2016

odoo

EXPERIENCE

Odoo Monitoring using Munin

Seinlet Nicolas, Technical consultant

@nseinlet

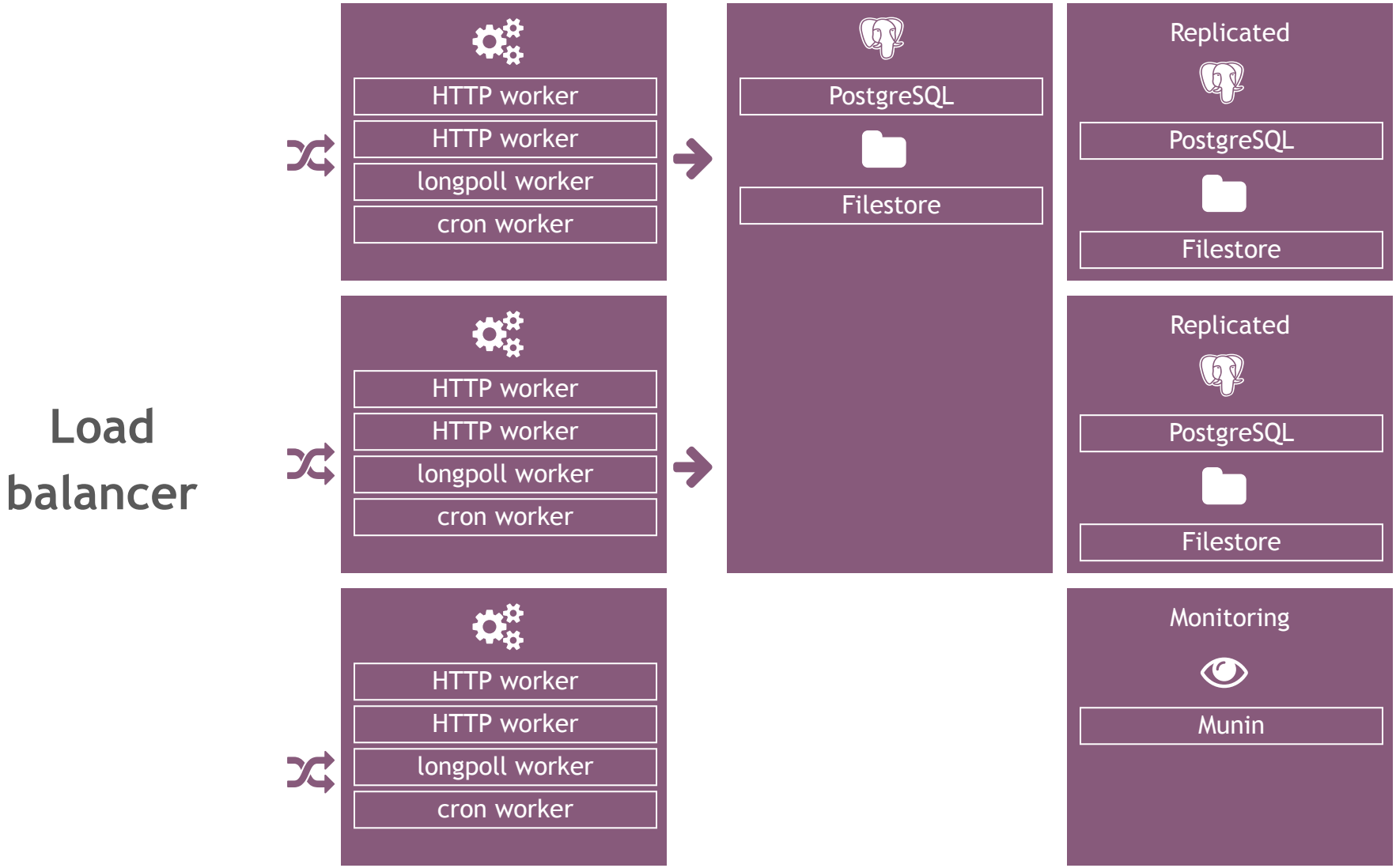
You can't control what you can't measure

Tom DeMarco

Schedule

1. Install munin
2. Monitor a first node
3. Monitor PostgreSQL
4. Monitor Odoo
5. Conclusions

Our infrastructure



Install Munin

Install Munin package

```
$ sudo apt install munin
```

This will install munin, as well as munin-node (for local server monitoring), some plugins, ...

We need only one Munin, but we'll need to deploy munin-node on each node to monitor.

Install NGinx

We'll use NGinx to serve static files generated by munin. It's also possible to use NGinx as a proxy to forward requests to munin-http.

```
$ sudo apt install nginx apache2-utils
$ sudo htpasswd -cb /etc/nginx/htpasswd admin admin
$ sudo vi /etc/nginx/sites-enabled/default
```

```
server {
    listen 80 default_server;
    listen [::]:80 default_server;

    location / {
        rewrite ^/$ munin/ redirect; break;
    }
    location /munin/static/ {
        alias /etc/munin/static/;
        expires modified +1w;
    }
    location /munin/ {
        auth_basic            "Restricted";
        # Create the htpasswd file with the htpasswd tool.
        auth_basic_user_file  /etc/nginx/htpasswd;

        alias /var/cache/munin/www/;
        expires modified +310s;
    }
}
```

Munin is up and running, and published in http.

Monitor a first Munin node

Install Munin node package

As a first node, we'll use our HaProxy server. We'll deploy on it standard monitoring as well as more specific ones.

```
$ sudo apt install munin-node
```

Allow munin server to monitor this node

```
$ sudo vi /etc/munin/munin-node.conf
```

```
...
host_name haodoo
allow ^192\.168\.56\.40$
...
```

Add this node in server configuration

```
$ sudo vi /etc/munin/munin.conf
```

```
...
[ha;haodoo]
    address 192.168.56.30
...
```

Check plugins on node

Munin is able to monitor pretty anything. The node has for job to measure and send datas to master. By default, some plugins are already deployed (ram, cpu, ...) but we can add somes more relevant to our system. A way to determine which plugins are usefull is to use munin-node-configure itself :

```
$ sudo munin-node-configure --suggest

Plugin                    | Used | Suggestions
------                    | ---- | -----------
acpi                      | no   | no [cannot read /proc/acpi/thermal_zone/*/temperature]
amavis                    | no   | no
apache_accesses           | no   | no [LWP::UserAgent not found]
apache_processes          | no   | no [LWP::UserAgent not found]
apache_volume             | no   | no [LWP::UserAgent not found]
apc_envunit_              | no   | no [no units to monitor]
bonding_err_              | no   | no [No /proc/net/bonding]
courier_mta_mailqueue     | no   | no [spooldir not found]
courier_mta_mailstats     | no   | no [could not find executable]
courier_mta_mailvolume    | no   | no [could not find executable]
cps_                      | no   | no
cpu                       | yes  | yes
...
```

Result show what's installed and what should be.

Install plugins on node

It's possible to add more plugins on node :

```
$ sudo apt install libwww-perl
$ ls /usr/share/munin/plugins/ha*
/usr/share/munin/plugins/haproxy_  /usr/share/munin/plugins/haproxy_ng
```

Those 2 plugins seems relevant to our particular node. Let's deploy them

```
$ sudo ln -s /usr/share/munin/plugins/haproxy_ /etc/munin/plugins/haproxy_backend
$ sudo ln -s /usr/share/munin/plugins/haproxy_ng /etc/munin/plugins/haproxy_ng
$ sudo vi /etc/munin/plugin-conf.d/munin-node
```

```
...
[haproxy_*]
env.url http://odoo:odoo@localhost:1936/haproxy-status?stats;csv
```

```
$ sudo service munin-node restart
```

Begin of result

MUNIN Overview

Problems

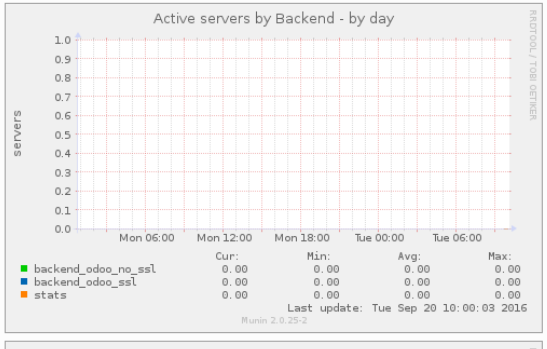
Critical (0)
Warning (0)
Unknown (0)

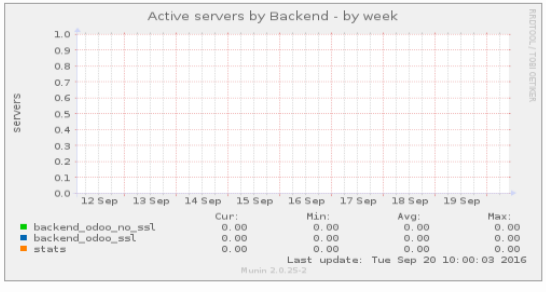
Groups

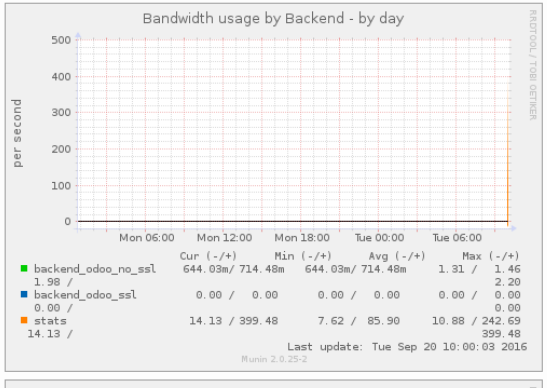
ha
localdomain

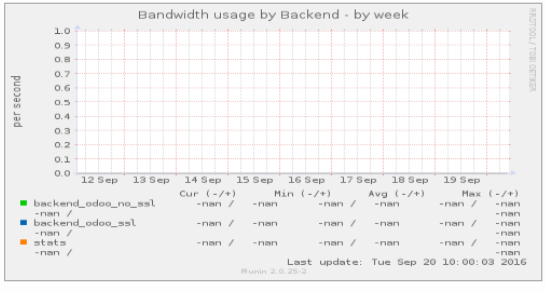
- ha
  - haodoo [disk haproxy network processes system]
- localdomain
  - localhost.localdomain [disk munin network processes system]

haproxy









Define alarm threshold

Monitoring is good, and is even better if I can define my own threshold for warning and critical level. Thoses levels can be defined on the node, in the plugin configuration if plugin handle it. Or can be defined on the server. In the case of the haproxy_ng plugin, it doesn't handle warning and critical levels. So, on the munin server, not the node:

```
$ sudo vi /etc/munin/munin.conf
```

```
[ha;haodoo]
    address 192.168.56.30
    HAPActive.backend_odoo_ssl_act.warning 2:
    HAPActive.backend_odoo_ssl_act.critical 1:
    HAPActive.backend_odoo_no_ssl_act.warning 2:
    HAPActive.backend_odoo_no_ssl_act.critical 1:
```

Field	Internal name	Type	Warn	Crit	Info
backend_odoo_no_ssl	backend_odoo_no_ssl_act	gauge	2:	1:	Number of active servers
backend_odoo_ssl	backend_odoo_ssl_act	gauge	2:	1:	Number of active servers
stats	stats_act	gauge			Number of active servers

Install extras plugins on node

It's also possible to add more plugins on node, from github for example, like the ones from https://github.com/munin-monitoring/contrib

```
$ sudo rm /etc/munin/plugins/haproxy_ng
$ cd /opt
$ sudo git clone https://github.com/munin-monitoring/contrib.git
$ sudo ln -s /opt/contrib/plugins/haproxy/haproxy_active_backend /etc/munin/plugins/haproxy_active_backe
$ sudo ln -s /opt/contrib/plugins/haproxy/haproxy_abort_backend /etc/munin/plugins/haproxy_abort_backend
$ sudo vi /etc/munin/plugin-conf.d/munin-node
```

```
[haproxy_*]
env.url http://odoo:odoo@localhost:1936/haproxy-status?stats;csv
env.backend backend_odoo_no_ssl backend_odoo_ssl
```

```
$ sudo service munin-node restart
```

3 Monitor PostgreSQL

Monitor PostgreSQL

As datas resides on this server, it's important to monitor this particular server. Depending on your configuration, some metrics are relevant or not, like replication delay. We add postgresql server 1 in munin config then check plugins on the postgresql node.

Allow munin server to monitor this node

```
$ sudo apt install munin-node libdbd-pg-perl
$ sudo vi /etc/munin/munin-node.conf
```

```
...
host_name odoo-pg1
allow ^192\.168\.56\.40$
...
```

Configure munin server to monitor it

```
$ sudo vi /etc/munin/munin.conf
```

```
[pg;odoo-pg1]
    address 192.168.56.21
[pg;odoo-pg2]
    address 192.168.56.22
```

Suggest PostgreSQL plugins

```
$ sudo munin-node-configure --suggest
...
postgres_autovacuum         | no   | yes
postgres_bgwriter           | no   | yes
postgres_cache_             | no   | yes (+ALL +odoo)
postgres_checkpoints        | no   | yes
postgres_connections_       | no   | yes (+ALL +odoo)
postgres_connections_db     | no   | yes
postgres_locks_             | no   | yes (+ALL +odoo)
postgres_oldest_prepared_xact_ | no   | no [Prepared transactions not enabled]
postgres_prepared_xacts_    | no   | no [Prepared transactions not enabled]
postgres_querylength_       | no   | yes (+ALL +odoo)
postgres_scans_             | no   | yes (+odoo)
postgres_size_              | no   | yes (+ALL +odoo)
...
```

Add PostgreSQL plugins

As suggested, deploy the PostgreSQL plugins

```
$ sudo munin-node-configure --shell
ln -s '/usr/share/munin/plugins/postgres_autovacuum' '/etc/munin/plugins/postgres_autovacuum'
ln -s '/usr/share/munin/plugins/postgres_bgwriter' '/etc/munin/plugins/postgres_bgwriter'
ln -s '/usr/share/munin/plugins/postgres_cache_' '/etc/munin/plugins/postgres_cache_ALL'
ln -s '/usr/share/munin/plugins/postgres_cache_' '/etc/munin/plugins/postgres_cache_odoo'
ln -s '/usr/share/munin/plugins/postgres_checkpoints' '/etc/munin/plugins/postgres_checkpoints'
ln -s '/usr/share/munin/plugins/postgres_connections_' '/etc/munin/plugins/postgres_connections_ALL'
ln -s '/usr/share/munin/plugins/postgres_connections_' '/etc/munin/plugins/postgres_connections_odoo'
ln -s '/usr/share/munin/plugins/postgres_connections_db' '/etc/munin/plugins/postgres_connections_db'
ln -s '/usr/share/munin/plugins/postgres_locks_' '/etc/munin/plugins/postgres_locks_ALL'
ln -s '/usr/share/munin/plugins/postgres_locks_' '/etc/munin/plugins/postgres_locks_odoo'
ln -s '/usr/share/munin/plugins/postgres_querylength_' '/etc/munin/plugins/postgres_querylength_ALL'
ln -s '/usr/share/munin/plugins/postgres_querylength_' '/etc/munin/plugins/postgres_querylength_odoo'
ln -s '/usr/share/munin/plugins/postgres_scans_' '/etc/munin/plugins/postgres_scans_odoo'
ln -s '/usr/share/munin/plugins/postgres_size_' '/etc/munin/plugins/postgres_size_ALL'
ln -s '/usr/share/munin/plugins/postgres_size_' '/etc/munin/plugins/postgres_size_odoo'
ln -s '/usr/share/munin/plugins/postgres_transactions_' '/etc/munin/plugins/postgres_transactions_ALL'
ln -s '/usr/share/munin/plugins/postgres_transactions_' '/etc/munin/plugins/postgres_transactions_odoo'
ln -s '/usr/share/munin/plugins/postgres_tuples_' '/etc/munin/plugins/postgres_tuples_odoo'
ln -s '/usr/share/munin/plugins/postgres_users' '/etc/munin/plugins/postgres_users'
ln -s '/usr/share/munin/plugins/postgres_xlog' '/etc/munin/plugins/postgres_xlog'
$ sudo munin-node-configure --shell | sudo sh
$ sudo service munin-node restart
```

Custom plugin

All thoses plugins are cool, but I want to monitor my streaming replication!

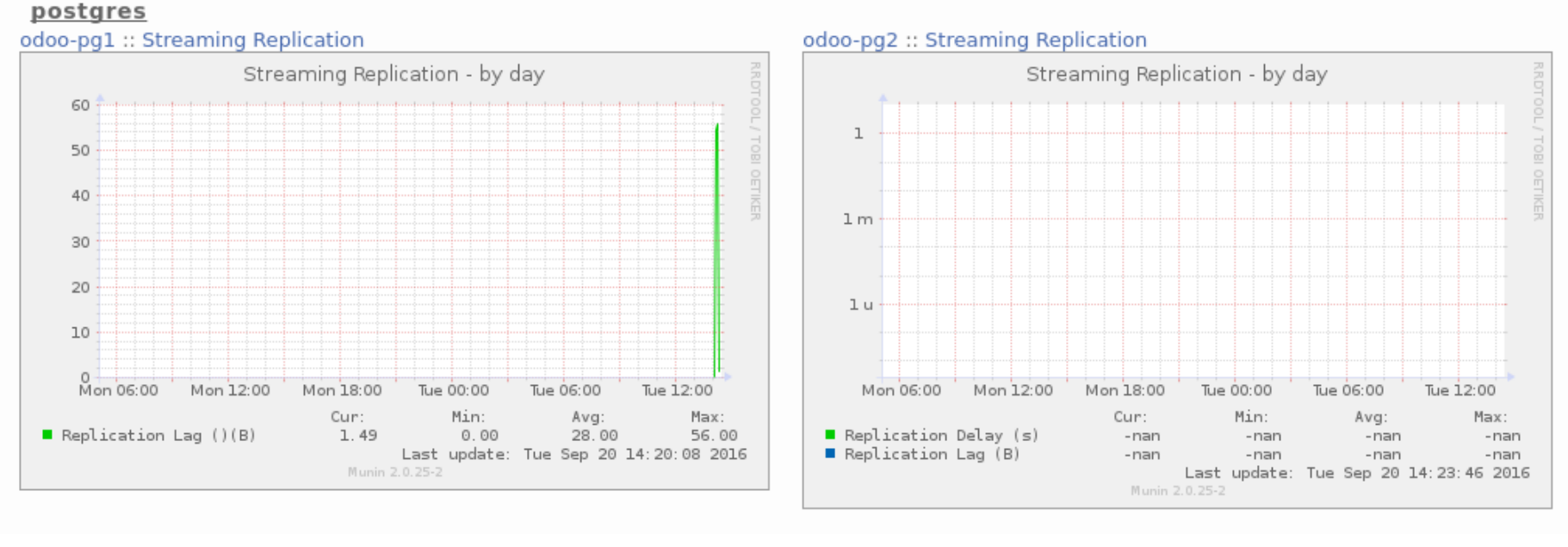
```
$ sudo vi /etc/munin/plugins/pg_replication
...
slave=`psql postgres -A -t -c "SELECT pg_is_in_recovery();"`
master=`psql postgres -A -t -c "SELECT count(client_addr) from pg_stat_replication;"`
...
case $1 in
    config)
        echo graph_category postgres
        echo graph_title Streaming Replication
...
if [[ $master -gt 0 ]]; then
    # master
    echo -n "delay_b.value "
    psql postgres -A -t -c "select pg_xlog_location_diff(sent_location, replay_location) from pg_stat_re
fi
...
```

```
$ sudo chmod a+x /etc/munin/plugins/pg_replication
$ sudo vi /etc/munin/plugin-conf.d/munin-node
```

```
[pg_*]
user postgres
group postgres
```

First custom graph

Don't forget ot restart munin-node on the node, and you can run munin-cron on the master as munin user to speed up a bit. You can also run munin-run pg_replication on the node to check if everything is ok.



Other usefull graphs

Some graphs are activated by default, and used to diagnose performance issues, size postgresql memory, ...

- Disk IOs
- Disk latency
- CPU Usage (iowait, steal, ...)
- Load average
- Memory usage (cache, ...)

4 Monitor Odoo

Adjust log-level

We want to monitor response time within odoo. To achieve this, we need to get the response time from odoo, in the odoo logs. In your odoo config file:

```
log_level = debug_rpc
```

This will generate logs like:

```
2016-09-22 12:23:11,205 30977 DEBUG 9dpm-test openerp.http.rpc.request: read_subscription_data: None
    None: time:0.014s mem: 1005056k -> 1005056k (diff: 0k)
2016-09-22 12:23:11,206 30977 INFO 9dpm-test werkzeug: 127.0.0.1 - - [22/Sep/2016 12:23:11] "POST
    /mail/read_subscription_data HTTP/1.1" 200 -
2016-09-22 12:23:11,485 30977 DEBUG 9dpm-test openerp.http.rpc.request: call_kw: order.category r
    ead: time:0.235s mem: 1005056k -> 1006080k (diff: 1024k)
2016-09-22 12:23:11,486 30977 INFO 9dpm-test werkzeug: 127.0.0.1 - - [22/Sep/2016 12:23:11] "POST
    /web/dataset/call_kw/order.category/read HTTP/1.1" 200 -
2016-09-22 12:23:11,750 30977 DEBUG 9dpm-test openerp.http.rpc.request: call_kw: sale.order.line
    read: time:0.152s mem: 1006080k -> 1007360k (diff: 1280k)
2016-09-22 12:23:11,752 30977 INFO 9dpm-test werkzeug: 127.0.0.1 - - [22/Sep/2016 12:23:11] "POST
    /web/dataset/call_kw/sale.order.line/read HTTP/1.1" 200 -
2016-09-22 12:23:12,074 30977 DEBUG 9dpm-test openerp.http.rpc.request: call_kw: sale.order.line
    read: time:0.169s mem: 1007360k -> 1008128k (diff: 768k)
2016-09-22 12:23:12,075 30977 INFO 9dpm-test werkzeug: 127.0.0.1 - - [22/Sep/2016 12:23:12] "POST
    /web/dataset/call_kw/sale.order.line/read HTTP/1.1" 200 -
```

Ask munin server to monitor

As usual, deploy munin-node on your server and ask munin main server to monitor it.

```
$ sudo apt install munin-node
$ sudo vi /etc/munin/munin-node.conf
```

```
...
host_name odoo-1
allow ^192\.168\.56\.40$
...
```

Configure munin server to monitor it

```
$ sudo vi /etc/munin/munin.conf
```

```
[odoo;odoo-1]
    address 192.168.56.10
[odoo;odoo-2]
    address 192.168.56.11
```

How many users?

A usual question is "how many concurrent users do I have?", and another one is "How many transactions/minute do they do?". As the number of workers and thus CPU is directly linked to this answer, it's important to answer.

We can distinguish two kind of requests. The longpoll ones, and the others.

Number of longpoll requests per minute ~= number of users, even inactive with just an open browser tab.

Other requests are sent by active users.

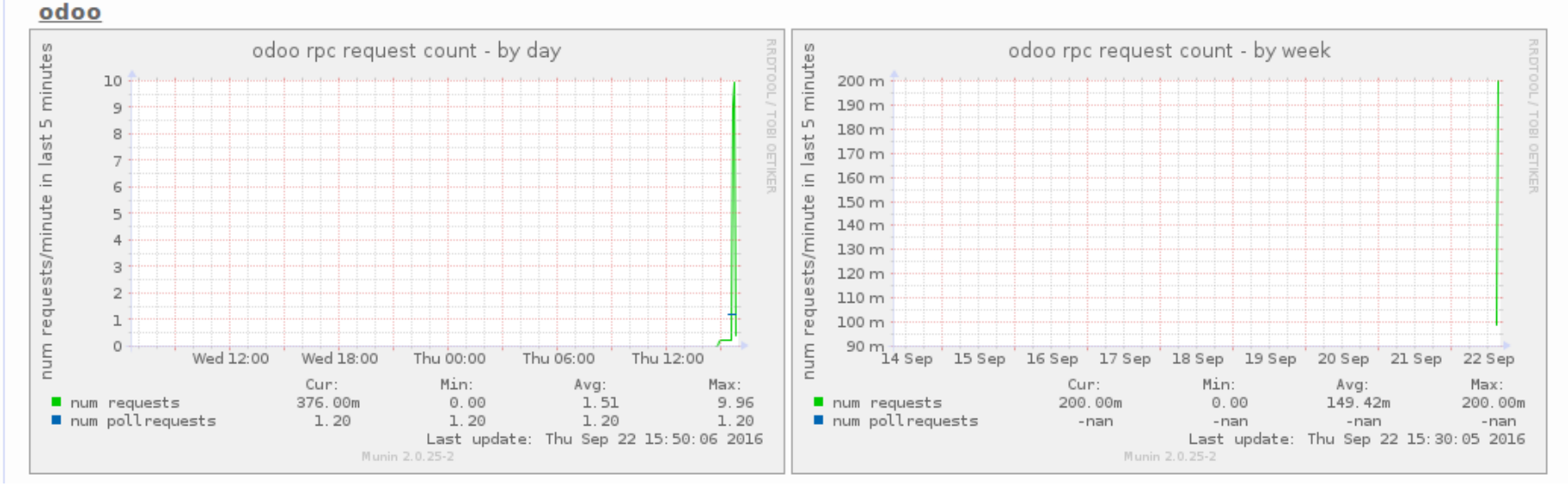
Script it

```
$ sudo vi /etc/munin/plugins/odootrmin
```

```
#!/bin/sh
#%# family=manual
#%# capabilities=autoconf suggest
case $1 in
  autoconf)
      exit 0
      ;;
  suggest)
      exit 0
      ;;
  config)
      echo graph_category odoo
      echo graph_title odoo rpc request count
      echo graph_vlabel num requests/minute in last 5 minutes
      echo requests.label num requests
      echo pollrequests.label num poll requests
      exit 0
      ;;
esac
# watch out for the time zone of the logs => using date -u for UTC timestamps
result=$(tail -60000 /var/log/odoo/odoo.log | grep -a "odoo.http.rpc.request" | grep -a -v "poll" | awk
pollresult=$(tail -60000 /var/log/odoo/odoo.log | grep -a "odoo.http.rpc.request" | grep -a "poll" | awk
echo "requests.value ${result}"
echo "pollrequests.value ${pollresult}"
exit 0
```

In my dashboard

Once munin node restarted, and munin cron ran, a new graph appears.



Measure response time

It's possible to extract the rpc response time from odoo logs. in /etc/munin/plugins/odooresponse:

```
#!/bin/sh
#%# family=manual
#%# capabilities=autoconf suggest
case $1 in
  config)
      echo graph_category odoo
      echo graph_title odoo rpc requests min/average response time
      echo graph_vlabel seconds
      echo graph_args --units-exponent -3
      echo min.label min
      echo min.warning 1
      echo min.critical 5
      echo avg.label average
      echo avg.warning 1
      echo avg.critical 5
      exit 0
      ;;
esac
# watch out for the time zone of the logs => using date -u for UTC timestamps
result=$(tail -60000 /var/log/odoo/odoo.log | grep -a "odoo.http.rpc.request" | grep -a -v "poll" | awk
echo -n "min.value "
echo ${result} | cut -d" " -f1
echo -n "avg.value "
echo ${result} | cut -d" " -f2
exit 0
```

In my dashboard

Once munin node restarted, and munin cron ran, a new graph appears.

Control business process

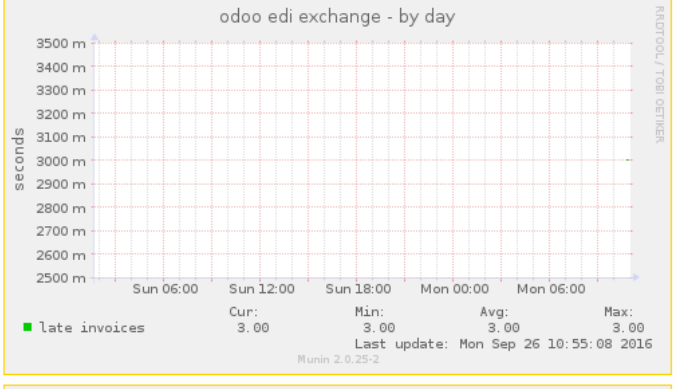
Sometimes, Munin can help you monitor a business critical process invisible for end users, for example an EDI exchange.

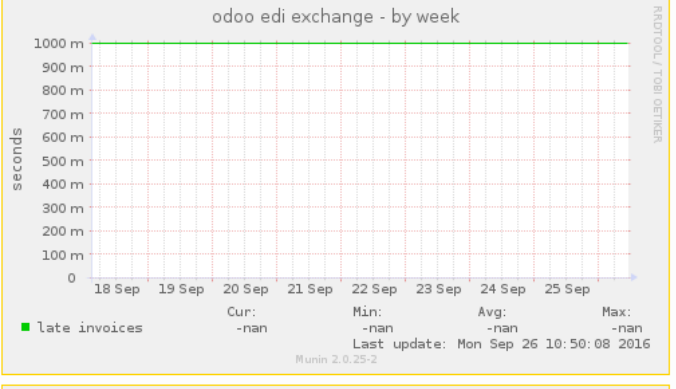
End user doesn't look at this process, and cannot solve a problem in most of the cases, so...

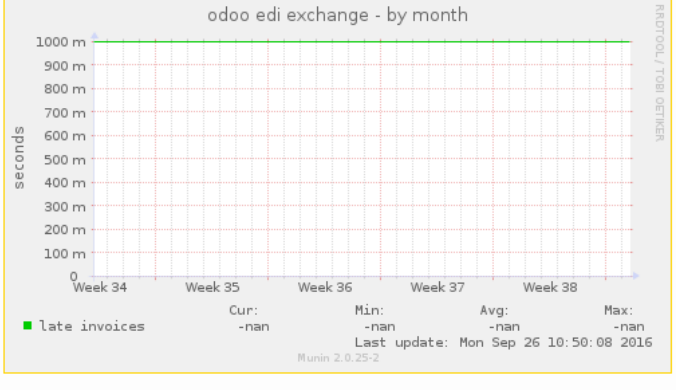
```
#!/bin/sh
#%# family=manual
#%# capabilities=autoconf suggest
case $1 in
    config)
        echo graph_category odoo
        echo graph_title odoo edi exchange
        echo graph_vlabel # documents
        echo graph_args --units-exponent 0
        echo late.label late invoices
        echo late.warning 1
        echo late.critical 3
        exit 0
        ;;
    esac
echo -n "late.value "
psql -A -t -d odoo1 -c "
select count(*) from account_invoice
inner join res_partner on account_invoice.partner_id=res_partner.id
where state in ('open','paid') and account_invoice.type in ('out_invoice', 'out_refund')
and date_INVOICE<=current_date - 1 and date_sent IS NULL
and res_partner.use_edi=true;"
exit 0
```

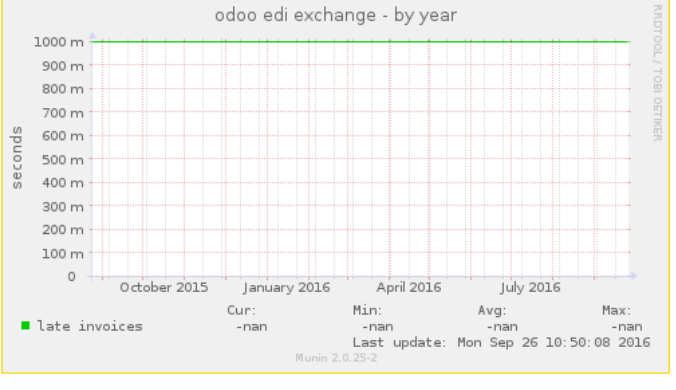
In my dashboard

- pg
  - odoo-pg1 [disk network nfs odoo postgres postgresql processes system]









Note: This service is in WARNING state because one of the values reported is outside the allowed range. Please see further down for information about the

Field	Internal name	Type	Warn	Crit	Info
late invoices	late	gauge	1	3	

Notify me

I want to be notified by email in case I'm away from office... To achieve this, add this in the munin.conf on the munin master server

```
contacts nse
contact.nse.command mail -s "Munin notif ${var:host}" nse@mycompany.com
contact.nse.always_send warning critical
```

5 Conclusions

Code available on https://github.com/nseinlet/runbot_ir

Simple to deploy

Highly flexible

Give values for configuration

Mandatory for perf analysis

Plenty of documentation and samples

2016 odoo EXPERIENCE

Any question?

#odooexperience

Nicolas Seinlet (nse@odoo.com)

@nseinlet