

Open in app ↗



Search



★ Get unlimited access to the best of Medium for less than \$1/week. [Become a member](#)

Setting up eBPF in macOS: A Beginner's Guide



kalaiarasan balaraman

4 min read · May 29, 2023



Listen



Share

... More

eBPF is a revolutionary technology that is currently transforming observability, security, and networking.

Short History Of it:

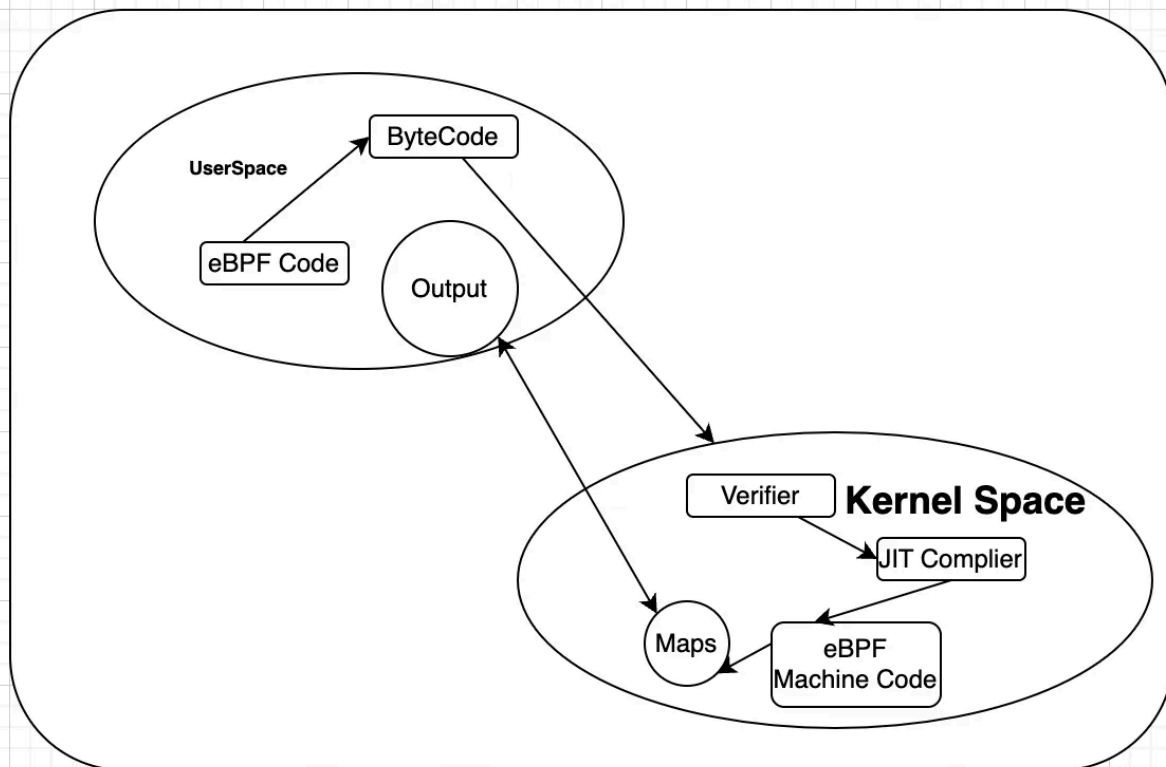
Back in 1992, Steven McCanne and Van Jacobson wrote a paper called “The BSD Packet Filter.” They wanted to make a way to filter packets from a network interface. They created something called the Berkeley Packet Filter (BPF), which is like a special computer program inside the kernel that can filter packets. BPF turned out to be useful not only for filtering packets, but also for tracing, measuring performance, and improving security.

Then, in 2014, two other people named Alexei Starovoitov and Daniel Borkmann started working on making BPF even better. They called it extended BPF (eBPF). eBPF can still filter packets like before, but it can also do other things like packet tracing, performance measurement, and major advantage is run program in userspace without changing kernel code.

How its works:

An eBPF program works when something specific happens, like a particular event occurs. We write eBPF programs using C Program, and then we compile it into eBPF bytecode using a tool called LLVM. Alternatively, we can use eBPF programs with projects like Cilium, bcc, and bpftrace, which make things easier for us. Once the program is compiled into eBPF bytecode, it gets loaded into the kernel and checked

for correctness. If everything is fine, the eBPF bytecode is translated into native code by the JIT compiler and executed.



In the picture, you can see something called eBPF maps. These maps are used by eBPF programs to gather, store, and share data Between kernel and userspace . There are different kinds of eBPF maps, like hash tables, arrays, and ring buffers.

Helper functions, tail, and function calls:

Helper functions are used to access functions in the kernel. Tail calls allow eBPF programs to call other eBPF programs. Function calls are used to call other functions within the eBPF program.

Setting up eBPF in macOS

Step1 : Hypervisor.framework to run VMs. To install it, you can use

```
brew install lima
```

Step2: Create below config

```

images:
# Try to use release-yyyyMMdd image if available. Note that release-yyyyMMdd wi
- location: "https://cloud-images.ubuntu.com/releases/22.04/release/ubuntu-22.0
  arch: "x86_64"
- location: "https://cloud-images.ubuntu.com/releases/22.04/release/ubuntu-22.
  arch: "aarch64"

mounts:
- location: "~"
  writable: true
- location: "/tmp/lima"
  writable: true
provision:
- mode: system
  script: |
    apt-get update
    apt-get install -y apt-transport-https ca-certificates curl clang llvm jq
    apt-get install -y libelf-dev libpcap-dev libbfd-dev binutils-dev build-ess
    apt-get install -y linux-tools-common linux-tools-5.15.0-41-generic bpfcc-t
    apt-get install -y python3-pip
    apt-get install --yes bsduils
    apt-get install --yes build-essential
    apt-get install --yes pkgconf
    apt-get install --yes llvm-12 clang-12
    apt-get install --yes clang-format-12
    apt-get install --yes zlib1g-dev libelf-dev
    apt-get install --yes protobuf-compiler
    apt-get install bpfcc-tools linux-headers-$(uname -r)

    sudo snap install --devmode bpftool

    # it downloads binaries with version appended
    # like llvm-strip-12, clang-12 etc
    # bpf stuff uses plain names like llvm-strip, clang and fails
    # to make them use this creating soft links with plain names
    for tool in "clang" "llc" "llvm-strip"
    do
        path=$(which $tool-12)
        sudo ln -s $path ${path%-*}
    done

    # uname -r returns kernel version
    # need linux-tools for kernel specific
    apt-get install --yes linux-tools-$(uname -r)

    # keep gp off, self signed cert issue else it'll fail to download
    # or add --no-check-certificate
    wget --quiet https://golang.org/dl/go1.20.1.linux-arm64.tar.gz

```

```
tar -C /usr/local -xzf go1.20.1.linux-arm64.tar.gz
echo 'export PATH=$PATH:/usr/local/go/bin' >> ~/.profile
```

Step 3: Just run `limactl start --name=ubuntu ubuntu-vm.yml` to create vm

Step4 : Start VM by `limactl shell ubuntu`

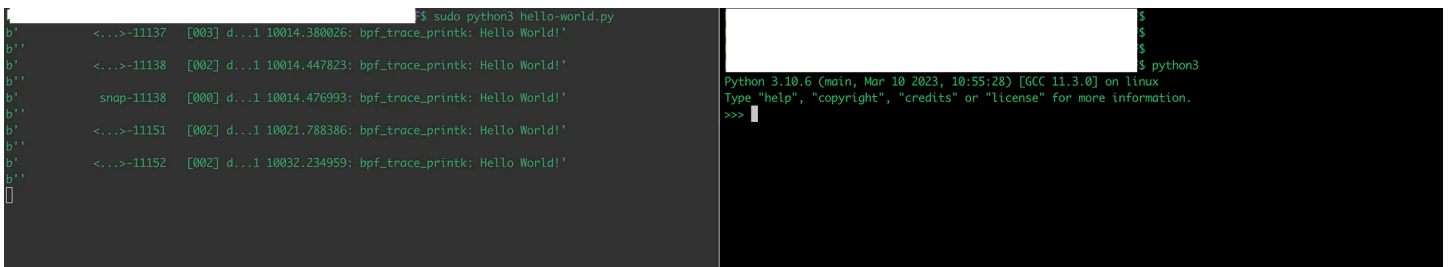
Step5: Create simple *hello-world.py* ebpf program and start by using `sudo python3 hello-world.py`

Below hello world eBPF code will starts a when new program executes.

```
#!/usr/bin/python3
from bcc import BPF
program = """
int hello(void *ctx) {
    bpf_trace_printk("Hello World!\\n");
    return 0; }
"""

b = BPF(text=program)
syscall = b.get_syscall_fnname("execve")
b.attach_kprobe(event=syscall, fn_name="hello")
b.trace_print()
```

Step6: Output



```
$ sudo python3 hello-world.py
<...>-11137 [003] d...1 10014.380026: bpf_trace_printk: Hello World!
<...>-11138 [002] d...1 10014.447823: bpf_trace_printk: Hello World!
snap-11138 [000] d...1 10014.476993: bpf_trace_printk: Hello World!
<...>-11151 [002] d...1 10021.788386: bpf_trace_printk: Hello World!
<...>-11152 [002] d...1 10032.234959: bpf_trace_printk: Hello World!
```

```
$ python3
Python 3.10.6 (main, Mar 10 2023, 10:55:28) [GCC 11.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Ebpf

Sre

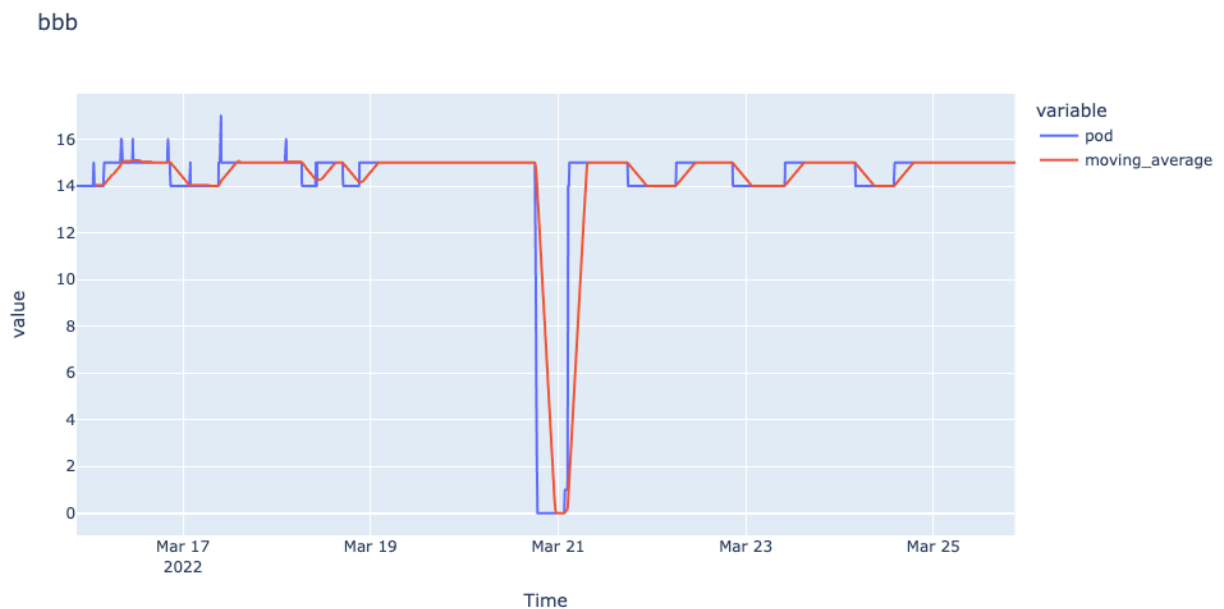
Observability

[Edit profile](#)

Written by kalaiarasan balaraman

3 Followers

More from kalaiarasan balaraman



 kalaiarasan balaraman

Building TimeSeries ML model with Prometheus DevOps Dataset

Below are Easy steps to build a model with Pycaret AutoML

2 min read · Apr 20, 2022



[See all from kalaiarasan balaraman](#)

Recommended from Medium



HashiCorp Terraform



Eric Larssen in Real Kinetic Blog

It's Time to Retire Terraform

Terraform exists in many people's hearts much like a friend or a loved one, or maybe even an enemy. Whether it's your job to maintain the...

10 min read · Apr 23, 2024



488



34



Current support features

Userspace eBPF shared memory map types:

- BPF_MAP_TYPE_HASH
- BPF_MAP_TYPE_ARRAY
- BPF_MAP_TYPE_RINGBUF
- BPF_MAP_TYPE_PERF_EVENT_ARRAY
- BPF_MAP_TYPE_PERCPU_ARRAY
- BPF_MAP_TYPE_PERCPU_HASH

User-kernel shared maps:

- BPF_MAP_TYPE_HASH
- BPF_MAP_TYPE_ARRAY
- BPF_MAP_TYPE_PERCPU_ARRAY
- BPF_MAP_TYPE_PERF_EVENT_ARRAY

Prog types can attached in userspace:

- tracepoint:raw_syscalls:sys_enter
- tracepoint:syscalls:sys_exit_*
- tracepoint:syscalls:sys_enter_*
- uretprobe:*
- uprobe:*

You can also define **other static tracepoints** and prog types in userspace app.

Support **22 kernel helper functions**

Support **kernel** or **userspace verifier**

Test JIT with **bpf_conformance**



yunwei37

bpftime: Extending eBPF from Kernel to User Space

Yu Sheng Zheng, Yu Tong

12 min read · Jan 15, 2024



11



4



Lists



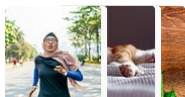
Staff Picks

634 stories · 945 saves



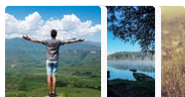
Stories to Help You Level-Up at Work

19 stories · 595 saves



Self-Improvement 101


20 stories · 1722 saves



Productivity 101

20 stories · 1601 saves



 Darek Barecki

k8spacket is fully based on eBPF right now

k8spacket uses eBPF tracepoint and Traffic Control qdisc filters to collect information about TCP traffic and TLS connection metadata.

5 min read · Mar 9, 2024



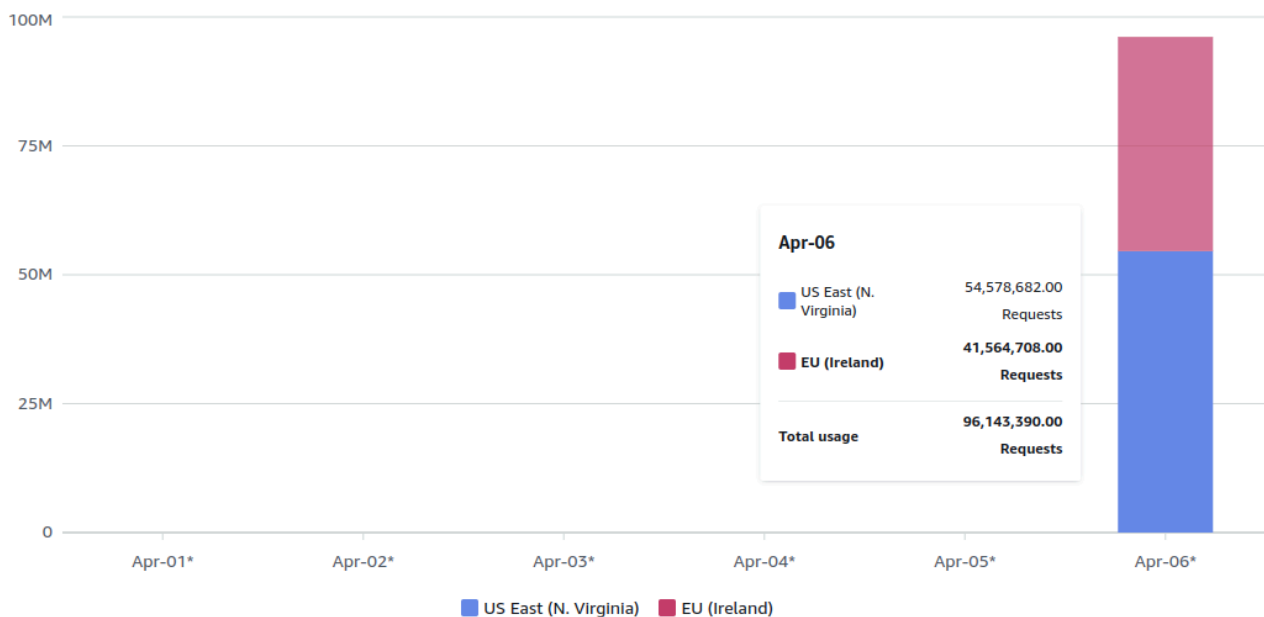
70



1



Usage (Requests)



 Maciej Pocwierz

How an empty S3 bucket can make your AWS bill explode

Imagine you create an empty, private AWS S3 bucket in a region of your preference. What will your AWS bill be the next morning?

4 min read · 5 days ago



7.9K



103



eBPF: Up and Running Part 1



Brendan Kamp in The Green Coder

eBPF Up and Running Part One

Walking through a simple BPF program from start to finish in C

8 min read · Feb 7, 2024



11





 Addo Zhang

Quick Exploration of Tetragon—A Security Observability and Execution Tool Based on eBPF

Tetragon is a versatile security observability and runtime policy enforcement tool that employs eBPF to apply policies and filtering...

6 min read · Nov 19, 2023

 3 

See more recommendations