27 Docker 容器监控方案概览

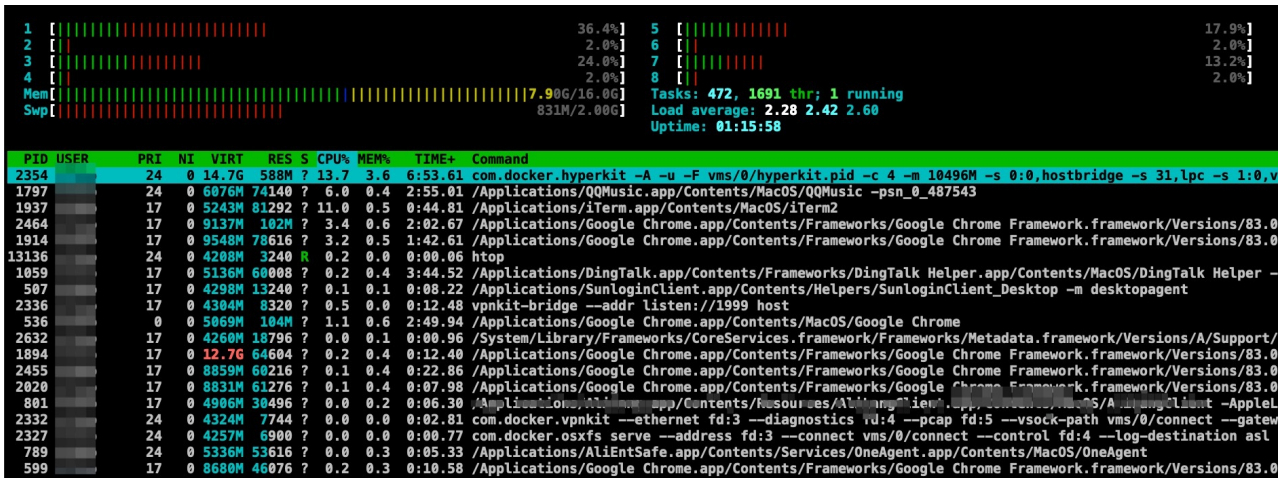
更新时间：2020-09-16 10:01:52

> 自信和希望是青年的特权。——大仲马

Docker 容器监控

这篇文章介绍了一下如何对 Docker 容器进行监控。

在正式说 Docker 容器监控之前，我们先来说一下简单的监控。监控一般根据监控目标的不同而不同，比如我们要监控一台主机，那么我们需要监控这台集群的内存和 CPU 使用，内存包括总内存、当前使用内存、剩余内存、甚至 Cache/Buffer 等，CPU 指标包括 CPU 使用率、CPU 用户态使用率、内核态使用率等。如果我们要监控一个进程，那么我们需要监控这个进程的 CPU 使用率和内存使用率等。比如我们在生产环境遇到服务器的 CPU 飙升到一个比较高的值，我们就需要知道各个进程的 CPU 使用率。下图是我本机的 htop （htop 命令类似系统命令 top）命令的显示截图。



那我们监控 Docker 容器要监控的一部分主要信息就是各个容器的资源使用情况，比如 CPU、内存等。下面介绍一下监控 Docker 容器的常用方式。

1. Docker Stats

首先介绍的是 Docker 自带的原生命令 stats，我们可以通过命令 `docker stats --help` 查看 stats 命令的介绍。

```
~ docker stats --help

Usage: docker stats [OPTIONS] [CONTAINER...]

Display a live stream of container(s) resource usage statistics

Options:
  -a, --all            Show all containers (default shows just running)
      --format string  Pretty-print images using a Go template
      --no-stream      Disable streaming stats and only pull the first result
      --no-trunc       Do not truncate output
```

如上所示，stats 命令用来实时显示容器的资源使用统计。我们下面以一个运行中的 busybox 容器为例 `docker stats --all <busybox-container-id>` 。显示的资源使用主要包括 CPU、内存、网络 IO、磁盘 IO 等。

```
CONTAINER ID   NAME                CPU %   MEM USAGE / LIMIT   MEM %   NET I/O   BLOCK I/O   PIDS
5b9e540eb9dd   upbeat_antonelli    0.00%   332KiB / 9.983GiB   0.00%   936B / 0B   0B / 0B     1
```

Docker Stats 命令用来监控容器有点类似系统自带的 top 命令，用来实时查看一个运行中的容器的资源使用情况，优点是命令是原生命令，简单易用，缺点也很明显，历史数据不会存储，所以一般需要和外部系统结合起来使用。

2. cAdvisor

cAdvisor 是 Container Advisor 的简称，Advisor 一般中文翻译成指导、顾问。从名字我们可以看出 cAdvisor 的定位是想充当用户的容器 “指导”，通过 cAdvisor 我们可以了解到容器的资源使用和性能情况。

如果我们要监控 Java 进程的资源使用情况，一般是使用 Java Agent 来做，在目标进程启动的时候把一个 Java Agent Class 作为参数传递进去，然后每个 Java Agent 再将进程的指标暴露出来。但是这种方式对于容器监控来说非常的繁琐，本来一个容器就是一个镜像的运行时实例，我们要添加 Java Agent 不是特别好加的。

通过 cAdvisor 来监控 Docker 容器，我们只需要在目标 Docker 容器的同一个主机上启动一个 cAdvisor 容器即可。也就是说 cAdvisor 容器可以监控这台机器上的所有的 Docker 容器。下面我们演示一下。

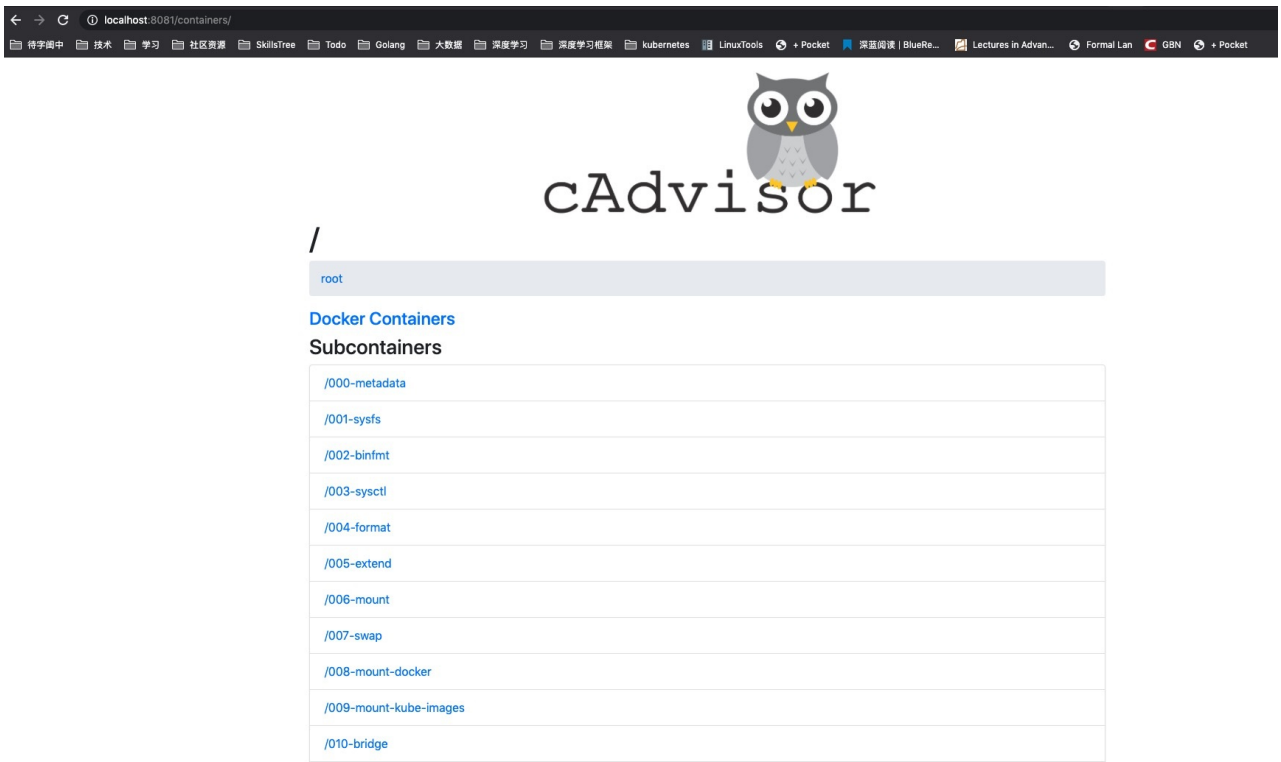
首先通过 docker pull 下载 cAdvisor 镜像，注意 cAdvisor 镜像的全称是 google/cadvisor，在 docker pull 的时候需要指定全称，不然会下载失败。

```
~ docker pull google/cadvisor
Using default tag: latest
latest: Pulling from google/cadvisor
Digest: sha256:815386ebbe9a3490f38785ab11bda34ec8dacf4634af77b8912832d4f85dca04
...
```

cAdvisor 启动也比较简单，通过下面命令

```
~ docker run \
  --volume=/:/rootfs:ro \
  --volume=/var/run:/var/run:rw \
  --volume=/sys:/sys:ro \
  --volume=/var/lib/docker/:/var/lib/docker:ro \
  --publish=8081:8080 \
  --detach=true \
  --name=cadvisor \
  google/cadvisor:latest
8a2fda9bb419544173e8a6ae7f0fb3b13d5b71a6fcddfc4fbaa74f6ccf8e2124
```

主要其中的 publish 参数是将 cAdvisor 内部的 8080 端口映射到主机的 8081 端口，然后我们用浏览器打开 localhost:8081 即可访问到容器的监控概括。

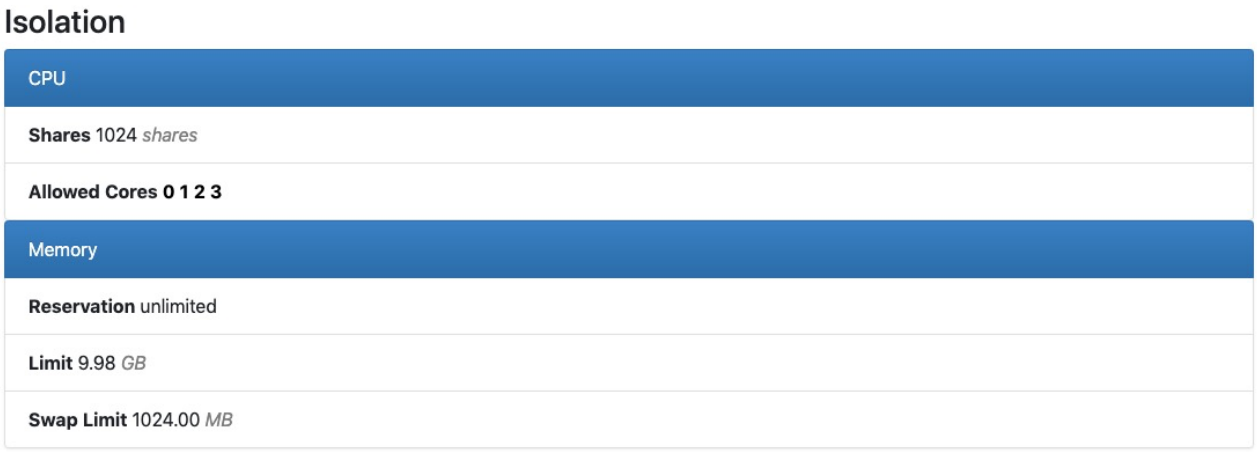


监控数据主要包括几个模块：

1.1 监控数据

Isolation

表示资源隔离情况。



资源使用概览

首先是资源使用概览，是以仪表盘来表示的，包括 CPU、内存、文件系统等。其中文件系统中的 #1、#2 表示的是不同的挂载点。

CPU 使用率

CPU 使用率包括三个部分，总的使用率，每个 core 使用率和用户态内核态使用率。我们可以看到三个图的曲线形状基本一致。

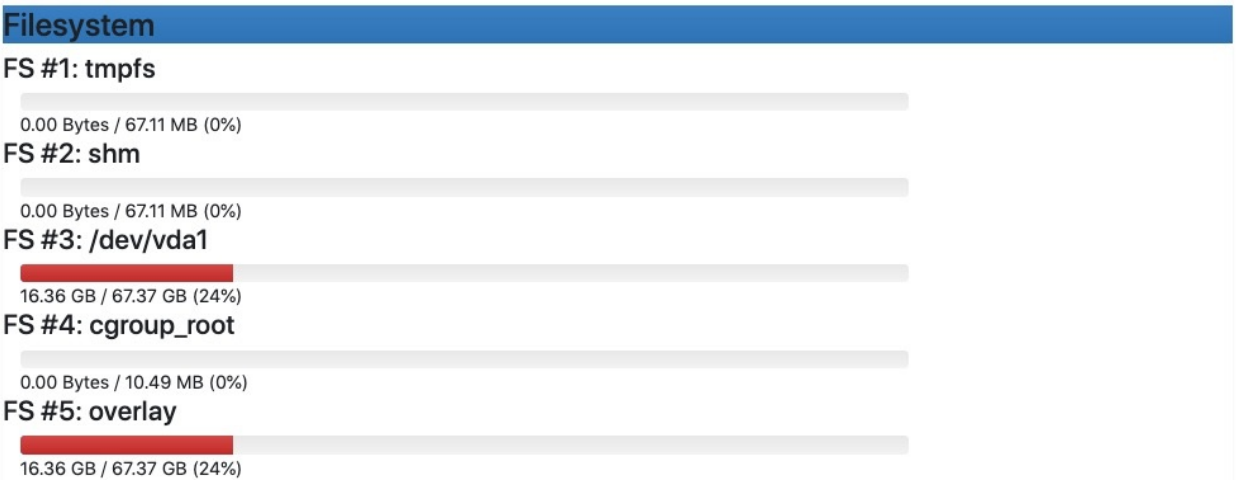

CPU
Total Usage
0.60
0.45
0.30
0.15
0.00
Cores
下午8:47:45
下午8:48:00
下午8:48:15
下午8:48:30
下午8:48:45
下午8:49:00
Total
Usage per Core
0.20
0.15
0.10
0.05
0.00
Cores
Core 0
Core 1
Core 2
Core 3
Usage Breakdown
0.24
0.18
0.12
0.06
0.00
Cores
User
Kernel

内存使用


Memory
Total Usage
2,100
1,950
1,800
1,650
Megabytes
下午8:49:30
下午8:49:45
下午8:50:00
下午8:50:15
下午8:50:30
下午8:50:45
Total
Hot
Usage Breakdown
1.99 GiB / 9.98 GiB (19%)

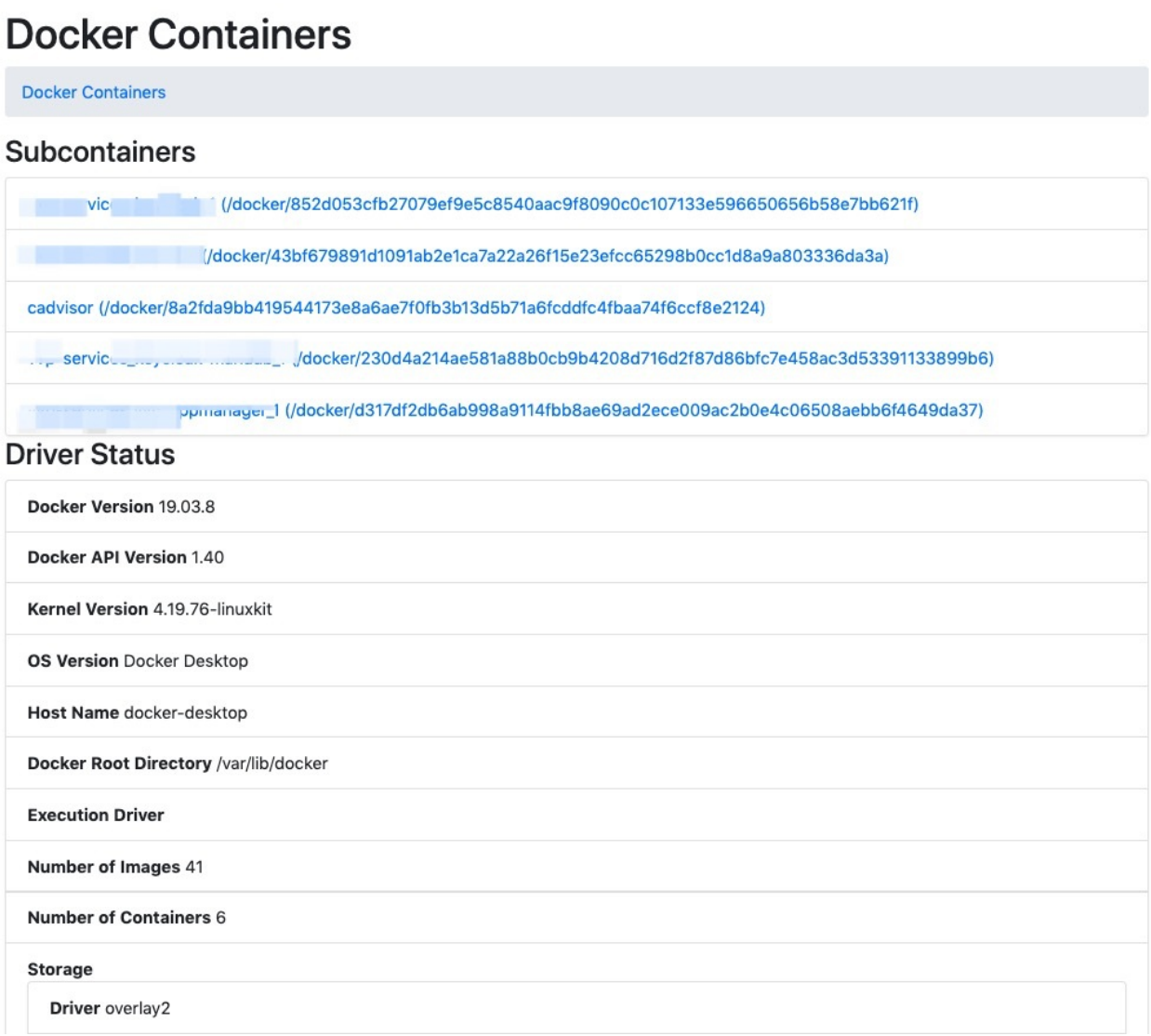
网络使用

文件系统使用



1.2 监控多个容器

我们点击页面顶部的 ***Docker Containers*** 可以查看这台机器上运行的所有 Docker 容器的情况，要想查看单个容器的监控数据直接点击 ***Subcontainers*** 列表中的容器即可，然后即可展示 1.1 中所示的内存。



3. Sysdig

在 Sysdig 的 github 链接 上下面的这句话来描述 sysdig:

> Linux system exploration and troubleshooting tool with first class support for containers.

简单翻译一下

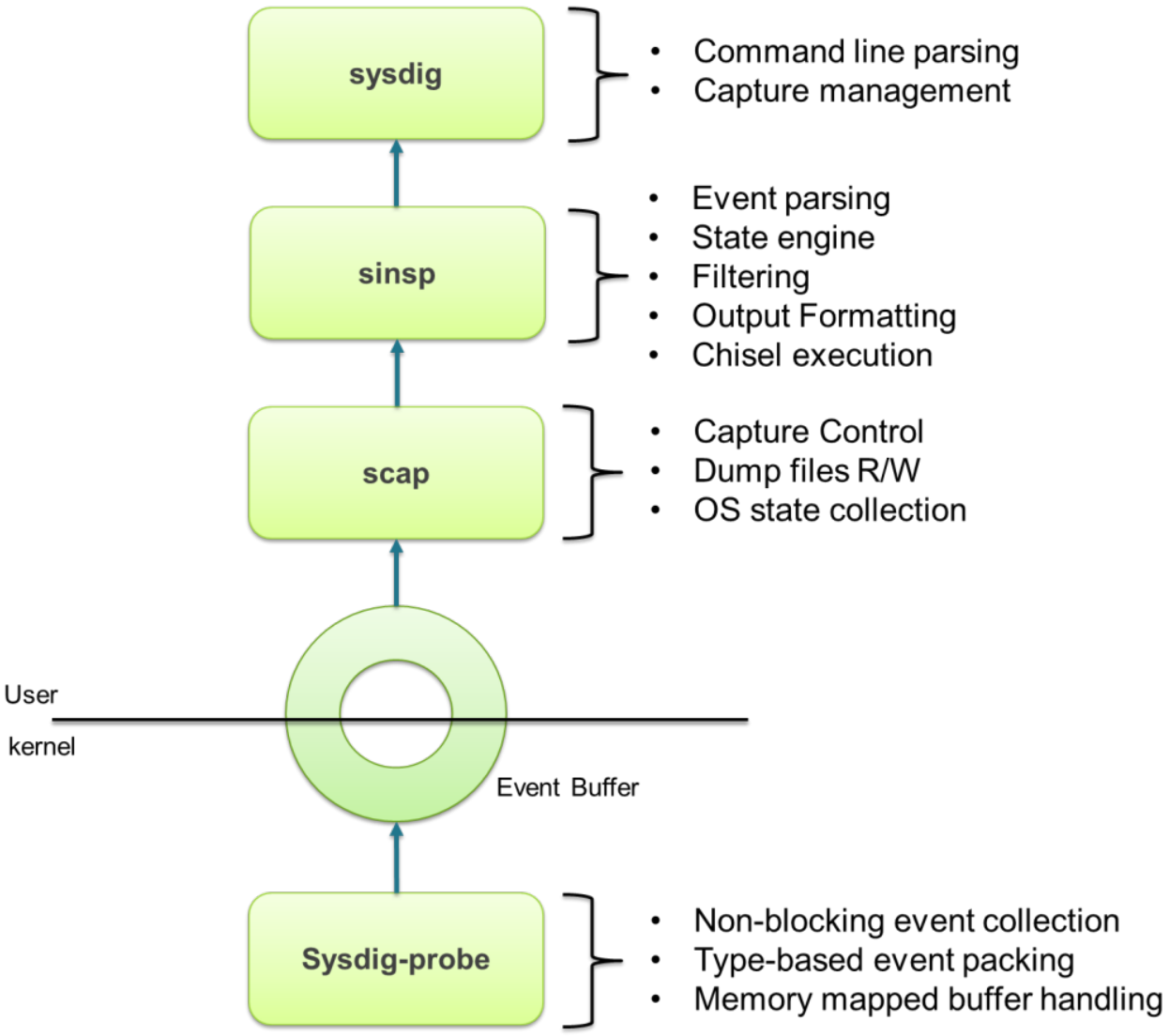
> 原生支持容器的Linux 系统探测和故障排查工具。

具体来说，Sysdig 由两个单词组成：system（系统）+ dig（挖掘），由此可以对 Sysdig 的定位窥见一斑：系统监控。实际上 Sysdig 是一个开源监控命令工具集，提供了操作系统层级的监控命令行功能，还原生支持 docker 的监控。另外 Sysdig 还提供了 Csysdig 工具，一个可以在命令行环境中交互式地、易用地、可视化监控信息查看界面。

Sysdig 是一款非常强大的“瑞士军刀”式的系统监控工具，通过 Sysdig 我们可以全方位的获取系统的性能指标，包括 CPU、内存、网络、IO 等。除了直接提供数据之外，Sysdig 还提供了丰富的命令行诊断工具直接对系统进行诊断式获取数据，比如：

- 按照 CPU 的使用率对进程进行排序，找到 CPU 使用率最高的那个；
- 按照发送网络数据报文的多少对进程进行排序；
- 找到打开最多文件描述符的进程；
- 查看哪些进程修改了指定的文件；
- 打印出某个进程的 HTTP 请求报文；
- 找到用时最久的系统调用；
- 查看系统中所有的用户都执行了哪些命令。
- ……

Sysdig 的强大得益于它的工作原理，简单来说，Sysdig 通过在内核模块中多种系统调用注册特定的 hook，这样当有系统调用发生和完成的时候，它会把系统调用相关的 metric 信息拷贝到特定的 buffer，然后用户模块的组件对数据信息处理（解压、解析、过滤等），并最终通过 sysdig 命令行和用户进行交互。



下面我们简单演示一下 Sysdig 的使用。首先我们通过命令 `curl -s https://s3.amazonaws.com/download.draios.com/stable/install-sysdig | sudo bash` 进行安装，这条命令会自动检测我们的系统版本，选择合适的版本进行安装。我这里测试的系统的 Ubuntu 系统可以安装成功，Mac OS 和 CentOS 都安装失败了。

```
root@xxx:~# curl -s https://s3.amazonaws.com/download.draios.com/stable/install-sysdig | sudo bash
* Detecting operating system
* Installing Sysdig public key
OK
* Installing sysdig repository
* Installing kernel headers
* Installing sysdig

Selecting previously unselected package dkms.
(Reading database ... 113127 files and directories currently installed.)
Preparing to unpack .../dkms_2.8.1-5ubuntu1_all.deb ...
Unpacking dkms (2.8.1-5ubuntu1) ...
Selecting previously unselected package sysdig.
Preparing to unpack .../sysdig_0.26.7_amd64.deb ...
Unpacking sysdig (0.26.7) ...
Setting up dkms (2.8.1-5ubuntu1) ...
Setting up sysdig (0.26.7) ...
Loading new sysdig-0.26.7 DKMS files...
Building for 5.4.0-31-generic
Building initial module for 5.4.0-31-generic
Done.
sysdig-probe.ko:
Running module version sanity check.
 - Original module
   - No original module exists within this kernel
 - Installation
   - Installing to /lib/modules/5.4.0-31-generic/updates/dkms/

depmod....

DKMS: install completed.
Processing triggers for man-db (2.9.1-1) ...
```

下面我们简单演示一下通过 sysdig 获取系统信息。

```
root@xxx:~# sysdig -c topprocs_net //获取使用网络最多的进程
```

Docker Containers

Docker Containers

Subcontainers

(/docker/852d053cfb27079ef9e5c8540aac9f8090c0c107133e596650656b58e7bb621f)

(/docker/43bf679891d1091ab2e1ca7a22a26f15e23efcc65298b0cc1d8a9a803336da3a)

cadvisor (/docker/8a2fda9bb419544173e8a6ae7f0fb3b13d5b71a6fcddfc4fbaa74f6ccf8e2124)

/docker/230d4a214ae581a88b0cb9b4208d716d2f87d86bfc7e458ac3d53391133899b6)

(/docker/d317df2db6ab998a9114fbb8ae69ad2ece009ac2b0e4c06508aebb6f4649da37)

Driver Status

Docker Version 19.03.8

Docker API Version 1.40

Kernel Version 4.19.76-linuxkit

OS Version Docker Desktop

Host Name docker-desktop

Docker Root Directory /var/lib/docker

Execution Driver

Number of Images 41

Number of Containers 6

Storage

Driver overlay2

root@xxx:~# sysdig -c topprocs_cpu //获取 CPU 使用率最高的进程列表

```
CPU%                Process             PID
------------------------------------------------------------------------------
3.00%               AliYunDun           1237
1.00%               exe                 1018
1.00%               sysdig              23274
0.00%               sshd                22868
0.00%               sshd                702
0.00%               containerd-shim     23008
0.00%               aliyun-service      878
0.00%               sh                  23027
0.00%               systemd-logind      663
0.00%               systemd-resolve     507
```

root@xxx:~# sysdig -c topprocs_file //查看磁盘使用最多的进程列表

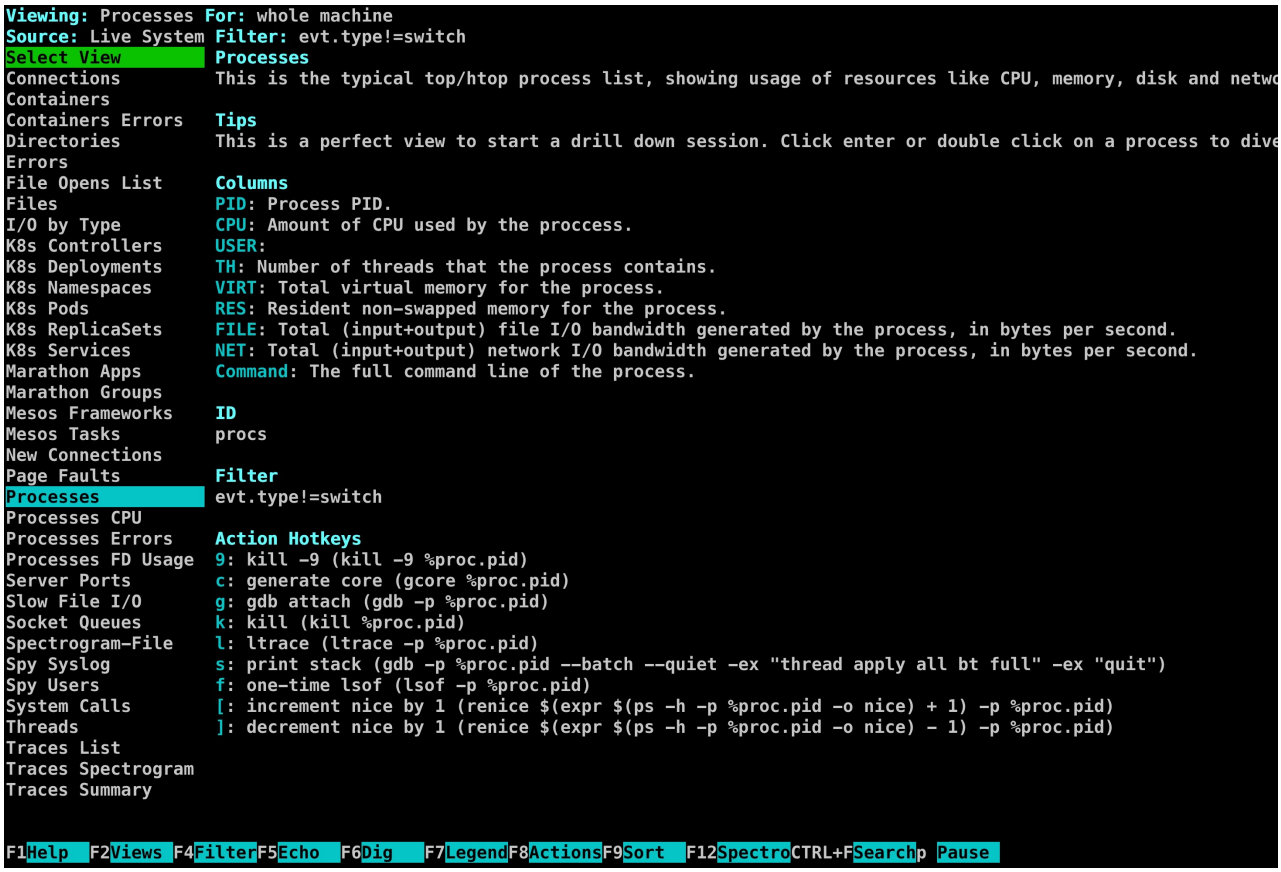
```
Bytes                Process              PID
--------------------------------------------------------------------------------
38.50KB              dockerd              22713
10.29KB              exe                  1018
612B                 systemd-resolve      507
284B                 sshd                 23089
184B                 sshd                 22907
52B                  AliYunDun            1237
```

当然 Sysdig 最强大的还是 Csysdig，我们来一起看一下。

```
root@xxx:~# csysdig
```

```
Viewing: Processes For: whole machine
Source: Live System Filter: evt.type!=switch
   PID     CPU USER           TH     VIRT      RES     FILE      NET Command
 23335    1.50 root            6     464M      30M        0    0.00 csysdig
  1237    1.00 root           26     110M      21M      17K   26.00 /usr/local/aegis/aegis_client/aegis_10_77/AliYunDun
  1018    0.50 root           24       2G      21M     361K   5.00K CmsGoAgent-Worker start
   311    0.00 root            1      34M      12M        0    0.00 /lib/systemd/systemd-journald
  1062    0.00 root            4      13M       5M        0    0.00 /usr/local/aegis/aegis_update/AliYunDunUpdate
 22713    0.00 root           15       1G      98M        0    0.00 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
   588    0.00 root            3     233M       7M        0    0.00 /usr/lib/accountsservice/accounts-daemon
 22984    0.00 root           12     966M      58M        0    0.00 docker run -ti busybox sh
   637    0.00 syslog          4     219M       5M        0    0.00 /usr/sbin/rsyslogd -n -iNONE
   650    0.00 _chrony         1       5M       2M        0    0.00 /usr/sbin/chronyd -F -1
 23089    0.00 root            1      14M      10M       4K   2.32K sshd: root@pts/2
     1    0.00 root            1     100M      12M        0    0.00 /sbin/init noibrs
 22907    0.00 root            1      14M       9M        0    0.00 sshd: root@pts/1
   728    0.00 root            1       8M       2M        0    0.00 /sbin/agetty -o -p -- \u --keep-baud 115200,38400,9600 ttyS0 vt220
  1004    0.00 root            7     547M      10M        0    0.00 /usr/local/cloudmonitor/CmsGoAgent.linux-amd64
 23008    0.00 root           10     106M       5M        0    0.00 containerd-shim -namespace moby -workdir /var/lib/containerd/io.containerd.
   735    0.00 root            1       8M       2M        0    0.00 /sbin/agetty -o -p -- \u --noclear tty1 linux
   601    0.00 messagebus      1       7M       4M        0    0.00 /usr/bin/dbus-daemon --system --address=systemd: --nofork --nopidfile --sys
 20044    0.00 root           15       1G      48M        0    0.00 /usr/bin/containerd
   878    0.00 root            3      42M       5M        0    0.00 /usr/sbin/aliyun-service
   613    0.00 root            1      31M      18M        0    0.00 /usr/bin/python3 /usr/bin/networkd-dispatcher --run-startup-triggers
  1098    0.00 root            1     101M       4M        0    0.00 (sd-pam)
   339    0.00 root            1      21M       5M        0    0.00 /lib/systemd/systemd-udevd
   504    0.00 systemd-net     1      26M       8M        0    0.00 /lib/systemd/systemd-networkd
   644    0.00 _chrony         1      13M       3M        0    0.00 /usr/sbin/chronyd -F -1
   663    0.00 root            1      16M       8M        0    0.00 /lib/systemd/systemd-logind
   702    0.00 root            1      12M       7M        0    0.00 sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups
 22740    0.00 root            1       2M     588K        0    0.00 bpfilter_umh
   600    0.00 root            1       9M       3M        0    0.00 /usr/sbin/cron -f
   507    0.00 systemd-res     1      24M      13M        0  306.00 /lib/systemd/systemd-resolved
 22868    0.00 root            1      14M       9M        0    0.00 sshd: root@pts/0
 22898    0.00 root            1      11M       5M        0    0.00 -bash
   692    0.00 daemon          1       4M       2M        0    0.00 /usr/sbin/atd -f
 22938    0.00 root            1      11M       5M        0    0.00 -bash
 23120    0.00 root            1      11M       6M        0    0.00 -bash
  1096    0.00 root            1      18M       9M        0    0.00 /lib/systemd/systemd --user
 23027    0.00 root            1       1M       4K        0    0.00 sh
F1Help  F2Views F4Filter F5Echo  F6Dig  F7Legend F8Actions F9Sort  F12Spectro CTRL+F Search p Pause
```

如上图所示，Csysdig 初始显示类似 top 命令显示的进程列表，包括 PID、CPU、内存、磁盘使用，网络等。我们注意看最下面一行的有一个 F2 Views，我们点击 F2 看一下。



我们可以看到 Csysdig 支持的 View 非常多，包括：

- Connections；
- Containers；
- Containers Errors；
- Directories。

值得一提的是还有几个 Kubernetes 的对象的 View，也就是说 Sysdig 也是可以用来监控 Kubernetes 的。

切换到 Containers View 模块上就能看到这台机器上面运行的所有容器的信息（我当前这台集群上面只运行了一个 busybox），如下。

4. 总结

本篇文章介绍了 Docker 监控的几种工具：

- Docker Stats，原生的命令；
- cAdvisor，Google 出品的专门用来监控 Docker 的工具；
- Sysdig，瑞士军刀式的系统监控工具，原生支持 Docker。

综合来看背靠一个好爹且专一的 cAdvisor 使用的更加广泛，比如 Kubernetes 中就自动集成了 cAdvisor。在这里我也推荐各位使用 cAdvisor 来扩展自己的监控系统。

}

26 大话容器设计模式

28 从 0 到 1 构建分布式高可用的 web 应用