Icinga notifications to Mastodon

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

First publication of this article on 23 April 2017

https://www.bortzmeyer.org/icinga-mastodon.html

I use Icinga to monitor my hosts and services. Notification of problems, with Icinga, is not hard-wired in the software, it is delegated to an external program. So, if you know how to send a message from a program, you can use this for notifications. Here, I explain how I did to toot (send) my Icinga notifications to Mastodon.

Mastodon is the latest trendy social network : unlike Twitter, Facebook, Slack or Instagram, it is decentralized, and does not depend on a given corporation. There is an API to perform Mastodon functions. I'm too lazy to write my own program, so I rely on the madonctl <https://github.com/McKael/madonctl>, written in Go. Let's install it. (If you use Debian like me, do note it does not compile on Debian stable, you'll need unstable, or a backport.)

```
% go get github.com/McKael/madonctl
```

Then, if the directory where `go get` installs the binaries is in your PATH, you can use the command :

```
%    madonctl
madonctl is a CLI tool for the Mastodon REST API.

You can use a configuration file to store common options.
...
```

Now, let's configure it with the name of your Mastodon instance, the user name at this instance, and your password :

```
% mkdir -p ~/.config/madonctl
% madonctl config dump -i MY_INSTANCE  -L MY_MASTODON_NAME -P MY_PASSWORD > ~/.config/madonctl/madonctl.yaml
```

Let's test that we can toot (post a message) :

```
%  madonctl toot "Writing a blog article"
- Status ID: 310679
  From: bortzmeyer
  Timestamp: 2017-04-23 18:56:59.141 +0000 UTC
  Contents: Writing a blog article
  URL: https://mastodon.gougere.fr/@bortzmeyer/310679
```

OK, now that the command-line tool works, let's configure Icinga. First, decide if you want your Icinga notifications to be public or not. In the first case, you'll simply send them without anything specific, like I did with the test toot above. In the second case, you'll probably use Mastodon "direct" option, as I do. Your toots will be only visible to you. Let's start with the `users.conf` file to configure the account that will receive the notification toots :

```
object User "icingaadmin" {
...
  email = "ME@MY.EMAIL.SITE"
  vars.mastodon = "MY_MASTODON_NAME"
}
```

I would have preferred to name the variable simply `mastodon` but Icinga does not let me create a new attribute for users (one of the annoying things with Icinga is to find out if a custom attribute is allowed or not ; "it depends" ; and it's not well documented.) So, I use the `vars` dictionary.

Now, let's create the notification command itself. Based on Icinga's email notification script, it will be a simple shell script wrapper around madonctl. `/mastodon-host-notification.sh` will be :

```
#!/bin/sh

export HOME=/var/lib/nagios

template=$(cat <<TEMPLATE
@$USERMASTODON Icinga $NOTIFICATIONTYPE - $HOSTDISPLAYNAME is $HOSTSTATE

Notification Type: $NOTIFICATIONTYPE

Host: $HOSTALIAS
Address: $HOSTADDRESS
State: $HOSTSTATE

Date/Time: $LONGDATETIME

Additional Info: $HOSTOUTPUT

Comment: [$NOTIFICATIONAUTHORNAME] $NOTIFICATIONCOMMENT
TEMPLATE
)

/usr/share/gocode/bin/madonctl toot --visibility direct $(/usr/bin/printf "%b" "$template")
```

And `mastodon-service-notification.sh` will be almost identical :

```
#!/bin/sh

export HOME=/var/lib/nagios

template=$(cat <<TEMPLATE
@$USERMASTODON Icinga $NOTIFICATIONTYPE - $HOSTDISPLAYNAME $SERVICEDISPLAYNAME is $SERVICESTATE

Notification Type: $NOTIFICATIONTYPE

Service: $SERVICEDESC
Host: $HOSTALIAS
Address: $HOSTADDRESS
State: $SERVICESTATE

Date/Time: $LONGDATETIME

Additional Info: $SERVICEOUTPUT

Comment: [$NOTIFICATIONAUTHORNAME] $NOTIFICATIONCOMMENT
TEMPLATE
)

/usr/share/gocode/bin/madonctl toot --visibility direct $(/usr/bin/printf "%b" "$template")
```

(And if you don't know the printf command, it's `timetolearn`.)

Now, let's declare this notification command to Icinga, in `commands.conf` :

```
object NotificationCommand "mastodon-host-notification" {
  command = [ SysconfDir + "/icinga2/scripts/mastodon-host-notification.sh" ]

  env = {
    NOTIFICATIONTYPE = "$notification.type$"
    HOSTALIAS = "$host.display_name$"
    HOSTADDRESS = "$address$"
    HOSTSTATE = "$host.state$"
    LONGDATETIME = "$icinga.long_date_time$"
    HOSTOUTPUT = "$host.output$"
    NOTIFICATIONAUTHORNAME = "$notification.author$"
    NOTIFICATIONCOMMENT = "$notification.comment$"
    HOSTDISPLAYNAME = "$host.display_name$"
    USERMASTODON = "$user.vars.mastodon$"
  }
}

object NotificationCommand "mastodon-service-notification" {
  command = [ SysconfDir + "/icinga2/scripts/mastodon-service-notification.sh" ]

  env = {
    NOTIFICATIONTYPE = "$notification.type$"
    HOSTALIAS = "$host.display_name$"
    HOSTADDRESS = "$address$"
    SERVICESTATE = "$service.state$"
    LONGDATETIME = "$icinga.long_date_time$"
    SERVICEOUTPUT = "$service.output$"
    NOTIFICATIONAUTHORNAME = "$notification.author$"
    NOTIFICATIONCOMMENT = "$notification.comment$"
    SERVICEDISPLAYNAME = "$service.display_name$"
    USERMASTODON = "$user.vars.mastodon$"
  }
}
```

We reference the scripts we just wrote. Note two things :

— The environment variable `USERMASTODON` derives from `user.vars.mastodon`, not just `user.mastodon`, because `mastodon` is not a built-in attribute,

— And we do not define the environment variable `HOME` in the `env` array above, since it seems ignored. Instead, we define it in the scripts (`export HOME=/var/lib/nagios`). Otherwise, madonctl cannot find the configuration file and complains "no instance provided".

Now, let's configure the notifications themselves, in `notifications.conf` :

```
apply Notification "mastodon-icingaadmin" to Host {
  import "mastodon-host-notification"

  user_groups = host.vars.notification.mastodon.groups
  users = host.vars.notification.mastodon.users

  assign where host.vars.notification.mastodon
}

apply Notification "mastodon-icingaadmin" to Service {
  import "mastodon-service-notification"

  user_groups = host.vars.notification.mastodon.groups
  users = host.vars.notification.mastodon.users

  assign where host.vars.notification.mastodon
}
```

We can now define the required variables for each host we're interested in, or in a general template if we want to be "tooted" for all our hosts. In `templates.conf` :

```
template Host "generic-host" {
...
  vars.notification["mastodon"] = {
    groups = [ "icingaadmins" ]
  }
}
```

And that's all. Restart Icinga and wait for the next problem to be "tooted". If you're impatient, break a host or a service to see what happens or, better, use the explicit notification function of Icinga (in the panel for a Host or a Service, near the top). You can see online an example of notification <https://mastodon.gougere.fr/@bortzmeyer/311154>.