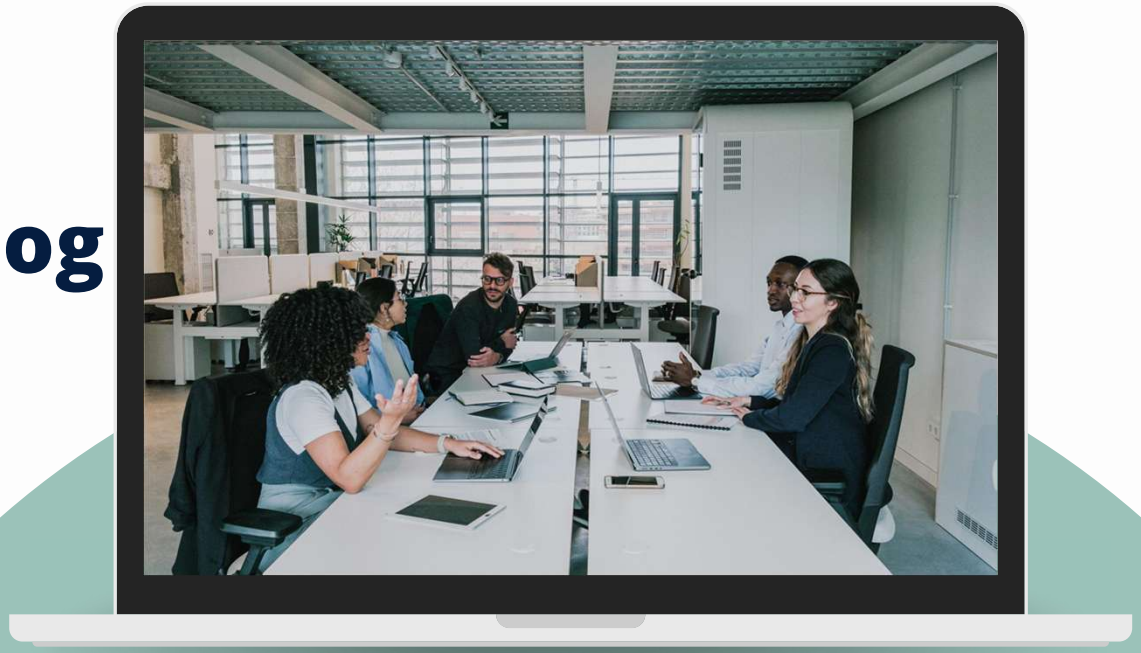
open-future
ZABBIX
PREMIUM PARTNER

POC : Using Fluentd with Zabbix API for Log Monitoring

By: Patrik Uytterhoeven



Introduction

One of our clients wanted to replace a monitoring tool that was not meeting their needs. They had set up a large number of custom scripts, and everything was sent to Fluentd.

As a proof of concept, we set up Zabbix and integrated Fluentd so that they could move to Zabbix without making any changes to their current setup. This would give time to replace later scripts with a built-in Zabbix functionality.

About us

Open-Future is a Belgian company specializing in Open Source and Linux solutions. It was founded on August 10, 2009, as a private limited company.

We focus on providing trainings, consultancy, and services in areas such as configuration management, backups, groupware, monitoring, security, and infrastructure solutions.

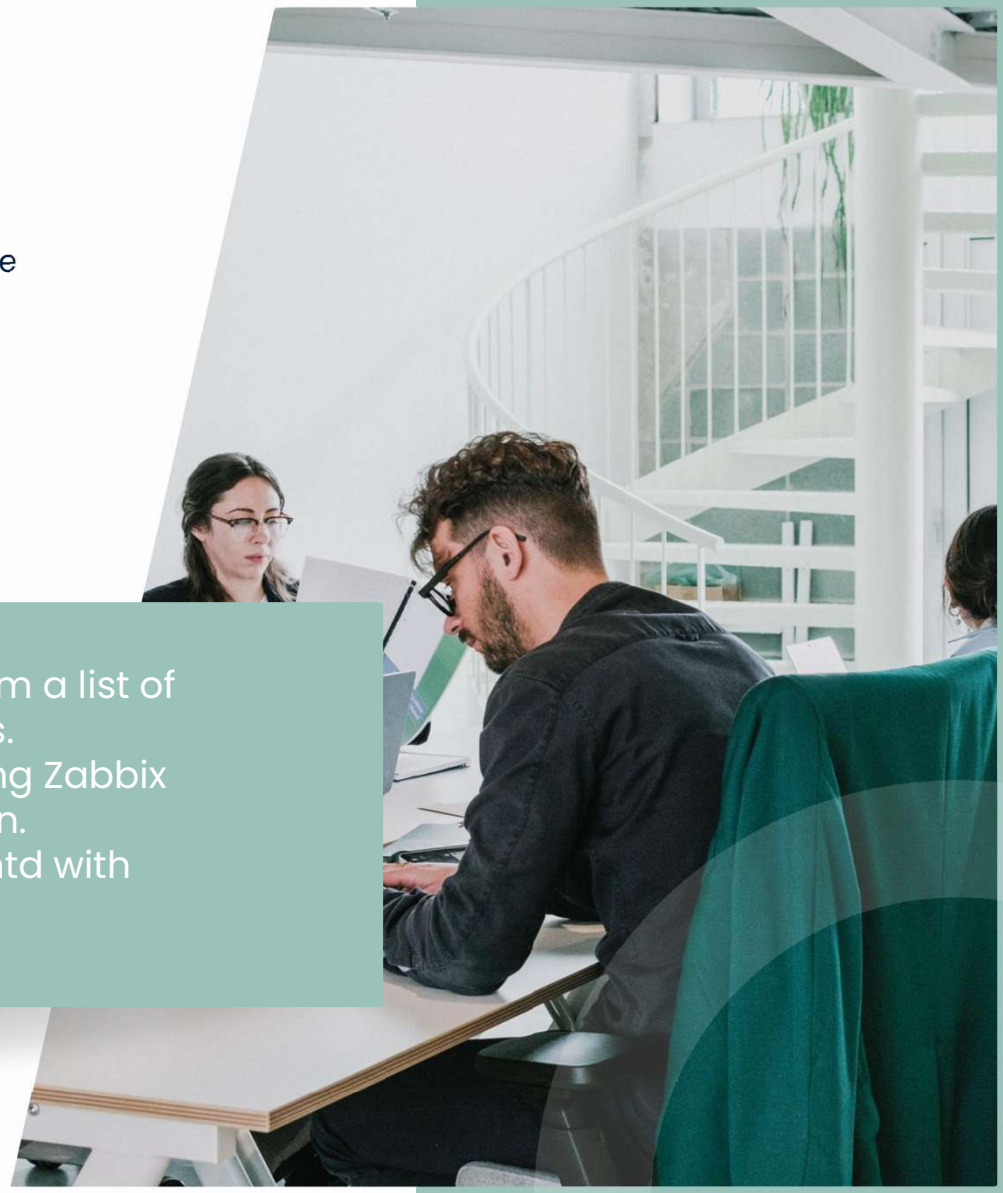
The problem

The client had a solution in place but was unhappy about it, as the solution was expensive and not doing what they wanted.

A variety of third-party tools and scripts had been placed in production to solve their needs.

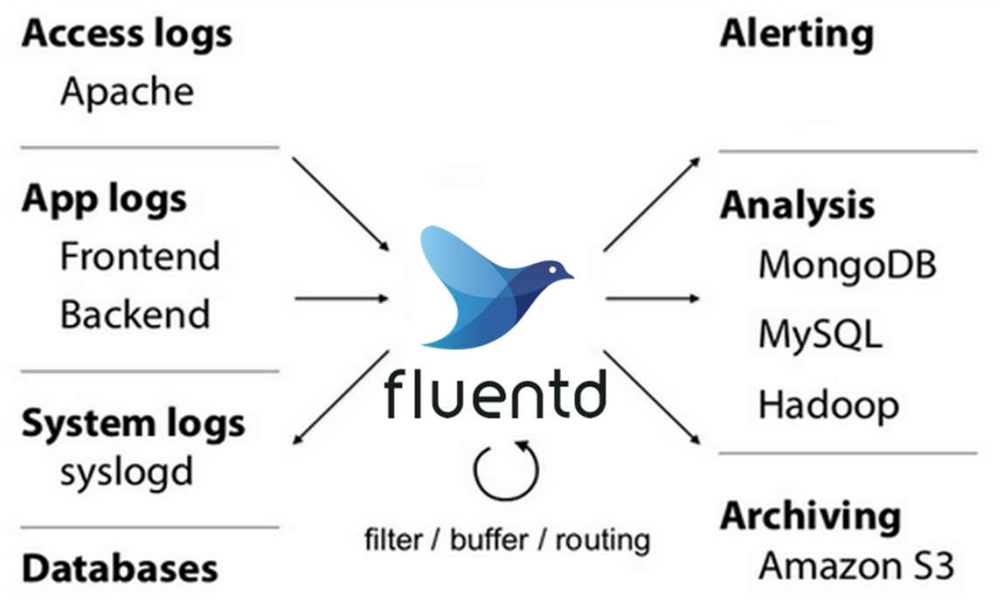
The Problem

The client had Fluentd in place to gather logs from a list of different applications and script outputs.
They wanted to keep this in place instead of using Zabbix agents as a first step to a quick migration.
We had to find a simple way to integrate Fluentd with Zabbix.



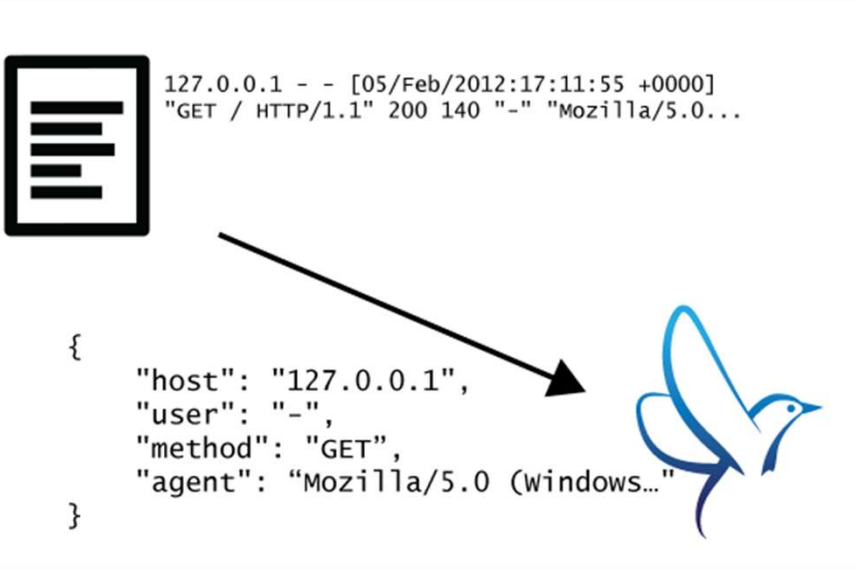
What is Fluentd?

Fluentd is an open-source data collector that unifies data collection and consumption for better use and understanding of data.



Unified Logging with JSON

Fluentd will try to structure data as JSON as much as possible.
This allows us to use Fluentd with the API of Zabbix 7.x to push data directly into an item.



How to configure Fluentd

First, we tell Fluentd to create a JSON for Zabbix and define the item where we would like to send our data.

```
##----------------------------
## check_healthcheck
##----------------------------

<filter Zabbix-payload>
  @type record_transformer
  enable_ruby
  <record>
     jsonrpc "2.0"
     method "history.push"
     params [{"itemid": "50034", "value": "${record['message']}" } ]
     auth null
     id 1
   </record>
   remove_keys message
</filter>
```

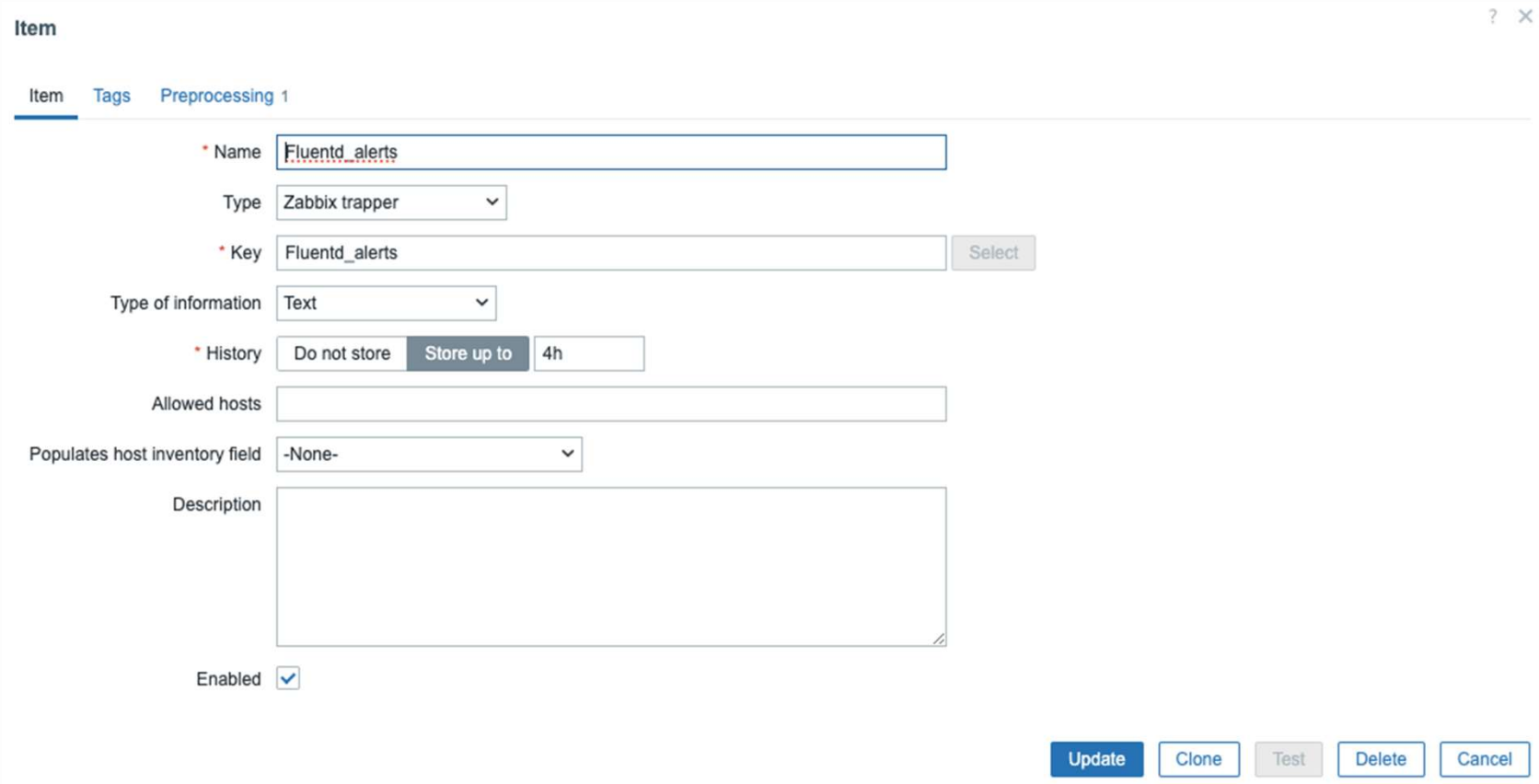
How to configure Fluentd

Next, we tell Fluentd to authenticate to our Zabbix API and send the content of our data.

```
##----------------------------
## send it
##----------------------------
<match Zabbix-payload>

    @type http
    endpoint http://Zabbix-server/zabbix/api_jsonrpc.php
    http_method post
    <format>
        @type json
    </format>
    json_array true
    headers {"Authorization":"Bearer <Zabbix API token>"}

    <buffer>
      flush_mode immediate
      chunk_limit_records 1
    </buffer>

</match>
```

Create our Zabbix item

The next step is to create our trapper item to catch the data from Fluentd.
We also need to add some preprocessing step, as our data from the scripts is separated with semicolons.



What does the date look like ?

This is how our actual data will look when Zabbix receives it from Fluentd:

[{"dateTime":"2025-01-29 12:04:34
+0100","hostname":”hostname.common.eu.corp.company.com","elementType":"logmon","elementName":"u01.app.oraopendbd.diag.rdbms.opendbd.OPENDBD.trace.alert_OPENDBD.log","parameter":"ALERT in u01.app.oraopendbd.diag.rdbms.opendbd.OPENDBD.trace.alert_OPENDBD.log","value":"ORA-1000 - please ignore","status":"ALERT"}]

Split the data with JS

The JS will format the data in JSON and also create a unique ID that we can use for our item key based on some parameters that are unique per item.

JavaScript

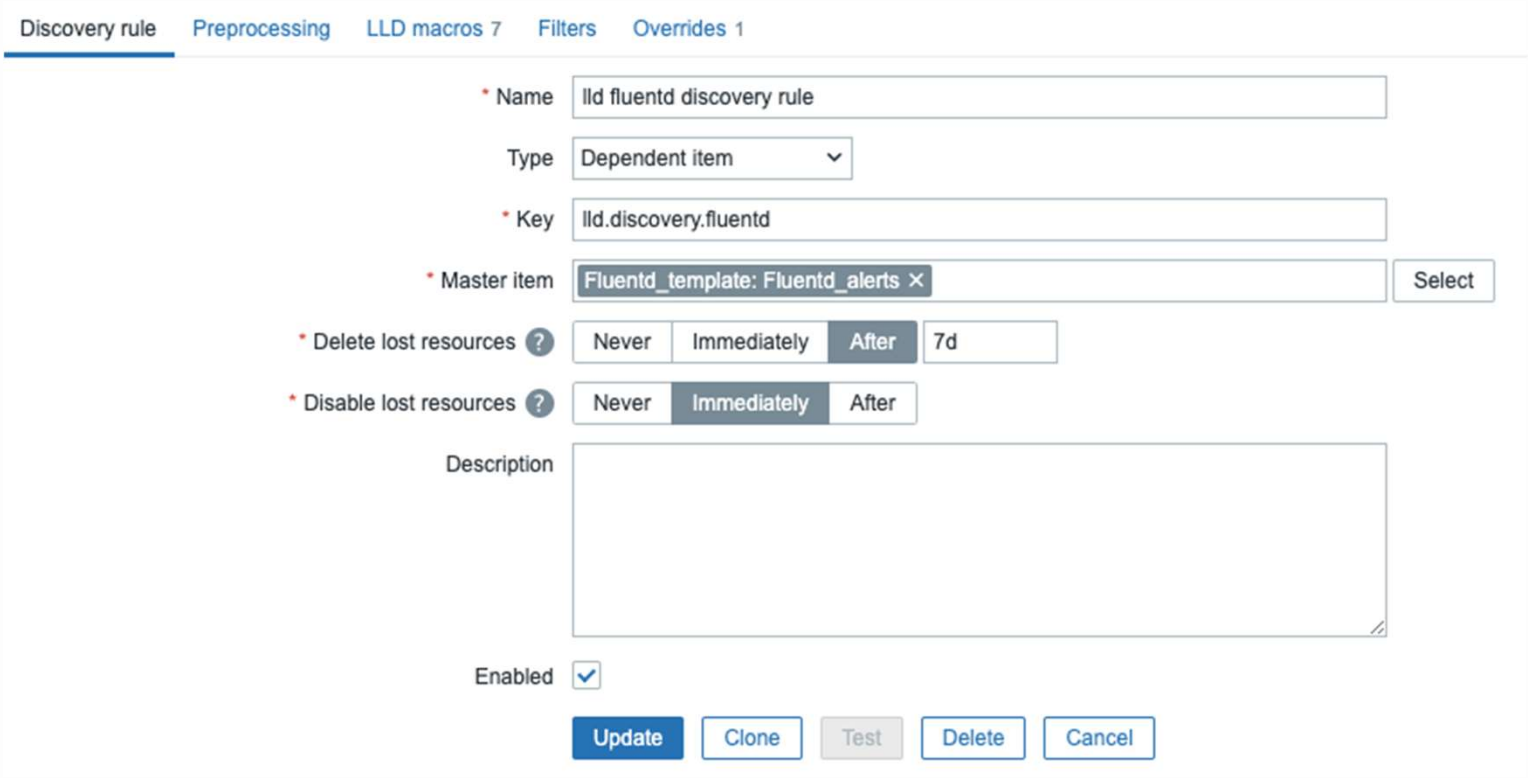
```
function (value) {
  var parts = value.split(';');
  var combinedString = [parts[1], parts[2], parts[3], parts[4]].join('|');


  var uniqId = sha256(combinedString).slice(0, 10);

  return JSON.stringify([{
      hostname: parts[1],
      elementType: parts[2],
      elementName: parts[3],
      parameter: parts[4],
      value: parts[5],
      status: parts[6],
      uniqId: uniqId // New SHA-256 encoded variable, limited to 10 characters
  }]);
}
```

65127 characters remaining

Apply Cancel

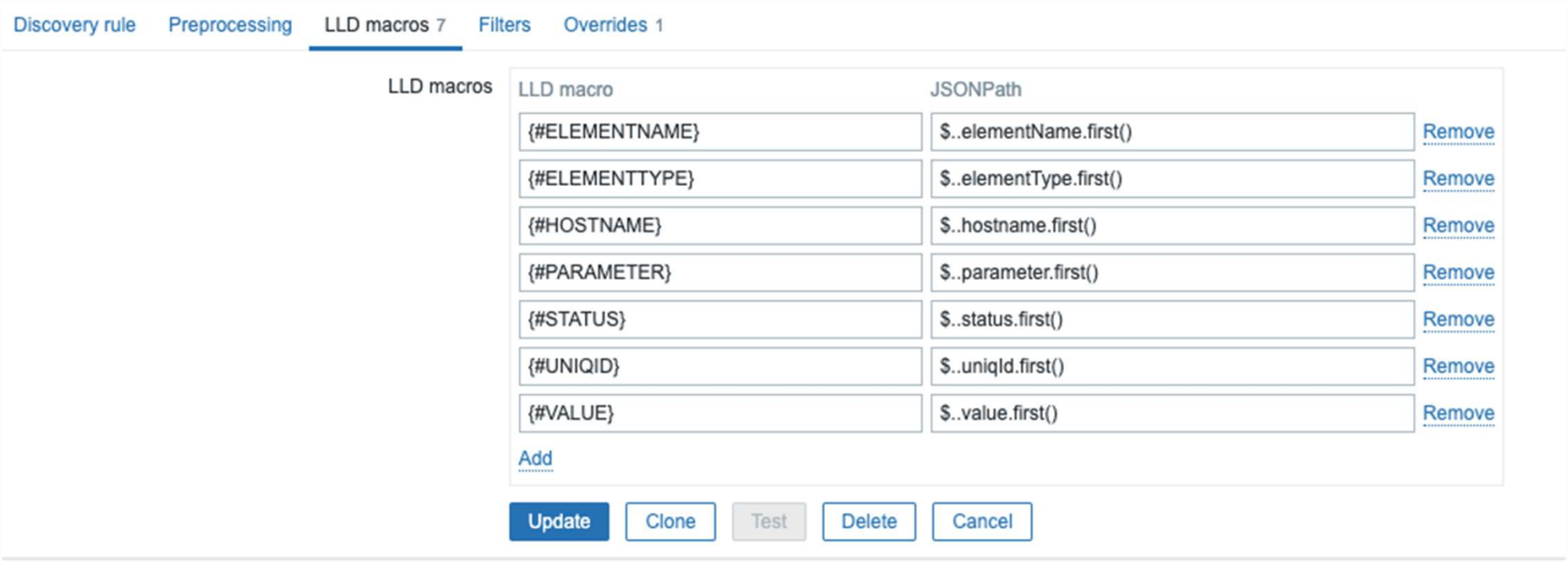
Time to create our LLD rule

We create an LLD rule that is a dependent item on our master item.
Of course, we need to use JSONPath to extract our macros so we can use them in our LLD items.



Time to create our LLD rule

With JSONPath, we are now able to create a set of LLD macros to use in our items.



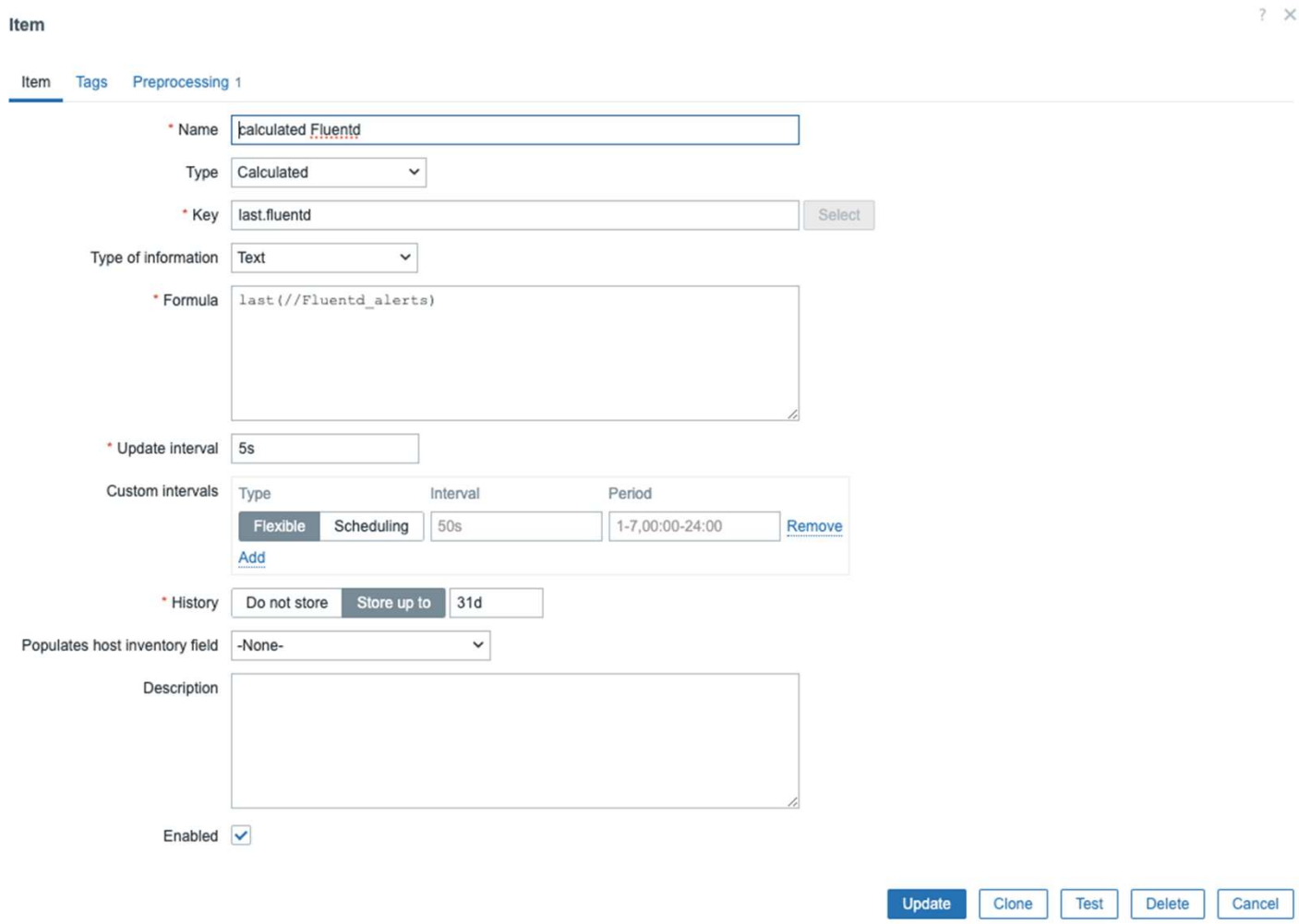
The Problem

We sent one payload to Zabbix and have to use this to create our LLD items.
We also need to use the same data to fill in the values, as Fluentd will only send this information to Zabbix once.

The solution is to create a calculated item to duplicate the data from our master item.
This item can be used to populate the items in our prototypes.

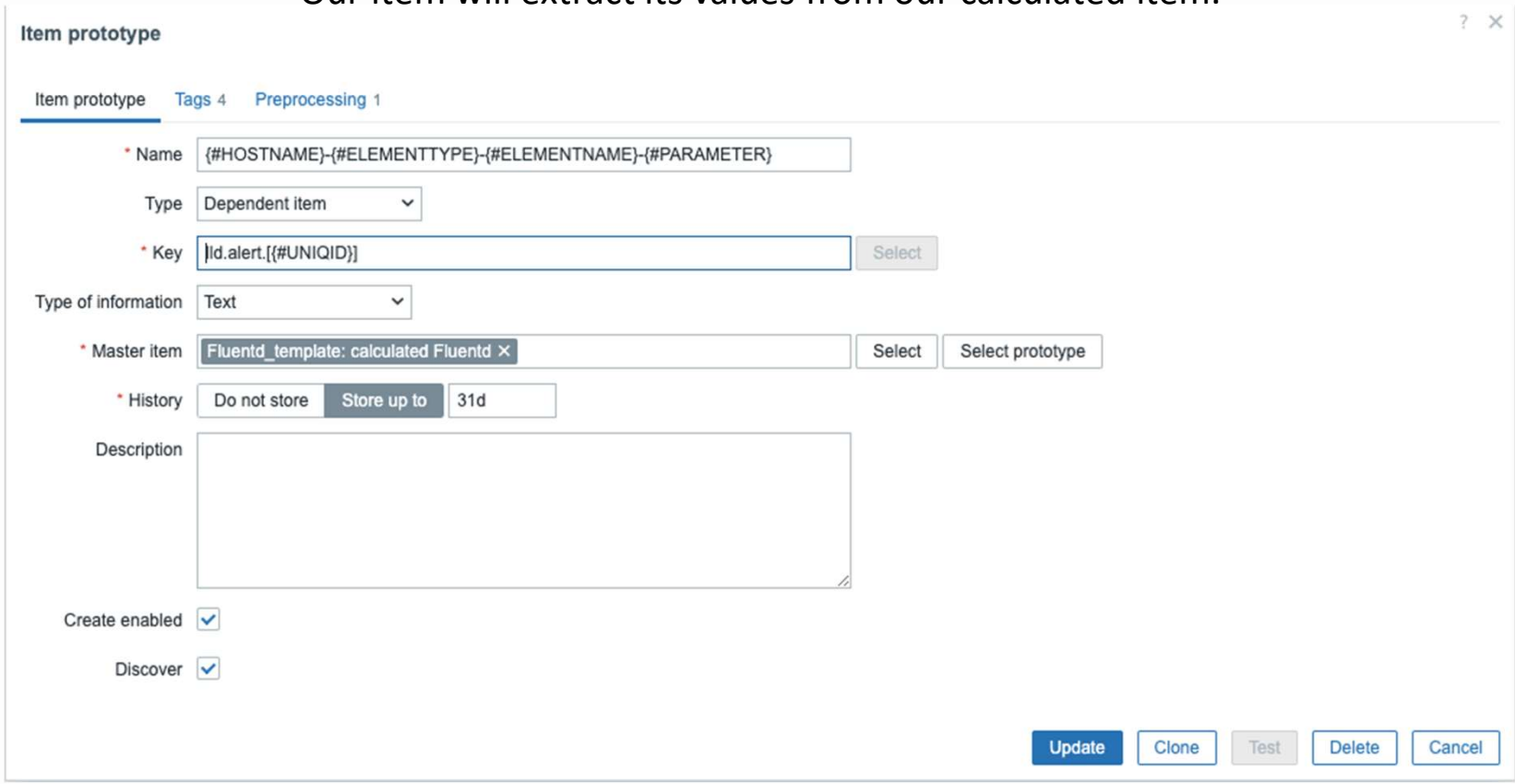
We also add a preprocessing step to discard duplicated values.

The Solution



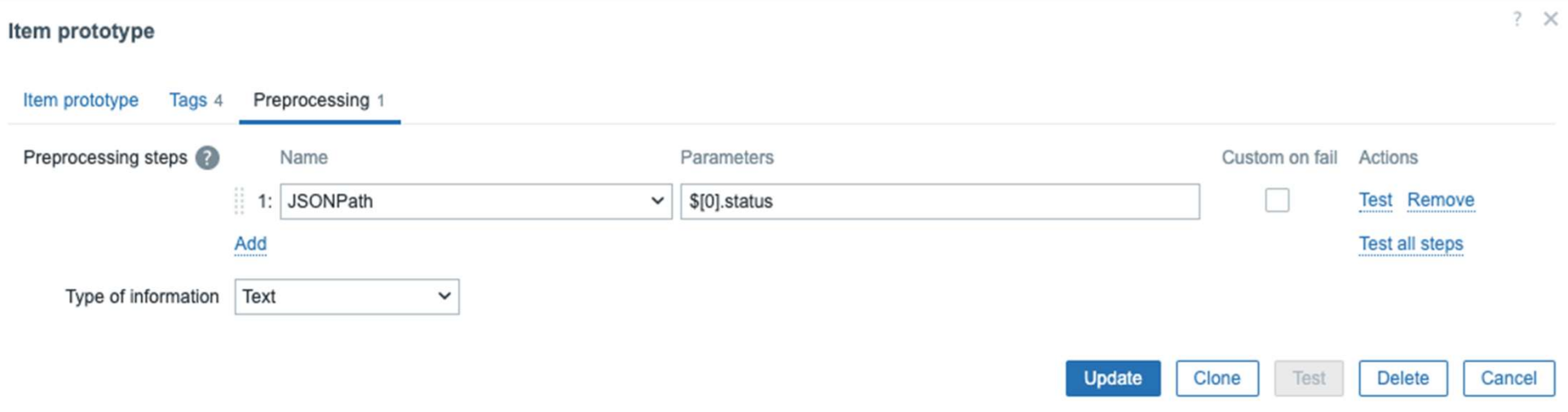
The LLD Item

The final step is to create our LLD item. It uses all the LLD filters we made,
and it also uses the Unique ID for our item key.
Our item will extract its values from our calculated item.



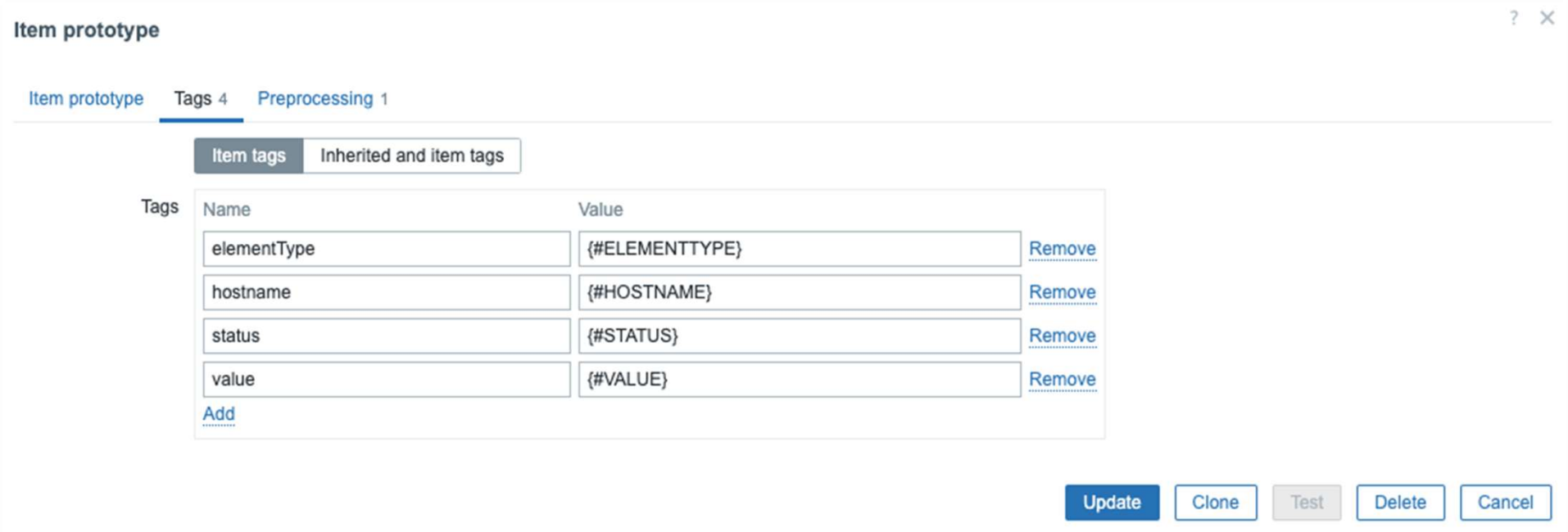
The LLD Item

Our item value is extracted in the preprocessing step.
For this, we use the status, as the status will be used in triggers to determine the severity.



The LLD Item

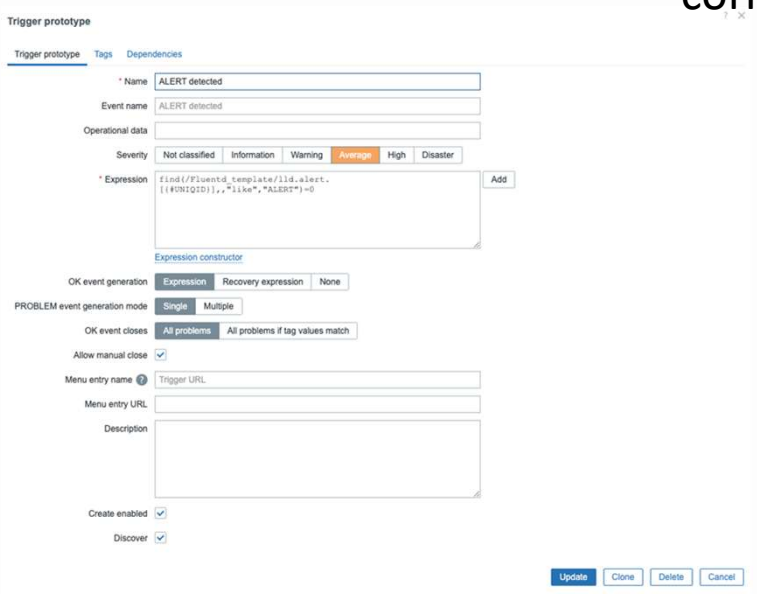
Since we extracted much more data, we use our tags to extract valuable information like hostname, value, elementType, etc.
This allows us to see which host sent the alert and what the error messages were.



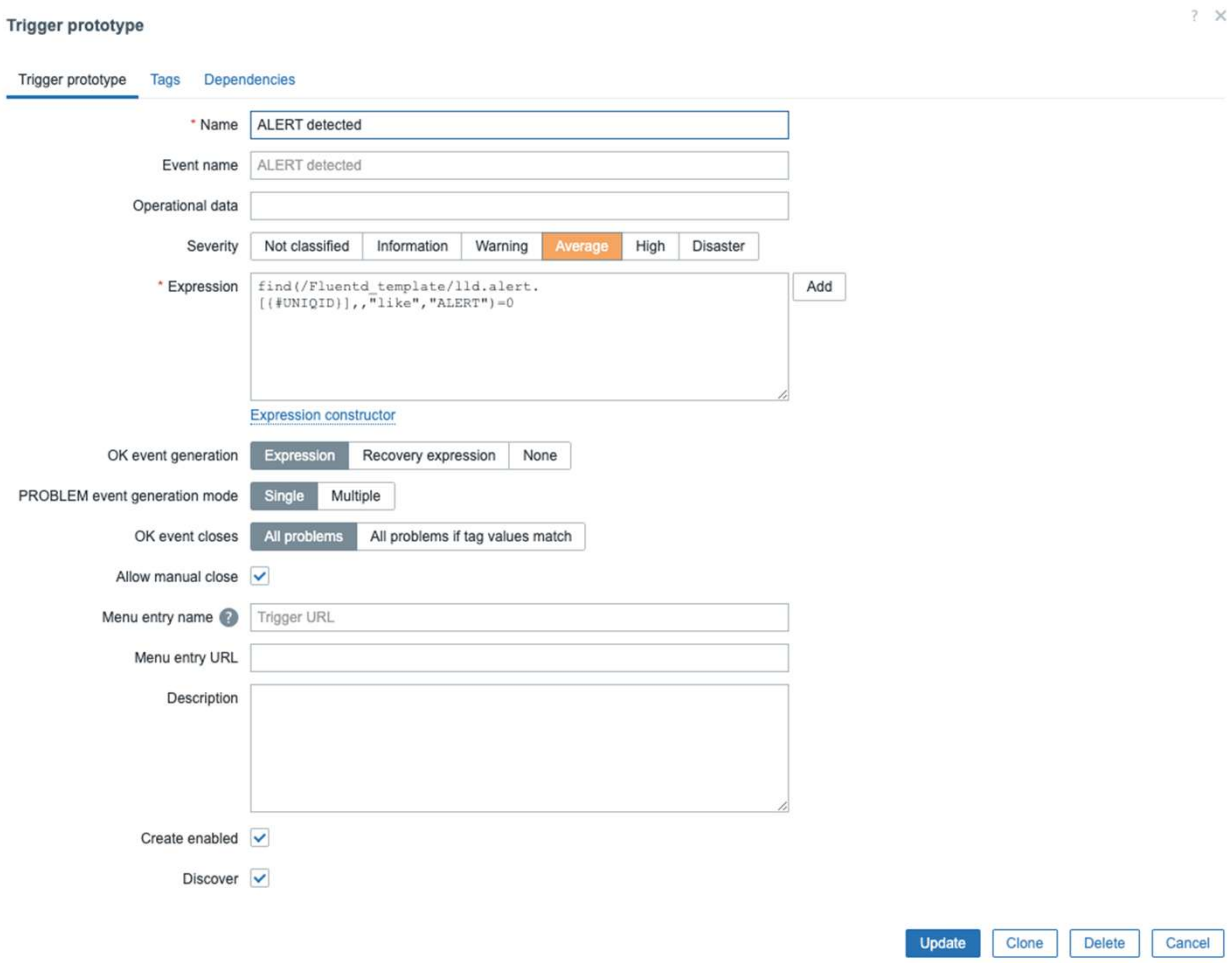
The LLD triggers

Our final step is to create some useful triggers.

Since our items will be populated with values
like ALERT, WARN, and INFO, we can map this with the
correct severity levels in Zabbix.

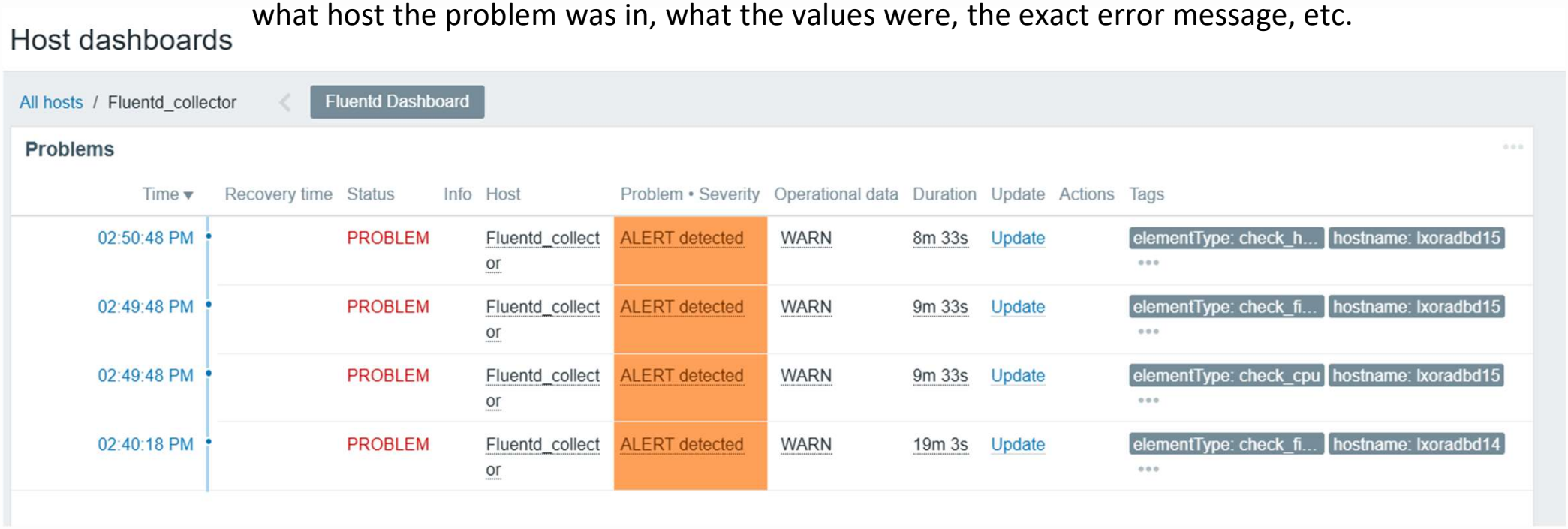


The LLD triggers



When the pieces fall together

The final step was to add a host dashboard with the alerts.
With the information in the tags, we can get more information on
what host the problem was in, what the values were, the exact error message, etc.



Limitations and the possible future

We are only able to send the data in one call to the Zabbix server.
This means we have to use the data to create our LLD discovery
and populate our items with values.

This is possible only with the workaround of the calculated item.
This poses limitations on the amount of data we will be able to process.

We are looking into Fluentd ATM for ways to solve this. A possible solution is to fix this with the @copy option.

Another option is to replace Fluentd in a later phase with Zabbix agents and the original scripts by making use of Userparameters.

open-future
ZABBIX
PREMIUM PARTNER

THANK YOU!

Patrik Uytterhoeven
Senior Consultant & Trainer
+32 2 255 70 70
www.open-future.be
Leuvensesteenweg 643 1930
Zaventem (Belgium)