Zabbix Advanced

Aula 04: Coleta via API e Integrações

4Linux - Curso Avançado

Agenda do Dia

1. **Fundamentos da API Zabbix**
    - JSON-RPC 2.0, métodos de autenticação, CRUD
2. **Automação de Operações Administrativas**
    - Criação em massa, CMDB sync, CI/CD
3. **Integrações com Sistemas Externos**
    - Ticketing, Dashboards, Cloud Providers
4. **Casos Práticos e Laboratórios**
    - Criar 100 hosts via CSV, sincronização CMDB

PARTE 1

Fundamentos da API Zabbix

Objetivos de Aprendizagem

Ao final desta aula, você será capaz de:

- ✅ Compreender fundamentos da API Zabbix
- ✅ Implementar autenticação (user/password e API tokens)
- ✅ Realizar operações CRUD via API
- ✅ Automatizar tarefas administrativas em massa
- ✅ Integrar Zabbix com sistemas externos
- ✅ Implementar controles de segurança
- ✅ Otimizar performance de integrações

Recap Aula 03

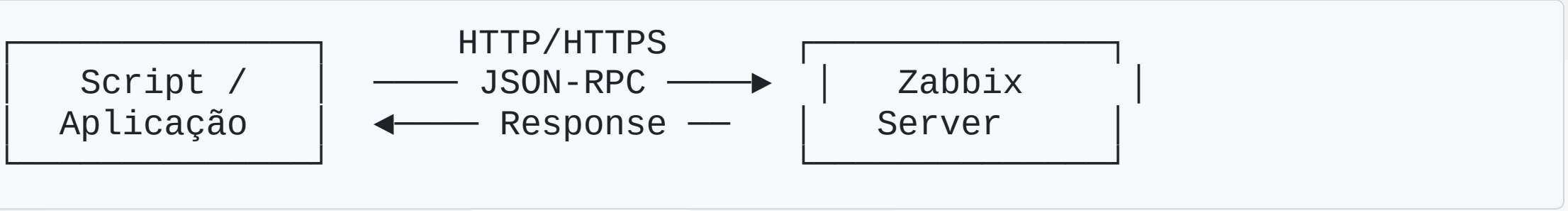
O que vimos:

- SNMP (v1, v2c, v3)
- Trabalho com MIBs (RFC1213, IF-MIB)
- OIDs e estrutura hierárquica
- GET vs GET-BULK (70% mais rápido)
- Discovery de interfaces
- Troubleshooting SNMP

Hoje: Automação e Integrações via API! 🚀

O Que É a API do Zabbix?

API = Interface programática para acesso completo ao Zabbix



Protocolo: JSON-RPC 2.0

Endpoint: `/api_jsonrpc.php`

Formato: JSON (request e response)

Por Que Usar a API?

Problemas que resolve:

❌ Criar 1000 hosts manualmente (inviável)
❌ Manter inventário sincronizado com CMDB
❌ Integrar monitoramento com CI/CD
❌ Gerar relatórios customizados
❌ Automatizar manutenções programadas

Solução: Automação via API!

✅ Cria 1000 hosts em segundos
✅ Sincronização automática
✅ Provisiona monitoramento no deploy
✅ Exporta dados para qualquer sistema
✅ Scripts de manutenção programada

Casos de Uso - ROI

Caso de Uso	Manual	API	Economia
Criar 100 hosts	50h	5min	**99.8%**
Sync CMDB	Desatualizado	Tempo real	**100%**
Deploy com monitoring	30min/host	30s	**99%**
Relatórios customizados	N/A	Automático	**∞**
Integrações tickets	Manual	Automático	**-80% tempo**

Resultado: Operações escaláveis, zero drift, alta automação

JSON-RPC 2.0 - Estrutura Básica

Requisição:

```
{
  "jsonrpc": "2.0",
  "method": "host.get",
  "params": {
    "output": ["hostid", "host"],
    "limit": 10
  },
  "auth": "auth_token_aqui",
  "id": 1
}
```

Campos obrigatórios:

- `jsonrpc` : Sempre "2.0"
- `method` : Formato `objeto.ação`
- `params` : Parâmetros do método
- `auth` : Token (exceto em user.login)
- `id` : Identificador da requisição

JSON-RPC 2.0 - Resposta

Sucesso:

```
{
  "jsonrpc": "2.0",
  "result": [
    {"hostid": "10001", "host": "Server-01"},
    {"hostid": "10002", "host": "Server-02"}
  ],
  "id": 1
}
```

Principais Objetos da API

Objeto	Descrição	Métodos Comuns *
`host`	Gerenciar hosts	get, create, update, delete
`item`	Gerenciar items	get, create, update, delete
`trigger`	Gerenciar triggers	get, create, update, delete
`template`	Gerenciar templates	get, create, update, delete
`hostgroup`	Grupos de hosts	get, create, update, delete
`user`	Usuários	get, create, update, login, logout
`problem`	Problemas ativos	get
`event`	Eventos históricos	get, acknowledge

Métodos de Autenticação

1. User/Password (Session-based)

Login:

```
{
  "jsonrpc": "2.0",
  "method": "user.login",
  "params": {
    "username": "Admin",
    "password": "zabbix"
  },
  "id": 1
}
```

Resposta:

```
{
  "result": "0424bd59b807674191e7d77572075f33"
}
```

🔑 Token retornado deve ser usado em todas requisições seguintes

Métodos de Autenticação

2. API Token ⭐

Como criar no Zabbix 7.0 LTS:

```
Users → API tokens → Create API token
- Name: IntegrationCMDB
- User: Admin (ou usuário específico)
- Set expiration date and time: (desmarcar = permanente)
- Enabled: ✓
```

Uso direto:

```
{
  "jsonrpc": "2.0",
  "method": "host.get",
  "params": {"output": ["hostid", "host"]},
  "auth": "f223afsf3e1344ee4e3ce69934a8389dd93",
  "id": 1
}
```

✅ **Não expira** | ✅ **Revogável** | ✅ **Auditável**

Comparação de Autenticação

Aspecto	User/Password	API Token
Segurança	⚠️ Média (senha texto)	✅ Alta (revogável)
Duração	Expira após inatividade	Permanente
Auditoria	Por usuário	Por token
Melhor para	Dev/Teste	**Produção**
Disponível	Todas versões	Zabbix 5.4+

🔒 **Recomendação:** Use API Tokens em produção!

Operações CRUD - CREATE

Criar host:

```
{
  "jsonrpc": "2.0",
  "method": "host.create",
  "params": {
    "host": "webserver-prod-01",
    "name": "Web Server Production 01",
    "interfaces": [{
      "type": 1, "main": 1, "useip": 1,
      "ip": "192.168.1.100", "port": "10050"
    }],
    "groups": [{"groupid": "10"}],
    "templates": [{"templateid": "10001"}]
  },
  "auth": "token",
  "id": 1
}
```

Resposta: `{"hostids": ["10084"]}`

Operações CRUD - READ

Listar hosts:

```
{
  "jsonrpc": "2.0",
  "method": "host.get",
  "params": {
    "output": ["hostid", "host", "name", "status"],
    "selectInterfaces": ["ip", "port"],
    "selectGroups": ["name"],
    "filter": {"status": "0"}
  },
  "auth": "token",
  "id": 1
}
```

Parâmetros comuns:

- `output` : Campos a retornar
- `filter` : Filtro exato
- `search` : Busca parcial
- `limit` : Limitar resultados
- `select*` : Incluir objetos relacionados

Operações CRUD - UPDATE

Atualizar IP do host:

```
{
  "jsonrpc": "2.0",
  "method": "host.update",
  "params": {
    "hostid": "10084",
    "interfaces": [{
      "interfaceid": "30001",
      "ip": "192.168.1.200"
    }]
  },
  "auth": "token",
  "id": 1
}
```

Atualização em massa:

```
{
  "method": "host.massupdate",
  "params": {
    "hosts": [{"hostid": "10084"}, {"hostid": "10085"}],
    "templates": [{"templateid": "10001"}]
  }
}
```

Operações CRUD - DELETE

Deletar host:

```
{
  "jsonrpc": "2.0",
  "method": "host.delete",
  "params": ["10084"],
  "auth": "token",
  "id": 1
}
```

Deletar múltiplos hosts:

```
{
  "jsonrpc": "2.0",
  "method": "host.delete",
  "params": ["10084", "10085", "10086"],
  "auth": "token",
  "id": 1
}
```

⚠️ **Cuidado:** Operação irreversível!

Boas Práticas de Performance

1. Selecionar apenas campos necessários:

❌ Ruim:

```
"params": {
  "output": "extend"  // Retorna TUDO (lento)
}
```

✅ Bom:

```
"params": {
  "output": ["hostid", "host"]  // Apenas o necessário
}
```

Resultado: 3x-10x mais rápido!

Boas Práticas de Performance

2. Usar limit em consultas grandes:

```
"params": {
  "output": ["hostid", "host"],
  "limit": 100
}
```

3. Paginação para grandes volumes:

```
limit = 100
offset = 0
while True:
    result = zabbix.host.get(limit=limit, offset=offset)
    if not result:
        break
    process(result)
    offset += limit
```

4. Cachear resultados quando possível

Segurança - Melhores Práticas

🔒 **Sempre use HTTPS** em produção
🔑 **API tokens** ao invés de user/password
📝 **Um token por integração** (facilita revogação)
🚫 **Nunca commitar tokens** em Git
🔄 **Rotacionar tokens** periodicamente
👤 **Usuários dedicados** para API (não Admin)
📊 **Monitore uso da API** (auditoria)
⚡ **Rate limiting** se expor externamente

PARTE 2

Automação de Operações Administrativas

Criação em Massa de Hosts

Cenário: Criar 100 novos servidores no Zabbix a partir de uma lista.

Abordagem Manual:

- Tempo: ~30 min por host = **50 horas**
- Erro humano: Alto
- Padronização: Baixa

Abordagem API:

- Tempo: **~5 minutos**
- Erro: Zero (validado)
- Padronização: 100%

ROI: 99.8% economia de tempo! ⚡

Script Python - Criar Hosts em Massa

```python
import csv
from pyzabbix import ZabbixAPI

zapi = ZabbixAPI('http://zabbix.local')
zapi.login('Admin', 'zabbix')

with open('servers.csv') as f:
    reader = csv.DictReader(f)
    for row in reader:
        zapi.host.create(
            host=row['hostname'],
            name=row['name'],
            interfaces=[{'type': 1, 'main': 1, 'useip': 1,
                         'ip': row['ip'], 'port': '10050'}],
            groups=[{'groupid': row['groupid']}],
            templates=[{'templateid': row['templateid']}]
        )
        print(f"✅ Criado: {row['hostname']}")
```

Formato CSV de Exemplo

```
hostname,name,ip,groupid,templateid
web-prod-01,Web Server 01,192.168.1.10,10,10001
web-prod-02,Web Server 02,192.168.1.11,10,10001
db-prod-01,Database Server 01,192.168.2.10,11,10002
db-prod-02,Database Server 02,192.168.2.11,11,10002
app-prod-01,App Server 01,192.168.3.10,12,10003
```

Executar:

```
python create_hosts.py
```

Resultado: 5 hosts criados em segundos! ✨

Sincronização com CMDB

Problema: CMDB desatualizado causa:

- ❌ Hosts órfãos no Zabbix
- ❌ Servidores sem monitoramento
- ❌ Inventário inconsistente

Solução: Sincronização automática via API

Script - Sync CMDB → Zabbix

```
import requests
from pyzabbix import ZabbixAPI

cmdb_servers = requests.get('https://cmdb.local/api/servers').json()

zapi = ZabbixAPI('http://zabbix.local')
zapi.login('Admin', 'zabbix')

existing = {h['host']: h for h in zapi.host.get(output=['hostid', 'host'])}

for server in cmdb_servers:
    if server['hostname'] not in existing:
        zapi.host.create(
            host=server['hostname'],
            name=server['name'],
            interfaces=[{'type': 1, 'main': 1, 'useip': 1,
                         'ip': server['ip'], 'port': '10050'}],
            groups=[{'groupid': server['group_id']}]
        )
        print(f"✅ Criado: {server['hostname']}")
    else:
        print(f"⏭️  Já existe: {server['hostname']}")
```

Sincronização Bidirecional

CMDB ← → Zabbix

Cenário 1: CMDB como fonte da verdade

- CMDB cria servidor → Script cria no Zabbix
- CMDB desativa → Script desabilita no Zabbix

Cenário 2: Enriquecer CMDB com dados Zabbix

- Zabbix coleta uptime → Enviar para CMDB
- Zabbix detecta problema → Atualizar status CMDB

Implementação: Cronjob (ex: `0 */4 * * *` - a cada 4h)

```
# /etc/cron.d/zabbix-cmdb-sync
0 */4 * * * user /opt/scripts/sync_cmdb_zabbix.py
```

Integração com CI/CD

Cenário: Deploy automático com monitoramento desde o dia zero

Pipeline típico:

```
Code → Build → Test → Deploy → [Configure Monitoring]
```

Implementação:

```yaml
# .gitlab-ci.yml
deploy_production:
  stage: deploy
  script:
    - ansible-playbook deploy.yml
    - python add_to_zabbix.py --host $CI_SERVER_NAME --ip $SERVER_IP
  only:
    - production
```

Resultado: Servidor deployado + monitorado automaticamente! 🚀

Script CI/CD - Adicionar ao Zabbix

```
#!/usr/bin/env python3
import argparse
from pyzabbix import ZabbixAPI

parser = argparse.ArgumentParser()
parser.add_argument('--host', required=True)
parser.add_argument('--ip', required=True)
parser.add_argument('--env', default='production')
args = parser.parse_args()

zapi = ZabbixAPI('http://zabbix.local')
zapi.login(user='api-cicd', password='token')

template_map = {'production': '10001', 'staging': '10002',
                'development': '10003'}

zapi.host.create(
    host=args.host,
    interfaces=[{'type': 1, 'main': 1, 'useip': 1,
                 'ip': args.ip, 'port': '10050'}],
    groups=[{'groupid': '10'}],
    templates=[{'templateid': template_map[args.env]}]
)
print(f"✅ Host {args.host} added to Zabbix")
```

Operações de Manutenção Programada

Cenário: Janela de manutenção semanal (domingos 2-6am)

Manual: Criar maintenance window toda semana ❌
API: Script automatizado ✅

Vantagens:

- Automação completa
- Sem intervenção humana
- Agendamento preciso
- Pode ser integrado ao CI/CD

Script - Manutenção Programada

```
from pyzabbix import ZabbixAPI
from datetime import datetime, timedelta

zapi = ZabbixAPI('http://zabbix.local')
zapi.login('Admin', 'zabbix')

next_sunday = datetime.now() + timedelta(days=(6-datetime.now().weekday()))
start_time = next_sunday.replace(hour=2, minute=0, second=0)
end_time = start_time + timedelta(hours=4)

zapi.maintenance.create(
    name=f"Weekly Maintenance - {start_time.strftime('%Y-%m-%d')}",
    active_since=int(start_time.timestamp()),
    active_till=int(end_time.timestamp()),
    hostids=['10084', '10085'],
    timeperiods=[{
        'timeperiod_type': 0,
        'start_date': int(start_time.timestamp()),
        'period': 14400
    }]
)
```

PARTE 3

Integrações com Sistemas Externos

Integração com Sistemas de Ticketing

Cenário: Problema no Zabbix → Abrir ticket automaticamente

Sistemas suportados:

- ServiceNow
- Jira Service Desk
- Zendesk
- OpsGenie
- PagerDuty
- TOPdesk

Fluxo:

```
Zabbix Trigger → Webhook → Ticketing System API
                         → Cria Ticket
                         → Atribui time suporte
                         → Envia notificação
```

Exemplo: Integração Zabbix → Jira

Configuração Webhook Zabbix:

```
Administration → Media types → Create media type
- Type: Webhook
- Name: Jira Integration
- Parameters:
  - jira_url: https://empresa.atlassian.net
  - jira_user: zabbix@empresa.com
  - jira_token: API_TOKEN
  - project_key: OPS
```

Script webhook (JavaScript):

```
var req = new HttpRequest();
req.addHeader('Content-Type: application/json');
req.addHeader('Authorization: Basic ' + btoa(params.jira_user + ':' + params.jira_token));

var data = {
  "fields": {
    "project": {"key": params.project_key},
    "summary": params.subject,
    "description": params.message,
    "issuetype": {"name": "Incident"}
  }
};

var resp = req.post(params.jira_url + '/rest/api/2/issue', JSON.stringify(data));
return resp;
```

Resultado: Ticket Automático

Trigger dispara no Zabbix:

```
Host: web-prod-01
Problem: High CPU usage (>90% for 5 min)
Severity: High
```

Ticket criado no Jira:

```
Project: OPS
Type: Incident
Priority: High
Summary: [Zabbix] High CPU usage on web-prod-01
Description:
  Host: web-prod-01
  Problem: High CPU usage (>90% for 5 min)
  Current value: 95%
  Time: 2025-01-10 14:35:00
  Link: http://zabbix.local/tr_events.php?triggerid=13491
```

Atribuição automática: Time de infraestrutura

Export de Dados para Dashboards Externos

Cenário: Stakeholders querem dashboards customizados

Destinos comuns:

- Grafana (visualização)
- Tableau (BI)
- Power BI (Microsoft)
- Elasticsearch (SIEM)
- Prometheus (métricas)

Vantagem: Combinar dados Zabbix + outras fontes

Exemplo: Zabbix → Grafana

1. Instalar plugin Grafana:

```
grafana-cli plugins install alexanderzobnin-zabbix-app
# Plugin disponível imediatamente, sem necessidade de restart
```

2. Habilitar plugin no Grafana:

```
Configuration → Plugins → Zabbix → Enable
```

3. Configurar datasource:

```
Configuration → Data sources → Add Zabbix
- URL: http://zabbix.local/api_jsonrpc.php
- Access: Server
- Auth: API Token
- Token: <seu_token_aqui>
```

3. Criar dashboard:

- Query: `host.get` + `item.get` + `history.get`
- Visualização: Time series, gauge, table, etc.

Resultado: Dashboard Grafana com dados do Zabbix! 📊

Export de Histórico para Análise

Cenário: Análise de tendências, ML, relatórios executivos

Script - Export para CSV:

```
from pyzabbix import ZabbixAPI
import csv
from datetime import datetime, timedelta

zapi = ZabbixAPI('http://zabbix.local')
zapi.login('Admin', 'zabbix')

end_time = datetime.now()
start_time = end_time - timedelta(days=30)

history = zapi.history.get(
    itemids=['23298'], time_from=int(start_time.timestamp()),
    time_till=int(end_time.timestamp()), output='extend',
    sortfield='clock', sortorder='ASC'
)

with open('cpu_history.csv', 'w') as f:
    writer = csv.writer(f)
    writer.writerow(['Timestamp', 'Value'])
    for item in history:
        writer.writerow([datetime.fromtimestamp(int(item['clock'])),
                         item['value']])
```

PARTE 4

Casos Práticos e Laboratórios

🔬 Laboratório Prático 1

Objetivo: Criar 100 hosts a partir de arquivo CSV

Arquivo: `servers.csv`

```
hostname,name,ip,groupid,templateid
web-01,Web Server 01,10.0.1.10,10,10001
web-02,Web Server 02,10.0.1.11,10,10001
...
```

Tarefas:

1. Baixar script Python: `create_hosts_bulk.py`
2. Instalar dependência: `pip install pyzabbix`
3. Configurar credenciais no script

4. Executar: `python create_hosts_bulk.py servers.csv`
5. Verificar no Zabbix: Configuration → Hosts

Resultado esperado: 100 hosts criados em ~2 minutos

🔬 Laboratório Prático 2

Objetivo: Sincronização com CMDB simulado

Cenário: JSON "CMDB" → Zabbix

```
[
  {"hostname": "app-01", "ip": "10.0.2.10", "env": "production"},
  {"hostname": "app-02", "ip": "10.0.2.11", "env": "production"},
  {"hostname": "app-03", "ip": "10.0.2.12", "env": "staging"}
]
```

Tarefas (30 min):

1. Criar arquivo `cmdb.json` com 10 servidores
2. Script: `sync_cmdb.py`
3. Executar primeira sync (cria hosts)
4. Modificar `cmdb.json` (adicionar/remover)
5. Executar segunda sync (detecta mudanças)

Desafio extra: Implementar lógica de desativação (não deletar)

Troubleshooting Comum

Problema 1: "Session terminated"

Causa: Token expirado (inatividade)

Solução:

```
try:
    result = zapi.host.get(...)
except ZabbixAPIException as e:
    if 'Session terminated' in str(e):
        zapi.login(USERNAME, PASSWORD)
        result = zapi.host.get(...)  # Retry
```

Problema 2: "No permissions to referred object"

Causa: Usuário sem permissão no objeto

Solução:

- Verificar user role
- Usar user com permissões adequadas
- Administration → User roles → Verificar permissions

Problema 3: Rate limiting / Timeout

Causa: Muitas requisições simultâneas

Solução:

```
import time

for server in servers:
    zapi.host.create(...)
    time.sleep(0.1)  # 100ms delay entre requests
```

Alternativa: Usar `massadd` ao invés de múltiplos `create`

Problema 4: Erro "Already exists"

Causa: Tentando criar host duplicado

Solução:

```
# Verificar antes de criar
existing = zapi.host.get(filter={'host': hostname})
if not existing:
    zapi.host.create(...)
else:
    print(f"⏭️  Já existe: {hostname}")
```

PARTE 5

Melhores Práticas e Segurança

Segurança - Checklist

🔒 **1. Use HTTPS** em produção (TLS 1.2+)

🔑 **2. API Tokens** ao invés de senha

- Um token por aplicação
- Descrição clara do uso
- Revogação fácil

👤 **3. Usuários dedicados para API**

- Não use conta Admin
- Permissões mínimas necessárias
- Nome descritivo (ex: `api-cmdb-sync`)

Segurança - Checklist (cont.)

📝 4. Auditoria

- Monitorar chamadas API suspeitas
- Log de criação/alteração via API
- Dashboard de uso da API

🚫 5. Proteção de credenciais

- Nunca commitar tokens em Git
- Usar variáveis de ambiente
- Secrets management (Vault, AWS Secrets)

Segurança - Rate Limiting

🛡️ 6. Rate limiting

- Nginx/Apache: limitar requisições por segundo
- Firewall: whitelist IPs autorizados
- Proteção contra DDoS e uso abusivo

Exemplo Nginx:

```
limit_req_zone $binary_remote_addr zone=api:10m rate=10r/s;

location /api_jsonrpc.php {
    limit_req zone=api burst=20;
}
```

Exemplo: Variáveis de Ambiente

Arquivo `.env`:

```
ZABBIX_URL=https://zabbix.empresa.com
ZABBIX_TOKEN=f223afsf3e1344ee4e3ce69934a8389dd93
```

Script Python:

```python
import os
from pyzabbix import ZabbixAPI

ZABBIX_URL = os.getenv('ZABBIX_URL')
ZABBIX_TOKEN = os.getenv('ZABBIX_TOKEN')

zapi = ZabbixAPI(ZABBIX_URL)
zapi.login(api_token=ZABBIX_TOKEN)
```

Executar:

```
export $(cat .env | xargs)
python script.py
```

`.env` no `.gitignore`! ✅

Performance - Otimizações

1. Consultas eficientes:

❌ Ruim:

```
for host in all_hosts:
    items = zapi.item.get(hostids=host['hostid'])
    # 1000 hosts = 1000 requests
```

✅ Bom:

```
all_items = zapi.item.get(hostids=[h['hostid'] for h in all_hosts])
# 1000 hosts = 1 request
```

Resultado: 1000x mais rápido!

Performance - Otimizações (cont.)

2. Paginação para grandes volumes:

```
def get_all_hosts(zapi):
    limit = 100
    offset = 0
    all_hosts = []

    while True:
        batch = zapi.host.get(output=['hostid', 'host'],
                              limit=limit, offset=offset)
        if not batch:
            break
        all_hosts.extend(batch)
        offset += limit

    return all_hosts
```

3. Cache quando possível:

- Templates (raramente mudam)
- Host groups
- Usar TTL apropriado (ex: 1 hora)

Performance - Operações em Massa

Cenário: Aplicar template em 1000 hosts

❌ Ruim (1000 requests):

```
for host in hosts:
    zapi.host.update(
        hostid=host['hostid'],
        templates=[{'templateid': '10001'}]
    )
```

✅ Bom (1 request):

```
zapi.host.massupdate(
    hosts=[{'hostid': h['hostid']} for h in hosts],
    templates=[{'templateid': '10001'}]
)
```

Tempo: 5 minutos → **5 segundos**!

Tratamento de Erros Robusto

```python
from pyzabbix import ZabbixAPI, ZabbixAPIException
import time

def create_host_with_retry(zapi, params, max_retries=3):
    for attempt in range(max_retries):
        try:
            result = zapi.host.create(**params)
            return result
        except ZabbixAPIException as e:
            if 'already exists' in str(e).lower():
                print(f"⏭️  Host já existe")
                return None
            elif 'session terminated' in str(e).lower():
                print(f"🔄 Reautenticando...")
                zapi.login(USERNAME, PASSWORD)
            elif attempt < max_retries - 1:
                wait = 2 ** attempt  # Exponential backoff
                print(f"⏳ Tentativa {attempt+1} falhou. Aguardando {wait}s...")
                time.sleep(wait)
            else:
                print(f"❌ Erro após {max_retries} tentativas: {e}")
                raise
```

Monitoramento da Própria API

Métrica importante: Tempo de resposta da API

```
import time

start = time.time()
result = zapi.host.get(output=['hostid'])
elapsed = time.time() - start

print(f"API response time: {elapsed:.2f}s")

# Enviar para Zabbix (meta-monitoramento!)
zapi.item.update(
    itemid='99999',  # Item "API Response Time"
    value=elapsed
)
```

Criar trigger: API response > 5s = Warning

Documentação e Exemplos

Recursos oficiais:

📖 **Documentação API:**
https://www.zabbix.com/documentation/current/en/manual/api

🐍 **PyZabbix (Python):**
https://github.com/lukecyca/pyzabbix

💎 **Zabbix Ruby Client:**
https://github.com/express42/zabbixapi

🟢 **Node.js Client:**
https://github.com/alexisSirignan/node-zabbix

🟦 **Exemplos práticos:**
https://github.com/zabbix/community-templates

Recursos Úteis

Ferramentas:

🛠️ **Postman Collection** - Testar API visualmente
🧪 **Zabbix API Tester** - Web UI para testes
📊 **Grafana Zabbix Plugin** - Visualização avançada
🤖 **Ansible Zabbix Modules** - Automação infraestrutura
🐋 **Zabbix Docker** - Ambiente de testes

Comunidade:

💬 **Forum:** https://www.zabbix.com/forum/
📱 **Telegram:** @ZabbixBrasil
🐦 **Twitter:** @zabbix

Recap dos Principais Conceitos

✅ **API Zabbix** = Automação total, integração, escala
✅ **JSON-RPC 2.0** = Protocolo simples e universal
✅ **Autenticação** = API Tokens (prod) > User/Pass (dev)
✅ **CRUD** = get, create, update, delete, massupdate
✅ **Use cases** = Hosts em massa, CMDB, CI/CD, ticketing
✅ **Segurança** = HTTPS, tokens, permissions, auditoria
✅ **Performance** = Paginação, cache, operações em massa
✅ **Tratamento erros** = Retry, backoff, logging

Mensagem-chave: API = Superpoder para escala! 🚀

Comparação Final

Sem API:

- ❌ Criação manual (50h para 100 hosts)
- ❌ Inventário desatualizado
- ❌ Sem integração CI/CD
- ❌ Relatórios limitados
- ❌ Tickets manuais

Com API:

- ✅ Criação automatizada (5min para 1000 hosts)
- ✅ CMDB sincronizado em tempo real
- ✅ Monitoramento desde o deploy
- ✅ Dashboards customizados ilimitados

Próximos Passos

1. ✅ **Implementar primeiro script** (criar hosts)
2. 🔍 **Explorar objetos da API** (items, triggers, templates)
3. 📊 **Integrar com 1 sistema externo** (Jira, Grafana)
4. 📈 **Automatizar operação recorrente** (maintenance, reports)
5. 🚨 **Configurar monitoramento da API** (response time)

Recursos Adicionais

Bibliotecas Python:

- `pyzabbix` - Mais popular
- `zabbix-api` - Alternativa

Exemplo completo no GitHub:

```
git clone https://github.com/4linux/zabbix-api-examples
cd zabbix-api-examples
pip install -r requirements.txt
python examples/create_hosts.py
```

Documentação offline:

- Download: `/usr/share/doc/zabbix-frontend-php/api`

ENCERRAMENTO

Perguntas Frequentes

Q: Posso usar API com Zabbix Proxy?

A: Não diretamente. API sempre no Zabbix Server.

Q: Qual limite de requisições por segundo?

A: Sem limite hard-coded, mas respeite performance (use paginação).

Q: API funciona com autenticação LDAP/SAML?

A: Sim! Faça login normal, token funciona igual.

Q: Posso usar API em scripts Shell?

A: Sim! Use `curl` com JSON. Python é mais conveniente.

Q: Como debugar requisições?

A: Use Postman, ou ative debug no Zabbix frontend.

🎉 Fim da Aula 04!

Próxima aula:

Alertas Avançados e Notificações