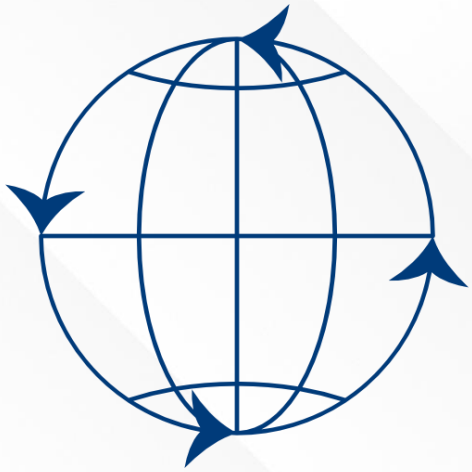




Opensource ICT Solutions

ZABBIX Summit 2026

So, You Want to Spend Less? Moving to Zabbix From LogicMonitor



Opensource ICT Solutions

Who am I?



Nathan Liefing
Zabbix Consultant / Trainer

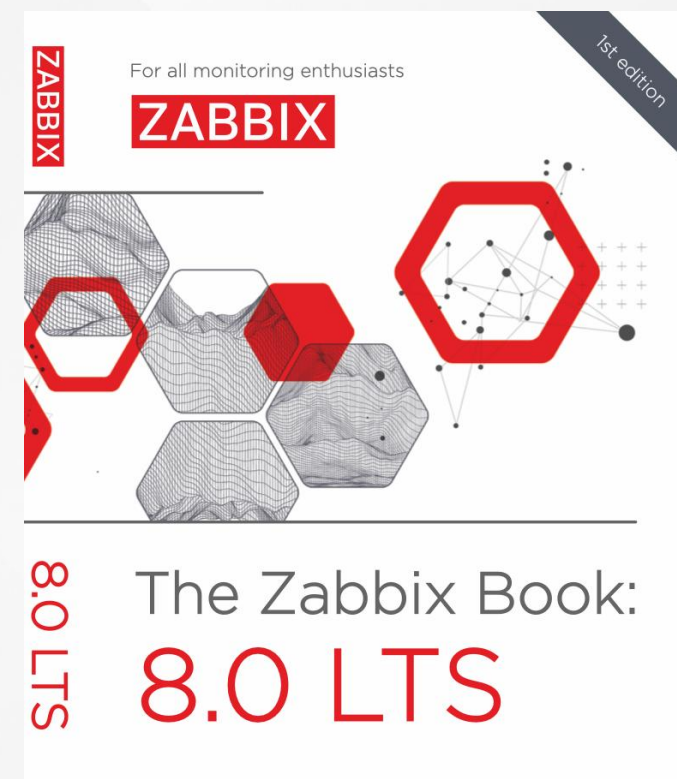
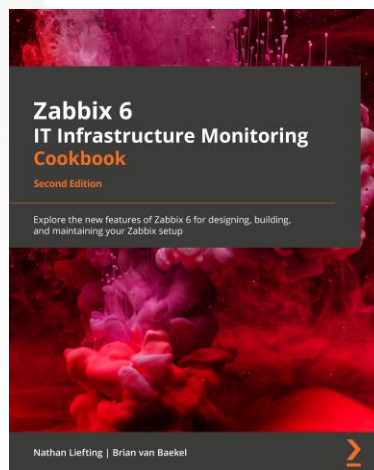
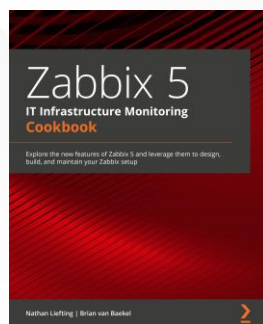


Opensource ICT Solutions

- Zabbix support
- Zabbix training
- Zabbix consultancy
- And more...



<https://www.linkedin.com/company/opensource-ict-solutions/>



ZABBIX • 2026 - onwards

ENTER A NEW ERA OF **ZABBIX**

The **observability** era starts now.

Built for 2026. Ready for what comes next.



OpenSource ICT Solutions

Datadog and Dynatrace – The big benefits are gone...

- ✓ Log file monitoring
- ✓ Application performance monitoring
Open Telemetry

 Netflow (8.2)

 Do we even need to discuss the licensing fees?



splunk[®]>



 **LogicMonitor**

Initial customer request

- Replace LogicMonitor with a self-hosted Zabbix platform
- Keep monitoring coverage while moving platforms
- Hosted across Europe and Asia with a clear HA and disaster-recovery model
- Automate deployment and migration without importing bad configuration



What is logicmonitor and why is it “popular”?

- SaaS platform exclusively: the customer does not operate the monitoring backend
- Collectors discover devices and send monitoring data to LogicMonitor
- DataSources combine collection, discovery, graphs and alert thresholds
- A broad built-in library makes onboarding fast. However, licensing scales with hybrid units



Why move?



\$16-\$53 per hybrid unit*

One Hybrid Unit is ~1:

- On-prem collector monitored device
- One cloud IaaS, seven cloud PaaS
- Five wireless access points

With about 5000 hybrid units:

\$172,500 per month



2x Database, 1x Arbitrator, 2x Web, 2x Zabbix servers
5x Zabbix proxy

Cloud hosting costs: ~\$1,500 per month**
Enterprise support: ~\$7,666 per month

Unlimited devices

\$ 9,122.31 per month
\$ 18,244.62 per month with DEV environment

*According to logicmonitor.com list prices – 27-04-2026

**depending on sizing, region and storage



The plan

- Requirements

- Move existing monitoring
- Cross region (Europe/Asia) database

- Scripted move

- Host import
- Web monitoring import

- Semi-scripted move

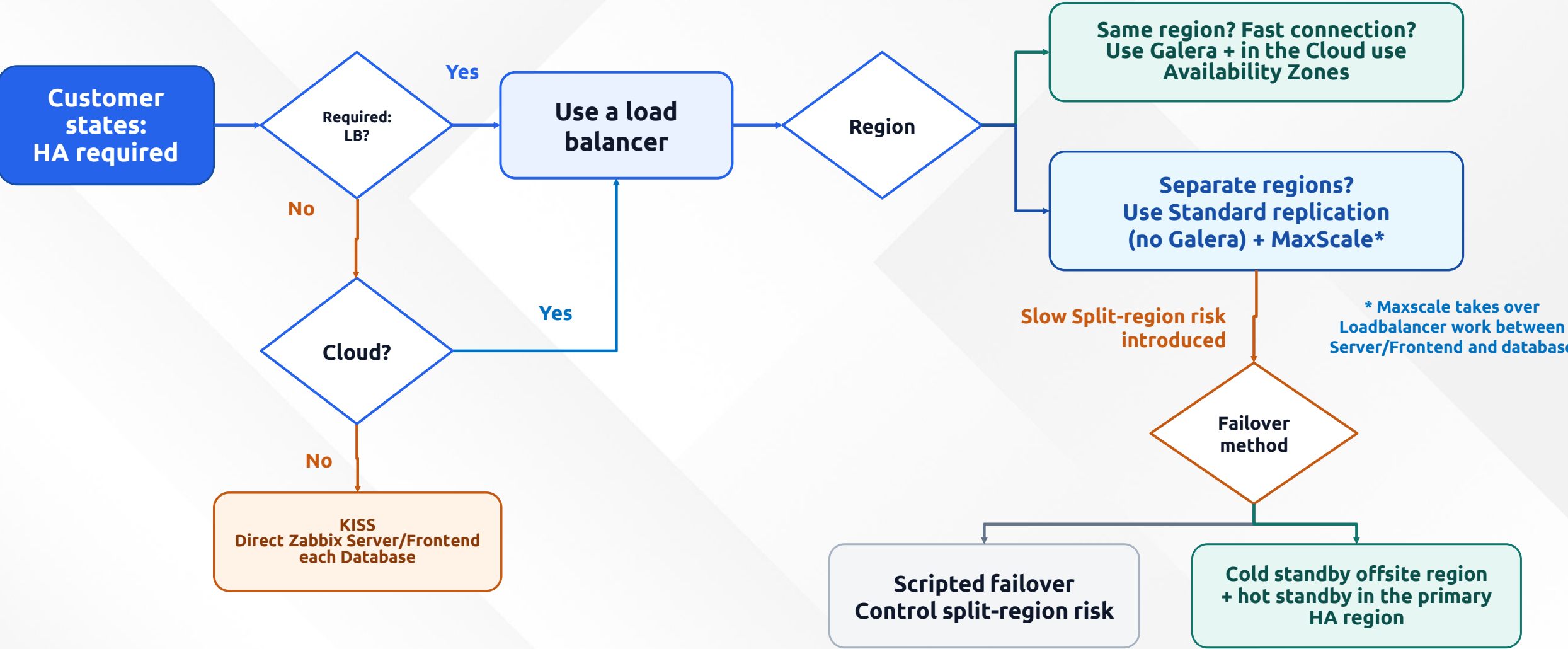
- Datasources to Templates
- Macros
- Etc.

- Manual work

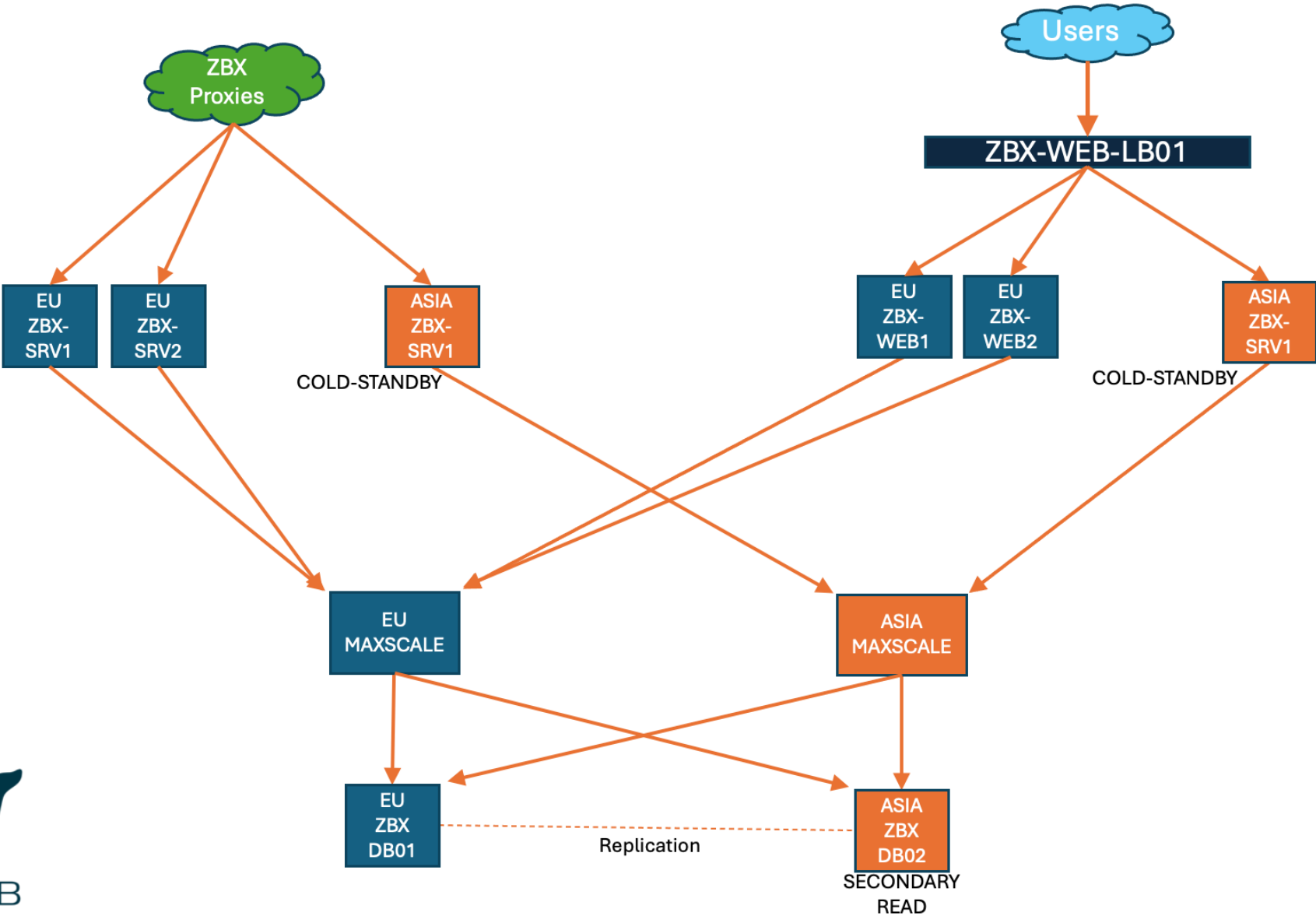
- Ansible install playbooks
 - Install + configure components
 - Configure Zabbix with things like SSM, Host groups, etc.
- Template cleanup
- Cloud monitoring



Design decisions



The design



Why MariaDB?

- Standard primary-replica replication is straightforward across distant regions
- One write primary keeps database ownership explicit
- Native replication integrates directly with MaxScale monitoring and routing
- Because Oracle already owns enough of your stack



Why MaxScale?

- One stable database endpoint for Zabbix servers and frontends
- Routes writes to the primary and reads to available replicas
- Detects topology changes and can redirect traffic after role changes
- Keeps database failover logic out of the application configuration



Automation decisions

- Ansible builds the platform and applies repeatable configuration
- API automation creates host groups, proxies, SSM settings, and encryption
- Scripts handle host and web-monitoring imports
- Manual approval remains where mappings or failover decisions carry risk



Current LogicMonitor setup

- Export devices, device groups, DataSource IDs, and resource properties
- Classify active, dead, cloud-only, and excluded resources
- Map departments, sites, and regions to Zabbix host and proxy groups
- Identify templates, interfaces, credentials, and alert routes before import



Ansible playbooks – Install components

- Zabbix server
- Zabbix frontend
- MariaDB Database
- MariaDB Maxscale

```
base-instance-setup.yml
cluster-mariadb.yml
configure-api-zabbix-actions.yml
configure-api-zabbix-authentication.yml
configure-api-zabbix-autoregistration.yml
configure-api-zabbix-host_templategroups.yml
configure-api-zabbix-mediatypes.yml
configure-api-zabbix-proxies.yml
configure-api-zabbix-reporting.yml
configure-api-zabbix-usergroups.yml
configure-swap.yml
deploy-cert-check-script.yml
install-mariadb.yml
install-maxscale.yml
install-nginx-zabbix-web.yml
install-zabbix-proxy.yml
install-zabbix-reporting.yml
install-zabbix-server.yml
maxscale-ssh-backup-db.yml
maxscale-ssh-backup-maxscale.yml
run-verify-cleanup.yml
```



Ansible playbooks – Zabbix proxies

- Zabbix proxies
 - Local on-site proxies
 - Proxies SSM managed cloud
 - Automated Unique PSK encryption method

```
# =====  
# 8. Queue proxy registration payload for API play  
# =====  
- name: Queue proxy registration payload for Zabbix API  
  tags: [proxy]  
  block:  
    - name: Read proxy PSK file  
      ansible.builtin.slurp:  
        src: /etc/zabbix/zbx_proxy.psk  
      register: proxy_psk_slurp  
      no_log: true  
  
    - name: Set proxy PSK fact  
      ansible.builtin.set_fact:  
        proxy_psk: "{{ proxy_psk_slurp.content | b64decode | trim }}"  
      no_log: true  
  
- name: Store proxy payload locally  
  ansible.builtin.set_fact:  
    zbx_proxy_payload:  
      proxy_name: "{{ ansible_facts['hostname'] }}"  
      tls_psk_identity: "{{ ansible_facts['hostname'] }}"  
      tls_psk: "{{ proxy_psk }}"  
      zbx_proxy_local_address: "{{ ansible_default_ipv4.address }}"  
      no_log: true
```



Ansible playbooks – Configure Zabbix

- Zabbix API config deployments
 - Create hostgroups
 - Create SSM settings
 - Configure and rotate PSK Autoregistration key
 - Configure proxies and proxy groups



Ansible playbooks – Install Zabbix agent

- Secure AWS SSM environment

```
- hosts: linux_agents
  become: yes
  gather_facts: true
  vars:
    ansible_connection: amazon.aws.aws_ssm
    ansible_user: ssm-user
    ansible_aws_ssm_region: "{{ placement.region | default('eu-central-1') }}"
    ansible_aws_ssm_bucket_name: "{{ lookup('env', 'ANSIBLE_AWS_SSM_BUCKET_NAME') | default('ansible-ssm-master-bucket', true) }}"
```

- Windows/Linux mixed environment
- Secure environment, easy to update



Datasource move – Templates built

- LogicMonitor
 - Datasource based
 - Many groovy scripts
- Zabbix
 - Templates
 - Want for native
 - Javascript where not possible

General cleanup while building templates



LoMo datasources vs Zabbix Templates

- LogicMonitor DataSources bundle AppliesTo logic, discovery, collection, graphs, and alerts
- Zabbix templates group reusable items, triggers, discovery rules, dashboards, and web scenarios
- Migration is a semantic mapping from DataSource IDs to supportable templates
- Prefer native Zabbix checks and preprocessing; use JavaScript only when needed



Host import script

- Fetch devices from LogicMonitor
- Filter out dead/cloud/excluded devices
- Resolve Zabbix groups and templates
- Create or update matching Zabbix hosts

```
lm_group_paths = fetch_lm_device_group_paths(int(d["id"]), d)
manual_group_names = classify_zabbix_groups_from_lm_groups(lm_group_paths)

if REQUIRE_MANUAL_GROUP_MATCH and not manual_group_names:
    print(f"[SKIPPED_NO_GROUP_MATCH] {display} ({lm_name}) | lm_groups={lm_group_paths}")
    return {"status": "skipped_no_group_match"}

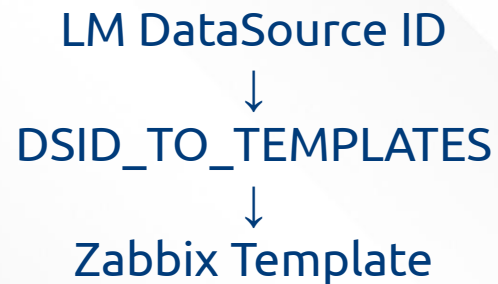
final_group_names = set(manual_group_names)
final_group_names.add(ZBX_GROUP_NAME)
groupids = [zbx_groupid_cache[name] for name in sorted(final_group_names)]

hostid, had_templates, used_ip, used_dns, reason, used_proxy = create_zabbix_host(
    final_group_names, groupids, d, template_id_cache, existing_hosts, update_existing, update_encryption
)
```



Host import script

- LM device groups are mapped to Zabbix host groups
- Export, verify, and then manual group mappings prevent random imports
- Hosts without a mapped Department group fall back to “No department”
- Every imported host also gets the main “Imported hosts from LoMo” group



```
raw_ds_ids = fetch_lm_device_datasource_ids(int(lm_id))
ds_ids = filter_datasource_ids(raw_ds_ids)

has_any_mapping = any(dsid in DSID_TO_TEMPLATES for dsid in ds_ids)
template_ds_ids = ds_ids
if update_existing:
    template_ds_ids &= UPDATE_ONLY_TEMPLATE_DSIDS
templates = templates_for_datasource_ids(template_ds_ids, template_id_cache)
```



Host import script

- LM DataSource IDs decide which Zabbix templates are linked
- Unsupported DataSources are ignored
- Existing Zabbix hosts can be updated with only selected new templates
- Host creation can be blocked if no usable template exists

```
params: Dict[str, Any] = {  
    "host": zbx_host,  
    "name": display or zbx_host,  
    "groups": [{"groupid": gid} for gid in groupids],  
    "status": 1 if CREATE_HOSTS_DISABLED else 0,  
    "tls_connect": ZBX_TLS_CONNECT,  
    "tls_accept": ZBX_TLS_ACCEPT,  
    "tls_psk_identity": ZBX_PSK_IDENTITY,  
    "tls_psk": ZBX_PSK,  
}
```



Manual cleanup

- Host name is normalized before creation
- Groups, templates, proxy group, TLS PSK, and interfaces are added
- IP/DNS is selected from LM properties
- SNMP interface is added when needed

```
if needs_snmp:
    if dev_props is None:
        dev_props = fetch_lm_device_properties(int(lm_id))
    snmp_port = int(get_prop_value(dev_props, "snmp.port") or ZBX_SNMP_PORT)
    snmp_details = build_snmp_details(dev_props)

interfaces = []
if ip_or_resolved or dnsname:
    interfaces.append({
        "type": 1, "main": 1, "useip": base_useip,
        "ip": base_ip, "dns": base_dns, "port": str(ZBX_AGENT_PORT),
    })

if needs_snmp and snmp_details:
    snmp_ip = base_ip if (ip_or_resolved or dnsname) else "0.0.0.0"
    snmp_dns = base_dns if (ip_or_resolved or dnsname) else ""
    snmp_useip = base_useip if (ip_or_resolved or dnsname) else 1
    interfaces.append({
        "type": 2,
        "main": 1,
        "useip": snmp_useip,
        "ip": snmp_ip,
        "dns": snmp_dns,
        "port": str(snmp_port),
        "details": snmp_details,
    })
```



Visualisation

- Rebuild the views operators actually use. Do not chase pixel-for-pixel parity
- Translate resources and properties into host groups, tags, and inventory
- Replace LogicMonitor graphs with Zabbix graphs, maps, and problem-focused widgets
- Validate every view against the decision it is meant to support



Actions / notifications

- Rebuild actions independently from checks and triggers
- Map severities, event tags, user groups, schedules, and escalation steps
- Configure and test each media type before enabling production notifications
- Start with controlled test problems; review the action log after every test



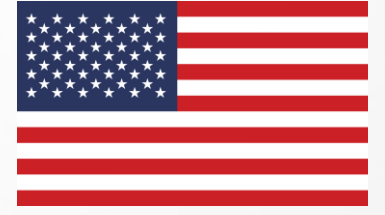
End result? NVPS? Failovers



- 6000 NVPS, Medium-sized environment
- Regional, cloud, local proxies
- Live failover tests, done! Some of them maybe not on purpose..



Contact



Opensource ICT Solutions B.V.

Tabakvest 87/4490
2000 Antwerpen
Belgium
T. +32 (0)3 377 69 36
E. info@oicts.be
W. <https://oicts.be>

Tappersweg 14-024
2031EV Haarlem
The Netherlands
T. +31 (0) 72 743 65 83
E. info@oicts.nl
W. <https://oicts.nl>

Opensource ICT Solutions LTD

2 Leman Street
London E1W 9US
United Kingdom
T. +44 (0) 20 4551 1827
E. info@oicts.co.uk
W. <https://oicts.co.uk>

Opensource ICT Solutions LLC

251 Little Falls Drive
Wilmington, DE 19808
United States
T. +1 (929) 377 1253
E. info@oicts.com
W. <https://oicts.com>



Opensource ICT Solutions