Lab 8.1: Nagios Notifications
IN719 Systems Administration

Introduction

Right now our Nagios servers are monitoring our servers and will note when something is amiss, but unless we're actually looking at the dashboard when something goes wrong, it wont do us much good. When Nagios identifies a fault in our systems we need it to notify us. Before we set this up, let's think a bit about our strategy for notifications. Getting notifications right is a trial-and-error process. We definitely do want to receive notifications when things go wrong, but we also don't want to be bothered by them for matters that are not urgent. If you receive notifications for things that aren't critical, eventually you will train yourself to ignore them and eventually you'll find you've ignored an important notifications. There's no one-size-fits-all notification setup. You just have to experiment and adjust to find the right level. Just a few days ago I modified an alert setting on my own systems when I found I was getting too many false alarms.

Then there is the matter of how to send the notifications. Email is by far the easiest method to configure, but email is only really appropriate for less urgent or informational notifications. At the other end of the scale, we could set up dedicated hardware capable of sending SMS alerts. This is a great approach, but since we're nearly also working with cloud hosted systems these days, hardware solutions aren't an option. Probably the most common notification method used now is to use Nagios plugins that relay alerts through services like Pager Duty or OpsGenie and have those service push alerts to team members via SMS or app notifications. These services cost money though, and I'm too cheap to foot the bill for this paper. We can, however, get a very similar result at no cost by using the Nagios Slack plugin and sending notifications to a Slack channel. Then you can use the Slack mobile app and get notifications on your phone.

So, in this lab we will configure our Nagios servers to send notifications via Slack.

1 Enable Nagios notifications in our Slack workspace

I've done this for our workspace, but for completeness I'll just point out that it's a required step.

2 Install Slack

You will need to install the Slack mobile app on your device. Log into the in719 Slack instance. If you haven't already, set up a channel for your team. Don't spam `#general` with your alerts.

3 Set up the Slack plugin for Nagios

Note that we're describing the steps required on our Nagios servers, but you'll need to set up one or more Puppet modules to manage this.

1. Install the Perl modules `libwww-perl libcrypt-ssleay-perl`.
2. Download the plugin file from `https://raw.github.com/tinyspeck/services-examples/master/nagios.pl`. Edit the file to set the values of `$opt_domain` and `$opt_token` as directed by the lecturer.
3. Copy the file to `/usr/lib/nagios/plugins` and set its permissions to 0755.
4. Create a file `/etc/nagios-plugins/config/slack.cfg` with the following contents (change the channel name to your channel):

```
define command {
      command_name notify-service-by-slack
      command_line /usr/lib/nagios/plugins/nagios.pl -field slack_channel=#your-channel -field \
      HOSTALIAS="$HOSTNAME$" -field SERVICEDESC="$SERVICEDESC$"  \
      -field SERVICESTATE="$SERVICESTATE$"  -field SERVICEOUTPUT="$SERVICEOUTPUT$" \
      -field NOTIFICATIONTYPE="$NOTIFICATIONTYPE$"
}

define command {
      command_name notify-host-by-slack
      command_line /usr/lib/nagios/plugins/nagios.pl -field slack_channel=#your-channel -field \
      HOSTALIAS="$HOSTNAME$" -field HOSTSTATE="$HOSTSTATE$" \
      -field HOSTOUTPUT="$HOSTOUTPUT$" \
      -field NOTIFICATIONTYPE="$NOTIFICATIONTYPE$"
}
```

5. Now, for any contacts you define, you have the option of using for your host and service notification command the commands `notify-host-by-slack` and `notify-service-by-slack`.

4 Configure a Nagios contact

Now we have notification commands ready to use. To use them, we need to configure a contact. We will create a special Slack contact to receive these notifications. Add the following contact resource to your Puppet module that manages Nagios:

```
nagios_contact { 'slack':
              target => '/etc/nagios3/conf.d/ppt_contacts.cfg',
              alias => 'Slack',
              service_notification_period => '24x7',
              host_notification_period => '24x7',
              service_notification_options => 'w,u,c,r',
              host_notification_options => 'd,r',
              service_notification_commands => 'notify-service-by-slack',
              host_notification_commands => 'notify-host-by-slack',
              email => 'root@localhost',
}
```

To make the contact useful we place it in a contact group. Below is an example Puppet resource for this.

```
nagios_contactgroup { 'slackgroup':
               target => '/etc/nagios3/conf.d/ppt_contactgroups.cfg',
               alias => 'Slack channel',
               members => 'slack',
}
```

For any services that you wish to be notified by Slack, modify the `contact_groups` setting to use the new `slackgroup` contact group.

5 Testing

Once everything is in place, we need to test to see if the alerts are received. One way to do this is to temporarily reconfigure a service check so that it will trigger a critical alert under normal circumstances. For example, look at this check command that monitors free disk space:

```
check_command                   check_all_disks!20%!10%
```

This check will fire a critical alert when the disk has 10% free space remaining. If we temporarily change that setting to 90%, we should get an alert that can notify via Slack.

Once you have notifications working, consider which Nagios checks you would like to send notifications and configure them to do so. Getting this working well will contribute to your success in the second assessment.