FreeBSD Administration - Support #543

Install a Puppet Master, Puppet Dashboard and Nagios Server with Nginx on FreeBSD

01/24/2015 08:53 PM - Daniel Curtis

Status:	Closed	**Start date:**	04/29/2014
Priority:	Normal	**Due date:**	
Assignee:	Daniel Curtis	**% Done:**	100%
Category:	Automated Server Management	**Estimated time:**	5.00 hours
Target version:	FreeBSD 9	**Spent time:**	8.00 hours

Description

Puppet and Nagios have been two of the most useful tools for my small business; puppet allowing me to manage many servers and services using Ruby to write my configs, and nagios allowing me to monitor the servers and services. This is a simple guide to setup something similar to the system I have on FreeBSD 9.2.

This will set up 2 web applications that can be viewed in any modern web browser at:

- **Default blank landing page** - http://puppet.example.com
- **Puppet Dashboard** - http://puppet.example.com:3000
- **Nagios** - http://nagios.example.com:8000
- **Puppet Master** - https://puppet.example.com:8140

The first will be the puppet dashboard and the second will be the nagios dashboard.

Prepare the Server

- Once the server baseline has been installed and root access has been obtained make sure the server is up to date:

```
pkg update && upgrade
portsnap fetch extract
```

- Portmaster will be useful for upgrading packages:

```
cd /usr/local/ports/ports-mgmt/portmaster
make install clean
pkg2ng
```

Install MariaDB server

- Start by installing the mariadb-server and mariadb-client packages:

```
portmaster databases/mariadb55-{server,client}
```

Configure MariaDB server

- Copy a base MariaDB configuration to use

```
cp /usr/local/share/mysql/my-small.cnf /var/db/mysql/my.cnf
```

- Tuning: Copy one of the default config files and change the max packet size to allow for the 17 MB data rows that Dashboard can occasionally send:

```
vi /var/db/mysql/my.cnf
```

- and modify max_allowed_packet to 32M

```
max_allowed_packet = 32M
```

- Enable and start MariaDB

```
echo 'mysql_enable="YES"' >> /etc/rc.conf
service mysql-server start
```

- Prepare Database for use by running the secure installation. Choose a root password and answer yes to all questions.

```
mysql_secure_installation
```

Create MariaDB Databases and Users

- Login to MariaDB and create appropriate databases and users.

```
mysql -u root -p
```

- and run the following SQL queries to create the **puppetmaster** database and user:

```
CREATE DATABASE dashboard_production CHARACTER SET utf8;
CREATE USER 'puppetmaster'@'127.0.0.1' IDENTIFIED BY 'SecretPuppetMasterPassword';
GRANT ALL PRIVILEGES ON puppetmaster.* TO 'puppetmaster'@'127.0.0.1';
flush privileges;
```

- and run the following SQL queries to create the puppet **dashboard** database and user:

```
CREATE DATABASE dashboard_production CHARACTER SET utf8;
CREATE USER 'dashboard'@'127.0.0.1' IDENTIFIED BY 'SuperSecretPassword';
GRANT ALL PRIVILEGES ON dashboard_production.* TO 'dashboard'@'127.0.0.1';
flush privileges;
```

- and run the following SQL queries to create the **nagios** database and user:

```
CREATE DATABASE nagios CHARACTER SET utf8;
CREATE USER 'nagios'@'127.0.0.1' IDENTIFIED BY 'SecretNagiosPassword';
GRANT ALL PRIVILEGES ON nagios.* TO 'nagios'@'127.0.0.1';
flush privileges;
```

Install Puppet Master

- Install the puppet package

```
portmaster sysutils/puppet devel/rubygem-rake sysutils/rubygem-bundler databases/rubygem-activ
erecord databases/rubygem-sqlite3 databases/rubygem-activerecord-mysql-adapter textproc/libxsl
t devel/git www/node
```

- Enable puppet in /etc/rc.conf:

```
echo 'puppet_enable="YES"' >> /etc/rc.conf
```

Puppet Initial Testing

- At this point, Puppet needs to be started so that all its SSL keys can be generated. This gives the chance to test that Puppet does work before anything else gets stacked on as well as ensures the SSL keys referenced by Nginx's config file are generated and in place before that step.

```
service puppetmaster onestart
```

- On **client.example.com** - start Puppet on the client system

```
service puppet onestart
```

  - Or

```
puppet agent -vt --waitforcert 60
```

- On **puppet.example.com** - sign client.example.com's SSL key on the Puppetmaster

```
puppet cert sign client.example.com
```

- On **client.example.com** - Run a test on the client to ensure it works and do a onestop afterwards

```
puppet agent --test
service puppet onestop
```

- **NOTE**: I encountered a problem while migrating the Admin node, the new server uses a newer version of Puppet, and broke the fileserver feature. To work around this:
  1. In **/usr/local/etc/puppet/fileserver.conf** put the name of your mount point, the path, and an allow * directive.:

```
[files]
  path /usr/local/etc/puppet/files
  allow *
```

  2. In **/usr/local/etc/puppet/auth.conf**:
     Use a regular expression path to match both the file_metadata and file_content endpoints followed by the name of your custom mount point. Then, use any combination of allow and allow_ip directives to control access.

```
path ~ ^/file_(metadata|content)/files/
auth yes
allow /^(.+\.)?example.com$/
allow_ip 192.168.100.0/24
```

Configure Puppet Master

- Configure your [puppetmasterd] section to reflect these settings:

```
vi /usr/local/etc/puppet/puppet.conf
```

  - And add/modify the following:

```
[puppetmasterd]
storeconfigs = true
dbadapter = mysql
dbuser = puppetmaster
dbpassword = SecretPuppetMasterPassword
dbserver = localhost
dbsocket = /var/run/mysqld/mysqld.sock
```

- To optimize some often run Puppet queries on your MySQL database, log in to the MariaDB server:

```
mysql -u root -p
```

  - And run the following to create the index:

```
create index exported_restype_title on resources (exported, restype, title(50));
```

Install Puppet Dashboard

Now its time to install Puppet Dashboard, a web frontend to display puppet reports.

- Install puppet-dashboard from git

```
cd /usr/local/www
git clone git://github.com/sodabrew/puppet-dashboard.git
```

- Manually create the 'puppet-dashboard' user and group:

```
pw groupadd -n puppet-dashboard -g 800
pw useradd -n puppet-dashboard -c "Puppet Dashboard,,," -u 800 -g puppet-dashboard -s /usr/sbi
n/nologin
```

- Then provide the required permissions to /usr/local/www/puppet-dashboard

```
chown -R puppet-dashboard:puppet-dashboard /usr/local/www/puppet-dashboard
```

Configure Puppet Dashboard

- Copy the example database YAML file and update with database information:

```
cd /usr/local/www/puppet-dashboard/config
cp database.yml.example database.yml
```

- Then edit the database.yml with the corrected information; make sure to replace the password and host:

```
vi database.yml
```

  - and modify the following parameter accordingly:

```
production:
  database: dashboard_production
  username: dashboard
  password: SuperSecretPassword
  encoding: utf8
  adapter: mysql2
```

- Change the ownership and harden the permissions:

```
chown puppet-dashboard:puppet-dashboard database.yml
chmod 660 database.yml
```

- Copy the example settings YAML file, no changes needed:

```
cd /usr/local/www/puppet-dashboard/config
cp settings.yml.example settings.yml
chown puppet-dashboard:puppet-dashboard settings.yml
chmod 660 settings.yml
```

- Fix shebang line in External Node Classifier Script.

```
sed -i '' -e 's/#! \/usr\/bin\/ruby/#!\/usr\/local\/bin\/ruby/' /usr/local/www/puppet-dashboar
d/bin/external_node
```

- Install gems required in the 'Gemfile' via the Rubygem Bundler. If the postgresql gem bundles is required additional dependencies are needed:

```
cd /usr/local/www/puppet-dashboard
bundle install --path vendor/bundle --without postgresql
```

- Generate secret_token. Cleanup any errors and the default token after generating the new one.

```
echo "secret_token: `bundle exec rake secret`" >> config/settings.yml
vi config/settings.yml
```

Setting up the database for Puppet Dashboard

- At this point the database was already installed with some blank tables. We need to run rake to finish the process with the database structure needed:

```
cd /usr/local/www/puppet-dashboard
env RAILS_ENV=production bundle exec rake db:setup
```

Testing That Dashboard is Working

- Run Dashboard using Ruby's built-in WEBrick server to validate functionality. It will be available at http://puppet.example.com:3000

```
cd /usr/local/www/puppet-dashboard
su -m puppet-dashboard -c 'bundle exec rails server'
```

- Before going into a production environment, Dashboard 2.0 must precompile assets for production:

```
env RAILS_ENV=production bundle exec rake assets:precompile
```

- Chown any files created up until now to the right owner:

```
chown -R puppet-dashboard:puppet-dashboard /usr/local/www/puppet-dashboard
```

Configuring Puppet

All agent nodes have to be configured to submit reports to the master. The master has to be configured to send reports to Dashboard. If you already have a working Puppet installation you can configure it to distribute the updated puppet.conf to your hosts.

- **puppet.conf** (on each agent)

```
[agent]
  report = true
```

- **puppet.conf** (on the Puppetmaster)

```
[master]
  reports = store, http
  reporturl = http://puppet.example.com:3000/reports/upload
  node_terminus = exec
  external_nodes = /usr/bin/env PUPPET_DASHBOARD_URL=http://puppet.example.com:3000 /usr/local
/www/puppet-dashboard/bin/external_node
```

li

- Testing Puppet's Connection to Dashboard. A new background task should show in the Dashboard UI at http://puppet.example.com:3000

```
puppet agent --test
```

- Dashboard ships a worker process manager under script/delayed_job. It can manually start delayed jobs via the following command:

```
su -m puppet-dashboard -c 'env RAILS_ENV=production bundle exec script/delayed_job -p dashboar
d -n 2 -m start'
```

Delayed Job Worker Init Script

However, rather than manually triggering background workers, this rc script will accomplish the same thing and ensure the background jobs get started on the next reboot.

- Create puppet dashboard FreeBSD init script:

```
vi /usr/local/etc/rc.d/dashboard_workers
```

- and add the following

```
#!/bin/sh

# PROVIDE: dashboard_workers
# REQUIRE: LOGIN
# KEYWORD: shutdown

# By default dashboard_workers uses flags '-n 1' for 1 worker.  This should be
# adjusted to the number of CPU cores.
dashboard_workers_enable=${dashboard_workers_enable:-"NO"}
dashboard_workers_flags=${dashboard_workers_flags:-"-n 1"}
# The default rails environment is set to production
dashboard_workers_env=${dashboard_workers_env:-"/usr/bin/env PATH=${PATH}:/usr/local/bin R
AILS_ENV=production"}
# The default user is set to puppet-dashboard and install location is set to
# /usr/local/share/puppet-dashboard.
dashboard_workers_user=${dashboard_workers_user:-"puppet-dashboard"}
dashboard_workers_chdir=${dashboard_workers_chdir:-"/usr/local/www/puppet-dashboard"}

. /etc/rc.subr

name="dashboard_workers"
rcvar="dashboard_workers_enable"
load_rc_config $name
extra_commands="reload run zap status"

# All commands call the same function and strip the fast|one|quiet prefix
# to deliver to the bundler.
reload_cmd="f_dashboard_workers reload"
restart_cmd="f_dashboard_workers restart"
run_cmd="f_dashboard_workers run"
start_cmd="f_dashboard_workers start"
status_cmd="f_dashboard_workers status"
stop_cmd="f_dashboard_workers stop"
zap_cmd="f_dashboard_workers zap"

# Use the function's ARVG $1 as the bundler program's '-m' flag
f_dashboard_workers() {
    cd $dashboard_workers_chdir && \
    su -m "$dashboard_workers_user" \
        -c "${dashboard_workers_env} bundle exec script/delayed_job ${rc_flags} -p dashboa
rd -m $1" || \
    echo "Failed to $1 dashboard_workers"
}

run_rc_command "$1"
```

- And make it executable:

```
chmod +x /usr/local/etc/rc.d/dashboard_workers
```

- With that in place, we need to override the defaults and enable the script along with setting '-n 4' workers to match the number of processor cores and ensure it's ready for a production workload.

```
echo 'dashboard_workers_enable="YES"' >> /etc/rc.conf
echo 'dashboard_workers_flags="-n 4"' >> /etc/rc.conf
service dashboard_workers start
```

Puppet Dashboard on Nginx with Passenger

- Install Nginx with Passenger

```
portmaster www/nginx
```

NOTE: Make sure to enable [X]PASSENGER during the nginx configuration.

- Install the Passenger gem:

```
portmaster www/rubygem-passenger
```

NOTE: Make sure to enable NGINX when running make config

- Create a configuration directory to make managing individual server blocks easier

```
mkdir /usr/local/etc/nginx/conf.d
```

- Configuring Nginx with Passenger, edit the **main nginx configuration file**:

```
vi /usr/local/etc/nginx/nginx.conf
```

  - And add/modify the following

```
user  www www;
worker_processes  4;
error_log  /var/log/nginx/error.log notice;
pid        /var/run/nginx.pid;

events {
  worker_connections  1024;
}

http {
  passenger_root /usr/local/lib/ruby/gems/2.1/gems/passenger-5.0.4;
  passenger_ruby /usr/local/bin/ruby;
  passenger_max_pool_size 15;
  passenger_pool_idle_time 300;
  #passenger_spawn_method direct; # Uncomment on Ruby 1.8 for ENC to work

  include       mime.types;
  default_type  application/octet-stream;
  sendfile      on;
  tcp_nopush    on;
  keepalive_timeout  65;
  tcp_nodelay        on;

  # Load config files from the /etc/nginx/conf.d directory
  include /usr/local/etc/nginx/conf.d/*.conf;
}
```

- Then add a **default site server block**:

```
vi /usr/local/etc/nginx/conf.d/default.conf
```

  - Add the following:

```
server {
  listen 80 default;
  server_name _;

  index index.html index.php;
  root /usr/local/www;

  # IP and IP ranges which should get access
  allow 10.0.0.0/24;
  allow 10.1.0.1;
  # all else will be denied
  deny all;

  # basic HTTP auth
  auth_basic "Restricted";
  auth_basic_user_file htpasswd;

  location ~ \.cgi$ {
    try_files $uri =404;
    include fastcgi_params;
    fastcgi_pass unix:/var/run/fcgiwrap/fcgiwrap.sock;
    fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
    fastcgi_param REMOTE_USER $remote_user;
  }

  location ~ \.php$ {
    try_files $uri =404;
    include fastcgi_params;
    fastcgi_pass 127.0.0.1:9000;
    fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
  }
}
```

- And create a **puppet master server block**:

```
vi /usr/local/etc/nagios/conf.d/puppetmaster.conf
```

  - And add the following:

```
server {
  listen       8140 ssl;
  server_name  puppet.example.com;

  passenger_enabled          on;
  passenger_set_header       HTTP_X_CLIENT_DN $ssl_client_s_dn;
  passenger_set_header       HTTP_X_CLIENT_VERIFY $ssl_client_verify;
  passenger_user             puppet;
  passenger_group            puppet;
  access_log                 /var/log/nginx/puppet_access.log;

  root                       /usr/local/etc/puppet/rack/public;
  ssl_certificate            /var/puppet/ssl/certs/puppet.example.com.pem;
  ssl_certificate_key        /var/puppet/ssl/private_keys/puppet.example.com.pem;
  ssl_crl                    /var/puppet/ssl/ca/ca_crl.pem;
  ssl_client_certificate     /var/puppet/ssl/certs/ca.pem;
  ssl_ciphers                SSLv2:-LOW:-EXPORT:RC4+RSA;
  ssl_prefer_server_ciphers  on;
  ssl_verify_client          optional;
  ssl_verify_depth           1;
  ssl_session_cache          shared:SSL:128m;
  ssl_session_timeout        5m;
}
```

- And create a **puppet dashboard server block**:

```
vi /usr/local/etc/nagios/conf.d/puppet-dashboard.conf
```

  - And add the following:

```
server {
  listen       3000;
  server_name  puppet.example.com;

  passenger_enabled on;
  passenger_user    puppet-dashboard;
  passenger_group   puppet-dashboard;

  access_log        /var/log/nginx/dashboard_access.log;

  root              /usr/local/www/puppet-dashboard/public;
}
```

- Create the log directory to prevent issues on startup:

```
mkdir /var/log/nginx
```

- Enable a daily log file rotation via newsyslog.conf:

```
printf "/var/log/nginx/*.log\t\t\t644 7\t * @T00 JG /var/run/nginx.pid 30\n" >> /etc/newsyslog
.conf
```

- If the puppetmaster service is still running from earlier testing, stop it now:

```
service puppetmaster onestop
```

- With initial setup of the Puppetmaster done, a RACK file that Nginx will use to start the Ruby application will be needed:

```
mkdir -p /usr/local/etc/puppet/rack/public
```

- Create the config.ru file

```
vi /usr/local/etc/puppet/rack/config.ru
```

  - And add the following

```
# Trimmed back FreeBSD Version of https://github.com/puppetlabs/puppet/blob/master/ext/rac
k/files/config.ru
$0 = "master"
ARGV << "--rack"
ARGV << "--confdir" << "/usr/local/etc/puppet"
ARGV << "--vardir" << "/var/puppet"
require 'puppet/util/command_line'
run Puppet::Util::CommandLine.new.execute
```

- Make the script executable

```
chown -R puppet:puppet /usr/local/etc/puppet/rack
```

- Enable nginx service and start it. At this point basic functionality is online:

```
echo 'nginx_enable="YES"' >> /etc/rc.conf
service nginx start
```

  - **NOTE**: I had a problem connecting to the puppet master, getting 403 errors, after I had setup Nginx/Passenger. The problem was with the /usr/local/etc/puppet/puppet.conf and there being a couple of parameters that needed to be taken out. If the following two lines are present, remove or comment them out. They are are provided in the nginx.conf:

```
#ssl_client_header = SSL_CLIENT_S_DN
#ssl_client_verify_header = SSL_CLIENT_VERIFY
```

Configuring Dashboard - Advanced Features

- Generating Certs and Connecting to the Puppet Master
  With separate Puppet/Dashboard systems the puppet cert sign dashboard will be on the Puppetmaster:

```
cd /usr/local/www/puppet-dashboard
su -m puppet-dashboard -c 'bundle exec rake cert:create_key_pair'
su -m puppet-dashboard -c 'bundle exec rake cert:request'
puppet cert sign dashboard
su -m puppet-dashboard -c 'bundle exec rake cert:retrieve'
```

Enabling Inventory Support

- Edit the **auth.conf** file on Puppet master

```
vi /usr/local/etc/puppet/auth.conf
```

  - Add the following:

```
path /facts
auth yes
method find, search
allow dashboard
```

Enabling the Filebucket Viewer

- Edit the **site.pp** on the Puppet master:

```
vi /usr/local/etc/puppet/manifests/site.pp
```

  - Add the following:

```
filebucket { "main":
  server => "{your puppet master}",
  path => false,
}
```

  - In either site.pp, in an individual init.pp, or in a specific manifest.

```
File { backup => "main" }
```

- Go back and add the line for Inventory Support

```
vi /usr/local/www/puppet-dashboard/config/settings.yml
```

  - And change the following parameters:

```
enable_inventory_service: true
use_file_bucket_diffs: true
```

- With all the updates made, restart so that it takes effect:

```
service nginx restart
```

- For future maintenance, periodic jobs to prune old reports and run DB optimization.

```
mkdir -p /usr/local/etc/periodic/monthly
vi /usr/local/etc/periodic/monthly/clean_dashboard_database.sh
```

  - And add the following:

```
#!/bin/sh
cd /usr/local/share/puppet-dashboard && \
    echo "Pruning Old Reports from Puppet Dashboard Database" && \
    /usr/bin/su -m puppet-dashboard -c '/usr/local/bin/bundle exec rake RAILS_ENV=producti
on reports:prune upto=3 unit=mon'  && \
    echo "Optimizing Database" && \
    /usr/bin/su -m puppet-dashboard -c '/usr/local/bin/bundle exec rake RAILS_ENV=producti
on db:raw:optimize'
```

  - And make it executable:

```
chmod 755 /usr/local/etc/periodic/monthly/clean_dashboard_database.sh
```

- And create a weekly script:

```
mkdir -p /usr/local/etc/periodic/weekly
vi /usr/local/etc/periodic/weekly/clean_puppet_reports.sh
```

  - And add the following:

```
#!/bin/sh
echo "Pruning Puppetmaster Reports greater than 7 days old"
echo -n "  Reports Removed:"
find /var/puppet/reports -mtime 7 | xargs rm -v | wc -l
```

  - And make it executable:

```
chmod 755 /usr/local/etc/periodic/weekly/clean_puppet_reports.sh
```

Install Nagios

- Start by installing nagios, php, and fcgi:

```
portmaster net-mgmt/nagios net-mgmt/nagios-plugins net-mgmt/nrpe www/spawn-fcgi www/fcgiwrap n
et/samba-smbclient databases/p5-DBI databases/p5-Class-DBI-mysql
```

- And then enable the service to start at boot:

```
echo 'nagios_enable="YES"' >> /etc/rc.conf
echo 'spawn_fcgi_enable="YES"' >> /etc/rc.conf
echo 'fcgiwrap_enable="YES"' >> /etc/rc.conf
echo 'fcgiwrap_user="www"' >> /etc/rc.conf
echo 'nrpe2_enable="YES"' >> /etc/rc.conf
```

Configure Nagios

- Now copy the sample config files to real config files

```
cd /usr/local/etc/nagios/
cp cgi.cfg-sample cgi.cfg
cp nagios.cfg-sample nagios.cfg
cp resource.cfg-sample resource.cfg

cd /usr/local/etc/nagios/objects/
cp commands.cfg-sample commands.cfg
cp contacts.cfg-sample contacts.cfg
cp localhost.cfg-sample localhost.cfg
cp printer.cfg-sample printer.cfg
cp switch.cfg-sample switch.cfg
cp templates.cfg-sample templates.cfg
cp timeperiods.cfg-sample timeperiods.cfg
```

- Now check you nagios configurations errors

```
nagios -v  /usr/local/etc/nagios/nagios.cfg
```

- Now set a password for the web interface

```
htpasswd -c /usr/local/etc/nagios/htpasswd.users nagiosadmin
```

- Next, create the **nagios server block**:

```
vi /usr/local/etc/nginx/conf.d/nagios.conf
```

    - And add the following

```
server {
  listen 8000 default;
  server nagios.example.com;

  index index.html index.php;
  root /usr/local/www;
```

```
  # IP and IP ranges which should get access
  allow 10.0.0.0/24;
  allow 10.1.0.1;
  # all else will be denied
  deny all;

  # basic HTTP auth
  auth_basic "Restricted";
  auth_basic_user_file htpasswd;

  location ~ \.cgi$ {
    try_files $uri =404;
    include fastcgi_params;
    fastcgi_pass unix:/var/run/fcgiwrap/fcgiwrap.sock;
    fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
    fastcgi_param REMOTE_USER $remote_user;
  }

  location ~ \.php$ {
    try_files $uri =404;
    include fastcgi_params;
    fastcgi_pass 127.0.0.1:9000;
    fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
  }
}
```

NOTE: It is important that the .cgi location has fastcgi_pass set to unix:/var/run/fcgiwrap/fcgiwrap.sock.

- Load the schema

```
cd /usr/local/share/doc/ndoutils
perl ./installdb -u nagios-p SecretNagiosPassword -h localhost -d nagios
```

  - *Example output*

```
DBD::mysql::db do failed: Table 'nagios.nagios_dbversion' doesn't exist at ./installdb lin
e 51.
** Creating tables for version 1.4b9
     Using mysql.sql for installation…
** Updating table nagios_dbversion
Done!
```

- Configure the /usr/local/etc/nagios/nagios.cfg file and change the following parameter:

```
event_broker_options=-1
```

- Add the following in "EVENT BROKER MODULE" section

```
broker_module=/usr/local/bin/ndomod.o config_file=/usr/local/etc/nagios/ndomod.cfg
```

- Configure the /usr/local/etc/nagios/ndo2db.cfg file, and change the below three lines:

```
db_user=nagios
db_pass=SecretNagiosPassword
debug_level=-1
```

- Start ndo2db

```
service ndo2db start
```

- Start Nagios

```
service nagios restart
```

- Start nginx, and check to see if nagios is working by opening a web browser and going to http://example.com/nagios:

```
service nginx start
```

- **NOTE**: I kept getting a permission problem when logging in to the Nagios interface. It turned out I needed to add the correct admin user:

```
vi /usr/local/etc/nagios/cgi.cfg
```

  - Then add/modify the following

```
use_authentication=1
#edit username
authorized_for_all_host_commands=username
authorized_for_all_hosts=username
authorized_for_all_service_commands=username
authorized_for_all_services=username
authorized_for_configuration_information=username
authorized_for_system_commands=username
authorized_for_system_information=username
```

Resources

- http://docs.puppetlabs.com/dashboard/manual/1.2/bootstrapping.html#configuring-dashboard
- http://docs.puppetlabs.com/dashboard/manual/1.2/bootstrapping.html#installing-puppet-dashboard
- http://docs.puppetlabs.com/dashboard/manual/1.2/bootstrapping.html#creating-and-configuring-a-mysql-database
- http://docs.puppetlabs.com/dashboard/manual/1.2/bootstrapping.html#testing-that-dashboard-is-working
  http://docs.puppetlabs.com/dashboard/manual/1.2/bootstrapping.html#configuring-puppet
- http://docs.puppetlabs.com/dashboard/manual/1.2/bootstrapping.html#starting-and-managing-delayed-job-workers
- http://docs.puppetlabs.com/dashboard/manual/1.2/bootstrapping.html#running-dashboard-in-a-production-quality-server
- http://z0mbix.github.io/blog/2012/03/01/use-nginx-and-passenger-to-power-your-puppet-master/
- http://www.watters.ws/mediawiki/index.php/Configure_puppet_master_using_nginx_and_mod_passenger
- http://docs.puppetlabs.com/dashboard/manual/1.2/configuring.html
- https://forums.freebsd.org/viewtopic.php?&t=42071
- http://projects.puppetlabs.com/issues/16686
- http://rlaskey.org/words/897/nagios-nginx-freebsd/
- http://www.unixmen.com/it-appears-as-though-you-do-not-have-permission-to-view-information-for-any-of-the-hosts-you-requested/

History

#2 - 01/25/2015 08:59 PM - Daniel Curtis

- Subject changed from Setting Up a Puppet Master, Puppet Dashboard, PuppetDB, Nagios, Nginx Administration Server to Setting Up a Puppet Master, Puppet Dashboard, Nagios, Nginx Administration Server

- Description updated

#3 - 01/27/2015 12:00 PM - Daniel Curtis

- Description updated

- Status changed from New to In Progress

#4 - 01/27/2015 12:03 PM - Daniel Curtis

- Description updated

#5 - 01/27/2015 12:15 PM - Daniel Curtis

- Description updated

#6 - 01/27/2015 12:17 PM - Daniel Curtis

- Description updated

#7 - 01/27/2015 12:21 PM - Daniel Curtis

- Description updated

#8 - 01/28/2015 10:03 AM - Daniel Curtis

- Project changed from 90 to FreeBSD Administration

#9 - 02/01/2015 07:08 PM - Daniel Curtis

- Description updated

#10 - 02/03/2015 08:28 PM - Daniel Curtis

- Subject changed from Setting Up a Puppet Master, Puppet Dashboard, Nagios, Nginx Administration Server to Setting Up a Puppet Master, Puppet Dashboard and Nagios Administration Server With Nginx on FreeBSD

- Description updated

#11 - 02/09/2015 03:32 PM - Daniel Curtis

- Description updated

#12 - 02/11/2015 11:27 AM - Daniel Curtis

- Description updated

- Status changed from In Progress to Resolved

#13 - 02/11/2015 11:29 AM - Daniel Curtis

- Description updated

#14 - 02/14/2015 10:48 AM - Daniel Curtis

- Target version set to FreeBSD 9

#15 - 02/14/2015 12:08 PM - Daniel Curtis

- Subject changed from Setting Up a Puppet Master, Puppet Dashboard and Nagios Administration Server With Nginx on FreeBSD to Installing Puppet Master, Puppet Dashboard and Nagios with Nginx on FreeBSD

- Category set to Automated Server Management

#16 - 02/17/2015 01:42 PM - Daniel Curtis

- Description updated

#17 - 02/22/2015 05:22 PM - Daniel Curtis

- Description updated

- Status changed from Resolved to Closed

#18 - 03/19/2015 08:26 PM - Daniel Curtis

- Description updated

#19 - 03/19/2015 08:40 PM - Daniel Curtis

- Description updated

#20 - 03/20/2015 04:34 PM - Daniel Curtis

- Subject changed from Installing Puppet Master, Puppet Dashboard and Nagios with Nginx on FreeBSD to Install a Puppet Master, Puppet

Dashboard and Nagios Server with Nginx on FreeBSD

- Description updated

#21 - 03/20/2015 04:45 PM - Daniel Curtis

- Description updated

#22 - 03/20/2015 05:10 PM - Daniel Curtis

- Description updated

#23 - 03/20/2015 08:01 PM - Daniel Curtis

- Description updated

#24 - 03/21/2015 11:50 AM - Daniel Curtis

- Description updated

#25 - 03/21/2015 12:14 PM - Daniel Curtis

- Description updated