Material utilizado no webinar da Zabbix e apresentado pela JLCP.

Pre-requisitos:

1. Ambiente Zabbix;
2. Cluster Kubernetes;
3. Helm Client;
4. Kubectl.

Não tem um ambiente com kubernetes e quer testar?

Utilize o kilercoda.

https://killercoda.com/playgrounds/scenario/kubernetes

Caso necessite:

Instalação Helm:

```
curl -ssl https://raw.githubusercontent.com/helm/helm/master/scripts/get-helm-3 |
bash
helm version --short
```

Instalação kubectl:

```
curl -LO "https://dl.k8s.io/release/$(curl -L -s
https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"

sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl
kubectl version --client
```

Laboratório:

1. Chart

Podemos realizar o download do repositorio inteiro.

```
git clone https://git.zabbix.com/scm/zt/kubernetes-helm.git
```

Ou Download do Chart.

```
helm repo add zabbix-chart-7.0
https://cdn.zabbix.com/zabbix/integrations/kubernetes-helm/7.0
```

navegue até o diretorio:

```
cd zabbix-helm-chrt
```

2. Helm Chart

Verifique a estrutura do chart, arquivos do helm:

```
└── zabbix-helm-chrt
    ├── Chart.lock
    ├── charts
    │   └── kube-state-metrics
    │       ├── Chart.yaml
    │       ├── README.md
    │       ├── templates
    │       │   ├── ciliumnetworkpolicy.yaml
    │       │   ├── clusterrolebinding.yaml
    │       │   ├── deployment.yaml
    │       │   ├── _helpers.tpl
    │       │   ├── kubeconfig-secret.yaml
    │       │   ├── networkpolicy.yaml
    │       │   ├── NOTES.txt
    │       │   ├── pdb.yaml
    │       │   ├── podsecuritypolicy.yaml
    │       │   ├── psp-clusterrolebinding.yaml
    │       │   ├── psp-clusterrole.yaml
    │       │   ├── rbac-configmap.yaml
    │       │   ├── rolebinding.yaml
    │       │   ├── role.yaml
    │       │   ├── serviceaccount.yaml
    │       │   ├── servicemonitor.yaml
    │       │   ├── service.yaml
    │       │   ├── stsdiscovery-rolebinding.yaml
    │       │   ├── stsdiscovery-role.yaml
    │       │   └── verticalpodautoscaler.yaml
    │       └── values.yaml
    ├── Chart.yaml
    ├── images
    │   └── architecture.png
    ├── LICENSE
    ├── README.md
    ├── templates
    │   ├── agent-psp-clusterrolebinding.yaml
    │   ├── agent-psp-clusterrole.yaml
    │   ├── agent-psp.yaml
    │   ├── agent-service-account.yaml
    │   ├── agent-service.yaml
    │   ├── clusterrole-binding.yaml
    │   ├── cluster-role.yaml
    │   ├── _helpers.tpl
```

```
      │   ├── NOTES.txt
      │   ├── secret.yaml
      │   ├── service-account.yaml
      │   ├── service.yaml
      │   ├── zabbix-agent.yaml
      │   ├── zabbix-java-gateway.yaml
      │   └── zabbix-proxy.yaml
      └── values.yaml

  6 directories, 45 files
```

3. Conhecendo arquivo de Values

Verifique o values e identifique o necessita de alteração.

```
values.yaml
```

4. Values

Alterar Values.

```
values.yaml
```

5. Helm tempalte:

Simular gerando os yaml com template:

```
helm template zabbix . -f .\values.yaml --debug
```

6. Instalar Creando Namespace:

```
kubectl create ns monitoring
```

Instalando:

```
helm upgrade --install zabbix . --dependency-update -f values.yaml -n monitoring
```

7. Namespace

Verificar tudo que foi criado no namespace.

```
helm template horus-frontend-staging . -f .\values-staging.yaml --debug
```

8. Secret

Capturando a secret para ser utilizada no ambiente.

```
kubectl get secret zabbix-service-account -n monitoring -o jsonpath={.data.token}
| base64 -d
```

9. Trobleshooting

Verificar todos os resources criados.

```
kubectl get all -n monitoring
```

Executar comandos no container.

```
kubectl exec -it pod/zabbix-proxy-64cfbbcbd8-f28d5 -- bash
```

Verificando logs

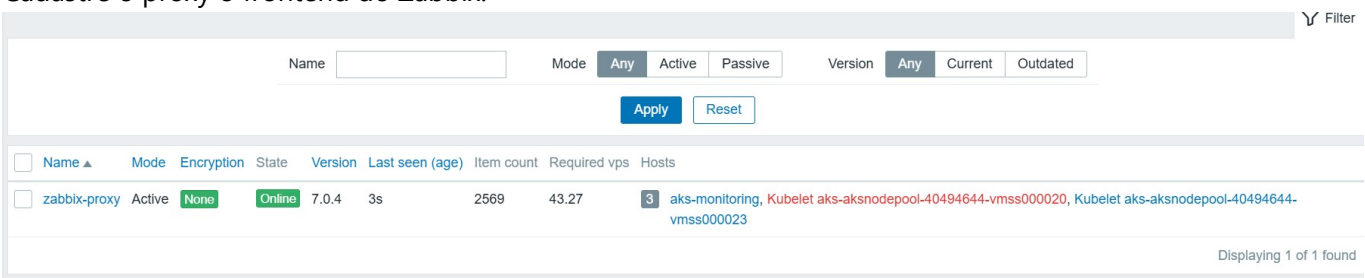
```
kubectl logs pod/zabbix-proxy-64cfbbcbd8-zq2dc
```

Verificando componentes:

```
kubectl get componentstatus
```

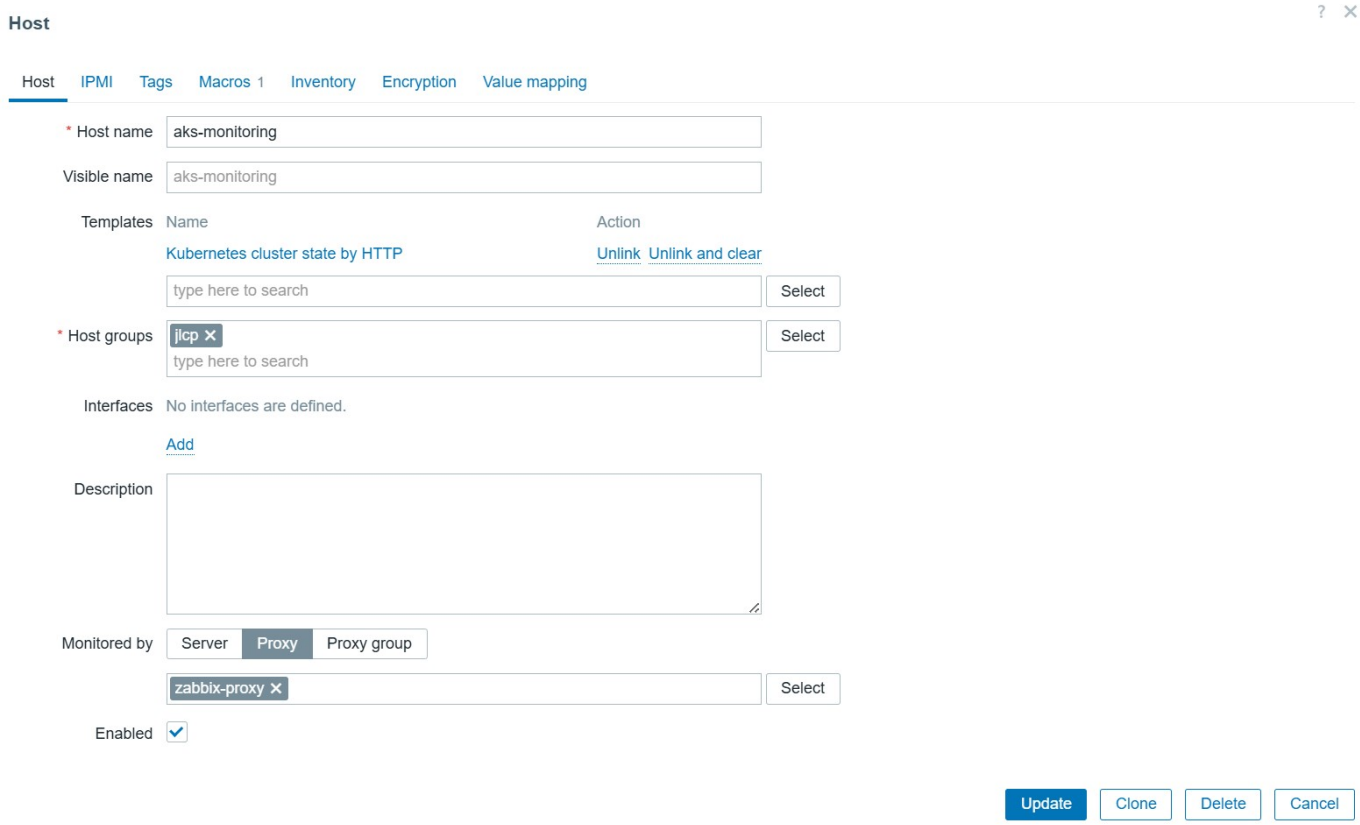
10. Cadastro de proxy no Zabbix.

Cadastre o proxy o frontend do Zabbix.



11. Criacão do host do Kubernetes.

Cadastro do host. Template: Kubernetes cluster state by HTTP Monitored by Proxy: Criado no passo anterior.



Inclusão da Secret para monitoria, esta chave é a que foi capturada no passo 8.

Dados Coletados: