

No Single Point of Failure:

Zabbix, PostgreSQL, and Patroni



Antti Hurme

- Senior Solution Architect at Elisa Oyj – Group IT Services (Finland)
- More than a decade of experience working with datacentre infrastructure
 - Hardware, Storage, Virtualization, Automation...
- Working with Zabbix since version 2.0



OCTOBER 7 ▪ 9, 2026
RIGA ▪ LATVIA

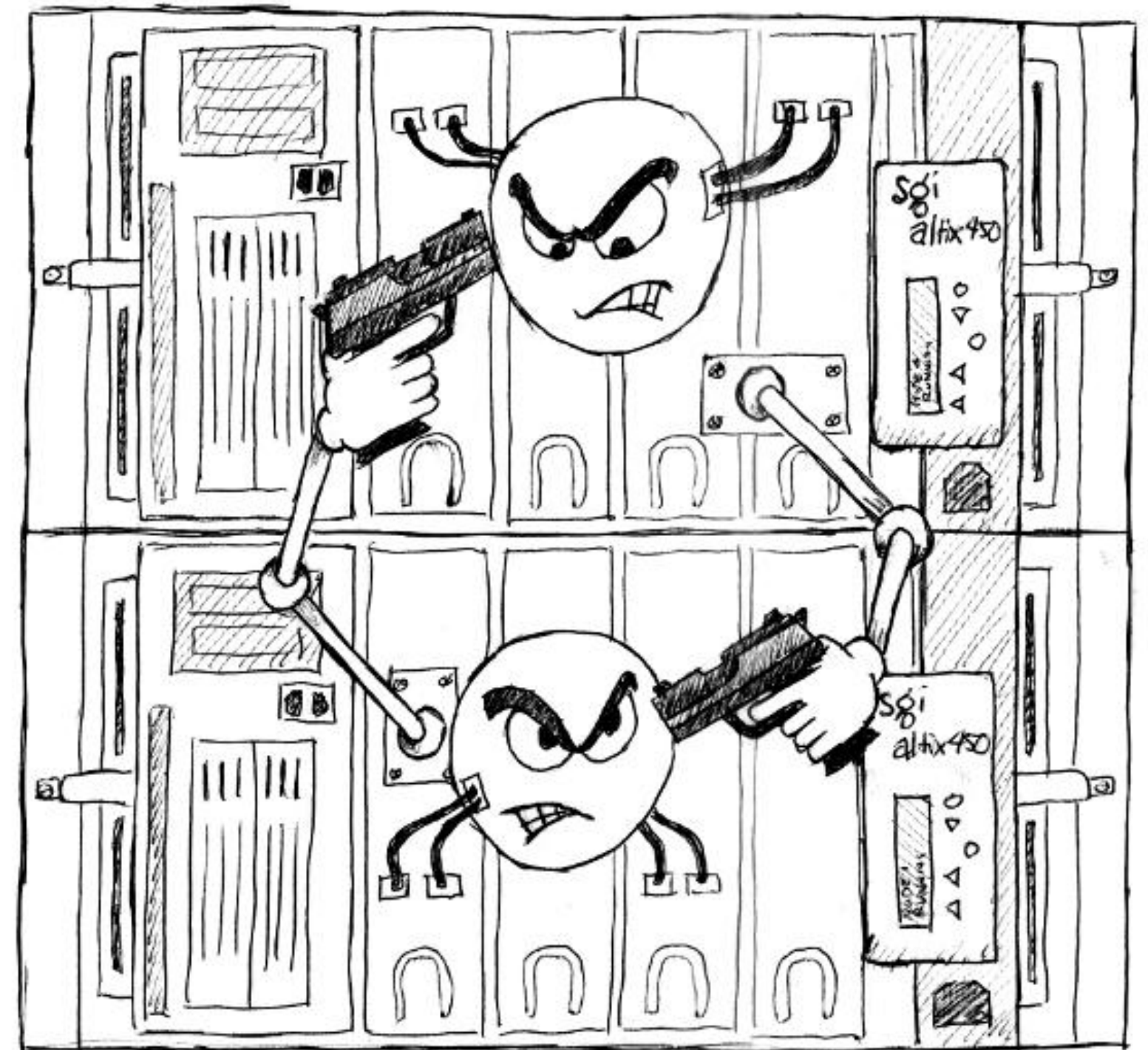


<https://www.null-byte.org>
<https://elisa.fi>

Zabbix High Availability and Scaling

Highly available monitoring is not achieved by adding more servers. It is achieved by ensuring every component knows what happens next when something fails.

- Zabbix High Availability and Scaling
- Scaling out with Proxy Groups
- Patroni and HA PostgreSQL for Zabbix
- TimescaleDB
- Zabbix 8 improvements
- Lessons Learned



DON'T ANYBODY MOVE ...

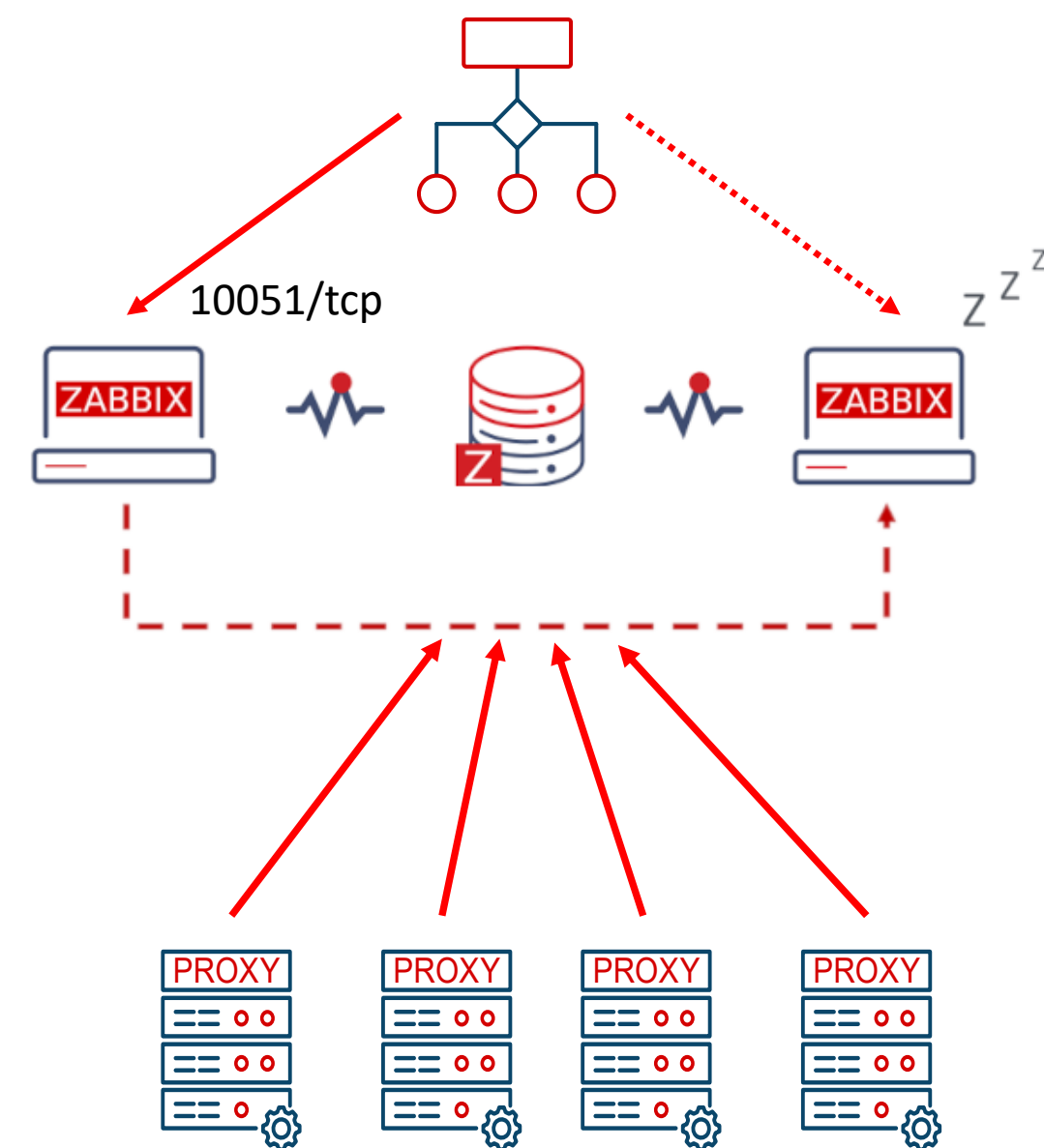
Zabbix High Availability and Scaling

The Basics



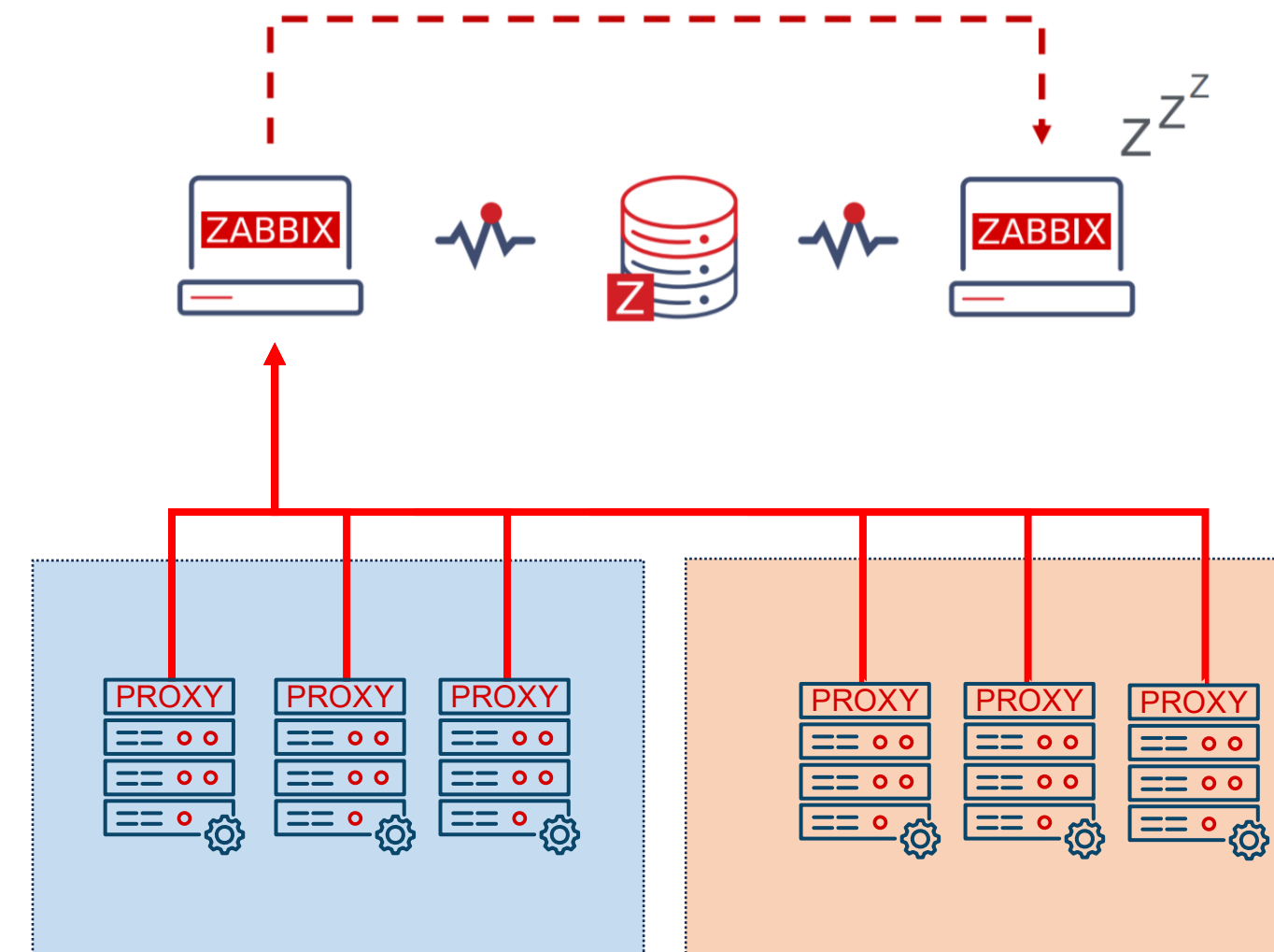
Zabbix High Availability

- Built-in high availability was introduced in Zabbix 6.0 LTS
 - Two or more nodes with a single active server
 - Only the active server node listens to port 10051/tcp, can be used with load-balancer to redirect traffic
 - Shared Database remains as a dependency
- Proxy group high availability was introduced in Zabbix 7.0 LTS
 - Multiple proxy servers with all proxy nodes actively participating
 - Proxy group level fail-over configuration
- Frontend and server roles can be separated
 - Frontend can be scaled independently
 - Server, frontend, and database should be isolated into separate network segments



Zabbix High Availability

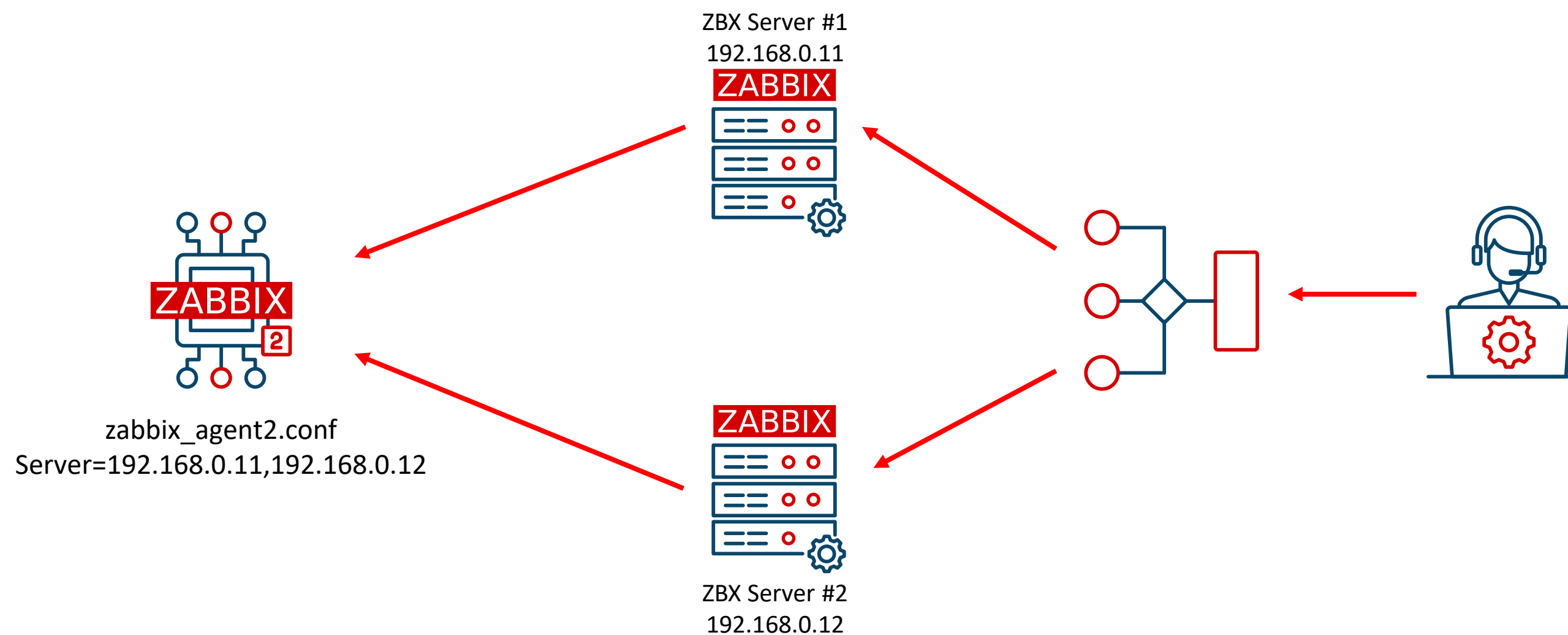
- Zabbix Proxy Groups
 - Automatic Failover between proxy servers
 - Built-in load balancing and scale-out architecture
 - High Availability configuration → Minimum amount of proxies
- Design considerations
 - Failure domain considerations (Datacentre, Network, Virtualization....)
 - Capacity planning based on failure domains
- Proxies can be configured in Disk, Memory, or Hybrid mode
 - Configure according to realistic expected outages
- Take into account your required NVPS and the amount of Zabbix collector and pre-processing processes you need running
- Configure all proxy servers identically!



**Think in terms of failure domains,
not how many proxies you have**

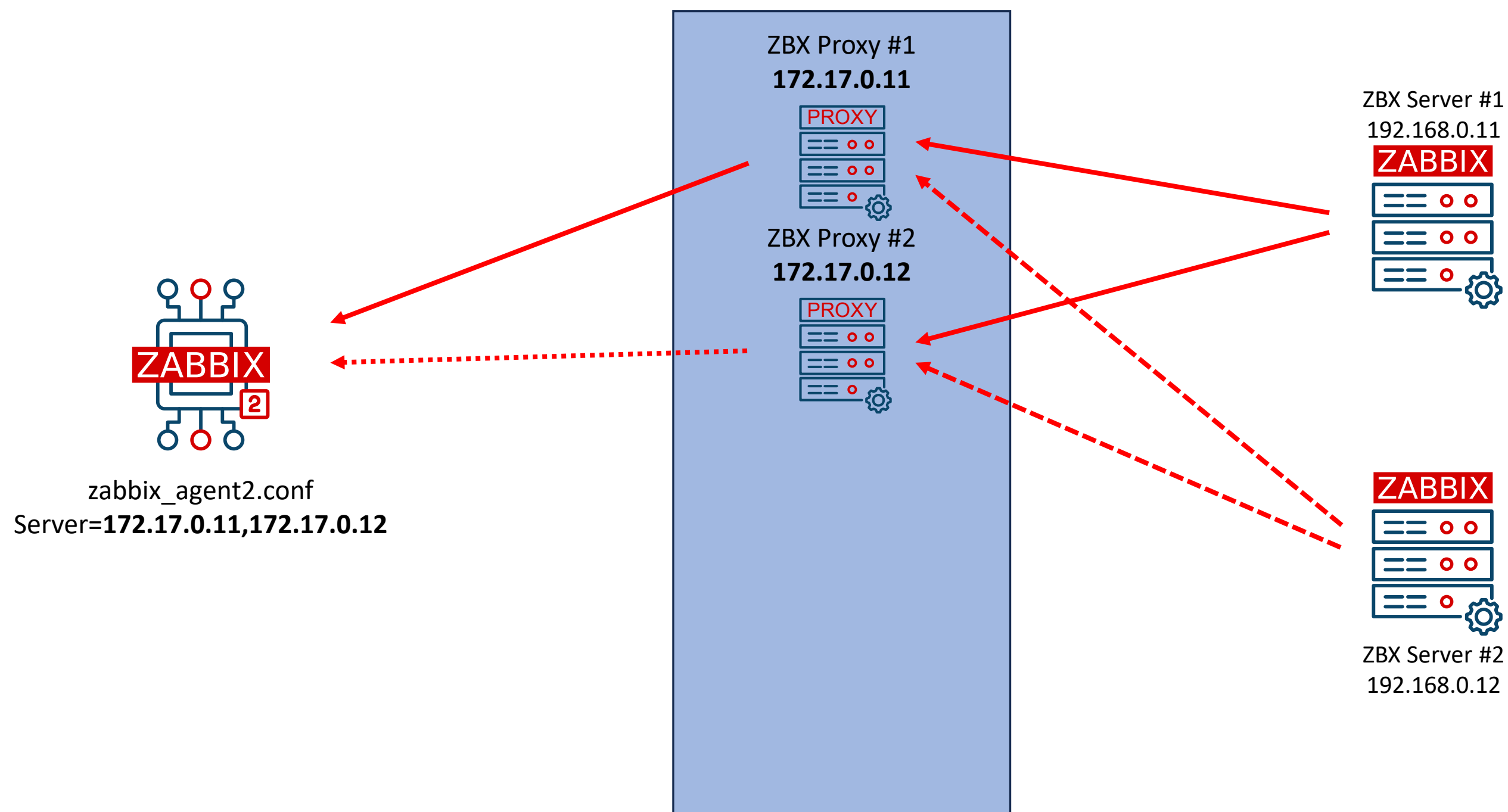
Initial passive-agent configuration

Zabbix Agent → comma separated list of allowed servers



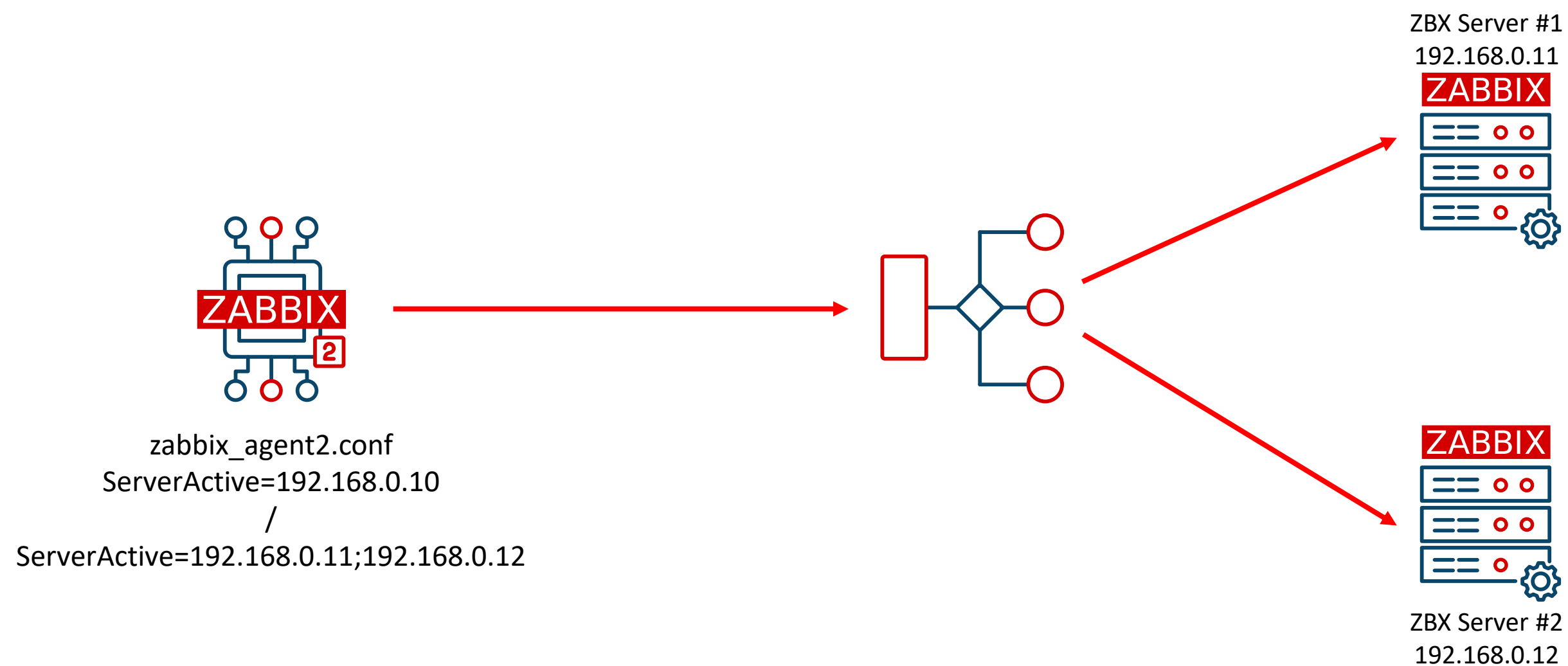
Moving passive checks from servers to the first proxy members

Zabbix Agent → comma separated list of allowed proxy servers



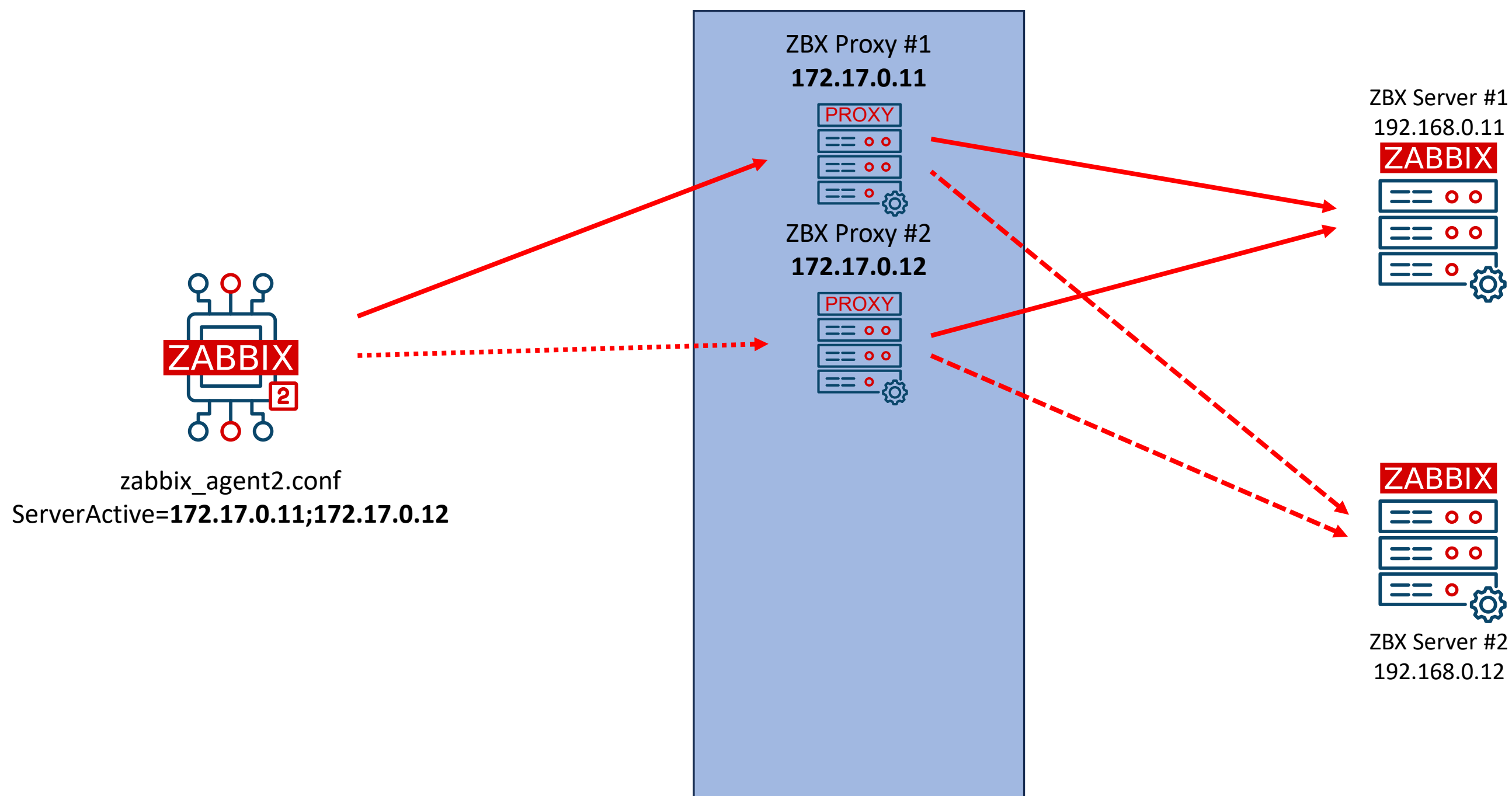
Initial active-agent configuration

- Zabbix Agent Active → semicolon separated list for HA servers
- LB for agents who don't support this feature



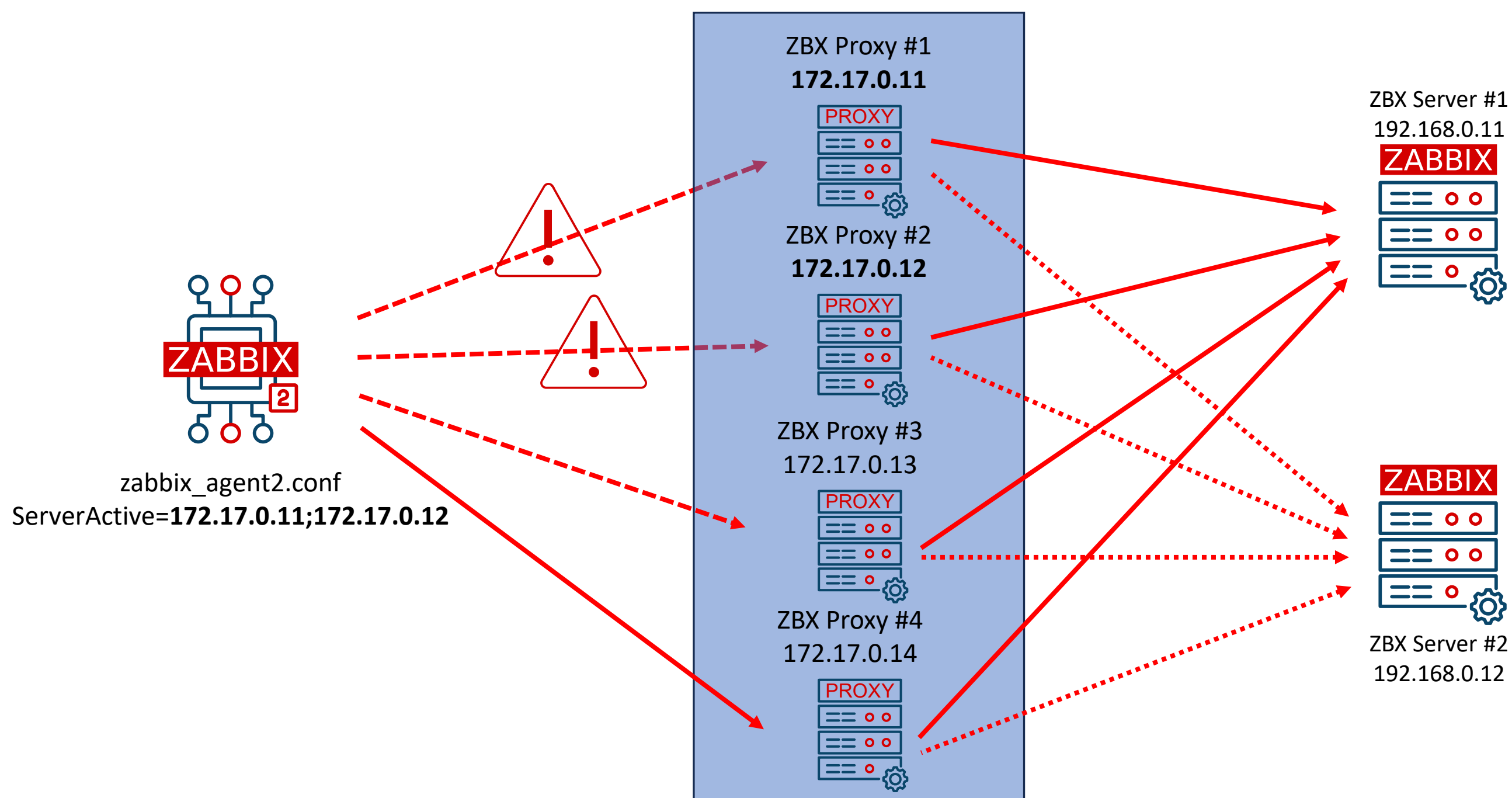
Moving active agent monitoring to proxy group

- Zabbix Agent Active → semicolon separated list for Zabbix Proxy Group servers
- Note: Zabbix Agent 7.0 or newer required!



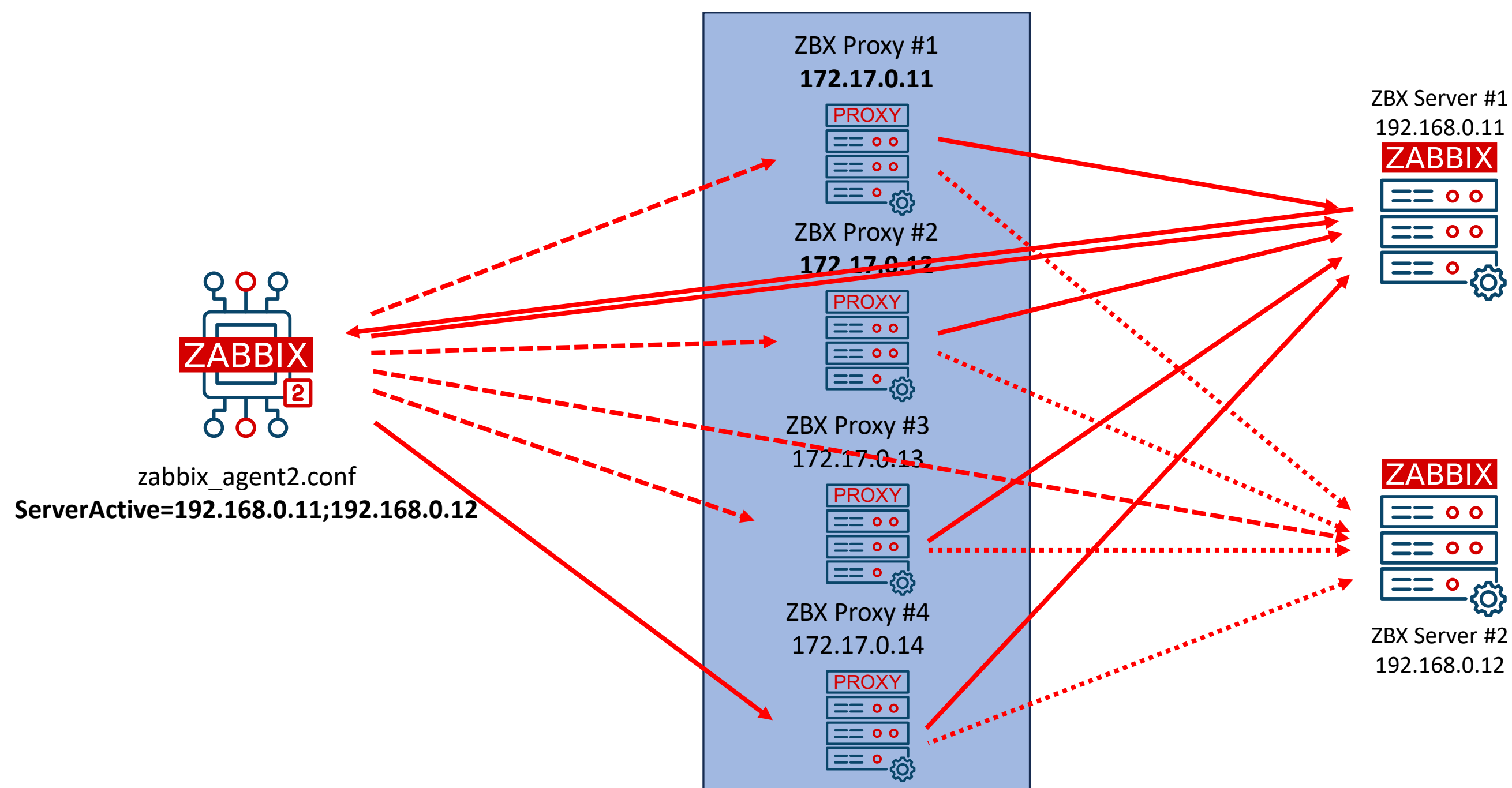
Agent Configuration issues when scaling out

- Zabbix Agent Active → semicolon separated list for Zabbix Proxy Group servers
- Note: Zabbix Agent 7.0 or newer required!



Migrating workloads to Proxy Groups

Zabbix Agent can always use Zabbix Servers as ServerActive target, even with proxies



Why invest in Zabbix HA and Proxy Groups?

Higher Availability

- Removes single points of failure from monitoring collection and processing

Improved Scalability

- Distributed collection reduces load on central infrastructure

Operational Flexibility

- Planned maintenance and upgrades can be performed with reduced service impact

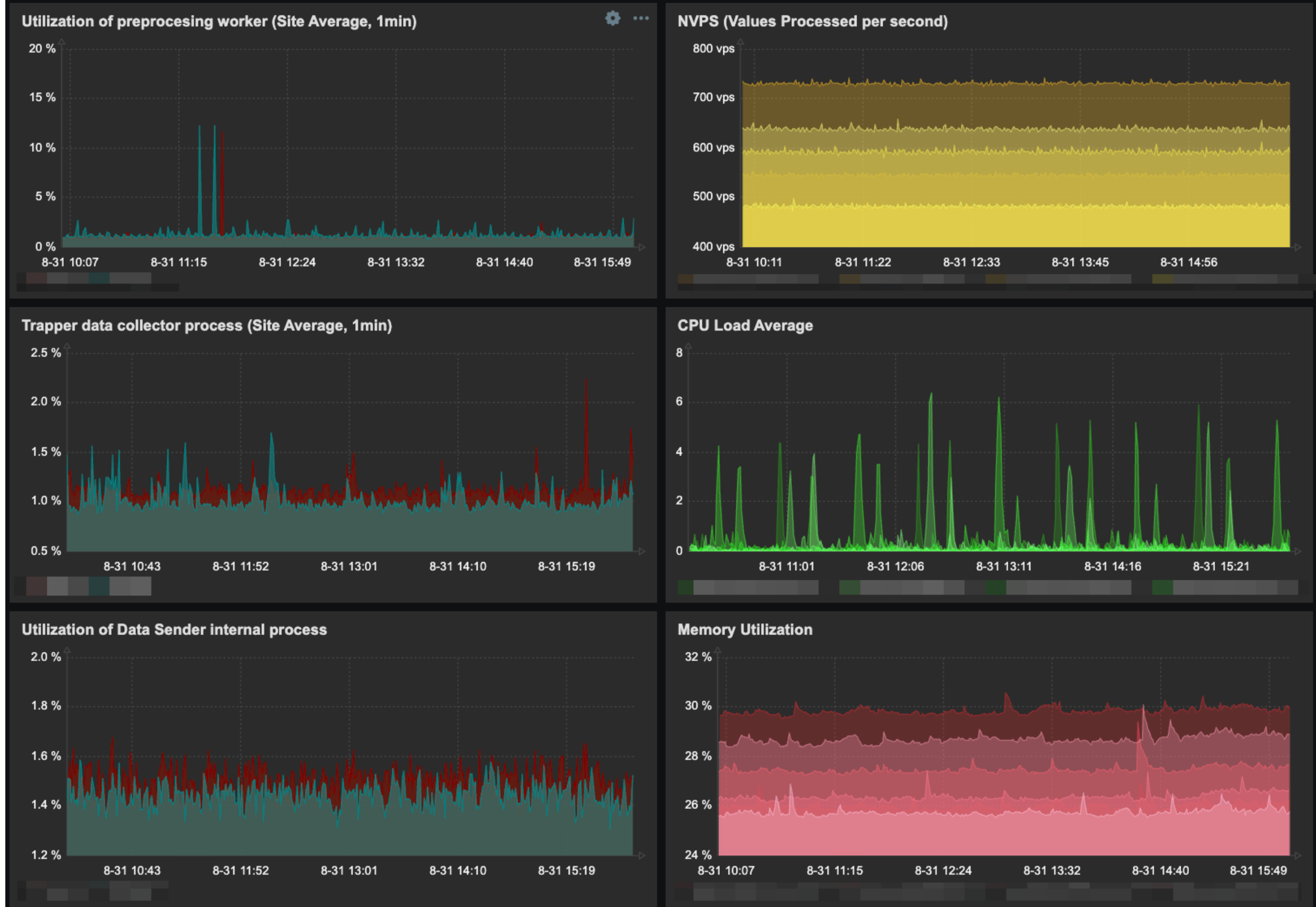
Failure Domain Isolation

- Enables monitoring services to survive infrastructure failures within defined failure domains

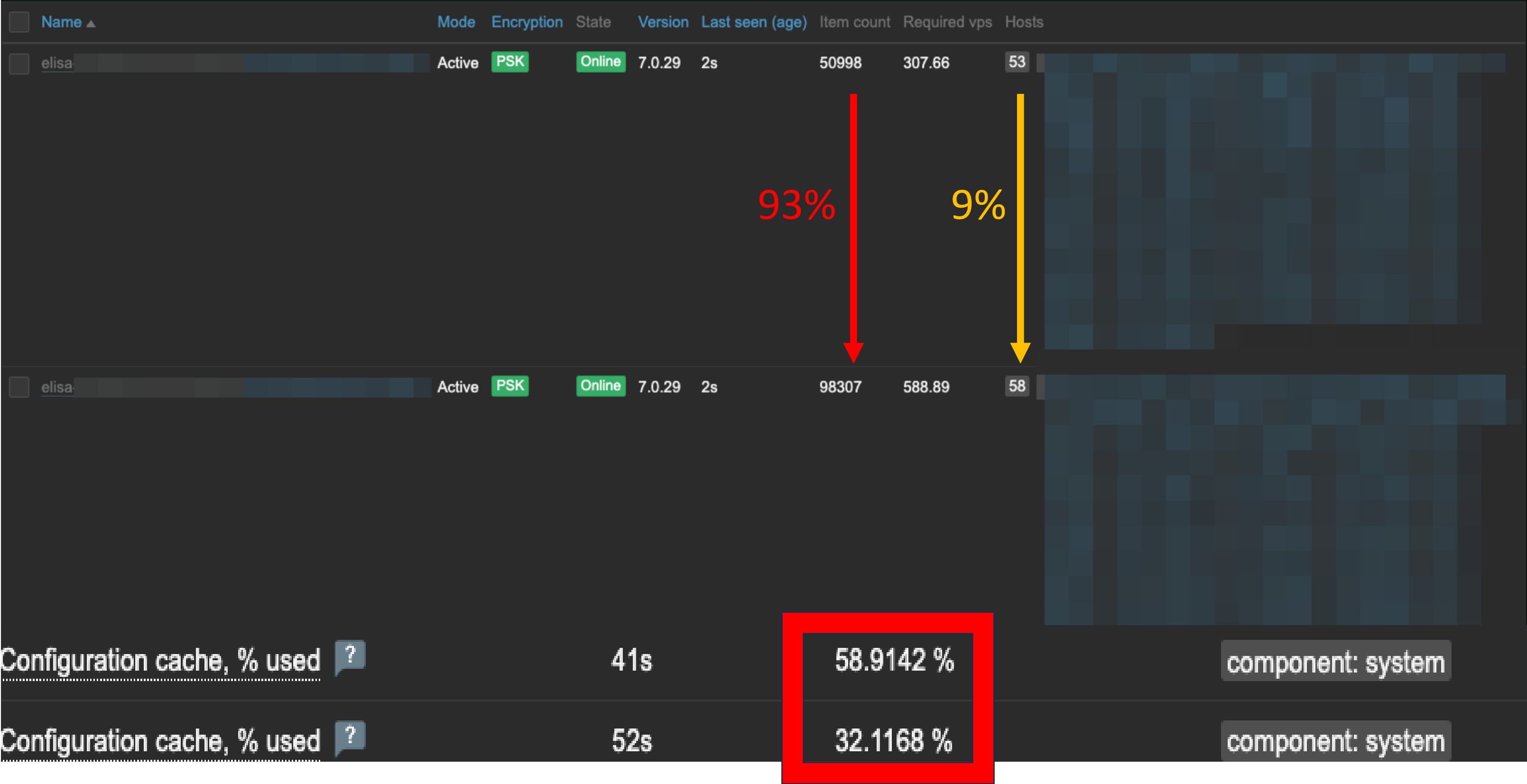
Future-Proof Architecture

- Easier growth from hundreds to thousands of hosts
- Supports increasing data volumes without major architectural redesign

Design for failure, scale for growth.



Real-World Zabbix Proxy Performance and Load



Real-World Zabbix Proxy Load Difference

Database Clustering

Removing the Single Point of Failure from Zabbix Dependencies



MIGRATING FROM MYSQL

ZABBIX



How many of you are using MySQL/MariaDB?

Why use PostgreSQL and Patroni?

- Historically considered operationally complex
- Patroni automates many of the operational tasks required for a highly available PostgreSQL cluster
- Automated Failover and split-brain protection
- PostgreSQL configuration management using Patroni
- Self-healing with automated restarts and tools to manage broken instances
- TimescaleDB extension support with compression

Why use PostgreSQL and Patroni?

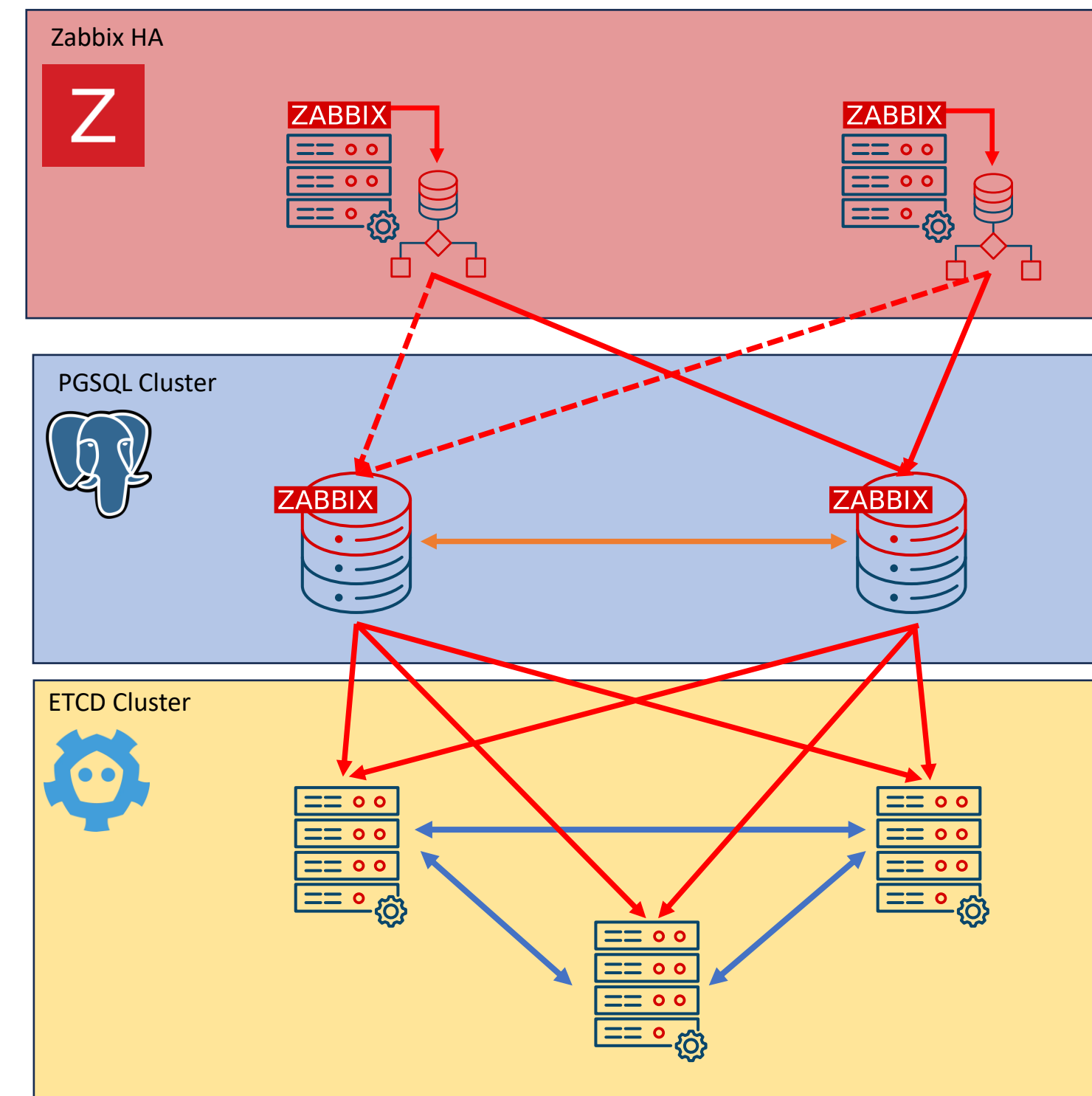
```
[[root@zbx-psql-02 pg_wal]# patronictl -c /etc/patroni/patroni.yml switchover
Current cluster topology
+ Cluster: zbx-psql-cluster (7664904391029510583) +-----+
| Member          | Host          | Role   | State   | TL | Receive LSN | Lag | Replay LSN | Lag |
+-----+-----+-----+-----+-----+-----+-----+-----+
| zbx-psql-01.zbx.summit | 172.17.0.203 | Replica | streaming | 14 | 1B/B20DA9D0 | 0 | 1B/B20DA9D0 | 0 |
| zbx-psql-02.zbx.summit | 172.17.0.204 | Leader  | running  | 14 |              |   |              |   |
+-----+-----+-----+-----+-----+-----+-----+-----+
[Primary [zbx-psql-02.zbx.summit]:
[Candidate ['zbx-psql-01.zbx.summit'] []:
When should the switchover take place (e.g. 2026-09-18T14:42 ) [now]:
Are you sure you want to switchover cluster zbx-psql-cluster, demoting current leader zbx-psql-02.zbx.summit? [y/N]: y
2026-09-18 13:43:03.34881 Successfully switched over to "zbx-psql-01.zbx.summit"
+ Cluster: zbx-psql-cluster (7664904391029510583) +-----+
| Member          | Host          | Role   | State   | TL | Receive LSN | Lag | Replay LSN | Lag |
+-----+-----+-----+-----+-----+-----+-----+-----+
| zbx-psql-01.zbx.summit | 172.17.0.203 | Leader  | running  | 14 |              |   |              |   |
| zbx-psql-02.zbx.summit | 172.17.0.204 | Replica | stopping |   | unknown     |   | unknown     |   |
+-----+-----+-----+-----+-----+-----+-----+-----+
[[root@zbx-psql-02 pg_wal]# patronictl -c /etc/patroni/patroni.yml list
+ Cluster: zbx-psql-cluster (7664904391029510583) +-----+
| Member          | Host          | Role   | State   | TL | Receive LSN | Lag | Replay LSN | Lag |
+-----+-----+-----+-----+-----+-----+-----+-----+
| zbx-psql-01.zbx.summit | 172.17.0.203 | Leader  | running  | 15 |              |   |              |   |
| zbx-psql-02.zbx.summit | 172.17.0.204 | Replica | streaming | 15 | 1B/B3042618 | 0 | 1B/B3042618 | 0 |
+-----+-----+-----+-----+-----+-----+-----+-----+
[[root@zbx-psql-02 pg_wal]#
```

```
[[root@zbx-psql-02 pg_wal]# patronictl -c /etc/patroni/patroni.yml edit-config
---
+++
@@ -7,16 +7,18 @@
     archive_timeout: 600s
     hot_standby: true
     logging_collector: 'on'
+   min_wal_size: 80MB
     max_connections: 500
     max_replication_slots: 10
     max_wal_senders: 5
-   max_wal_size: 1GB
+   max_wal_size: 2GB
     shared_buffers: 2GB
     shared_preload_libraries: timescaledb
     timescaledb.max_background_workers: 16
     wal_keep_segments: 10
     wal_level: replica
     wal_log_hints: true
+   wal_keep_size: 512MB
   pg_hba:
     - host replication replicator 127.0.0.1/32 trust
     - host replication replicator 172.17.0.0/24 md5

[Apply these changes? [y/N]: y
Configuration changed
```

Design Considerations

- Failure Domains and recovery targets
 - How critical is the data you're writing to and reading from the database?
 - Latency between sites?
- Connection methods
 - External load-balancers available? Routing and network domains
- Distributed Consensus
 - ETCD/Zookeeper/Consul
 - Quorum is ideally installed on separate nodes
- Backups
 - How to take your backup (Primary or Stand-by node) and how can you restore it?
- Security
 - Patroni and ETCD must be secured over default configuration



PostgreSQL Upgrading

- Minor upgrades are straightforward
 - Upgrade passive node and wait for replication to catch up
 - Controlled switchover of active node
 - Upgrade the remaining node(s) and allow replication to catch up
- Major upgrades are migration projects
 - Existing replicas cannot simply continue replication after a major version upgrade
 - The upgrade strategy should be driven by acceptable downtime, complexity, and risk
 - Accepting some downtime often results in a significantly simpler upgrade process
 - Verify compatibility between **Zabbix, PostgreSQL, Patroni, and TimescaleDB**
 - Test upgrade procedures before production implementation
 - Always have a validated rollback plan if something goes wrong

PostgreSQL Upgrading

- Be careful with versions and upgrades
 - Workaround is to AllowUnsupportedDBVersions, but I would not recommend that!
 - After upgrading binaries, remember to update the extension in PostgreSQL
- Compression is amazing!

34 results found in 17ms

ZBXNEXT-10720

Updated maximum supported TimescaleDB version to 2.29

7.0.30

API

Server

ZBXNEXT-10669

Updated maximum supported TimescaleDB version to 2.28

7.0.29

API

Server

ZBXNEXT-10618

Updated maximum supported TimescaleDB version to 2.27

7.0.28

API

Server

ZBXNEXT-10553

Updated maximum supported TimescaleDB version to 2.26

7.0.25

API

Server

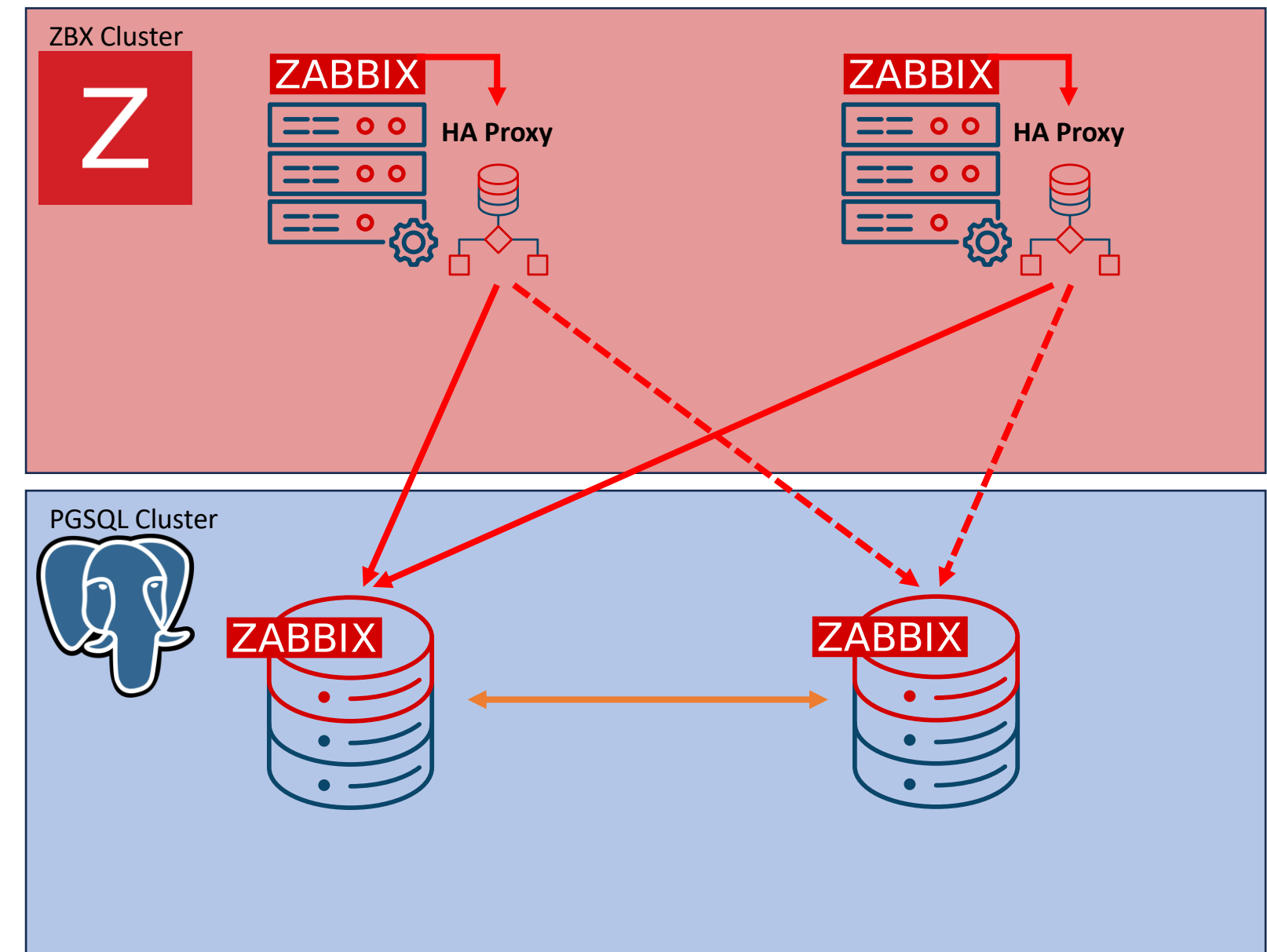
hypertable	table_gib	index_gib	toast_gib	total_gib	recent_uncomp_gib	before_comp_gib	compressed_gib	coverage_pct	comp_ratio	savings_pct
history_uint	143.47	120.65	118.82	382.94	0.00	2315.52	144.59	100.00	16.01	93.76
history	120.73	89.76	127.19	337.68	0.00	1946.81	147.46	100.00	13.20	92.43
history_text	52.73	7.31	25.52	85.56	25.46	659.35	27.27	96.28	24.18	95.86
trends_uint	16.69	13.78	5.94	36.41	10.02	240.51	6.67	96.00	36.06	97.23
trends	9.15	6.59	16.14	31.88	0.00	161.96	16.96	100.00	9.55	89.53
history_str	1.51	0.60	0.11	2.22	0.32	10.27	1.19	96.95	8.66	88.45
auditlog	1.82	0.10	0.21	2.13	0.00	1.49	2.13	100.00	0.70	-43.20
history_log	0.01	0.00	0.00	0.01	0.00	0.01	0.01	100.00	1.01	0.63
history_bin	0.00	0.00	0.00	0.00	0.00					

(9 rows)

Zabbix and HA PostgreSQL today

- Zabbix requires external load-balancer to redirect traffic to the active database node
- HA Proxy can be installed on the Zabbix Server Node itself to provide automatic failover
- Patroni Rest API can be used to determine the primary database node

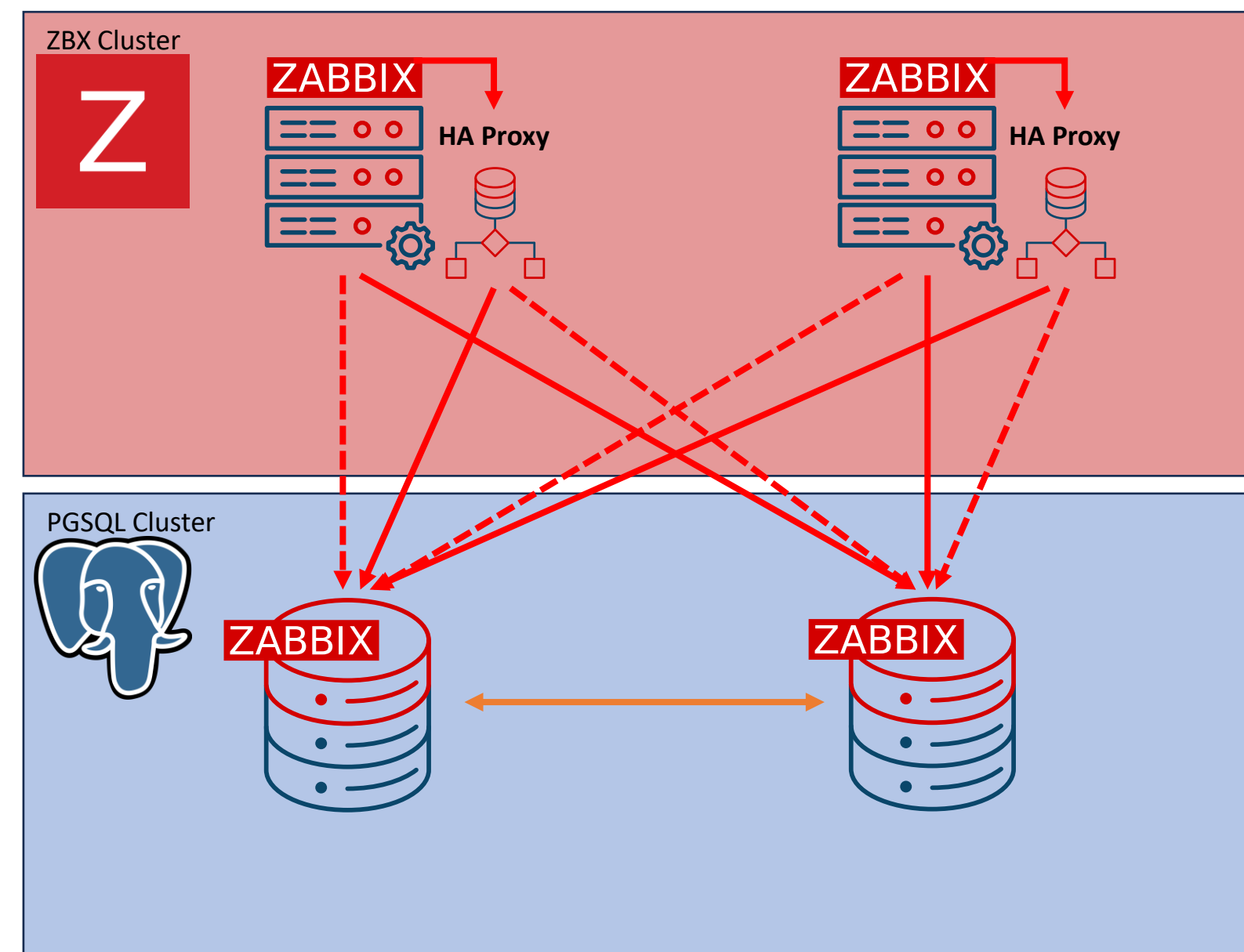
Patroni provides leader election, not a database endpoint.



```
[[root@zbx-psql-02 ~]# curl http://172.17.0.204:8008/leader | jq
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           %         0         0             Dload  Upload    Total   Spent    Left   Speed
100    539      0   539      0         0    401k      0  --:--:-- --:--:-- --:--:--   526k
{
  "state": "running",
  "postmaster_start_time": "2026-09-02 15:43:13.089191+03:00",
  "role": "primary",
  "xlog": {
    "location": 80312853896
  },
  "timeline": 10,
  "replication": [
    {
      "username": "replicator",
      "application_name": "zbx-psql-01.zbx.summit",
      "client_addr": "172.17.0.203",
      "state": "replicating"
    }
  ]
}
```

Zabbix 8 – PostgreSQL HA

- [ZBXNEXT-6492](#)
 - Possibility for usage jdbc connection string for PostgreSQL Patroni cluster
- DBHost= server configuration can be written as a comma-separated list
- Removes the need for a HA Proxy or LB deployment to simplify deployment
- All hosts are tried until a read-write connection is established
- Patroni is still needed to manage the PostgreSQL Cluster configuration, failover, split-brain, etc...



Lessons Learned

- How long can a DB node be down and not lag too much to be out of sync? We had to adjust wal configuration several times
- You can start with a single database node, expand to a PGSQL Cluster on VMs, and then migrate to physical database nodes as your deployment grows
- Upgrading Zabbix and PostgreSQL with minimal downtime actually works
 - PostgreSQL 16 to 18 upgrade was done in a few hours
- Migration of monitored hosts to Zabbix proxies can easily be done gradually
- We've experiences close to zero downtime since we've used started using Zabbix proxies, Zabbix HA and PostgreSQL with Patroni
- Patroni does not check that configuration is valid

postgresql:

parameters:

checkpoint_completion_target: 0.9

effective_cache_size: 64GB

max_connections: 300

max_parallel_maintenance_workers: 2

max_parallel_workers: 8

max_parallel_workers_per_gather: 2

max_wal_senders: 10

max_wal_size: 30GB

max_worker_processes: 8

min_wal_size: 1GB

shared_buffers: 64GB

wal_keep_size: 50GB

work_mem: 32MB



shared_buffers (integer)

Sets the amount of memory the database server uses for shared memory buffers. The default is typically 32 megabytes (32MB),

Thank You!

Any Questions?