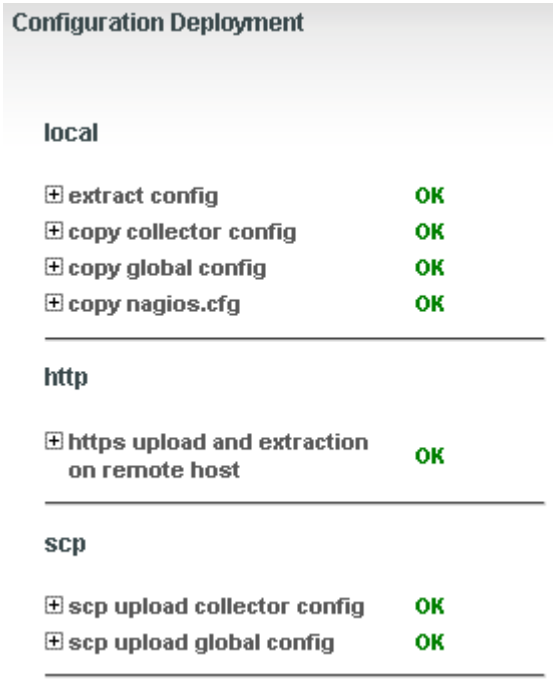
Deployment

This module provides you with some deployment possibilities.

We hope its usefully and will also help you, to bring the nagios configuration easy to its location.

About



To bring the configuration to your Nagios, you first have to generate the configuration.

Until 1.2.6 you had to move the files by yourself.
Now, **since NConf version 1.3**, we provide the "deployment" module which can help you doing deployment right from the NConf GUI.

- Simple configuration
- Direct feedback in the GUI
- Separate modules (community can create new ones)
- Modules can be integrated easily

Until now we support following deployment modules:

- **local**
- **scp**
- **rsync**
- **HTTPS/HTTP**

setup / configure deployment

The configuration for the deployment is done in the `config/deployment.ini`

Some examples are included in the `config.orig/deployment.ini` file.

Copy this to `config/deployment.ini` and remove the semicolons ";" of the section you want to activate.
Be sure to use correct path's with appropriate permissions (for webserver)

For more details read the whole documentation.

update from 1.2.6

If you have updated from a previous NConf version, you can read this part to migrate/setup your *basic* deployment.
If you made a more complex deployment, or want to know more about which type of deployment are available, please refer to the section below : Types.

1. copy example configuration from `config.orig/deployment.ini` to `config/deployment.ini`
2. migrate following variables from `config/nconf.php` to `config/deployment.ini`
   - CONF_DEPLOY_URL → host
   - CONF_DEPLOY_USER → user
   - CONF_DEPLOY_PWD → password
3. delete or comment out the existing CONF_DEPLOY values in `config/nconf.php`

example

an example from the old config to the new one:

old config in `config/nconf.php`

```
define('CONF_DEPLOY_URL',  "https://webserver.mydomain.com/incoming_config.php");
define('CONF_DEPLOY_USER', "deployUser");
define('CONF_DEPLOY_PWD',  "deployPass");
```

new config in `config/deployment.ini`

```
[webserver mydomain.com]
type        = https
source_file = "/var/www/nconf/output/NagiosConfig.tgz"
host        = "https://webserver.mydomain.com/incoming_config.php"
user        = "deployUser"
password    = "deployPass"
```

Types

This is the overview of all deployment types.
It can help you understand how to read the information given in the types documentation

How to read the information

"`YOUR_SPECIFIC_INPUT`"

- UPPERCASE and CODE-text in double quotes means you have to enter your specific *text* / *path* / *command* or what else it refers.

`option1/option2`

- lowercase are options, the "/" (slash) splits the options, so you have to choose ether option1 OR option2

title

- for each configuration, you have to create a section which starts with a text in brackets: **`[some text]`**
- this will also be used as title of the output in the deployment-GUI
- if you want to define an other text as the one in brackets, you can use the option **title**

```
[bracket text will be the title]
...
```

```
[some bracket text]
title = "this will be the title"
...
```

Deployment types

Detailed documentation about the different types including examples:

- local
- scp
- rsync
- https

restarting nagios

We have implemented the possibility to set a command for restarting nagios.

Nevertheless you have to configure your system that its possible to do this.
That means you have to grant permissions for the webserver user(mostly apache) to execute your specified command like restarting nagios.

Most system administrators (you) should know how to configure that for their systems. We will show you a possible way here:

sudo

sudo allows a permitted user to execute a command as the superuser or another user.
So we could add the apache user to the sudoers list with the appropriate command.
Here we also set the *NOPASSWD* option

- visudo (edit the sudoers file)
- `apache  ALL=NOPASSWD: /etc/rc.d/init.d/nagios reload`

Now it is possible for the user apache to restart nagios right in the deployment.

Additional information

For security reason, you could also limit the access for apache (not using root):

Allow apache to run things as user “nagios”:

```
apache ALL = (nagios) NOPASSWD: /your/comand...
```

Then you can set the `reload_command` to something like this:

```
reload_command = "/usr/bin/sudo -u nagios /your/comand..."
```

Tip: If you have troubles, check if permissions are set correct for the `/var/nagios/spool/checkresults`.

permissions

you could also set permissions on the nagios binary which can restart it.

example:

- give apache execute rights on */etc/rc.d/init.d/nagios*

Configuration examples

Here we show you some configurations which could be useful for you.

deploy extracted config files to different server

first extract files to local existing directory:

```
[local deploy extract]
type        = local
source_file = "/var/www/nconf/output/NagiosConfig.tgz"
target_file = "/tmp/nagios_config/"
action      = extract
```

now copy all these files with scp to other server

```
[scp dummy-host]
type        = scp
host        = "dummy-host"
source_file = "/tmp/nagios_config/"
target_file = "~"
user        = dummy-user
identity_file = "id_dsa"
ssh_options   = " -o 'StrictHostKeyChecking no' -o 'ConnectTimeout 15'"
```

Now the files should be copied to your server.

for developers

If you want to make other deployment modules, contact us. You could also just try it out with an existing one.

the HTTPS module, seems to be the easiest to look how the basic structure looks like

Send us your deployment module if you want to share it with other peoples!

local

This method can be used for local deployments. This means your Nagios is on the same machine as NConf.

You can also use this module for preparing some files for an other module.

config options

option (bold=required)	possible values	description
type	local	defines this deployment type
title	"`TITLE`"	…
source_file	"`FULL_FILE_PATH`"	define the PATH of the source file (PATH with FILENAME)
target_file	"`FULL_FILE_PATH`"	define the target PATH **NOTE**: If you copy a **file**, be sure to use PATH and FILENAME (`/etc/nagios.cfg`) If you copy a **directory** or use **extract** be sure that PATH ends with a slash "/" (`/etc/nagios/`) These commands will be executed as the user your web server runs under. Be sure any files/directories used in these commands are accessible as necessary by this user. If possible, the script will try to generate the target directory if it does not exist.
action	extract/copy	extract: will unpack the basic NagiosConf.tgz file copy: will copy single file or directory

reload_command	"`SHELL_COMMAND`"	define the SHELL_COMMAND to execute after copy/extract
		nagios restart: "`/etc/init.d/nagios reload`"
		NOTE: check permissions
		This command will escape shell metacharacters (php: escapeshellcmd() [http://php.net/manual/en/function.escapeshellcmd.php])
		You can define multiple commands to execute when you use empty brackets (**[]**) after the option:
		`reload_command[] = "command1"`
		`reload_command[] = "command2"`
		They are executed in the order written

reload_command
For more information about reloading nagios & permissions, please have also a look here:

- restarting_nagios

examples

extract the files direct to nagios and reload nagios

```
1 [local deploy extract]
2 type        = local
```

```
source_file = "/var/www/nconf/output/NagiosConfig.tgz"
target_file = "/etc/nagios/"
action      = extract
reload_command = "/etc/init.d/nagios reload"
```

move the tarball to other location

```
[local deploy tgz]
type        = local
source_file = "/var/www/nconf/output/NagiosConfig.tgz"
target_file = "/tmp/NagiosConfig.tgz"
action      = copy
```

scp

This method can be used if you want to deploy the config files over scp.

config options

option (bold=required)	possible values	description
type	scp	defines this deployment type
title	"TITLE"	…
host	"HOSTNAME or HOST_IP"	Host IP or FQDN
source_file	"FILE_PATH"	define the PATH of the source files and directories multiple paths should be splitted with a space
target_file	"FILE_PATH"	define the target PATH
user	"SCP_USER_NAME"	scp user
identity_file	"ID_DSA_FILE_PATH"	location of the identity_file relative path will look in the key folder in the scp module static path (starting with "/")
ssh_options	"SSH_OPTIONS"	ssh options
reload_command	"REMOTE_SHELL_COMMAND"	define the SHELL_COMMAND to execute after copy/extract **nagios restart**: "/etc/init.d/nagios reload" **NOTE:** check apache user permissions

examples

send tarball to a different server

the servers name is "dummy-host"

```
[scp dummy-host]
type        = scp
host        = "dummy-host"
source_file = "/var/www/nconf/output/NagiosConfig.tgz"
target_file = "~"
user        = dummy-user
identity_file = "id_dsa"
ssh_options   = " -o 'StrictHostKeyChecking no' -o 'ConnectTimeout 15'"
```

copy already extracted data to a remote host and restart nagios

- of course you first have to extract the configuration with the local module
- (in this example to "/tmp/nconf/")
- you can define multiple source files and directories :(for example: collector1 and global)

```
[scp to nagios]
type        = scp
host        = "192.168.100.101"
source_file = "/tmp/nconf/collector1/ /tmp/nconf/global/"
target_file = "/etc/nagios/"
```

```
user         = "nconf-copy"
identity_file  = "/home/nconf-copy/.ssh/id_dsa"
ssh_options    = " -o 'StrictHostKeyChecking no' -o 'ConnectTimeout 15'"
reload_command = "/etc/init.d/nagios reload"
```

rsync

This method can be used if you want to deploy the config files over rsync. This will only upload the changed files.

The reload command (reload nagios) will also only be executed if some files are synced.

config options

option (bold=required)	possible values	description
type	rsync	defines this deployment type
title	"TITLE"	…
host	"HOSTNAME or HOST_IP"	Host IP or FQDN
source_file	"FILE_PATH"	define the PATH of the source files and directories multiple paths should be splitted with a space **ATTENTION:** unlike scp, rsync changes the behavior of the slash after a directory name, please read the info box for more information
target_file	"FILE_PATH"	define the target PATH
user	"USER_NAME"	rsync/ssh user
rsync_options	"RSYNC_OPTIONS"	rsync options *for example rsync over ssh:* *" -caz -e 'ssh -i include/modules/deployment/rsync/id_dsa -o StrictHostKeyChecking=no -o ConnectTimeout=15'"*
reload_command	"REMOTE_SHELL_COMMAND"	define the SHELL_COMMAND to execute after copy/extract **nagios restart**: "/etc/init.d/nagios reload" **NOTE:** check apache user permissions
identity_file	"ID_DSA_FILE_PATH"	**only used for reload_command ssh connection!** location of the identity_file relative path will look in the key folder in the rsync module static path (starting with "/")
ssh_options	"SSH_OPTIONS"	**only used for reload_command ssh connection!** ssh options

examples

rsync already extracted data to a remote host and restart nagios

- of course you first have to extract the configuration with the local module
- (in this example to "/tmp/nconf/")
- you can define multiple source files and directories :(for example: collector1 and global)

```
[rsync to nagios]
type        = rsync
host        = "192.168.100.101"
```

```
source_file = "/tmp/nconf/collector1 /tmp/nconf/global"
target_file = "/etc/nagios/"
user        = "nconf-copy"
rsync_options  = " -caz -e 'ssh -i /home/nconf-copy/.ssh/id_dsa -o StrictHostKeyCh
reload_command = "/etc/init.d/nagios reload"
identity_file  = "/home/nconf-copy/.ssh/id_dsa"
ssh_options    = " -o 'StrictHostKeyChecking no' -o 'ConnectTimeout 15'"
```

http

This method can be used if you want to deploy the config files over http or https.

This function is adapted from the previous releases where the CONF_DEPLOY_* settings only pushed to 1 server. Now you can configure multiple hosts, and so reach multiple hosts over http.

(or configure multiple http parts with different options)

config options

option (bold=required)	possible values	description
type	http	defines this deployment type
title	"`TITLE`"	…
host	"`HTTP/HTTPS_LINK`"	http or https address of script, which takes the data. for details on our included script please have a look here (incoming_config.php) **host[]** = "…" you can also set multiple hosts for one config section
source_file	"`FILE_PATH`"	define the PATH of the source file
user	"`HTTPS_USERNAME`"	https user
password	"`HTTPS_PASSWORD`"	https password
remote_execute	"`TRUE/FALSE`"	define if the `$exec_command` (configured in the incoming script) should be executed after successful upload **NOTE:** check apache user permissions
remote_action	extract/copy	**extract:** will unpack the basic NagiosConf.tgz file **copy:** will copy single file or directory

multiple hosts

If you want to use the configuration for multiple hosts, use the "[]" for each host config:

- host[] = "`HOST_ADDRESS_1`"
- host[] = "`HOST_ADDRESS_2`"

incoming_config.php

location:

- you can find this script in the NConf directory under

```
nconf/ADD-ONS/incoming_config.php
```

usage:

- this file should be placed on the server, where you want to retrieve the files sent over the HTTP module
- it receives the files, copy it to a directory
- Its also possible to configure a restart command, to refresh nagios on this remote machine after successful upload.

Here is a list of variables, which **must** be configured in the script, before using it

option (bold=required)	possible values	description
$targetdir	"FILE_PATH"	define the target PATH
$exec_command	"EXEC_COMMAND"	define the exec_command (restart nagios or what you want to do)
$tar	define tar command	
$gunzip	"GUNZIP_COMMAND"	define gunzip command

examples

send tarball to a HTTPS host

```
[https dummy-host]
type        = http
source_file = "/var/www/nconf/output/NagiosConfig.tgz"
host        = "https://dummy-host.mydomain.com/incoming_config.php"
user        = "deployUser"
password    = "deployPass"
```

send tarball to multiple HTTPS host

```
[https dummy-host-1 and dummy-host-2]
type        = http
source_file = "/var/www/nconf/output/NagiosConfig.tgz"
host[]      = "https://dummy-host-1.mydomain.com/incoming_config.php"
host[]      = "https://dummy-host-2.mydomain.com/nconf_files/incoming_config.php"
user        = "deployUser"
password    = "deployPass"
```

you could also just use a HTTP connection

This will only use **HTTP** *(unsecure)*, you should know what you are doing if you want to use HTTP. We do NOT recommand you to use this (especially if you use connections outside your infrastructure)

```
[http dummy-host]
type        = http
source_file = "/var/www/nconf/output/NagiosConfig.tgz"
host        = "http://dummy-host.mydomain.com/incoming_config.php"
user        = ""
password    = ""
```