JKT-IX

Closing the Loop: Telemetry Driven Traffic Reroute for JKT-IX Backbones

APIX 2026

Today's Agenda

- **Problem:** Intermittent packet loss, manual ops, slow MTTR
- **Solution:** Telemetry + Grafana alerts + auto reroute
- **System Architecture:** Exporter → Prometheus → Grafana → Webhook → Switches
- **Future Enhancements:** Smarter logic, secrets hardening, richer observability

Why we built this?

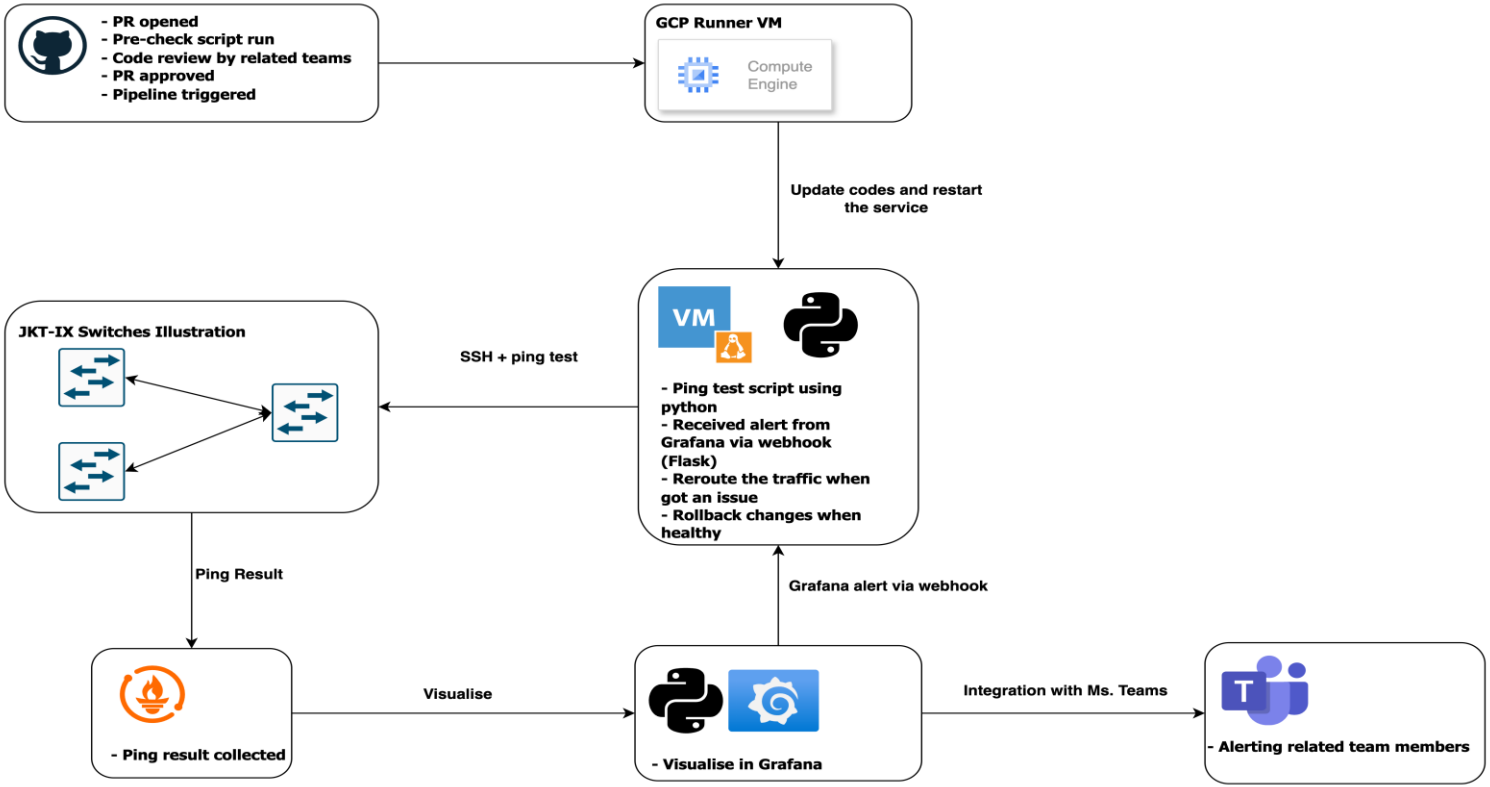
Pain Points

- Intermittent packet loss on JKT-IX backbones, no clear timestamp pattern.
- Manual ping tests, manual routing protocol tweaks = slow MTTR.
- Risky ad-hoc changes (plaintext creds, manual deploys).

Goals

- Continuous ping telemetry from JKT-IX switches.
- Visualize latency & packet loss in Grafana, retain history.
- Alert Product/OAM team instantly when reach certain amount of threshold.
- Auto-reroute traffic when issue appeared then revert when healthy.
- Safe, repeatable CI/CD flow (GitHub Actions → VM) for all script changes.

System Architecture



Example case

When Prometheus detects packet loss or high latency, it is displayed in Grafana.

Grafana then triggers an alert that runs a script to automatically reroute the traffic.

Once conditions return to normal, another script is triggered to roll back the change.

```python
def ssh_execute(host, commands):
    """
    Open an interactive shell and send the list of commands.
    """
    ssh = paramiko.SSHClient()
    ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
    ssh.connect(host, username=

    shell = ssh.invoke_shell()
    # give the shell a moment to be ready
    time.sleep(1)
    shell.recv(1000)

    for cmd in commands:
        shell.send(cmd + "\n")
        time.sleep(0.5)          # allow the device to process
    # Optionally read output for logging
    output = shell.recv(5000).decode(errors='ignore')
    logging.debug(f"{host} output:\n{output}")

    shell.close()
    ssh.close()

# ── 4. Reroute Logic ──────────────────────────────────────────
def set_ospf_cost(backbone, path, enable: bool):
    for host, iface in PATH_MAP[backbone][path]:
        cmds = [
            "configure terminal",
            f"interface {iface}",
            f"{'ip ospf cost 200' if enable else 'no ip ospf cost 200'}",
            "end"
        ]
        logging.info(f"{'Setting' if enable else 'Removing'} cost on {host} {iface}")
        ssh_execute(host, cmds)

# ── 5. Webhook Receiver ───────────────────────────────────────
@app.route("/alert", methods=["POST"])
def alert_webhook():
    raw = request.get_data(as_text=True)
    app.logger.info("=== RAW GRAFANA PAYLOAD ===\n%s", raw)
    data = request.get_json()
    for alert in data.get("alerts", []):
        b = alert["labels"].get("backbone")
        p = alert["labels"].get("path")
        status = alert["status"]  # "firing" or "resolved"
        if b ==                   and p in PATH_MAP[b]:
            if status == "firing":
                logging.warning(f"Alert firing for {b}/{p}: applying reroute")
                set_ospf_cost(b, p, enable=True)
            elif status == "resolved":
                logging.info(f"Alert resolved for {b}/{p}: restoring cost")
                set_ospf_cost(b, p, enable=False)
    return "", 200
```

Future Enhancements

- **Smarter Reroute Logic**
  - Add cool-down & max-change counters to prevent flapping
- **Config & Secrets Hardening**
  - Move creds to Vault/Keycloak
- **Observability & Audit**
  - Push reroute events to Loki/ELK for searchable history
  - Add traceroute/jitter metrics & SLA dashboards

JKT-IX

Happy peering!