BPF

- Berkeley Packet Filter

→ EBPF : extended BPF

→ allows to run sandboxed program directly within Kernel

→ Think of it as a mini-virtual machine that allows you to write and execute program that can hook into various Kernel events:

→ It can hook into n/w events, system calls, functions entry / exit, Kernel tracepoint, hardware events

→ way is eBPF so powerful ?

- traditional perf monitoring tools rely on user space applications to collect data from Kernel using interfaces like /proc, sysfs

- this creates overhead due to context switching and copying large amount of data

- eBPF provides its Kernel executions and

provides programmability

→ Some Terms to get started on eBPF

- Read the concepts of BPF and difference between cBDF and eBPF
- source eBPF.io and Brandon Gregg's blog
- important terms like hook, map. helper, verifier and JIT compiler

→ Tools to get started with:

- BCC :
- bpf trace :
- Many common performance issues can be diagnosed with bpf trace
- BCC : set of predefined tools that also allows you to write custom program in python and C.

→ Need to install bpftrace and bcc tools using apt-get in linux systems

using bpftrace

→ bpftrace can be used to trace 'open' using 'bpftrace -e ....')

→ Need to do

TODO

→ For the bcc tools, there are prebuilt tools like execsnoop, opensnoop, biosnoop, tcplife etc.