DEPLOYING AND CONFIGURING ZABBIX 5.4 IN A MULTI-TENANT ENVIRONMENT
MONITORING REQUIREMENTS OF MULTI-TENANT ENVIRONMENTS

- Granular role/permission schema
- Ability to process large amounts of data
- Redundancy
- Infrastructure scalability
- Support of different monitoring approaches
SUPPORTED MONITORING APPROACHES - AGENT

- Supports passive (polling) and active modes (trapping)

Supports deployment on different platforms (Windows/Unix-like) and provides access to OS level metrics, collecting data from log files and metric collection via custom scripts and commands.
WHAT IF AGENT IS NOT AN OPTION?

- SNMP, HTTP, IPMI and SSH agentless monitoring
- Simple checks (ICMP pings, port status)
- Database and Java monitoring
- External scripts
- Data aggregations and calculations
- VMware monitoring
- Web monitoring
- Synthetic monitoring
LATEST IMPROVEMENTS

☑️ Version 5.2
  ◦ Permission logic reworked – user roles have been implemented
  ◦ Support for external vault
  ◦ Granular frontend configuration

☑️ Version 5.4
  ◦ Ability to send scheduled reports
  ◦ Performance improvements
ZABBIX AND MULTI-TENANT ENVIRONMENTS

Monitoring client ABC

Service provider

Monitoring Client BCD
ZABBIX PROXY

Collects and preprocesses the data. Supports all of the metric collection methods.

☑ Data gets compressed before forwarding it to the server
☑ Data collection continues even in case of network interruptions
☑ A single connection is used to send the data to the server
☑ Enables remote script execution
☑ Can be used to improve scalability
ZABBIX PROXY

Select the direction of the connection

Active proxy

Passive proxy
DATA PREPROCESSING - THROTTLING

- By using preprocessing, we can decrease the amount of stored data by discarding repeating values.
- This preprocessing method is called Throttling.

<table>
<thead>
<tr>
<th>Discard Unchanged</th>
<th>Discard Unchanged Heartbeat 2 minutes</th>
</tr>
</thead>
<tbody>
<tr>
<td>00:00 value - 1</td>
<td>00:00 value - 1</td>
</tr>
<tr>
<td>00:01 value - 2 OK</td>
<td>00:05 value - 1 dropped</td>
</tr>
<tr>
<td>00:02 value - 2 Dropped</td>
<td>01:05 value - 1 dropped</td>
</tr>
<tr>
<td>00:02 value - 2 Dropped</td>
<td>01:55 value - 1 dropped</td>
</tr>
<tr>
<td>10:02 value - 2 Dropped</td>
<td>02:00 value - 1 SENT</td>
</tr>
<tr>
<td>50:02 value - 2 Dropped</td>
<td>02:01 value - 1 dropped</td>
</tr>
</tbody>
</table>
PERMISSIONS

☑ Permissions are based on User group to host group relations
☑ Three user types – User, Admin, Super admin
☑ User Roles
PERMISSIONS - ROLES

Roles grant the ability to configure access to specific UI elements, actions and restrict API calls in a granular fashion.

Access to actions

- Create and edit dashboards and screens
- Create and edit maps
- Create and edit maintenance
- Acknowledge problems
- Close problems
- Change severity
- Add problem comments
- Execute scripts
PERMISSIONS – USER GROUPS

- Access to hosts is defined on User group level
- Users can have Full/Read only/Deny permissions on particular host groups

<table>
<thead>
<tr>
<th>User group</th>
<th>Permissions</th>
<th>Tag filter</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Permissions</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Host group</td>
<td>Permissions</td>
</tr>
<tr>
<td></td>
<td>Application Servers</td>
<td>Read-write</td>
</tr>
<tr>
<td></td>
<td>Linux servers</td>
<td>Read-write</td>
</tr>
<tr>
<td></td>
<td>Network devices</td>
<td>Read-write</td>
</tr>
<tr>
<td></td>
<td>Riga Servers</td>
<td>Read-write</td>
</tr>
</tbody>
</table>
PERMISSIONS – HOST GROUPS

- Using group hierarchy can be very beneficial in such scenarios
HIGH AVAILABILITY

- Having near 100% uptime can be especially critical for such environments
- HA can be used to minimize downtime and add redundancy
- Tools such as PCS, Corosync, Pacemaker are used to enable the HA
- Out of the box HA planned for Zabbix 6.0
HIGH AVAILABILITY - NOTES

- Odd number of nodes should be used to achieve quorum
- Only Active/Passive cluster architecture is supported
- Two nodes should never be active at the same time
- STONITH mechanism can prevent such a scenario (recommended)
- Failure to abide by these restrictions can result in database consistency issues
HIGH AVAILABILITY - PROXY

HA solution can also be implemented for Zabbix proxies, though it can add a significant HA configuration management overhead.

HA for proxies can also be implemented by using Zabbix API scripts:

```json
{
    "jsonrpc": "2.0",
    "method": "host.massupdate",
    "params": {
        "hosts": [
            {
                "hostid": "10396"
            },
            {
                "hostid": "10397"
            }
        ],
        "proxy_hostid": 10398
    },
    "auth": "d9040f24a183638e00940fe64059420b",
    "id": 1
}
```
DATABASE REPLICATION

- Database replication can be used as a form of redundancy for the Zabbix DB
- Multiple replication approaches are supported (Master/Slave, Master/Master, Multi-master)
DATABASE PERFORMANCE TUNING

- In most scenarios using the default DB settings is not sufficient in such environments.
- Hardware requirements should also be taken into account. The DB hosts should have sufficient memory and storage should be selected according to the I/O requirements (SSD/RAID setup).
- Ideally, consult a DBA for optimal DB configuration.

![PGTune](image)
HISTORY TABLE PARTITIONING

- On large instances the Housekeeper cannot keep up with cleaning out history and trend tables in a timely fashion
- This has a negative effect on overall DB performance
- Partitioning is recommended for history/trend tables
- This is supported out of the box with PostgreSQL + TimescaleDB

ALTER TABLE trends_uint PARTITION BY RANGE ( clock)
(PARTITION p2020_10 VALUES LESS THAN (UNIX_TIMESTAMP("2020-11-01 00:00:00")) ENGINE = InnoDB,
PARTITION p2020_11 VALUES LESS THAN (UNIX_TIMESTAMP("2020-12-01 00:00:00")) ENGINE = InnoDB,
PARTITION p2020_12 VALUES LESS THAN (UNIX_TIMESTAMP("2021-01-01 00:00:00")) ENGINE = InnoDB,
PARTITION p2021_01 VALUES LESS THAN (UNIX_TIMESTAMP("2021-02-01 00:00:00")) ENGINE = InnoDB,
PARTITION p2021_02 VALUES LESS THAN (UNIX_TIMESTAMP("2021-03-01 00:00:00")) ENGINE = InnoDB,
PARTITION p2021_03 VALUES LESS THAN (UNIX_TIMESTAMP("2021-04-01 00:00:00")) ENGINE = InnoDB);

- Community provided partitioning scripts are publicly available
  https://github.com/OpensourceICTSolutions/zabbix-mysql-partitioning-perl
HISTORY TABLE PARTITIONING - TIMESCALEDB

- TimescaleDB plugin for PostgreSQL DB backends supports out of the box partitioning
- No need to rely on community scripts
- TimescaleDB parameter \textit{chunk\_time\_interval} defines the partition chunk size.
- In addition, TimescaleDB provides the compression of history/trends
- Compressed data becomes read-only and cannot be changed post compression
TO-DO LIST

☑ Zabbix version with role support is deployed (>=5.2)
☑ Proxy servers are deployed in the client data centers
☑ Application and DB servers use HA/Replication solutions
☑ Partitioning has been implemented to improve the DB performance
☑ Throttling has been implemented to reduce the amount of incoming data
☑ Data base is tuned in accordance with the available hardware resources
QUESTIONS?

ARTŪRS LONTONS
TECHNICAL SUPPORT ENGINEER
ZABBIX
THANK YOU!

ARTŪRS LONTONS
TECHNICAL SUPPORT ENGINEER
ZABBIX