Failover de Rota Default no Rocky Linux (gateway primário → fallback 4G)

🔎 Resumo

Implementa failover simples de gateway: a cada **2 minutos**, o host testa a saída via **192.168.31.2** (primário). Se não houver conectividade (ping para 8.8.8.8/1.1.1.1), alterna a **rota default** para **192.168.31.4** (*4G*). No próximo ciclo, tenta voltar ao primário.

🗂 Sumário

Visão geral

- **Primário:** `192.168.31.2`
- **Backup (4G):** `192.168.31.4`
- **Período:** a cada **2 minutos** (via `systemd.timer`)
- **Teste real de saída:** o script força rotas /32 temporárias para os destinos de teste **via o gateway avaliado**, evitando falso-positivo.

Script (`/usr/local/sbin/route-failover.sh`)

Edite com `vim /usr/local/sbin/route-failover.sh` e cole:

```
#!/usr/bin/env bash
set -euo pipefail

# Gateways
GW1="192.168.31.2"    # primário
GW2="192.168.31.4"    # 4g (backup)

# Alvos externos de teste (pelo menos 2 é melhor)
TARGETS=("8.8.8.8" "1.1.1.1")
PING_OPT="-c 2 -W 2"

log() { logger -t route-failover "$*"; }
```

```
# Descobre a interface que alcança cada gateway; cai para a default se
necessário
detect_dev_for() {
  local ip="$1"
  local dev
  dev="$(ip route get "$ip" 2>/dev/null | awk '{for(i=1;i<=NF;i++) if($i=="dev")
{print $(i+1); exit}}')"
  if [ -z "${dev:-}" ]; then
    dev="$(ip route show default | awk '/default/ {print $5; exit}')"
  fi
  echo "$dev"
}

test_via_gw() {
  # Força rota *somente* para os alvos via o GW informado, testa ping e limpa
  local gw="$1"
  local dev="$2"

  local ok=1
  for t in "${TARGETS[@]}"; do
    ip route replace "$t"/32 via "$gw" dev "$dev" || true
    if ping $PING_OPT "$t" >/dev/null 2>&1; then
      ok=0
      break
    fi
  done
  # Limpa rotas /32 (sem erro se não existirem)
  for t in "${TARGETS[@]}"; do
    ip route del "$t"/32 via "$gw" dev "$dev" 2>/dev/null || true
  done
  return $ok  # 0 = sucesso; 1 = falha
}

main() {
  local cur gw1_dev gw2_dev
  cur="$(ip route show default | awk '/default/ {print $3; exit}')"
  gw1_dev="$(detect_dev_for "$GW1")"
  gw2_dev="$(detect_dev_for "$GW2")"

  if test_via_gw "$GW1" "$gw1_dev"; then
    # Primário OK -> garante default via GW1
    if [ "$cur" != "$GW1" ]; then
      ip route replace default via "$GW1" dev "$gw1_dev" metric 100
      log "Retorno ao gateway primário $GW1 (dev $gw1_dev)"
    else
      log "Primário $GW1 já em uso (dev $gw1_dev)"
    fi
    exit 0
  fi

  # Primário falhou; tenta backup
  if test_via_gw "$GW2" "$gw2_dev"; then
    if [ "$cur" != "$GW2" ]; then
      ip route replace default via "$GW2" dev "$gw2_dev" metric 100
```

```
      log "Failover para gateway de backup $GW2 (dev $gw2_dev)"
    else
      log "Backup $GW2 já em uso (dev $gw2_dev)"
    fi
    exit 0
  fi

  # Nenhum dos dois saiu à internet; mantém o que está
  log "Nenhuma rota válida (GW1 e GW2 falharam); mantendo gateway atual: ${cur:-
nenhum}"
}

main
```

Permissão de execução:

```
chmod +x /usr/local/sbin/route-failover.sh
```

Service e Timer (2 minutos)

Crie o service com `vim /etc/systemd/system/route-failover.service`:

```
[Unit]
Description=Failover de rota default baseado em teste de conectividade
Wants=network-online.target
After=network-online.target

[Service]
Type=oneshot
ExecStart=/usr/local/sbin/route-failover.sh
```

Crie o timer com `vim /etc/systemd/system/route-failover.timer`:

```
[Unit]
Description=Executa o failover de rota a cada 2 minutos

[Timer]
OnBootSec=2min
OnUnitActiveSec=2min
AccuracySec=30s
Persistent=true

[Install]
WantedBy=timers.target
```

Recarregue e habilite:

```
systemctl daemon-reload
systemctl enable --now route-failover.timer
systemctl start route-failover.service
systemctl status route-failover.timer --no-pager
```

Habilitar, testar e logs

Rodar manualmente e conferir logs:

```
/usr/local/sbin/route-failover.sh
journalctl -t route-failover -n 50 --no-pager
```

Acompanhar em tempo real:

```
journalctl -t route-failover -f
```

Ver a rota default atual:

```
ip route show default
```

Troubleshooting & notas

- **NetworkManager reescrevendo rota?** Sem problemas: o próximo ciclo do timer corrige.
- **Alvos de teste:** adicione mais IPs públicos em `TARGETS=(...)` se quiser maior robustez.
- **Ambos os gateways fora:** o script **não** muda nada; tenta novamente no próximo ciclo.
- **Latência vs. período:** para ambientes ruidosos, considere aumentar `-c`/`-W` no `PING_OPT`.

Checklist

- ☑ `route-failover.sh` criado com **GW2=192.168.31.4**
- ☑ Service e Timer criados
- ☑ `OnUnitActiveSec=2min` aplicado
- ☑ `systemctl daemon-reload` executado
- ☑ Timer **enable + start**
- ☑ Logs confirmam failover/retorno

Referências rápidas

```
# Forçar um ciclo agora
systemctl start route-failover.service

# Ver rota default e interface
ip route show default

# Logs do componente
journalctl -t route-failover -n 100 --no-pager
```

Assinatura

Criado por **Jeferson Salles** LinkedIn: https://www.linkedin.com/in/jmsalles/ E-mail: jefersonmattossalles@gmail.com GitHub: https://github.com/jmsalles/IaC/