ZABBIX

Desvendando na prática: LLD Aninhada

Francisco Cunha

Coordenador de TI

Palestrantes

Francisco Vieira da Cunha

- Zabbix Certified Professional (ZCP)
- 15 anos de atuação em TI, foco em infraestrutura, observabilidade e automação
  - Especialização em monitoramento de grandes ambientes críticos
  - Especialista responsável por um dos maiores ambientes Zabbix do Brasil contendo mais de 45k Hosts e 5 Milhões de métricas
  - Engajado em práticas DevOps para o mundo da Observabildiade
- Atuação em projetos estratégicos de governo, contratos com relevância nacional.
- Pós graduação (Em andamento)
  - Liderança e Gestão de TI

Introdução ao LLD

O que é LLD (Low-Level Discovery)?

O que são as entidades da LLD?

LLD (método tradicional)

Como funciona:

- Descoberta gera **Itens e Triggers**

LLD (método tradicional)

Como funciona:

- Descoberta gera **Itens e Triggers**
- Descoberta gera **hosts filhos** a partir de um **host prototype**

LLD (método tradicional)

Como funciona:

- Descoberta gera **Itens e Triggers**
- Descoberta gera **hosts filhos** a partir de um **host prototype**
  - Cada **entidade** vira um **host** separado no **Zabbix**

Vantagens:

- Boa visibilidade isolada (entidade = hosts) em caso de **host prototype**.

LLD (método tradicional)

Como funciona:

- Descoberta gera **Itens e Triggers**
- Descoberta gera **hosts filhos** a partir de um **host prototype**
  - Cada **entidade** vira um **host** separado no **Zabbix**

Vantagens:

- Boa visibilidade isolada (entidade = hosts) em caso de **host prototype**.
- Útil para casos onde se quer inventário e gestão independente.

LLD (método tradicional)

Como funciona:

- Descoberta gera **Itens e Triggers**
- Descoberta gera **hosts filhos** a partir de um **host prototype**
  - Cada **entidade** vira um **host** separado no **Zabbix**

Vantagens:

- Boa visibilidade isolada (entidade = hosts) em caso de **host prototype**.
- Útil para casos onde se quer inventário e gestão independente.

Desvantagens:

- Multiplica hosts no **Zabbix**, pode gerar **overhead** grande.

LLD (método tradicional)

ZABBIX

Como funciona:

- Descoberta gera **Itens e Triggers**
- Descoberta gera **hosts filhos** a partir de um **host prototype**
  - Cada **entidade** vira um **host** separado no **Zabbix**

Vantagens:

- Boa visibilidade isolada (entidade = hosts) em caso de **host prototype**.
- Útil para casos onde se quer inventário e gestão independente.

Desvantagens:

- Multiplica hosts no **Zabbix**, pode gerar **overhead** grande.
- Escalabilidade sem hierarquia (clusters com milhares de pods = explosão de hosts independentes, sem relação entre si)

LLD Nested (aninhada) – NOVIDADE 7.4

Como funciona:

- Descoberta pode ser **aninhada** (cluster → namespace → pod → container).

LLD Nested (aninhada) – NOVIDADE 7.4

Como funciona:

- Descoberta pode ser **aninhada** (cluster → namespace → pod → container).
- Cada nível alimenta o próximo dentro da mesma árvore de LLD.

LLD Nested (aninhada) – NOVIDADE 7.4

Como funciona:

- Descoberta pode ser **aninhada** (cluster → namespace → pod → container).
- Cada nível alimenta o próximo dentro da mesma árvore de LLD.

Vantagens:

- Estrutura hierárquica fiel ao ambiente real.

LLD Nested (aninhada) – NOVIDADE 7.4

Como funciona:

- Descoberta pode ser **aninhada** (cluster → namespace → pod → container).
- Cada nível alimenta o próximo dentro da mesma árvore de LLD.

Vantagens:

- Estrutura hierárquica fiel ao ambiente real.
- Escalabilidade com hierarquia → não cria milhares de hosts, mantém dentro de um único contexto.

LLD Nested (aninhada) – NOVIDADE 7.4

Como funciona:

- Descoberta pode ser **aninhada** (cluster → namespace → pod → container).
- Cada nível alimenta o próximo dentro da mesma árvore de LLD.

Vantagens:

- Estrutura hierárquica fiel ao ambiente real.
- Escalabilidade com hierarquia → não cria milhares de hosts, mantém dentro de um único contexto.
- Ideal para **Kubernetes, cloud e ambientes dinâmicos**

LLD Nested (aninhada) – NOVIDADE 7.4

Como funciona:

- Descoberta pode ser **aninhada** (cluster → namespace → pod → container).
- Cada nível alimenta o próximo dentro da mesma árvore de LLD.

Vantagens:

- Estrutura hierárquica fiel ao ambiente real.
- Escalabilidade com hierarquia → não cria milhares de hosts, mantém dentro de um único contexto.
- Ideal para **Kubernetes, cloud e ambientes dinâmicos**
- Facilita correlação de métricas (CPU/memória de container ligado ao pod/namespace)

LLD Nested (aninhada) – NOVIDADE 7.4

ZABBIX

Como funciona:

- Descoberta pode ser **aninhada** (cluster → namespace → pod → container).
- Cada nível alimenta o próximo dentro da mesma árvore de LLD.

Vantagens:

- Estrutura hierárquica fiel ao ambiente real.
- Escalabilidade com hierarquia → não cria milhares de hosts, mantém dentro de um único contexto.
- Ideal para **Kubernetes, cloud e ambientes dinâmicos**
- Facilita correlação de métricas (CPU/memória de container ligado ao pod/namespace)

Desvantagens:

- Recurso novo → menos material e exemplos disponíveis.
- Pode exigir mais cuidado ao projetar os macros {#} entre níveis.

Tabela comparativa

Aspecto	LLD com Host Prototype	LLD Nested (Aninhado)
Estrutura	Cada entidade vira host separado	Descoberta hierárquica dentro do mesmo host
Escalabilidade	Gera muitos hosts (alto overhead)	Mais eficiente, sem explosão de hosts e com hierarquia definida
Visibilidade	Cada host é independente	Hierarquia clara (cluster → ns → pod → container)
Casos de uso típicos	Inventário de dispositivos, VMs	Kubernetes, cloud, containers, ambientes dinâmicos
Complexidade de configuração	Mais simples (já consolidado)	Nova abordagem, exige desenho cuidadoso

Conclusão

Ambas são úteis, porém devemos é **avaliar o caso de uso** e escolher a abordagem que mais se adapta ao nosso cenário.

Bora para prática?