FreeBSD Administration - Support #691

Install a Puppet Master, Puppet Dashboard and Icinga2 Server with Nginx on FreeBSD

11/03/2015 12:05 PM - Daniel Curtis

Status:	Closed	**Start date:**	11/03/2015
Priority:	Normal	**Due date:**	
Assignee:	Daniel Curtis	**% Done:**	100%
Category:	Automated Server Management	**Estimated time:**	6.00 hours
Target version:	FreeBSD 9	**Spent time:**	8.00 hours

Description

This is a simple guide to setup a Puppet Master server along with Puppet Dashboard and Icinga2 with PostgreSQL as the database backend 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
- **Icinga** - http://icinga.example.com:8000
- **Puppet Master** - https://puppet.example.com:8140

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:

```
pkg install portmaster
pkg2ng
```

Install PostgreSQL 9.4

- Install PostgreSQL:

```
pkg install postgresql94-{server,client}
```

- Enable PostgreSQL at boot:

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

- Initialize the database:

```
service postgresql initdb
```

- Start PostgreSQL:

```
service postgresql start
```

- Edit the postgres config file:

```
vi /usr/local/pgsql/data/postgresql.conf
```

  - And modify the following:

```
listen_addresses = '*'
```

- Edit the pg_hba config file:

```
vi /usr/local/etc/pgsql/data/pg_hba.conf
```

  - And modify the following:

```
# TYPE  DATABASE    USER        CIDR-ADDRESS          METHOD only

# Local connections
local   all         all                               trust

# IPv4 local connections:
host    all         all         127.0.0.1/32          trust

# IPv6 local connections:
host    all         all         ::1/128               trust

# IPv4 connections:
host    all         all         192.168.10.0/24       md5

# IPv6 local connections:
host    all         all         1234::abcd/64         md5
```

- Restart postgresql:

```
service postgresql restart
```

Create a new user and database

- Switch to the pgsql user and enter into the psql prompt:

```
su pgsql
psql -d template1
```

  - Create the **puppetmaster** user and database:

```
CREATE USER puppetmasteruser WITH PASSWORD 'SuperSecretPuppetmasterPassword';

CREATE DATABASE puppetmasterdb OWNER puppetmasteruser;

GRANT ALL PRIVILEGES ON DATABASE "puppetmasterdb" to puppetmasteruser;
```

  - Create the **dashboard** user and database:

```
CREATE USER dashboarduser WITH PASSWORD 'SuperSecretDashboardPassword';
```

```
CREATE DATABASE dashboarddb OWNER dashboarduser;

GRANT ALL PRIVILEGES ON DATABASE "dashboarddb" to dashboarduser;
```

- Create the **icinga** user and database:

```
CREATE USER icingauser WITH PASSWORD 'SuperSecretIcingaPassword';

CREATE DATABASE icingadb OWNER icingauser;

GRANT ALL PRIVILEGES ON DATABASE "icingadb" to icingauser;
```

- Exit from the postgres user

```
\q
exit
```

- Test the connection on a remote host:

```
psql -h pg.example.com -U puppetmasteruser -W puppetmasterdb
```

Install Puppet Master

- Install the puppet package

```
pkg install puppet rubygem-rake rubygem-bundler rubygem-activerecord rubygem-sqlite3 rubygem-p
g libxslt git node portupgrade
```

- Create a puppet config from the package sample:

```
cp /usr/local/etc/puppet/puppet.conf-dist /usr/local/etc/puppet/puppet.conf
```

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

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

- And add the following to the bottom of the file:

```
[main]
  logdir=/var/log/puppet
  vardir=/var/puppet
  ssldir=/var/puppet/ssl
  rundir=/var/run/puppet
  factpath=$vardir/lib/facter
  server=puppetmaster.example.com
  report=true
  pluginsync=true
  environment = production
  confdir = /usr/local/etc/puppet

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

```
  environment = production

[master]
  storeconfigs = true
  dbadapter = postgresql
  dbuser = puppetmasteruser
  dbpassword = SuperSecretPuppetmasterPassword
  dbserver = localhost
  dbname = puppetmasterdb
```

- Create folders for node and class definition files:

```
mkdir /usr/local/etc/puppet/manifests/nodes
mkdir /usr/local/etc/puppet/manifests/classes
```

- Then create a default site.pp file:

```
import "nodes/*.pp"
vi /usr/local/etc/puppet/manifests/site.pp
```

  - And add the following:

```
node default {
  notify { "Hey ! It works !": }
}

# The filebucket option allows for file backups to the server
filebucket { main: server => 'puppetmaster.altservice.com', path=> false, }

# Set global defaults - including backing up all files to the main filebucket and adds a g
lobal path
File { backup => main }
Exec { path => "/usr/bin:/usr/sbin/:/bin:/sbin" }
```

- Create a puppet node config for **puppetmaster.example.com**:

```
vi /usr/local/etc/puppet/manifests/nodes/puppetmaster.example.com.pp
```

  - And Add the following:

```
node 'puppetmaster.example.com' {
  notify { "Hey ! It works !": }
}
```

- Create a puppet node config for **client.example.com**:

```
vi /usr/local/etc/puppet/manifests/nodes/client.example.com.pp
```

  - And Add the following:

```
node 'client.example.com' {
  notify { "Hey ! It works !": }
}
```

- Enable puppet in /etc/rc.conf:

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

- 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.

```
echo 'puppetmaster_enable="YES"' >> /etc/rc.conf
service puppetmaster start
```

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

```
puppet agent -vt
```

Add Pkgng Module

- Install the pkgng module to add the pkgng package provider:

```
puppet module install zleslie-pkgng
```

  - Now packages can be installed using pkgng as the package manager:

```
package { 'puppet':
  ensure => installed,
  provider => pkgng,
}
```

Enable Puppet File Server

- Make the path for the puppet files to be served from:

```
mkdir /usr/local/etc/puppet/files
```

- Create the puppet fileserver configuration:

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

  - And put the name of your mount point, the path, and an allow * directive.:

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

Next, edit the puppet authorization configuration:

```
vi /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. Add the following before the path / statement:

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

Install Puppet on Remote Node

- Install puppet:

```
pkg install puppet portupgrade
```

- Create a basic puppet config:

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

  - And add the following:

```
[main]
  logdir=/var/log/puppet
  vardir=/var/puppet
  ssldir=/var/puppet/ssl
  rundir=/var/run/puppet
  factpath=$vardir/lib/facter
  configtimeout=600
  runinterval=28800
  report=true
  pluginsync=true
  server=puppetmaster.example.com
```

- Generate a puppet private key and send the CSR to the puppetmaster node for signing:

```
puppet agent -vt --waitforcert 30
```

Sign the Puppet CSR

- On the **puppetmaster.example.com** node run the following to sign the certi

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

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: dashboarddb
  username: dashboarduser
  password: SuperSecretDashboardPassword
  encoding: utf8
  adapter: postgresql
  host: localhost
```

- 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 via bundler:

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

- 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
```

- 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
```

- 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
```

- 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 'env RAILS_ENV=production bundle exec rails server'
```

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.

- Edit the **puppet.conf** on the Puppetmaster node:

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

  - And add the following to the [master] section:

```
[master]
  storeconfigs = true
  dbadapter = postgresql
  dbuser = puppetmasteruser
  dbpassword = SuperSecretPuppetmasterPassword
  dbserver = localhost
  dbname = puppetmasterdb
  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/l
ocal/www/puppet-dashboard/bin/external_node
```

- 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 with Nginx and Passenger

- Install the Passenger gem:

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

NOTE: Make sure to enable NGINX and [X]SYMLINK when running make config

- Install Nginx with Passenger

```
portmaster www/nginx
```

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

- 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;
    passenger_ruby /usr/local/bin/ruby;
    passenger_max_pool_size 15;
    passenger_pool_idle_time 300;

    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;
}
```

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

```
vi /usr/local/etc/nginx/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;
}
```

- 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
```

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

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

- Edit the **auth.conf** file on the puppetmaster

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

  - Add the following:

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

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

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

  - Add the following:

```
filebucket { server => "{ main: server => 'puppetmaster.example.com', path=> false, }",
  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 Icinga2

- Start by installing icinga2:

```
pkg install icinga2
```

NOTE: Make sure to uncheck **[ ]MYSQL** during the icinga2 package configuration.

- Enable the icinga2 ido-pgsql feature:

```
ln -s /usr/local/etc/icinga2/features-available/ido-pgsql.conf /usr/local/etc/icinga2/features
-enabled/
```

- Edit the icinga2 ido-pgsql config:

```
vi /usr/local/etc/icinga2/features-enabled/ido-pgsql.conf
```

  - And modify the following:

```
object IdoPgsqlConnection "ido-pgsql" {
  user = "icingauser"
  password = "SuperSecretIcingaPassword"
  host = "localhost"
  database = "icingadb"
}
```

- Load the icinga2 ido-pgsql schema

```
psql -h localhost -U icingauser -W -d icingadb < /usr/local/share/icinga2-ido-pgsql/schema/pgs
ql.sql
```

- Start and enable the service to start at boot:

```
echo 'icinga2_enable="YES"' >> /etc/rc.conf
service icinga2 start
```

Install Icingaweb2

- Install icingaweb2:

```
pkg install icingaweb2 php56-pgsql pecl-imagick
```

- Configure the default PHP settings

```
cp /usr/local/etc/php.ini-production /usr/local/etc/php.ini
```

- Set the local timezone in the php config:

```
sed -i '' -e 's/;date.timezone\ =/date.timezone\ =\ America\/Los_Angeles/' /usr/local/etc/php.ini
```

- Edit /usr/local/etc/php-fpm.conf:

```
vi /usr/local/etc/php-fpm.conf
```

  - Make the following changes:

```
listen = /var/run/php-fpm.sock
listen.owner = www
listen.group = www
listen.mode = 0660
```

- Start and enable PHP-FPM at boot:

```
echo 'php_fpm_enable="YES"' >> /etc/rc.conf
service php-fpm start
```

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

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

  - And add the following

```
server {
  listen 8000 default;
  server_name icinga.example.com;

  index index.html index.php;
  root /usr/local/www/icingaweb2/public;

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

  location ~ \.php$ {
    root /usr/local/www/icingaweb2/public
    fastcgi_pass unix:/var/run/php-fpm.sock;
    fastcgi_index index.php;
    include fastcgi_params;
    fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
    fastcgi_param ICINGAWEB_CONFIGDIR /usr/local/etc/icingaweb2;
    fastcgi_param REMOTE_USER $remote_user;
  }

  location / {
    root /usr/local/www/icingaweb2/public;
    autoindex on;
    index index.php;
    try_files $1 $uri $uri/ /icingaweb2/index.php$is_args$args;
  }
}
```

- Restart nginx:

```
service nginx restart
```

- Now create a configuration token:

```
cd /usr/local/www/icingaweb2 && ./bin/icingacli setup token create --config=/usr/local/etc/ici
ngaweb2
```

- Enter the token created earlier on Icinga Web 2's setup interface at http://icinga.example.com:8000/setup then complete the setup process.

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://docs.puppetlabs.com/puppet/latest/reference/config_important_settings.html
- 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

#1 - 11/03/2015 06:15 PM - Daniel Curtis

- Description updated

#2 - 11/04/2015 03:32 PM - Daniel Curtis

- Description updated

- Status changed from New to In Progress

- % Done changed from 0 to 50

#3 - 11/04/2015 03:52 PM - Daniel Curtis

- Description updated

#4 - 11/05/2015 10:18 AM - Daniel Curtis

- Description updated

- Status changed from In Progress to Resolved

- % Done changed from 50 to 100

#5 - 11/11/2015 04:49 PM - Daniel Curtis

- Description updated

#6 - 11/27/2015 03:43 PM - Daniel Curtis

- Status changed from Resolved to Closed