Zabbix Advanced

Aula 11: Monitoramento de Bancos de Dados

MySQL, MariaDB e PostgreSQL

4Linux - Curso Avançado

Agenda do Dia

1. **Fundamentos do Monitoramento de BD**
    - Por que monitorar, métricas essenciais
2. **MySQL/MariaDB com Template Oficial**
    - Zabbix Agent 2, Plugin MySQL nativo
3. **PostgreSQL com Template Oficial**
    - Zabbix Agent 2, Plugin PostgreSQL nativo

Agenda do Dia (cont.)

4. **Métricas Avançadas**
    - Conexões, IOPS, buffers, slow queries, cache
5. **Triggers para Degradação**
    - Falha de conexão, performance
6. **Laboratórios Práticos com Agent 2**
    - Lab 1: MySQL local com template
    - Lab 2: Dashboard MySQL
    - Lab 3: PostgreSQL Coffee Shop

PARTE 1

Fundamentos do Monitoramento de BD

Por Que Monitorar Bancos de Dados?

Bancos de dados são:

- 🎯 Componente crítico da aplicação
- 📊 Gargalo comum de performance
- 💾 Armazenamento de dados sensíveis
- ⚡ Ponto único de falha (SPOF)

Problemas comuns:

- Queries lentas (slow queries)
- Falta de conexões disponíveis
- Disk I/O alto
- Cache hit ratio baixo
- Replicação com lag

Métricas Essenciais de BD

Categoria	Métricas	Importância
Conexões	Ativas, máximo, uso %	ALTA
Performance	QPS, TPS, slow queries	ALTA
Recursos	CPU, RAM, Disk I/O	ALTA
Cache	Hit ratio, buffer pool	MÉDIA
Replicação	Lag, status	ALTA (se usar)
Bloqueios	Locks, deadlocks	MÉDIA

Estratégias de Monitoramento

Abordagem recomendada:

1. **Templates oficiais do Zabbix ⭐ USAR SEMPRE PRIMEIRO!**
   - Prontos, testados, mantidos pela comunidade
   - **MySQL by Zabbix agent 2**
   - **PostgreSQL by Zabbix agent 2**
   - Já incluem: conexões, performance, cache, replicação
   - ✅ Cobrem 95% das necessidades

2. **UserParameters customizados** (quando necessário)
    - Apenas para métricas específicas do negócio
    - Queries customizadas não cobertas pelo template
    - Exemplo: Tamanho de tabelas específicas, queries de negócio

Permissões de Acesso ao BD

Princípio do menor privilégio:

❌ NÃO fazer:

```
GRANT ALL PRIVILEGES ON *.* TO 'zabbix'@'localhost';
```

✅ Fazer (MySQL - conforme doc oficial v7.0):

```
-- Permissões mínimas para monitoramento
GRANT REPLICATION CLIENT,PROCESS,SHOW DATABASES,SHOW VIEW ON *.* TO 'zbx_monitor'@'%';
```

✅ Fazer (PostgreSQL 10+ - conforme doc oficial v7.0):

```
-- Usar role pg_monitor (recomendado)
CREATE USER zbx_monitor WITH PASSWORD 'senha' INHERIT;
GRANT pg_monitor TO zbx_monitor;
```

Justificativa:

- Zabbix só precisa LER métricas
- Não precisa ESCREVER ou ALTERAR dados
- Reduz risco de segurança
- Seguimos documentação oficial v7.0 LTS

PARTE 2

MySQL/MariaDB

Template MySQL by Zabbix Agent 2

O que o template já inclui:

✅ **Conexões:** Threads connected, Max connections, Connection errors
✅ **Performance:** QPS, TPS, Slow queries
✅ **Cache:** InnoDB buffer pool hit ratio, key cache hit ratio
✅ **Replicação:** Slave status, Seconds behind master
✅ **Disco I/O:** InnoDB data reads/writes
✅ **Locks:** Table locks, deadlocks
✅ **Uptime, Version, Status**

🎯 **Recomendação:** Use este template SEMPRE! Não crie items do zero.

Passo 1: Criar Usuário de Monitoramento MySQL

No MySQL/MariaDB:

```
-- Criar usuário (% permite acesso de qualquer host)
CREATE USER 'zbx_monitor'@'%' IDENTIFIED BY '4linux';

-- Conceder permissões conforme documentação oficial
GRANT REPLICATION CLIENT,PROCESS,SHOW DATABASES,SHOW VIEW ON *.* TO 'zbx_monitor'@'%';

-- Aplicar
FLUSH PRIVILEGES;
```

Para MariaDB 10.5.8-5+ com replicação:

```
GRANT REPLICATION CLIENT,PROCESS,SHOW DATABASES,SHOW VIEW,SLAVE MONITOR ON *.* TO 'zbx_monitor'@'%';
```

Testar:

```
mysql -uzbx_monitor -p'4linux' -e "SHOW STATUS"
```

Passo 2: Configurar Zabbix Agent 2 para MySQL

Editar arquivo de configuração:

```
sudo nano /etc/zabbix/zabbix_agent2.conf
```

Adicionar configuração do plugin MySQL:

```
Plugins.Mysql.Uri=tcp://zbx_monitor:4linux@localhost:3306
Plugins.Mysql.KeepAlive=300
```

⚠️ IMPORTANTE:

- Credenciais ficam no `zabbix_agent2.conf`
- **NÃO precisa** criar `.my.cnf` (método antigo do Agent 1)
- Plugin MySQL nativo do Agent 2 usa a URI diretamente

Reiniciar agent:

Passo 3: Testar Plugin MySQL

Testar se o plugin está funcionando:

```
# Testar conexão ao MySQL via agent
zabbix_agent2 -t mysql.ping

# Testar coleta de métricas
zabbix_agent2 -t mysql.version
zabbix_agent2 -t mysql.status[Threads_connected]
```

Saída esperada:

```
mysql.ping [s|1]
mysql.version [s|8.0.35-0ubuntu0.22.04.1]
mysql.status[Threads_connected] [u|8]
```

✅ Plugin MySQL configurado e funcionando!

Passo 4: Aplicar Template no Zabbix Frontend

No Zabbix Frontend:

```
1. Configuration → Hosts → [Selecionar host MySQL]

2. Templates → Link new templates

3. Buscar: "MySQL by Zabbix agent 2"

4. Add → Update
```

Aguardar 1-2 minutos para coleta iniciar.

Verificar:

```
Monitoring → Latest data
Filter: MySQL
```

✅ **Template aplicado! 50+ items já coletando dados automaticamente!**

Métricas MySQL Incluídas no Template

Categoria	Métricas do Template
Conexões	Threads connected, Max connections, Connection errors, Aborted connections
Performance	Queries per second, Transactions per second, Slow queries
Cache	InnoDB buffer pool hit ratio, Key cache hit ratio
Disco I/O	InnoDB data reads/writes, InnoDB log writes
Replicação	Slave status, Seconds behind master
Locks	Table locks waited, InnoDB row lock waits

Triggers Incluídas no Template

O template já inclui triggers prontas:

✅ **MySQL: Service is down** (Disaster)
✅ **MySQL: Too many connections (> 80%)** (Warning)
✅ **MySQL: Slow queries rate is high** (Warning)
✅ **MySQL: InnoDB buffer pool hit rate is low (< 95%)** (Warning)
✅ **MySQL: Replication lag is high (> 30s)** (Warning)
✅ **MySQL: Failed to fetch info** (Warning)

Você pode ajustar os thresholds nas macros do template:

```
{$MYSQL.CONN_USAGE.MAX.WARN} = 80
{$MYSQL.SLOW_QUERIES.MAX.WARN} = 3
{$MYSQL.BUFF_UTIL.MIN.WARN} = 50
```

PARTE 3

PostgreSQL

Template PostgreSQL by Zabbix Agent 2

O que o template já inclui:

✅ **Conexões:** Active connections, Max connections, Connection rate
✅ **Performance:** Commits/Rollbacks per second, Transactions per second
✅ **Cache:** Cache hit ratio, Buffers, Shared memory
✅ **Replicação:** Replication lag, Replication status
✅ **Disco I/O:** Blocks read/hit, Tuples fetched/returned
✅ **Database size:** Size per database (LLD)
✅ **Locks:** Deadlocks, waiting queries

🎯 **Recomendação:** Use este template! Não crie scripts customizados.

Passo 1: Criar Usuário PostgreSQL

Conectar como postgres:

```
sudo -u postgres psql
```

Para PostgreSQL 10 e superior (recomendado):

```
-- Criar usuário com INHERIT (conforme doc oficial)
CREATE USER zbx_monitor WITH PASSWORD '4linux' INHERIT;

-- Conceder role pg_monitor
GRANT pg_monitor TO zbx_monitor;
```

Para PostgreSQL 9.6 e inferior:

```
CREATE USER zbx_monitor WITH PASSWORD '4linux';
GRANT SELECT ON pg_stat_database TO zbx_monitor;
ALTER USER zbx_monitor WITH SUPERUSER;
```

Passo 2: Configurar pg_hba.conf (se necessário)

Permitir conexão do usuário zbx_monitor:

```
sudo nano /etc/postgresql/14/main/pg_hba.conf
```

Adicionar linha:

```
# TYPE  DATABASE        USER            ADDRESS                 METHOD
host    all             zbx_monitor     127.0.0.1/32            md5
```

Recarregar configuração:

```
sudo systemctl reload postgresql
```

Passo 3: Configurar Zabbix Agent 2 para PostgreSQL

Editar arquivo de configuração:

```
sudo nano /etc/zabbix/zabbix_agent2.conf
```

Adicionar configuração do plugin PostgreSQL:

```
Plugins.Postgres.Uri=tcp://zbx_monitor:4linux@localhost:5432/postgres
Plugins.Postgres.KeepAlive=300
```

⚠️ Opcional - SSL:

```
# Para conexões SSL
Plugins.Postgres.Uri=tcp://zbx_monitor:4linux@localhost:5432/postgres?sslmode=require
```

Reiniciar agent:

Passo 4: Testar Plugin PostgreSQL

Testar se o plugin está funcionando:

```
# Testar conexão ao PostgreSQL via agent
zabbix_agent2 -t pgsql.ping

# Testar coleta de métricas
zabbix_agent2 -t pgsql.db.discovery
zabbix_agent2 -t pgsql.connections
```

Saída esperada:

```
pgsql.ping [s|1]
pgsql.db.discovery [s|{"data":[{"{#DBNAME}":"postgres"},...]}]
pgsql.connections [u|12]
```

✅ Plugin PostgreSQL configurado e funcionando!

Passo 5: Aplicar Template no Zabbix Frontend

No Zabbix Frontend:

```
1. Configuration → Hosts → [Selecionar host PostgreSQL]

2. Templates → Link new templates

3. Buscar: "PostgreSQL by Zabbix agent 2"

4. Add → Update
```

Aguardar 1-2 minutos para coleta iniciar.

Verificar:

```
Monitoring → Latest data
Filter: PostgreSQL
```


✅ **Template aplicado! 40+ itens já coletando dados automaticamente!**

Triggers Incluídas no Template PostgreSQL

O template já inclui triggers prontas:

✅ **PostgreSQL: Service is down** (Disaster)
✅ **PostgreSQL: Too many connections (> 80%)** (Warning)
✅ **PostgreSQL: Checkpoints are happening too frequently** (Warning)
✅ **PostgreSQL: Cache hit ratio is low (< 90%)** (Warning)
✅ **PostgreSQL: Replication lag is high** (Warning)
✅ **PostgreSQL: Failed to fetch info** (Warning)

Você pode ajustar os thresholds nas macros do template:

```
{$PG.CONN_TOTAL_PCT.MAX.WARN} = 80
{$PG.CACHE_HITRATIO.MIN.WARN} = 90
{$PG.REPL_LAG.MAX.WARN} = 10
```

PARTE 4

Métricas Avançadas

Slow Queries - MySQL

Habilitar slow query log:

```
-- No MySQL
SET GLOBAL slow_query_log = 'ON';
SET GLOBAL long_query_time = 2;  -- 2 segundos
SET GLOBAL slow_query_log_file = '/var/log/mysql/slow.log';
```

Script para contar slow queries:

```
#!/bin/bash
# /usr/local/bin/mysql_slow_count.sh

SLOW_LOG="/var/log/mysql/slow.log"
MINUTES=${1:-5}

# Contar queries lentas nos últimos N minutos
find $SLOW_LOG -mmin -$MINUTES -exec wc -l {} \; 2>/dev/null | awk '{sum+=$1} END {print sum}'
```

Slow Queries - PostgreSQL

Habilitar log de queries lentas:

```
-- postgresql.conf
ALTER SYSTEM SET log_min_duration_statement = 2000;  -- 2 segundos
SELECT pg_reload_conf();
```

Usar pg_stat_statements (recomendado):

```
-- Habilitar extensão
CREATE EXTENSION pg_stat_statements;

-- Query para slow queries
SELECT query, calls, mean_exec_time, max_exec_time
FROM pg_stat_statements
WHERE mean_exec_time > 2000
ORDER BY mean_exec_time DESC
LIMIT 10;
```

Monitorar Disk I/O - MySQL

```
#!/bin/bash
# /usr/local/bin/mysql_io.sh

case $1 in
    innodb_data_reads)
        mysql -e "SHOW STATUS LIKE 'Innodb_data_reads'" -sN | awk '{print $2}'
        ;;
    innodb_data_writes)
        mysql -e "SHOW STATUS LIKE 'Innodb_data_writes'" -sN | awk '{print $2}'
        ;;
    innodb_data_read)
        mysql -e "SHOW STATUS LIKE 'Innodb_data_read'" -sN | awk '{print $2}'
        ;;
    innodb_data_written)
        mysql -e "SHOW STATUS LIKE 'Innodb_data_written'" -sN | awk '{print $2}'
        ;;
esac
```

Dependent items com "Change per second" para obter IOPS

Table Locks - MySQL

Detectar bloqueios de tabela:

```
-- Mostrar bloqueios ativos
SELECT * FROM information_schema.INNODB_LOCKS;

-- Mostrar transações aguardando locks
SELECT * FROM information_schema.INNODB_LOCK_WAITS;
```

UserParameter:

```
UserParameter=mysql.innodb_locks,mysql -e \
"SELECT COUNT(*) FROM information_schema.INNODB_LOCKS" -sN
```

Trigger:

```
Name: MySQL has table locks
Expression: last(/host/mysql.innodb_locks)>0
```

Deadlocks - PostgreSQL

```
-- Query para detectar deadlocks
SELECT
    blocked_locks.pid AS blocked_pid,
    blocking_locks.pid AS blocking_pid,
    blocked_activity.query AS blocked_statement
FROM pg_catalog.pg_locks blocked_locks
JOIN pg_catalog.pg_stat_activity blocked_activity ON blocked_activity.pid = blocked_locks.pid
JOIN pg_catalog.pg_locks blocking_locks ON blocking_locks.locktype = blocked_locks.locktype
JOIN pg_catalog.pg_stat_activity blocking_activity ON blocking_activity.pid = blocking_locks.pid
WHERE NOT blocked_locks.granted AND blocking_locks.granted;
```

Tamanho de Banco de Dados

MySQL:

```
SELECT
    table_schema AS database_name,
    ROUND(SUM(data_length + index_length) / 1024 / 1024, 2) AS size_mb
FROM information_schema.TABLES
GROUP BY table_schema;
```

PostgreSQL:

```
SELECT
    datname AS database_name,
    pg_size_pretty(pg_database_size(datname)) AS size
FROM pg_database
ORDER BY pg_database_size(datname) DESC;
```

PARTE 5

Triggers para Degradação

Trigger: Conexões Esgotando

MySQL:

```
Name: MySQL connection usage critical
Expression:
  (last(/host/mysql.threads_connected) /
   last(/host/mysql.max_connections)) > 0.9
Severity: High
Description: MySQL usando > 90% das conexões disponíveis
```

Recovery expression:

```
(last(/host/mysql.threads_connected) /
 last(/host/mysql.max_connections)) < 0.7
```

Trigger: Performance Degradada

Slow Queries aumentando:

```
Name: MySQL slow queries increasing
Expression:
  (last(/host/mysql.slow_queries) - last(/host/mysql.slow_queries,#2)) > 100
Severity: Warning
Description: > 100 slow queries desde última coleta
```

Cache Hit Ratio baixo:

```
Name: MySQL buffer pool hit ratio low
Expression:
  avg(/host/mysql.buffer.pool.hit.ratio,10m) < 95
Severity: Warning
Description: Hit ratio < 95% por 10 minutos - cache insuficiente
```

Trigger: Disk I/O Alto

MySQL IOPS:

```
Name: MySQL disk reads too high
Expression: avg(/host/mysql.innodb_data_reads.rate,5m)>1000
Severity: Warning
Description: > 1000 leituras de disco/segundo - cache insuficiente
```

PostgreSQL:

```
Name: PostgreSQL disk blocks read too high
Expression:
  change(/host/pgsql.blks_read[postgres]) > 10000
Severity: Warning
Description: > 10k blocos lidos de disco desde última coleta
```

Trigger: Deadlocks

PostgreSQL:

```
# Script para contar deadlocks
psql -U zbx_monitor -d postgres -t -c \
"SELECT deadlocks FROM pg_stat_database WHERE datname='postgres'"
```

```
Name: PostgreSQL deadlocks detected
Expression: change(/host/pgsql.deadlocks)>0
Severity: Average
Description: Deadlocks detectados no banco de dados
```

PARTE 6

Laboratórios Práticos

Lab 1: Monitorar MySQL Local com Agent 2

Objetivo: Configurar monitoramento do MySQL usando template oficial

1. Criar usuário no MySQL:

```
mysql -uroot
```

```
-- Dropar se já existir
DROP USER IF EXISTS 'zbx_monitor'@'%';

-- Criar usuário (conforme doc oficial)
CREATE USER 'zbx_monitor'@'%' IDENTIFIED BY '4linux';

-- Conceder permissões (documentação oficial v7.0 LTS)
GRANT REPLICATION CLIENT,PROCESS,SHOW DATABASES,SHOW VIEW ON *.* TO 'zbx_monitor'@'%';

FLUSH PRIVILEGES;
EXIT;
```

2. Testar conexão:

Lab 1: Configurar Zabbix Agent 2

3. Editar configuração do Agent 2:

```
sudo nano /etc/zabbix/zabbix_agent2.conf
```

4. Adicionar plugin MySQL (no final do arquivo):

```
### MySQL Plugin ###
Plugins.Mysql.Uri=tcp://zbx_monitor:4linux@localhost:3306
Plugins.Mysql.KeepAlive=300
```

5. Reiniciar Agent 2:

```
sudo systemctl restart zabbix-agent2
sudo systemctl status zabbix-agent2
```

Lab 1: Testar Plugin MySQL

6. Testar plugin:

```
zabbix_agent2 -t mysql.ping
zabbix_agent2 -t mysql.version
zabbix_agent2 -t mysql.status[Threads_connected]
```

Saída esperada:

```
mysql.ping [s|1]
mysql.version [s|8.0.35-0ubuntu0.22.04.1]
mysql.status[Threads_connected] [u|10]
```

✅ Se tudo OK, prosseguir para aplicar o template!

Lab 1: Aplicar Template no Frontend

7. No Zabbix Frontend:

```
1. Configuration → Hosts → Zabbix server

2. Templates → Select → Buscar: "MySQL by Zabbix agent 2"

3. Add → Update
```

8. Aguardar 2 minutos e verificar:

```
Monitoring → Latest data
Host: Zabbix server
Application: MySQL
```

✅ Você verá 50+ métricas coletadas automaticamente!

Lab 2: Visualizar Métricas do Template MySQL

Objetivo: Explorar as métricas coletadas automaticamente

1. Ver todos os items coletados:

```
Monitoring → Latest data
Host: Zabbix server
Tags: component:health, component:network
```

2. Métricas importantes para observar:

- **MySQL: Service status** - Status do serviço (0=down, 1=up)
- **MySQL: Threads connected** - Conexões ativas
- **MySQL: Connection usage** - % de conexões em uso
- **MySQL: Queries per second** - Taxa de queries
- **MySQL: Slow queries rate** - Queries lentas/segundo
- **MySQL: InnoDB buffer pool hit rate** - Cache hit ratio

3. Verificar triggers ativas:

```
Monitoring → Problems
Filter: MySQL
```

Lab 2: Criar Dashboard MySQL

4. Criar dashboard personalizado:

```
Dashboards → Create dashboard
Name: MySQL Monitoring
```

5. Adicionar widgets:

```
Widget 1: Graph
  Name: MySQL Connections
  Items: MySQL: Threads connected

Widget 2: Gauge
  Name: Connection Usage
  Item: MySQL: Connection usage
  Thresholds: 0-70 (green), 70-90 (yellow), 90-100 (red)

Widget 3: Graph (Stacked)
  Name: MySQL QPS
```

Lab 3: PostgreSQL da Coffee Shop com Agent 2

Objetivo: Monitorar PostgreSQL da aplicação Coffee Shop usando template

Banco: PostgreSQL do container app (172.16.1.111)

Database: ecommerce

Usuário: zabbix / Zbx2024!

1. Testar conexão ao PostgreSQL:

```
# No servidor Zabbix
psql -h 172.16.1.111 -U zabbix -d ecommerce -c "SELECT version()"
# Digite senha: Zbx2024!
```

Lab 3: Configurar Agent 2 para PostgreSQL

2. Editar configuração do Agent 2:

```
sudo nano /etc/zabbix/zabbix_agent2.conf
```

3. Adicionar plugin PostgreSQL:

```
### PostgreSQL Plugin ###
Plugins.Postgres.Uri=tcp://zabbix:Zbx2024!@172.16.1.111:5432/ecommerce
Plugins.Postgres.KeepAlive=300
```

4. Reiniciar Agent 2:

```
sudo systemctl restart zabbix-agent2
```

Lab 3: Testar e Aplicar Template

5. Testar plugin:

```
zabbix_agent2 -t pgsql.ping
zabbix_agent2 -t pgsql.db.discovery
zabbix_agent2 -t pgsql.connections
```

6. Aplicar template no Frontend:

```
Configuration → Hosts → [Host da aplicação]
Templates → Select → "PostgreSQL by Zabbix agent 2"
Add → Update
```

7. Verificar coleta:

```
Monitoring → Latest data
Host: [Host da aplicação]
Filter: PostgreSQL
```

PARTE 7

Troubleshooting e Best Practices

Problema 1: UserParameter Não Retorna Dados

Sintoma: Item "Not supported"

Checklist:

1. Script executável?

```
ls -l /usr/local/bin/mysql_stats.sh
# Deve ter permissão +x
```

2. Usuário zabbix pode executar?

```
sudo -u zabbix /usr/local/bin/mysql_stats.sh threads_connected
# Deve retornar número
```

3. Credenciais corretas?

```
sudo -u zabbix mysql -e "SHOW STATUS"
```

Problema 1: Continuação

4. .my.cnf tem permissões corretas?

```
ls -l /var/lib/zabbix/.my.cnf
# Deve ser 600 e owner zabbix:zabbix
```

5. UserParameter configurado?

```
grep mysql.threads_connected /etc/zabbix/zabbix_agentd.d/*.conf
```

6. Agent reiniciado?

```
sudo systemctl restart zabbix-agent
```

Problema 2: Permissões de BD Insuficientes

Sintoma: Erro "Access denied"

MySQL:

```
SHOW GRANTS FOR 'zbx_monitor'@'%';

-- Deve mostrar:
-- GRANT REPLICATION CLIENT,PROCESS,SHOW DATABASES,SHOW VIEW ON *.* TO 'zbx_monitor'@'%'
```

PostgreSQL:

```
\du zbx_monitor

-- Deve mostrar:
-- Roles: pg_monitor
```

Solução: Re-executar GRANTs dos slides iniciais

Problema 3: Slow Queries Não Aparecem

MySQL:

```
SHOW VARIABLES LIKE 'slow_query_log';
-- Deve estar ON

SHOW VARIABLES LIKE 'long_query_time';
-- Deve ser <= tempo das queries que quer capturar
```

PostgreSQL:

```
SHOW log_min_duration_statement;
-- Deve ser > 0 (em millisegundos)
```

Logs estão sendo escritos?

```
sudo tail -f /var/log/mysql/slow.log
sudo tail -f /var/log/postgresql/postgresql-*.log
```

Problema 4: Replicação Lag Sempre Alto

Causas comuns:

1. Network latency entre master e slave

```
# No slave
ping master-ip
# Latência deve ser < 10ms idealmente
```

2. Slave hardware inferior ao master

```
# Comparar CPU/RAM/Disk
top
iostat -x 1
```

3. Queries pesadas no slave

```
-- MySQL
```

Problema 4: Solução

4. Ajustar paralelismo (MySQL):

```
-- MySQL 8.0+
SET GLOBAL slave_parallel_workers = 4;
SET GLOBAL slave_parallel_type = 'LOGICAL_CLOCK';
```

PostgreSQL:

```
-- postgresql.conf
max_wal_senders = 10
wal_keep_size = 1GB
```

Best Practice 1: Monitorar Múltiplos Databases

Usar LLD (Low-Level Discovery):

```
#!/bin/bash
# /usr/local/bin/mysql_discovery.sh

mysql -e "SHOW DATABASES" -sN | \
grep -v -E "^(information_schema|performance_schema|mysql|sys)$" | \
awk '{print "{\"{#DBNAME}\":\""$1"\"}"}'  | \
paste -sd, | sed 's/.*/{"data":[\0]}/'
```

UserParameter:

```
UserParameter=mysql.db.discovery,/usr/local/bin/mysql_discovery.sh
```

Discovery rule no Zabbix: Criar item prototypes para cada database

Best Practice 2: Alertas Inteligentes

❌ Ruim:

```
Expression: last(/host/mysql.threads_connected)>100
```

✅ Bom:

```
Expression:
  (last(/host/mysql.threads_connected) /
   last(/host/mysql.max_connections)) > 0.8
  and
  min(/host/mysql.threads_connected,5m) > 50
```

Por quê?

- Usa % em vez de valor absoluto
- Confirma problema por 5 minutos (evita spikes)

Best Practice 3: Monitorar Crescimento de BD

```
#!/bin/bash
# /usr/local/bin/mysql_db_size.sh

DB=$1

mysql -e "SELECT SUM(data_length + index_length)
FROM information_schema.TABLES
WHERE table_schema='$DB'" -sN
```

Item com preprocessing:

```
Type: Zabbix agent
Key: mysql.db.size[myapp]
Preprocessing:
  1. Simple change  (armazena apenas quando muda)
Type: Numeric (unsigned)
Units: B
```

Best Practice 4: Backup Validation

Monitorar se backups estão acontecendo:

```
#!/bin/bash
# /usr/local/bin/check_last_backup.sh

BACKUP_DIR="/var/backups/mysql"
LATEST=$(find $BACKUP_DIR -name "*.sql.gz" -mtime -1 | wc -l)

if [ $LATEST -gt 0 ]; then
    echo "1"  # Backup OK (últimas 24h)
else
    echo "0"  # Backup velho
fi
```

Trigger:

```
Name: MySQL backup is outdated
Expression: last(/host/mysql.backup.check)=0
```

Best Practice 5: Documentar Tudo

Template com Description:

```
Template: MySQL Monitoring

Description:
  - UserParameters em /etc/zabbix/zabbix_agentd.d/mysql.conf
  - Script em /usr/local/bin/mysql_stats.sh
  - Usuário BD: zbx_monitor (senha em /var/lib/zabbix/.my.cnf)
  - Requer: PROCESS, REPLICATION CLIENT grants

Macros:
  {$MYSQL_MAX_CONN_USAGE} = 80  (%)
  {$MYSQL_SLOW_QUERY_TIME} = 2  (segundos)
  {$MYSQL_REPL_LAG_WARN} = 60   (segundos)
```

Recursos Adicionais

Documentação oficial:

- Zabbix Templates
- MySQL Performance Schema
- PostgreSQL Statistics Views

Templates da comunidade:

- Percona Monitoring Templates
- PostgreSQL Zabbix Templates

Revisão da Aula

Aprendemos:

1. ✅ Fundamentos do monitoramento de BD
2. ✅ **Templates oficiais:** MySQL by Zabbix agent 2
3. ✅ **Templates oficiais:** PostgreSQL by Zabbix agent 2
4. ✅ Configuração de plugins nativos no Agent 2
5. ✅ Métricas automáticas: conexões, cache, I/O, slow queries, replicação
6. ✅ Triggers e dashboards prontos dos templates
7. ✅ 3 laboratórios práticos com Agent 2
8. ✅ Troubleshooting e best practices

🎯 **Principais vantagens dos templates oficiais:**

- Setup rápido (4 passos)
- 50+ métricas automáticas
- Triggers prontas e testadas
- Fácil manutenção

Próxima Aula

Aula 12: Performance do Zabbix e Backup

Tópicos:

- Otimização do Zabbix Server (frontend, banco, pollers)
- Identificação de gargalos
- Monitoramento do próprio Zabbix (housekeeper, queue)
- Estratégias de backup e restore
- Segurança: TLS, autenticação, hardening

Perguntas?

Obrigado!

4Linux - Zabbix Advanced Course