker@node2:~$ sudo /etc/init.d/docker restart
Need TLS certs for node2,127.0.0.1,10.0.2.15,192.168.99.102
-------------------

# Check to see if docker is running.
# If this is the first or second nodes, the etcd cluster
#  will complain because it hasn't achieved a quorum yet.
docker@node2:~$ sudo tail -f /var/log/docker.log
time="2016-03-07T22:15:19.579838639Z" level=debug msg="Registering DELETE, /volumes/
↪{name:.*}"
time="2016-03-07T22:15:19.579943547Z" level=debug msg="Registering POST, /build"
time="2016-03-07T22:15:19.580001959Z" level=info msg="API listen on [::]:2376"
time="2016-03-07T22:15:19.580066049Z" level=info msg="API listen on /var/run/docker.
↪sock"
time="2016-03-07T22:15:19.680998567Z" level=debug msg="2016/03/07 22:15:19 [DEBUG]␣
↪serf: messageJoinType: node2\n"
time="2016-03-07T22:15:19.681292971Z" level=debug msg="2016/03/07 22:15:19 [DEBUG]␣
↪serf: messageJoinType: node2\n"
time="2016-03-07T22:15:19.701548847Z" level=debug msg="2016/03/07 22:15:19 [DEBUG]␣
↪serf: messageJoinType: node2\n"
time="2016-03-07T22:15:19.880935252Z" level=debug msg="2016/03/07 22:15:19 [DEBUG]␣
↪serf: messageJoinType: node2\n"
time="2016-03-07T22:15:19.901325865Z" level=debug msg="2016/03/07 22:15:19 [DEBUG]␣
↪serf: messageJoinType: node2\n"
time="2016-03-07T22:15:23.858113731Z" level=debug msg="Watch triggered with 3 nodes"␣
↪discovery=etcd
time="2016-03-07T22:15:25.639472644Z" level=debug msg="Watch triggered with 3 nodes"␣
↪discovery=etcd
^C
docker@node2:~$ exit


############################
# CREATE OVERLAY NETWORK: #
############################

[dimsenv] mboggess@dimsdev2:~ () $ docker-machine ssh node0
                        ##         .
                  ## ## ##        ==
               ## ## ## ## ##    ===
           /"""""""""""""""""\___/ ===
      ~~~ {~~ ~~~~ ~~~ ~~~~ ~~~ ~ /  ===- ~~~
           \_______ o           __/
             \     \         __/
              \____\_______/
 _                 _   _____     _            _
```

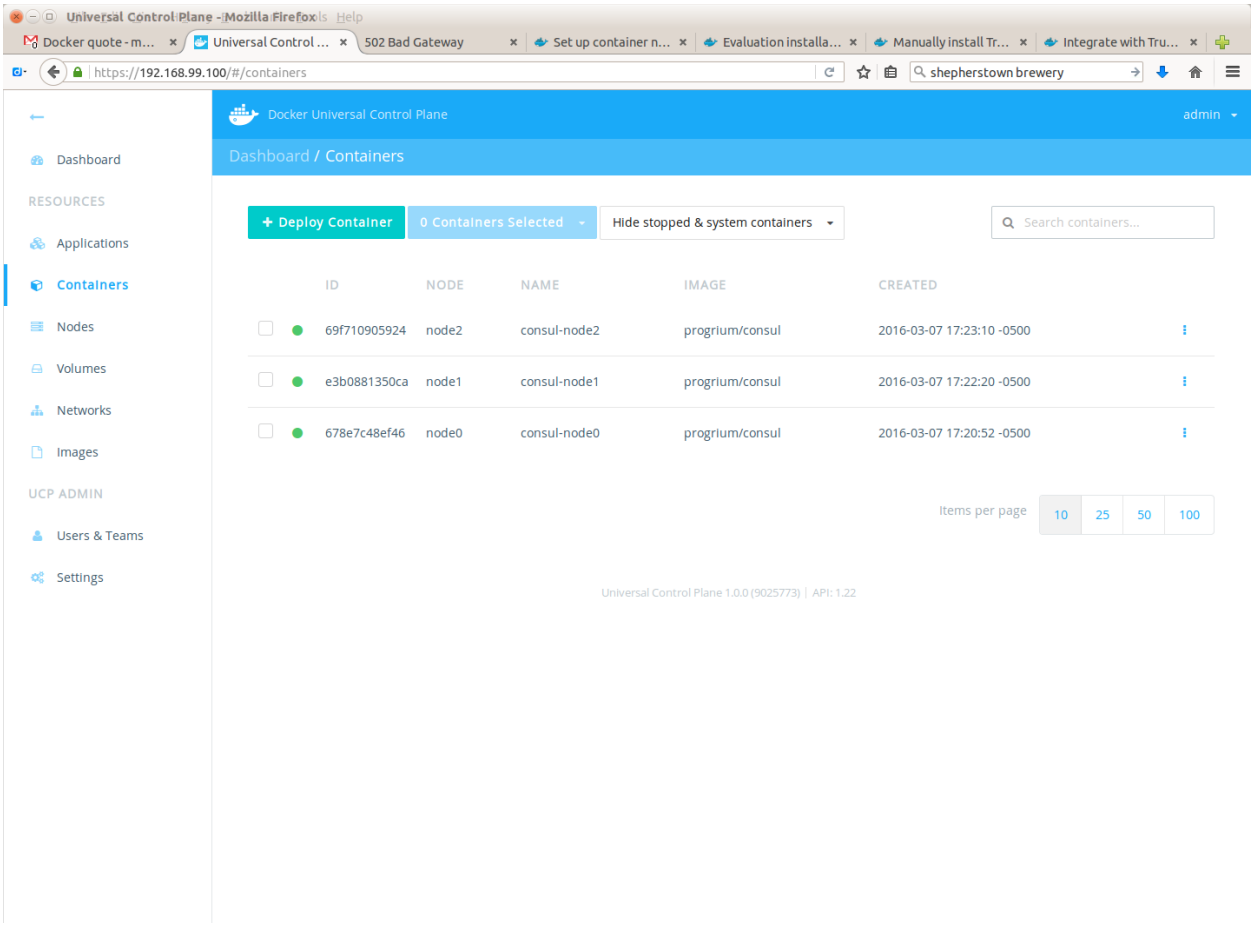
```
| |__   ___    ___ | |_|___ \ __| | ___    ___| | _____ _ __
| '_ \ / _ \ / _ \| __| __) / _` |/ _ \ / __| |/ / _ \ '__|
| |_) | (_) | (_) | |_ / __/ (_| | (_) | (__|   <  __/ |
|_.__/ \___/ \___/ \__|_____\__,_|\___/ \___|_|\_\___|_|
Boot2Docker version 1.10.2, build master : 611be10 - Mon Feb 22 22:47:06 UTC 2016
Docker version 1.10.2, build c3959b1
docker@node0:~$ docker network create -d overlay --subnet=10.4.0.0/16 data.local
96f50ce413255df69e8dfe616190c90d3380d25204c16b27be0c66ebe04057cf
docker@node0:~$ docker network ls
NETWORK ID          NAME                DRIVER
96f50ce41325        data.local          overlay
1faf51793e49        bridge              bridge
b1b5091c0275        none                null
4faefa7c2e28        host                host
docker@node0:~$ exit
[dimsenv] mboggess@dimsdev2:~ () $ docker info
Containers: 7
 Running: 7
 Paused: 0
 Stopped: 0
Images: 7
Server Version: 1.10.2
Storage Driver: aufs
 Root Dir: /mnt/sda1/var/lib/docker/aufs
 Backing Filesystem: extfs
 Dirs: 59
 Dirperm1 Supported: true
Execution Driver: native-0.2
Logging Driver: json-file
Plugins:
 Volume: local
 Network: host bridge overlay null
Kernel Version: 4.1.18-boot2docker
Operating System: Boot2Docker 1.10.2 (TCL 6.4.1); master : 611be10 - Mon Feb 22␣
↪22:47:06 UTC 2016
OSType: linux
Architecture: x86_64
CPUs: 1
Total Memory: 1.909 GiB
Name: node0
ID: AFDQ:Z3N3:OPVT:VTYE:CF5Y:FIZ6:7BIJ:6N6I:GXQ3:DFOP:QV6H:XQKM
Debug mode (server): true
 File Descriptors: 70
 Goroutines: 113
 System Time: 2016-03-07T22:17:05.675047056Z
 EventsListeners: 1
 Init SHA1:
 Init Path: /usr/local/bin/docker
 Docker Root Dir: /mnt/sda1/var/lib/docker
Labels:
 provider=virtualbox
Cluster store: etcd://192.168.99.100:12379
Cluster advertise: 192.168.99.100:12376

###########################
# RUN CONSUL CONTAINERS: #
###########################
```

```
# Remember to run the eval command for each node
#  before running the Consul docker run command!
[dimsenv] mboggess@dimsdev2:~ () $ eval $(docker-machine env node0)
[dimsenv] mboggess@dimsdev2:~ () $ docker run -d --name=consul-node0 --net=data.local␣
↪-v /mnt:/data -p 192.168.99.100:8300:8300 -p 192.168.99.100:8301:8301 -p 192.168.99.
↪100:8301:8301/udp -p 192.168.99.100:8302:8302 -p 192.168.99.100:8302:8302/udp -p␣
↪192.168.99.100:8400:8400 -p 192.168.99.100:8500:8500 -p 192.168.99.100:8600:8600 -p␣
↪172.17.0.1:53:53/udp progrium/consul -node node0 -server -dc local -advertise 192.
↪168.99.100 -bootstrap-expect 3
Unable to find image 'progrium/consul:latest' locally
latest: Pulling from progrium/consul
c862d82a67a2: Pull complete
0e7f3c08384e: Pull complete
0e221e32327a: Pull complete
09a952464e47: Pull complete
60a1b927414d: Pull complete
4c9f46b5ccce: Pull complete
417d86672aa4: Pull complete
b0d47ad24447: Pull complete
fd5300bd53f0: Pull complete
a3ed95caeb02: Pull complete
d023b445076e: Pull complete
ba8851f89e33: Pull complete
5d1cefca2a28: Pull complete
Digest: sha256:8cc8023462905929df9a79ff67ee435a36848ce7a10f18d6d0faba9306b97274
Status: Downloaded newer image for progrium/consul:latest
678e7c48ef46be766e198a414d901078fad1110cf688425703758ce53f1d5758

# EVAL!
[dimsenv] mboggess@dimsdev2:~ () $ eval $(docker-machine env node1)
[dimsenv] mboggess@dimsdev2:~ () $ docker run -d --name=consul-node1 --net=data.local␣
↪-v /mnt:/data -p 192.168.99.101:8300:8300 -p 192.168.99.101:8301:8301 -p 192.168.99.
↪101:8301:8301/udp -p 192.168.99.101:8302:8302 -p 192.168.99.101:8302:8302/udp -p␣
↪192.168.99.101:8400:8400 -p 192.168.99.101:8500:8500 -p 192.168.99.101:8600:8600 -p␣
↪172.17.0.1:53:53/udp progrium/consul -node node1 -server -dc local -advertise 192.
↪168.99.101 -join 192.168.99.100
Unable to find image 'progrium/consul:latest' locally
latest: Pulling from progrium/consul
c862d82a67a2: Pull complete
0e7f3c08384e: Pull complete
0e221e32327a: Pull complete
09a952464e47: Pull complete
60a1b927414d: Pull complete
4c9f46b5ccce: Pull complete
417d86672aa4: Pull complete
b0d47ad24447: Pull complete
fd5300bd53f0: Pull complete
a3ed95caeb02: Pull complete
d023b445076e: Pull complete
ba8851f89e33: Pull complete
5d1cefca2a28: Pull complete
Digest: sha256:8cc8023462905929df9a79ff67ee435a36848ce7a10f18d6d0faba9306b97274
Status: Downloaded newer image for progrium/consul:latest
e3b0881350ca414ac9a652f8fd65f80714c108d316d00a3810162a2f0fc5491e

# EVAL!
[dimsenv] mboggess@dimsdev2:~ () $ eval $(docker-machine env node2)
[dimsenv] mboggess@dimsdev2:~ () $ docker run -d --name=consul-node2 --net=data.local␣
↪-v /mnt:/data -p 192.168.99.102:8300:8300 -p 192.168.99.102:8301:8301 -p 192.168.99.
↪102:8301:8301/udp -p 192.168.99.102:8302:8302 -p 192.168.99.102:8302:8302/udp -p␣
↪192.168.99.102:8400:8400 -p 192.168.99.102:8500:8500 -p 192.168.99.102:8600:8600 -p␣
↪172.17.0.1:53:53/udp progrium/consul -node node2 -server -dc local -advertise 192.
↪168.99.102 -join 192.168.99.100
```

```
Unable to find image 'progrium/consul:latest' locally
latest: Pulling from progrium/consul
c862d82a67a2: Pull complete
0e7f3c08384e: Pull complete
0e221e32327a: Pull complete
09a952464e47: Pull complete
60a1b927414d: Pull complete
4c9f46b5ccce: Pull complete
417d86672aa4: Pull complete
b0d47ad24447: Pull complete
fd5300bd53f0: Pull complete
a3ed95caeb02: Pull complete
d023b445076e: Pull complete
ba8851f89e33: Pull complete
5d1cefca2a28: Pull complete
Digest: sha256:8cc8023462905929df9a79ff67ee435a36848ce7a10f18d6d0faba9306b97274
Status: Downloaded newer image for progrium/consul:latest
69f710905924070ae83f5833129cef38f7409b5ecabdc7335111fe5430571a45
```

Once you've completed the steps outlined, you should be able to go to https://<controller-ip>:443, log in with "admin" and the password you gave during the prompt, submit the license, and see the following:





14.2 Further Information

As more is learned about Docker Datacenter, particularly admin-related information, it will be documented here.

CHAPTER 15

Debugging and Development

This chapter covers some tools and tactics used for testing and debugging misbehaving system components, or obtaining sufficient detail about how subsystem components work in order to control them to achieve project goals and requirement objectives. Executing some command line or triggering an action in a user interface that results in the system appearing to "hang" can be caused by many things. Just looking at the surface and seeing no action is useless in determining the root cause of the issue. The ability to turn a "black box" into a "transparent box" is invaluable to the process of testing and debugging.

Hint: Some useful resources on the processes of *testing and debugging* are listed here:

- Testing and Debugging
- White-box testing, Wikipedia
- White-Box Testing, by Oliver Cole, March 1, 2000

15.1 Determining File System Affects of Running Programs

Many programs used in the DIMS project consume gigabytes of disk storage, often in hidden locations that are not obvious to the user. The act of taking an Ubuntu 14.04 install ISO image, converting it with Packer into a BOX file, turning that into a Virtualbox image, and instantiating a Vagrant virtual machine can turn just under 1 GB into 3-5 GB of storage. Multiply that by a dozen or more virtual machines and this quickly can add up. If you are not aware of what gets created, and you change names of virtual machines, you can end up with a huge amount of wasted disk space with unused virtual disks, virtual machine images, etc.

For this reason, every programmer developing tools that use programs like this should be methodical about understanding every process in terms of inputs, process, and **outputs**, such that it is possible to **see** what is produced to help the person using your tools know what is happening, and to **undo** those effects in a controlled way to simplify cleaning up.

Hint: An easy way to help the user of your tools is to be organized and put large files related to a workflow like Packer->Virtualbox->Vagrant VM creation under a single directory path like `/vm` that can deleted in one step, backed

up and moved to another system with a larger hard drive, or expanded by mounting a second hard drive onto that directory path as a mount point. Scattering the files across many unrelated subdirectories in random locations and depths within the user's `$HOME` directory tree makes it much harder to handle a situation where the hard drive on a laptop reaches 100% utilization.

Let's take a look at a portion of the workflow of Packer->Virtualbox->Vagrant creation to see how to white-box disk utilization and space management.

We start by changing directory into the `$GIT/dims-packer` repository where tools for creating Vagrants using Packer are kept. We create an initial empty file to serve as a marker in time for then locating any files that were created **after** this file.

Note: The example here will search through a set of directories that were chosen based on knowledge that they exist and are used by various tools. To obtain this knowledge, it is often helpful to start looking at the root of the filesystem (`/`) and look for **any** files in **any** directories, which you will quickly find has a lot of unrelated file system additions that just happen to have been made at the same time as the program you were running. A more precise way to identify where files are created is to trace execution of the program in question, following all forked children, using a program like `strace` and/or `ltrace`, however these tools require a much deeper understanding of how the Unix/Linux kernel works.

```
$ cd $GIT/dims-packer
```

```
$ cd $GIT/dims-packer
$ find /home/dittrich/.vagrant.d/ /home/dittrich/.packer.d/ /home/dittrich/
↪VirtualBox\ VMs/ . /vm -newer foo -ls
56230373    4 drwxrwxr-x   7 dittrich dittrich     4096 Mar 15 13:14 /home/dittrich/.
↪vagrant.d/
56230688    0 -rw-rw-r--   1 dittrich dittrich        0 Mar 15 13:14 /home/dittrich/.
↪vagrant.d/data/machine-index/index.lock
56230689    4 drwxr-xr-x   2 dittrich dittrich     4096 Mar 15 13:14 /home/dittrich/.
↪packer.d/
56230691    4 -rw-rw-r--   1 dittrich dittrich      318 Mar 15 13:14 /home/dittrich/.
↪packer.d/checkpoint_cache
55314574    4 drwx------   6 dittrich dittrich     4096 Mar 15 13:24 /home/dittrich/
↪VirtualBox\ VMs/
58589344 2183628 -rw-------   1 dittrich dittrich 2240348160 Mar 15 13:27 /home/
↪dittrich/VirtualBox\ VMs/vagrant-run-ns1_default_1458069887689_42029/ns1_box-disk1.
↪vmdk
58987212    4 drwx------   2 dittrich dittrich     4096 Mar 15 13:24 /home/dittrich/
↪VirtualBox\ VMs/devserver
55574611    4 drwxrwxr-x  19 dittrich dittrich     4096 Mar 15 13:24 .
58590167    4 drwxrwxr-x   2 dittrich dittrich     4096 Mar 15 13:24 ./vagrant-output
58590705 633044 -rw-rw-r--   1 dittrich dittrich 648229139 Mar 15 13:24 ./vagrant-
↪output/packer_devserver_box_virtualbox.box
55574679    4 drwxrwxr-x   2 dittrich dittrich     4096 Mar 15 13:14 ./ubuntu_64_
↪vagrant
55574655    4 -rw-rw-r--   1 dittrich dittrich     3263 Mar 15 13:14 ./ubuntu_64_
↪vagrant/devserver-base.json
55574654    4 -rw-rw-r--   1 dittrich dittrich     3044 Mar 15 13:14 ./ubuntu_64_
↪vagrant/devserver-box.json
58590704    4 drwxr-xr-x   2 dittrich dittrich     4096 Mar 15 13:21 ./output-
↪devserver
58590711   12 -rw-------   1 dittrich dittrich    10629 Mar 15 13:20 ./output-
↪devserver/devserver.ovf
58590712 620528 -rw-------   1 dittrich dittrich 635416064 Mar 15 13:21 ./output-
↪devserver/devserver-disk1.vmdk
```

```
55574612    4 drwxrwxr-x   8 dittrich dittrich     4096 Mar 15 13:27 ./.git
```

```
[dimsenv] dittrich@dimsdemo1:~/dims/git/dims-packer (feature/dims-696*) $ touch foo2
[dimsenv] dittrich@dimsdemo1:~/dims/git/dims-packer (feature/dims-696*) $ find /home/
↪dittrich/.vagrant.d/ /home/dittrich/.packer.d/ /home/dittrich/VirtualBox\ VMs/ . /
↪vm -newer foo2 -ls
56230373    4 drwxrwxr-x   7 dittrich dittrich     4096 Mar 15 13:33 /home/dittrich/.
↪vagrant.d/
56230688    0 -rw-rw-r--   1 dittrich dittrich        0 Mar 15 13:33 /home/dittrich/.
↪vagrant.d/data/machine-index/index.lock
55574612    4 drwxrwxr-x   8 dittrich dittrich     4096 Mar 15 13:33 ./.git
53346305    4 drwxr-xr-x   5 dittrich dittrich     4096 Mar 15 13:33 /vm
53346306    4 drwxrwxr-x   2 dittrich dittrich     4096 Mar 15 13:33 /vm/devserver
53346314    4 -rw-rw-r--   1 dittrich dittrich        1 Mar 15 13:33 /vm/devserver/.
↪vagrant-IP
53346310    4 -rw-rw-r--   1 dittrich dittrich        6 Mar 15 13:33 /vm/devserver/.
↪vagrant-ISDESKTOP
53346311    4 -rw-rw-r--   1 dittrich dittrich        7 Mar 15 13:33 /vm/devserver/.
↪vagrant-VMTYPE
53346312    4 -rw-rw-r--   1 dittrich dittrich        7 Mar 15 13:33 /vm/devserver/.
↪vagrant-PLATFORM
53346309    4 -rw-rw-r--   1 dittrich dittrich       10 Mar 15 13:33 /vm/devserver/.
↪vagrant-NAME
53346313    4 -rw-rw-r--   1 dittrich dittrich       32 Mar 15 13:33 /vm/devserver/.
↪vagrant-BOXNAME
53346316    4 -rw-rw-r--   1 dittrich dittrich       26 Mar 15 13:33 /vm/devserver/.
↪vagrant-VAGRANTFILEPATH
53346319    8 -rwxrwxr-x   1 dittrich dittrich     4351 Mar 15 13:33 /vm/devserver/
↪test.vagrant.ansible-current
53346318    8 -rw-rw-r--   1 dittrich dittrich     4245 Mar 15 13:33 /vm/devserver/
↪Makefile
53346315    4 -rw-rw-r--   1 dittrich dittrich        1 Mar 15 13:33 /vm/devserver/.
↪vagrant-FORWARDPORT
53346308    4 -rw-rw-r--   1 dittrich dittrich     2738 Mar 15 13:33 /vm/devserver/
↪Vagrantfile
53346307    4 -rw-rw-r--   1 dittrich dittrich     2028 Mar 15 13:33 /vm/devserver/.
↪vagrant_show
53346317    4 -rw-rw-r--   1 dittrich dittrich      199 Mar 15 13:33 /vm/devserver/
↪hosts
```

```
[dimsenv] dittrich@dimsdemo1:~/dims/git/dims-packer (feature/dims-696*) $ touch foo3
[dimsenv] dittrich@dimsdemo1:~/dims/git/dims-packer (feature/dims-696*) $ find /home/
↪dittrich/.vagrant.d/ /home/dittrich/.packer.d/ /home/dittrich/VirtualBox\ VMs/ . /
↪vm -newer foo3 -ls
56230373    4 drwxrwxr-x   7 dittrich dittrich     4096 Mar 15 13:48 /home/dittrich/.
↪vagrant.d/
56230681    4 drwxrwxr-x   4 dittrich dittrich     4096 Mar 15 13:34 /home/dittrich/.
↪vagrant.d/data
56232110    0 -rw-rw-r--   1 dittrich dittrich        0 Mar 15 13:34 /home/dittrich/.
↪vagrant.d/data/lock.dotlock.lock
56230688    0 -rw-rw-r--   1 dittrich dittrich        0 Mar 15 13:48 /home/dittrich/.
↪vagrant.d/data/machine-index/index.lock
56232608    0 -rw-rw-r--   1 dittrich dittrich        0 Mar 15 13:34 /home/dittrich/.
↪vagrant.d/data/lock.machine-action-fab0a1f680af28d59f47b677629a540a.lock
56230682    4 drwxrwxr-x   2 dittrich dittrich     4096 Mar 15 13:35 /home/dittrich/.
↪vagrant.d/tmp
56230680    4 drwxrwxr-x  11 dittrich dittrich     4096 Mar 15 13:35 /home/dittrich/.
↪vagrant.d/boxes
```

```
58987205    4 drwxrwxr-x   3 dittrich dittrich     4096 Mar 15 13:35 /home/dittrich/.
↪vagrant.d/boxes/packer_devserver_box_virtualbox
58987206    4 drwxrwxr-x   3 dittrich dittrich     4096 Mar 15 13:35 /home/dittrich/.
↪vagrant.d/boxes/packer_devserver_box_virtualbox/0
58987207    4 drwxrwxr-x   2 dittrich dittrich     4096 Mar 15 13:35 /home/dittrich/.
↪vagrant.d/boxes/packer_devserver_box_virtualbox/0/virtualbox
58987202 646144 -rw-rw-r--   1 dittrich dittrich 661647360 Mar 15 13:35 /home/
↪dittrich/.vagrant.d/boxes/packer_devserver_box_virtualbox/0/virtualbox/devserver_
↪box-disk1.vmdk
58987203    4 -rw-rw-r--   1 dittrich dittrich       26 Mar 15 13:35 /home/dittrich/.
↪vagrant.d/boxes/packer_devserver_box_virtualbox/0/virtualbox/metadata.json
58987200    4 -rw-rw-r--   1 dittrich dittrich      258 Mar 15 13:34 /home/dittrich/.
↪vagrant.d/boxes/packer_devserver_box_virtualbox/0/virtualbox/Vagrantfile
58987201   12 -rw-rw-r--   1 dittrich dittrich    10785 Mar 15 13:34 /home/dittrich/.
↪vagrant.d/boxes/packer_devserver_box_virtualbox/0/virtualbox/box.ovf
55574611    4 drwxrwxr-x  19 dittrich dittrich     4096 Mar 15 13:48 .
55574612    4 drwxrwxr-x   8 dittrich dittrich     4096 Mar 15 13:48 ./.git
55575296    4 -rw-rw-r--   1 dittrich dittrich     2590 Mar 15 13:48 ./make-devserver-
↪201603151348.txt
53346306    4 drwxrwxr-x   5 dittrich dittrich     4096 Mar 15 13:48 /vm/devserver
53346314    4 -rw-rw-r--   1 dittrich dittrich       14 Mar 15 13:48 /vm/devserver/.
↪vagrant-IP
53346310    4 -rw-rw-r--   1 dittrich dittrich        6 Mar 15 13:48 /vm/devserver/.
↪vagrant-ISDESKTOP
53346311    4 -rw-rw-r--   1 dittrich dittrich        7 Mar 15 13:48 /vm/devserver/.
↪vagrant-VMTYPE
53346312    4 -rw-rw-r--   1 dittrich dittrich        7 Mar 15 13:48 /vm/devserver/.
↪vagrant-PLATFORM
53346309    4 -rw-rw-r--   1 dittrich dittrich       10 Mar 15 13:48 /vm/devserver/.
↪vagrant-NAME
53346313    4 -rw-rw-r--   1 dittrich dittrich       32 Mar 15 13:48 /vm/devserver/.
↪vagrant-BOXNAME
53347678    4 drwxrwxr-x  10 dittrich dittrich     4096 Mar 15 13:48 /vm/devserver/
↪dims-keys
53347720    0 -rw-rw-r--   1 dittrich dittrich        0 Mar 15 13:48 /vm/devserver/
↪dims-keys/README.rd
53347719    4 -rw-rw-r--   1 dittrich dittrich       43 Mar 15 13:48 /vm/devserver/
↪dims-keys/.gitignore
53347722    4 drwxrwxr-x   2 dittrich dittrich     4096 Mar 15 13:48 /vm/devserver/
↪dims-keys/ansible-pub
. . .
53347752    4 -rw-rw-r--   1 dittrich dittrich      402 Mar 15 13:48 /vm/devserver/
↪dims-keys/ssh-pub/dims_andclay_rsa.pub
53347775    4 -rw-rw-r--   1 dittrich dittrich       79 Mar 15 13:48 /vm/devserver/
↪dims-keys/ssh-pub/dims_eliot_rsa.sig
53346320    4 drwxrwxr-x   3 dittrich dittrich     4096 Mar 15 13:34 /vm/devserver/.
↪vagrant
53346321    4 drwxrwxr-x   3 dittrich dittrich     4096 Mar 15 13:34 /vm/devserver/.
↪vagrant/machines
53346322    4 drwxrwxr-x   3 dittrich dittrich     4096 Mar 15 13:34 /vm/devserver/.
↪vagrant/machines/default
53346323    4 drwxrwxr-x   2 dittrich dittrich     4096 Mar 15 13:34 /vm/devserver/.
↪vagrant/machines/default/virtualbox
53346316    4 -rw-rw-r--   1 dittrich dittrich       26 Mar 15 13:48 /vm/devserver/.
↪vagrant-VAGRANTFILEPATH
53346318    8 -rw-rw-r--   1 dittrich dittrich     4245 Mar 15 13:48 /vm/devserver/
↪Makefile
53346315    4 -rw-rw-r--   1 dittrich dittrich        1 Mar 15 13:48 /vm/devserver/.
↪vagrant-FORWARDPORT
```

```
53346308    4 -rw-rw-r--   1 dittrich dittrich     2751 Mar 15 13:48 /vm/devserver/
↪Vagrantfile
53346307    4 -rw-rw-r--   1 dittrich dittrich     2041 Mar 15 13:48 /vm/devserver/.
↪vagrant_show
53346317    4 -rw-rw-r--   1 dittrich dittrich      212 Mar 15 13:48 /vm/devserver/
↪hosts
```

```
$ touch foo4
$ find /home/dittrich/.vagrant.d/ /home/dittrich/.packer.d/ /home/dittrich/
↪VirtualBox\ VMs/ . /vm -newer foo4 -ls
58589344 2183628 -rw-------   1 dittrich dittrich 2240348160 Mar 15 14:17 /home/
↪dittrich/VirtualBox\ VMs/vagrant-run-ns1_default_1458069887689_42029/ns1_box-disk1.
↪vmdk
55574611    4 drwxrwxr-x  19 dittrich dittrich     4096 Mar 15 14:13 .
55576829   28 -rw-rw-r--   1 dittrich dittrich    27191 Mar 15 14:13 ./Makefile
55574612    4 drwxrwxr-x   8 dittrich dittrich     4096 Mar 15 14:13 ./.git
53870594    4 drwxrwxr-x   5 dittrich dittrich     4096 Mar 15 14:15 /vm/vagrant-run-
↪ns1
53870676    4 -rw-rw-r--   2 dittrich dittrich        4 Mar 15 14:13 /vm/vagrant-run-
↪ns1/ns1.dims
53870676    4 -rw-rw-r--   2 dittrich dittrich        4 Mar 15 14:13 /vm/vagrant-run-
↪ns1/ns1.local
53346306    4 drwxrwxr-x   5 dittrich dittrich     4096 Mar 15 13:51 /vm/devserver
53347790    4 -rw-rw-r--   1 dittrich dittrich     2756 Mar 15 13:51 /vm/devserver/
↪Vagrantfile
```

15.2 Testing Code on Branches

The DIMS development environment uses Python virtual environments on developer workstations in order to isolate functional "production" code from more volatile development or test quality code. This means that development code on feature branches can safely be tested by following these basic steps:

1. Create a new Python virtual environment that is a clone of the current stable code base.
2. Pull the latest code from the repos to be tested, and check out the latest code.
3. Install the programs from these branches into the new Python virtual environment, using programs like `dims.install.dimscommands` and/or `dims.ansible-playbook` as necessary.
4. Test the code, possibly using the `test.runner` script to invoke scripted tests that help validate the funtionality of the new code.

When you are satisfied with the tests, the Python virtual environment can safely be deleted (or you can switch back to the "production" default `dimsenv` Python virtual environment) to return to using stable program code.

Let's say we are going to test the repos `ansible-playbooks` and `dims-packer`, both on the feature branch named `feature/interfaces`.

First, create the new Python virtual environment:

```
[dimsenv] dittrich@dimsdemo1:~ () $ dimsenv.update --venv interfaces -v
[+] NO virtual environment identified and is active
[+] User virtual environment "interfaces" needs replacement
[+] Running: virtualenv-clone /opt/dims/envs/dimsenv /home/dittrich/dims/envs/
↪interfaces
[+] Installing pre/post scripts from /opt/dims/envs into /home/dittrich/dims/envs
[+] Processing files for /home/dittrich/dims/envs/interfaces/bin
```

```
[+] Installed keys.host.create
[+] Installed dyn_inv.py
[+] Installed dims-ci-utils.update.user
[+] Installed fix.adddeploylinks
[+] Installed fix.ansible_managed
[+] Installed fix.addlinks
[+] Installed fix.addvars
[+] Installed fix.includes
[+] Installed fix.removetrailingspaces
[+] Installed dims.install.createusb
[+] Installed dims.makedocset
[+] Installed dims.boot2docker
[+] Installed dims.buildvirtualenv
[+] Installed dims.bumpversion
[+] Installed dims.jj2
[+] Installed dims.git.repoversion
[+] Installed dims.nas.mount
[+] Installed dims.nas.umount
[+] Installed dims.remote.setupworkstation
[+] Installed dims.swapcapslockctrl
[+] Installed dims.shutdown
[+] Installed dims.sphinx-autobuild
[+] Installed test.ansible.yaml
[+] Installed test.md5.output
[+] Installed test.supervisor
[+] Installed test.yaml.validate
[+] Installed dims.localcluster.create
[+] Installed dims.localcluster.start
[+] Installed dims.localcluster.stop
[+] Installed dims.localcluster.destroy
[+] Installed dims.localcluster.status
[+] Installed dims.localcluster.runscript
[+] Installed dims.clusterconfig.nas
[+] Installed dims.clusterconfig.local
[+] Installed dims.clusterconfig.list
[+] Installed dims.cluster.runscript
[+] Installed dims.elasticsearch.service
[+] Installed test.vagrant.ansible-current
[+] Installed test.vagrant.factory
[+] Installed test.vagrant.list
[+] Installed test.packer.factory
[+] Installed test.packer.list
[+] Installed test.vagrant.listvms
[+] Successfully installed 43 programs
[-] To enable the interfaces virtual environment, do "exec bash" or log out/log in
```

Activate the new virtual environment:

```
[dimsenv] dittrich@dimsdemo1:~ () $ workon interfaces
[+] Virtual environment 'dimsenv' activated [ansible-playbooks v1.3.33]
[interfaces] dittrich@dimsdemo1:~ () $
```

Update the first repo and check out the desired branch for testing.

```
[interfaces] dittrich@dimsdemo1:~ () $ cd $GIT/ansible-playbooks
[interfaces] dittrich@dimsdemo1:~/dims/git/ansible-playbooks (develop*) $ git hf␣
↪update && git hf pull && git checkout feature/interfaces
Fetching origin
```

```
remote: Counting objects: 71, done.
remote: Compressing objects: 100% (44/44), done.
remote: Total 44 (delta 33), reused 0 (delta 0)
Unpacking objects: 100% (44/44), done.
From git.devops.develop:/opt/git/ansible-playbooks
   d3ae79a..cd789e9  develop    -> origin/develop
 * [new branch]      feature/interfaces -> origin/feature/interfaces
Summary of actions:
- Any changes to branches at origin have been downloaded to your local repository
- Any branches that have been deleted at origin have also been deleted from your␣
↪local repository
- Any changes from origin/master have been merged into branch 'master'
- Any changes from origin/develop have been pulled into branch 'develop'
- Any resolved merge conflicts have been pushed back to origin
- You are now on branch 'develop'
Fetching origin
Summary of actions:
Branch feature/interfaces set up to track remote branch feature/interfaces from␣
↪origin by rebasing.
Switched to a new branch 'feature/interfaces'
[dimsenv] dittrich@dimsdemo1:~/dims/git/ansible-playbooks (feature/interfaces*) $
```

Update the subsequent repo(s), as necessary, and check out the desired branch for testing.

```
[interfaces] dittrich@dimsdemo1:~/dims/git/ansible-playbooks (feature/interfaces*) $␣
↪cd $GIT/dims-packer
[interfaces] dittrich@dimsdemo1:~/dims/git/dims-packer (feature/interfaces*) $ git hf␣
↪update && git hf pull && git checkout feature/interfaces
Fetching origin
remote: Counting objects: 72, done.
remote: Compressing objects: 100% (55/55), done.
remote: Total 61 (delta 30), reused 0 (delta 0)
Unpacking objects: 100% (61/61), done.
From git.devops.develop:/opt/git/dims-packer
   069d966..2d47264  feature/interfaces -> origin/feature/interfaces

Summary of actions:
- Any changes to branches at origin have been downloaded to your local repository
- Any branches that have been deleted at origin have also been deleted from your␣
↪local repository
- Any changes from origin/master have been merged into branch 'master'
- Any changes from origin/develop have been merged into branch 'develop'
- Any resolved merge conflicts have been pushed back to origin
- You are now on branch 'feature/interfaces'
Fetching origin
Updating 069d966..2d47264
Fast-forward
 Makefile-vagrant.j2                                                  |  2 +-
 Vagrantfile.j2                                                       | 23␣
↪+++++++++++++---------
 scripts/all/create-network-interfaces.sh                             | 17␣
↪++++++++++++++----
 scripts/all/{network-capture.sh => network-debug.sh}                 |  1 -
 scripts/jessie/after-up/00-create-network-interfaces.sh              |  1 +
 scripts/{xenial/post-provision.sh => jessie/after-up/05-jessie-networking.sh} | 22␣
↪++++++++++++++++++-----
 scripts/jessie/after-up/10-network-debug.sh                          |  1 +
 scripts/jessie/post-provision.sh                                     | 42 --
↪----------------------------------------
```

```
 scripts/trusty/after-up/00-create-network-interfaces.sh                    |  1 +
 scripts/trusty/after-up/10-network-debug.sh                                |  1 +
 scripts/trusty/post-provision.sh                                           | 42 --
↪------------------------------------------
 scripts/wheezy/after-up/00-create-network-interfaces.sh                    |  1 +
 scripts/wheezy/after-up/10-network-debug.sh                                |  1 +
 scripts/wheezy/post-provision.sh                                           | 42 --
↪------------------------------------------
 scripts/xenial/after-up/00-create-network-interfaces.sh                    |  1 +
 scripts/xenial/after-up/10-network-debug.sh                                |  1 +
 test.vagrant.factory                                                       | 62␣
↪++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 17 files changed, 87 insertions(+), 174 deletions(-)
 rename scripts/all/{network-capture.sh => network-debug.sh} (99%)
 create mode 120000 scripts/jessie/after-up/00-create-network-interfaces.sh
 rename scripts/{xenial/post-provision.sh => jessie/after-up/05-jessie-networking.sh}␣
↪(68%)
 create mode 120000 scripts/jessie/after-up/10-network-debug.sh
 delete mode 100755 scripts/jessie/post-provision.sh
 create mode 120000 scripts/trusty/after-up/00-create-network-interfaces.sh
 create mode 120000 scripts/trusty/after-up/10-network-debug.sh
 delete mode 100755 scripts/trusty/post-provision.sh
 create mode 120000 scripts/wheezy/after-up/00-create-network-interfaces.sh
 create mode 120000 scripts/wheezy/after-up/10-network-debug.sh
 delete mode 100755 scripts/wheezy/post-provision.sh
 create mode 120000 scripts/xenial/after-up/00-create-network-interfaces.sh
 create mode 120000 scripts/xenial/after-up/10-network-debug.sh

Summary of actions:
- Any changes from origin/feature/interfaces have been pulled into branch 'feature/
↪interfaces'
Already on 'feature/interfaces'
Your branch is up-to-date with 'origin/feature/interfaces'.
[interfaces] dittrich@dimsdemo1:~/dims/git/dims-packer (feature/interfaces*) $
```

Now that both repos have been pulled, and their respective `feature/interfaces` branches checked out, install any updated programs to be tested:

```
[interfaces] dittrich@dimsdemo1:~/dims/git/dims-packer (feature/interfaces*) $ dims.
↪install.dimscommands -v
[+] Processing files for /home/dittrich/dims/envs/interfaces/bin
[+] Installed test.vagrant.factory
[+] Successfully installed 1 program
```

15.3 Debugging Vagrant

Vagrant is used to create and destroy Virtual Machine sub-systems within DIMS deployments. It is designed to create replicable development environments, but is also being used to instantiate replica DIMS deployments to facilitate not only development, but also testing and isolation of deployments to exercise and document system administration tasks.

Vagrant, like Ansible and some other open source tools used in the DIMS project, sometimes are sparse on documentation, especially of advanced features necessary for small-scale distributed systems deployment. This can make debugging harder, since the functionality is wrapped in a black box that is the `vagrant` command line (which may be buried within a `Makefile` and/or Bash script.)

15.3.1 Verbosity in Vagrant Provisioners

The first way to turn this black box into a white box is to enable debugging within the provisioners being called, such as the `ansible` provisioner. To do this, the `Vagrantfile` produced by DIMS scripts allows an environment variable `VERBOSITY` to be passed to the `ansible` provisioner:

```
. . .
ANSIBLE_VERBOSITY = ENV['VERBOSITY'].nil? ? "vv" : ENV['VERBOSITY']
  . . .
  config.vm.provision "ansible" do |ansible|
    . . .
    # Use the environment variable VERBOSITY to change verbosity level.
    ansible.verbose = ANSIBLE_VERBOSITY
    . . .
  end
. . .
```

This mechanism adds verbosity to the provisioner being called **by** Vagrant, but does nothing to help you see what Vagrant itself is doing before and after the `ansible` provisioner is called.

15.3.2 Vagrant Debug Logging

To get debugging output from Vagrant itself, there is another environment variable that produces log output from Vagrant onto `stderr`, which can be redirected to a file for examination (shown here is the context of a GNU `Makefile`):

```
#HELP up - Do 'vagrant up --no-provision'
.PHONY: up
up:
        @if [ "$(VAGRANT_STATUS)" != "running" ]; then \
            echo "[+] vagrant up --no-provision"; \
            VAGRANT_LOG=debug vagrant up --no-provision 2>/tmp/vagrant-$(FQDN).log; \
         fi
```

The output is quite voluminous and shows not only what Vagrant is doing internally, but also how it is calling programs like `vboxmanage` to manipulate the Vagrant Virtual Machine.

```
. . .
 INFO global: Vagrant version: 1.8.6
 INFO global: Ruby version: 2.2.5
 INFO global: RubyGems version: 2.4.5.1
 INFO global: VAGRANT_LOG="debug"
. . .
 INFO subprocess: Starting process: ["/usr/bin/VBoxManage", "sharedfolder", "add",
↪"16425ef3-0e00-4c8e-89aa-116065f1cb36", "--name", "home_ansible_vagrant", "--
↪hostpath", "/vm/run/blue14"]
 INFO subprocess: Command not in installer, restoring original environment...
DEBUG subprocess: Selecting on IO
DEBUG subprocess: Waiting for process to exit. Remaining to timeout: 32000
DEBUG subprocess: Exit status: 0
 INFO subprocess: Starting process: ["/usr/bin/VBoxManage", "setextradata", "16425ef3-
↪0e00-4c8e-89aa-116065f1cb36", "VBoxInternal2/SharedFoldersEnableSymlinksCreate/home_
↪ansible_sources", "1"]
 INFO subprocess: Command not in installer, restoring original environment...
DEBUG subprocess: Selecting on IO
DEBUG subprocess: Waiting for process to exit. Remaining to timeout: 32000
```

```
DEBUG subprocess: Exit status: 0
 INFO subprocess: Starting process: ["/usr/bin/VBoxManage", "sharedfolder", "add",
↪"16425ef3-0e00-4c8e-89aa-116065f1cb36", "--name", "home_ansible_sources", "--
↪hostpath", "/vm/cache/sources"]
 INFO subprocess: Command not in installer, restoring original environment...
DEBUG subprocess: Selecting on IO
DEBUG subprocess: Waiting for process to exit. Remaining to timeout: 32000
DEBUG subprocess: Exit status: 0
. . .
```

Caution: Remember that environment variables, once set, are inheritted by all child processes (unless they are **unset** before another sub-process is started). Using an environment variable to enable logging in a program means that not only the initial process running the `vagrant` program, **but all child processes created by this parent process** that also run `vagrant` will have debugging output on `stdout`. If the `Vagrantfile` itself instructs `vagrant` to directly run `vagrant`, the parent process will receive this output on `stderr` and may interpret it to mean "failure" when there is actually no real error.

```
# Shell provisioner to do post-provisioning actions
# See https://github.com/emyl/vagrant-triggers/wiki/Trigger-recipes
config.trigger.after [:provision], :stdout => true do
  run "vagrant ssh -c '/opt/dims/bin/trigger.runner --state after-provision'"
end
```

In this case, make sure that `:stderr => false` is included on the trigger configuration line to either prevent output to `stderr` and/or config non-error exit code values.

15.3.3 Use the Source, Luke

Lastly, you may need to read the Vagrant source code itself to find out how Vagrant operates. For example, the file `vagrant/plugins/providers/virtualbox/action/network.rb` shows the defaults used by Vagrant for Virtualbox virtual networking using DHCP:

```
#------------------------------------------------------------------
# DHCP Server Helper Functions
#------------------------------------------------------------------

DEFAULT_DHCP_SERVER_FROM_VBOX_INSTALL = {
  network_name: 'HostInterfaceNetworking-vboxnet0',
  network:      'vboxnet0',
  ip:           '192.168.56.100',
  netmask:      '255.255.255.0',
  lower:        '192.168.56.101',
  upper:        '192.168.56.254'
}.freeze
```

This shows the default range for dynamically assigned addresses (which you will want to avoid using for any static addresses on the same network to avoid possible conflicts.)

The source code, a few lines lower, also shows what and how Vagrant will log the fact that it is creating a DHCP server to manage addresses:

```
@logger.debug("Creating a DHCP server...")
@env[:machine].provider.driver.create_dhcp_server(interface[:name], config)
```

The string "Creating a DHCP server..." is what you would look for in the log output produced by setting the `VAGRANT_LOG` environment variable as described earlier.

CHAPTER 16

Robot Framework

This chapter briefly introduces the Robot Framework, how to install it, how to configure it with PyCharm, and other helpful tidbits for running Robot Framework tests.

16.1 Overview

Robot Framework is an open source product developed by Google. It is a keyword-driven framework that automates the testing process. Robot has builtin test libraries, external test libraries, and you can develop your own test libraries in Python or Java. Additionally, Robot's tests are written in natural language, and are thus easy to develop and read.

16.2 Installation

Prerequisites:

- Python 2.7
- python-setuptools
- python-pip
- python-wxgtk2.8

Then run:

```
pip install robotframework
```

16.3 Configuring with Pycharm

For features like syntax highlighting of Robot files and "jump to source" for Robot test library functions, you need to install a PyCharm plugin. The one for the Robot Framework is called intellibot.

To install the intellibot plugin:

- Go to `Preferences > Plugins > Browse repositories`
- Type "intellibot" in the search box
- Click `Download and install`

This will allow you to hover over a function defined in some test library and click to jump to the source. It also enables syntax highlighting of `*.robot` files.

If there are any other file types you'd like to include in having Robot syntax highlighting, do the following:

- Go to `Preferences > File Types > Robot Feature Files`
- Click the "+" in a box
- Add `*.txt` or any other file extension for which you'd like to have Robot syntax highlighting always enabled.

There are many video tutorials for installing and configuring Robot Framework and PyCharm, including this one.

16.4 Libraries

There are three main categories of test libraries: standard, external, and yours. Standard libraries are "builtin", meaning you don't need to "include" those files to have access to their functions. These standard libraries are comprised of commands that execute code to handle things like verifications, conversions, and other multi-purpose functions.

Most tutorials, etc. focus on testing web applications. In order to run these tests, you need Selenium2Library. This is one of the many "external" libraries you can use. To install and use, simply run

```
pip install robotframework-selenium2library
```

External library files from which you'd like to access code must be included in the test scripts.

In addition to the standard and external libraries, you can create your own test libraries. The Robot Framework website says "it's a breeze". As would be expected, test library files you create and wish to use must be included in the test scripts.

All test libraries are written in Java or Python. To learn more about how to develop your own test libraries, check out the framework's User Guide section on test library APIs.

16.5 Other Helpful Hints

Examples of the following will be added in the near future.

- Syntax
  - In keywords, use a <TAB> on lines defining commands (the list under the keyword heading
  - When using a variable in a keyword, put two (2) spaces between the keyword and the variable
- Logic
  - It's often helpful to have a validation check at the end of a keyword's set of commands. For example, Once a web page is up, validate that a certain string you are expecting to be there does indeed exist.

16.6 Basic Project Structure

The following is the basic layout of a Robot Framework project:

```
..
+- project_root
    +- run_tests.py
    +- keywords
    |   +- lib
    +- tests
        +- test1.robot
        +- test2.robot
```

where `project_root` has the main controller for executing tasks. This might be a Python script called "run_tests.py" as in the Intro to Robot and Examples tutorial does. The subfolder `keywords` holds keyword files of the Robot Framework commands. The subfolder `lib` holds the python-based (or java-based) keyword extensions. And the subfolder `tests` holds all the test scripts for the project.

16.7 Running Tests

Much, much more will be added to this section as we learn more, but for now, here is the basic way to run a Robot test:

```
$ pybot tests/test.robot
```

This will place all the results from test.robot in the directory from which you ran the script. To change the directory where you wish the result output to reside, add the `-d` flag:

```
$ pybot -d $RESULTS_DIR tests/test.robot
```

16.8 Tutorials

- Configure Pycharm and Robot
- Intro to Robot and Examples

CHAPTER 17

Appendices

17.1 Setting up DNS using dnsmasq

DNS is designed to be a distributed, hierarchical, system of mapping names to IP addresses (and the reverse, IP addresses to names).

Note: To learn more about it, see:

1. How the Domain Name System (DNS) Works, by Verisign
2. How does DNS work?, by D.J. Berstein
3. The Domain Name System, Wikipedia

The important thing to note is that if you only have one DNS server set, and that server only serves it's own names and does not make recursive queries for domains it does not know, you will not be able to talk to many hosts on the internet.

These instructions will configure `dnsmasq` on a developer's workstation (e.g., a laptop computer) to serve as the primary DNS server for that system, using a local DNS server behind a VPN (when connected to the VPN) or recursively directing queries to Google's name servers for all other DNS requests. This should work for any DNS requests made from this client system in a reliable way.

17.1.1 Mac OS X configuration

Attention: The following instructions are partially specific to Mac OS X, which handles network configuration using **System Preferences...**

Start by creating a *Location* called *VPN" to use for controlling the DNS configuration of the Mac. Figure :ref:'networklocation' shows the location *VPN* enabled.

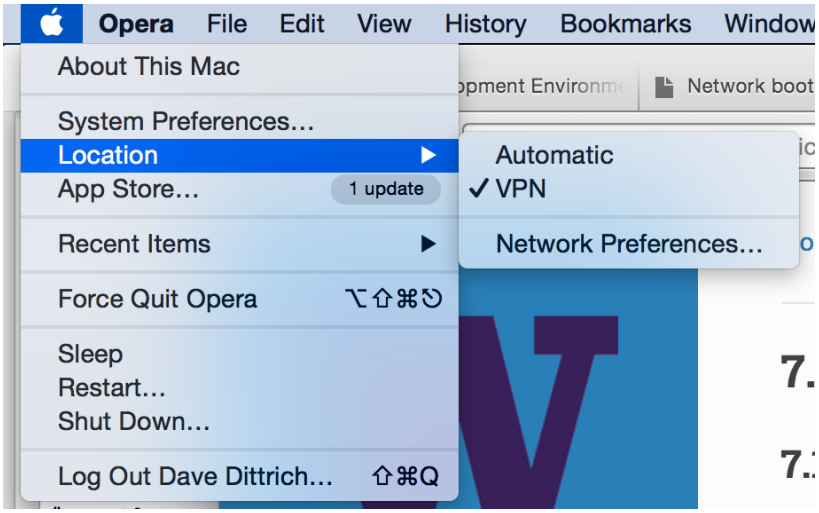


Fig. 17.1: Mac OS X Network Preferences, Location

Figure *Mac OS X Network Preferences, DNS* shows the settings for *DNS Server* and *Search*. Set *DNS Server* to only be `127.0.0.1` for force all DNS queries to *only* go to the local `dnsmasq` server. The *Search* list should include your normal domain that you want to be appended to any short (or *partially qualified*) DNS names.

When set this way, as soon as the network is enabled on any interface (be it WiFi, Bluetooth, USB ethernet, or wired ethernet), the *VPN* location will be enabled and the host's `/etc/resolv.conf` file will be set to look like this:

```
#
# Mac OS X Notice
#
# This file is not used by the host name and address resolution
# or the DNS query routing mechanisms used by most processes on
# this Mac OS X system.
#
# This file is automatically generated.
#
search devops.develop ops.develop
nameserver 127.0.0.1
```

Edit the `dnsmasq` configuration file (`/opt/local/etc/dnsmasq.conf` on the Mac, and `/etc/dnsmasq.conf` on Ubuntu 14.04). Set the following variables as shown in the examples.

1. Add a line referencing an alternative `resolve.conf` file to control upstream DNS servers.

   ```
   # Change this line if you want dns to get its upstream servers from
   # somewhere other that /etc/resolv.conf
   #resolv-file=/etc/resolv.conf
   resolv-file=/etc/resolv.dnsmasq
   ```

2. Set the `server` entries for forward lookups containing the top level domain `devops.develop` and reverse maps for the DIMS VPN network range (`192.168.88.0/24`, which is expressed as `88.168.192.in-addr.arpa` for DNS reverse mappings) as shown in the highlighted lines here:

   ```
   # Add other name servers here, with domain specs if they are for
   # non-public domains.
   #server=/localnet/192.168.0.1
   server=/devops.develop/192.168.88.101
   server=/dims-dev.devops.develop/127.0.0.1
   ```

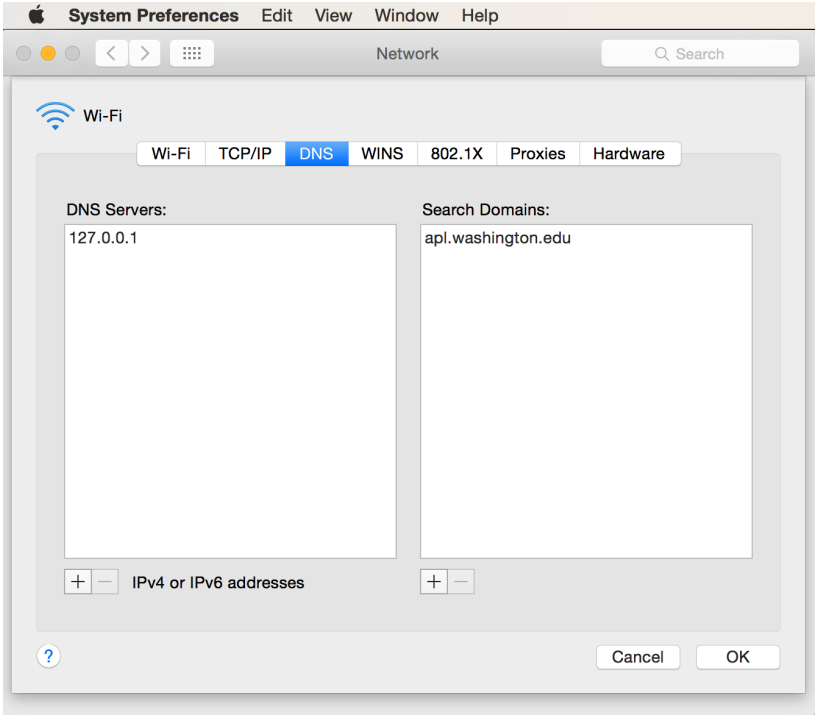


Fig. 17.2: Mac OS X Network Preferences, DNS

```
# Example of routing PTR queries to nameservers: this will send all
# address->name queries for 192.168.3/24 to nameserver 10.1.2.3
#server=/3.168.192.in-addr.arpa/10.1.2.3
server=/88.168.192.in-addr.arpa/192.168.88.101
```

Note: The second `server` line in the first set above creates a test domain `dims-dev.devops.develop` that is served by this `dnsmasq` server on the local host address. This allows you to test services running on the loopback interface.

Attention: Mac users will need to configure `dnsmasq` to run under `_mdnsresponder` account. Use the following lines:

```
# If you want dnsmasq to change uid and gid to something other
# than the default, edit the following lines.
user=_mdnsresponder
group=_mdnsresponder
```

3. If you also have a home network that you wish to associate with a specific alternative domain, use the `domain` setting as shown here:

```
# Set a different domain for a particular subnet
#domain=wireless.thekelleys.org.uk,192.168.2.0/24
domain=home,192.168.1.0/24
```

4. **Caution:** When switching a VPN connection on and off, where you are trying to use non-public DNS names served by a server behind the VPN, you may encounter a situation where queries for a non-public domain are sent to public DNS servers, which will return an "NXDOMAIN" response, which looks like this

```
Host abcquq12examfooltest.com not found: 3(NXDOMAIN)
```

The local server may cache this result. When you then connect to the VPN and regain access to the private server which should be able to now respond with the correct result, your next attempt to resolve the domain may find the cached NXDOMAIN result and tell you the domain still does not exist (when you are assuming that it does, since the VPN is now up.) This is both confusing, and frustrating, unless you are aware of how DNS caching works.

To prevent this problem, disable negative caching as follows:

```
# If you want to disable negative caching, uncomment this.
no-negcache
```

5. **Attention:** As a debugging mechanism, you may need to enable logging of DNS queries and/or DHCP transactions. Do that by uncommenting the following lines:

```
# For debugging purposes, log each DNS query as it passes through
# dnsmasq.
log-queries

# Log lots of extra information about DHCP transactions.
log-dhcp
```

6. Create the alternative `resolv.conf` file referenced in the `dnsmasq.conf` file above to have the contents shown here:

```
[dittrich@localhost etc]$ cat resolv.dnsmasq
search devops.develop ops.develop
nameserver 8.8.8.8
nameserver 192.168.88.101
nameserver 128.95.120.1
```

7. Test the configuration.

   With VPN disconnected:

```
[dittrich@localhost etc]$ dig @127.0.0.1 jira.devops.develop

; <<>> DiG 9.8.3-P1 <<>> @127.0.0.1 jira.devops.develop
; (1 server found)
;; global options: +cmd
;; connection timed out; no servers could be reached
```

   With VPN enabled:

```
[dittrich@localhost etc]$ dig @127.0.0.1 jira.devops.develop

; <<>> DiG 9.8.3-P1 <<>> @127.0.0.1 jira.devops.develop
; (1 server found)
;; global options: +cmd
;; Got answer:
```

```
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 58384
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;jira.devops.develop.    IN      A

;; ANSWER SECTION:
jira.devops.develop. 0   IN      A       192.168.88.97

;; Query time: 18 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Wed Jul  1 17:32:54 2015
;; MSG SIZE  rcvd: 60
```

17.1.2 Logging

```
On 7/2/15 6:47 AM, Linda Parsons wrote:

> Do you know where the queries are logged on os x? I can't find logs
> anywhere. I can see in /var/log/system.log where dnsmasq is stopped and
> started - that's it.
```

`syslog` and `rsyslog` are very fine-grained and controllable in terms of where logs go (i.e., which file, which logging host, both, etc.), though each program choses what facility and severity level it will log at. Here is excerpt from `dnsmasq` man page:

```
-8, --log-facility=<facility>
Set the facility to which dnsmasq will send syslog entries, this defaults
to DAEMON, and to LOCAL0 when debug mode is in operation. If the facility
given contains at least one '/' character, it is taken to be a filename,
and dnsmasq logs to the given file, instead of syslog. If the facility is
'-' then dnsmasq logs to stderr. (Errors whilst read- ing configuration
will still go to syslog, but all output from a successful startup, and all
output whilst running, will go exclusively to the file.) When logging to a
file, dnsmasq will close and reopen the file when it receives SIGUSR2.
This allows the log file to be rotated without stopping dnsmasq.

...

When it receives SIGUSR2 and it is logging direct to a file (see
--log-facility ) dnsmasq will close and reopen the log file. Note that
during this operation, dnsmasq will not be running as root. When it first
creates the logfile dnsmasq changes the ownership of the file to the
non-root user it will run as. Logrotate should be configured to create a
new log file with the ownership which matches the existing one before
sending SIGUSR2. If TCP DNS queries are in progress, the old logfile will
remain open in child processes which are handling TCP queries and may
continue to be written. There is a limit of 150 seconds, after which all
existing TCP processes will have expired: for this reason, it is not wise
to configure log- file compression for logfiles which have just been
rotated. Using logrotate, the required options are create and delay-
compress.
```

So `dnsmasq` can bypass `syslog`/`rsyslog` filters and log directly to a file.

Note: Adding the option `log-facility=/var/log/dnsmasq.log` diverts log messages into the file `var/log/dnsmasq.log`.

Caution: `dnsmasq`, when logging directly to a file, does *not* handle rolling of the log file or otherwise limiting its growth. The file will just continue to grow without bounds over time. You can rename the file at any time, then send the `SIGUSR2` signal to the `dnsmasq` process, which will open a new log file. (See the man page output above for more details.)

Note: Ok, I figured out that `dnsmasq` logs to `/var/log/debug.log` in general, which led me to realize these messages have a log level of `debug`. But on Mac OS X the default is not to log debug messages. I had to edit the `/etc/asl.conf` file to set the log level to `debug`. Then the debug messages would show up in the console using all messages. Keep the level at debug for a short time but have turned it off as it slows down the system a lot. I could see from the debug statements how the request to `127.0.0.1` were being forwarded.

Caution: Setting the full system logging level to `debug` just to get messages from one service is over-kill. It is preferable to force the specific service to log at a `facility` and/or `severity` level that is then filtered by `syslog`/`rsyslog`, allowing just those messages you want to be logged to go to a place you want them to go. The `log-faility` option above works better for this.

17.1.3 Split-Horizon DNS

Organizations often use non-routable network address ranges, as defined by RFC 1918 - Address Allocation for Private Internets, on the *internal* portion of a firewalled network that also has *external* internet-facing

The video DNS Split Brain explains some of the issues of handling DNS mappings in situations where networks are partitioned. An organization may have service domain names be the *same* to point to separate internal and external resources, even though they have completely different IP addresses. A web server, for example, may be accessible to users on the internet with limited public content, while another server that has the *same fully-qualified domain name* may be hosted on the inside of a firewall and VPN with different content that is private to the organization. Having multiple DNS servers, rather than just one DNS server, and configuring them to properly answer and/or forward DNS requests differently (depending on the *perspective* of the client making the request) adds complexity for system administration, but can simplify things from a user perspective when trying to access a resource.

References on configuring `dnsmasq` and the concept of *Split-horizon DNS* are included in the dittrich:dns Section of the home page of dittrich:homepage.

17.2 Using a Case-Sensitive sparse image on Mac OS X

At the beginning of Section *Source Code Management with Git*, a caution block describes a problem involving sharing source code repositories between systems having file systems that are *case-sensitive* with other operating systems having file systems that are *case-insensitive*.

This section provides the steps for creating a case-sensitive sparse HFS file image that can be mounted on a Mac OS X system to better integrate with Git source respositories using case-sensitive file and/or directory names in the respository.

Note: This example arbitrarily uses an 8GB sparse image. Change size as necessary for your own situation.

We are going to take the existing contents of a directory (`$HOME/dims/git` in this case) and replace it with a mounted case-sensitive journalled HFS sparse disk image. We are using a sparse image to avoid needlessly wasting space by allocating a disk image larger than is necessary.

1. Use the OS X *Disk Image* app to create a sparse image. This is shown in Figure *Creating a sparse image with Disk Utility*.

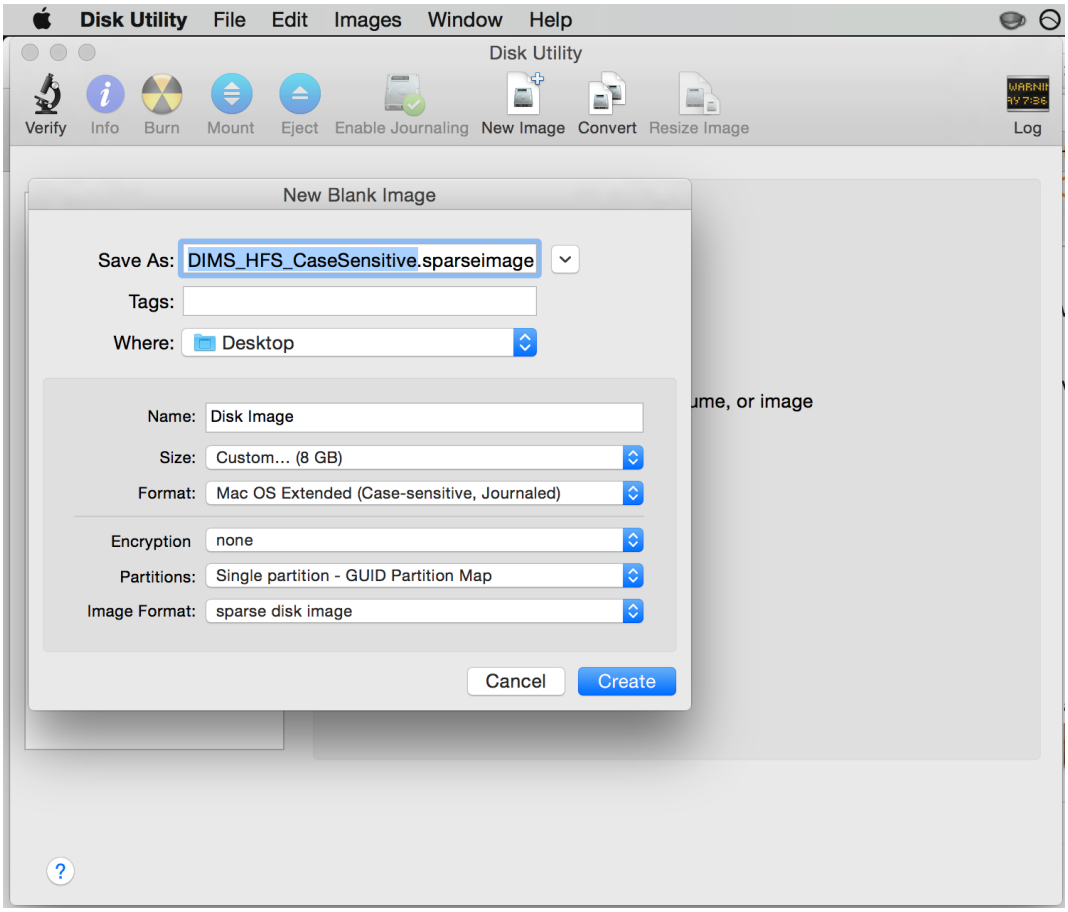


Fig. 17.3: Creating a sparse image with Disk Utility

2. Move the existing directory to another name, so we can replace that directory with an empty directory to act a mount point for our sparse bundle:

```
[dittrich@localhost ~]$ cd ~/dims
[dittrich@localhost dims]$ mv git git.tmp
[dittrich@localhost dims]$ mkdir git
```

3. Mount the sparse image using `hdiutil`:

```
1  [dittrich@localhost dims]$ hdiutil attach -mountpoint ~/dims/git ~/Desktop/DIMS_
   ↪HFS_CaseSensitive.sparseimage
2  /dev/disk3              GUID_partition_scheme
3  /dev/disk3s1            EFI
```

```
  /dev/disk3s2            Apple_HFS                          /Users/dittrich/dims/git
  [dittrich@localhost dims]$ mount
  /dev/disk1 on / (hfs, local, journaled)
  devfs on /dev (devfs, local, nobrowse)
  map -hosts on /net (autofs, nosuid, automounted, nobrowse)
  map auto_home on /home (autofs, automounted, nobrowse)
  /dev/disk2s1 on /Volumes/_mdSSD (hfs, local, nodev, nosuid, journaled, noowners)
  /dev/disk3s2 on /Users/dittrich/dims/git (hfs, local, nodev, nosuid, journaled,␣
↪noowners, mounted by dittrich)
```

4. Move the files from the temporary directory into the case-sensitive mounted volume, or re-clone any repositories that were causing problems with case-sensitive files, then delete the temporary directory.

```
[dittrich@localhost dims]$ mv git.tmp/* git
[dittrich@localhost dims]$ rmdir git.tmp
```

5. Add lines to your `~/.bash_profile` file to ensure this sparse image is mounted at the start of every initial login session.

```
mount | grep -q "$HOME/dims/git"
if [ $? -eq 1 ]; then
        hdiutil attach -mountpoint ~/dims/git ~/Desktop/DIMS_HFS_CaseSensitive.
↪sparseimage 2>/dev/null
        mount | grep -q "$HOME/dims/git"
        if [ $? -ne 0 ]; then
                echo "[---] Failed to mount ~/Desktop/DIMS_HFS_CaseSensitive.
↪sparseimage to ~/dims/git"
        fi
fi
```

You should see something like the following for the initial terminal window:

```
Last login: Fri Feb 13 04:48:45 on ttys005


[+++] DIMS shell initialization
[+++] Sourcing /opt/dims/etc/bashrc.dims.d/bashrc.dims.virtualenv ...
[+++] Activating DIMS virtual environment (dimsenv)
[+++] (Create file /Users/dittrich/.DIMS_NO_DIMSENV_ACTIVATE to disable)
[+++] Virtual environment dimsenv activated
[+++] Mounted sshfs gituser@git.devops.develop:cfg as /Users/dittrich/dims/cfg
/dev/disk3              GUID_partition_scheme
/dev/disk3s1            EFI
/dev/disk3s2            Apple_HFS                          /Users/dittrich/dims/git
/dev/disk3s2 on /Users/dittrich/dims/git (hfs, local, nodev, nosuid, journaled,␣
↪noowners, mounted by dittrich)
```

Note: If you forgot to set a volume label when you created the sparse image file in Disk Utility, and the disk image just created gets mounted as "Disk Image", you may wish to change the label. To do this, after the volume is mounted you can rename it using the command:

```
[dittrich@localhost docs (develop)]$ diskutil rename "Disk Image" DIMS_Git
Volume on disk2s2 renamed to DIMS_Git
```

If/when the sparse image becomes filled, you can compact it using `hdiutil` as described in the *superuser* post

Shrink a .sparseimage.

```
[dimsenv] dittrich@27b:~ () $ hdiutil eject /Users/dittrich/dims/git
"disk2" unmounted.
"disk2" ejected.
[dimsenv] dittrich@27b:~ () $ hdiutil compact ~/Desktop/DIMS_HFS_CaseSensitive.
↪sparseimage
Starting to compact...
Reclaiming free space...
..................................................................................
↪..................................................................................
↪..........................
Finishing compaction...
..................................................................................
↪..................................................................................
↪................................
Reclaimed 1.3 GB out of 6.2 GB possible.
```

17.3 Google Hangouts 'Original Version' Screenshare Instructions

- Everyone can do it, at the same time!
- Hover your cursor over the left hand edge of your Hangouts window.
- A menu will slide out with lots of icons.
- To screenshare, click the second icon down, a green monitor with a white arrow pointing to the right.
- You can choose to share your whole desktop or individual windows of other applications you have open on your desktop. It doesn't appear you can share all windows of an application, such as Terminal. If you have 5 Terminal windows open, you can only share 1 of them. You can open multiple tabs, and those will be shared.
- Resizing of windows works just fine when screensharing also.

Note: This document is written in Restructured Text (reST) using Sphinx.

For more information, see *Documenting DIMS Components*.

CHAPTER 18

Contact

Section author: Dave Dittrich (@davedittrich) <dittrich @ u.washington.edu>

CHAPTER 19

License


```
Berkeley Three Clause License
=============================

Copyright (c) 2013-2017 University of Washington. All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this
list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice,
this list of conditions and the following disclaimer in the documentation
and/or other materials provided with the distribution.

3. Neither the name of the copyright holder nor the names of its contributors
may be used to endorse or promote products derived from this software without
specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```