Zabbix Advanced

Aula 06: Supressão e Correlação de Incidentes

4Linux - Curso Avançado

Agenda do Dia

1. **Fundamentos da Supressão de Eventos**
    - Conceitos, tipos, impactos
2. **Event Correlation - Fundamentos**
    - Correlação temporal, por tags, por padrões
3. **Supressão Durante Manutenção**
    - Planejamento, boas práticas

Agenda do Dia (cont.)

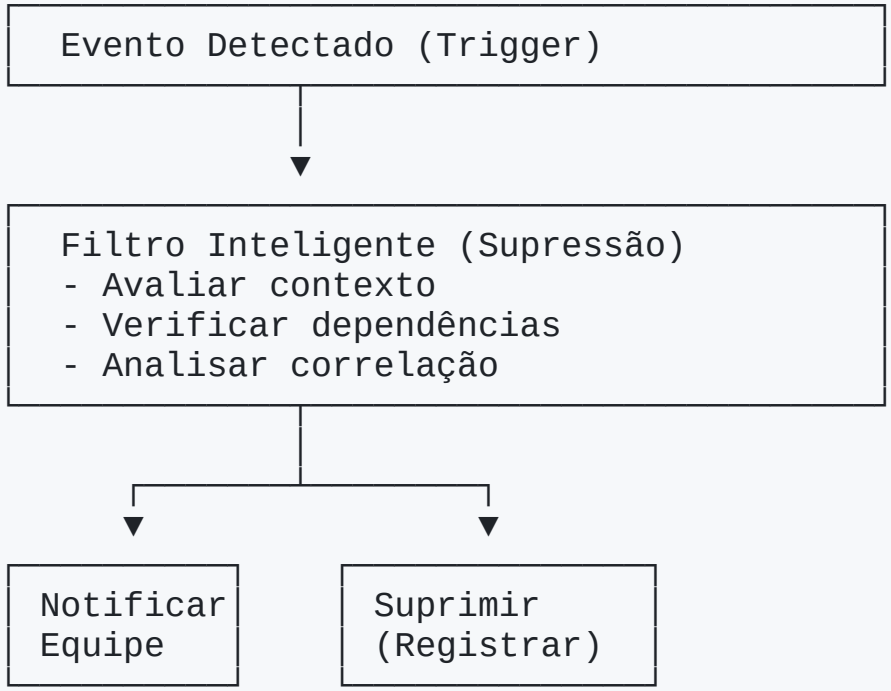
4. **Laboratórios Práticos**
    - Event Correlation via GUI
    - Maintenance Periods
    - Validação de eficácia
5. **Troubleshooting e Métricas**
    - Problemas comuns, indicadores de sucesso

PARTE 1

Fundamentos da Supressão de Eventos

O Que É Supressão de Eventos?

Supressão = Controle de **quando e como** alertas são notificados



Eventos continuam registrados, mas não geram notificações desnecessárias

Cenário Real - O Problema

Sem supressão:

```
Switch core fica offline ➜

  → 1 alerta: Switch core unreachable
  → 50 alertas: Servidores conectados unreachable
  → 200 alertas: Serviços nestes servidores down
  → 100 alertas: Websites inacessíveis
  ──────────────────────────────────────────────
  = 351 notificações para 1 problema! 🔥
```

Cenário Real - O Problema (cont.)

Com supressão:

```
Switch core fica offline ➜

  → 1 alerta: Switch core unreachable
  → 350 alertas suprimidos ✅
  = Foco no problema real!
```

Por Que Supressão É Importante?

Problemas causados por falta de supressão:

- 🔴 **Fadiga de alertas**: Operadores ignoram notificações
- ⏱️ **Tempo de resposta aumentado**: Triagem de alertas redundantes
- 💰 **Custo operacional**: Horas desperdiçadas
- 📉 **Impacto no SLA**: Demora em identificar causa raiz

Benefícios de Supressão

Benefícios da supressão bem implementada:

- ✅ **Redução de 60-80%** no volume de notificações
- ✅ **MTTR reduzido em 40-50%**: Identificação mais rápida
- ✅ **Melhoria na qualidade de vida** da equipe
- ✅ **ROI mensurável**: 2-3 horas/dia economizadas (ambientes 500+ hosts)

Tipos de Supressão no Zabbix

1. Supressão por Dependência (Trigger Dependencies)

- Relacionamentos hierárquicos entre hosts/serviços
- Trigger A depende de Trigger B → suprime A quando B está em problema

2. Supressão por Manutenção (Maintenance Periods)

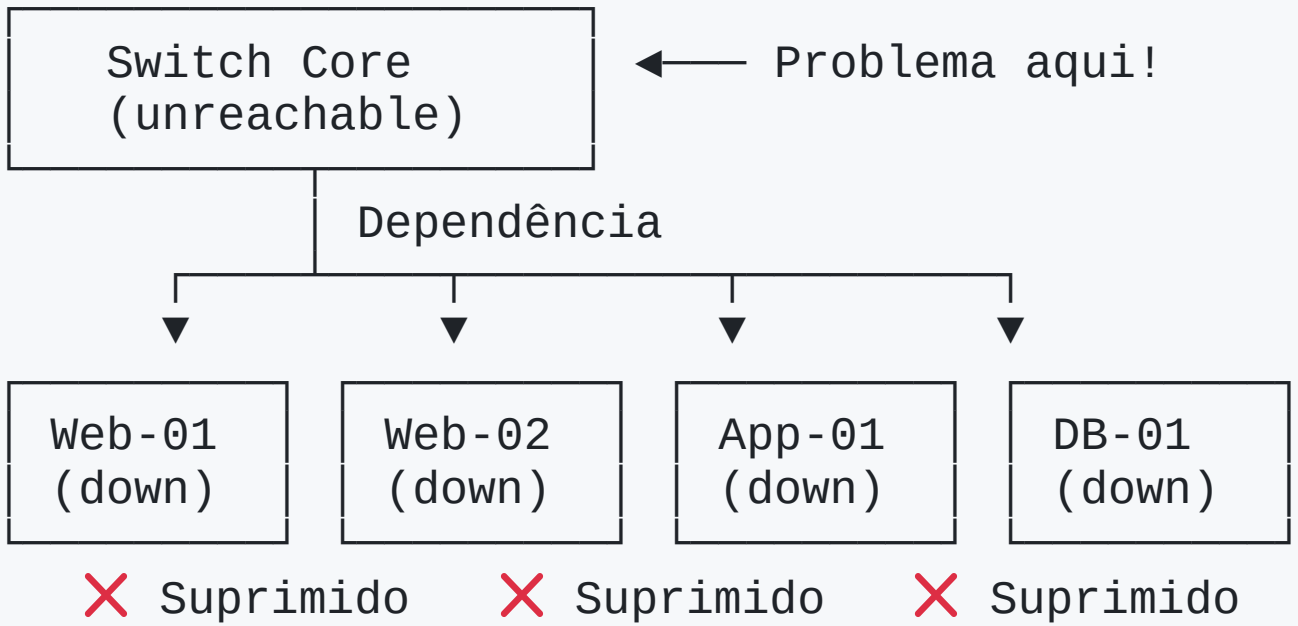
- Janelas programadas de trabalho
- Evita alertas durante atualizações/patches

3. Supressão por Correlação (Event Correlation)

- Regras lógicas complexas
- Analisa múltiplos eventos em tempo real
- Decisões inteligentes sobre notificações

Supressão por Dependência

Como funciona:



Quando Usar Dependência?

Quando usar:

- Infraestrutura com dependências claras (rede, storage, DB)
- Arquiteturas hierárquicas (LB → Web → App → DB)
- Serviços compartilhados (DNS, authentication)

Configurando Dependência

Data collection → Hosts → [host] → Triggers → [trigger] → Dependencies

```
# Exemplo conceitual

Host: servidor-web-01
Trigger: "HTTP service is down"

Dependencies (Add):
  1. Host: switch-core-01
     Trigger: "Network device is unreachable"

  2. Host: firewall-01
     Trigger: "Firewall is not responding"
```

Resultado:

- Se switch ou firewall caírem, alerta HTTP é suprimido
- Equipe recebe apenas alerta do switch/firewall (causa raiz)

Supressão por Manutenção

Maintenance Periods = Janelas programadas de supressão

Tipos:

1. **With data collection**
    - Suprime notificações
    - **Continua coletando dados**
    - Host online com problemas esperados
2. **Without data collection**
    - **Para completamente** a coleta
    - Host estará desligado/offline

Quando Usar Manutenção

Cenário	Tipo	Duração
Security patches	With data	2-4h
OS updates	With data	3-6h
Hardware replacement	Without data	1-8h
App deployment	With data	30min-2h
Database optimization	With data	2-4h
Network maintenance	With data	2-3h
DR test	With data	4-8h
Ambiente dev (contínuo)	With data	Permanente

Boas Práticas - Manutenção

✅ **Duração apropriada:**

- 2-4 horas para manutenção típica
- Adicionar buffer de 25-50%
- Evitar janelas de 12-24h

✅ **Timing correto:**

- Fora de horário comercial
- Evitar janelas de backup (01:00-03:00)
- Evitar fechamento mensal (dias 28-3)
- Evitar picos de negócio (9-11h, 14-16h)

Boas Práticas - Manutenção (cont.)

✅ Escopo específico:

```
# ❌ EVITAR
scope: ALL_HOSTS

# ✅ CORRETO
scope: GROUP_web_servers
hosts: ["web-prod-01", "web-prod-02"]
tags: ["application:web-portal", "environment:production"]
```

Boas Práticas - Manutenção (cont.)

✅ **Notificação prévia:**

- 7 dias antes: stakeholders, management
- 48 horas antes: ops, dev, support teams
- 24 horas antes: todos os usuários afetados
- 1 hora antes: ops-team (confirmação)
- Ao completar: todos previamente notificados

✅ **Exceções críticas:**

- **NUNCA** suprimir eventos de segurança
- **NUNCA** suprimir severity "Disaster"
- Configurar exceções explícitas

Supressão por Correlação

Event Correlation = Identificar relacionamentos entre eventos

Tipos de correlação:

1. **Correlação Temporal**
    - Eventos próximos no tempo (ex: múltiplos alertas CPU em 5 min)
2. **Correlação por Tags**
    - Eventos com tags relacionadas (ex: mesma aplicação, mesmo datacenter)

Quando Usar Correlação

3. **Correlação por Padrão**
    - Eventos que seguem sequência conhecida (ex: DB slow → App timeout → User complaints)
4. **Correlação por Dependência**
    - Similar ao trigger dependency, mas mais flexível

Quando Usar Correlação

✅ Quando usar Event Correlation:

- Ambientes complexos com **>100 hosts**
- Necessidade de agrupar eventos relacionados
- Identificação automática de root cause
- Redução de ruído em arquiteturas distribuídas
- Microserviços e containers (eventos dispersos)

Exemplo:

```
10 alertas "CPU alta" no mesmo cluster em 5 minutos
   ↓ Correlação
1 evento agregado "Cluster overload"
```

Tabela de Decisão - Quando Suprimir

Cenário	Usar Supressão?	Tipo Recomendado	Observações
Manutenção programada	✅ Sim	Maintenance Period	2-4h, escopo específico
Falha em cascata de rede	✅ Sim	Dependencies ou Correlation	Suprimir downstream
Evento de segurança	❌ **NÃO**	**Nenhum**	**NUNCA suprimir**
Severity "Disaster"	❌ **NÃO**	**Nenhum**	**Sempre notificar**
Ambiente dev	✅ Sim	Maintenance	Contínuo
Deploy de aplicação	✅ Sim	Maintenance	Janela conhecida
Testes de carga	✅ Sim	Maintenance	Evitar alertas perf
Performance correlacionada	✅ Sim	Event Correlation	CPU+Mem+Disk
Horário comercial crítico	❌ Não	Nenhum	Visibilidade total

Impactos da Supressão Mal Configurada

Problema 1: Over-Suppression (Supressão Excessiva)

Sintomas:

- 🔴 Problemas críticos não são notificados
- 🔴 Incidentes descobertos apenas quando usuários reclamam
- 🔴 Métricas de disponibilidade não refletem realidade

Exemplo ERRADO:

```
maintenance_period:
  scope: ALL_HOSTS                  # ❌ Muito abrangente!
  duration: 48_HOURS                # ❌ Muito longa!
  type: without_data_collection # ❌ Para tudo!
```

Corrigindo Over-Suppression

Configuração CORRETA:

```
maintenance_period:
  name: "Web Servers Patching"
  scope: GROUP_web_servers      # ✅ Específico
  duration: 3_HOURS             # ✅ Apropriado
  type: with_data_collection    # ✅ Mantém dados

  exceptions:
    - security_triggers         # ✅ Exceção importante
    - severity: Disaster        # ✅ Sempre notificar
    - tags:
        - "business_critical:true"
```

Como evitar:

- Escopo específico (hosts, grupos ou tags)
- Duração limitada (2-4h típico)
- Exceções para eventos críticos
- Revisar periodicamente manutenções ativas

Under-Suppression

Problema 2: Supressão Insuficiente

Sintomas:

- Equipe recebe centenas de alertas para um único problema
- Fadiga de alertas (equipe ignora notificações)
- Tempo desperdiçado triando alertas redundantes

Solução:

```
# Configurar dependency chain
network_dependencies:
  core_switch:
    trigger: "Network device is unreachable"
    suppresses:
      - all_connected_hosts      # 50 servidores
      - all_services_on_hosts    # 200 serviços
      - all_websites             # 100 websites

# Resultado:
#   1 alerta: Switch core
#   350 alertas suprimidos ✅
```

Timing Incorreto

Problema 3: Janelas que conflitam com operações críticas

Períodos críticos a EVITAR:

Período	Quando	Impacto
Fechamento mensal	Dias 28-3	VERY HIGH
Janelas de backup	01:00-03:00	MEDIUM
Picos de negócio	9-11h, 14-16h	HIGH
Black Friday	Nov 25 - Dez 2	VERY HIGH
Natal/Ano Novo	Dez 15-31	VERY HIGH

Checklist - Planejamento de Manutenção

Antes de criar uma janela de manutenção, verificar:

- [ ] Não conflita com horário comercial crítico
- [ ] Não conflita com janela de backup
- [ ] Não conflita com fechamento mensal
- [ ] Não conflita com eventos de negócio (Black Friday, etc)
- [ ] Equipe foi notificada com 24-48h de antecedência
- [ ] Plano de rollback está definido
- [ ] Duração é realista (inclui margem de segurança 25-50%)
- [ ] Escopo está claramente definido
- [ ] Exceções para eventos críticos estão configuradas
- [ ] Aprovação formal obtida (ticket/change request)

PARTE 2

Event Correlation - Fundamentos

O Que É Event Correlation?

Event Correlation = Identificar relacionamentos entre eventos

Analogia de trânsito:

- Um semáforo quebrado → Congestionamento em 1 rua
- Vários semáforos quebrados na mesma avenida → Problema na central de controle
- Correlação identifica que não são 5 problemas, mas 1 só (central)

Event Correlation

No Zabbix permite:

- **Agrupar eventos relacionados**: 10 alertas CPU → 1 evento "Cluster overload"
- **Identificar root cause**: DB slow + App timeout + User complaints → "DB performance issue"
- **Reduzir ruído**: Falha de rede + 50 hosts down → Alertar apenas rede
- **Ações context-aware**: Diferentes ações baseadas no padrão

Impacto Mensurável

Comparação: Com vs Sem Correlação

Métrica	Sem Correlação	Com Correlação	Melhoria
Alertas por dia	200-300	40-60	**-75%**
Tempo p/ root cause	15-30 min	2-5 min	**-80%**
Falsos positivos	30-40%	5-10%	**-75%**
Satisfação equipe	Baixa	Alta	↑↑↑
MTTR (Mean Time To Repair)	45 min	20 min	**-55%**

ROI: Ambiente com 500 hosts economiza:

- 2-3 horas/dia de trabalho operacional
- 10-15 horas/mês de análise de incidentes
- Redução de 60-80% em notificações desnecessárias

Correlação Temporal

Eventos próximos no tempo são frequentemente relacionados

Janelas temporais recomendadas:

Tipo de Problema	Janela	Razão
Rede	2-10 min	Falhas propagam rapidamente
Performance (CPU/Mem)	5-15 min	Degradação gradual
Segurança	30-60 min	Ataques coordenados espaçados
Aplicações	10-20 min	Dependências entre serviços

Exemplo:

```
5 servidores com CPU > 90% dentro de 5 minutos
  ↓ Correlação temporal
Provável causa comum: Carga de trabalho, DDoS, job agendado
```

Correlação por Tags

Tags permitem agrupar eventos relacionados

Taxonomia de tags úteis:

```
tags:
  component: [cpu, memory, disk, network, application]
  location: [datacenter-sp, datacenter-rj, aws-us-east]
  environment: [production, staging, development]
  tier: [frontend, backend, database, cache]
  application: [api-auth, api-payment, web-portal]
  severity: [low, medium, high, disaster]
  business_unit: [sales, finance, operations]
  criticality: [critical, high, medium, low]
```

Correlação por Tags (cont.)

Exemplo de regra:

```
correlation_rule:
  name: "Backend Performance Correlation"
  conditions:
    - new_event_tag: "tier" equals "backend"
    - new_event_tag: "component" equals "cpu"
    - time_window: 10_minutes
  action: create_composite_event
```

Correlação por Padrões

Padrões comuns de problemas:

Padrão	Sequência de Eventos	Root Cause
Database Cascade	DB slow → App timeout → HTTP 500	Database performance
Network Failure	Switch down → Hosts unreachable → Services unavailable	Network infrastructure
Resource Exhaustion	Disk 80% → 90% → 95% → App crash	Disk space management
Memory Leak	Memory 60% → 80% → 95% → OOM killer	Application memory leak
DDoS Attack	Traffic spike → CPU high → Connection refused	Network attack

Benefício: Identificação automática de causa raiz

Exemplo de Padrão - Database Cascade

```
correlation_pattern:
  name: "Database Performance Cascade"

  pattern_sequence:
    - event: "Database query time > 5s"
      max_age: 5min

    - event: "Application response time > 10s"
      max_age: 3min

    - event: "HTTP 500 errors increasing"
      max_age: 2min

  confidence_threshold: 80%

  actions:
    - suppress_downstream_events: true
    - create_incident:
        severity: high
        assigned_to: database-team
        title: "Database Performance Cascade Detected"
    - run_diagnostic_script: "/usr/local/bin/diagnose_db_performance.sh"
    - notify_escalation:
        level: 2
        delay: 10min
```

Configurando Correlação via GUI

Administration → Event correlation → Create correlation

Passo 1: Configurações Básicas

```
Name: Network Infrastructure Cascade
Description: Suprime hosts quando switch core falha
Status: Enabled
```

Passo 2: Definir Conditions

Tipos de condições disponíveis:

1. **Old event tag** - Tag do evento anterior
2. **New event tag** - Tag do novo evento
3. **New event host group** - Grupo do host
4. **Event tag pair** - Par de tags relacionadas
5. **Old/New event tag value** - Valores de tags

Conditions - Exemplo Prático

Cenário: Suprimir hosts quando switch core falha

```
Conditions (type: AND/OR):

  Condition 1:
    Type: Old event tag
    Tag: component
    Operator: equals
    Value: network-switch

  Condition 2:
    Type: New event tag
    Tag: component
    Operator: equals
    Value: host

  Condition 3:
    Type: New event host group
    Host group: Servers connected to switch-core-01
    Operator: equals
```

Operations - Ações da Correlação

Passo 3: Configurar Operations

Opções disponíveis:

1. **Close old event**
    - Novo evento é mais importante
    - Fecha o evento anterior
2. **Close new event**
    - Novo evento é redundante
    - Suprime o novo (mantém o antigo)

3. **Ambas as opções**
    - Fecha ambos
    - Cria evento agregado (opcional)

Exemplo Completo - GUI

```
Name: "Suppress Hosts When Switch Fails"

Conditions (AND):
  1. Old event tag "component" equals "network-switch"
  2. Old event tag "severity" >= "high"
  3. New event tag "component" equals "host"
  4. New event host group equals "Servers-Switch-Core-01"
  5. Time window: 10 minutes

Operations:
  - Close new event

Result:
  - Switch core unreachable → ✅ Notificado
  - 50 servidores unreachable → ❌ Suprimidos (new events closed)
  - Equipe recebe apenas 1 alerta
```

Boas Práticas - Event Correlation

1. Evolução Gradual:

```
Fase 1 (Semana 1-2):
  - 2-3 correlações básicas
  - Network failures, cascading services

Fase 2 (Semana 3-4):
  - Correlações temporais
  - Performance patterns

Fase 3 (Mês 2):
  - Padrões específicos do ambiente
  - Aplicações críticas

Fase 4 (Mês 3+):
  - Otimização e refinamento
  - Machine Learning (futuro)
```

Boas Práticas (cont.)

2. Taxonomia de Tags Consistente:

```
# Tags obrigatórias em TODOS os triggers
required_tags:
  - component: [cpu, memory, disk, network, app]
  - environment: [production, staging, dev]
  - tier: [frontend, backend, database]

# Tags opcionais
optional_tags:
  - location, business_unit, criticality, owner
```

Boas Práticas (cont.)

3. Limitar Número de Correlações:

- Máximo **15-20 correlações** ativas
- Performance do Zabbix Server degradada com >30
- Priorizar correlações com maior impacto

Boas Práticas (cont.)

4. Monitorar Eficácia:

```
metrics_to_track:
  - reduction_rate: "> 60%"
    # (Alertas antes - Alertas depois) / Alertas antes

  - false_positive_rate: "< 5%"
    # Eventos suprimidos incorretamente

  - time_to_root_cause: "< 5 minutos"
    # Tempo para identificar causa raiz

  - alert_fatigue_index: "< 20 alertas/operador/dia"
    # Carga por operador
```

Boas Práticas (cont.)

5. Documentar Cada Regra:

- Purpose, Trigger conditions, Action taken
- Justification, Expected reduction, Owner
- Review frequency (trimestral)

PARTE 3

Supressão Durante Manutenção Programada

Planejamento de Janelas de Manutenção

4 Dimensões Críticas:

1. **Timing** - Quando executar
2. **Duração** - Quanto tempo
3. **Escopo** - O que será afetado
4. **Comunicação** - Quem notificar

Objetivo:

- Minimizar impacto no negócio
- Evitar conflitos com operações críticas
- Transparência para stakeholders

Timing - Quando Executar

Horários recomendados:

```
✅ IDEAL:
  days: [Saturday, Sunday]
  hours: [02:00-06:00]
  reasoning: "Mínimo impacto, baixo uso"

✅ ACEITÁVEL:
  days: [Tuesday, Wednesday, Thursday]
  hours: [22:00-02:00]
  reasoning: "Baixo uso, evita início/fim de semana"

❌ EVITAR:
  days: [Monday, Friday]
  hours: [06:00-22:00]
  reasoning: "Alto impacto, picos de uso"
```

Conflitos a Evitar

Períodos críticos:

Período	Quando	Razão
Janelas de backup	01:00-03:00	Sobrecarga I/O
Fechamento mensal	Dias 28-3	Processos contábeis
Horário de pico	10-11h, 14-16h	Alto uso
Black Friday	Nov 25 - Dez 2	Evento crítico
Natal/Ano Novo	Dez 15-31	Evento crítico
Início/Fim trimestre	Últimos 3 dias	Relatórios

Checklist: Validar com calendário de negócio antes de agendar

Duração - Quanto Tempo

Regra de buffer:

```
best_practice:
  rule: "Adicionar 25-50% buffer"

  example:
    work_planned: "2 hours"
    buffer: "+1 hour"
    total_window: "3 hours"

  reasoning:
    - Problemas inesperados
    - Rollback se necessário
    - Validação pós-manutenção
    - Margem de segurança
```

Duração - Quanto Tempo (cont.)

Duração máxima recomendada:

- Manutenção típica: 4 horas
- Manutenção complexa: 8 horas
- Evitar: >12 horas (risco de over-suppression)

Escopo - O Que Será Afetado

```
# ❌ Muito amplo - EVITAR
bad_scope:
  hosts: "ALL"
  impact: "TODO o monitoramento suprimido!"

# ✅ Específico por Host Groups
good_scope_1:
  host_groups: ["Web Servers - Production"]
  impact: "Apenas web servers"

# ✅ Específico por Hosts
good_scope_2:
  hosts: ["web-prod-01", "web-prod-02", "web-prod-03"]
  impact: "3 hosts específicos"

# ✅ Específico por Tags (mais flexível)
good_scope_3:
  tags:
    - "application:web-portal"
    - "environment:production"
    - "datacenter:sp"
  impact: "Hosts que combinam todas as tags"
```

Comunicação - Timeline

Quem notificar e quando:

Quando	Quem	Conteúdo
7 dias antes	Stakeholders, Management	Aviso formal, aprovação
48 horas antes	Ops, Dev, Support teams	Detalhes técnicos, preparação
24 horas antes	Todos os usuários afetados	Aviso de downtime esperado
1 hora antes	Ops-team	Confirmação final, go/no-go
Início	Ops-team	Manutenção iniciada
Fim	Todos previamente notificados	Conclusão, status

Template de comunicação:

- O que será feito, quando, duração esperada
- Sistemas afetados, impacto esperado
- Contato para emergências

Criando Maintenance Period - GUI

Data collection → Maintenance → Create maintenance period

Passo 1: Informações Básicas

Name: `[Scope] - [Type] - [Date]`

- Exemplo: `WebServers-Prod - Security Patching - 2025-02-10`

Description: Incluir informações completas

- **Purpose:** Security patches (CVE-2025-xxxx)
- **Expected downtime:** 3 hours
- **Systems affected:** web-prod-01, web-prod-02, web-prod-03
- **Impact:** Web portal may be temporarily unavailable
- **Rollback plan:** Rollback snapshots available

Criando Maintenance Period (cont.)

Passo 2: Tipo de Manutenção

Maintenance type:

- **With data collection**
  - ✅ Host online, problemas esperados
  - ✅ Mantém histórico de dados
  - ✅ Usar para: Patches, updates, restarts
- **Without data collection**
  - ✅ Host completamente offline
  - ✅ Economiza recursos do Zabbix
  - ✅ Usar para: Hardware replacement, host desligado

Criando Maintenance Period (cont.)

Passo 3: Definir Período

Opção 1: One time only (uma vez)

- **Active since:** 2025-02-10 02:00:00
- **Active till:** 2025-02-10 05:00:00 (3 horas)

Opção 2: Daily (diário)

- **Every N days:** 1
- **Start time:** 02:00
- **Period:** 3h

Opção 3: Weekly (semanal)

- **Days:** Saturday
- **Start time:** 02:00
- **Period:** 4h

Opção 4: Monthly (mensal)

- **Day of month:** 1st Saturday (primeiro sábado)
- **Start time:** 02:00
- **Period:** 4h

Criando Maintenance Period (cont.)

Passo 4: Definir Escopo (Hosts and groups tab)

Método 1: Por Host Groups

- Click em **Add** (Host groups)
- Selecionar: `Web Servers - Production`

Método 2: Por Hosts Específicos

- Click em **Add** (Hosts)
- Selecionar: `web-prod-01`, `web-prod-02`, `web-prod-03`

Método 3: Por Tags (mais flexível ✅)

- **Tag name:** `application`
  Operator: `equals`
  Tag value: `web-portal`
- Click **AND**
- **Tag name:** `environment`
  Operator: `equals`
  Tag value: `production`

Dica: Tags são mais flexíveis (hosts adicionados automaticamente)

Validação Pós-Manutenção

Checklist após completar manutenção:

- [ ] ✅ **Verificar serviços online**

```
systemctl status apache2 mysql
```

- [ ] ✅ **Testar conectividade**

```
curl -I https://portal.empresa.com
```

- [ ] ✅ **Verificar logs do Zabbix**

```
tail -100 /var/log/zabbix/zabbix_agentd.log
```

- [ ] ✅ **Confirmar métricas normais** (Dashboard do Zabbix)
- [ ] ✅ **Encerrar maintenance period** (se não foi automático)
- [ ] ✅ **Notificar conclusão** (email para stakeholders)

Post-Mortem - Documentação

Após cada manutenção, documentar:

Planejado vs Real:

- Duração planejada: 3h
- Duração real: 2h 45min ✅

Issues encontrados:

- Web-prod-02 não reiniciou automaticamente
- Necessário restart manual do serviço

Resoluções aplicadas:

- Ajustado systemd service para auto-restart

Action items:

- [ ] Criar script de validação pré-manutenção
- [ ] Atualizar runbook com problema encontrado

Lições aprendidas:

- Buffer de 1h foi suficiente
- Horário 02:00 adequado (zero usuários online)

PARTE 4

Laboratórios Práticos

🔬 Laboratório Prático 1

Objetivo: Configurar Event Correlation via GUI

Tempo: 20 minutos

Cenário:

- Switch core: `switch-core-01`
- 20 servidores web conectados: `web-prod-01` a `web-prod-20`
- Problema: Switch fica offline → 20 alertas (switch + 20 servidores)
- Meta: Apenas 1 alerta (switch) deve ser enviado

Resultado esperado:

- Switch unreachable → ✅ Notificado
- 20 servidores unreachable → ❌ Suprimidos

Lab 1 - Passo 1: Preparação

1. Verificar tags nos triggers:

```
Data collection → Hosts → switch-core-01 → Triggers
  → "Network device is unreachable"
  → Tags: Adicionar tag "component:network-switch"

Data collection → Hosts → web-prod-XX → Triggers
  → "Host is unreachable"
  → Tags: Adicionar tag "component:host"
```

2. Verificar host groups:

```
Data collection → Host groups
  → Verificar grupo: "Web Servers - Production"
  → Members: web-prod-01 até web-prod-20
```

Lab 1 - Passo 2: Criar Correlação

Administration → Event correlation → Create correlation

```
Name: Suppress Hosts When Switch Fails
Description: Suprime hosts quando switch core fica offline

Conditions (type: AND):
  1. Condition type: Old event tag
     Tag: component
     Operator: equals
     Value: network-switch

  2. Condition type: New event tag
     Tag: component
     Operator: equals
     Value: host

  3. Condition type: New event host group
     Host group: Web Servers - Production
     Operator: equals
```

Lab 1 - Passo 3: Configurar Operations

```
Operations:
  ☑ Close new event
    - Eventos de hosts serão fechados (suprimidos)
    - Evento do switch permanece ativo

  ☐ Close old events
    - Não marcar (queremos manter alerta do switch)
```

Click: Add

Status: Enabled ✅

Lab 1 - Passo 4: Validação

Simular falha do switch:

```
# No Zabbix Server, desabilitar switch temporariamente
# Data collection → Hosts → switch-core-01
# Status: Disabled (aguardar 2-3 minutos)
```

Verificar resultado:

```
Monitoring → Problems

Esperado:
  ✅ 1 problema: switch-core-01 - Network device is unreachable
  ❌ 0 problemas: web-prod-XX (suprimidos por correlação!)
```

Reabilitar switch:

```
Data collection → Hosts → switch-core-01 → Status: Enabled
```

🔬 Laboratório Prático 2

Objetivo: Criar Maintenance Period Recorrente

Tempo: 15 minutos

Cenário:

- Patches de segurança toda primeira segunda-feira do mês
- Horário: 02:00-05:00 (3 horas)
- Escopo: Apenas web servers de produção
- Tipo: With data collection (hosts ficam online)

Lab 2 - Passo 1: Criar Maintenance

Data collection → Maintenance → Create maintenance period

```
Name: WebServers-Prod - Monthly Security Patching

Description:
  Purpose: Monthly security patches
  Expected downtime: 3 hours
  Systems affected: Web servers production (all)
  Impact: Web services may experience brief interruptions
  Rollback plan: Automated snapshots before patching
  Contact: ops-team@empresa.com
  Approval: Recurrent maintenance (approved CHG-2025-R001)

Active since: 2025-02-03 02:00:00  # Primeira segunda-feira
Maintenance type: ● With data collection
```

Lab 2 - Passo 2: Configurar Período

```
Periods:
  Period type: • Monthly

  Month: • Every

  Day of month: 1st Monday
    # Primeira segunda-feira de cada mês

  Start time: 02:00

  Period: 3h

  Start date: 2025-02-01
```

Resultado: Manutenção recorrente automática todo mês

Lab 2 - Passo 3: Definir Escopo

Hosts and groups tab:

```
Método 1 - Por Host Group:
  Host groups → Add
    → Web Servers - Production

Método 2 - Por Tags (mais flexível):
  Tags → Add
    Tag name: tier
    Operator: equals
    Tag value: frontend

    AND

    Tag name: environment
    Operator: equals
    Tag value: production
```

Click: Add

Lab 2 - Passo 4: Validação

Verificar manutenção criada:

```
Data collection → Maintenance

Deve aparecer:
  Name: WebServers-Prod - Monthly Security Patching
  Type: Recurring (Monthly)
  Next run: 2025-02-03 02:00:00
  Affected hosts: [lista de web servers]
  Status: Active ✅
```

Testar (opcional):

- Editar maintenance → Mudar Active since para "agora + 2 min"
- Aguardar 2 min
- Verificar: Monitoring → Problems (sem alertas de web servers)

🔬 Laboratório Prático 3

Objetivo: Validar Eficácia de Correlação

Tempo: 25 minutos

Ferramentas necessárias:

- Acesso ao database Zabbix (read-only)
- Python 3.x com psycopg2 ou mysql-connector

Meta: Medir reduction rate de uma correlação implementada

Lab 3 - Passo 1: Coletar Baseline

Query SQL - Alertas antes da correlação:

```sql
-- Contar eventos nos últimos 7 dias
SELECT
    DATE(FROM_UNIXTIME(clock)) as date,
    COUNT(*) as total_events,
    COUNT(DISTINCT hostid) as affected_hosts
FROM events
WHERE source = 0  -- Trigger events
  AND object = 0  -- Trigger object
  AND clock >= UNIX_TIMESTAMP(NOW() - INTERVAL 7 DAY)
GROUP BY DATE(FROM_UNIXTIME(clock))
ORDER BY date DESC;

-- Eventos por host
SELECT
    h.host,
    COUNT(*) as event_count
FROM events e
JOIN triggers t ON e.objectid = t.triggerid
JOIN functions f ON t.triggerid = f.triggerid
JOIN items i ON f.itemid = i.itemid
JOIN hosts h ON i.hostid = h.hostid
WHERE e.clock >= UNIX_TIMESTAMP(NOW() - INTERVAL 7 DAY)
GROUP BY h.host
```

Lab 3 - Passo 2: Identificar Oportunidades

Analisar resultados da query:

```
Exemplo de output:
  switch-core-01: 1 evento
  web-prod-01: 12 eventos
  web-prod-02: 12 eventos
  web-prod-03: 12 eventos
  ...
  web-prod-20: 12 eventos

Total: 241 eventos (1 switch + 20 hosts x 12 cada)
```

Oportunidade identificada:

- Switch + hosts conectados geram alertas simultâneos
- Correlação pode reduzir 240 eventos → 1 evento
- **Reduction rate potencial: 99,5%**

Lab 3 - Passo 3: Implementar Correlação

Implementar correlação do Lab 1:

```
Administration → Event correlation → Create correlation

Name: Suppress Hosts When Switch Fails
Conditions:
  - Old event tag "component" = "network-switch"
  - New event tag "component" = "host"
  - New event host group = "Web Servers - Production"
Operations:
  - Close new event ✅
```

Aguardar eventos naturais ou simular falha para testar

Lab 3 - Passo 4: Comparar Resultados

Query SQL - Eventos correlacionados (suprimidos):

```
-- Eventos fechados por correlação
SELECT
    DATE(FROM_UNIXTIME(e.clock)) as date,
    COUNT(*) as suppressed_events
FROM event_suppress es
JOIN events e ON es.eventid = e.eventid
WHERE e.clock >= UNIX_TIMESTAMP(NOW() - INTERVAL 7 DAY)
GROUP BY DATE(FROM_UNIXTIME(e.clock))
ORDER BY date DESC;

-- Cálculo de reduction rate
SELECT
    (SELECT COUNT(*) FROM events
     WHERE clock >= UNIX_TIMESTAMP(NOW() - INTERVAL 1 DAY)) as total_events,
    (SELECT COUNT(*) FROM event_suppress es
     JOIN events e ON es.eventid = e.eventid
     WHERE e.clock >= UNIX_TIMESTAMP(NOW() - INTERVAL 1 DAY)) as suppressed,
    ROUND(
        (SELECT COUNT(*) FROM event_suppress es
         JOIN events e ON es.eventid = e.eventid
         WHERE e.clock >= UNIX_TIMESTAMP(NOW() - INTERVAL 1 DAY)) * 100.0 /
        (SELECT COUNT(*) FROM events
```

PARTE 5

Troubleshooting e Métricas

Troubleshooting - Correlação Não Funciona

Sintomas: Eventos ainda sendo notificados mesmo com correlação

Diagnóstico:

1. **Verificar se correlação está habilitada:**

```
Administration → Event correlation → [sua correlação]
Status: Enabled? ✅
```

2. **Verificar logs do Zabbix Server:**

```
tail -100 /var/log/zabbix/zabbix_server.log | grep -i correlation
```

3. **Validar conditions (case-sensitive!):**

```
# ❌ ERRADO
Tag: Component  (C maiúsculo)
```

Troubleshooting (cont.)

Soluções:

1. **Corrigir grafia de tags:**
   - Padronizar: tudo minúsculo, traços ao invés de espaços
   - `component:network-switch` não `Component:Network Switch`
2. **Ajustar janela temporal:**

```
# Se eventos estão espaçados por 15 min
time_window: 900  # 15 minutos (não 300/5min)
```

3. **Simplificar conditions:**
   - Começar com 2-3 conditions simples
   - Adicionar complexidade gradualmente

Troubleshooting - Muitos Falsos Positivos

Sintomas: Eventos importantes sendo suprimidos incorretamente

Diagnóstico:

```
-- Query: Eventos suprimidos com severity alta
SELECT
    h.host,
    t.description as trigger_name,
    t.priority as severity,
    FROM_UNIXTIME(e.clock) as event_time
FROM event_suppress es
JOIN events e ON es.eventid = e.eventid
JOIN triggers t ON e.objectid = t.triggerid
JOIN functions f ON t.triggerid = f.triggerid
JOIN items i ON f.itemid = i.itemid
JOIN hosts h ON i.hostid = h.hostid
WHERE t.priority >= 4  -- High or Disaster
  AND e.clock >= UNIX_TIMESTAMP(NOW() - INTERVAL 7 DAY)
ORDER BY e.clock DESC;
```

Troubleshooting - Falsos Positivos (cont.)

Soluções:

1. **Adicionar exceções para alta severidade:**

```
Conditions (add):
  - Old event severity: <= High  # Só suprimir se não for crítico
```

2. **Adicionar exceção para tags críticas:**

```
Conditions (add):
  - New event tag "business_critical" does not equal "true"
```

3. **Revisar conditions da correlação:**
   - Tornar mais específica
   - Adicionar mais filtros (host group, location, etc)

Troubleshooting - Maintenance Não Suprime

Sintomas: Alertas continuam sendo enviados durante manutenção

Diagnóstico:

1. **Verificar escopo:**

```
-- Query: Hosts em manutenção agora
SELECT
    h.host,
    m.name as maintenance_name,
    FROM_UNIXTIME(m.active_since) as start,
    FROM_UNIXTIME(m.active_till) as end
FROM hosts h
JOIN maintenances_hosts mh ON h.hostid = mh.hostid
JOIN maintenances m ON mh.maintenanceid = m.maintenanceid
WHERE m.active_since <= UNIX_TIMESTAMP(NOW())
  AND m.active_till >= UNIX_TIMESTAMP(NOW());
```

Troubleshooting - Maintenance (cont.)

3. **Verificar timezone:**

```
Administration → General → GUI
  Default time zone: (verificar)

User profile → Time zone: (verificar)
```

Problema comum: Timezone diferente → janela não ativa

4. **Verificar actions:**

```
Alerts → Actions → Trigger actions → [sua action]

Operations:
  Conditions: Não há condição que FORCE notificação
              mesmo durante manutenção
```

Troubleshooting - Performance Degradada

Sintomas: Zabbix Server lento após implementar correlações

Diagnóstico:

```
-- Query: Queries lentas relacionadas a correlação
SHOW PROCESSLIST;  -- MySQL
-- ou
SELECT * FROM pg_stat_activity;  -- PostgreSQL

-- Verificar uso de CPU do Zabbix Server
top -u zabbix
```

Causas comuns:

- Muitas correlações (>30)
- Janelas temporais muito longas
- Condições muito complexas

Troubleshooting - Performance (cont.)

Soluções:

1. **Reduzir número de correlações:**
   - Máximo recomendado: 15-20
   - Priorizar correlações com maior impacto
2. **Otimizar conditions:**
   - Usar host groups ao invés de tags (mais rápido)
   - Simplificar lógica (menos conditions)
3. **Adicionar índices no banco:**

```
-- PostgreSQL
CREATE INDEX idx_events_clock_source
  ON events(clock, source);
```

Métricas de Sucesso

Operacionais:

Métrica	Baseline	Meta	Como Medir
Alertas/operador/dia	150-200	30-50	Query SQL events/day
Time to root cause	15-30 min	2-5 min	Tempo médio de análise
False positive rate	N/A	< 5%	Eventos incorretamente suprimidos
Noise reduction rate	N/A	> 60%	(Antes - Depois) / Antes
MTTR	45 min	20 min	Tempo médio de resolução
Incident escalations	20-30/mês	5-10/mês	Tickets escalados

Métricas - Queries SQL

1. Alertas por operador por dia:

```
SELECT
    DATE(FROM_UNIXTIME(a.clock)) as date,
    COUNT(*) / (SELECT COUNT(*) FROM users WHERE type = 1) as alerts_per_operator
FROM alerts a
WHERE a.clock >= UNIX_TIMESTAMP(NOW() - INTERVAL 30 DAY)
GROUP BY DATE(FROM_UNIXTIME(a.clock))
ORDER BY date DESC;
```

2. Noise reduction rate:

```
SELECT
    (SELECT COUNT(*) FROM events
     WHERE clock >= UNIX_TIMESTAMP(NOW() - INTERVAL 7 DAY)) as total_events,
    (SELECT COUNT(*) FROM event_suppress es
     JOIN events e ON es.eventid = e.eventid
     WHERE e.clock >= UNIX_TIMESTAMP(NOW() - INTERVAL 7 DAY)) as suppressed_events,
    ROUND(
        (SELECT COUNT(*) FROM event_suppress es
         JOIN events e ON es.eventid = e.eventid
         WHERE e.clock >= UNIX_TIMESTAMP(NOW() - INTERVAL 7 DAY)) * 100.0 /
        (SELECT COUNT(*) FROM events
```

Dashboard de Eficácia

Criar dashboard para acompanhar métricas:

```
Dashboard: Correlation & Suppression Effectiveness

Widgets:
  1. Graph: Events per day (before/after correlation)
  2. Graph: Suppression rate over time
  3. Plain text: Current noise reduction rate (> 60%?)
  4. Plain text: False positive rate (< 5%?)
  5. Top hosts: Most affected by correlation
  6. Problems: Critical events (never suppressed)
  7. Graph: MTTR trend
  8. Plain text: Alerts per operator per day
```

Revisar mensalmente e ajustar correlações conforme necessário

Revisão da Aula

O que aprendemos:

✅ Fundamentos da supressão de eventos
✅ Tipos de supressão (Dependência, Manutenção, Correlação)
✅ Event Correlation (temporal, tags, padrões)
✅ Planejamento de janelas de manutenção
✅ Boas práticas e erros comuns
✅ Configuração via GUI e API
✅ Laboratórios práticos
✅ Troubleshooting e métricas de sucesso

Impacto esperado:

- Redução de 60-80% em notificações

Próximos Passos

Após a aula:

1. **Implementar 2-3 correlações básicas** em seu ambiente
2. **Criar maintenance periods** para manutenções recorrentes
3. **Medir baseline** de alertas/dia antes de correlações
4. **Implementar dashboard** de eficácia
5. **Revisar mensalmente** e ajustar correlações

Fase 2 (Semanas 3-4):

- Adicionar correlações temporais
- Implementar padrões específicos do ambiente
- Documentar cada correlação criada

Recursos Adicionais

Documentação Oficial:

- Event Correlation
- Maintenance Periods
- Trigger Dependencies

Blog Posts:

- "Reducing Alert Fatigue with Zabbix Event Correlation"
- "Best Practices for Maintenance Windows"

Comunidade:

- Zabbix Forums: https://www.zabbix.com/forum
- Zabbix Share: https://share.zabbix.com

Perguntas?

Dúvidas sobre:

- Supressão de eventos
- Event Correlation
- Maintenance Periods
- Troubleshooting

Obrigado!

Até a próxima aula! 🚀

4Linux - Zabbix Advanced

📧 suporte@4linux.com.br

🌐 www.4linux.com.br