paradrop Documentation

Release 0.2.1

Paradrop Labs

December 01, 2016

Contents

This section of the documentation is devoted to describing the apps that run on Paradrop.

If this is your first time seeing Paradrop, please start with the Getting Started page.

Contents:

CHAPTER 1

Getting Started

This will quickly take you through the process of bringing up a Hello World chute in a virtual machine on your computer.

NOTE: These instructions assume you are running Ubuntu. The steps to launch a virtual machine may be different for other environments.

1.1 Environment setup

These steps wil download our router image and launch it a virtual machine.

1. Install required packages:

```
sudo apt-get install qemu-kvm
```

2. Download the latest image (paradrop_router.img.tgz) from our releases.
3. Extract the image:

```
tar xf paradrop_router.img.tgz
```

4. Launch the VM:

```
kvm -m 1024 \
-netdev user,id=net0,hostfwd=tcp::8000-:8000,hostfwd=tcp::8022-:22,hostfwd=tcp::8080-:80,hostfwd
-device virtio-net-pci,netdev=net0 -drive file=paradrop_router.img,format=raw
```

1.2 Activating your Router

Follow these steps the first time you start a new physical or virtual Paradrop router. Activation associates the router with your account on paradrop.org so that you can manage the router and install chutes from the chute store.

1. Make an account on paradrop.org if you do not have one.
2. From the Routers tab, click Create Router. Give your router a unique name and an optional description to help you remember it and click Submit.
3. On the router page, find the "Router ID" and "Password" fields. You will need to copy this information to the router so that it can connect.

4. Open the router portal at http://localhost:8080. Click the Login button and enter the information from the paradrop.org router page. You may need to refresh the page before the messages appear. **Important**: Although the Login button remains on the page, you only need to complete this step one time for a router. If the activation was successful, you should see the following messages:

```
This router is provisioned.
The HTTP connection is ready.
The WAMP connection is ready.
```

1.3 Installing Chutes

1. Make an account on paradrop.org and make sure you have an activated, online router.
2. Go to the Chute Store tab on paradrop.org. There you will find some public chutes such as the hello-world chute. You can also create your own chutes here.
3. Click on the hello-world chute, click Install, click your router's name to select it, and finally, click Install.
4. That will take you to the router page again where you can click the update item to monitor its progress. When the installation is complete, an entry will appear under the Chutes list.
5. The hello-world chute starts a webserver, which is accessible at http://localhost:8000. Once the installation is complete, test it in a web browser.

CHAPTER 2

Chute Development

Minimally, a chute has a Dockerfile, which contains instructions for building and preparing the application to run on Paradrop. A chute will usually also require scripts, binaries, configuration files, and other assets. For integration with the Paradrop toolset, we highly recommend developing a chute as a GitHub project, but other organization methods are possible.

We will examine the hello-world chute as an example of a complete Paradrop application.

2.1 Structure

Our hello-world chute is a git project with the following files:

```
chute/index.html
Dockerfile
README.md
```

The top-level contains a README and a special file called "Dockerfile", which will be discussed below. As a convention, we place files that will be used by the running application in a subdirectory called "chute". This is not necessary but helps keep the project organized. Valid alternatives include "src" or "app".

2.2 Dockerfile

The Dockerfile contains instructions for building and preparing an application to run on Paradrop. Here is a minimal Dockerfile for our hello-world chute:

```
FROM nginx
ADD chute/index.html /usr/share/nginx/html/index.html
```

FROM nginx

The FROM instruction specifies a base image for the chute. This could be a Linux distribution such as "ubuntu:14.04" or an standalone application such as "nginx". The image name must match an image in the Docker public registry. We recommend choosing from the official repositories. Here we use "nginx" for a light-weight web server.

ADD chute/index.html index.html

The ADD instruction copies a file or directory from the source repository to the chute filesystem. This is useful for installing scripts or other files required by the chute and are part of the source repository. The <source> path should be inside the respository, and the <destination> path should be an absolute path or a path inside the chute's working directory. Here we install the index.html file from our source repository to the search directory used by nginx.

Other useful commands for building chutes are RUN and CMD. For a complete reference, please visit the official Dockerfile reference.

Here is an alternative implementation of the hello-world Dockerfile that demonstrates some of the other useful instructions.

```
FROM ubuntu:14.04
RUN apt-get update && apt-get install -y nginx
ADD chute/index.html /usr/share/nginx/html/index.html
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

Here we use a RUN instruction to install nginx and a CMD instruction to set nginx as the command to run inside the chute container. Using "ubuntu:14.04" as the base image gives access to any packages that can be installed through apt-get.

2.3 Persistent Data

Each running chute has a persistent data storage that is not visible to other chutes. By default the persistent data directory is named "/data" inside the chute's filesystem. Files stored in this directory will remain when upgrading or downgrading the chute and are only removed when uninstalling the chute.

2.4 System Information

The Paradrop instance tools share system information with chutes through a read-only directory named "/paradrop". Chutes that are configured with a WiFi access point will find a file in this directory that lists wireless clients. In future versions there will also be information about Bluetooth and other wireless devices.

2.4.1 dnsmasq-wifi.leases

This file lists client devices that have connected to the WiFi AP and received a DHCP lease. This is a plain text file with one line for each device containing the following space-separated fields.

1. DHCP lease expiration time (seconds since Unix epoch).
2. MAC address.
3. IP address.
4. Host name, if known.
5. Client ID, if known; the format of this field varies between devices.

The following example shows two devices connected to the chute's WiFi network.

```
1480650200 00:11:22:33:44:55 192.168.128.130 android-ffeeddccbbaa9988 *
1480640500 00:22:44:66:88:aa 192.168.128.170 someones-iPod 01:00:22:44:66:88:aa
```

CHAPTER 3

Chute Deployment

Starting with version 0.2, developers should use our website, paradrop.org for creating chutes and installing them on Paradrop routers. The command line tools from Paradrop 0.1 are not currently supported.

As our website is under development, please refer to the instructions on the website or contact us for help with making and installing a chute.

CHAPTER 4

The Paradrop Instance System

This section focuses on the *Instance Tools*. This is the set of daemons and tools required to allow the Paradrop platform to function on virtual machines and real hardware. Use the information below to learn about Snappy Ubuntu and how we leverage it to create next generation smart routers.

Contents:

4.1 Build System

Paradrop includes a set of build tools to make development as easy as possible.

Currently this system takes the form of a bash script that automates installation and execution, but in time this may evolve into a published python package. This page outlines the steps required to manually build the components required to develop with paradrop.

Components in the build process:

- *Installing and running Ubuntu Snappy*
- *Building paradrop*
- *Installing paradrop*
- **'Creating chutes'_**

We recommend using Ubuntu 14.04 as the build environment for this version of Paradrop. Ubuntu 16.04 will not work because the snappy development tools have changed. The next release of Paradrop will use the new tools.

You will only need to follow these instructions if you will be making changes to the Paradrop instance tools. Otherwise, you can use our pre-built Paradrop disk image. Please visit the Getting Started page.

4.1.1 Installing and running Ubuntu Snappy

Snappy is an Ubuntu release focusing on low-overhead for a large set of platforms. These instructions are for getting a Snappy instance up and running using 'kvm'.

Download and unzip a snappy image:

```
wget http://releases.ubuntu.com/15.04/ubuntu-15.04-snappy-amd64-generic.img.xz
unxz ubuntu-15.04-snappy-amd64-generic.img.xz
```

Launch the snappy image using kvm:

```
kvm -m 512 -redir :8090::80 -redir :8022::22 ubuntu-15.04-snappy-amd64-generic.img
```

Connect to local instance using ssh:

```
ssh -p 8022 ubuntu@localhost
```

4.1.2 Building paradrop

Snappy is a closed system (by design!). Arbitrary program installation is not allowed, so to allow paradrop access to the wide world of `pypi` the build system relies on two tools.

- `virtualenv` is a tool that creates encapsulated environments in which python packages can be installed.
- `pex` can compress python packages into a zip file that can be executed by any python interpreter.
- `snappy` is a tool for building snap packages. *Note:* Ubuntu 16.04 uses snapcraft instead, which produces incompatible packages.

First, set *DEV_MACHINE_IP=paradrop.org* in pdbuild.conf. The build script will refuse to run if this variable is not set.

Install the necessary development tools:

```
./pdbuild.sh setup
```

Build the Paradrop snap package:

```
./pdbuild.sh build
```

4.1.3 Installing paradrop

Install dependencies on the virtual machine:

```
./pdbuild.sh install_deps
```

Install the newly created Paradrop snap package:

```
./pdbuild.sh install_dev
```

4.2 Snappy Confinement

Snappy confines running applications in two ways: directory isolation and mandatory access control (MAC). Directory isolation means the application cannot leave its installed directory. MAC means the application cannot execute any system commands or access any files it does not have explicit, predetermined permissions to.

MAC is the more serious hurdle for paradrop development. Snaps declare permissions through an AppArmor profile.

4.2.1 Getting started with Profile Generation

Install tools and profiles:

```
sudo apt-get install apparmor-profiles apparmor-utils
```

List active profiles:

```
sudo apparmor_status
```

Profiles in complain mode log behavior, while those in enforce mode actively restrict it.

The following steps assume paradrop is installed on the system and not on a virtualenv.

Create a new, blank profile:

```
cd /etc/apparmor.d/
sudo aa-autodep paradrop
```

Use aa-complain to put the profile in complain mode:

```
sudo aa-complain paradrop
```

Excercise the application! AppArmor will surreptitiously watch the program in the background and log all behavior. Once finished, use the following command to go through the resulting requests, approve or deny them, and autogenerate a profile:

```
sudo aa-logprof
```

4.3 Documentation and tests

Documentation is handled by sphinx and readthedocs.

Testing is a joint effort between nosetests, travis-ci, and coveralls.

4.3.1 Documentation

Information about docs creation, management, and display.

Sphinx reads files in reStructuredText and builds a set of HTML pages. Every time a new commit is pushed to github, readthedocs automatically updates documentation.

Additionally, sphinx knows all about python! The directives `automodule`, `autoclass`, `autofunction` and more instruct sphinx to inspect the code located in `src/` and build documentation from the docstrings within.

For example, the directive `.. automodule:: paradrop.backend` will build all the documentation for the given package. See google for more instructions.

All docstring documentation is rebuilt on every commit (unless there's a bug in the code.) Sphinx does not, however, know about structural changes in code! To alert sphinx of these changes, use the `autodoc` feature:

```
sphinx-apidoc -f -o docs/api paradrop/paradrop/
```

This scans packages in the `src/paradrop` directory and creates .rst files in `docs/api`. The root file `index.rst` links to `modules.rst`, connecting the newly generated api code with the main documentation.

To create the documentation locally, run:

```
cd docs
make html
python -m SimpleHTTPServer 9999
```

Open your web browser of choice and point it to http://localhost:9999/_build/html/index.html.

4.3.2 Testing

As mentioned above, all testing is automatically run by travis-ci, a continuous integration service.

To manually run tests, install nosetest:

```
pip install nose
```

Install the required packages:

```
pip install -r docs/requirements.txt
```

Run all tests:

```
nosetests
```

Well thats easy. How does nose detect tests? All tests live in the `tests/` directory. Nose adheres to a simple principle: anything marked with `test` in its name is most likely a test. When writing tests, make sure all functions begin with `test`.

Coverage analysis detects how much of the code is used by a test suite. If the result of the coverage is less than 100%, someone slacked. Install coveralls:

```
pip install coveralls
```

Run tests with coverage analysis:

```
nosetests --with-coverage --cover-package=paradrop
```

CHAPTER 5

Architecture

This section details some of the non-obvious architectural features of paradrop.

This is a work in progress. Check back later!

CHAPTER 6

Known Issues

Please check here for issues during setup.

6.1 Issues using pdbuild.sh

These issues are related to the *Instance Tools* found on github.

6.1.1 Issue 1: `pdbuild.sh install` fails

Docker snap is missing inside of virtual router, run `pdbuild.sh install_deps`:

```
issues while running ssh command: Installing /tmp/paradrop_0.1.0_all.snap
2015/08/11 21:22:57 Signature check failed, but installing anyway as requested
/tmp/paradrop_0.1.0_all.snap failed to install: missing frameworks: docker
```

6.1.2 Issue 2: `pdbuild.sh install` fails

This is a known issue for the Paradrop team, if you get this please email us at developers@paradrop.io:

```
Installing paradrop_0.1.0_all.snap from local environment

issues while running ssh command: Installing /tmp/paradrop_0.1.0_all.snap
2015/08/11 21:29:48 Signature check failed, but installing anyway as requested
/tmp/paradrop_0.1.0_all.snap failed to install: [start paradrop_pdconfd_0.1.0.service]
failed with exit status 1: Job for paradrop_pdconfd_0.1.0.service failed.
See "systemctl status paradrop_pdconfd_0.1.0.service" and "journalctl -xe" for details.
```

6.1.3 Issue 3: `pdbuild.sh up` fails

This is very common and will happen if you delete your VM and setup a fresh one, which will have a different key. The solution is simple and is stated in the error message. Follow the instructions to remove the key from your known_hosts file.

```
Failed to setup keys: failed to setup keys: issues while running ssh command:
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
```

```
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that a host key has just been changed.
The fingerprint for the ECDSA key sent by the remote host is
e6:ec:b1:93:7d:91:84:50:19:36:14:8e:ce:ef:6a:0b.
Please contact your system administrator.
```

6.2 Issues with the hardware or operating system

6.2.1 Issue 1: Docker fails to start after a reboot

This can happen if the 'docker.pid' file was not properly cleaned up, which causes the docker daemon to conclude that it is already running.

To fix this, remove the pid file on the router and reboot.

```
sudo rm /var/lib/apps/docker/1.11.2/run/docker.pid
sudo reboot
```

6.2.2 Issue 2: WiFi devices are not detected after a reboot

Occasionally, when routers start up the WiFi devices are not detected properly. When this happens the command `iw dev` will display nothing instead of the expected devices. This is usually remedied by rebooting.

CHAPTER 7

Frequently Asked Questions

Please check here for any FAQ's.

CHAPTER 8

Our Code

8.1 Subpackages

8.1.1 paradrop.backend package

Subpackages

paradrop.backend.exc package

Submodules

paradrop.backend.exc.executionplan module

paradrop.backend.exc.files module

paradrop.backend.exc.name module

paradrop.backend.exc.plangraph module

paradrop.backend.exc.resource module

paradrop.backend.exc.runtime module

paradrop.backend.exc.state module

paradrop.backend.exc.struct module

paradrop.backend.exc.traffic module

Module contents

paradrop.backend.fc package

Submodules

paradrop.backend.fc.chutestorage module

paradrop.backend.fc.configurer module

paradrop.backend.fc.updateObject module

Module contents

paradrop.backend.pdconfd package

Subpackages

paradrop.backend.pdconfd.config package

Submodules

paradrop.backend.pdconfd.config.base module

paradrop.backend.pdconfd.config.command module

paradrop.backend.pdconfd.config.dhcp module

paradrop.backend.pdconfd.config.firewall module

paradrop.backend.pdconfd.config.manager module

paradrop.backend.pdconfd.config.network module

paradrop.backend.pdconfd.config.wireless module

Module contents

Submodules

paradrop.backend.pdconfd.client module

paradrop.backend.pdconfd.main module

Module contents

paradrop.backend.pdfcd package

Submodules

paradrop.backend.pdfcd.apichute module

paradrop.backend.pdfcd.apiinternal module

paradrop.backend.pdfcd.apiutils module

paradrop.backend.pdfcd.server module

Module contents

Module contents

8.1.2 paradrop.lib package

Subpackages

paradrop.lib.api package

Submodules

paradrop.lib.api.pdapi module

paradrop.lib.api.pdrest module

Module contents

paradrop.lib.config package

Submodules

paradrop.lib.config.configservice module

paradrop.lib.config.devices module

paradrop.lib.config.dhcp module

paradrop.lib.config.dockerconfig module

paradrop.lib.config.firewall module

paradrop.lib.config.network module

paradrop.lib.config.osconfig module

paradrop.lib.config.pool module

paradrop.lib.config.uciutils module

paradrop.lib.config.wifi module

Module contents

paradrop.lib.utils package

Submodules

paradrop.lib.utils.addresses module

paradrop.lib.utils.dockerapi module

paradrop.lib.utils.pdos module

paradrop.lib.utils.pdosq module

paradrop.lib.utils.restart module

paradrop.lib.utils.storage module

paradrop.lib.utils.uci module

Module contents

8.2 Submodules

8.3 paradrop.main module

8.4 Module contents

CHAPTER 9

Paradrop

Paradrop is a software platform that enables apps to run on Wi-Fi routers. We call these apps “chutes” like a parachute. The name Paradrop comes from the fact that we are enabling the ability to “drop” supplies and resources (“apps”) into a difficult and not well-travelled environment - the home.

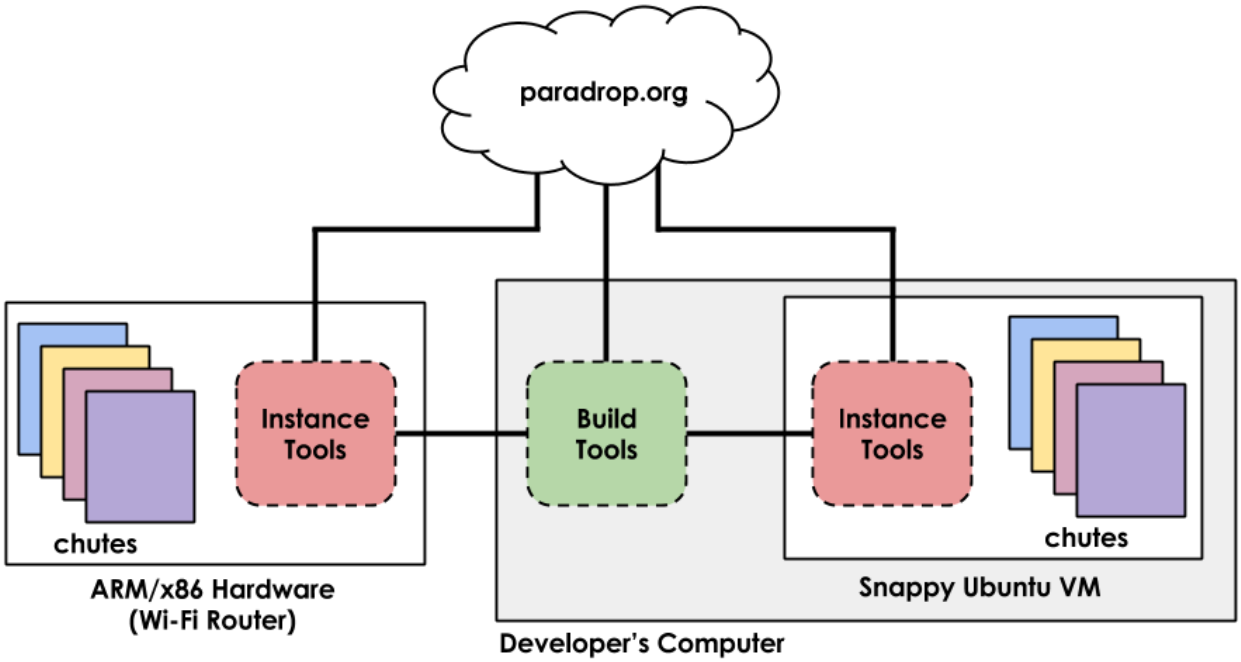
Our early versions of Paradrop relied on OpenWrt, however we are revamping the platform and tailoring it towards a broader developer community. Paradrop now runs on top of Snappy Ubuntu, a trimmed-down and secured operating system that can run on ARM and x86. We also enable our apps through containerization by leveraging Docker.

CHAPTER 10

The Paradrop workflow

There are two components to the Paradrop platform:

- The build tools - our CLI that enables registration, login, and control. With version 0.2 and up, Paradrop routers can be managed through our cloud management service instead of the CLI.
- The instance tools - our configuration daemons and tools to launch apps on hardware.



As you can see from the image above, we will refer to *Build Tools* when we talk about the CLI program running on your development computer that controls and communicates with the rest of the Paradrop platform. Treat this tool as your window into the rest of the Paradrop world. Our *Instance Tools* leverage programs like Docker to allow Paradrop apps to run on router hardware. This "hardware" could be a Raspberry Pi, or even a virtual machine on your computer that acts as a router (which is why we call it an Instance sometimes). Using Paradrop, you can actually plug in a USB Wi-Fi adapter and turn a virtual machine on your computer into a real router with our platform!

CHAPTER 11

Getting Started

Please visit the Getting Started page for a quick introduction to Paradrop.

CHAPTER 12

Where to go from here?

We have advanced app examples found under Apps on Paradrop. If you are interested in working on the instance side of paradrop (our github code) than check out: The Paradrop Instance System.

CHAPTER 13

What if I don't have Ubuntu?

With version 0.2 and up, all of Paradrop's capabilities can be managed through our web-based service at paradrop.org.