

Zabbix Advanced

Aula 05: Configuração de Alertas e Ações Automatizadas

4Linux - Curso Avançado

Agenda do Dia

1. Fundamentos das Actions

- Arquitetura, event sources, condições avançadas

2. Configuração de Media Types

- Telegram, Slack, MS Teams, Webhooks

3. Templates de Notificação Avançados

- Baseados em severidade, recuperação, reports

Agenda do Dia (cont.)

4. Scaling Inteligente

- Por horário, níveis hierárquicos

5. Laboratórios Práticos

- Media types personalizados, auto-remediation

PARTE 1

Fundamentos das Actions no Zabbix

O Que São Actions?

Actions = Automação de resposta a eventos

Evento (Trigger) → Condições → Actions

- └─ Notificações
- └─ Comandos Remotos
- └─ Webhooks

Transformam monitoramento passivo em ativo:

- ✗ Apenas detectar problemas
- ✓ Detectar + Notificar + Remediar

Anatomia de uma Action

3 Elementos Fundamentais:

1. Conditions (Condições)

- Critérios para execução
- Filtros: severidade, host group, trigger name, horário

2. Operations (Operações)

- Ações quando problema ocorre
- Send message, execute command, webhook

3. Recovery Operations

- Ações quando problema é resolvido

Event Sources

Tipos de eventos que geram actions:

Source	Descrição	Exemplo
Triggers	Problemas de itens	CPU > 90%
Discovery	Auto discovery	Novo host na rede
Auto registration	Host se registra	Container iniciado
Internal	Eventos internos	Item unsupported
Service	Monitoramento de serviço	SLA breach

Mais comum: Triggers (99% dos casos)

Event Values para Triggers

```
EVENT_VALUES = {  
    0: "OK",           # Problema resolvido  
    1: "PROBLEM"      # Problema ativo  
}
```

Fluxo típico:

1. Trigger dispara (PROBLEM)
→ Action executa Operations
2. Problema resolve (OK)
→ Action executa Recovery Operations

Severidades (Prioridades)

```
0: "Not classified"    # Sem classificação
1: "Information"       # Informativo
2: "Warning"          # Alerta
3: "Average"           # Médio
4: "High"              # Alto
5: "Disaster"          # Desastre
```

Uso em Actions:

- Notificar apenas severidade \geq High
- Escalar Disaster para gerência
- Auto-remediation apenas para Average/High

Condições Avançadas

Operadores Lógicos:

- **AND:** Todas condições verdadeiras
- **OR:** Pelo menos uma verdadeira
- **AND/OR:** Combinação customizada

Exemplo 1: Problemas críticos em produção

```
Trigger severity >= High  
AND  
Host group = "Production Servers"  
AND  
Trigger name like "CPU|Memory|Disk"
```

Exemplos de Condições Práticas

Exemplo 2: Horário comercial apenas

```
Event time >= 08:00  
AND  
Event time <= 18:00  
AND  
Host group != "Development"
```

Exemplo 3: Problemas recorrentes não-acked

```
Trigger value = PROBLEM  
AND  
Event acknowledged = No  
AND  
Age of event >= 5m
```








PARTE 2

Configuração de Media Types

O Que São Media Types?

Media Type = Canal de comunicação para notificações

Tipos disponíveis:

-  **Email** (SMTP)
-  **SMS** (gateways SMS)
-  **Telegram** (bot)
-  **Slack** (webhook)
-  **MS Teams** (webhook)
-  **Webhook** (customizado)
-  **Script** (executável)

Cada tipo tem parâmetros específicos

Integração com Telegram

Pré-requisitos: Criar Bot

Passo 1: Criar bot via BotFather

1. Abra Telegram → Procure `@BotFather`
2. Digite `/newbot`
3. Nome: "Zabbix Monitor"
4. Username: "empresa_zabbix_bot"
5. **Anote o Token:** `123456789:ABCdefGHIjklMNOpqrSTUVwxyz`

Telegram - Configurar


Passo 2: Obter Chat ID

1. Adicione bot ao grupo
2. Envie mensagem: `/start`
3. Acesse: `https://api.telegram.org/bot<TOKEN>/getUpdates`
4. **Anote o chat.id** (ex: -1001234567890)

Telegram - Teste do Bot

```
# Teste básico
BOT_TOKEN="SEU_TOKEN_AQUI"
CHAT_ID="SEU_CHAT_ID"

curl -s -X POST \
  "https://api.telegram.org/bot$BOT_TOKEN/sendMessage" \
  -d "chat_id=$CHAT_ID" \
  -d "text=📞 Teste do bot Zabbix - $(date)"
```

Resultado esperado: Mensagem no grupo/chat do Telegram 

Telegram - Configurar no Zabbix

Alerts → Media types → Create media type

Name: Telegram
Type: Webhook
Script: (usar template oficial Zabbix)


Parameters:


- bot_token: {\$TELEGRAM_BOT_TOKEN}
- chat_id: {\$TELEGRAM_CHAT_ID}
- parse_mode: Markdown
- disable_notification: false


Macros globais (Data collection → Hosts → Macros):


- {\$TELEGRAM_BOT_TOKEN} = seu token
- {\$TELEGRAM_CHAT_ID} = seu chat ID

Template de Mensagem Telegram


 ****Problema detectado**** no host {HOST.NAME}


 ****Hora:**** {EVENT.TIME} em {EVENT.DATE}

 ****Problema:**** {EVENT.NAME}

 ****Severidade:**** {EVENT.SEVERITY}

 ****Valor Atual:**** {ITEM.LASTVALUE}

 ****Link:**** {\$ZABBIX_URL}/tr_events.php?triggerid={TRIGGER.ID}

 ****Tags:**** {EVENT.TAGS}

Resultado: Mensagem formatada com emojis e links clicáveis!

PARTE 3

Templates de Notificação Avançados

Templates Baseados em Severidade

Objetivo: Notificações diferentes por severidade

Disaster/High:

● CRÍTICO - AÇÃO IMEDIATA REQUERIDA

Host: {HOST.NAME}

Problema: {EVENT.NAME}

Hora: {EVENT.TIME} ({EVENT.AGE})

Valor atual: {ITEM.LASTVALUE}

Threshold: {TRIGGER.EXPRESSION}

PROCEDIMENTO:

1. Verificar logs: /var/log/app.log
2. Verificar processos: systemctl status app
3. Escalar se não resolver em 15min

Dashboard: {\$ZABBIX_URL}/zabbix.php?action=dashboard.view

Templates Baseados em Severidade (cont.)

Warning/Average:

⚠ Atenção - {EVENT.NAME}

Host: {HOST.NAME}

Severidade: {EVENT.SEVERITY}

Valor: {ITEM.LASTVALUE}

Monitorar e investigar se persistir.

Information:

ℹ INFO: {EVENT.NAME} em {HOST.NAME}

Valor: {ITEM.LASTVALUE}



Resultado: Equipe sabe prioridade instantaneamente!

Templates de Recuperação

Recovery Message (Problema resolvido):

✓ RESOLVIDO: {EVENT.NAME}

Host: {HOST.NAME}

Problema: {EVENT.RECOVERY.NAME}

Duração: {EVENT.DURATION}

Resolvido em: {EVENT.RECOVERY.TIME} {EVENT.RECOVERY.DATE}

Status atual: {ITEM.LASTVALUE}

🎉 Sistema operando normalmente!

Importância:

- Confirma resolução

- Mostra duração do incidente

Template para Status Report

Mensagem periódica (ex: todo dia 8am):

 RELATÓRIO DIÁRIO - {DATE}

Hosts monitorados: {?func=count(host.get({"status":0}))}

Hosts com problema: {?func=count(problem.get({"recent":true}))}

⚠ Top 5 Problemas Ativos:


{?func=foreach(problem.get({"limit":5}), "- {HOST.NAME}: {EVENT.NAME}\n")}

🔥 Severidades:

- Disaster: {?func=count(problem.get({"severities":"5"}))}

- High: {?func=count(problem.get({"severities":"4"}))}

- Average: {?func=count(problem.get({"severities":"3"}))}

 Disponibilidade últimas 24h: 99.8%

Macros Úteis para Templates

Macro	Descrição	Exemplo
{HOST.NAME}	Nome do host	web-prod-01
{EVENT.NAME}	Nome do problema	High CPU usage
{EVENT.SEVERITY}	Severidade	High
{EVENT.TIME}	Hora do evento	14:35:22
{EVENT.DATE}	Data do evento	2025-01-10

Macros Úteis para Templates (cont.)

Macro	Descrição	Exemplo
<code>{EVENT.AGE}</code>	Idade do problema	15m 30s
<code>{EVENT.DURATION}</code>	Duração	2h 15m
<code>{ITEM.LASTVALUE}</code>	Último valor	95%
<code>{TRIGGER.EXPRESSION}</code>	Expressão	<code>{host:cpu.util.last()}>90</code>

Documentação:

<https://www.zabbix.com/documentation/current/en/manual/appendix/macros>

PARTE 4

Scaling Inteligente

O Que É Scaling?

Scaling = Notificar níveis hierárquicos progressivamente

Cenário:

Problema detectado

↓

0min: Notificar → Plantão (1º nível)

↓

15min: Se não resolvido → Coordenador (2º nível)

↓

30min: Se não resolvido → Gerência (3º nível)

Objetivo: Garantir que problemas críticos sejam endereçados!

Configuração de Scaling

Users → User groups

Grupo: Plantão

- Users: João, Maria, Pedro
- Permissions: Read-write
- **Frontend access: Enabled**

Configuração de Scaling (cont.)

Grupo: Coordenadores

- Users: Ana (coordenadora)
- Permissions: Read-write

Grupo: Gerência

- Users: Carlos (gerente)
- Permissions: Read-write

Action com Scaling

Alerts → Actions → Trigger actions → Create action

Conditions:

```
Trigger severity >= High  
AND  
Host group = Production Servers
```

Operations:

Step 1-1 (delay 0s):

- Send to user group: Plantão
- Message: Template padrão

Action com Scaling (cont.)

Step 2-2 (delay 15m):

- Send to user group: Coordenadores
- Message: " ⚠ Problema persistente após 15min"

Step 3-3 (delay 30m):

- Send to user group: Gerência
- Message: " 🔴 CRÍTICO: Não resolvido após 30min"

Scaling por Horário

Cenário: Plantões diferentes por turno

Action 1: Turno Manhã (8h-16h)

Conditions:

- Event time \geq 08:00 AND \leq 16:00
- Severity \geq Average

Operations:

- Step 1: Plantão Manhã
- Step 2: Supervisor Manhã (após 15min)

Scaling por Horário (cont.)

Action 2: Turno Tarde (16h-24h)

Conditions:

- Event time \geq 16:00 AND \leq 24:00
- Severity \geq Average

Operations:

- Step 1: Plantão Tarde
- Step 2: Supervisor Tarde (após 15min)

Scaling por Horário (cont.)

Action 3: Turno Noite (0h-8h)

Conditions:

- Event time `>= 00:00 AND <= 08:00`
- Severity `>= High` # Apenas alta severidade à noite

Operations:

- Step 1: Plantão Noite
- Step 2: Coordenador ON-CALL (após 10min)
- Step 3: Gerente ON-CALL (após 20min, apenas Disaster)

🌙 **Resultado:** Notificações adequadas ao horário!

PARTE 5

Laboratórios Práticos



Laboratório Prático 1

Objetivo: Configurar Media Type Telegram manualmente

Tarefas:

1. Criar bot Telegram (BotFather)
2. Obter Token e Chat ID
3. Configurar media type no Zabbix:
 - Alerts → Media types → Create media type
 - Type: Webhook
 - Parameters: bot_token, chat_id



Laboratório Prático 1 (cont.)

4. Adicionar media ao usuário:

- Users → Users → [seu user]
- Media tab → Add

5. Testar: Criar trigger de teste e aguardar notificação

Lab 1 - Script Webhook Telegram

JavaScript webhook (simplificado):

```
var params = JSON.parse(value);
var req = new XMLHttpRequest();

var url = 'https://api.telegram.org/bot' +
    params.bot_token + '/sendMessage';

var message = '🚨 *Problema*: ' + params.subject + '\n\n' +
    params.message;

var payload = {
    chat_id: params.chat_id,
    text: message,
    parse_mode: 'Markdown'
};

req.setRequestHeader('Content-Type: application/json');
var response = req.post(url, JSON.stringify(payload));

if (req.getStatus() !== 200) {
    throw 'Telegram API error.' + response;
}
```



Laboratório Prático 2

Objetivo: Sistema de Auto-Remediation com Comando Remoto

Conceito: Zabbix detecta serviço down e executa reinício automático

Cenário: Monitorar múltiplos serviços e reiniciar automaticamente



Laboratório Prático 2 (cont.)

Triggers a criar:

1. Name: Apache is down
Expression: `{web-prod-01:net.tcp.service[http].last()}=0`
Severity: High
Tags: service:apache2
2. Name: MySQL is down
Expression: `{web-prod-01:net.tcp.service[mysql].last()}=0`
Severity: High
Tags: service:mysql
3. Name: SSH is down
Expression: `{web-prod-01:net.tcp.service[ssh].last()}=0`
Severity: High
Tags: service:ssh



Laboratório Prático 2 (cont.)

Action: Auto-restart Services

Conditions:

- Trigger severity \geq High
- Tag name = service

Operations:

1. Send message:

Message: "🔄 Serviço {EVENT.TAGS.service} está down. Iniciando auto-remediation..."

2. Run remote command:

Target: Current host

Type: Custom script

Execute on: Zabbix agent

Commands: /usr/lib/zabbix/alertscripts/restart_service.sh {EVENT.TAGS.service}

Recovery Operations:

- Send message:

Message: "✅ Serviço {EVENT.TAGS.service} voltou ao normal após remediation"

Lab 2 - Arquivos do Laboratório






 **Diretório:** `Labs/Aula_05_Lab2_Auto_Remediation/`

Arquivos disponíveis:

1. `restart_service.sh` - Script de auto-remediation
2. `INSTRUcoes.md` - Instruções passo a passo completas
3. `sudoers_zabbix` - Configuração sudoers
4. `zabbix_agentd_remotecommands.conf` - Config do agent

Lab 2 - Script de Auto-Remediation

Script principal (restart_service.sh):

-  Whitelist de serviços
-  Validação de parâmetros
-  Logging detalhado
-  Verificação de status
-  Tratamento de erros

Lab 2 - Configuração do Zabbix Agent

2. Configurar permissões e sudoers:

```
# Criar script e dar permissão
sudo mkdir -p /usr/lib/zabbix/alertscripts
sudo nano /usr/lib/zabbix/alertscripts/restart_service.sh
# (colar o script acima)
sudo chmod +x /usr/lib/zabbix/alertscripts/restart_service.sh
sudo chown zabbix:zabbix /usr/lib/zabbix/alertscripts/restart_service.sh

# Criar arquivo de log
sudo touch /var/log/zabbix-remediation.log
sudo chown zabbix:zabbix /var/log/zabbix-remediation.log

# Configurar sudoers (permitir zabbix executar systemctl)
sudo visudo -f /etc/sudoers.d/zabbix
```

Lab 2 - Configuração do sudoers

Adicionar ao sudoers:

```
zabbix ALL=(ALL) NOPASSWD: /usr/lib/zabbix/alertscripts/restart_service.sh
```

3. Habilitar comandos remotos no agent:

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

Descomentar/adicionar:

```
EnableRemoteCommands=1  
LogRemoteCommands=1  
AllowRoot=0
```

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

Lab 2 - Testando Auto-Remediation

4. Criar itens de monitoramento:

Data collection → Hosts → web-prod-01 → Items → Create item

Item 1:

Name: Apache service status
Type: Simple check
Key: net.tcp.service[http]
Type of information: Numeric (unsigned)
Update interval: 1m

Item 2:

Name: MySQL service status
Type: Simple check
Key: net.tcp.service[mysql]
Update interval: 1m

5. Testar manualmente:

Segurança em Auto-Remediation

⚠ **CUIDADO:** Comandos remotos são poderosos!

Regras de Segurança:

1. ✅ Whitelist de comandos permitidos
2. ✅ Validação de entrada (evitar injection)
3. ✅ Logging completo (auditoria)
4. ✅ Permissões mínimas (não usar root)
5. ✅ Testes extensivos antes de produção
6. ❌ **NUNCA** comandos destrutivos (rm, drop, delete)



Laboratório Prático

Configurar no `zabbix_agentd.conf`:

```
AllowRoot=0  
EnableRemoteCommands=1  
LogRemoteCommands=1
```

PARTE 6

Troubleshooting e Boas Práticas

Troubleshooting - Notificações Não Chegam

Problema 1: Usuário sem media configurado

Verificar:

Users → Users → [user] → Media

Solução: Adicionar media type ao usuário

Problema 2: Media desabilitado ou fora do horário

Verificar:

User → Media → When active: 1-7,00:00-24:00

Solução: Ajustar horários ou severity levels

Problema 3: Action desabilitada

Verificar:

Alerts → Actions → Trigger actions → Status: Enabled

Solução: Habilitar action

Problema 4: Condições da action não atendem

Debug:

Monitoring → Problems → Click problem → Messages

Se não houver mensagem, action não executou.

Verificar:

- Condições da action
- Severity do trigger
- Host group

Troubleshooting - Logs

Logs do Zabbix Server:

```
# Ver logs de alertas  
tail -f /var/log/zabbix/zabbix_server.log | grep -i alert  
  
# Ver erros de media  
grep -i "media.*error" /var/log/zabbix/zabbix_server.log  
  
# Ver execução de comandos remotos  
grep -i "remote command" /var/log/zabbix/zabbix_server.log
```

Aumentar verbosidade (zabbix_server.conf):

```
LogLevel=4 # 0-5, maior = mais verbose
```

Performance - Rate Limiting

Problema: Muitas notificações simultâneas

Solução 1: Media type throttling

```
# No media type Telegram
max_sessions: 10 # Max conexões simultâneas
max_attempts: 3  # Tentativas antes de falhar
timeout: 10s     # Timeout por request
```

Solução 2: Action throttling

```
# Na action
operations:
- default_operation_step_duration: 60s
- pause_operations_for_suppressed_problems: enabled
```

Performance - Agrupamento de Alertas

Problema: 100 hosts com CPU alta → 100 notificações

Solução: Summary message

```
🚨 ALERTA EM MASSA - {EVENT.DATE} {EVENT.TIME}

{?func=summary(problem.get({"severities":"4,5"}), "group")}

Total de problemas: {?func=count(problem.get({"recent":true}))}

Detalhes: {$ZABBIX_URL}/zabbix.php?action=problem.view
```

Configurar na action:

- Operations → Custom message
- Delay entre mensagens: 5 minutos
- Agrupa múltiplos problemas em uma notificação

Segurança - Validação de Comandos

Nunca confie em input direto!

✗ Ruim:

```
# Vulnerável a command injection  
systemctl restart $1
```

Segurança - Validação de Comandos (cont.)

✓ Bom:

```
#!/bin/bash
# Whitelist de serviços permitidos
ALLOWED_SERVICES="apache2 nginx mysql postgresql"

SERVICE=$1

if [[ ! " $ALLOWED_SERVICES " =~ " $SERVICE " ]]; then
    echo "Serviço não permitido: $SERVICE"
    exit 1
fi

# Sanitizar input
SERVICE=$(echo "$SERVICE" | tr -cd '[:alnum:]')

systemctl restart "$SERVICE"
```

Segurança - Criptografia de Credenciais

Problema: Tokens em texto plano nas macros

Solução: Zabbix Vault

```
# Configurar Hashicorp Vault
# zabbix_server.conf

VaultURL=https://vault.empresa.com:8200
VaultToken=s.xxxxxxx
VaultDBPath=kv/zabbix

# Usar macros especiais
{$VAULT:telegram/bot_token}
{$VAULT:slack/webhook_url}
```

Alternativa: Variáveis de ambiente no servidor

Recursos Úteis

Documentação oficial:

 Actions:

<https://www.zabbix.com/documentation/current/en/manual/config/notifications/action>

 Media types:

<https://www.zabbix.com/documentation/current/en/manual/config/notifications/media>

 Macros:

<https://www.zabbix.com/documentation/current/en/manual/appendix/macros>

Templates prontos:

 Telegram:



<https://git.zabbix.com/projects/ZBX/repos/zabbix/browse/templates/media/telegr>

Recap dos Principais Conceitos

- ✓ **Actions** = Automação de resposta (notificar + executar)
- ✓ **Media Types** = Canais de comunicação (Telegram, Slack, etc.)
- ✓ **Condições** = Filtros inteligentes (severidade, horário, grupo)
- ✓ **Templates** = Mensagens customizadas por contexto
- ✓ **Scaling** = Notificação hierárquica progressiva
- ✓ **Auto-remediation** = Execução automática de correções
- ✓ **Segurança** = Validação, whitelist, auditoria
- ✓ **Monitoring** = Monitorar o próprio sistema de alertas

Mensagem-chave: Alertas inteligentes salvam vidas (e noites de sono)! ^{zz}Z

Comparação Final

Aspecto	 Sem Actions	 Com Actions
Notificações	Problemas detectados mas ninguém notificado	Notificações instantâneas
Resolução	Manual e lenta	Auto-remediation para problemas comuns
Escalamento	Caótico (ligações, emails ad-hoc)	Automático e ordenado
Auditoria	Sem histórico de notificações	Auditoria completa
Criticidade	Problemas críticos passam despercebidos	SLA garantido

 **Fim da Aula 05!**

Próxima aula:
Supressão e Correlação de Incidentes