Zabbix Advanced

Aula 02: Criação de Hosts e Métodos de Coleta

4Linux - Curso Avançado

Agenda do Dia

1. **Fundamentos dos Métodos**
    - Visão geral e critérios de escolha
2. **Agente Zabbix**
    - Arquitetura, itens nativos, UserParameters
3. **SNMP**
    - Versões, MIBs, OIDs, laboratório

Agenda do Dia (continuação)

4. **Criação de Hosts**
    - Frontend e API
5. **Discovery e Auto-registro**
    - Network discovery, auto registration
6. **Mapeamento e Macros**
    - Value mapping, context macros

PARTE 1

Fundamentos dos Métodos de Coleta

Recap Aula 01

O que vimos:

- Templates e padronização
- Itens com pré-processamento
- Triggers compostos com histerese
- Dependências hierárquicas
- JavaScript avançado
- Macros e precedência

Hoje: Como coletar os dados!

Métodos Disponíveis no Zabbix

Principais:

- 🤖 Zabbix Agent (Active/Passive)
- 📡 SNMP (v1/v2c/v3)
- 🌐 HTTP Agent (APIs REST)

Outros:

- ☕ JMX (Java applications)
- 🖥️ IPMI (hardware físico)
- 🗄️ Database Monitor
- 📜 External Check
- 🔑 SSH/Telnet Agent

Comparação: Agent vs SNMP vs HTTP

Aspecto	Agent	SNMP	HTTP
Instalação	Requer	Nativo	Não requer
Performance	⚡ Excelente	🔸 Boa	🔸 Boa
Latência	<100ms	100-300ms	200-1000ms
Segurança	PSK/TLS	SNMPv3	HTTPS/OAuth
Firewall	TCP 10050/10051	UDP 161	TCP 80/443
Customização	UserParameters	MIBs limitado	JSONPath/Regex
Escalabilidade	10k+ hosts	5k+ devices	Média

Guia de Decisão Rápida

Cenário	Recomendado	Justificativa
Servidores Linux/Windows	**Agent**	Métricas detalhadas, performance
Switches/Routers/Firewalls	**SNMP**	Protocolo nativo, padrão
Impressoras/UPS/IoT	**SNMP**	Agentless, baixo overhead
APIs REST (cloud/SaaS)	**HTTP**	Único método disponível
Aplicações web custom	**HTTP**	Healthchecks, métricas
Containers Docker	**Agent 2**	Suporte nativo
Kubernetes	**HTTP**	Metrics Server API
Aplicações Java	**JMX**	Métricas JVM

ROI: Escolha Correta

Cenário: 500 dispositivos (300 servers + 200 network)

❌ **Opção 1: Tudo via Agent**

- Instalação: 300 servers × 15min = 75h
- Network devices: **NÃO SUPORTAM** → 200 não monitorados ❌
- Resultado: **40% sem monitoramento**

✅ Opção 2: Híbrido Agent + SNMP

- Servers (300): Agent → 75h instalação
- Network (200): SNMP → **0h instalação** (já habilitado)
- Resultado: **100% monitorado**
- **Economia:** 50 horas + cobertura total 🎯

PARTE 2

Agente Zabbix - Análise Detalhada

Arquitetura do Agente

Componentes:

- `zabbix_agentd` → Processo principal
- `/etc/zabbix/zabbix_agentd.conf` → Configuração
- Log files → Atividade e erros
- Buffer interno → Modo ativo
- Módulos loadable → Extensões

Versões:

- Agent 1 (C) → Clássico, estável
- Agent 2 (Go) → Plugins, async, melhor performance

Modo Passivo vs Ativo

Modo Passivo:

```
Servidor  →  Request: system.cpu.load[all,avg1]  →  Agent
Servidor  ←  Response: 2.45                      ←  Agent
```

- Servidor inicia conexão
- Porta 10050 (agent escuta)
- Resposta imediata

Modo Ativo:

```
Agent  →  Request: Lista de itens      →  Servidor
Agent  ←  Response: [item1, item2]    ←  Servidor
Agent  →  Data: [value1, value2]       →  Servidor
```

- Agent inicia conexão
- Porta 10051 (servidor escuta)
- Buffer para falhas de rede

Configuração do Agente

Modo Passivo:

```
# /etc/zabbix/zabbix_agentd.conf
Server=192.168.1.100,192.168.1.101
ListenPort=10050
ListenIP=0.0.0.0
StartAgents=3
```

Modo Ativo:

```
ServerActive=192.168.1.100:10051
Hostname=web-server-01
RefreshActiveChecks=120
BufferSend=5
BufferSize=100
```

Itens Nativos: Sistema

```
# Informações gerais
system.hostname           # Nome do sistema
system.uname              # Kernel info
system.uptime             # Tempo de atividade
system.users.num          # Usuários conectados

# CPU
system.cpu.load[all,avg1]      # Load average 1min
system.cpu.util[,user]         # CPU user %
system.cpu.num[online]         # CPUs online

# Memória
vm.memory.size[available]      # Memória disponível
vm.memory.size[total]          # Memória total
```

Itens Nativos: Filesystem e Rede

Filesystem:

```
vfs.fs.size[/,used]          # Espaço usado
vfs.fs.size[/,pfree]         # % livre
vfs.fs.inode[/,pfree]        # % inodes livres
vfs.fs.discovery             # Discovery de FS
```

Rede:

```
net.if.in[eth0]              # Bytes recebidos
net.if.out[eth0]             # Bytes enviados
net.tcp.listen[80]           # Porta TCP listening
net.if.discovery             # Discovery de interfaces
```

Demonstração: zabbix_get

```
# Testar itens nativos
zabbix_get -s 192.168.1.10 -k system.uname
zabbix_get -s 192.168.1.10 -k system.cpu.load[all,avg1]
zabbix_get -s 192.168.1.10 -k vm.memory.size[available]
zabbix_get -s 192.168.1.10 -k vfs.fs.size[/,pused]
zabbix_get -s 192.168.1.10 -k net.if.in[eth0]
```

Você testará agora!

UserParameters Personalizados

Sintaxe:

```
UserParameter=key[*],command
```

Exemplos:

```
# Status de serviço
UserParameter=service.status[*],systemctl is-active $1

# MySQL conexões
UserParameter=mariadb.connections[*],\
  mysqladmin -u$1 -p$2 extended-status | \
  grep -w "Threads_connected" | cut -d"|" -f3

# Docker containers rodando
UserParameter=docker.containers.running,\
  docker ps -q | wc -l
```

🔬 Laboratório Prático 1

Objetivo: Criar UserParameters customizados

Exercício 1: Serviços systemd (15 min)

1. Criar UserParameter: `service.status[*]`
2. Testar local: `zabbix_agentd -t service.status[apache2]`
3. Testar remoto: `zabbix_get -s <host> -k service.status[apache2]`
4. Criar item no Zabbix

Checklist de Validação do Agente

Instalação:

- [ ] Agente instalado: `dpkg -l | grep zabbix-agent`
- [ ] Versão compatível: `zabbix_agentd -V`
- [ ] Config existe: `ls /etc/zabbix/zabbix_agentd.conf`
- [ ] Server configurado: `grep "^Server=" config`

Funcionamento:

- [ ] Serviço rodando: `systemctl status zabbix-agent`
- [ ] Porta aberta: `netstat -tulpn | grep 10050`
- [ ] Teste: `zabbix_get -s <ip> -k agent.ping`
- [ ] Logs sem erros: `tail -f zabbix_agentd.log`

PARTE 3

SNMP - Simple Network Management Protocol

Fundamentos do SNMP

Componentes:

- **SNMP Manager** → Zabbix Server/Proxy (cliente)
- **SNMP Agent** → Serviço nos dispositivos (servidor)
- **MIB** (Management Information Base) → Base de dados
- **OID** (Object Identifier) → Identificador único
- **PDU** (Protocol Data Unit) → Unidades de dados

Protocolo: UDP porta 161

Versões do SNMP

SNMPv1 (1988):

- ✅ Primeira versão, amplamente suportada
- ❌ Sem criptografia (plain text)
- ❌ Contadores 32 bits
- ❌ Autenticação simples (community)

SNMPv2c (1993):

- ✅ Contadores 64 bits
- ✅ Bulk operations (GetBulk)
- ✅ Melhor performance
- ❌ Ainda sem criptografia

SNMPv3: Segurança

Níveis de Segurança:

```
noAuthNoPriv  → Sem autenticação, sem criptografia
authNoPriv    → Com autenticação, sem criptografia
authPriv      → Com autenticação E criptografia ✅
```

Configuração SNMPv3:

```
# Dispositivo (Cisco)
snmp-server user zabbix-user ZABBIX-GROUP v3 \
  auth sha myauthpass \
  priv aes 128 myprivpass
```

Uso em Produção: Sempre SNMPv3 authPriv! 🔒

MIBs e OIDs Essenciais

System MIB (RFC 1213):

```
sysDescr      1.3.6.1.2.1.1.1.0    Descrição do sistema
sysUpTime     1.3.6.1.2.1.1.3.0    Uptime
sysName       1.3.6.1.2.1.1.5.0    Nome
sysLocation   1.3.6.1.2.1.1.6.0    Localização
```

Interface MIB:

```
ifDescr       1.3.6.1.2.1.2.2.1.2.X     Descrição interface X
ifOperStatus  1.3.6.1.2.1.2.2.1.8.X     Status operacional X
ifInOctets    1.3.6.1.2.1.2.2.1.10.X    Bytes recebidos
ifOutOctets   1.3.6.1.2.1.2.2.1.16.X    Bytes enviados
```

Ferramentas SNMP

Instalação:

```
# Ubuntu/Debian
sudo apt-get install snmp snmp-mibs-downloader

# CentOS/RHEL
sudo yum install net-snmp net-snmp-utils
```

Comandos:

```
# Get valor único
snmpget -v2c -c public 192.168.1.1 1.3.6.1.2.1.1.1.0

# Walk (árvore de valores)
snmpwalk -v2c -c public 192.168.1.1 1.3.6.1.2.1.2.2.1

# Bulk walk (mais eficiente)
snmpbulkwalk -v2c -c public 192.168.1.1 1.3.6.1.2.1.2
```

Demonstração SNMP

```
# SNMPv1
snmpget -v1 -c public 192.168.1.1 \
  1.3.6.1.2.1.1.1.0

# SNMPv2c
snmpwalk -v2c -c public 192.168.1.1 \
  1.3.6.1.2.1.2.2.1

# SNMPv3
snmpget -v3 -u zabbix-user -l authPriv \
  -a SHA -A myauthpass \
  -x AES -X myprivpass \
  192.168.1.1 1.3.6.1.2.1.1.1.0
```

Vamos testar juntos!

SNMP no Zabbix

Criar Host SNMP:

1. Configuration → Hosts → Create host
2. Host name: `Switch-Core-01`
3. Groups: `Network devices`
4. Interfaces: Add **SNMP interface**
    - IP: 127.0.0.1
    - Port: 161
    - SNMP version: SNMPv2
    - Community: `public`
5. Templates: `Linux by SNMP`

Itens SNMP no Zabbix

Método Legado:

```
SNMP OID: 1.3.6.1.2.1.1.1.0
```

Método Recomendado:

```
SNMP OID: get[1.3.6.1.2.1.1.1.0]
```

Bulk Operations:

```
SNMP OID: walk[1.3.6.1.2.1.2.2.1.2]
```

Discovery de Interfaces:

- Automático via LLD
- Template já inclui!

🔬 Laboratório Prático SNMP

Exercício 1: Crie o host Server Zabbix SNMP

1. Habilitar SNMP no servidor
2. Criar host no Zabbix (interface SNMP)
3. Aplicar template "Template Linux by SNMP"
4. Validar discovery de interfaces
5. Verificar coleta de métricas

PARTE 4

Criação de Hosts via Frontend e API

Criação via Frontend

Passo a passo:

1. Configuration → Hosts → Create host
2. **Host name:** web-server-01
3. **Groups:** Linux servers
4. **Interfaces:**
    - Agent: 192.168.1.10:10050
    - SNMP: 192.168.1.10:161 (se aplicável)
5. **Templates:** Template OS Linux by Zabbix agent
6. **Macros:** (se necessário)
7. Add

Criação via API: Autenticação

1. Obter token:

```
curl -X POST http://zabbix.example.com/api_jsonrpc.php \
  -H "Content-Type: application/json-rpc" \
  -d '{
    "jsonrpc": "2.0",
    "method": "user.login",
    "params": {
      "username": "Admin",
      "password": "zabbix"
    },
    "id": 1
  }'
```

Resposta:

```
{"jsonrpc":"2.0","result":"a1b2c3d4e5f6...","id":1}
```

Criação via API: host.create

```
curl -X POST https://zabbix.example.com/api_jsonrpc.php \
  -H "Content-Type: application/json-rpc" \
  -d '{
    "jsonrpc": "2.0",
    "method": "host.create",
    "params": {
      "host": "web-server-02",
      "interfaces": [{
        "type": 1,
        "main": 1,
        "useip": 1,
        "ip": "192.168.1.11",
        "dns": "",
        "port": "10050"
      }],
      "groups": [{"groupid": "2"}],
      "templates": [{"templateid": "10001"}]
    },
    "auth": "a1b2c3d4e5f6...",
    "id": 2
  }'
```

🔬 Laboratório Prático API

Exercício 1: Criar host via curl

1. Autenticar na API
2. Obter token
3. Criar host com interface Agent
4. Vincular template
5. Validar criação (host.get)

PARTE 5

Network Discovery e Auto Registration

Network Discovery

Funcionamento:

- Zabbix faz scan da rede
- Range de IPs configurável
- Checks: ICMP, SNMP, Agent, HTTP
- Actions baseadas em discovery
- Cria hosts automaticamente

Quando usar:

- Ambientes dinâmicos
- Muitos dispositivos similares
- Descobrir infraestrutura desconhecida

Configurar Network Discovery

Configuration → Discovery:

1. Name: `Production Network Discovery`
2. IP range: `192.168.1.1-254`
3. Update interval: `1h`
4. Checks:
    - ICMP ping
    - Zabbix agent (port 10050)
    - SNMP v2 (community: public)
5. Uniqueness criteria: `IP address`

Actions:

- Auto-add host
- Apply template
- Send notification

Active Agent Auto Registration

Diferença do Discovery:

- **Discovery:** Servidor procura hosts
- **Auto-registration:** Hosts se registram

Vantagens:

- Mais eficiente
- Ideal para ambientes cloud/container
- Hosts se auto-classificam (metadata)
- Escalável

Configuração:

- No agente: `ServerActive` + `HostMetadata`
- No Zabbix: Action de auto-registro

Configurar Auto-registro

No Agente:

```
# /etc/zabbix/zabbix_agentd.conf
ServerActive=172.16.1.110:10051
Hostname=web-server-03
HostMetadata=linux,production,web
HostInterface=192.168.1.3
```

No Zabbix - Action:

1. Configuration → Actions → Autoregistration actions
2. Name: `Auto-register production Linux`
3. Conditions:
    - Host metadata contains `production`

4. Operations:
    - Add to group: `Production Servers`
    - Link template: `Template OS Linux by Zabbix agent`

Metadata para Classificação

Exemplos de Metadata:

```
# Ambiente + Tipo + Aplicação
HostMetadata=linux,production,web
HostMetadata=linux,development,database
HostMetadata=windows,production,app

# Action processa com regex
Condition: Host metadata matches regex "production.*web"
Operation: Link template "Template Nginx"
```

Flexível e poderoso!

🔬 Laboratório Prático Auto-registro

Objetivo: Implementar auto-registro

Tarefas:

1. Configurar agente:
   - `ServerActive=<zabbix-server>`
   - `HostMetadata=linux,production,web`
2. Criar Action no Zabbix:
   - Condição: metadata contains "production"
   - Adicionar a grupo "Production Servers"
   - Vincular template apropriado
3. Reiniciar agente

PARTE 7

Mapeamento de Valores e Macros

Value Mapping

Problema:

- Item retorna: 0, 1, 2
- Trigger mostra: "Status is 0"
- Usuário: "O que significa 0?" 🤔

Solução:

```
Value Map: Interface Status
0 → Down
1 → Up
2 → Unknown
```

Resultado:

- Latest Data: "Status is **Up**" ✅

Macros Aplicadas aos Métodos

SNMP:

```
{$SNMP.COMMUNITY} = "public"
{$SNMP.COMMUNITY:DMZ} = "dmz_community"
{$SNMP.COMMUNITY:DATACENTER} = "dc_community"
```

HTTP Agent:

```
{$API.TOKEN} = "Bearer abc123..."
{$API.TOKEN:PRODUCTION} = "Bearer prod_xyz..."
{$API.TOKEN:DEVELOPMENT} = "Bearer dev_123..."
```

Agent:

```
{$AGENT.TIMEOUT} = "30"
```

Context Macros na Prática

Configurar:

```
Global:
{$SNMP.COMMUNITY:192.168.1.*} = "network_comm"
{$SNMP.COMMUNITY:10.0.*.*} = "datacenter_comm"
{$SNMP.COMMUNITY} = "default_comm"
```

Resolução:

- Switch 192.168.1.1 → usa "network_comm"
- Switch 10.0.1.1 → usa "datacenter_comm"
- Switch 172.16.1.1 → usa "default_comm"

Flexível e seguro! 🔒

ENCERRAMENTO

Comparação Final

Método	Quando Usar	Vantagem	Desvantagem
Agent	Servers	Performance, detalhes	Instalação
SNMP	Network devices	Universal, agentless	OIDs complexos
HTTP	APIs, SaaS	Flexível, moderno	Latência

Regra de ouro:

- Pode instalar? → **Agent**
- Dispositivo de rede? → **SNMP**
- API REST? → **HTTP Agent**

Recursos Úteis

📚 **Documentação API:**
https://www.zabbix.com/documentation/7.0/en/manual/api

📡 **SNMP MIBs:**
http://www.oidview.com/mibs/0/md-0.html

🧪 **JSONPath Tester:**
https://jsonpath.com/

🌐 **API de Teste:**
https://jsonplaceholder.typicode.com/

Troubleshooting Comum

Agent não conecta:

→ Verificar firewall (10050/10051), Server=, conectividade

Auto-registro não funciona:

→ Verificar ServerActive=, HostMetadata=, logs, Action

SNMP timeout:

→ Community string, firewall UDP 161, timeout

JSONPath erro:

→ Validar JSON response online, sintaxe

API permission error:

→ Role do usuário, permissões, token válido

Obrigado!

Até a próxima aula! 🚀