SORINT SIRCLE I&DM - CASE STUDY

INTEGRATION BETWEEN VEEAM'S KASTEN K10 AND ZABBIX

NEXT GENERATION SYSTEM INTEGRATOR

Italy | Spain | UK | Germany | France | USA

www.sorintlab.it

DOCUMENT INFORMATION

COPYRIGHT

SUMMARY

1. INTRODUCTION

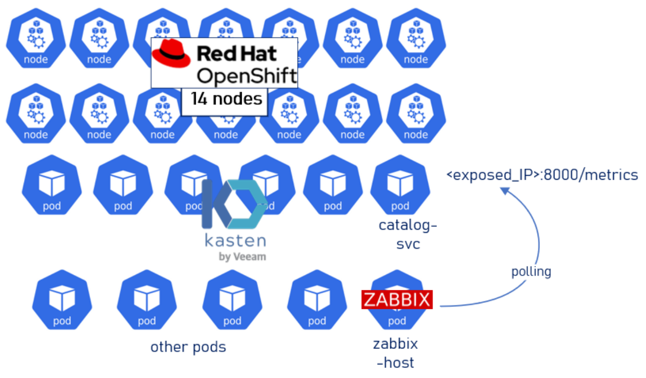
Kasten K10 represents Veeam's solution for Micro-services backup. The Kasten K10 by Veeam data management platform, purpose-built for Kubernetes, provides enterprise operations teams an easy-to-use, scalable, and secure system for backup/restore, disaster recovery, and mobility of Kubernetes applications.

For monitoring purposes, Kasten K10 solution provides an internal metrics solution through the use of Grafana – starting from K10 version 4.5 -. However, Kasten K10 does not natively integrate with third-party monitoring solutions through the use of standard monitoring protocols such as SNMP, e-mail, or external in-guest agents. It is worth mentioning that Kasten K10 is composed of a Prometheus deployment, which can be leveraged to integrate with third-party monitoring solutions through the exposure of Prometheus' metrics.

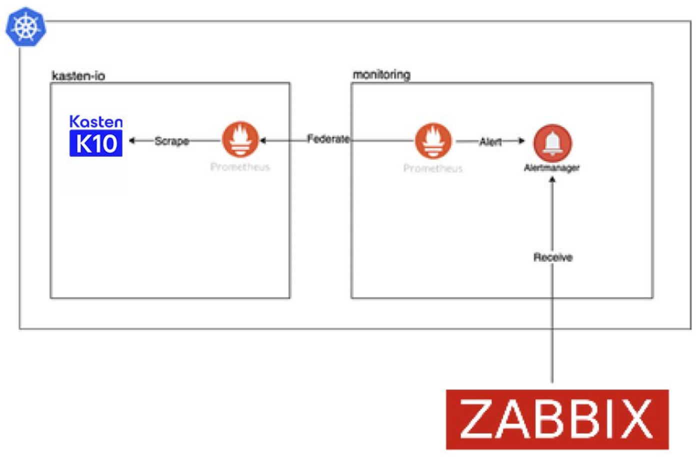
A popular monitoring product, 100% open source, is Zabbix. Zabbix provides a direct integration with Prometheus, but only through the use of a "exporter" tool, which however is intended for virtual machines and not for micro services, like the environment where Kasten K10 is needed.

Typically, in order to get metrics data from Prometheus, one between two options must be chosen:

- **Option A:** Prometheus can be exposed as a service to directly get the metrics through its catalogs. **This option is the easiest setup.**



- **Option B**: Federated - There are different use cases for federation. Commonly, it is used to either achieve scalable Prometheus monitoring setups or to pull related metrics from one service's Prometheus into another. The second use case is the reason why federation is a good option. Data is then scraped from Kasten K10 Prometheus - Monitoring — K10 documentation (kasten.io) -. **This option is the most flexible and clean setup.**



The option described further in this document is the exposure of Prometheus metrics as a service – section 3. In case the option of Prometheus Federation has been chosen, skip the 3rd section.

2. PREREQUISITES

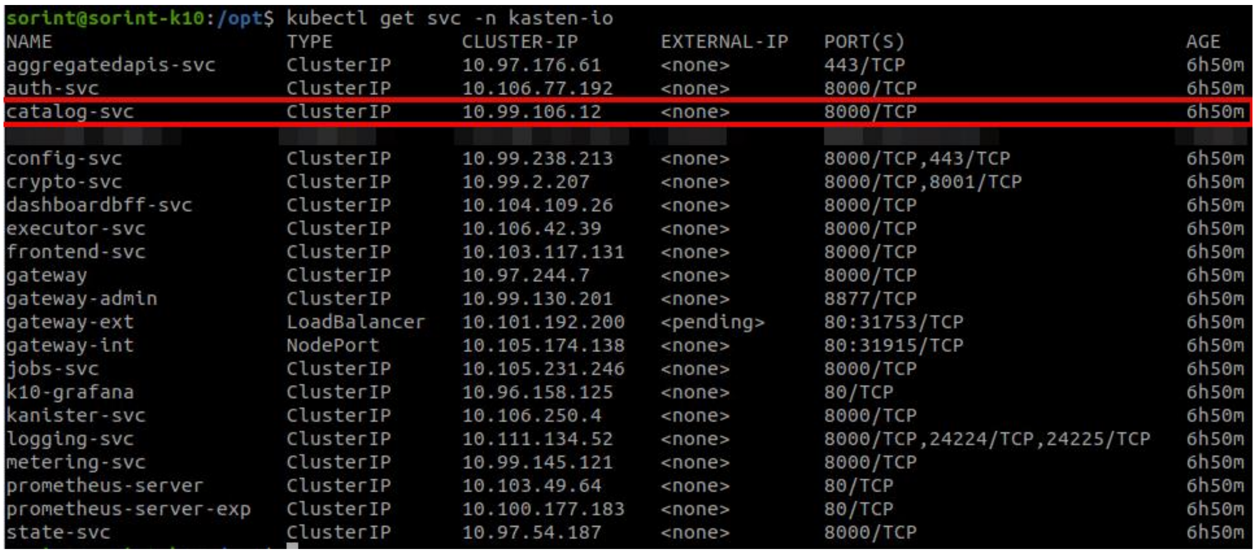
To integrate with Kasten K10's Prometheus:

- Zabbix requires at least version 4.2 (Zabbix 4.2 — Prometheus Integration - Zabbix Blog)
- If Option A has been chosen:
    - There is the need to expose at least one Kasten Service (catalog-svc) to the Zabbix host that will be used to gather metrics.
    - A Zabbix-host (even the master one) will need to access the exposed service / Prometheus Federated instance to gather the required metrics
- If option B has been chosen:
    - An already installed Prometheus instance
    - The Prometheus instance shall be able to communicate with Kasten K10 dashboard in order to get federated metrics
    - The Prometheus instance shall be able to reachable from Zabbix
- Zabbix Admin credentials
- **Warning:** Zabbix version 5.2.16 does not support the first() function 1 Supported trigger functions [Zabbix Documentation 5.2] while Zabbix 5.4 does 4 History functions [Zabbix Documentation 5.4]

3. EXPOSE KASTEN K10'S PROMETHEUS METRICS

3.1. Option A: Expose Catalog svc on K8s Cluster

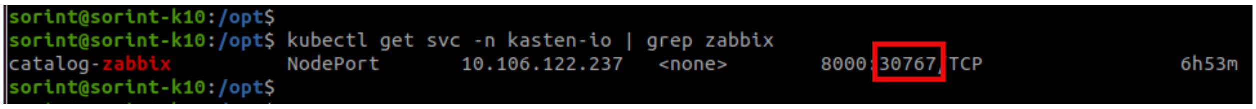
Check that the Kubernetes service "catalog-svc" is active and present.



```
sorint@sorint-k10:/opt$ kubectl get svc -n kasten-io
NAME                    TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)                           AGE
aggregatedapis-svc      ClusterIP      10.97.176.61     <none>        443/TCP                           6h50m
auth-svc                ClusterIP      10.106.77.192    <none>        8000/TCP                          6h50m
catalog-svc             ClusterIP      10.99.106.12     <none>        8000/TCP                          6h50m
[redacted]
config-svc              ClusterIP      10.99.238.213    <none>        8000/TCP,443/TCP                  6h50m
crypto-svc              ClusterIP      10.99.2.207      <none>        8000/TCP,8001/TCP                 6h50m
dashboardbff-svc        ClusterIP      10.104.109.26    <none>        8000/TCP                          6h50m
executor-svc            ClusterIP      10.106.42.39     <none>        8000/TCP                          6h50m
frontend-svc            ClusterIP      10.103.117.131   <none>        8000/TCP                          6h50m
gateway                 ClusterIP      10.97.244.7      <none>        8000/TCP                          6h50m
gateway-admin           ClusterIP      10.99.130.201    <none>        8877/TCP                          6h50m
gateway-ext             LoadBalancer   10.101.192.200   <pending>     80:31753/TCP                      6h50m
gateway-int             NodePort       10.105.174.138   <none>        80:31915/TCP                      6h50m
jobs-svc                ClusterIP      10.105.231.246   <none>        8000/TCP                          6h50m
k10-grafana             ClusterIP      10.96.158.125    <none>        80/TCP                            6h50m
kanister-svc            ClusterIP      10.106.250.4     <none>        8000/TCP                          6h50m
logging-svc             ClusterIP      10.111.134.52    <none>        8000/TCP,24224/TCP,24225/TCP      6h50m
metering-svc            ClusterIP      10.99.145.121    <none>        8000/TCP                          6h50m
prometheus-server       ClusterIP      10.103.49.64     <none>        80/TCP                            6h50m
prometheus-server-exp   ClusterIP      10.100.177.183   <none>        80/TCP                            6h50m
state-svc               ClusterIP      10.97.54.187     <none>        8000/TCP                          6h50m
```

Here are a couple of examples which we used in our lab (vanilla Kubernetes cluster, only one server, master untainted):

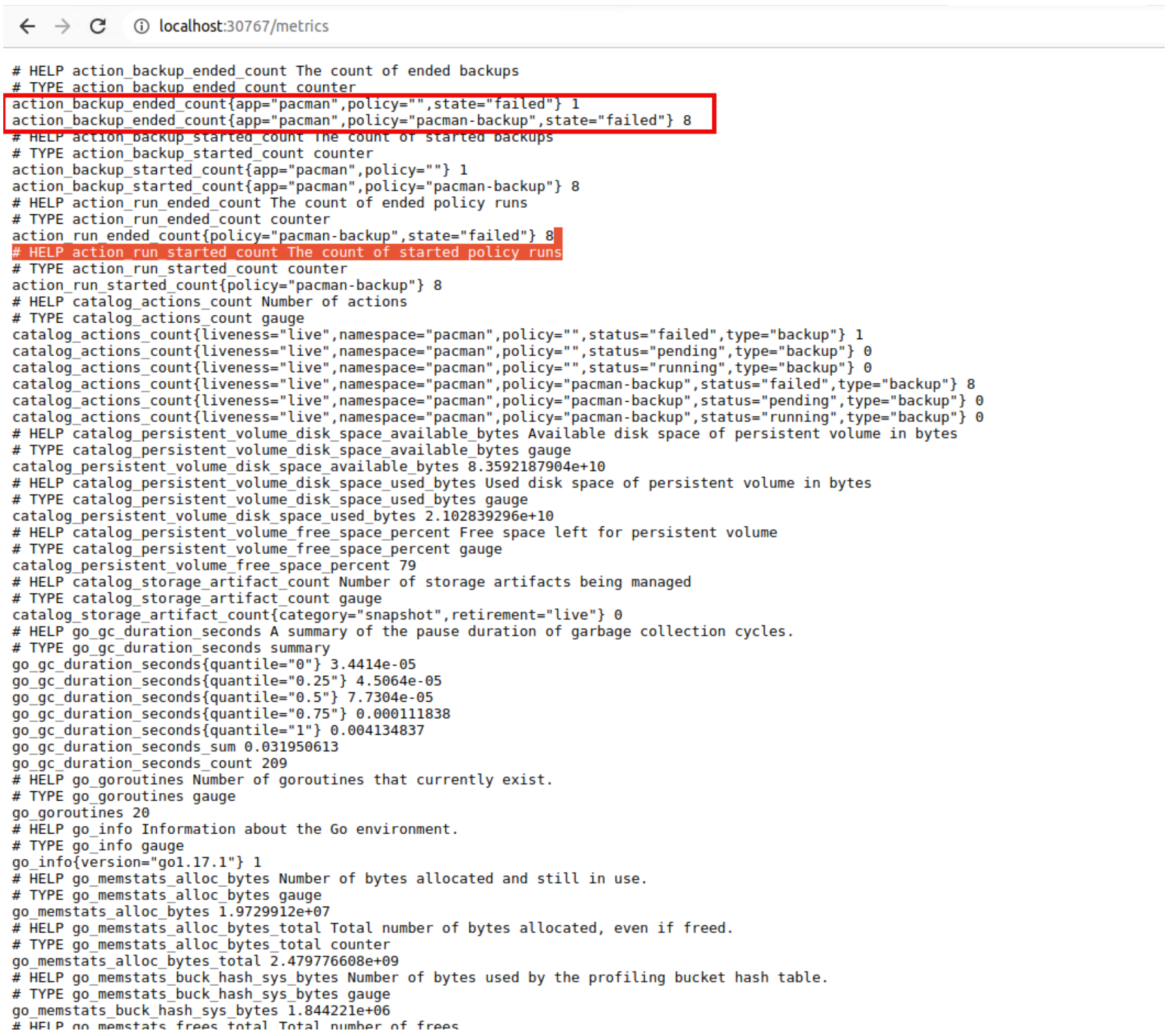
- Exposing the service as NodePort (via node IP):
  *kubectl expose svc catalog-svc --name catalog-zabbix --type=**NodePort** -n kasten-io*
  The server with the Zabbix agent must be able to reach the target machine on the ports highlighted by the command
  kubectl get svc -n kasten-io | grep <exposed_svc>



```
sorint@sorint-k10:/opt$
sorint@sorint-k10:/opt$ kubectl get svc -n kasten-io | grep zabbix
catalog-zabbix          NodePort       10.106.122.237   <none>        8000:30767/TCP                6h53m
sorint@sorint-k10:/opt$
```

- Exposing the service as LoadBalancer:
  *kubectl expose svc catalog-svc --name catalog-zabbix --type=**LoadBalancer** -n kasten-io*

Check from the browser, if possible, from the machine with Zabbix agent, that the metrics of interest are displayed at the URL http://<IP_LoadBalancer_or_NodeIP>:<Port>/metrics



localhost:30767/metrics

```
# HELP action_backup_ended_count The count of ended backups
# TYPE action_backup_ended_count counter
action_backup_ended_count{app="pacman",policy="",state="failed"} 1
action_backup_ended_count{app="pacman",policy="pacman-backup",state="failed"} 8
# HELP action_backup_started_count The count of started backups
# TYPE action_backup_started_count counter
action_backup_started_count{app="pacman",policy=""} 1
action_backup_started_count{app="pacman",policy="pacman-backup"} 8
# HELP action_run_ended_count The count of ended policy runs
# TYPE action_run_ended_count counter
action_run_ended_count{policy="pacman-backup",state="failed"} 8
# HELP action_run_started_count The count of started policy runs
# TYPE action_run_started_count counter
action_run_started_count{policy="pacman-backup"} 8
# HELP catalog_actions_count Number of actions
# TYPE catalog_actions_count gauge
catalog_actions_count{liveness="live",namespace="pacman",policy="",status="failed",type="backup"} 1
catalog_actions_count{liveness="live",namespace="pacman",policy="",status="pending",type="backup"} 0
catalog_actions_count{liveness="live",namespace="pacman",policy="",status="running",type="backup"} 0
catalog_actions_count{liveness="live",namespace="pacman",policy="pacman-backup",status="failed",type="backup"} 8
catalog_actions_count{liveness="live",namespace="pacman",policy="pacman-backup",status="pending",type="backup"} 0
catalog_actions_count{liveness="live",namespace="pacman",policy="pacman-backup",status="running",type="backup"} 0
# HELP catalog_persistent_volume_disk_space_available_bytes Available disk space of persistent volume in bytes
# TYPE catalog_persistent_volume_disk_space_available_bytes gauge
catalog_persistent_volume_disk_space_available_bytes 8.3592187904e+10
# HELP catalog_persistent_volume_disk_space_used_bytes Used disk space of persistent volume in bytes
# TYPE catalog_persistent_volume_disk_space_used_bytes gauge
catalog_persistent_volume_disk_space_used_bytes 2.102839296e+10
# HELP catalog_persistent_volume_free_space_percent Free space left for persistent volume
# TYPE catalog_persistent_volume_free_space_percent gauge
catalog_persistent_volume_free_space_percent 79
# HELP catalog_storage_artifact_count Number of storage artifacts being managed
# TYPE catalog_storage_artifact_count gauge
catalog_storage_artifact_count{category="snapshot",retirement="live"} 0
# HELP go_gc_duration_seconds A summary of the pause duration of garbage collection cycles.
# TYPE go_gc_duration_seconds summary
go_gc_duration_seconds{quantile="0"} 3.4414e-05
go_gc_duration_seconds{quantile="0.25"} 4.5064e-05
go_gc_duration_seconds{quantile="0.5"} 7.7304e-05
go_gc_duration_seconds{quantile="0.75"} 0.000111838
go_gc_duration_seconds{quantile="1"} 0.004134837
go_gc_duration_seconds_sum 0.031950613
go_gc_duration_seconds_count 209
# HELP go_goroutines Number of goroutines that currently exist.
# TYPE go_goroutines gauge
go_goroutines 20
# HELP go_info Information about the Go environment.
# TYPE go_info gauge
go_info{version="go1.17.1"} 1
# HELP go_memstats_alloc_bytes Number of bytes allocated and still in use.
# TYPE go_memstats_alloc_bytes gauge
go_memstats_alloc_bytes 1.9729912e+07
# HELP go_memstats_alloc_bytes_total Total number of bytes allocated, even if freed.
# TYPE go_memstats_alloc_bytes_total counter
go_memstats_alloc_bytes_total 2.479776608e+09
# HELP go_memstats_buck_hash_sys_bytes Number of bytes used by the profiling bucket hash table.
# TYPE go_memstats_buck_hash_sys_bytes gauge
go_memstats_buck_hash_sys_bytes 1.844221e+06
# HELP go_memstats_frees_total Total number of frees
```

If, on Kubernetes side, all prerequisites are ready, it is possible to proceed to the next phase (Zabbix configuration)

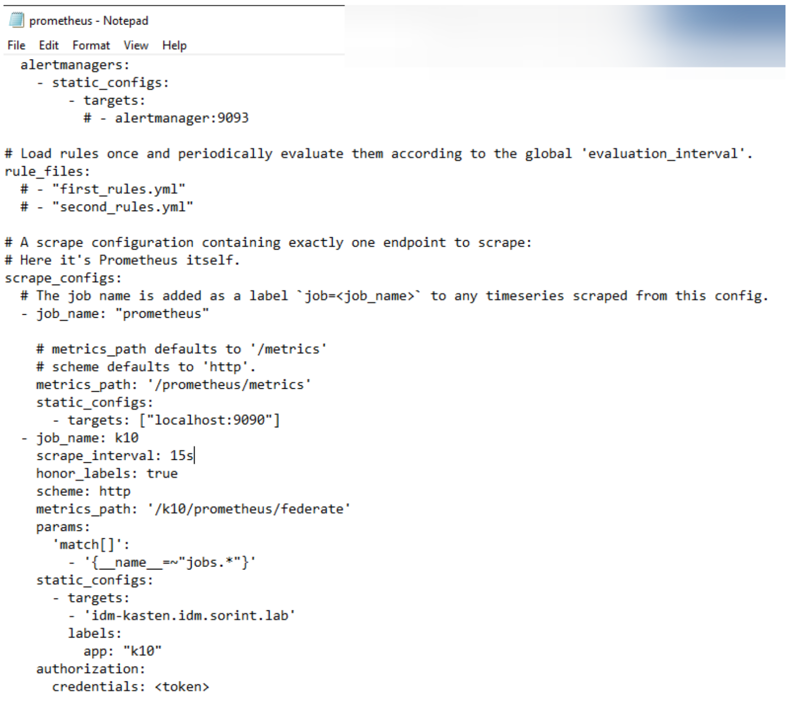
3.2. Option B: Prometheus Federation

Setup the Prometheus instance.

Configure the job to scrap data from Kasten K10 Prometheus:

- Add this job parameters to Prometheus YAML config
  https://docs.kasten.io/latest/operating/monitoring.html#using-k10-s-prometheus-endpoint

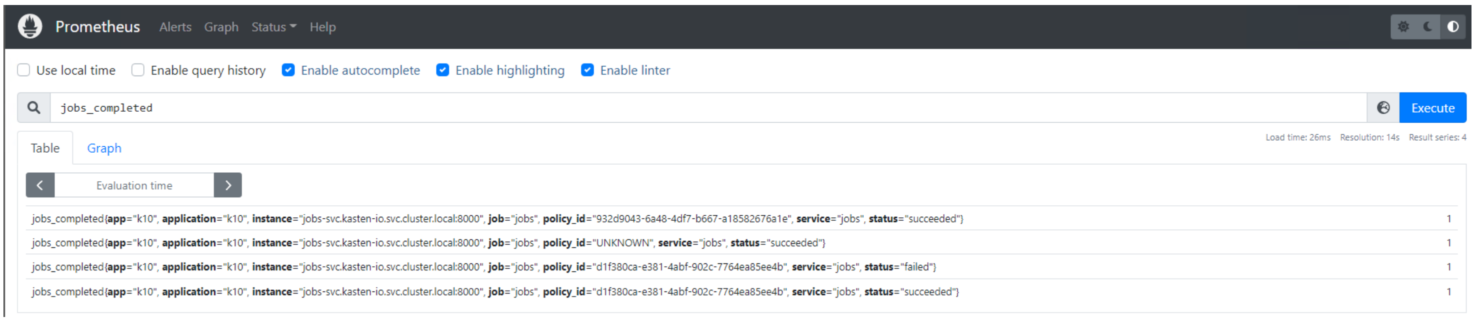
```
- job_name: k10
  scrape_interval: 15s
  honor_labels: true
  scheme: http
  metrics_path: '/k10/prometheus/federate'
  params:
   'match[]':
     - '{__name__=~"jobs.*"}'
  static_configs:
   - targets:
     - '<FQDN for Kasten K10 Prometheus endpoint>'
     labels:
       app: "k10"
  authorization:
   credentials: <token>
```


prometheus - Notepad

```
  alertmanagers:
    - static_configs:
        - targets:
          # - alertmanager:9093

# Load rules once and periodically evaluate them according to the global 'evaluation_interval'.
rule_files:
  # - "first_rules.yml"
  # - "second_rules.yml"

# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs:
  # The job name is added as a label `job=<job_name>` to any timeseries scraped from this config.
  - job_name: "prometheus"

    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.
    metrics_path: '/prometheus/metrics'
    static_configs:
      - targets: ["localhost:9090"]
  - job_name: k10
    scrape_interval: 15s
    honor_labels: true
    scheme: http
    metrics_path: '/k10/prometheus/federate'
    params:
      'match[]':
        - '{__name__=~"jobs.*"}'
    static_configs:
      - targets:
        - 'idm-kasten.idm.sorint.lab'
        labels:
          app: "k10"
    authorization:
      credentials: <token>
```

Check that data is correctly scraped:

- From the dashboard, do a query (example: jobs_completed) and check that data is correcty displayed.



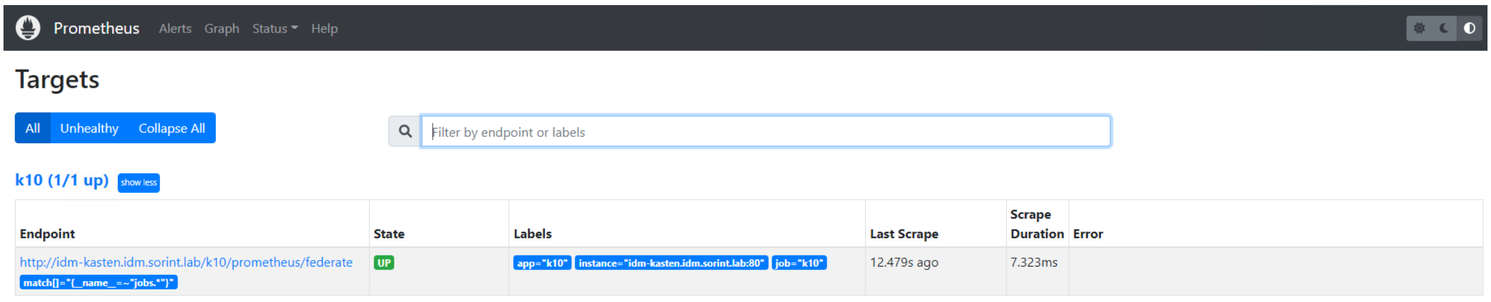
Use local time: ☐ Enable query history: ☐ Enable autocomplete: ✓ Enable highlighting: ✓ Enable linter: ✓

jobs_completed

Load time: 26ms Resolution: 14s Result series: 4

```
jobs_completed{app="k10", application="k10", instance="jobs-svc.kasten-io.svc.cluster.local:8000", job="jobs", policy_id="932d9043-6a48-4df7-b667-a18582676a1e", service="jobs", status="succeeded"}   1
jobs_completed{app="k10", application="k10", instance="jobs-svc.kasten-io.svc.cluster.local:8000", job="jobs", policy_id="UNKNOWN", service="jobs", status="succeeded"}   1
jobs_completed{app="k10", application="k10", instance="jobs-svc.kasten-io.svc.cluster.local:8000", job="jobs", policy_id="d1f380ca-e381-4abf-902c-7764ea85ee4b", service="jobs", status="failed"}   1
jobs_completed{app="k10", application="k10", instance="jobs-svc.kasten-io.svc.cluster.local:8000", job="jobs", policy_id="d1f380ca-e381-4abf-902c-7764ea85ee4b", service="jobs", status="succeeded"}   1
```

- From the Status -> Targets dashboard



Targets

k10 (1/1 up)

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://idm-kasten.idm.sorint.lab/k10/prometheus/federate match[]="{__name__=~"jobs.*"}"	UP	app="k10" instance="idm-kasten.idm.sorint.lab:80" job="k10"	12.479s ago	7.323ms	

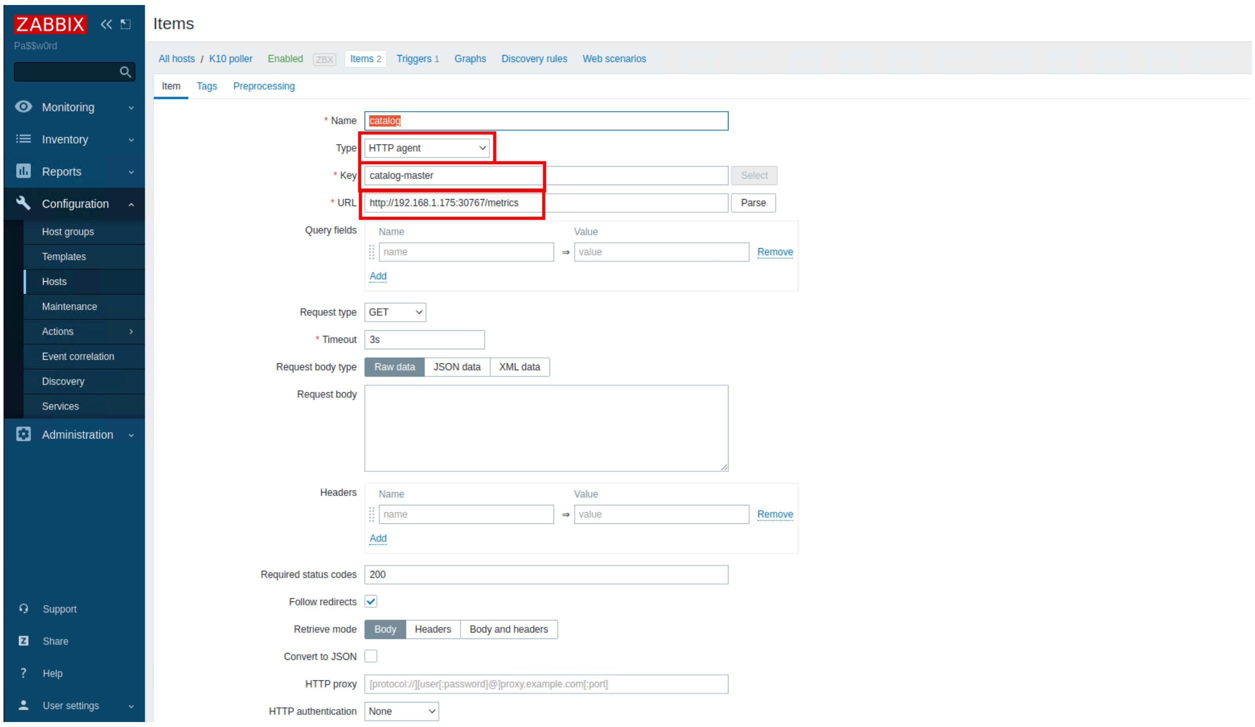
If Prometheus instance is able to correctly pull data, you can proceed with the next step.

4. ZABBIX CONFIGURATION

The following procedures are strongly subjected to the version of Zabbix installed. In case of mismatch of commands, refer to Zabbix documentation to find alternatives to the configuration suggested below.

4.1. ZABBIX SETUP

- Login to Zabbix
- Move to "Configuration -> Hosts"
- It is necessary to create a Zabbix Item that is the "parent" Item. This item will have as given only the http endpoint exposed by the Kubernetes cluster to the Zabbix server
- Zabbix Item creation:
    - **Name:** descriptive item name
    - **Type:** HTTP agent
    - **Key:** catalog-master
    - **URL:** http://<IP_LoadBalancer_o_NodeIP>:<Port>/metrics



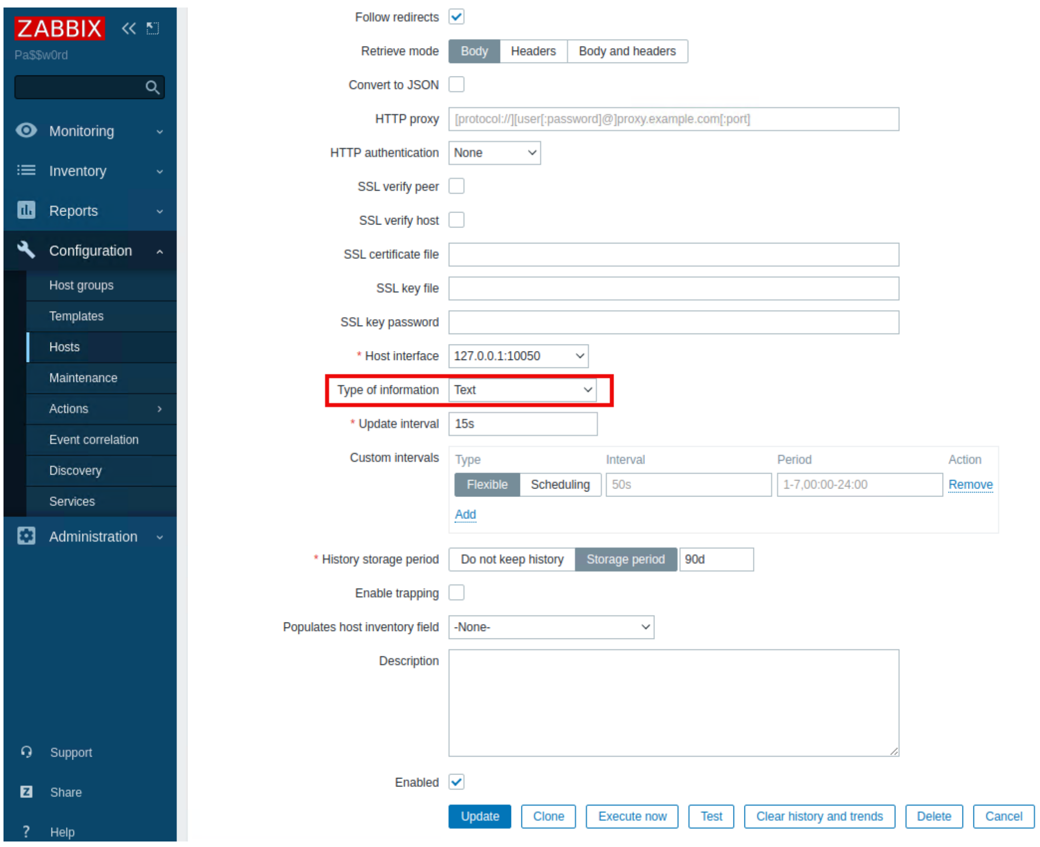
Items

All hosts / K10 poller Enabled ZBX Items 2 Triggers 1 Graphs Discovery rules Web scenarios

Item Tags Preprocessing

* Name	catalog
Type	HTTP agent
* Key	catalog-master
* URL	http://192.168.1.175:30767/metrics
Query fields	Name: name ⇒ Value: value
Request type	GET
* Timeout	3s
Request body type	[✓ Raw data] JSON data XML data
Request body	
Headers	Name: name ⇒ Value: value
Required status codes	200
Follow redirects	✓
Retrieve mode	[✓ Body] Headers Body and headers
Convert to JSON	☐
HTTP proxy	[protocol://][user[:password]@]proxy.example.com[:port]
HTTP authentication	None

    - **Type of information:** Text



Follow redirects: ✓
Retrieve mode: [✓ Body] Headers | Body and headers
Convert to JSON: ☐
HTTP proxy: [protocol://][user[:password]@]proxy.example.com[:port]
HTTP authentication: None
SSL verify peer: ☐
SSL verify host: ☐
SSL certificate file:
SSL key file:
SSL key password:
* Host interface: 127.0.0.1:10050
Type of information: Text
* Update interval: 15s
Custom intervals:

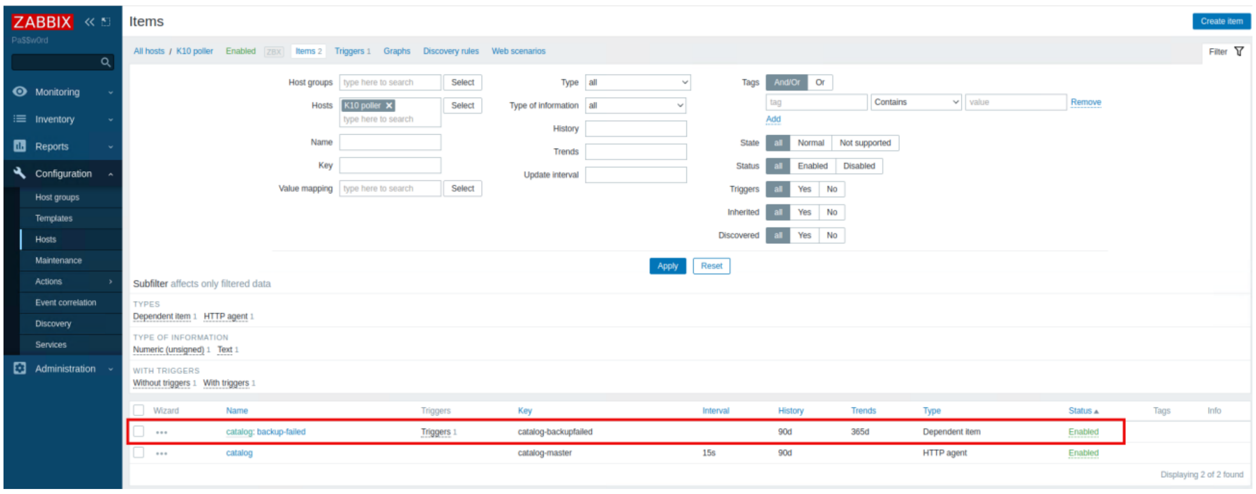
Type	Interval	Period	Action
[✓ Flexible] Scheduling	50s	1-7,00:00-24:00	Remove

Add

* History storage period: Do not keep history [✓ Storage period] 90d
Enable trapping: ☐
Populates host inventory field: -None-
Description:
Enabled: ✓

Update | Clone | Execute now | Test | Clear history and trends | Delete | Cancel

- Click Update.
- We need to create a new Item for the real metrics monitoring



All hosts / K10 poller Enabled ZBX Items 2 Triggers 1 Graphs Discovery rules Web scenarios

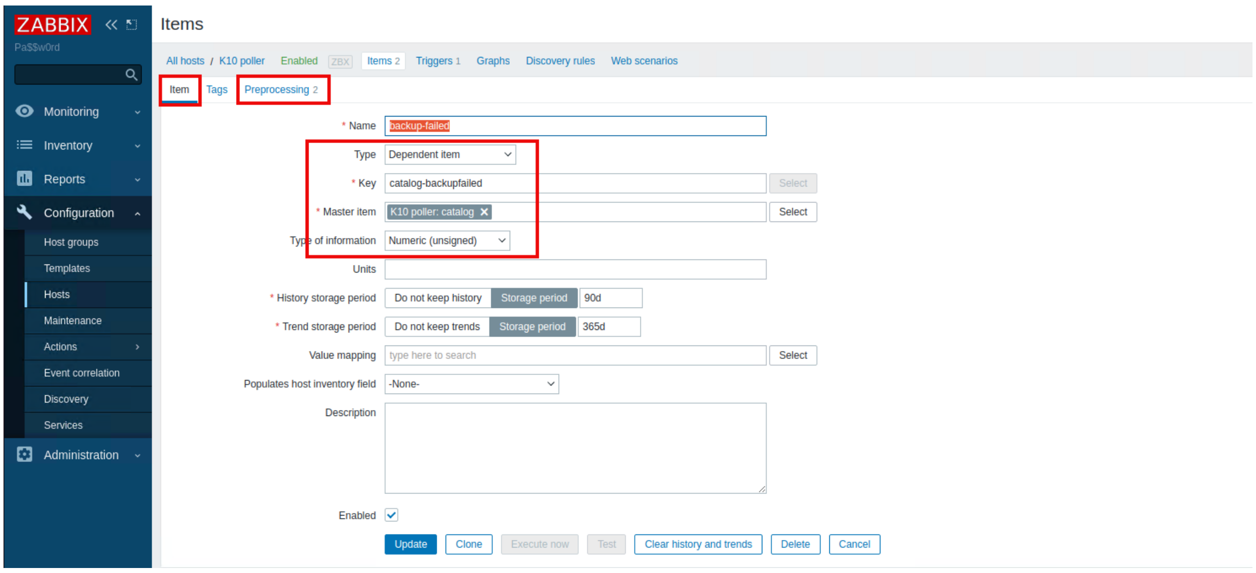
Host groups: type here to search (Select)
Hosts: K10 poller (Select)
Name:
Key:
Value mapping: type here to search (Select)
Type: all
Type of information: all
History:
Trends:
Update interval:
Tags: [✓ And/Or] Or — tag, Contains, value, Remove; Add
State: [✓ all] Normal | Not supported
Status: [✓ all] Enabled | Disabled
Triggers: [✓ all] Yes | No
Inherited: [✓ all] Yes | No
Discovered: [✓ all] Yes | No

Subfilter affects only filtered data
TYPES: Dependent item 1 HTTP agent 1
TYPE OF INFORMATION: Numeric (unsigned) 1 Text 1
WITH TRIGGERS: Without triggers 1 With triggers 1

Wizard	Name	Triggers	Key	Interval	History	Trends	Type	Status ▲	Tags	Info
•••	catalog: backup-failed	Triggers 1	catalog-backupfailed		90d	365d	Dependent item	Enabled		
•••	catalog		catalog-master	15s	90d		HTTP agent	Enabled		

Displaying 2 of 2 found

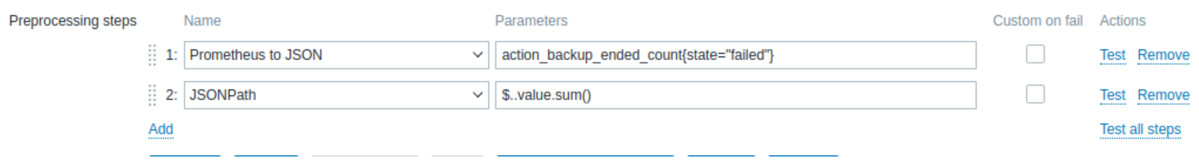
- Create an item with these caratheristics
  - **Type:** Dependent Item
  - **Key:** catalog-backupfailed
  - **Master-**item: <the parent item just created>
  - **Type of information:** Numeric (unsigned)


ZABBIX
Pa$$w0rd
Monitoring
Inventory
Reports
Configuration
Host groups
Templates
Hosts
Maintenance
Actions
Event correlation
Discovery
Services
Administration
Items
All hosts / K10 poller Enabled ZBX Items 2 Triggers 1 Graphs Discovery rules Web scenarios
Item Tags Preprocessing 2

* Name	backup-failed
Type	Dependent item
* Key	catalog-backupfailed
* Master item	K10 poller: catalog
Type of information	Numeric (unsigned)
Units	
* History storage period	Do not keep history / Storage period: 90d
* Trend storage period	Do not keep trends / Storage period: 365d
Value mapping	type here to search
Populates host inventory field	-None-
Description	
Enabled	✓

Update Clone Execute now Test Clear history and trends Delete Cancel

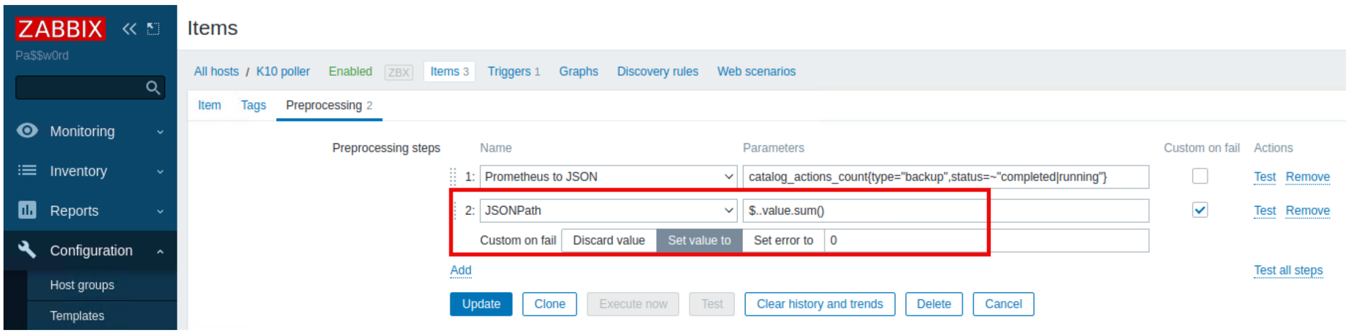
- Move to “preprocessing”
  - Row 1: Prometheus to JSON (min version Zabbix 4.2)
    action_backup_ended_count{state=”failed”}
  - Row 2: insert JSONPath to get the value
    $..value.sum()



Preprocessing steps

	Name	Parameters	Custom on fail	Actions
1:	Prometheus to JSON	action_backup_ended_count{state="failed"}	☐	Test Remove
2:	JSONPath	$..value.sum()	☐	Test Remove

- To avoid errors in case of empty array, flag the **“Custom on fail”** option. Then, select **“Set value to”** and put **0** as value


ZABBIX
Pa$$w0rd
Monitoring
Inventory
Reports
Configuration
Host groups
Templates
Items
All hosts / K10 poller Enabled ZBX Items 3 Triggers 1 Graphs Discovery rules Web scenarios
Item Tags Preprocessing 2

Preprocessing steps

	Name	Parameters	Custom on fail	Actions
1:	Prometheus to JSON	catalog_actions_count{type="backup",status=~"completed|running"}	☐	Test Remove
2:	JSONPath	$..value.sum()	✓	Test Remove
	Custom on fail: Discard value [✓ Set value to] Set error to	0		

Add
Test all steps
Update Clone Execute now Test Clear history and trends Delete Cancel

- Click on Save

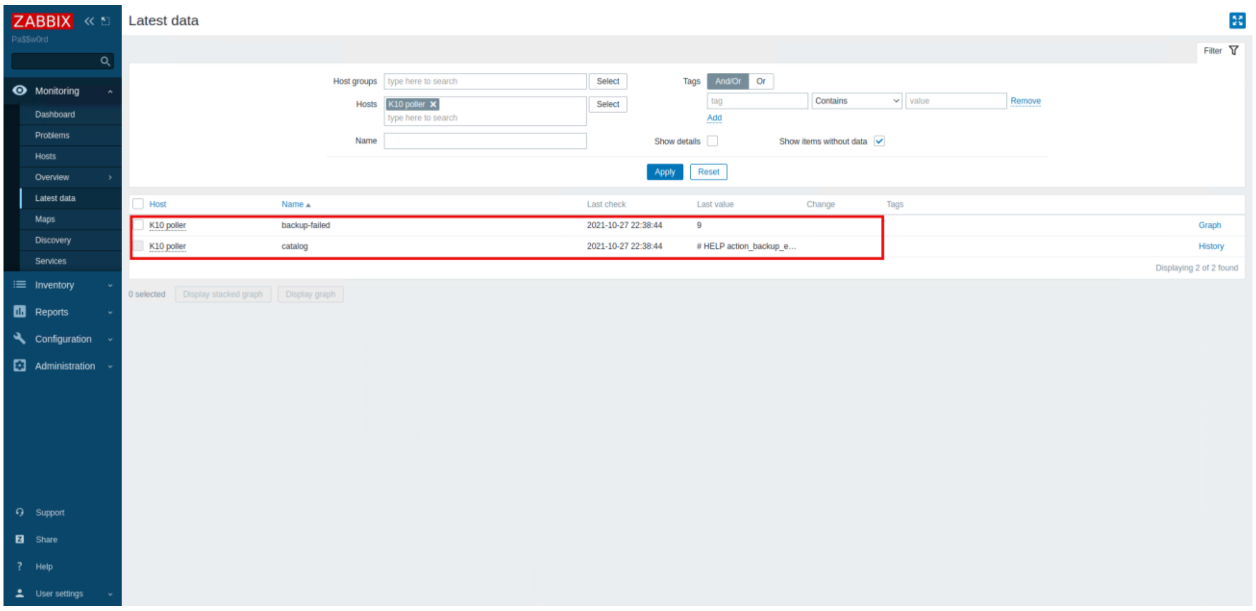
Then create all the items that will be the real object of monitoring. For any other monitoring inputs, follow the following documentation: https://docs.kasten.io/latest/operating/monitoring.html.

Other values of interest can be obtained from the Grafana dashboard integrated in Kasten K10. By clicking on “Edit” under each value (eg Backup Failed), you will find the Prometheus query used to obtain this value. The purpose is to replicate it in the JSON + JSONPath combination.

N.B. use incremental-only monitoring to monitor any growth in the number of failed jobs as the metric is a counter

4.2. Monitoring

Monitor on "Latest data" that the values are actually present. If the fields are blank after several minutes, go to the troubleshooting section.



Latest data

Filter

Host groups: type here to search [Select]
Hosts: K10 poller ✕ [Select]
Name:
Tags: [And/Or] Or — tag | Contains | value | Remove — Add
Show details ☐ Show items without data ✓

Apply Reset

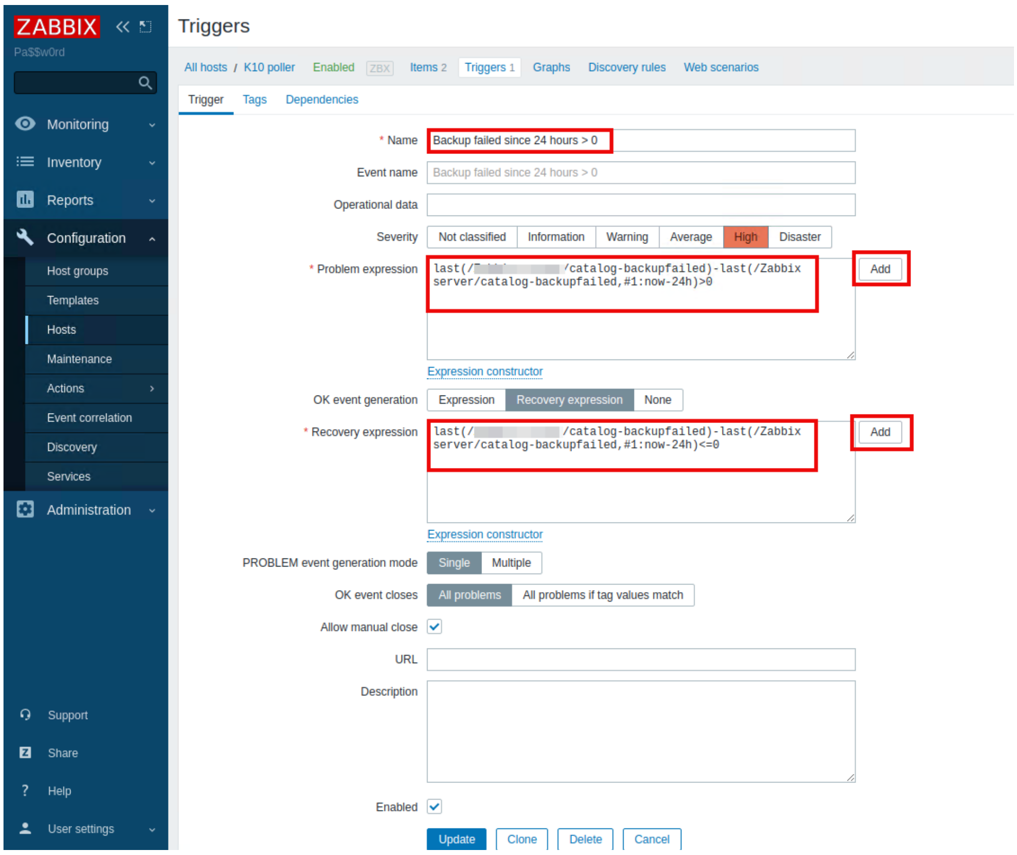
	Host	Name ▲	Last check	Last value	Change	Tags	
☐	K10 poller	backup-failed	2021-10-27 22:38:44	9			Graph
☐	K10 poller	catalog	2021-10-27 22:38:44	# HELP action_backup_e…			History

Displaying 2 of 2 found

0 selected | Display stacked graph | Display graph

4.3. Trigger

Refer to this documentation 1 Configuring a trigger [Zabbix Documentation 5.4]



Triggers

All hosts / K10 poller | Enabled | ZBX | Items 2 | Triggers 1 | Graphs | Discovery rules | Web scenarios

Trigger | Tags | Dependencies

* Name	Backup failed since 24 hours > 0
Event name	Backup failed since 24 hours > 0
Operational data	
Severity	Not classified / Information / Warning / Average / [✓ High] / Disaster
* Problem expression	last(/[redacted]/catalog-backupfailed)-last(/Zabbix server/catalog-backupfailed,#1:now-24h)>0
	Expression constructor
OK event generation	Expression / [✓ Recovery expression] / None
* Recovery expression	last(/[redacted]/catalog-backupfailed)-last(/Zabbix server/catalog-backupfailed,#1:now-24h)<=0
	Expression constructor
PROBLEM event generation mode	[✓ Single] / Multiple
OK event closes	[✓ All problems] / All problems if tag values match
Allow manual close	✓
URL	
Description	
Enabled	✓

Update Clone Delete Cancel

- **Name:** trigger descriptive name
- **Problem expression:** compare the value of the previous X hours with the current value. It must not exceed the value of 0 in the case of failed backups. Example

  (Zabbix version 5.4)

  last(/<zabbix-host>/catalog-backup-failed) - first(/<zabbix-host>/catalog-backup-failed,4h) > 0

  (Zabbix version 5.2)

  last(/<zabbix-host>/catalog-backup-failed) - last(/k10-poller/catalog-backup-failed,#1:now-4h) > 0

  Note: evaluate using the "Add" button for help.
- **Recovery expression:** compare the value of the previous X hours with the current value in order to automatically resolve the incident if the value returns to normal.

  (Zabbix version 5.4)

  last(/<zabbix-host>/catalog-backup-failed) - first(/<zabbix-host>/catalog-backup-failed,4h) <= 0

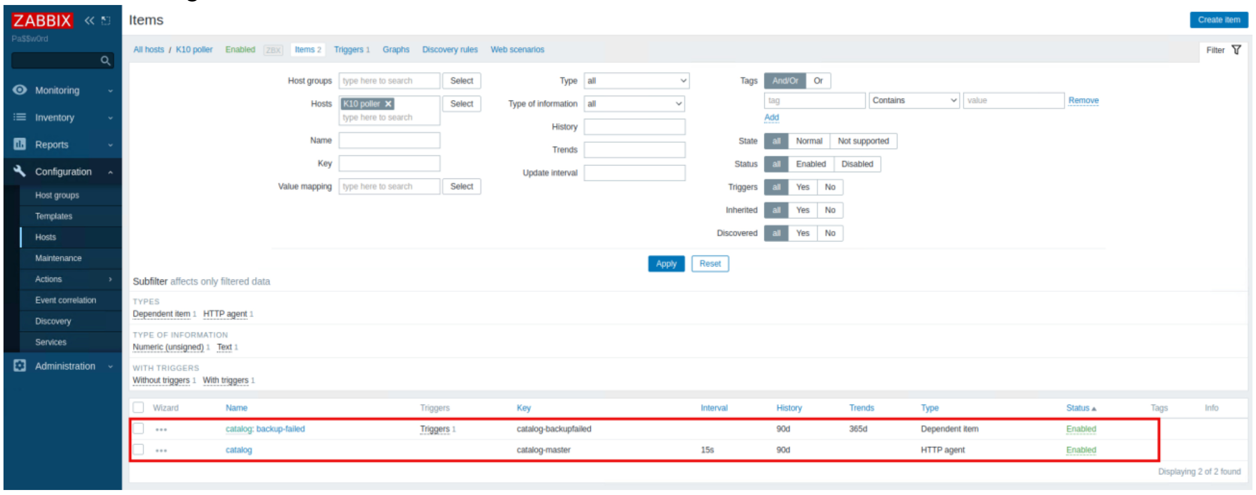
  (Zabbix version 5.2)

  last(/<zabbix-host>/catalog-backup-failed) - last(/k10-poller/catalog-backup-failed,#1:now-4h) <= 0
- **Allow manual close:** customer can decide to flag this or not
- Click on **Save**

5. TROUBLESHOOTING SUGGESTIONS

5.1. ZABBIX ITEM PROBLEMS

- Move to “Configuration -> Hosts”



Host groups: type here to search (Select)
Hosts: K10 poller (Select)
Name:
Key:
Value mapping: type here to search (Select)
Type: all
Type of information: all
History:
Trends:
Update interval:
Tags: [✓ And/Or] Or — tag, Contains, value, Remove, Add
State: [✓ all] Normal Not supported
Status: [✓ all] Enabled Disabled
Triggers: [✓ all] Yes No
Inherited: [✓ all] Yes No
Discovered: [✓ all] Yes No

Subfilter affects only filtered data
TYPES
Dependent item 1 HTTP agent 1
TYPE OF INFORMATION
Numeric (unsigned) 1 Text 1
WITH TRIGGERS
Without triggers 1 With triggers 1

	Wizard	Name	Triggers	Key	Interval	History	Trends	Type	Status ▲	Tags	Info
	•••	catalog: backup-failed	Triggers 1	catalog-backupfailed		90d	365d	Dependent item	Enabled		
	•••	catalog		catalog-master	15s	90d		HTTP agent	Enabled		

Displaying 2 of 2 found

- Click on “Items” and check if there are any [!] shown

In case they are present, if problems are reported with the section of the JSONPath:

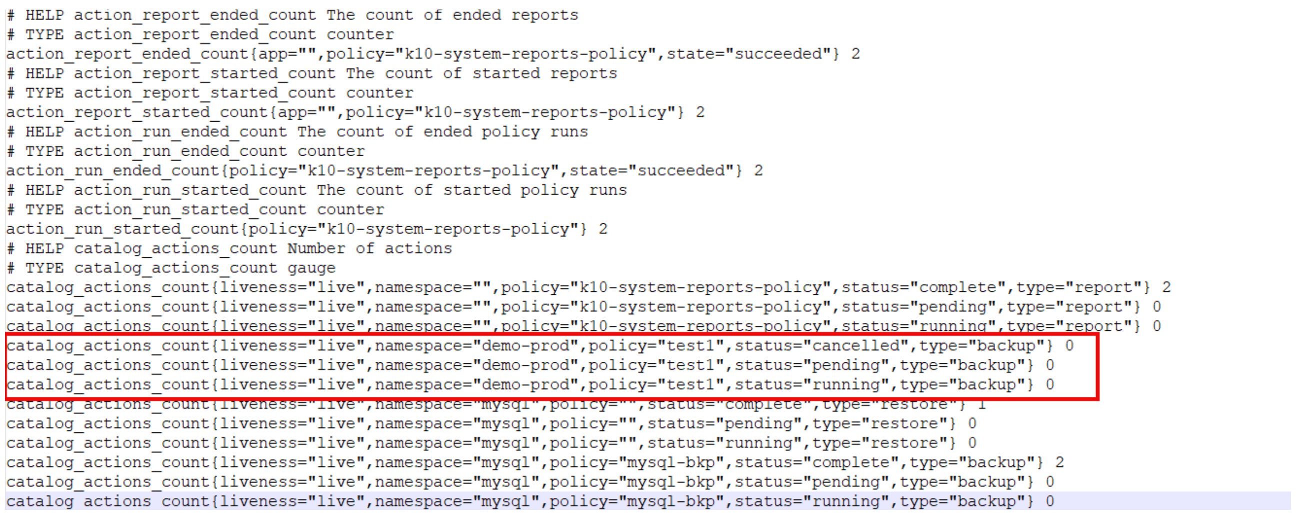
- Modify **Type of Information** of the item in Text
- Leave only the option **“Prometheus to JSON”** under preprocessing, remove the row “**JSONPath”**,
- Move on “Latest Data” to analize the data.
- Copy and paste the output on https://jsonpath.herokuapp.com/
- Test some JSONPaths to find out the correct one. If the JSONPath is correct, you can test again the configuration.

6. APPENDIX A

6.1. METRICS FOR BACKUP JOB STATUS

6.1.1. CREATE AN ITEM FILDERED BY NAMESPACE "DEMO-PROD"

The subsequent metrics will be provided as an example:



```
# HELP action_report_ended_count The count of ended reports
# TYPE action_report_ended_count counter
action_report_ended_count{app="",policy="k10-system-reports-policy",state="succeeded"} 2
# HELP action_report_started_count The count of started reports
# TYPE action_report_started_count counter
action_report_started_count{app="",policy="k10-system-reports-policy"} 2
# HELP action_run_ended_count The count of ended policy runs
# TYPE action_run_ended_count counter
action_run_ended_count{policy="k10-system-reports-policy",state="succeeded"} 2
# HELP action_run_started_count The count of started policy runs
# TYPE action_run_started_count counter
action_run_started_count{policy="k10-system-reports-policy"} 2
# HELP catalog_actions_count Number of actions
# TYPE catalog_actions_count gauge
catalog_actions_count{liveness="live",namespace="",policy="k10-system-reports-policy",status="complete",type="report"} 2
catalog_actions_count{liveness="live",namespace="",policy="k10-system-reports-policy",status="pending",type="report"} 0
catalog_actions_count{liveness="live",namespace="",policy="k10-system-reports-policy",status="running",type="report"} 0
catalog_actions_count{liveness="live",namespace="demo-prod",policy="test1",status="cancelled",type="backup"} 0
catalog_actions_count{liveness="live",namespace="demo-prod",policy="test1",status="pending",type="backup"} 0
catalog_actions_count{liveness="live",namespace="demo-prod",policy="test1",status="running",type="backup"} 0
catalog_actions_count{liveness="live",namespace="mysql",policy="",status="complete",type="restore"} 1
catalog_actions_count{liveness="live",namespace="mysql",policy="",status="pending",type="restore"} 0
catalog_actions_count{liveness="live",namespace="mysql",policy="",status="running",type="restore"} 0
catalog_actions_count{liveness="live",namespace="mysql",policy="mysql-bkp",status="complete",type="backup"} 2
catalog_actions_count{liveness="live",namespace="mysql",policy="mysql-bkp",status="pending",type="backup"} 0
catalog_actions_count{liveness="live",namespace="mysql",policy="mysql-bkp",status="running",type="backup"} 0
```

- Get the number of **failed, pending or cancelled backup** jobs, filtered by **namespace "demo-prod"**

  Prometheus to JSON:

  catalog_actions_count{namespace="demo-prod",status=~"failed|cancelled|pending",type="backup"}

  JSONPath:

  $..value.sum()

- Get the number of **completed** or **running backup** jobs, filtered by **namespace "demo-prod"**

  Prometheus to JSON:

  catalog_actions_count{namespace="demo-prod",status=~"running|completed",type="backup"}

  JSONPath:

  $..value.sum()

- Get the number of **failed, pending or cancelled export** jobs, filtered by **namespace "demo-prod"**

  Prometheus to JSON:

  catalog_actions_count{namespace="demo-prod",status=~"failed|pending|cancelled",type="export"}

  JSONPath:

  $..value.sum()

- Get the number of **completed** or **running export** jobs, filtered by **namespace "demo-prod"**

  Prometheus to JSON:

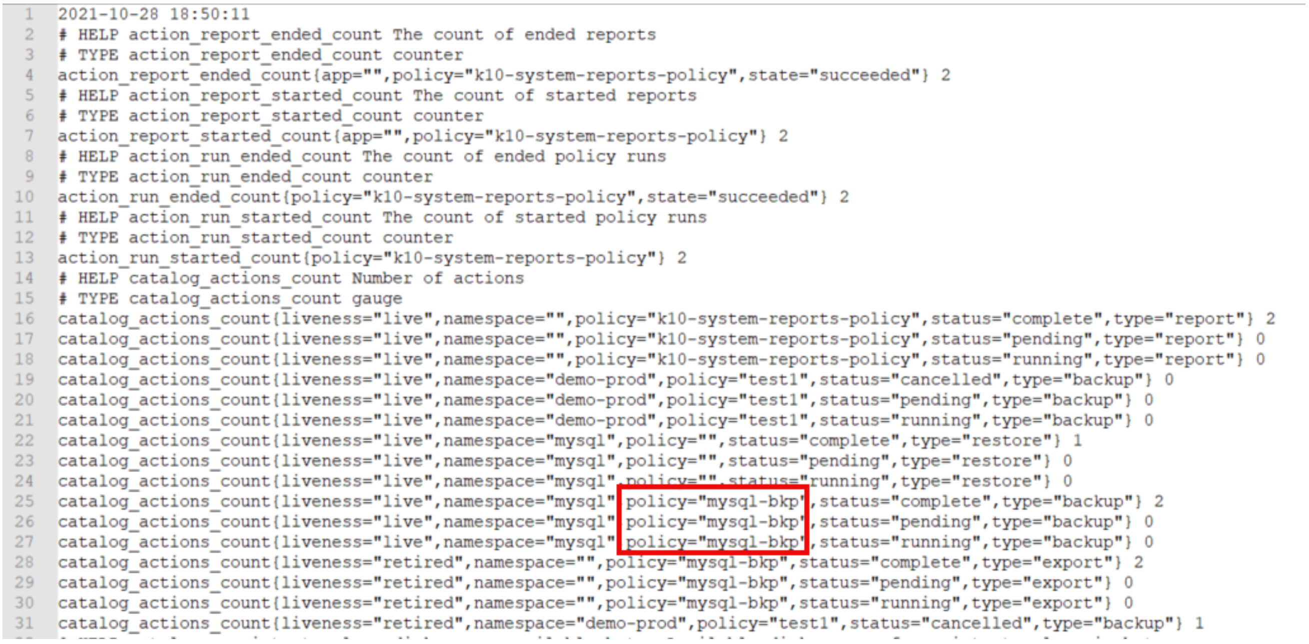
  catalog_actions_count{namespace="demo-prod",status=~"running|completed",type="export"}

  JSONPath:

  $..value.sum()

6.1.2. CREATE AN ITEM FILDERED BY POLICY "MYSQL-BKP"

The subsequent metrics will be provided as an example:



```
2021-10-28 18:50:11
# HELP action_report_ended_count The count of ended reports
# TYPE action_report_ended_count counter
action_report_ended_count{app="",policy="k10-system-reports-policy",state="succeeded"} 2
# HELP action_report_started_count The count of started reports
# TYPE action_report_started_count counter
action_report_started_count{app="",policy="k10-system-reports-policy"} 2
# HELP action_run_ended_count The count of ended policy runs
# TYPE action_run_ended_count counter
action_run_ended_count{policy="k10-system-reports-policy",state="succeeded"} 2
# HELP action_run_started_count The count of started policy runs
# TYPE action_run_started_count counter
action_run_started_count{policy="k10-system-reports-policy"} 2
# HELP catalog_actions_count Number of actions
# TYPE catalog_actions_count gauge
catalog_actions_count{liveness="live",namespace="",policy="k10-system-reports-policy",status="complete",type="report"} 2
catalog_actions_count{liveness="live",namespace="",policy="k10-system-reports-policy",status="pending",type="report"} 0
catalog_actions_count{liveness="live",namespace="",policy="k10-system-reports-policy",status="running",type="report"} 0
catalog_actions_count{liveness="live",namespace="demo-prod",policy="test1",status="cancelled",type="backup"} 0
catalog_actions_count{liveness="live",namespace="demo-prod",policy="test1",status="pending",type="backup"} 0
catalog_actions_count{liveness="live",namespace="demo-prod",policy="test1",status="running",type="backup"} 0
catalog_actions_count{liveness="live",namespace="mysql",policy="",status="complete",type="restore"} 1
catalog_actions_count{liveness="live",namespace="mysql",policy="",status="pending",type="restore"} 0
catalog_actions_count{liveness="live",namespace="mysql",policy="",status="running",type="restore"} 0
catalog_actions_count{liveness="live",namespace="mysql",policy="mysql-bkp",status="complete",type="backup"} 2
catalog_actions_count{liveness="live",namespace="mysql",policy="mysql-bkp",status="pending",type="backup"} 0
catalog_actions_count{liveness="live",namespace="mysql",policy="mysql-bkp",status="running",type="backup"} 0
catalog_actions_count{liveness="retired",namespace="",policy="mysql-bkp",status="complete",type="export"} 2
catalog_actions_count{liveness="retired",namespace="",policy="mysql-bkp",status="pending",type="export"} 0
catalog_actions_count{liveness="retired",namespace="",policy="mysql-bkp",status="running",type="export"} 0
catalog_actions_count{liveness="retired",namespace="demo-prod",policy="test1",status="cancelled",type="backup"} 1
```

- Get the number of **failed, pending or cancelled backup** jobs, filtered by **policy "mysql-bkp"**

  Prometheus to JSON:

  catalog_actions_count{policy="mysql-bkp",status=~"failed|cancelled|pending",type="backup"}

  JSONPath:

  $..value.sum()

- Get the number of **completed** or **running backup** jobs, filtered by **policy "mysql-bkp"**

  Prometheus to JSON:

  catalog_actions_count{policy="mysql-bkp",status=~"running|completed",type="backup"}

  JSONPath:

  $..value.sum()

- Get the number of **failed, pending or cancelled export** jobs, filtered by **policy "mysql-bkp"**

  Prometheus to JSON:

  catalog_actions_count{policy="mysql-bkp",status=~"failed|pending|cancelled",type="export"}

  JSONPath:

  $..value.sum()

- Get the number of **completed** or **running export** jobs, filtered by **policy "mysql-bkp"**

  Prometheus to JSON:

  catalog_actions_count{policy="mysql-bkp",status=~"running|completed",type="export"}

  JSONPath:

  $..value.sum()

7. APPENDIX B

7.1. References

1. Integration between Kasten K10 Prometheus and Slack:
   https://blog.kasten.io/posts/how-to-set-up-alerts-in-kasten-k10-to-immediately-catch-failed-backups
2. Prometheus Federation:
   https://prometheus.io/docs/prometheus/latest/federation/
3. Kasten K10 monitoring documentation:
   https://docs.kasten.io/latest/operating/monitoring.html
4. How to scrape data from Kasten K10 Prometheus:
   https://docs.kasten.io/latest/operating/monitoring.html#using-k10-s-prometheus-endpoint