



Guia completo — Monitorar Kubernetes “puro” (kubeadm) no Zabbix 7.4

 **Objetivo:** coletar estado do **cluster**, **nós**, **kubelet** e **control plane** via templates oficiais “by HTTP”, usando **ServiceAccount + RBAC**, **token** longa duração e **descobertas (LLD)** que criam os hosts automaticamente.

Resumo

- Criaremos **ServiceAccount** e **RBAC** mínimos para leitura.
- Emitiremos **token** de longa duração (lab).
- No Zabbix, criaremos 2 **hosts base** (dummy) e vincularemos os **templates de Kubernetes**.
- Ajustaremos **macros** ( **sem /api** no macro de URL).
- Forçaremos a **LLD** (“Check now”), validaremos dados e afinaremos ruído.
- Incluo **dashboard**, **notificações** e **troubleshooting** (ex.: nó NotReady).

Sumário

- [Topologia do ambiente](#)
- [O que será monitorado \(visão geral\)](#)
- [Pré-requisitos rápidos](#)
- [1\) RBAC: ServiceAccount + permissões](#)
- [2\) Token longa duração e testes de API](#)
- [3\) Zabbix: hosts base, templates e macros](#)
- [4\) Forçar a descoberta \(LLD\) e validar](#)
- [5\) Ajustes úteis \(ruído, dashboard, notificações\)](#)
- [6\) Troubleshooting \(erros comuns\)](#)
- [FAQ](#)
- [Checklist final](#)
- [Referências rápidas](#)

Topologia do ambiente

- **Cluster kubeadm v1.30.14**
- **Nós**
 - **k8s-cp1** — 192.168.31.40 (control-plane)
 - **k8s-w1** — 192.168.31.41 (worker)
 - **k8s-w2** — 192.168.31.42 (worker)
- **SO:** Rocky Linux 10 nos três
- **Zabbix 7.4** (Server + Frontend)

- **Zabbix agent** já instalado nos nós

O que será monitorado (visão geral)

Componente	Itens/alertas principais
Cluster/Objetos	Pods Pending , CrashLoopBackOff , Restarts elevados; Deployments abaixo do desired; Jobs/CronJobs falhando; Namespaces; Services/Endpoints
Control plane	Saúde do API Server, Scheduler, Controller-Manager (/readyz/healthz)
Nó (Node)	Condições (Ready , MemoryPressure , DiskPressure , etc.), capacidade/allocatable, contagem de pods
Kubelet	Coleta via HTTP proxy pelo API server (sem abrir porta 10250 externamente)
Sistema (Linux)	CPU, RAM, FS, rede via “Linux by Zabbix agent” (aplicado automaticamente aos nós descobertos)

Pré-requisitos rápidos

- **kubectl** funcional no **k8s-cp1** (ou em qualquer host com acesso ao API).
- Zabbix Server alcança **https://192.168.31.40:6443** pela rede.
- **firewalld** **desativado** ou regra liberada (no seu ambiente está inativo).

```
kubectl version
kubectl cluster-info
kubectl get nodes -o wide
```

1) RBAC: ServiceAccount + permissões

Crie namespace, ServiceAccount e **ClusterRole/Binding** com leitura mínima (inclui **nodes**, recursos core e apps/batch, EndpointSlices, Ingress e metrics.k8s.io opcional).

```
vim zbx-rbac.yaml
```

```
apiVersion: v1
kind: Namespace
metadata:
  name: monitoring
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: zabbix-service-account
```

```
namespace: monitoring
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: zabbix-k8s-read
rules:
  - apiGroups: ["" ]
    resources:
[nodes,namespaces,pods,services,endpoints,persistentvolumeclaims,persistentvolum
es,events]
    verbs: [get,list,watch]
  - apiGroups: ["apps"]
    resources: [deployments,daemonsets,statefulsets,replicasets]
    verbs: [get,list,watch]
  - apiGroups: ["batch"]
    resources: [jobs,cronjobs]
    verbs: [get,list,watch]
  - apiGroups: ["discovery.k8s.io"]
    resources: [endpointslices]
    verbs: [get,list,watch]
  - apiGroups: ["networking.k8s.io"]
    resources: [ingresses]
    verbs: [get,list,watch]
  - apiGroups: ["metrics.k8s.io"]
    resources: [nodes,pods]
    verbs: [get,list,watch]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: zabbix-k8s-read-binding
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: zabbix-k8s-read
subjects:
  - kind: ServiceAccount
    name: zabbix-service-account
    namespace: monitoring
```

```
kubectl apply -f zbx-rbac.yaml
kubectl auth can-i get nodes --as=system:serviceaccount:monitoring:zabbix-
service-account
kubectl auth can-i list pods -A --as=system:serviceaccount:monitoring:zabbix-
service-account
```

*Ambos devem responder **yes**.*

2) Token longa duração e testes de API

Gere token **válido por 1 ano** (laboratório):

```
kubectl -n monitoring create token zabbix-service-account --duration=8760h > /tmp/kube.zbx.token  
export TOKEN=$(cat /tmp/kube.zbx.token)
```

Teste **endpoint base e nodes**:

```
curl -sk -H "Authorization: Bearer $TOKEN" https://192.168.31.40:6443/api | jq .  
curl -sk -H "Authorization: Bearer $TOKEN" https://192.168.31.40:6443/api/v1/nodes | jq .
```

Se listar os nós, **RBAC + token + API** estão ok.

3) Zabbix: hosts base, templates e macros

Os templates “Kubernetes ... by HTTP” usam **hosts base** (dummy) para guardar macros e executar a LLD. Eles **criam** automaticamente os hosts reais (nós e componentes) e aplicam “Linux by Zabbix agent”.

3.1) Templates a usar (Zabbix 7.4)

- **Kubernetes nodes by HTTP**
- **Kubernetes cluster state by HTTP** (*Os de API Server/Scheduler/Controller-Manager/Kubelet serão aplicados automaticamente às entidades descobertas.*)

3.2) Criar hosts base (dummy)

Crie **dois hosts** (nomes sugeridos; use os seus se preferir):

- **Host:** `k8s-nodes`
 - **Interface:** Agent `127.0.0.1` (dummy)
 - **Template:** `Kubernetes nodes by HTTP`
- **Host:** `k8s-cluster-state`
 - **Interface:** Agent `127.0.0.1` (dummy)
 - **Template:** `Kubernetes cluster state by HTTP`

3.3) Macros (importante!)

⚠ **Não adicione `/api` no macro de URL** se a descrição do seu template indicar formato `<scheme>://<host>:<port>`. O template monta os caminhos internamente.

No **k8s-nodes** e no **k8s-cluster-state**:

- `{KUBE.API.URL}` = `https://192.168.31.40:6443` (base sem `/api`)
- `{KUBE.API.TOKEN}` = cole o conteúdo de `/tmp/kube.zbx.token`

Dica: em **Host** → **Macros** → “**Inherited and host macros**” verifique o valor efetivo.

4) Forçar a descoberta (LLD) e validar

4.1) “Check now” nas regras de LLD

No Zabbix Frontend:

- **Configuration** → **Hosts** → **k8s-nodes** → aba “**Discovery rules**” → clique **Check now** na regra (ex.: “Kubernetes nodes: discovery”).
- **Configuration** → **Hosts** → **k8s-cluster-state** → aba “**Discovery rules**” → **Check now** nas regras de cluster/componentes.

Para acelerar, você pode recarregar o cache de config do Server:

```
sudo -u zabbix zabbix_server -R config_cache_reload
```

4.2) Validar

- **Configuration** → **Hosts**: devem surgir **k8s-cp1**, **k8s-w1**, **k8s-w2** (normalmente já com **Linux by Zabbix agent** linkado). Ajuste a **interface/IP** de cada host para o IP real.

```
zabbix_get -s 192.168.31.40 -k agent.ping
zabbix_get -s 192.168.31.41 -k agent.ping
zabbix_get -s 192.168.31.42 -k agent.ping
```

- **Monitoring** → **Latest data**: devem aparecer itens de **nodes/pods/deployments/jobs**.

4.3) (Opcional) Item HTTP de teste (sanidade)

```
# No host base "k8s-nodes", crie:
# Type: HTTP agent
# Name: TEST /api/v1/nodes
# URL: {KUBE.API.URL}/api/v1/nodes
# Header: Authorization: Bearer {KUBE.API.TOKEN}
# Verify peer/host: No
```

Se esse item retorna JSON, o token/macro estão corretos.

5) Ajustes úteis (ruído, dashboard, notificações)

5.1) Filtrar ruído por namespace (LLD)

Em **k8s-cluster-state** → **Discovery rules**, abra as regras de pods/deployments e inclua um **Filter** por **{#NAMESPACE}**:

```
^(kube-system|default|ingress-nginx)$
```

(Depois amplie conforme for instalando componentes.)

5.2) Dashboard rápido

- **Monitoring** → **Dashboards** → **Create** (“K8s — Visão Geral”)
 - **Problems** (Group = Kubernetes)
 - **Top** → “Pods com mais Restarts”
 - **Top** → “Nodes NotReady (últimas 3h)”
 - **Graph (classic)** → itens do **k8s-cluster-state** (ex.: “Pods por fase”, “Deployments sem réplicas desejadas”)

5.3) Notificações

Crie **Action** para severidades *High/Critical* com condição **Group = Kubernetes** (ou Tag = kubernetes) e envie para seu e-mail/Teams.

6) Troubleshooting (erros comuns)

6.1) 403 “cannot get resource ‘v1’...”

- Macro de URL com **/api** por engano. **Fix:** `{KUBE.API.URL}` = `https://192.168.31.40:6443` (sem **/api**).
- Token expirado: gere novo e cole na macro.

```
kubectl -n monitoring create token zabbix-service-account --duration=8760h > /tmp/kube.zbx.token
```

6.2) LLD não cria nada

- **Check now** na LLD e veja “Show latest error”.
- Verifique **RBAC**:

```
kubectl auth can-i get nodes --as=system:serviceaccount:monitoring:zabbix-service-account
```

```
kubectl auth can-i list pods -A --as=system:serviceaccount:monitoring:zabbix-
service-account
```

- Teste API com `curl` (URL/macro + token):

```
curl -sk -H "Authorization: Bearer $TOKEN" https://192.168.31.40:6443/api | jq .
curl -sk -H "Authorization: Bearer $TOKEN"
https://192.168.31.40:6443/api/v1/nodes | jq .
```

6.3) Nó `NotReady`

No nó afetado (ex.: `k8s-w2`):

```
kubectl describe node k8s-w2 | sed -n '1,200p'
kubectl -n kube-system get pods -o wide
sudo journalctl -u kubelet -b --no-pager | tail -n 200
```

Ajustes comuns (containerd + cgroup + sysctl):

```
# containerd: habilitar SystemdCgroup=true
sudo test -f /etc/containerd/config.toml || sudo containerd config default |
sudo tee /etc/containerd/config.toml >/dev/null
sudo vim /etc/containerd/config.toml
# [plugins."io.containerd.grpc.v1.cri".containerd.runtimes.runc.options]
# SystemdCgroup = true
sudo systemctl restart containerd kubelet

# sysctl p/ rede
sudo bash -lc 'cat >/etc/sysctl.d/99-k8s.conf <<EOF
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1
EOF'
sudo sysctl --system

# swap off
sudo swapoff -a
sudo sed -r -i 's/^([#].*\sswap\s)/#\1/' /etc/fstab
sudo systemctl restart kubelet
```

6.4) TLS/Certificados

Para teste use `-k` no `curl`. Em produção, importe a CA do cluster no host do Zabbix (ou habilite verificação nos itens HTTP).

FAQ

Posso usar Zabbix Proxy dentro do cluster? Sim. Ajuda a reduzir latência/tráfego e estabiliza a LLD. Pode ser instalado via **Helm**.

Preciso abrir a porta 10250 (kubelet)? Não. O template “Kubelet by HTTP” acessa via **proxy do API server** usando o mesmo token.

Quero `kubectl top`/métricas. Instale **metrics-server**; os templates funcionam sem ele, mas o `top` e métricas `k8s.io` ficam disponíveis.

Checklist final

- ☐ RBAC aplicado (`zabbix-k8s-read` + binding)
 - ☐ `kubectl auth can-i get/list nodes = yes`
 - ☐ Token criado com `--duration=8760h` e **colado** nas macros
 - ☐ **Hosts base** criados: `k8s-nodes` (Nodes by HTTP) e `k8s-cluster-state` (Cluster state by HTTP)
 - ☐ **Macros:** `{KUBE.API.URL}=https://192.168.31.40:6443 (⚠ sem /api) + {KUBE.API.TOKEN}`
 - ☐ **LLD:** “Check now” executado sem erros
 - ☐ Hosts dos nós (`k8s-cp1`, `k8s-w1`, `k8s-w2`) criados com **Linux by Zabbix agent**
 - ☐ Dashboard com **Problems/Top/Graph (classic)**
 - ☐ Action de **notificações** ativa
-

Referências rápidas

```
# Ver versão/saúde/states
kubectl version
kubectl cluster-info
kubectl get nodes -o wide

# RBAC (validação rápida)
kubectl auth can-i get nodes --as=system:serviceaccount:monitoring:zabbix-
service-account
kubectl auth can-i list pods -A --as=system:serviceaccount:monitoring:zabbix-
service-account

# Token (1 ano)
kubectl -n monitoring create token zabbix-service-account --duration=8760h >
/tmp/kube.zbx.token
export TOKEN=$(cat /tmp/kube.zbx.token)

# Teste API
curl -sk -H "Authorization: Bearer $TOKEN" https://192.168.31.40:6443/api | jq .
curl -sk -H "Authorization: Bearer $TOKEN"
https://192.168.31.40:6443/api/v1/nodes | jq .

# Forçar LLD e recarregar config
sudo -u zabbix zabbix_server -R config_cache_reload
```



```
# Sanidade do agent
zabbix_get -s 192.168.31.40 -k agent.ping
zabbix_get -s 192.168.31.41 -k agent.ping
zabbix_get -s 192.168.31.42 -k agent.ping
```

Criado por Jeferson Salles LinkedIn: <https://www.linkedin.com/in/jmsalles/> E-mail: jefersonmattossalles@gmail.com.