Zabbix Advanced

Aula 12: Performance e Backup

Otimização e Disaster Recovery

4Linux - Curso Avançado

Objetivos da Aula

1. **Performance do Zabbix**
    - Identificar gargalos e otimizar componentes
    - Dimensionar hardware adequadamente
    - Monitorar o próprio Zabbix (self-monitoring)
2. **Estratégias de Backup**
    - Backup de banco de dados e configurações
    - Disaster Recovery e testes de restore

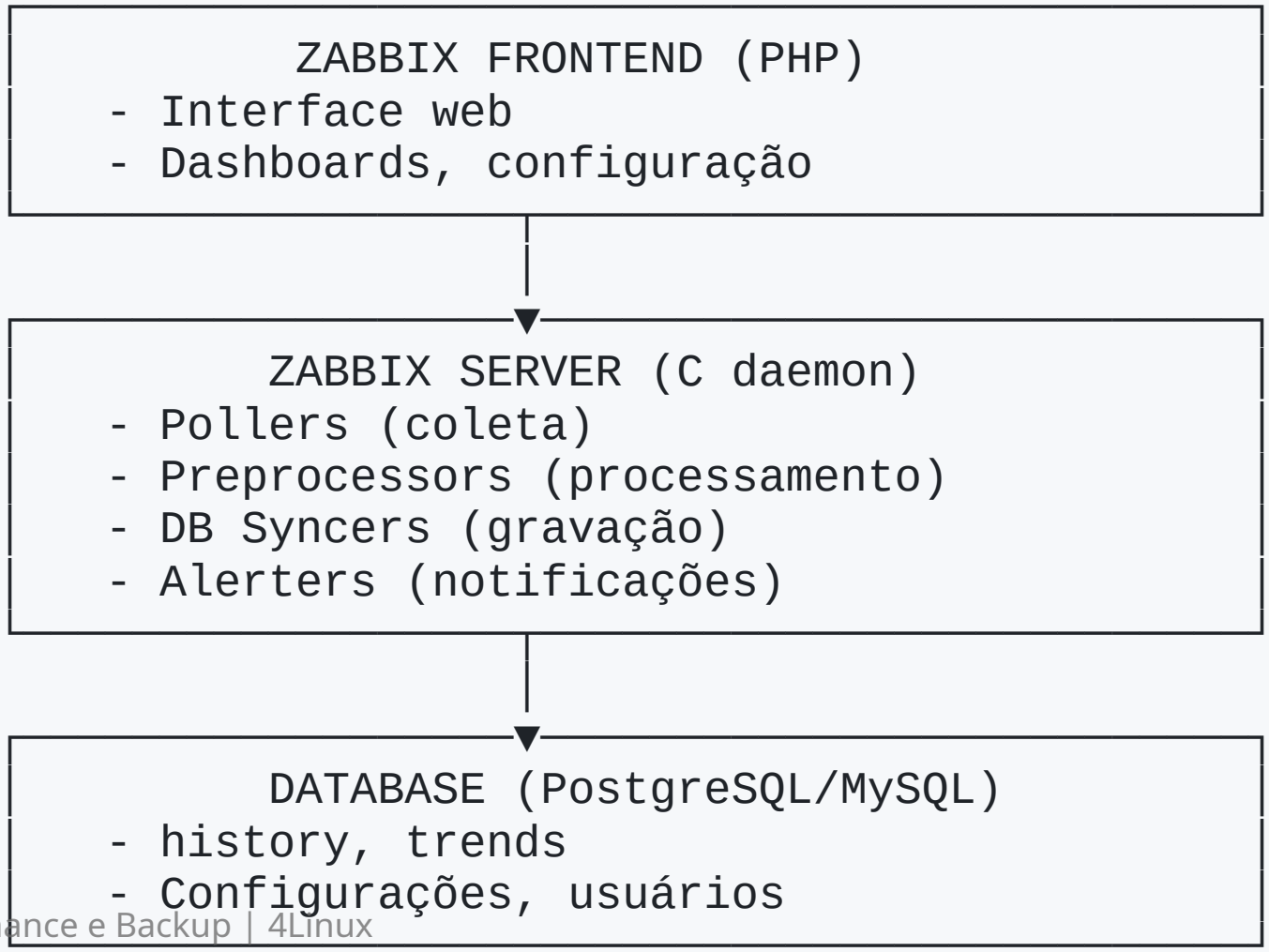
Agenda do Dia

Parte 1: Fundamentos de Performance
Parte 2: Otimização do Zabbix Server
Parte 3: Otimização do Banco de Dados
Parte 4: Backup e Disaster Recovery
Parte 5: Laboratórios Práticos

PARTE 1

Fundamentos de Performance do Zabbix

Arquitetura do Zabbix - Componentes



Sintomas de Problemas de Performance

1. Coleta lenta ou atrasada

- Sintoma: Métricas desatualizadas, gaps em gráficos
- Causa: Pollers insuficientes, timeouts, agents sobrecarregados

2. Interface lenta

- Sintoma: Dashboards demoram para carregar
- Causa: Queries SQL complexas, PHP mal configurado

3. Banco de dados sobrecarregado

- Sintoma: CPU alta no servidor de BD, queries lentas
- Causa: Falta de índices, tabelas gigantes, housekeeper ineficiente

Sintomas de Problemas (cont.)

4. Fila de dados crescente

- Sintoma: Zabbix queue aumentando constantemente
- Causa: History syncers insuficientes, disco lento

5. Alto uso de memória

- Sintoma: Zabbix Server consumindo muita RAM, OOM kills
- Causa: Cache mal dimensionado, muitas conexões abertas

Métricas de Performance do Zabbix

Internal Monitoring - O Zabbix monitora a si mesmo:

Item	Descrição	Valor Ideal
`zabbix[queue]`	Total de itens aguardando coleta	< 100
`zabbix[queue,10m]`	Itens atrasados > 10 min	0
`zabbix[wcache,values]`	Valores no cache de escrita	< 75%
`zabbix[preprocessing_queue]`	Fila de preprocessamento	< 1000
`zabbix[process,poller,avg,busy]`	% ocupado dos pollers	< 75%
`zabbix[process,history syncer,avg,busy]`	% ocupado syncers	< 75%

Dimensionamento de Hardware

NVPS (New Values Per Second) = Métrica chave

```
NVPS = Total de Items / Intervalo Médio de Coleta

Exemplo:
- 10.000 items
- Intervalo médio: 60 segundos
- NVPS = 10.000 / 60 = 166 NVPS
```

Dimensionamento: Ambiente Pequeno

Até 1.000 NVPS:

- **CPU:** 2 cores
- **RAM:** 8 GB
- **Disco:** 50 GB SSD
- **Hosts:** ~100 hosts
- **Items:** ~10.000 items
- **Database:** MySQL/PostgreSQL (mesmo servidor)

Exemplo: Pequena empresa, monitoramento básico

Dimensionamento: Ambiente Médio

1.000 - 10.000 NVPS:

- **CPU:** 4 cores
- **RAM:** 16 GB
- **Disco:** 200 GB SSD
- **Hosts:** ~500 hosts
- **Items:** ~50.000 items
- **Database:** Servidor separado (8GB RAM, 4 cores)

Exemplo: Empresa média, monitoramento completo

Dimensionamento: Ambiente Grande

10.000 - 100.000 NVPS:

- **CPU:** 16 cores
- **RAM:** 64 GB
- **Disco:** 500 GB NVMe SSD
- **Hosts:** ~5.000 hosts
- **Items:** ~500.000 items
- **Database:** Servidor dedicado (32GB RAM, 8 cores)
- **Particionamento:** Recomendado

Exemplo: Grande empresa, infraestrutura complexa

Dimensionamento: Ambiente Muito Grande

> 100.000 NVPS:

- **CPU:** 32+ cores
- **RAM:** 96+ GB
- **Disco:** 1+ TB NVMe SSD em RAID
- **Hosts:** ~50.000+ hosts
- **Items:** ~5.000.000+ items
- **Database:** Cluster de banco de dados
- **Zabbix Proxy:** Distribuir carga
- **Particionamento:** Obrigatório

PARTE 2

Otimização do Zabbix Server

Parâmetros Críticos: zabbix_server.conf

Arquivo: `/etc/zabbix/zabbix_server.conf`

Grupos de parâmetros:

1. **Pollers e Coletores** - Coleta de dados
2. **Preprocessamento** - Processamento de dados
3. **History Syncers** - Gravação no banco
4. **Cache de Memória** - Performance de leitura/escrita
5. **Timeouts** - Reliability vs Performance

StartPollers - Coleta de Dados

Número de processos para coletar dados ativos

```
# Padrão (muito baixo!)
StartPollers=5

# Recomendado para ambiente médio
StartPollers=20

# Recomendado para ambiente grande
StartPollers=50
```

Como dimensionar:

- Verificar `zabbix[process,poller,avg,busy]`
- Se > 75%, aumentar StartPollers
- Cada poller consome ~10MB RAM

StartPollersUnreachable

Pollers dedicados para hosts inacessíveis

```
# Padrão
StartPollersUnreachable=1

# Recomendado (evita travar pollers normais)
StartPollersUnreachable=5
```

Por que importante?

- Hosts unreachable causam timeouts longos
- Sem pollers dedicados, travam pollers normais
- Coleta de hosts OK fica lenta

StartTrappers e StartPingers

StartTrappers - Recebe dados de agents ativos

```
# Padrão
StartTrappers=5

# Recomendado para muitos agents ativos
StartTrappers=10
```

StartPingers - ICMP pings

```
# Padrão
StartPingers=1

# Recomendado se muito ICMP
StartPingers=5
```

StartPreprocessors - Processamento

Processos para preprocessamento (JavaScript, regex, etc.)

```
# Padrão
StartPreprocessors=3

# Recomendado para ambiente médio
StartPreprocessors=10

# Recomendado para ambiente grande com muito JS
StartPreprocessors=20
```

Sintoma de insuficiência:

- `zabbix[preprocessing_queue]` > 1000
- Gaps em gráficos de itens com preprocessing

StartDBSyncers - Gravação no Banco

Processos que gravam dados no banco (history, trends)

```
# Padrão
StartDBSyncers=4

# Recomendado para ambiente médio
StartDBSyncers=8

# Recomendado para ambiente grande
StartDBSyncers=16
```

Como dimensionar:

- Verificar `zabbix[wcache,values]`
- Se constantemente > 75%, aumentar StartDBSyncers
- Também otimizar banco de dados (índices, IOPS)

Cache de Memória - CacheSize

Cache para histórico de dados antes de gravar no BD

```
# Padrão (muito pequeno!)
CacheSize=8M

# Recomendado para ambiente médio
CacheSize=128M

# Recomendado para ambiente grande
CacheSize=512M

# Ambiente muito grande
CacheSize=2G
```

⚠️ **IMPORTANTE:** Cache pequeno = dados perdidos em picos!

Cache de Memória - ValueCacheSize

Cache para valores recentes (usado em triggers, gráficos)

```
# Padrão
ValueCacheSize=8M

# Recomendado para ambiente médio
ValueCacheSize=128M

# Recomendado para ambiente grande
ValueCacheSize=512M
```

Benefício:

- Triggers avaliam mais rápido
- Dashboards carregam mais rápido
- Menos queries ao banco

Cache de Memória - HistoryCacheSize

Cache para histórico antes de processar

```
# Padrão
HistoryCacheSize=16M

# Recomendado para ambiente médio
HistoryCacheSize=256M

# Recomendado para ambiente grande
HistoryCacheSize=1G
```

Timeout - Balanceamento

Timeout global para coleta de dados

```
# Padrão (muito baixo!)
Timeout=3

# Recomendado (balance entre reliability e performance)
Timeout=10
```

Problema comum:

❌ **Timeout muito baixo (3s):** Itens de rede lenta falham constantemente

❌ **Timeout muito alto (30s):** Pollers ficam travados aguardando resposta

✅ **Solução:** ~10s é um bom balanço

Monitorando o Zabbix Server

✅ Template nativo "Zabbix server" (Zabbix 7.0):

O Zabbix 7.0 já inclui template completo com 60+ items internos!

Principais items já incluídos:

```
✅ zabbix[queue] - Total de items na fila
✅ zabbix[queue,10m] - Items atrasados > 10 min
✅ zabbix[wcache,values] - Valores no cache de escrita
✅ zabbix[process,poller,avg,busy] - % ocupado dos pollers
✅ zabbix[process,history syncer,avg,busy] - % ocupado syncers
✅ zabbix[boottime] - Startup time
✅ zabbix[uptime] - Uptime
... e muitos outros!
```

🎯 **Recomendação:** Use o template nativo, não crie do zero!

Triggers Incluídas no Template Nativo

O template "Zabbix server" já inclui triggers prontas:

✅ **Zabbix server: High queue size (> 100)** (Warning)
✅ **Zabbix server: Delayed items (> 10 min)** (High)
✅ **Zabbix server: Write cache usage is high (> 75%)** (Warning)
✅ **Zabbix server: Pollers are busy (> 75%)** (Warning)
✅ **Zabbix server: History syncers are busy (> 75%)** (Warning)
✅ **Zabbix server: Preprocessors are busy (> 75%)** (Warning)

Customizar triggers:

- Ajuste thresholds nas **macros do template** conforme sua carga
- Exemplo: `{$ZABBIX.SERVER.QUEUE.MAX.WARN}` = 100 (padrão) → 500 (ambiente grande)

PARTE 3

Otimização do Banco de Dados

PostgreSQL: Configurações Essenciais

Arquivo: `/etc/postgresql/13/main/postgresql.conf`

4 áreas críticas:

1. **Memória** - shared_buffers, effective_cache_size
2. **Write-Ahead Log (WAL)** - Checkpoints, wal_size
3. **Conexões** - max_connections
4. **Autovacuum** - Limpeza automática

PostgreSQL: Memória - shared_buffers

Cache de dados em memória (MAIS IMPORTANTE!)

```
# Padrão (muito pequeno!)
shared_buffers = 128MB

# Recomendado: 25% da RAM total
# Para servidor com 16GB RAM:
shared_buffers = 4GB

# Para servidor com 64GB RAM:
shared_buffers = 16GB
```

Regra: 25% da RAM do servidor dedicado

PostgreSQL: effective_cache_size

Estimativa de memória disponível para cache do OS

```
# Padrão
effective_cache_size = 4GB

# Recomendado: 50-75% da RAM total
# Para servidor com 16GB RAM:
effective_cache_size = 12GB

# Para servidor com 64GB RAM:
effective_cache_size = 48GB
```

⚠️ **Não aloca memória!** Apenas informa ao PostgreSQL quanto pode usar

PostgreSQL: work_mem e maintenance_work_mem

work_mem - Memória para operações de sort e hash

```
# Padrão
work_mem = 4MB

# Recomendado para ambiente médio
work_mem = 32MB

# Recomendado para ambiente grande
work_mem = 64MB
```

maintenance_work_mem - VACUUM, CREATE INDEX

```
maintenance_work_mem = 1GB
```

PostgreSQL: WAL (Write-Ahead Log)

Checkpoints - Controle de escrita no disco

```
# Intervalo entre checkpoints
checkpoint_timeout = 15min  # Padrão: 5min

# Distribuição do I/O do checkpoint
checkpoint_completion_target = 0.9  # Padrão: 0.5

# Tamanho máximo do WAL antes de checkpoint
max_wal_size = 4GB  # Padrão: 1GB
```

Benefício: Menos I/O espasmódico, performance mais estável

PostgreSQL: Autovacuum (CRÍTICO!)

Limpeza automática de dados mortos

```
# SEMPRE manter ON para Zabbix!
autovacuum = on

# Número de workers
autovacuum_max_workers = 6  # Padrão: 3

# Intervalo entre execuções
autovacuum_naptime = 30s  # Padrão: 1min
```

⚠️ NUNCA desabilitar autovacuum no Zabbix!

- Tabelas history/trends crescem exponencialmente
- Performance degrada rapidamente
- Banco pode ficar inutilizável

MySQL: InnoDB Buffer Pool

Cache de dados InnoDB (PARÂMETRO MAIS IMPORTANTE!)

```
# Padrão (muito pequeno!)
innodb_buffer_pool_size = 128M

# Recomendado: 70-80% da RAM (servidor dedicado)
# Para servidor com 16GB RAM:
innodb_buffer_pool_size = 12G

# Para servidor com 64GB RAM:
innodb_buffer_pool_size = 48G
```

Este é O parâmetro que mais impacta performance no MySQL!

MySQL: InnoDB Buffer Pool Instances

Divisão do buffer pool (melhor concorrência)

```
# Padrão
innodb_buffer_pool_instances = 1

# Recomendado (1 instance por GB, max 64)
# Para 12GB buffer pool:
innodb_buffer_pool_instances = 12

# Para 48GB buffer pool:
innodb_buffer_pool_instances = 48
```

Benefício: Reduz contenção em ambientes com muitos threads

MySQL: InnoDB Log e Flush

innodb_log_file_size - Tamanho do log de transações

```
innodb_log_file_size = 512M  # Padrão: 48M
```

innodb_flush_log_at_trx_commit - Durabilidade vs Performance

```
# Padrão (máxima durabilidade) - LENTO
innodb_flush_log_at_trx_commit = 1

# Recomendado para Zabbix (melhor performance, risco mínimo)
innodb_flush_log_at_trx_commit = 2
```

Valores:

- `1` = Flush a cada commit (LENTO, máxima durabilidade)
- `2` = Flush para OS cache (RÁPIDO, perda < 1s em crash)

Particionamento de Tabelas

Por que particionar?

Tabelas `history*` e `trends*` crescem exponencialmente.

Sem particionamento:

❌ DELETE de dados antigos demora horas
❌ Queries lentas em tabelas gigantes (> 100GB)
❌ VACUUM/OPTIMIZE trava o banco

Com particionamento:

✅ DROP de partição antiga = instantâneo
✅ Queries filtradas por data = muito mais rápidas
✅ Manutenção sem impacto

Particionamento: PostgreSQL

Zabbix 6.0+ já cria tabelas particionadas por padrão!

Verificar partições:

```
-- Conectar ao PostgreSQL
sudo -u postgres psql zabbix

-- Verificar partições
SELECT
    schemaname,
    tablename,
    pg_size_pretty(pg_total_relation_size(schemaname||'.'||tablename)) AS size
FROM pg_tables
WHERE tablename LIKE 'history%'
ORDER BY pg_total_relation_size(schemaname||'.'||tablename) DESC
LIMIT 20;
```

Resultado Esperado - Partições PostgreSQL

```
 schemaname |       tablename          |  size
------------+--------------------------+---------
 public     | history_p2025_10         | 4523 MB
 public     | history_p2025_09         | 4102 MB
 public     | history_uint_p2025_10    | 3890 MB
 public     | history_uint_p2025_09    | 3654 MB
 public     | trends_p2025_10          | 890 MB
 ...
```

✅ **Particionado por mês automaticamente!**

Particionamento: MySQL

MySQL requer configuração manual

```
-- ATENÇÃO: Backup antes de executar!

ALTER TABLE history PARTITION BY RANGE (clock) (
    PARTITION p2025_01 VALUES LESS THAN (UNIX_TIMESTAMP('2025-02-01 00:00:00')),
    PARTITION p2025_02 VALUES LESS THAN (UNIX_TIMESTAMP('2025-03-01 00:00:00')),
    PARTITION p2025_03 VALUES LESS THAN (UNIX_TIMESTAMP('2025-04-01 00:00:00')),
    -- ... continuar para meses futuros
    PARTITION pmax VALUES LESS THAN MAXVALUE
);
```

Gerenciamento de Partições MySQL

Mensal - Dropar partições antigas e criar novas:

```
-- Dropar partição antiga (dados > 90 dias)
ALTER TABLE history DROP PARTITION p2025_01;

-- Adicionar nova partição (próximo mês)
ALTER TABLE history ADD PARTITION (
    PARTITION p2025_04 VALUES LESS THAN (UNIX_TIMESTAMP('2025-05-01 00:00:00'))
);
```

⚠️ Automatizar com cron mensal!

PARTE 4

Backup e Disaster Recovery

O Que Precisa Ser Feito Backup?

Ordem de prioridade:

1. **BANCO DE DADOS** ⭐⭐⭐ (CRÍTICO!)
   - PostgreSQL/MySQL: Configurações, histórico, trends
   - Tamanho: Pode ser MUITO grande (100GB+)
2. **Arquivos de Configuração** ⭐⭐
   - `/etc/zabbix/` - zabbix_server.conf, zabbix_agentd.conf
   - Tamanho: < 1MB

3. **Scripts Customizados** ⭐
    - `/usr/lib/zabbix/externalscripts/`, `/usr/lib/zabbix/alertscripts/`
4. **Frontend** (Opcional - pode reinstalar)

Backup PostgreSQL: Full Backup

Script de backup completo:

```
#!/bin/bash
# Script: backup_zabbix_postgresql.sh

BACKUP_DIR="/backup/zabbix"
DATE=$(date +%Y%m%d_%H%M%S)
BACKUP_FILE="zabbix_backup_${DATE}.sql.gz"

mkdir -p $BACKUP_DIR

# Fazer backup compactado
sudo -u postgres pg_dump zabbix | gzip > $BACKUP_DIR/$BACKUP_FILE

# Verificar se backup foi criado
if [ -f "$BACKUP_DIR/$BACKUP_FILE" ]; then
    echo "Backup criado com sucesso: $BACKUP_FILE"
else
    echo "ERRO: Backup falhou!"
    exit 1
fi
```

Agendar Backup Automático

Tornar executável e agendar no cron:

```
# Copiar script
sudo cp backup_zabbix_postgresql.sh /usr/local/bin/
sudo chmod +x /usr/local/bin/backup_zabbix_postgresql.sh

# Crontab - backup diário às 2h da manhã
sudo crontab -e

# Adicionar linha:
0 2 * * * /usr/local/bin/backup_zabbix_postgresql.sh >> /var/log/zabbix_backup.log 2>&1
```

Resultado: Backup diário automático às 2h AM, mantém 7 dias

Backup Apenas da Configuração (Sem Histórico)

Útil para disaster recovery rápido (perde histórico, mantém configuração)

```
#!/bin/bash
BACKUP_DIR="/backup/zabbix_config"
DATE=$(date +%Y%m%d_%H%M%S)

# Tabelas a excluir (histórico)
EXCLUDE_TABLES="history history_uint history_str history_log history_text trends trends_uint"

EXCLUDE_OPTS=""
for table in $EXCLUDE_TABLES; do
    EXCLUDE_OPTS="$EXCLUDE_OPTS --exclude-table-data=public.$table"
done

# Fazer backup sem histórico
sudo -u postgres pg_dump $EXCLUDE_OPTS zabbix | gzip > $BACKUP_DIR/zabbix_config_${DATE}.sql.gz
```

Tamanho: ~100-500MB vs ~50-500GB (completo)

Restore do Backup PostgreSQL

```
#!/bin/bash
BACKUP_FILE="/backup/zabbix/zabbix_backup_20251108_020000.sql.gz"

# ATENÇÃO: Isso vai SOBRESCREVER o banco atual!
read -p "Tem certeza? Banco atual será PERDIDO! (yes/no): " confirm

if [ "$confirm" != "yes" ]; then
    echo "Restore cancelado."
    exit 1
fi

# Parar Zabbix Server
sudo systemctl stop zabbix-server

# Dropar banco existente e criar novo
sudo -u postgres psql -c "DROP DATABASE zabbix;"
sudo -u postgres psql -c "CREATE DATABASE zabbix OWNER zabbix;"

# Restaurar backup
zcat $BACKUP_FILE | sudo -u postgres psql zabbix

# Iniciar Zabbix Server
sudo systemctl start zabbix-server
```

Backup MySQL

```
#!/bin/bash
BACKUP_DIR="/backup/zabbix"
DATE=$(date +%Y%m%d_%H%M%S)
MYSQL_USER="root"
MYSQL_PASS="senha_root"

# Fazer backup
mysqldump -u$MYSQL_USER -p$MYSQL_PASS \
    --single-transaction \
    --quick \
    --lock-tables=false \
    zabbix | gzip > $BACKUP_DIR/zabbix_backup_${DATE}.sql.gz

# Remover backups antigos
find $BACKUP_DIR -name "zabbix_backup_*.sql.gz" -mtime +7 -delete
```

Opções importantes:

- `--single-transaction` : Backup consistente sem lock (InnoDB)
- `--quick` : Não carrega toda tabela na memória

Backup de Arquivos de Configuração

```
#!/bin/bash
# Script: backup_zabbix_config_files.sh

BACKUP_DIR="/backup/zabbix_config"
DATE=$(date +%Y%m%d_%H%M%S)
BACKUP_FILE="zabbix_config_files_${DATE}.tar.gz"

mkdir -p $BACKUP_DIR

# Fazer backup de todos arquivos de configuração
tar -czf $BACKUP_DIR/$BACKUP_FILE \
    /etc/zabbix/ \
    /usr/lib/zabbix/externalscripts/ \
    /usr/lib/zabbix/alertscripts/ \
    /etc/ssl/zabbix/ \
    2>/dev/null

echo "Backup de configuração criado: $BACKUP_FILE"
```

Estratégia de Backup: Regra 3-2-1

Regra 3-2-1 de Backup:

- **3** cópias dos dados (original + 2 backups)
- **2** tipos de mídia diferentes (disco local + nuvem)
- **1** cópia off-site (fora do datacenter)

Estratégia de Backup Zabbix Completa

DIÁRIO (2h da manhã):

- Backup completo do banco de dados
- Retenção: 7 dias no disco local
- Destino: `/backup/zabbix/`

SEMANAL (Domingo 3h):

- Backup completo do banco + arquivos config
- Retenção: 4 semanas
- Destino: NAS remoto

MENSAL (Dia 1, 4h):

- Backup completo FULL
- Retenção: 12 meses
- Destino: AWS S3 / Azure Blob (cloud)

Teste de Restore (DR Drill)

⚠️ **IMPORTANTE:** Backups sem testes de restore são backups inúteis!

Teste mensal recomendado:

1. Criar ambiente de teste (VM isolada)
2. Fazer restore do backup mais recente
3. Verificar:
    - Zabbix Server inicia corretamente?
    - Frontend acessível?
    - Dados de host visíveis?
    - Triggers funcionando?
    - Usuários conseguem logar?

4. Documentar tempo de restore
5. Documentar problemas encontrados

PARTE 5

Laboratórios Práticos

Laboratório 1: Explorar Template Nativo "Zabbix Server"

Objetivo: Conhecer e customizar o template nativo de monitoramento do Zabbix

⚠️ **IMPORTANTE:** O Zabbix 7.0 já inclui template nativo "Zabbix server" com 60+ items!

Tarefas:

1. Verificar se template "Zabbix server" está aplicado no host "Zabbix server"
2. Explorar items já coletados:
    - `zabbix[queue]` - Fila de coleta
    - `zabbix[wcache,values]` - Cache de escrita
    - `zabbix[process,poller,avg,busy]` - Ocupação dos pollers

3. Criar **dashboard customizado** com métricas críticas
4. Ajustar thresholds de triggers conforme seu ambiente

Lab 1: Passo a Passo

1. Verificar template aplicado:

```
Configuration → Hosts → Zabbix server
Aba "Templates"
```

✅ Deve ver: "Zabbix server" template já linkado

2. Explorar items coletados:

```
Monitoring → Latest data
Host: Zabbix server
Tags: component:zabbix
```

Você verá 60+ items já coletando automaticamente!

Lab 1: Criar Dashboard Customizado

3. Criar dashboard de performance:

```
Dashboards → Create dashboard
Name: Zabbix Server Performance
```

Adicionar 4 widgets:

Widget 1: Graph - Queue Size

- Item: `zabbix[queue]`
- Alert se > 100

Widget 2: Gauge - Write Cache Usage

- Item: `zabbix[wcache,values]`
- Thresholds: 0-75 (green), 75-90 (yellow), 90-100 (red)

Lab 1: Dashboard (continuação)

Widget 3: Graph (Stacked) - Process Busy %

- Items:
  - `zabbix[process,poller,avg,busy]`
  - `zabbix[process,history syncer,avg,busy]`
  - `zabbix[process,preprocessor,avg,busy]`

Widget 4: Plain Text - Queue Delayed

- Item: `zabbix[queue,10m]`
- Alert se > 0

4. Ajustar triggers:

- Modificar thresholds das triggers do template conforme sua carga

Laboratório 2: Otimização do Zabbix Server

Objetivo: Ajustar parâmetros do zabbix_server.conf

Tarefas:

1. Editar `/etc/zabbix/zabbix_server.conf`:
    - Aumentar `StartPollers` de 5 para 20
    - Aumentar `CacheSize` de 8M para 128M
    - Aumentar `ValueCacheSize` de 8M para 128M
2. Reiniciar Zabbix Server
3. Monitorar `zabbix[queue]` - deve reduzir

Laboratório 3: Backup Completo

Objetivo: Criar script de backup automatizado

Tarefas:

1. Criar diretório `/backup/zabbix`
2. Criar script de backup PostgreSQL/MySQL
3. Testar backup manual
4. Agendar backup no cron (diário às 2h AM)

Laboratório 4: Disaster Recovery Drill

Objetivo: Testar restore completo

Cenário: Servidor de produção falhou, precisa restaurar backup

Tarefas:

1. Fazer backup completo do ambiente atual
2. **SIMULAR DESASTRE:** Dropar banco de dados
3. Parar Zabbix Server
4. Restaurar backup
5. Iniciar Zabbix Server
6. Verificar:
    - Frontend acessível?
    - Hosts visíveis?

Checklist Final: Ambiente Otimizado

Performance:

- [x] Self-monitoring configurado
- [x] Parâmetros do zabbix_server.conf otimizados
- [x] Banco de dados otimizado (shared_buffers, buffer_pool)
- [x] Particionamento habilitado
- [x] Frontend com cache habilitado

Backup:

- [x] Script de backup automatizado
- [x] Cron configurado (diário)
- [x] Teste de restore realizado
- [x] Backup off-site configurado

Recursos e Referências

Documentação Oficial:

- Zabbix Performance Tuning: https://www.zabbix.com/documentation/current/en/manual/installation/requirements
- PostgreSQL Tuning: https://pgtune.leopard.in.ua/
- MySQL Tuning: https://dev.mysql.com/doc/refman/8.0/en/optimization.html

Ferramentas:

- pg_stat_statements (PostgreSQL query analysis)
- mysqltuner (MySQL optimization script)
- Zabbix internal items documentation

Comunidade:

- Zabbix Forums: https://www.zabbix.com/forum/
- Zabbix Discord

Boas Práticas: Resumo

Performance:

✅ Dimensionar hardware baseado em NVPS
✅ Monitorar o próprio Zabbix (self-monitoring)
✅ Ajustar pollers conforme necessidade
✅ Otimizar banco de dados (memória, WAL, particionamento)

Backup:

✅ Seguir regra 3-2-1
✅ Automatizar backups diários
✅ Testar restore mensalmente
✅ Backup de configuração separado (rápido recovery)

Perguntas?

Dúvidas sobre Performance e Backup?

Obrigado!

Zabbix Advanced - Aula 12

Performance e Backup