bpftrace Hands-On Labs

bpftrace is a dynamic tracing technology on Linux that can revolutionise the way we view our software and systems. It layers on top of existing BPF goodness to provide a simple to use scripting language that will give you new and fresh insights into the runtime behaviour of the systems you create. It was initially developed by Alistair Robertson and now has a solid developer community on GitHub.

This hands-on lab is designed to be completed in half a day though that may vary depending upon how thorough you want to be with the exercises. It has been written as a kind of "hands-on bootstrapping" guide for the bpftrace beginner and aims to give you enough information to kick-start your productivity with bpftrace. Note that it is by no means a complete treatment of the subject and many things are missed out for the sake of brevity.

We suggest that you complete the labs in the order they are presented below. The sections in each lab generally contain a mix of presented information and suggested exercises. We strongly suggest that you manually run any bpftrace scripts that are used in explanations and feel free to modify them and see what happens!

Finally, note that the lab is designed to be ran stand alone and without a lecturer but it is probably at its best when undertook in a shared learning environment (i.e, having people around to discuss your problems and ideas with!). For help later or on your own, please feel free to post any and all bpftrace questions in the bpftrace users Workplace group.

Prerequisites

1. Currently this lab can only be ran on a Meta devserver so make sure you've bought yours to the party.
2. Unless specified differently, all commands will be executed as the `root` user so ensure you have access to that account.

Environment Setup

1. The 'fb-bpftrace' package should already be installed on your devserver. To verify this execute the following command:

```
# rpmquery fb-bpftrace
```

If the above `rpmquery` command returns "`package fb-bpftrace is not installed`" then manually install the package:

```
# dnf install -q -y fb-bpftrace
```

1. Copy `bpftrace-hol.tar.gz` to your dev server:

```
# cp /mnt/persistent-public/jonhaslam/public_html/bpftrace-hol.tar.gz ~
```

1. If you want to view the pdf's for the course on your laptop then copy `bpftrace-hol.tar.gz` from your dev server to you laptop:

```
# scp <unixname>@<hostname>:/home/<unixname>/bpftrace-hol.tar.gz .
```

or download the archive via https:

```
# wget https://home.fburl.com/~jonhaslam/bpftrace-hol.tar.gz .
```

1. Unpack the archive:

```
# cd ~
# tar zxvf bpftrace-hol.tar.gz
bpftrace-hol/
bpftrace-hol/load_generators/
bpftrace-hol/load_generators/kprobes/
bpftrace-hol/load_generators/kprobes/kprobeme
bpftrace-hol/load_generators/syscalls/
bpftrace-hol/load_generators/syscalls/closeall
bpftrace-hol/load_generators/syscalls/tempfiles
bpftrace-hol/load_generators/syscalls/mapit
bpftrace-hol/load_generators/uprobes/
bpftrace-hol/load_generators/uprobes/uprobeme
bpftrace-hol/load_generators/usdt/
bpftrace-hol/load_generators/usdt/thrift/
bpftrace-hol/load_generators/usdt/thrift/cpp2/
bpftrace-hol/load_generators/usdt/thrift/if/
bpftrace-hol/load_generators/usdt/usdt-passwd
bpftrace-hol/load_generators/utils/
bpftrace-hol/load_generators/utils/allprobes.py
bpftrace-hol/bpfhol
bpftrace-hol/bpftrace.pdf
bpftrace-hol/kprobe.pdf
bpftrace-hol/README.pdf
bpftrace-hol/syscalls.pdf
bpftrace-hol/uprobe.pdf
bpftrace-hol/usdt.pdf

# cd bpftrace-hol
```

1. You're good to go!

NOTE: a number of exercises in the lab will make use of the `bpfhol` binary that is located at the top level directory of the `bpftrace-hol` package install. Its purpose is to run load generators in the background that aid in demonstrating features of the bpftrace language. When executed, the `bpfhol` binary presents the following menu:

```
1. syscalls
2. kprobes
3. usdt
4. uprobes
8. stop current generator
9. exit
```

Simply select the integer value corresponding to the area you have been told to run load generators for, e.g., `1` for syscalls, `2` for kprobes, etc. . If at any time you're not sure whether you already have load generators running you can simply select option `8` to kill all existing load generators that may be running.

Some of the lab exercises will give you a small hint as to what bpftrace language primitive to use to solve them. If this is the case then it is generally expected that you will look up the language feature in the online reference guide.

Example solutions for all lab exercises are provided in the solutions document.

Please use your instructor to discuss any issues or problems you may have. Everyone including them is on a learning curve with bpftrace so your question or problem will always be valuable.

Finally, these pdf documents have been converted from github markdown to pdf using pandoc. While pandoc does an analyzing job at this there are some areas where it gets a bit confused, especially marking up code blocks. Apologies in advance for any formatting that looks awkward.

Labs