Documentação e Tutorial: Monitoramento de Hypervisors e VMs Hyper-V com Zabbix e Grafana

Introdução

Este documento serve como uma documentação abrangente e um tutorial detalhado para o projeto de monitoramento de Hypervisors e suas Máquinas Virtuais (VMs) Hyper-V, utilizando a poderosa combinação do Zabbix para coleta de dados e o Grafana para visualização interativa. Desenvolvido por **Nilo Trigueiro** (LinkedIn: www.linkedin.com/in/nilotrigueiro), este projeto visa fornecer uma solução robusta e visualmente intuitiva para acompanhar o desempenho e a saúde de ambientes de virtualização.

O objetivo principal desta documentação é desmistificar a implementação e o funcionamento do dashboard, detalhando cada componente e explicando como as informações são coletadas, processadas e apresentadas. Ao final, o leitor terá uma compreensão clara da arquitetura, do código subjacente (HTML, CSS e JavaScript) e de como interpretar os dados exibidos no painel.

Visão Geral do Projeto

Em ambientes de TI modernos, a virtualização é um pilar fundamental, e o monitoramento eficaz de Hypervisors e VMs é crucial para garantir a estabilidade, o desempenho e a disponibilidade dos serviços. Este projeto aborda essa necessidade, integrando o Zabbix, uma ferramenta de monitoramento de código aberto amplamente utilizada, com o Grafana, uma plataforma de análise e visualização de dados.

Componentes Chave da Solução

1. **Zabbix:** Atua como o motor de coleta de dados. Ele é configurado para coletar métricas e informações detalhadas dos servidores Hypervisor e das VMs Hyper-V. Isso inclui dados sobre CPU, memória, disco, rede, status do sistema operacional e informações de hardware.
2. **Grafana:** É a interface de visualização. Ele se conecta ao Zabbix como uma fonte de dados e permite a criação de dashboards personalizados e dinâmicos. O grande

diferencial deste projeto reside no uso de um painel customizado para apresentar as informações de forma rica e interativa.
3. **Painel HTML Graphics (Grafana):** Este é o coração da interface visual. Diferente dos painéis tradicionais do Grafana, o painel HTML Graphics permite a inserção direta de código HTML para a estrutura, CSS para o estilo e JavaScript para a lógica de dados e interatividade. Isso oferece uma flexibilidade incomparável para criar layouts e apresentações de dados altamente personalizados.

Fluxo de Dados e Apresentação

O fluxo de dados pode ser resumido da seguinte forma:

- **Coleta:** O Zabbix coleta ativamente dados dos Hypervisors e VMs Hyper-V através de agentes ou protocolos de monitoramento (como SNMP, WMI, etc.).
- **Armazenamento:** Os dados coletados são armazenados no banco de dados do Zabbix.
- **Consulta:** O Grafana consulta o Zabbix para obter os dados necessários para o dashboard.
- **Processamento e Renderização:** O painel HTML Graphics no Grafana recebe esses dados. O JavaScript embarcado no painel processa os dados, o HTML define a estrutura de como esses dados serão exibidos, e o CSS garante que tudo seja apresentado de forma visualmente atraente e responsiva.

Este projeto demonstra uma abordagem poderosa para ir além das capacidades padrão de monitoramento, criando uma experiência de usuário aprimorada e uma visualização de dados mais significativa para administradores de sistemas e equipes de operações.

Pré-requisitos

Para replicar ou entender completamente este projeto, é fundamental ter os seguintes pré-requisitos e conhecimentos:

- **Zabbix Server:** Uma instância do Zabbix Server configurada e em funcionamento. Este é o coração do sistema de monitoramento, responsável por coletar, processar e armazenar os dados.
- **Zabbix Agent (ou métodos de coleta):** Os Hypervisors e VMs Hyper-V a serem monitorados devem ter o Zabbix Agent instalado e configurado, ou serem acessíveis via outros métodos de coleta de dados suportados pelo Zabbix (ex: SNMP, WMI para Windows, SSH para Linux, etc.). É crucial que os itens de monitoramento necessários para coletar as informações de hardware (fabricante, modelo, número de série, processador, núcleos, slots de memória, etc.) estejam configurados no Zabbix.

- **Grafana Server:** Uma instância do Grafana Server instalada e em funcionamento. O Grafana será a plataforma onde o dashboard será importado e visualizado.
- **Plugin Zabbix para Grafana:** O plugin oficial do Zabbix para Grafana (`alexanderzobnin-zabbix-datasource`) deve estar instalado e configurado no Grafana, apontando para a sua instância do Zabbix Server. Este plugin permite que o Grafana consulte os dados do Zabbix.
- **Plugin HTML Graphics para Grafana:** O painel utilizado neste dashboard é o `gapit-htmlgraphics-panel` . Este plugin deve estar instalado no seu Grafana para que o dashboard seja renderizado corretamente. Ele pode ser instalado via `grafana-cli plugins install gapit-htmlgraphics-panel` .
- **Conhecimentos Básicos:**
  - **Zabbix:** Familiaridade com a criação de hosts, itens, triggers e templates.
  - **Grafana:** Conhecimento básico sobre a criação e edição de dashboards, adição de fontes de dados e manipulação de painéis.
  - **HTML:** Compreensão da estrutura de documentos web e das tags HTML básicas.
  - **CSS:** Noções de estilização de elementos HTML, seletores e propriedades CSS.
  - **JavaScript:** Conhecimento básico de manipulação do DOM (Document Object Model), variáveis, funções e estruturas de controle.

Sem esses pré-requisitos, a importação e o funcionamento do dashboard podem não ocorrer como esperado, e a compreensão dos códigos pode ser dificultada.

Importando o Dashboard no Grafana

Para utilizar o dashboard em seu ambiente Grafana, siga os passos abaixo:

1. **Acesse o Grafana:** Abra seu navegador e acesse a URL do seu Grafana Server.
2. **Importar Dashboard:** No menu lateral do Grafana, navegue até `Dashboards` -> `Import` .
3. **Carregar JSON:** Na tela de importação, você terá algumas opções. Escolha a opção para carregar o arquivo JSON diretamente. Clique em `Upload JSON file` e selecione o arquivo `DASHBOARDHYPERVISOR52-1749579753620.json` que você possui.
4. **Configurar Opções:** Após carregar o JSON, o Grafana apresentará algumas opções de configuração:
   - **Name:** Você pode manter o nome padrão ou alterá-lo para algo mais descritivo.
   - **Folder:** Escolha uma pasta para organizar seu dashboard (opcional).
   - **Unique Identifier (UID):** O UID será gerado automaticamente ou mantido do JSON. Não é necessário alterar.

- **Zabbix Data Source: Crucial!** No campo `DS_ALEXANDERZOBNIN-ZABBIX-DATASOURCE`, selecione a sua fonte de dados Zabbix configurada no Grafana. Esta é a conexão que o dashboard usará para buscar os dados do Zabbix.

5. **Importar:** Clique no botão `Import`. O dashboard será importado e você será redirecionado para ele.

Após a importação, o dashboard começará a tentar buscar os dados do Zabbix. Se os pré-requisitos estiverem corretamente configurados e os itens de monitoramento no Zabbix estiverem coletando dados dos seus Hypervisors, as informações começarão a aparecer no dashboard.

Análise e Comentários Aprofundados dos Códigos

O coração deste dashboard interativo reside nos códigos CSS, HTML e JavaScript incorporados no painel HTML Graphics do Grafana. Cada um desempenha um papel vital na estruturação, estilização e funcionalidade dinâmica da visualização. A seguir, detalharemos cada um desses componentes, com comentários explicativos para facilitar a compreensão.

Código CSS: Estilização e Responsividade

O CSS (Cascading Style Sheets) é responsável por toda a parte visual do dashboard, desde as cores e fontes até o layout e a responsividade em diferentes tamanhos de tela. Ele garante que a interface seja intuitiva e agradável aos olhos.

```
 /* Estilo geral para body */
body { /* Define estilos globais para o corpo do painel, estabelecendo a base visual. */
  margin: 0; /* Remove a margem padrão do elemento <body>, garantindo que o conteúdo ocupe todo o espaço disponível. */
  font-family: \'Segoe UI\', Tahoma, Geneva, Verdana, sans-serif; /* Define uma pilha de fontes preferenciais, começando com 'Segoe UI' para sistemas Windows, seguida por fontes genéricas sem serifa. */
  background-color: #121212; /* Define a cor de fundo do painel como um tom escuro de cinza, ideal para dashboards em ambientes de monitoramento. */
  color: #fff; /* Define a cor padrão do texto como branco, contrastando com o fundo escuro. */
  line-height: 1.6; /* Aumenta o espaçamento entre as linhas de texto para melhorar a legibilidade. */
  display: flex; /* Ativa o modelo de layout flexbox para o corpo, permitindo um controle flexível sobre o alinhamento e distribuição dos itens filhos. */
  gap: 20px; /* Define um espaçamento de 20 pixels entre os itens flex (os cards de informação). */
  padding: 30px; /* Adiciona um preenchimento interno de 30 pixels em todos os lados do corpo, criando uma margem interna para o conteúdo. */
  box-sizing: border-box; /* Garante que o padding e a borda sejam incluídos na
```

```
largura e altura total do elemento, facilitando o cálculo de layout. */
}

/* Card do sistema operacional */
.os-card { /* Estilos aplicados ao card que exibe informações do sistema
operacional e a imagem do servidor. */
    display: flex; /* Utiliza flexbox para organizar o conteúdo interno (imagem do
servidor e informações do SO). */
    flex-direction: column; /* Organiza os itens flex (imagem e informações) em uma
coluna vertical. */
    align-items: center; /* Centraliza os itens flex horizontalmente dentro do card. */
    border: 1px solid #444; /* Adiciona uma borda fina e discreta de cor cinza escuro.
*/
    padding: 20px; /* Adiciona preenchimento interno de 20 pixels ao redor do
conteúdo do card. */
    border-radius: 12px; /* Arredonda os cantos do card, conferindo um visual mais
moderno. */
    background-color: #1e1e1e; /* Define a cor de fundo do card como um cinza
ligeiramente mais claro que o fundo do painel. */
    color: #fff; /* Mantém a cor do texto dentro do card como branco. */
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.3); /* Adiciona uma sombra sutil ao card,
criando um efeito de profundidade. */
    transition: transform 0.3s ease; /* Aplica uma transição suave de 0.3 segundos
para a propriedade 'transform', usada no efeito hover. */
    min-width:
20%; /* Define uma largura mínima de 20% da largura do contêiner pai. */
    max-width: 25%; /* Define uma largura máxima de 25% da largura do contêiner
pai. */
    min-height: 185px; /* Define uma altura mínima para o card. */
    max-height: 185px; /* Define uma altura máxima para o card. */
    margin-left: 20px; /* Adiciona uma margem à esquerda do card. */
    margin-right: 20px; /* Adiciona uma margem à direita do card. */
    align-content: center; /* Alinha o conteúdo ao centro quando há múltiplas linhas
(útil em layouts flex-wrap). */
    justify-content: center; /* Centraliza o conteúdo verticalmente dentro do card. */
}

.os-card:hover { /* Estilos aplicados quando o cursor do mouse passa sobre o card do
sistema operacional. */
  transform: scale(1.01); /* Aumenta ligeiramente o tamanho do card em 1%, criando
um efeito de destaque. */
}

.server-image-wrapper { /* Contêiner para a imagem do servidor, permitindo um
controle mais preciso sobre seu layout. */
  margin-bottom: 20px; /* Adiciona um espaçamento de 20 pixels abaixo da imagem
do servidor. */
  width: 100%; /* Garante que o wrapper ocupe 100% da largura disponível em seu
contêiner pai. */
  max-width: 250px; /* Limita a largura máxima do wrapper a 250 pixels. */
}
```

```
.server-image { /* Estilos aplicados à imagem do servidor dentro do card. */
  width: 100%; /* Faz com que a imagem ocupe 100% da largura de seu contêiner (.server-image-wrapper). */
  height: auto; /* Mantém a proporção original da imagem, ajustando a altura automaticamente. */
  object-fit: contain; /* Redimensiona a imagem para que ela caiba completamente dentro de seu contêiner, sem cortar, mas mantendo sua proporção. */
  border-radius: 8px; /* Arredonda os cantos da imagem. */
}

.os-info-row { /* Linha que agrupa o ícone do sistema operacional e suas informações textuais. */
  display: flex; /* Utiliza flexbox para alinhar o ícone e o texto lado a lado. */
  align-items: center; /* Alinha os itens (ícone e texto) verticalmente ao centro. */
  justify-content: center; /* Centraliza os itens horizontalmente dentro da linha. */
  gap: 20px; /* Adiciona um espaçamento de 20 pixels entre o ícone e o bloco de informações do SO. */
  width: 100%; /* Garante que a linha ocupe 100% da largura disponível. */
}

.os-image { /* Estilos aplicados ao ícone do sistema operacional. */
  width: 60px; /* Define uma largura fixa de 60 pixels para o ícone. */
  height: 60px; /* Define uma altura fixa de 60 pixels para o ícone. */
  object-fit: contain; /* Redimensiona o ícone para caber dentro de suas dimensões, mantendo a proporção. */
}

.os-info { /* Contêiner para o título e o valor do sistema operacional. */
  display: flex; /* Utiliza flexbox. */
  flex-direction: column; /* Organiza o título e o valor em uma coluna. */
  text-align: left; /* Alinha o texto à esquerda dentro deste contêiner. */
}

.os-title { /* Estilos para o rótulo do sistema operacional (ex: "Sistema Operacional"). */
  font-weight: 700; /* Define a fonte como negrito. */
  font-size: 18px; /* Define o tamanho da fonte. */
  margin-bottom: 4px; /* Adiciona uma pequena margem abaixo do título. */
  color: #aaa; /* Define a cor do texto como um cinza claro. */
}

.os-value { /* Estilos para o valor dinâmico do sistema operacional (ex: "Windows Server 2019"). */
  font-size: 14px; /* Define o tamanho da fonte. */
  font-weight: 600; /* Define a fonte como semi-negrito. */
  color: #fff; /* Define a cor do texto como branco. */
}

/* Container de informações do servidor */
.server-info-horizontal { /* Estilos para o contêiner principal que exibe as informações físicas do servidor e da memória em um layout horizontal. */
  background-color: #1e1e1e; /* Cor de fundo semelhante ao .os-card. */
  border-radius: 12px; /* Cantos arredondados. */
```

```
  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.3); /* Sombra para profundidade. */
  padding: 20px 30px; /* Preenchimento interno, com mais espaço nas laterais. */
  color: #fff; /* Cor do texto branco. */
  box-sizing: border-box; /* Inclui padding e borda no tamanho total. */
  display: flex; /* Utiliza flexbox. */
  flex-direction: column; /* Organiza as seções (CPU e Memória) em coluna. */
  gap: 20px; /* Espaçamento entre as seções. */
  min-width: 70%; /* Largura mínima de 70% do contêiner pai. */
  max-width: 75%; /* Largura máxima de 75% do contêiner pai. */
  min-height: 230px; /* Altura mínima. */
  max-height: 230px; /* Altura máxima. */
  margin-left: 20px; /* Margem à esquerda. */
  margin-right: 20px; /* Margem à direita. */
  justify-content: space-
between; /* Distribui o espaço verticalmente entre as seções. */
  align-items: stretch; /* Estica os itens para preencher a largura disponível. */
}

.title-line { /* Linha horizontal usada como divisor visual para títulos de seção. */
  height: 1px; /* Altura da linha. */
  background-color: #ccc; /* Cor da linha, um cinza claro. */
  width: 100%; /* Ocupa 100% da largura do contêiner pai. */
  margin-top: 5px; /* Margem superior. */
  margin-bottom: 10px; /* Margem inferior. */
}

/* Cada linha horizontal dentro do container */
.server-info-horizontal .row { /* Estilos para cada linha de informações dentro do
contêiner horizontal, como as informações de CPU ou memória. */
  display: flex; /* Utiliza flexbox para organizar os blocos de informação lado a lado.
*/
  flex-wrap:
nowrap; /* Impede que os blocos de informação quebrem para a próxima linha,
mantendo-os em uma única linha. */
  justify-content: space-between; /* Distribui o espaço horizontalmente entre os
blocos de informação. */
  overflow-x: auto; /* Adiciona uma barra de rolagem horizontal se o conteúdo
exceder a largura do contêiner. */
  min-width: 0; /* Permite que o contêiner encolha se necessário. */
  align-content:
center; /* Alinha o conteúdo ao centro (útil em layouts flex-wrap, mas aqui mantém o
alinhamento). */
  flex-direction: row; /* Organiza os itens em uma linha. */
}

/* Cada bloco de informação */
.server-info-horizontal .info-block { /* Estilos para cada bloco individual de
informação (ex: Fabricante, Modelo, Núcleos). */
  flex: 0 1 auto; /* Define como o item flex se comporta: não cresce (0), pode encolher
(1), e tem uma base de tamanho automática. */
  display: flex; /* Utiliza flexbox. */
  flex-direction: column; /* Organiza o rótulo e o valor em uma coluna. */
```

```
  word-wrap: break-word; /* Permite que palavras longas quebrem para a próxima linha. */
  overflow-wrap: break-word; /* Outra propriedade para controle de quebra de palavras. */
  white-space: normal; /* Permite que o texto quebre normalmente. */
}

/* Label dos blocos */
.server-info-horizontal .label { /* Estilos para o rótulo de cada bloco de informação (ex: "Fabricante", "Modelo"). */
  font-size: 0.85rem; /* Tamanho da fonte ligeiramente menor. */
  font-weight: 600; /* Semi-negrito. */
  color: #aaa; /* Cor cinza claro. */
  margin-bottom: 6px; /* Margem inferior. */
  cursor: help; /* Altera o cursor para um ponto de interrogação, indicando que há uma dica de ferramenta (tooltip). */
}

/* Valor dos blocos */
.server-info-horizontal .value { /* Estilos para o valor dinâmico de cada bloco de informação. */
  font-size: 1rem; /* Tamanho da fonte padrão. */
  font-weight: 700; /* Negrito. */
  color: #fff; /* Cor branca. */
  word-break: break-word; /* Permite quebra de palavras em qualquer ponto para evitar overflow. */
  white-space: normal; /* Espaço em branco normal. */
  overflow-wrap:
anywhere; /* Permite quebra de palavras em qualquer lugar, mesmo no meio de uma palavra. */
}

/* Divisor entre os blocos */
.server-info-horizontal .divider { /* Estilos para o divisor vertical entre os blocos de informação. */
  width: 1px; /* Largura do divisor. */
  background-color: #333; /* Cor do divisor, um cinza escuro. */
  margin: 0 8px; /* Margem horizontal para espaçamento. */
  height: auto; /* Altura automática para preencher o espaço disponível. */
}

/* Responsividade */
@media (max-width: 900px) { /* Media query que aplica estilos específicos quando a largura da tela é de 900 pixels ou menos, otimizando para dispositivos menores. */
  body { /* Ajusta o layout do corpo para telas menores. */
    flex-direction: column; /* Altera a direção dos itens flex para coluna, empilhando os cards verticalmente. */
    padding: 15px; /* Reduz o preenchimento geral do corpo. */
  }
  .os-card { /* Ajustes para o card do sistema operacional em telas menores. */
    flex: none; /* Remove o comportamento flexível, permitindo controle total da largura. */
```

```
    width: 100%; /* Ocupa 100% da largura disponível. */
    max-width: 500px; /* Limita a largura máxima para evitar que o card fique muito largo em telas médias. */
    margin-bottom: 30px; /* Adiciona uma margem inferior para separar os cards empilhados. */
  }
  .server-info-horizontal { /* Ajustes para o contêiner de informações do servidor em telas menores. */
    width: 100%; /* Ocupa 100% da largura disponível. */
  }
  .server-info-horizontal .row { /* Ajustes para as linhas de informação dentro do contêiner horizontal. */
    flex-wrap: wrap; /* Permite que os blocos de informação quebrem para a próxima linha se não houver espaço suficiente. */
  }
  .server-info-horizontal .info-block { /* Ajustes para os blocos de informação individuais. */
    min-width: 140px; /* Define uma largura mínima para os blocos, garantindo que o conteúdo seja legível. */
  }
  .server-info-horizontal .divider { /* Oculta os divisores verticais em telas menores, pois os itens estarão empilhados. */
    display: none; /* Não exibe o divisor. */
  }
}
```

Código HTML: Estrutura e Conteúdo

O HTML (HyperText Markup Language) é a espinha dorsal do painel, definindo a estrutura e o conteúdo que será exibido. Ele organiza as informações em seções lógicas, como o card do sistema operacional e as informações físicas do servidor.

```
 <!DOCTYPE html>
<html lang=\"pt-BR\"> <!-- Declara o tipo de documento como HTML5 e define o idioma principal da página como Português do Brasil, o que é importante para acessibilidade e renderização correta de caracteres. -->
<head>
  <meta charset=\"UTF-8\" /> <!-- Especifica a codificação de caracteres para o documento como UTF-8, garantindo que todos os caracteres especiais e acentuações sejam exibidos corretamente. -->
  <meta name=\"viewport\" content=\"width=device-width, initial-scale=1\" /> <!-- Configura a viewport para dispositivos móveis, garantindo que a página seja renderizada de forma responsiva e se ajuste à largura da tela do dispositivo, com um zoom inicial de 100%. -->
  <title>Servidor e SO</title> <!-- Define o título da página, que geralmente aparece na aba do navegador ou na barra de título da janela. -->

  <!-- Link para o CSS externo -->
  <link rel=\"stylesheet\" href=\"styles.css\" /> <!-- Inclui um arquivo CSS externo
```

```
chamado "styles.css". No contexto do painel HTML Graphics do Grafana, este CSS é injetado diretamente no painel, mas a sintaxe de link externo é mantida para clareza e organização. -->
</head>
<body>

  <!-- Card do SO e imagem do servidor -->
  <!-- src antigo: https://media.xbyte.com/servers/Dell-PowerEdge-R540-Server_01.png -->
  <div class=\"os-card\"> <!-- Um contêiner <div> com a classe "os-card" que agrupa visualmente as informações do sistema operacional e a imagem do servidor. Esta classe é estilizada no CSS para criar um "card" visual. -->
    <div class=\"server-image-wrapper\"> <!-- Um wrapper <div> para a imagem do servidor, permitindo um controle mais preciso sobre o layout e o dimensionamento da imagem via CSS. -->
      <img
        src=\"/public/img/nilo/servidores/Dell-PowerEdge-R540.png\" <!-- A tag <img> exibe a imagem do servidor. O atributo `src` aponta para o caminho da imagem. Note que é um caminho relativo, indicando que a imagem deve ser acessível a partir do ambiente onde o Grafana está sendo executado (provavelmente um servidor web que serve arquivos estáticos). -->
        alt=\"Servidor\" <!-- O atributo `alt` fornece um texto alternativo para a imagem, que é exibido se a imagem não puder ser carregada ou para usuários com leitores de tela, melhorando a acessibilidade. -->
        class=\"server-image\" <!-- A classe "server-image" é usada para aplicar estilos CSS específicos a esta imagem. -->
      />
    </div>

    <div class=\"os-info-row\"> <!-- Uma <div> com a classe "os-info-row" que organiza o ícone do sistema operacional e suas informações textuais em uma linha. -->
      <img
        src=\"https://img.icons8.com/ios_filled/512/228BE6/windows-10.png\" <!-- A tag <img> para o ícone do sistema operacional. O `src` aponta para uma imagem externa hospedada no `icons8.com`, representando o logo do Windows 10. -->
        alt=\"OS Logo\" <!-- Texto alternativo para o logo do SO. -->
        class=\"os-image\" <!-- Classe CSS para estilização do ícone do SO. -->
        id=\"os-image\" <!-- O `id="os-image"` é crucial, pois permite que o JavaScript manipule dinamicamente o atributo `src` desta imagem, alterando o logo do SO com base nos dados recebidos do Zabbix. -->
      />
      <div class=\"os-info\">
<!-- Um contêiner <div> com a classe "os-info" para agrupar o título e o valor do sistema operacional. -->
        <div class=\"os-title\">Sistema Operacional</div> <!-- Uma <div> com a classe "os-title" que exibe o rótulo estático "Sistema Operacional". -->
        <div id=\"htmlgraphics-value\" class=\"os-value\">Windows Server 2019</div>
<!-- Uma <div> com o `id="htmlgraphics-value"` e a classe "os-value". Este é o elemento onde o nome do sistema operacional será exibido. O valor "Windows Server 2019" é um placeholder inicial que será substituído dinamicamente pelo JavaScript. -->
      </div>
    </div>
```

```
  </div>

  <!-- Informações do servidor com layout horizontal mais compacto -->
  <div class=\"server-info-horizontal\"> <!-- Um contêiner <div> com a classe "server-info-horizontal" que organiza as informações físicas detalhadas do servidor em um layout horizontal, otimizado para economia de espaço. -->
      <div class=\"container1-cpu\"> <!-- Uma <div> com a classe "container1-cpu" que agrupa as informações relacionadas à CPU e ao hardware geral do servidor. -->
          <div class=\"title1\">Informações Físicas do Servidor</div> <!-- Um título para esta seção. -->
          <div class=\"title-line\"></div> <!-- Uma <div> com a classe "title-line" que serve como um divisor visual abaixo do título. -->
          <div class=\"row\">
<!-- Uma <div> com a classe "row" que organiza os blocos de informação desta seção em uma linha. -->
          <div class=\"info-block\"> <!-- Um bloco individual de informação para o Fabricante. Cada "info-block" contém um rótulo e um valor. -->
              <div class=\"label\" data-tooltip=\"Fabricante do servidor\">Fabricante</div> <!-- O rótulo "Fabricante" com um atributo `data-tooltip` que fornece uma dica de ferramenta ao passar o mouse. -->
              <div id=\"fabricante\" class=\"value\">Dell Inc.</div> <!-- O valor do fabricante, com um `id="fabricante"` para atualização via JavaScript. O valor "Dell Inc." é um placeholder. -->
          </div>
          <div class=\"divider\"></div> <!-- Um divisor vertical entre os blocos de informação. -->
          <div class=\"info-block\"> <!-- Bloco para o Modelo do servidor. -->
              <div class=\"label\" data-tooltip=\"Modelo do servidor\">Modelo</div>
              <div id=\"modelo\" class=\"value\">PowerEdge R540</div>
          </div>
          <div class=\"divider\"></div>
          <div class=\"info-block\"> <!-- Bloco para o Número de Série do servidor. -->
              <div class=\"label\" data-tooltip=\"Número de série do servidor\">Nº de Série</div>
              <div id=\"numero-serie\" class=\"value\">18NDYR2</div>
          </div>
          <div class=\"divider\"></div>
          <div class=\"info-block\"> <!-- Bloco para o Processador do servidor. -->
              <div class=\"label\" data-tooltip=\"Modelo do processador\">Processador</div>
              <div id=\"processador\" class=\"value\">Intel(R) Xeon(R) Silver 4110 CPU @ 2.10GHz</div>
          </div>
          <div class=\"divider\"></div>
          <div class=\"info-block\"> <!-- Bloco para o número de Núcleos físicos. -->
              <div class=\"label\" data-tooltip=\"Número de núcleos físicos\">Núcleos</div>
              <div id=\"nucleos\" class=\"value\">16</div>
          </div>
          </div>
      </div>
```

```
    <div class=\"container2-cpu\" > <!-- Uma <div> com a classe "container2-cpu" que
agrupa as informações relacionadas à memória física do servidor. -->
        <div class=\"title2\">Informações Físicas da Memória</div>
<!-- Um título para esta seção. -->
        <div class=\"title-line\"></div> <!-- Um divisor visual abaixo do título. -->
        <div class=\"row\">
<!-- Uma <div> com a classe "row" que organiza os blocos de informação desta seção
em uma linha. -->
        <div class=\"info-block\"> <!-- Bloco para o Total de Slots de Memória. -->
          <div class=\"label\" data-tooltip=\"Total de slots de memória disponíveis\">Slots
Memória (total)</div>
          <div id=\"total-slots\" class=\"value\">16</div>
        </div>
        <div class=\"divider\"></div>
        <div class=\"info-block\"> <!-- Bloco para os Slots Ocupados. -->
          <div class=\"label\" data-tooltip=\"Quantidade de slots de memória
ocupados\">Slots Ocupados</div>
          <div id=\"slots-ocupados\" class=\"value\">8</div>
        </div>
        <div class=\"divider\"></div>
        <div class=\"info-block\"> <!-- Bloco para o Tipo de Memória do Módulo 1. -->
          <div class=\"label\" data-tooltip=\"Tipo da memória do módulo 1\">Tipo
Memória Módulo 1</div>
          <div id=\"tipo-memoria\" class=\"value\">DDR4</div>
        </div>
        <div class=\"divider\"></div>
        <div class=\"info-block\">
<!-- Bloco para a Velocidade da Memória do Módulo 1. -->
          <div class=\"label\" data-tooltip=\"Velocidade da memória do módulo 1 em
MHz\">Velocidade Memória Módulo 1 (MHz)</div>
          <div id=\"velocidade-memoria\" class=\"value\">2666</div>
        </div>
        <div class=\"divider\"></div>
        <div class=\"info-block\"> <!-- Bloco para o Tamanho da Memória do Módulo 1.
-->
          <div class=\"label\" data-tooltip=\"Tamanho da memória do módulo 1 em
GB\">Tamanho Memória Módulo 1 (GB)</div>
          <div id=\"size-memoria\" class=\"value\">16</div>
        </div>
        </div>
    </div>

  </div>

</body>
</html>
```

Código JavaScript: Lógica e Interatividade (onInit e onRender)

O JavaScript é a camada de inteligência do painel HTML Graphics, responsável por buscar os dados do Zabbix, processá-los e atualizar dinamicamente o conteúdo HTML. Ele é dividido em duas funções principais: `onInit` e `onRender`.

Função `onInit`

A função `onInit` é executada apenas uma vez, no momento em que o painel é carregado e inicializado no Grafana. Ela é ideal para configurações iniciais, como a definição de valores padrão ou ajustes baseados no tema do Grafana.

```
 // Sets the text from customProperties
const htmlgraphicsText = htmlNode.getElementById(\'htmlgraphics-text\'); // Obtém uma referência ao elemento HTML com o ID 'htmlgraphics-text'. A variável `htmlNode` é fornecida pelo ambiente do Grafana HTML Graphics e representa o DOM do painel.

if (htmlgraphicsText) { // Verifica se o elemento foi encontrado no DOM para evitar erros caso o ID não exista.
  htmlgraphicsText.textContent = customProperties.text; // Define o conteúdo de texto do elemento. `customProperties` é um objeto que pode conter dados configurados no painel do Grafana. Neste caso, ele espera uma propriedade `text`.

  // Change the text color based on the theme
  if (theme.isDark) { // `theme` é um objeto fornecido pelo Grafana que contém informações sobre o tema atual (claro ou escuro). `theme.isDark` é um booleano.
    htmlgraphicsText.style.color = \'green\'; // Se o tema for escuro, a cor do texto é definida como verde para melhor visibilidade.
  } else { // Se o tema não for escuro (ou seja, claro).
    htmlgraphicsText.style.color = \'red\'; // A cor do texto é definida como vermelho.
  }
}
```

Explicação: Embora o código `onInit` presente no JSON seja um exemplo simples de como interagir com o DOM e o tema do Grafana, ele demonstra o potencial de personalização. Para este dashboard específico, a maior parte da lógica dinâmica está na função `onRender`, que lida com a atualização dos dados do Zabbix.

Função `onRender`

A função `onRender` é a mais importante para a funcionalidade deste dashboard. Ela é executada sempre que os dados do Zabbix são atualizados (por exemplo, a cada intervalo de atualização do painel no Grafana). É aqui que os dados brutos do Zabbix são processados e usados para preencher os elementos HTML do dashboard.

```
 // Cria uma referência local para evitar conflito
const serverData = data; // `data` é uma variável global fornecida pelo Grafana que contém todas as séries de dados retornadas pela consulta ao Zabbix. Criar uma referência local (`serverData`) é uma boa prática para evitar modificações acidentais na variável global e melhorar a legibilidade.

// --- Função auxiliar para pegar valor da série pelo índice ---
function getSerieValue(index) { // Esta função auxiliar simplifica a obtenção do valor mais recente de uma série de dados específica, identificada pelo seu índice na array `serverData.series`.
  const serie =
serverData.series[index]; // Acessa o objeto da série de dados no índice fornecido.
  if (!serie) return \"Sem dado\"; // Se a série não existir (índice inválido ou dados ausentes), retorna uma string indicando a falta de dados.

  const valueField = serie.fields.find(f => f.name === \"Value\"); // Dentro de cada série, os dados são organizados em `fields`. Esta linha procura um campo chamado "Value", que é onde o Grafana geralmente armazena os valores numéricos ou textuais das métricas.
  if (!valueField) return \"Sem dado\"; // Se o campo "Value" não for encontrado na série, retorna "Sem dado".

  const lastIndex = valueField.values.length - 1; // Obtém o índice do último elemento na array `values` do campo "Value". O último elemento geralmente representa o dado mais recente.
  return valueField.values.get(lastIndex); // Retorna o valor correspondente ao `lastIndex`. O método `get()` é usado para acessar o valor de um `DataFrameView` do Grafana.
}

// Dados do servidor
// As linhas abaixo utilizam a função `getSerieValue` para extrair dados específicos das séries do Zabbix. É crucial que a ordem das séries retornadas pelo Zabbix (e, consequentemente, seus índices) corresponda à expectativa do script. Por exemplo, a série no índice 1 é esperada para ser o Fabricante, no índice 2 o Modelo, e assim por diante.
const fabricante = getSerieValue(1);       // Valor do Fabricante do servidor.
const modelo = getSerieValue(2);           // Valor do Modelo do servidor.
const numeroSerie = getSerieValue(3);      // Valor do Número de Série do servidor.
const processadorRaw = getSerieValue(4);   // Valor bruto do Processador. Este valor pode ser uma string JSON que precisa de parsing.
const nucleosRaw = getSerieValue(5);       // Valor bruto dos Núcleos. Também pode ser uma string JSON.
const totalSlotsMemoria = getSerieValue(6); // Valor do Total de Slots de Memória.
const slotsOcupados = getSerieValue(7);    // Valor dos Slots de Memória Ocupados.
const tipoMemoriaModulo1 = getSerieValue(8); // Valor do Tipo de Memória do Módulo 1.
const velocidadeMemoriaModulo1 = getSerieValue(9); // Valor da Velocidade da Memória do Módulo 1.
const sizeMemoriaModulo1 = getSerieValue(10);      // Valor do Tamanho da
```

Memória do Módulo 1.

```
// Processador com contagem e destaque
let processador = \"Sem dado\"; // Inicializa a variável que armazenará a descrição formatada do processador.
if (processadorRaw) { // Verifica se há dados para o processador.
  try { // Bloco try-catch para lidar com possíveis erros no parsing JSON.
    const lista = JSON.parse(processadorRaw); // Tenta converter a string `processadorRaw` em um array JavaScript. Isso é necessário se o Zabbix retornar uma lista de processadores como uma string JSON.
    const contagem = {}; // Objeto para armazenar a contagem de cada modelo de CPU.
    lista.forEach(cpu => { // Itera sobre cada item na lista de CPUs.
      contagem[cpu] = (contagem[cpu] || 0) + 1; // Incrementa a contagem para o modelo de CPU atual. Se o modelo ainda não estiver no objeto `contagem`, ele é inicializado com 0 antes de ser incrementado.
    });
    const descricoes = Object.entries(contagem).map(([modelo, quantidade]) => { // Converte o objeto `contagem` em um array de pares [modelo, quantidade] e mapeia cada par para uma string descritiva.
      const plural = quantidade > 1 ? \"processadores\" : \"processador\"; // Determina se a palavra "processador" deve ser plural ou singular.
      return `Existem <span class=\"highlight\">${quantidade}</span> ${plural} – Modelo ${modelo}`; // Retorna uma string formatada. A tag `<span>` com a classe `highlight` pode ser usada para estilizar a quantidade no CSS.
    });
    processador = descricoes.join(\"<br>\"); // Junta todas as descrições com a tag `<br>` para criar quebras de linha no HTML, exibindo cada modelo de CPU em uma nova linha.
  } catch (e) { // Captura qualquer erro que ocorra no bloco try (por exemplo, se `processadorRaw` não for um JSON válido).
    console.error(\"Erro ao processar processadorRaw:\", e); // Loga o erro no console do navegador (útil para depuração).
    processador = processadorRaw; // Em caso de erro, o valor bruto é usado como fallback.
  }
}

// Núcleos físicos
let nucleos = \"Sem dado\"; // Inicializa a variável para os núcleos.
if (nucleosRaw) { // Verifica se há dados para os núcleos.
  try { // Bloco try-catch para parsing JSON.
    const lista = JSON.parse(nucleosRaw); // Converte a string `nucleosRaw` em um array JavaScript.
    nucleos = lista.join(\" + \"); // Junta os valores dos núcleos com " + " (ex: "8 + 8" para dois processadores de 8 núcleos).
  } catch (e) { // Captura erros.
    console.error(\"Erro ao processar nucleosRaw:\", e); // Loga o erro.
    nucleos = nucleosRaw; // Fallback para o valor bruto.
  }
}
```

```
// Atualizar DOM com os dados do servidor
// As linhas abaixo atualizam o conteúdo dos elementos HTML correspondentes aos IDs definidos no HTML. `htmlNode` é a referência ao documento DOM do painel.
htmlNode.getElementById(\'fabricante\').textContent = fabricante; // Atualiza o texto do elemento com ID 'fabricante'.
htmlNode.getElementById(\'modelo\').textContent = modelo; // Atualiza o texto do elemento com ID 'modelo'.
htmlNode.getElementById(\'numero-serie\').textContent = numeroSerie; // Atualiza o texto do elemento com ID 'numero-serie'.
htmlNode.getElementById(\'processador\').innerHTML = processador; // Atualiza o HTML interno do elemento 'processador'. `innerHTML` é usado aqui porque a string `processador` pode conter tags HTML (como `<span>` e `<br>`).
htmlNode.getElementById(\'nucleos\').textContent = nucleos; // Atualiza o texto do elemento com ID 'nucleos'.

// Atualizar DOM com novos dados de memória
// Atualiza os elementos HTML relacionados à memória, utilizando o operador de coalescência nula (??) para garantir que "Sem dado" seja exibido se o valor for `null` ou `undefined`.
htmlNode.getElementById(\'total-slots\').textContent = totalSlotsMemoria ?? \"Sem dado\";
htmlNode.getElementById(\'slots-ocupados\').textContent = slotsOcupados ?? \"Sem dado\";
htmlNode.getElementById(\'tipo-memoria\').textContent = tipoMemoriaModulo1 ?? \"Sem dado\";
htmlNode.getElementById(\'velocidade-memoria\').textContent = velocidadeMemoriaModulo1 ?? \"Sem dado\";
htmlNode.getElementById(\'size-memoria\').textContent = sizeMemoriaModulo1 ?? \"Sem dado\";

// Dados do sistema operacional
const htmlgraphicsValue = htmlNode.getElementById(\'htmlgraphics-value\'); // Referência ao elemento que exibe o nome do SO.
const osImage = htmlNode.getElementById(\'os-image\'); // Referência ao elemento da imagem do SO.

if (htmlgraphicsValue) { // Verifica se o elemento do valor do SO existe.
  const osFieldNames = [ // Array de possíveis nomes de campos para a métrica do sistema operacional no Zabbix. Isso permite flexibilidade na nomeação da métrica.
    \"Sistema Operacional\",
    \"SistemaOperacional\",
    \"OS\",
    \"Operating System\",
    \"operatingsystem\",
  ];

  let osSerie = serverData.series.find(serie => // Procura na array de séries de dados do Zabbix por uma série cujo nome (case-insensitive) corresponda a um dos nomes em `osFieldNames`.
    osFieldNames.some(name => serie.name?.toLowerCase() === name.toLowerCase())
```

```
  );\n\n  if (!osSerie) osSerie = serverData.series[0]; // Fallback: Se nenhuma série com os nomes esperados for encontrada, assume-se que a primeira série (`serverData.series[0]`) contém a informação do SO. Isso pode ser um ponto de falha se a primeira série não for o SO.

  if (osSerie) { // Se uma série de SO foi encontrada (ou assumida).
    const osValueField = osSerie.fields.find(f => f.name === \"Value\"); // Procura o campo "Value" dentro da série do SO.
    if (osValueField) { // Se o campo "Value" existir.
      const osName = osValueField.values.get(osValueField.values.length - 1); // Obtém o nome do sistema operacional.
      htmlgraphicsValue.textContent = osName; // Atualiza o texto do elemento HTML com o nome do SO.

      // Lógica para mudar a imagem do SO com base no nome
      if (osImage) { // Verifica se o elemento da imagem do SO existe.
        if (osName.toLowerCase().includes(\'windows\')) { // Se o nome do SO contiver "windows" (case-insensitive).
          osImage.src = \'https://img.icons8.com/ios_filled/512/228BE6/windows-10.png\'; // Define o `src` da imagem para o logo do Windows.
        } else if (osName.toLowerCase().includes(\'linux\')) { // Se o nome do SO contiver "linux".
          osImage.src = \'https://img.icons8.com/ios_filled/512/228BE6/linux.png\'; // Define o `src` da imagem para o logo do Linux.
        } else { // Para qualquer outro sistema operacional não explicitamente mapeado.
          osImage.src = \'https://img.icons8.com/ios_filled/512/228BE6/question-mark.png\'; // Define o `src` para um ícone de ponto de interrogação, indicando um SO desconhecido.
        }
      }
    }
  }
}
```

Uso e Interpretação do Dashboard

Após a importação e configuração bem-sucedida do dashboard, você terá uma ferramenta visualmente rica para monitorar seus Hypervisors e VMs Hyper-V. Esta seção detalha como interpretar as informações apresentadas e como o dashboard pode ser usado no dia a dia.

Layout Geral do Dashboard

O dashboard é dividido em seções principais, projetadas para fornecer uma visão rápida e detalhada do ambiente:

1. **Título Principal:** No topo, um título destacado (ex: "MONITORAMENTO DO HYPERVISOR 25052RECDF") identifica o servidor que está sendo monitorado. Este título é estático no JSON fornecido, mas pode ser parametrizado no Grafana para selecionar diferentes Hypervisors.
2. **Card do Sistema Operacional e Imagem do Servidor:** Localizado à esquerda, este card exibe a imagem do servidor físico e as informações do sistema operacional instalado no Hypervisor. A imagem do SO (Windows ou Linux) é atualizada dinamicamente com base nos dados do Zabbix.
3. **Informações Físicas do Servidor (CPU):** À direita do card do SO, esta seção apresenta detalhes cruciais sobre o hardware do Hypervisor, incluindo:
    - **Fabricante:** O fabricante do servidor (ex: Dell Inc.).
    - **Modelo:** O modelo específico do servidor (ex: PowerEdge R540).
    - **Nº de Série:** O número de série único do equipamento.
    - **Processador:** Detalhes sobre o(s) processador(es) instalado(s), incluindo modelo e quantidade. O script JavaScript processa essa informação para exibir a contagem de processadores de forma clara.
    - **Núcleos:** O número total de núcleos físicos do processador.
4. **Informações Físicas da Memória:** Abaixo das informações da CPU, esta seção detalha a configuração da memória RAM do Hypervisor:
    - **Slots Memória (total):** O número total de slots de memória disponíveis na placa-mãe.
    - **Slots Ocupados:** Quantos slots de memória estão atualmente preenchidos.
    - **Tipo Memória Módulo 1:** O tipo de memória (ex: DDR4) do primeiro módulo detectado.
    - **Velocidade Memória Módulo 1 (MHz):** A velocidade de operação da memória em MHz.
    - **Tamanho Memória Módulo 1 (GB):** O tamanho em Gigabytes do primeiro módulo de memória.

Como os Dados são Apresentados

Os dados exibidos no dashboard são coletados pelo Zabbix e, em seguida, processados pelo JavaScript do painel HTML Graphics. É importante entender a relação entre as métricas do Zabbix e os campos exibidos no dashboard:

- **Mapeamento de Dados:** O script JavaScript (`onRender`) utiliza a função `getSerieValue(index)` para extrair os dados das séries retornadas pelo Zabbix. O

`index` (índice) é a posição da métrica na lista de dados que o Zabbix envia para o Grafana. Portanto, é fundamental que as métricas no Zabbix estejam configuradas e retornem os dados na ordem esperada pelo script.

- **Atualização Dinâmica:** Sempre que o Zabbix envia novos dados para o Grafana (conforme o intervalo de atualização configurado no painel), a função `onRender` é executada novamente, atualizando todos os campos no dashboard com os valores mais recentes.
- **Tratamento de Dados Complexos:** O script demonstra inteligência ao lidar com dados como o "Processador". Se o Zabbix retornar uma lista de processadores (por exemplo, em formato JSON), o script é capaz de contar e exibir a quantidade de cada modelo, tornando a informação mais legível.
- **Responsividade:** O CSS garante que o layout do dashboard se adapte a diferentes tamanhos de tela. Em telas menores (como tablets ou smartphones), os cards e seções se reorganizam verticalmente para manter a usabilidade.

Dicas para Interpretação e Solução de Problemas

- **Verifique a Fonte de Dados:** Se o dashboard não estiver exibindo dados, o primeiro passo é verificar se a fonte de dados Zabbix está corretamente configurada no Grafana e se ela consegue se comunicar com o Zabbix Server.
- **Itens de Monitoramento no Zabbix:** Certifique-se de que os itens de monitoramento no Zabbix que coletam as informações de hardware (fabricante, modelo, etc.) estão ativos e coletando dados. Os nomes das métricas no Zabbix devem corresponder (ou serem mapeados) aos nomes esperados pelo script JavaScript.
- **Logs do Navegador:** Para depurar problemas no painel HTML Graphics, abra o console de desenvolvedor do seu navegador (geralmente F12). O script JavaScript pode imprimir mensagens de erro (`console.error`) que podem ajudar a identificar problemas no processamento dos dados ou na atualização do DOM.
- **Ordem das Séries:** A função `getSerieValue(index)` depende da ordem das séries de dados. Se você alterar a consulta do painel no Grafana ou a forma como o Zabbix retorna os dados, os índices podem mudar, e o script precisará ser ajustado.
- **Personalização:** Este dashboard serve como um excelente ponto de partida. Você pode personalizá-lo ainda mais, adicionando novas métricas do Zabbix, alterando o layout HTML, ajustando os estilos CSS ou expandindo a lógica JavaScript para exibir mais informações ou criar interações adicionais.

Este tutorial visa capacitá-lo a não apenas usar este dashboard, mas também a entender sua construção e adaptá-lo às suas necessidades específicas de monitoramento. Com o Zabbix e o Grafana, as possibilidades de visualização e análise de dados são vastas, e este projeto é um exemplo prático de como aproveitar esse potencial.