Docker rates.sh

docker, commands

This prints out a table of instantaneous network bandwidth use.

Python version

```
#!/usr/bin/env python3
"""
rates.py – instantaneous per-container network rates (B/s / Mbps)
Python 3.11.9 – no external packages required.

Usage examples:
  ./rates.py            # default 1s interval, show all containers
  ./rates.py 2 --top 10 # 2s interval, show top 10
"""
from __future__ import annotations
import subprocess, time, sys, shutil, signal, os
from datetime import datetime
from typing import Dict, Tuple

INTERVAL = float(sys.argv[1]) if len(sys.argv) > 1 and
sys.argv[1].replace('.','',1).isdigit() else 1.0
TOP = None
# simple argument parsing for --top N
if "--top" in sys.argv:
    try:
        TOP = int(sys.argv[sys.argv.index("--top") + 1])
    except Exception:
        TOP = None

NAME_WIDTH = 30

def run(cmd: list[str]) -> str:
    res = subprocess.run(cmd, capture_output=True, text=True)
    return res.stdout.strip()

def list_container_ids() -> list[str]:
    out = run(["docker", "ps", "-q"])
    if not out:
        # fallback to podman if docker missing
        out = run(["podman", "ps", "-q"])
    return [x for x in out.splitlines() if x.strip()]

def inspect_container(id_: str) -> Tuple[str, int]:
    """Return (name, pid). name without leading slash."""
    out = run(["docker", "inspect", "--format", "{{.Name}}\t{{.State.Pid}}",
id_])
    if not out:
        out = run(["podman", "inspect", "--format",
"{{.Name}}\t{{.State.Pid}}", id_])
    if not out:
        return (id_, 0)
    parts = out.split("\t")
    name = parts[0].lstrip("/") if parts else id_
    pid = int(parts[1]) if len(parts) > 1 and parts[1].isdigit() else 0
    return name, pid

def read_net_dev_total(pid: int) -> Tuple[int,int]:
    """Return (rx_total, tx_total) bytes summed across interfaces for a
PID's net namespace."""
    path = f"/proc/{pid}/net/dev"
    try:
        with open(path, "r") as f:
            lines = f.readlines()[2:]
    except Exception:
        return 0, 0
    rx = 0
    tx = 0
    for line in lines:
        cols = line.split()
        # linux /proc/net/dev fields: iface: rx_bytes ... tx_bytes ...
        # after split, rx_bytes is at index 1, tx_bytes at index 9 (0-based)
        if len(cols) >= 10:
            try:
                rx += int(cols[1])
                tx += int(cols[9])
            except ValueError:
                continue
    return rx, tx

def human_bytes(v: int) -> str:
    if v >= (1<<30):
        return f"{v/(1<<30):.2f} GB"
    if v >= (1<<20):
        return f"{v/(1<<20):.2f} MB"
    if v >= (1<<10):
        return f"{v/(1<<10):.2f} KB"
    return f"{v} B"

def mbps(v: int) -> float:
    return v / 125000.0

def truncate(s: str, w: int) -> str:
    return (s if len(s) <= w else s[:w-3] + "...")

def clear_screen():
    print("\x1b[2J\x1b[H", end="")
```

```
def gather_snapshot() -> Dict[str, Tuple[str,int,int,int]]:
    """
    return dict: id -> (name, pid, rx_total, tx_total)
    """
    ids = list_container_ids()
    snap = {}
    for cid in ids:
        name, pid = inspect_container(cid)
        rx, tx = (0,0)
        if pid > 0 and os.path.exists(f"/proc/{pid}/net/dev"):
            rx, tx = read_net_dev_total(pid)
        snap[cid] = (name, pid, rx, tx)
    return snap

def main():
    prev = gather_snapshot()
    # honour terminal width for name column
    cols = shutil.get_terminal_size((120, 20)).columns
    global NAME_WIDTH
    NAME_WIDTH = min(NAME_WIDTH, max(12, cols - 70))
    try:
        while True:
            time.sleep(INTERVAL)
            now = gather_snapshot()
            rows = []
            ts = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
            for cid, (name, pid, rx_now, tx_now) in now.items():
                _, _, rx_prev, tx_prev = prev.get(cid, ("",0,0,0))
                # if container restarted, prev might be absent -> diffs are
rx_now-0 which is fine but might be large;
                # handle negative or weird results
                drx = rx_now - rx_prev
                dtx = tx_now - tx_prev
                if drx < 0: drx = 0
                if dtx < 0: dtx = 0
                # convert to per-second (interval may not be exactly
constant)
                rx_rate = int(drx / INTERVAL)
                tx_rate = int(dtx / INTERVAL)
                total = rx_rate + tx_rate
                rows.append({"name": name, "rx": rx_rate, "tx": tx_rate,
"tot": total})
            # sort descending by total
            rows.sort(key=lambda r: r["tot"], reverse=True)
            if TOP:
                rows = rows[:TOP]
            clear_screen()
            header = f"{ts}  (interval {INTERVAL}s)  containers:
{len(rows)}"
            print(header)
            name_w = NAME_WIDTH
            hdr = f"{'NAME':{name_w}}  {'RX':>12}   {'TX':>12}   {'TOT':>12}
{'RX(Mbps)':>9} {'TX(Mbps)':>9} {'TOT(Mbps)':>9}"
            print(hdr)
            print("-" * len(hdr))
            for r in rows:
                n = truncate(r["name"], name_w)
                rx_h = human_bytes(r["rx"])
                tx_h = human_bytes(r["tx"])
                tot_h = human_bytes(r["tot"])
                rx_m = f"{mbps(r['rx']):6.2f}"
                tx_m = f"{mbps(r['tx']):6.2f}"
                tot_m = f"{mbps(r['tot']):6.2f}"
                print(f"{n:{name_w}}  {rx_h:>12}   {tx_h:>12}   {tot_h:>12}
{rx_m:>9} {tx_m:>9} {tot_m:>9}")
            sys.stdout.flush()
            prev = now
    except KeyboardInterrupt:
        print("\nexiting.")
        return

if __name__ == "__main__":
    main()
```

Bash version

```
#!/usr/bin/env bash
# Instantaneous per-container network rates (B/s). No extra packages.
INTERVAL=1
TMPDIR=${TMPDIR:-/tmp}
PREV="$TMPDIR/net.prev.$$.txt"
NOW="$TMPDIR/net.now.$$.txt"
OUT="$TMPDIR/net.out.$$.txt"

cleanup(){ rm -f "$PREV" "$NOW" "$OUT"; exit 0; }
trap cleanup INT TERM

collect() {
  target="$1"
  : > "$target"
  docker ps -q | while read -r id; do
    [ -z "$id" ] && continue
    name=$(docker inspect --format '{{.Name}}' "$id" 2>/dev/null | sed
's#/##')
    pid=$(docker inspect --format '{{.State.Pid}}' "$id" 2>/dev/null)
    if [ -r "/proc/$pid/net/dev" ]; then
      read rx tx < <(awk 'NR>2{rx+=$2;tx+=$10}END{print rx,tx}'
/proc/"$pid"/net/dev 2>/dev/null || echo "0 0")
    else
      rx=0; tx=0
    fi
    # id<TAB>name<TAB>pid<TAB>rx<TAB>tx
    printf '%s\t%s\t%s\t%s\t%s\n' "$id" "$name" "$pid" "$rx" "$tx" >>
"$target"
  done
}

# seed previous snapshot
collect "$PREV"
sleep "$INTERVAL"

while true; do
  ts=$(date +"%F %T")
  collect "$NOW"

  # load previous counters into associative arrays
  declare -A PRX PTX PNAME
  while IFS=$'\t' read -r id name pid rx tx; do
    PRX["$id"]=$rx
    PTX["$id"]=$tx
    PNAME["$id"]=$name
  done < "$PREV"

  : > "$OUT"
  while IFS=$'\t' read -r id name pid arx atx; do
    brx=${PRX[$id]:-0}
    btx=${PTX[$id]:-0}
    rx_rate=$((arx - brx))
    tx_rate=$((atx - btx))
    [ "$rx_rate" -lt 0 ] && rx_rate=0
    [ "$tx_rate" -lt 0 ] && tx_rate=0
    total=$((rx_rate + tx_rate))
    # total<TAB>name<TAB>rx_rate<TAB>tx_rate<TAB>total
    printf '%d\t%s\t%d\t%d\t%d\n' "$total" "$name" "$rx_rate" "$tx_rate"
"$total" >> "$OUT"
  done < "$NOW"

  echo "==== $ts (rates over ${INTERVAL}s) ===="
  if [ -s "$OUT" ]; then
    sort -t $'\t' -nr -k1,1 "$OUT" | awk -F'\t' '{printf "%-28s RX:%8d B/s
TX:%8d B/s  TOT:%8d B/s\n", $2, $3, $4, $5}'
  else
    echo "No containers found."
  fi

  mv "$NOW" "$PREV"
  sleep "$INTERVAL"
done
```