Neutron L3 Agent Dead Runbook

A senior engineer's incident runbook for when neutron-l3-agent goes XXX/dead and routers, gateways, and floating IPs fall off the network. DevOps AI ToolKit · devopsaitoolkit.com

When `openstack network agent list` shows the L3 agent as `XXX` (alive `False`) or `dead`, Neutron stops managing the `qrouter-*` namespaces on that host: tenant routers lose their external gateway, floating IPs stop answering, and north-south traffic for every router scheduled to that agent goes dark. The agent is a process that lives inside a namespace-owning container and talks to `neutron-server` over RPC. An `XXX` state almost always means one of three things — the agent process is down, its RabbitMQ heartbeat lapsed so `report_state` never landed, or the box is so loaded the report is late. Work read-only from the control plane inward, then out to the datapath.

> Assumes a Kolla-Ansible deployment. Adjust container names for your tooling. Run read-only checks first; only restart the narrowest component, with a change record and ideally a second engineer.

1. Confirm the Agent State (control-plane view)

```
# Is the L3 agent actually down, and since when?
openstack network agent list --agent-type l3 -c ID -c Host -c Alive -c State -c "Binary"

# Full detail on the specific agent (last heartbeat, config)
openstack network agent show <agent-id> -f yaml | grep -Ei "alive|admin_state|heartbeat|host|binary"

# How many routers are stranded on this agent?
openstack network agent list --routers --agent <agent-id> -f value | wc -l
```

- `Alive = XXX` (`:-)` is healthy) means `neutron-server` hasn't seen a `report_state` within `agent_down_time` (default 75s).
- A *stale* `last_heartbeat_time` with a live process points at RPC/heartbeat (§3), not a crashed agent.
- Note the router count now — it tells you the blast radius and whether HA (§6) will cover you.

2. Neutron L3 Agent Container + Logs

```
# Is the container even running on the affected host?
docker ps --filter name=neutron_l3_agent --format '{{.Names}} {{.Status}}'

# Recent errors: crashes, tracebacks, AMQP loss, sync failures
docker logs --tail=200 neutron_l3_agent 2>&1 \
  | grep -Ei "error|traceback|amqp|timeout|failed to|resync|rpc"

# Is the report_state loop still firing?
docker logs --tail=200 neutron_l3_agent 2>&1 | grep -Ei "report_state|heartbeat|state_report"
```

- A repeating traceback or `Agent out of sync` → the process is up but wedged; a restart (§7) usually clears it.
- `Container exited` / restart-looping → read the last log lines for the fatal cause before restarting blindly.
- Log silent for minutes while wall-clock advances → event loop is blocked (often the same cause as missed heartbeats in §3).

3. RabbitMQ / RPC Connectivity for the Agent

```
# Did RabbitMQ drop this agent's heartbeat?
docker logs --tail=300 rabbitmq 2>&1 | grep -i "missed heartbeats from client"

# The L3 plugin RPC queue and the agent's own consumer count
docker exec rabbitmq rabbitmqctl list_queues name messages consumers \
  | grep -Ei "q-l3-plugin|l3_agent|q-plugin"

# Is the agent host still holding a connection to the broker?
docker exec rabbitmq rabbitmqctl list_connections peer_host user state | grep -i <l3-agent-host-ip>
```

- `consumers = 0` on `q-l3-plugin` for this host → the agent lost its subscription; `report_state` can't reach `neutron-server`. Restart the agent (§7).
- Missed-heartbeat lines naming the agent's `peer_host` → the AMQP connection silently died; the agent thinks it's connected but the broker closed it.
- `messages` climbing on the agent queue with `consumers = 0` → confirms a disconnected/dead consumer, not a broker fault.

4. qrouter Namespaces on the Agent Host

```
# List router namespaces the agent should own (run on the affected host)
docker exec neutron_l3_agent ip netns list | grep qrouter

# Inspect one router's namespace: interfaces and routes present?
docker exec neutron_l3_agent ip netns exec qrouter-<router-id> ip -br a
docker exec neutron_l3_agent ip netns exec qrouter-<router-id> ip route
```

- No `qrouter-*` namespaces while routers are scheduled here → the agent never (re)built state; a resync/restart is needed.
- Namespaces present but missing `qr-*` (internal) or `qg-*` (gateway) interfaces → partial/incomplete sync; check §2 logs for the failing device.
- Default route inside the namespace should point at the external subnet gateway — a missing default route is why floating IPs go unreachable.

---

5. External Gateway (qg- interface)

```
# The gateway port lives on qg-<port>; confirm it's UP with the SNAT/gateway IP
docker exec neutron_l3_agent ip netns exec qrouter-<router-id> ip -br a show type veth | grep qg-

# Can the router reach its external gateway from inside the namespace?
docker exec neutron_l3_agent ip netns exec qrouter-<router-id> \
  ping -c3 -W2 <external-subnet-gateway-ip>
```

- `qg-*` `DOWN` or absent → no north-south path; the router is islanded even if internal ports are fine.
- Gateway ping fails but `qg-*` is UP → suspect the physical/provider network or `br-ex` mapping, not the L3 agent itself.
- For DVR, the centralized SNAT namespace is `snat-<router-id>` on the network node — check it too.

---

6. Keepalived / HA Routers (if L3 HA)

```
# Which agent currently holds the master VIP for an HA router?
openstack network agent list --router <router-id> -c Host -c "HA State" -c Alive

# Inside the namespace, keepalived state + the HA VIP
docker exec neutron_l3_agent ip netns exec qrouter-<router-id> \
  ip -br a | grep -E "ha-|<ha-vip>"
docker logs --tail=100 neutron_l3_agent 2>&1 | grep -i keepalived
```

- With L3 HA, a dead agent should trigger failover: another agent's `HA State` flips to `active`. Confirm a standby actually took over before treating this as a full outage.
- Two agents both showing `active` (split-brain VIP) → the `ha-*` keepalived VRRP traffic is being dropped; do not restart both at once.
- If no standby exists (single-agent router), §7 restart is the fastest path back.

---

7. Restart Decision Tree

> **Restart the L3 agent, not the broker.** Restarting `neutron_l3_agent` re-subscribes its RPC queue and resyncs namespaces, which fixes most `XXX` / `consumers=0` cases without risking clustered services. Rebuilding namespaces briefly blips the datapath — do it deliberately.

```
L3 agent XXX / dead?
 ├─ RabbitMQ partitioned or alarmed? ──── yes → fix broker FIRST, escalate; do not restart agents blindly
 └─ no
    ├─ HA router with a healthy standby active? ── yes → verify failover held; fix this agent out of the hot path
    ├─ container exited / crash-looping?       ── yes → read fatal log line, then docker restart neutron_l3_agent
    ├─ process up but queue consumers=0?        ── yes → docker restart neutron_l3_agent (re-subscribes RPC)
    └─ namespaces/qg- missing, agent otherwise ok? yes → docker restart neutron_l3_agent (forces full resync)
```

```
# Narrowest fix: restart only the L3 agent on the affected host
docker restart neutron_l3_agent

# Re-verify it comes back alive and re-owns its routers
openstack network agent show <agent-id> -f value -c alive
docker exec neutron_l3_agent ip netns list | grep -c qrouter
```

- After restart, watch for the namespaces to reappear and `Alive` to flip to `:-)` within ~1–2 heartbeat cycles.
- If it goes `XXX` again within minutes, stop restarting — the cause is upstream (RabbitMQ heartbeats, host load, or network partition). Escalate.

---

8. Floating IP Path Verification (end-to-end)

```
# Confirm the floating IP is bound and where
openstack floating ip show <floating-ip> -c fixed_ip_address -c router_id -c status

# The DNAT/SNAT rules live in the router namespace
docker exec neutron_l3_agent ip netns exec qrouter-<router-id> iptables -t nat -S | grep <floating-ip>

# ARP for the floating IP should answer on qg-
docker exec neutron_l3_agent ip netns exec qrouter-<router-id> \
  ip neigh show dev qg-<port-id>
```

- Missing `DNAT` / `SNAT` rules for the floating IP → the agent didn't program NAT; resync via restart (§7).
- Rules present but no external reachability → the datapath past `qg-` / `br-ex` is the issue (provider network, upstream router), not Neutron.

---

9. Copy/Paste One-Shot Triage

```
AGENT=<l3-agent-id>; ROUTER=<router-id>
echo "== agent state ==";   openstack network agent show $AGENT -f value -c alive -c host -c binary
echo "== container ==";     docker ps --filter name=neutron_l3_agent --format '{{.Status}}'
echo "== agent errors ==";  docker logs --tail=100 neutron_l3_agent 2>&1 | grep -Eic "error|traceback|amqp|timeout"
echo "== l3 queue ==";      docker exec rabbitmq rabbitmqctl list_queues name messages consumers | grep -Ei "q-l3-plugin|l3_agent"
echo "== missed hb ==";     docker logs --tail=200 rabbitmq 2>&1 | grep -c "missed heartbeats"
echo "== namespaces ==";    docker exec neutron_l3_agent ip netns list | grep -c qrouter
echo "== gw iface ==";      docker exec neutron_l3_agent ip netns exec qrouter-$ROUTER ip -br a 2>/dev/null | grep qg-
```

---

10. Incident Notes Template

```
INCIDENT: Neutron L3 Agent Dead (XXX)
Started (UTC):          ____    Detected by: ____    Sev: ____
Impact:                 (routers stranded ____ , tenants ____ , floating IPs down ____)
Agent:                  id ____  host ____  Alive ____  last_heartbeat ____
Container:              [ ] running  [ ] exited/looping   log signature: ____
RPC/RabbitMQ:           [ ] q-l3-plugin consumers ____  [ ] missed heartbeats from ____
                        [ ] partitions: ____  [ ] alarms: ____
Datapath:               [ ] qrouter namespaces present ____  [ ] qg- UP? ____
                        [ ] gateway ping ____  [ ] FIP NAT rules present? ____
HA (if L3 HA):          [ ] standby took over? ____  [ ] split-brain VIP? ____
Action taken:           (agent restarted / resync / broker action + who + when)
Result:                 [ ] recovered @ ____   [ ] escalated to ____
Root cause:             ____
Follow-ups:             (heartbeat tuning, host load, HA coverage, alerts) ____
```
