Open Source Backup Conference

Monitoring Bareos with Icinga 2

Version: 1.0

We love Open Source

Table of Contents

1 Environment

Pre-installed Software

- Bareos
- Bareos Database (PostgreSQL)
- Bareos WebUI
- Icinga 2
- IDO (MariaDB)
- Icinga Web 2
- Graphite

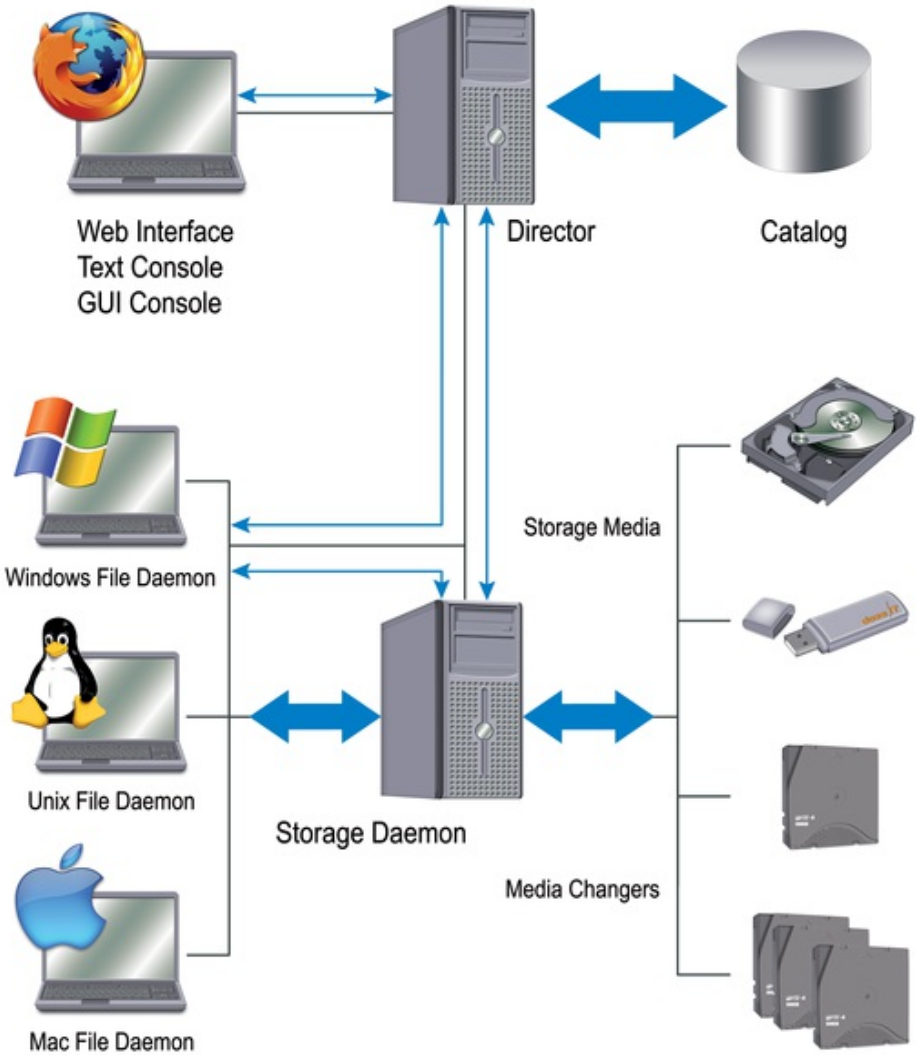
2 Introduction

2.1 Bareos

What is Bareos?

- **B**ackup **A**rchiving **Re**covery **O**pen **S**ourced
- Backup, archiving and recovery of current operating systems
- Open Source Fork of Bacula (http://bacula.org)
- Forked 2010 (http://bareos.org)
- AGPL v3 License (https://github.com/bareos/bareos)
- A lot of new features:
    - LTO Hardware encryption
    - Bandwidth limitation
    - Cloud storage connection
    - New console commands
    - Many more

Bareos Structure



2.2 Icinga 2

Icinga – Open Source Enterprise Monitoring

Icinga is a **scalable** and **extensible** monitoring system which checks the **availability** of your resources, notifies users of outages and provides extensive **BI** data.

- International community project
- Everything developed by the Icinga Project is Open Source
- Originally forked from Nagios in 2009
- Independent version Icinga 2 since 2014

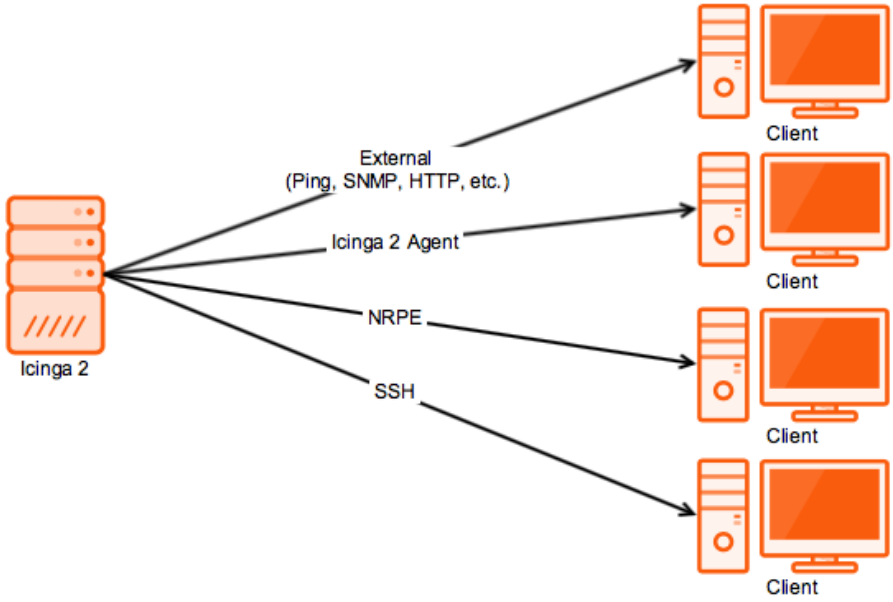
Icinga – Availability Monitoring

- Monitors everything
- Gathering status
- Collect performance data
- Notifies using any channel
- Considers dependencies
- Handles events
- Checks and forwards logs
- Deals with performance data
- Provides SLA data

What is Icinga 2?

- Core based on C++ and Boost
    - Supports all major *NIX and Windows platforms
- Powerful configuration language
- Supports MySQL/MariaDB and PostgreSQL
- Includes a extensive template library
- Logstash and Graylog for logs
- Graphite, OpenTSDB or InfluxDB for performance data
- Puppet, Chef and Ansible support
- Distributed Monitoring
- High-Availability Clustering
- HTTP RESTful API (since 2.4)

Icinga 2 Check Methods

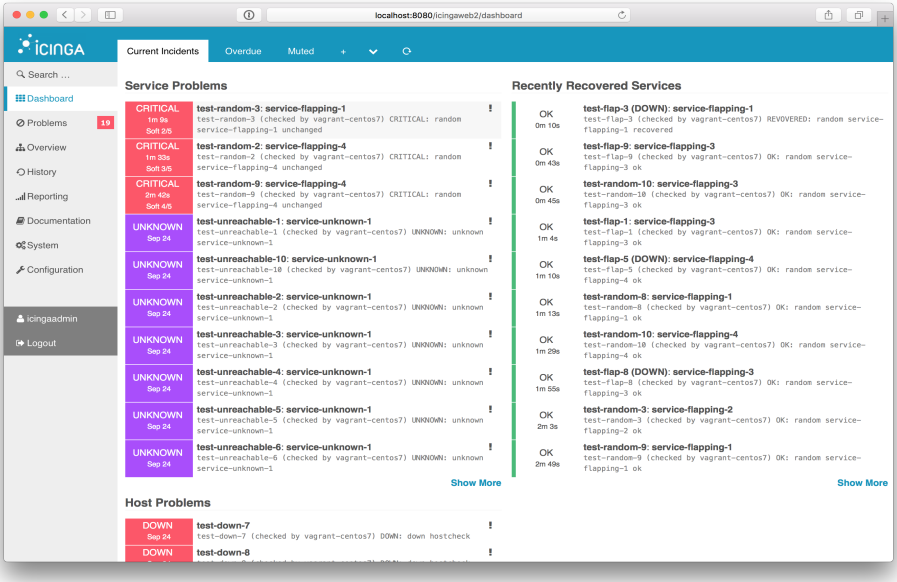


2.3 Icinga Web 2

What is Icinga Web 2?

- Developed from scratch using
  - PHP 5.3
  - Zend Framework 1 and jQuery
- Easy to extend and embed into other projects
- Simple INI configuration
- Small footprint, fast and responsive
- Supports MySQL/MariaDB and PostgreSQL as IDO backend
- Basic API and CLI
- PDF, JSON and CSV export
- Multiple authentication methods and URL filters
- Role based access control (RBAC) and one click actions

Icinga Web 2 – Dashboard



3 Host

Icinga 2 Host Object Configuration

Host object:

```
object Host "bareos.localdomain" {
  address = "192.168.56.101"
  address6 = "fe80::a00:27ff:fedd:f44c"

  max_check_attempts = 3
  check_interval = 1m
  retry_interval = 30s

  vars.os = "Linux"
  vars.role = "Bareos"

  check_command = "hostalive"
}

# icinga2 daemon -C

# systemctl reload icinga2.service
# icinga2 object list --type Host --name bareos.localdomain
```

Icinga 2 Host Object Configuration with Template

Host template:

```
template Service "generic-host" {
  max_check_attempts = 3
  check_interval = 1m
  retry_interval = 30s
}
```

Host object:

```
object Host "bareos.localdomain" {
  import "generic-host"

  address = "192.168.56.101"
  address6 = "fe80::a00:27ff:fedd:f44c"

  vars.os = "Linux"
  vars.role = "Bareos"

  check_command = "hostalive"
}
```

4 Active Checks

4.1 Basic

Basic Checks

- Hardware (Thomas Krenn IPMI, Dell Open Manage, etc.)
- Rechability (Ping, SSH, etc.)
- System (Users, Time, DNS, Updates, etc.)
- Disk Usage
- CPU Usage
- Memory Usage

Reachability

Service apply rule:

```
apply Service "ping4" {
  max_check_attempts = 5
  check_interval = 1m
  retry_interval = 30s

  check_command = "ping4"

  assign where host.address
}

# systemctl reload icinga2.service
# icinga2 object list --type Service --name ping4
```

- Assign **ping4** Service to all Hosts with IPv4 address dynamically

Reachability

Service template:

```
template Service "generic-service" {
  max_check_attempts = 5
  check_interval = 1m
  retry_interval = 30s
}
```

Service apply rule:

```
apply Service "ping4" {
  import "generic-service"

  check_command = "ping4"

  assign where host.address
}
```

- Assign **ping4** Service to all Hosts with IPv4 address dynamically

System Users

```
apply Service "users" {
  import "generic-service"

  check_command = "users"
  command_endpoint = host.name

  assign where host.vars.os == "Linux"
}

# systemctl reload icinga2.service
# icinga2 object list --type Service --name users
```

Disk Usage

```
apply Service "disk" {
  import "generic-service"

  check_command = "disk"
  command_endpoint = host.name

  vars.disk_local = true

  assign where host.vars.os == "Linux"
}

# systemctl reload icinga2.service
# icinga2 object list --type Service --name disk
```

CPU Usage

```
apply Service "load" {
  import "generic-service"

  check_command = "load"
  command_endpoint = host.name

  vars.load_wload1 = 5.0
  vars.load_wload5 = 4.0
  vars.load-wload15 = 3.0

  vars.load_cload1 = 10.0
  vars.load_cload5 = 6.0
  vars.load_cload15 = 4.0

  assign where host.vars.os == "Linux"
}

# systemctl reload icinga2.service
# icinga2 object list --type Service --name load
```

4.2 Services

Services

- Bareos
  - Director
  - File Daemon
  - Storage Daemon
- Database
  - PostgreSQL
  - MySQL/MariaDB
- Web
  - Apache

Bareos Services

Director:

- Process: **bareos-dir**
- Port: **9101/TCP**

File Daemon:

- Process: **bareos-fd**
- Port: **9102/TCP**

Storage Daemon:

- Process: **bareos-sd**
- Port: **9103/TCP**

Database Services

PostgreSQL:

- Process: **postgresql**
- Port: **5432/TCP**

MySQL/MariaDB:

- Process: **mysqld/mariadb**
- Port: **3306/TCP**

Web Services

Apache:

- Process: **apache2/httpd**
- Port: **80/TCP, 443/TCP**

Plugins for Service Checks

Standard Plugins (Monitoring Plugins)

- check_procs – Process
- check_tcp – Port

No functionality testing, only availability!

check_procs

check_procs – Examples

```
# ./check_procs -C bareos-sd -c 1:1 -p 1 -u bareos
PROCS OK: 1 process with command name 'bareos-sd', PPID = 1
UID = 996 (bareos) | procs=1;;1:1;0;

# ./check_procs -C bareos-dir -c 1:1 -p 1
PROCS CRITICAL: 0 processes with command name 'bareos-dir',
PPID = 1 | procs=0;;1:1;0;

# ./check_procs -C bareos-fd -c 1:1 -p 1 \
-a /etc/bareos/bareos-fd.conf
PROCS OK: 1 process with command name 'bareos-fd', PPID = 1
args '/etc/bareos/bareos-fd.conf' | procs=1;;1:1;0;

# ./check_procs -C postgres -c 1:1 -p 1 -u postgres
PROCS OK: 1 process with command name 'postgres', PPID = 1,
UID = 26 (postgres) | procs=1;;1:1;0;

# ./check_procs -C httpd -w 8:10 -c 1:30 -a /usr/sbin/httpd
PROCS WARNING: 6 processes with command name 'httpd',
args '/usr/sbin/httpd' | procs=6;8:10;1:30;0;
```

check_procs – Service Apply Rule Example

```
apply Service "bareos-sd-process" {
  import "generic-service"

  check_command = "procs"
  command_endpoint = host.name

  vars.procs_command  = "bareos-sd"
  vars.procs_critical = "1:1"
  vars.procs_ppid     = "1"
  vars.procs_user     = "bareos"

  assign where host.vars.role == "Bareos"
}

# systemctl reload icinga2.service
# icinga2 object list --type Service --name bareos-sd-proce
```

http://docs.icinga.org/icinga2/latest/doc/module/icinga2/chapter/plugin-check-commands?highlight-search=check_procs#plugin-check-command-processes

check_tcp

check_tcp – Examples

```
# ./check_tcp -p 9101
TCP OK - 0,000 second response time on port 9101|
time=0,000092s;;;0,000000;10,000000

# ./check_tcp -p 9103
connect to address 127.0.0.1 and port 9103:
Connection refused

# ./check_tcp -p 5432
TCP OK - 0,000 second response time on port 5432|
time=0,000127s;;;0,000000;10,000000
```

check_tcp – Service Apply Rule Example

```
apply Service "bareos-dir-port" {
  import "generic-service"

  check_command = "tcp"
  command_endpoint = host.name

  vars.tcp_port = "9101"

  assign where host.vars.role == "Bareos"
}

# systemctl reload icinga2.service
# icinga2 object list --type Service --name bareos-dir-port
```

http://docs.icinga.org/icinga2/latest/doc/module/icinga2/chapter/plugin-check-commands?highlight-search=check_tcp#plugin-check-command-tcp

4.3 Functionality

check_bareos

check_bareos – Plugin

Available from GitHub (https://github.com/widhalmt/check_bareos) with better support for MySQL/MariaDB

Available from GitHub (https://github.com/theGidy/check_bareos) or Icinga Exchange (https://exchange.icinga.org/hrstldlr/check_bareos)

check_bareos – Arguments

- Job
  - State
  - Runtime
- Status
  - Backup size (total, empty, oversized, failed)
- Tape
  - Number
  - State (empty, expire, replace)

check_bareos – Installation

```
# yum install python-psycopg2 MySQL-python
# git clone https://github.com/widhalmt/check_bareos.git
# cp check_bareos/check_bareos.py /usr/lib64/nagios/plugins
# /usr/lib64/nagios/plugins/check_bareos.py --help
```

Copy Plugin Check Command:

```
# cp check_bareos/contrib/icinga2-commands.conf \
/etc/icinga2/conf.d/bareos.conf
```

check_bareos – Examples

```
# ./check_bareos.py -u bareos -p bareos -d p status -fb -c
OK - Only 0 Backups failed in the last 1 days|Failed=0;5;1;

# ./check_bareos.py -u bareos -p bareos -d p status -e -w 1
WARNING - 1 successful 'F','D','I' backups are empty!|
EmptyBackups=1;1;3;;

# ./check_bareos.py -u bareos -p bareos -d p job -js -st E
OK - 0.0 Jobs are in the state: Terminated with Errors|
Terminated with Errors=0.0;5;10;;

# ./check_bareos.py -u bareos -p bareos -d p tape -ex
CRITICAL - Only 0.0 expired|Expired=0.0;5;10;;
```

check_bareos – Service Apply Rule Example (1/3)

```
apply Service "bareos-job" {
  import "generic-service"

  check_command = "bareos-job"
  command_endpoint = host.name

  vars.bareos_runtimejobs = true
  vars.bareos_time = "4"
  vars.bareos_state = "R"
  vars.bareos_warning = "1"
  vars.bareos_critial = "4"

  assign where host.vars.role == "Bareos"
}

# systemctl reload icinga2.service
# icinga2 object list --type Service --name bareos-job
```

check_bareos – Service Apply Rule Example (2/3)

```
apply Service "bareos-status" {
  import "generic-service"

  check_command = "bareos-status"
  command_endpoint = host.name

  vars.bareos_totalbackupssize = true
  vars.bareos_diff = true
  vars.bareos_warning = "400"
  vars.bareos_critical = "500"

  assign where host.vars.role == "Bareos"
}

# systemctl reload icinga2.service
# icinga2 object list --type Service --name bareos-status
```

check_bareos – Service Apply Rule Example (3/3)

```
apply Service "bareos-tape" {
  import "generic-service"

  check_command = "bareos-tape"
  command_endpoint = host.name

  vars.bareos_emptytapes = true
  vars.bareos_warning = "15"
  vars.bareos_critical = "10"

  assign where host.vars.role == "Bareos"
}

# systemctl reload icinga2.service
# icinga2 object list --type Service --name bareos-tape
```

4.4 Databases

Database Backends

- PostgreSQL
- MySQL/MariaDB
- Sqlite (Testing)

PostgreSQL

check_postgres – Plugin

- Available from: https://bucardo.org/wiki/Check_postgres or GitHub (https://github.com/bucardo/check_postgres)
- Written in Perl
  - Uses psql binary
- Privileges:
  - Select on public schema
  - Select on data for query checks
  - Superuser for some actions
  - Local execution for some other

check_postgres – Actions

- connection
- custom_query
- database_size
- hot_standby_delay
- last_vacuum
- locks
- query_time

And many more...

check_postgres – Installation

```
# yum install perl-Data-Dumper
# wget http://bucardo.org/downloads/check_postgres-2.22.0.t
# tar -xf check_postgres-2.22.0.tar.gz
# cp check_postgres-2.22.0/check_postgres.pl \
/usr/lib64/nagios/plugins/
# /usr/lib64/nagios/plugins/check_postgres.pl --help
```

check_postgres – Examples

```
# ./check_postgres.pl -H 127.0.0.1 --dbname=bareos \
--dbuser=bareos --dbpass=bareos --action=connection
POSTGRES_CONNECTION OK: DB "bareos" (host:127.0.0.1)
version 9.2.15 | time=0.01s

# ./check_postgres.pl -H 127.0.0.1 --dbname=bareos \
--dbuser=bareos --dbpass=bareos --action=backends -w 10 -c
POSTGRES_BACKENDS OK: DB "bareos" (host:127.0.0.1)
2 of 100 connections (2%) | time=0.01s bareos=2;10;50;0;100
postgres=0;10;50;0;100 template0=0;10;50;0;100
template1=0;10;50;0;100

# ./check_postgres.pl -H 127.0.0.1 --dbname=bareos \
--dbuser=bareos --dbpass=bareos --action=database_size \
-w 50MB -c 100MB
POSTGRES_DATABASE_SIZE OK: DB "bareos" (host:127.0.0.1)
bareos: 7959672 (7773 kB) postgres: 6648952 (6493 kB)
template1: 6648952 (6493 kB) template0: 6529540 (6377 kB) |
time=0.01s bareos=7959672;52428800;104857600
postgres=6648952;52428800;104857600 template1=6648952;52428
104857600 template0=6529540;52428800;104857600
```

check_postgres – Service Apply Rule Example

```
apply Service "postgres-connection" {
  import "generic-service"

  check_command = "postgres"
  command_endpoint = host.name

  vars.postgres_host = "127.0.0.1"
  vars.postgres_dbname = "bareos"
  vars.postgres_dbuser = "bareos"
  vars.postgres_dbpass = "bareos"
  vars.postgres_action = "connection"

  assign where host.vars.role == "Bareos"
}

# systemctl reload icinga2.service
# icinga2 object list --type Service --name postgres-connec
```

http://docs.icinga.org/icinga2/latest/doc/module/icinga2/chapter/plugin-check-commands?highlight-search=check_postgres#plugins-contrib-command-postgres

MySQL/MariaDB

Plugins for MySQL/MariaDB

- check_mysql
- check_mysql_health
- Percona Monitoring Plugins

check_mysql – Plugin

- Available from: https://www.monitoring-plugins.org/doc/man/check_mysql.html or GitHub (https://github.com/monitoring-plugins/monitoring-plugins)
- Standard Plugin (Monitoring Plugin)
- DEB or RPM packages
- Written in C
- GNU General Public License (GPL)

check_mysql – Example

```
# ./check_mysql -d bareos -u bareos -p bareos
Uptime: 503  Threads: 2  Questions: 4654  Slow queries: 0
Opens: 54  Flush tables: 2  Open tables: 80  Queries per
second avg: 9.252|Connections=207c;;; Open_files=21;;;
Open_tables=80;;; Qcache_free_memory=0;;; Qcache_hits=0c;;;
Qcache_inserts=0c;;; Qcache_lowmem_prunes=0c;;;
Qcache_not_cached=0c;;; Qcache_queries_in_cache=0;;;
Queries=4655c;;; Questions=4654c;;; Table_locks_waited=0c;;
Threads_connected=2;;; Threads_running=1;;; Uptime=503c;;;
```

check_mysql – Service Apply Rule Example

```
apply Service "mysql" {
  import "generic-service"

  check_command = "mysql"
  command_endpoint = host.name

  vars.mysql_database = "bareos"
  vars.mysql_username = "bareos"
  vars.mysql_password = "bareos"

  assign where host.vars.role == "Bareos"
}

# systemctl reload icinga2.service
# icinga2 object list --type Service --name mysql
```

http://docs.icinga.org/icinga2/latest/doc/module/icinga2/chapter/plugin-check-commands?highlight-search=check_mysql#plugin-check-command-mysql

check_mysql_health – Plugin

- Available from: https://labs.consol.de/de/nagios/check_mysql_health/inde or GitHub (https://github.com/lausser/check_mysql_health)
- Written in Perl by Gerhard Laußer from ConSol
  - Uses DBI and DBD::mysql
- GNU General Public License (GPL)
- Privileges:
  - Usage for most modes
  - Replication client for all slave modes
  - Select for `sql` mode

check_mysql_health - Modes

- connection-time
- uptime
- threads-connected
- q(uery)cache-hitrate
- tablecache-hitrate
- long-running-procs
- index-usage
- slave-sql-running
- sql

And many more...

check_mysql_health – Installation

```
# yum install perl-Digest-MD5
# wget https://labs.consol.de/assets/downloads/nagios/ \
check_mysql_health-3.0.0.5.tar.gz
# tar -xf check_mysql_health-3.0.0.5.tar.gz
# cd check_mysql_health-3.0.0.5/
# ./configure --with-nagios-user=icinga \
--with-nagios-group=icinga --libexecdir=/usr/lib64/nagios/p
# make
# make install
# /usr/lib64/nagios/plugins/check_mysql_health --help
```

check_mysql_health – Examples

```
# ./check_mysql_health --username=bareos --password=bareos
--database=bareos --mode=connection-time
OK - 0.07 seconds to connect as bareos |
'connection_time'=0.07;1;5;;

# ./check_mysql_health --username=bareos --password=bareos
--database=bareos --mode=threads-connected --warning 5 \
--critical 10
OK - 2 client connection threads | 'threads_connected'=2;5;

# ./check_mysql_health --username=bareos --password=bareos
--database=bareos --mode=tablecache-hitrate
OK - table cache hitrate 148.15%, 20.00% filled |
'tablecache_hitrate'=148.15%;99:;95:;0;100
'tablecache_fillrate'=20%;;;0;100
```

check_mysql_health – Service Apply Rule Example

```
apply Service "mysql-connection-time" {
  import "generic-service"

  check_command = "mysql_health"
  command_endpoint = host.name

  vars.mysql_health_username = "bareos"
  vars.mysql_health_password = "bareos"
  vars.mysql_health_database = "bareos"
  vars.mysql_health_mode = "connection-time"

  assign where host.vars.role == "Bareos"
}

# systemctl reload icinga2.service
# icinga2 object list --type Service --name mysql-connectio
```

http://docs.icinga.org/icinga2/latest/doc/module/icinga2/chapter/plugin-check-commands?highlight-search=mysql_health#plugins-contrib-command-mysql_health

Percona Monitoring Plugins

- Available from: https://www.percona.com/software/mysql-tools/percona-monitoring-plugins or GitHub (https://github.com/percona/percona-monitoring-plugins)

Percona Monitoring Plugins – Installation

```
# rpm -Uhv https://www.percona.com/downloads/\
percona-monitoring-plugins/1.1.6/\
percona-nagios-plugins-1.1.6-1.noarch.rpm
# rpm -ql percona-nagios-plugins
/usr/lib64/nagios/plugins/pmp-check-aws-rds.py
/usr/lib64/nagios/plugins/pmp-check-lvm-snapshots
/usr/lib64/nagios/plugins/pmp-check-mysql-deadlocks
/usr/lib64/nagios/plugins/pmp-check-mysql-deleted-files
/usr/lib64/nagios/plugins/pmp-check-mysql-file-privs
/usr/lib64/nagios/plugins/pmp-check-mysql-innodb
/usr/lib64/nagios/plugins/pmp-check-mysql-pidfile
/usr/lib64/nagios/plugins/pmp-check-mysql-processlist
/usr/lib64/nagios/plugins/pmp-check-mysql-replication-delay
/usr/lib64/nagios/plugins/pmp-check-mysql-replication-runni
/usr/lib64/nagios/plugins/pmp-check-mysql-status
/usr/lib64/nagios/plugins/pmp-check-mysql-ts-count
/usr/lib64/nagios/plugins/pmp-check-pt-table-checksum
/usr/lib64/nagios/plugins/pmp-check-unix-memory
```

Percona Monitoring Plugins – Examples

```
# ./pmp-check-mysql-innodb
OK longest blocking idle transaction sleeps for 0 second

# ./pmp-check-mysql-deleted-files
OK no deleted files

# ./pmp-check-mysql-processlist
OK 0 unauthenticated, 0 locked, 0 copy to table, 0 statisti
processes=0;16;32;0;

# ./pmp-check-unix-memory -c 95 -w 90
OK Memory 81% used | memory_used=81;90;95;0;100
```

4.5 WebUI

WebUI

- Plugins
  - check_http
- End to End
  - Webinject (http://www.webinject.org)
  - CasperJS (http://casperjs.org)
  - Selenium (http://www.seleniumhq.org)

check_http

check_http – Examples

```
# ./check_http -H 192.168.56.101 -u /bareos-webui -f follow
HTTP OK: HTTP/1.1 200 OK - 3688 bytes in 0.122 second respo
time |time=0.122051s;;;0.000000 size=3688B;;;

# ./check_http -H 192.168.56.101 -u /bareos-webui -f follow
-r bareos.png
HTTP OK: HTTP/1.1 200 OK - 3688 bytes in 0.118 second respo
time |time=0.118181s;;;0.000000 size=3688B;;;0

# ./check_http -H 192.168.56.101 -u /bareos-webui -f follow
-r bareos2.png
HTTP CRITICAL: HTTP/1.1 200 OK - pattern not found - 3688 b
in 0.139 second response time |time=0.138861s;;;0.000000
size=3688B;;;0
```

check_http – Service Apply Rule Example

```
apply Service "bareos-webui" {
  import "generic-service"

  check_command = "http"

  vars.http_uri = "/bareos-webui"
  vars.http_onredirect = "follow"
  vars.http_expect_body_regex = "bareos.png"

  assign where host.vars.role == "Bareos"
}

# systemctl reload icinga2.service
# icinga2 object list --type Service --name bareos-webui
```

WebInject

WebInject & Plugin

- Available from: http://www.webinject.org or GitHub (https://github.com/sni/Webinject)
- Free free tool for automated testing of web applications and web services

Plugin "*check_webinject*":

- Available from: https://labs.consol.de/de/nagios/check_webinject/index.ht

WebInject – Installation

```
# wget https://labs.consol.de/assets/downloads/\
Webinject-1.84.tar.gz
# tar -xf Webinject-1.84.tar.gz
# yum install perl-ExtUtils-MakeMaker perl-CPAN perl-XML-Si
perl-Crypt-SSLeay perl-Test-Simple
# perl Makefile.PL
# make
# make test
# make install
# make clean
# /usr/local/bin/webinject.pl --help

# wget https://labs.consol.de/assets/downloads/\
check_webinject-1.84.tar.gz
# tar -xf check_webinject-1.84.tar.gz
# cp check_webinject /usr/lib64/nagios/plugins/
```

WebInject – Config file

```
# vim /usr/lib64/nagios/plugins/config.xml
<testcasefile>testcasefile.xml</testcasefile>
<useragent>Webinject Application Tester</useragent>
<globaltimeout>30</globaltimeout>
<reporttype>nagios</reporttype>
```

WebInject – Testcase file (1/2)

```
# vim /usr/lib64/nagios/plugins/testcasefile.xml
<testcases repeat="1">

<case
    id="1"
    url="http://192.168.56.101/bareos-webui/"
    verifyresponsecode="200"
    verifypositive="Bareos - Login"
    errormessage="Webinterface down"
/>

<case
    id="2"
    method="post"
    url="http://192.168.56.101/bareos-webui/auth/login"
    postbody="director=localhost-dir&consolename=bareos\
    &password=bareos&submit=Login"
    verifyresponsecode="302"
    errormessage="Login failed"
/>
```

WebInject – Testcase file (2/2)

```
<case
    id="3"
    url="http://192.168.56.101/bareos-webui/director/"
    verifyresponsecode="200"
    verifypositive="Daemon started"
    errormessage="Login failed"
/>

<case
    id="4"
    url="http://192.168.56.101/bareos-webui/auth/logout"
    verifyresponsecode="302"
    errormessage="Logout failed"
/>

</testcases>
```

check_webinject – Example

```
# ./check_webinject -c /usr/lib64/nagios/plugins/config.xml
WebInject OK - All tests passed successfully in 0.636 secon
time=0.636s;0;30;0;0 case1=0.165s case2=0.092s;0;0;0;0 \
case3=0.232s;0;0;0;0 case4=0.08s;0;0;0;0
```

check_webinject – Service Apply Rule Example

```
apply Service "bareos-webui-login" {
  import "generic-service"

  check_command = "webinject"

  vars.webinject_config_file = "/usr/lib64/nagios/plugins/\
  config.xml"

  assign where host.vars.role == "Bareos"
}

# systemctl reload icinga2.service
# icinga2 object list --type Service --name bareos-webui-lo
```

5 Passive Events

Run Script

Passive Events for Icinga 2 via "**Run Script** Dir Job" resource.
Shortcuts:

- Run Before Job
- Run After Job
- Run After Failed Job
- Client Run Before Job
- Client Run After Job

And many other combinations...

---

http://doc.bareos.org/master/html/bareos-manual-main-reference.html#directiveDirJobRun%20Script

Icinga 2 Service Apply Rule for Passive Events

```
apply Service "backup" {
  check_command = "passive"

  assign where host.vars.os == "Linux"
}

# systemctl reload icinga2.service
# icinga2 object list --type Service --name backup
```

5.1 NSCA

Nagios Service Check Acceptor (NSCA)

- Old fashioned (Latest release Jan 2012)
- Additional daemon on Icinga server
- Uses local commandpipe for command execution
- Client use unencrypted transport of check results

https://github.com/NagiosEnterprises/nsca

5.2 NSCA-ng

Nagios Service Check Acceptor – next generation (NSCA-ng)

- More active development than NSCA (Latest release Oct 2014)
- No compatibility between NSCA clients and NSCA-ng server (or vice versa)
- Improvements:
  - Security
  - Performance
  - Extensibility
  - Portability

---

http://www.nsca-ng.org
https://github.com/weiss/nsca-ng

5.3 Icinga 2 API

Icinga 2 API

- Since Icinga 2.4
- Extends internal API to provide HTTP requests to:
    - Query, create, modify and delete config objects
    - Perform actions (reschedule checks, etc.)
    - Subscribe to event streams
    - Manage configuration packages
    - Evaluate script expressions
- Secured by:
    - Transport Layer Security
    - Password or x.509 client certificate
    - Fine granular permissions
    - Accept header

Use Icinga 2 API

Enable Icinga 2 API:

```
# icinga2 node wizard
Please specify if this is a satellite setup \
  ('n' installs a master setup) [Y/n]: n
Starting the Master setup routine...
Please specifiy the common name (CN) [icinga2.localdomain]:
...

# systemctl restart icinga2.service
```

Create API User:

```
object ApiUser "bareos" {
  password = "bareos"
  permissions = [ "*" ]
}

# systemctl reload icinga2.service
```

Set Downtime Before Job via Icinga 2 API

File */etc/bareos/bareos-dir.conf:*

```
Job {
...
  Run Script {
    Command = "/usr/lib/bareos/scripts/schedule-downtime.sh
    Runs When = Before
    Runs On Client = No
  }
}

# systemctl restart bareos-dir.service
```

Set Downtime Before Job via Icinga 2 API

File */usr/lib/bareos/scripts/schedule-downtime.sh:*

```
#!/bin/bash

starttime=$(date +%s)
endtime=$(date "+1 hour" +%s)

read -r -d '' BODY <<EOF
{ "start_time": $starttime, "end_time": $endtime, "duration
  "3600", "fixed": false, "author": "bareos", "comment":
  "Scheduled downtime for Backup" }
EOF

curl -k -s -u bareos:bareos -H 'Accept: application/json' -
  "https://icinga2.localdomain:5665/v1/actions/schedule-dow
  ?host=$1" -d "BODY"

# chmod +x /usr/lib/bareos/scripts/schedule-downtime.sh
```

http://docs.icinga.org/icinga2/latest/doc/module/icinga2/chapter/icinga2-api#icinga2-api-actions-schedule-downtime

Remove Downtime After Job via Icinga 2 API

File */etc/bareos/bareos-dir.conf:*

```
Job {
...
  Run Script {
    Command = "/usr/lib/bareos/scripts/remove-downtime.sh \
    Runs When = After
    Runs On Client = No
  }
}

# systemctl restart bareos-dir.service
```

Remove Downtime After Job via Icinga 2 API

File */usr/lib/bareos/scripts/remove-downtime.sh:*

```
#!/bin/bash

curl -k -s -u bareos:bareos -H 'Accept: application/json' -
  "https://icinga2.localdomain:5665/v1/actions/remove-downt
  ?host=$1"

# chmod +x /usr/lib/bareos/scripts/remove-downtime.sh
```

http://docs.icinga.org/icinga2/latest/doc/module/icinga2/chapter/icinga2-api#icinga2-api-actions-remove-downtime

Change Service State After Failed Job via Icinga 2 API

```
apply Service "backup" {
  check_command = "passive"

  assign where host.vars.os == "Linux"
}

# systemctl reload icinga2.service
# icinga2 object list --type Service --name backup
```

Change Service State After Failed Job via Icinga 2 API

File */etc/bareos/bareos-dir.conf:*

```
Job {
...
  Run Script {
    Command = "/usr/lib/bareos/scripts/process-check-result
      \%c \%i \%j"
    Runs When = After
    Runs On Failure = Yes
    Runs On Success = No
    Runs On Client = No
  }
}

# systemctl restart bareos-dir.service
```

Change Service State After Failed Job via Icinga 2 API

File */usr/lib/bareos/scripts/process-check-result.sh:*

```
#!/bin/bash

read -r -d '' BODY <<EOF
{ "exit_status": "2", "plugin_output": "Job $2 $3 failed"}
EOF

curl -k -s -u bareos:bareos -H 'Accept: application/json' -
  "https://icinga2.localdomain:5665/v1/actions/process-
  check-result?service=$1!backup" -d "BODY"

# chmod +x /usr/lib/bareos/scripts/process-check-result.sh
```

Example output: **Job 22 bareos.localdomain.2016-08-19_15.30.00_05 failed**

---

http://docs.icinga.org/icinga2/latest/doc/module/icinga2/chapter/icinga2-api#icinga2-api-actions-process-check-result

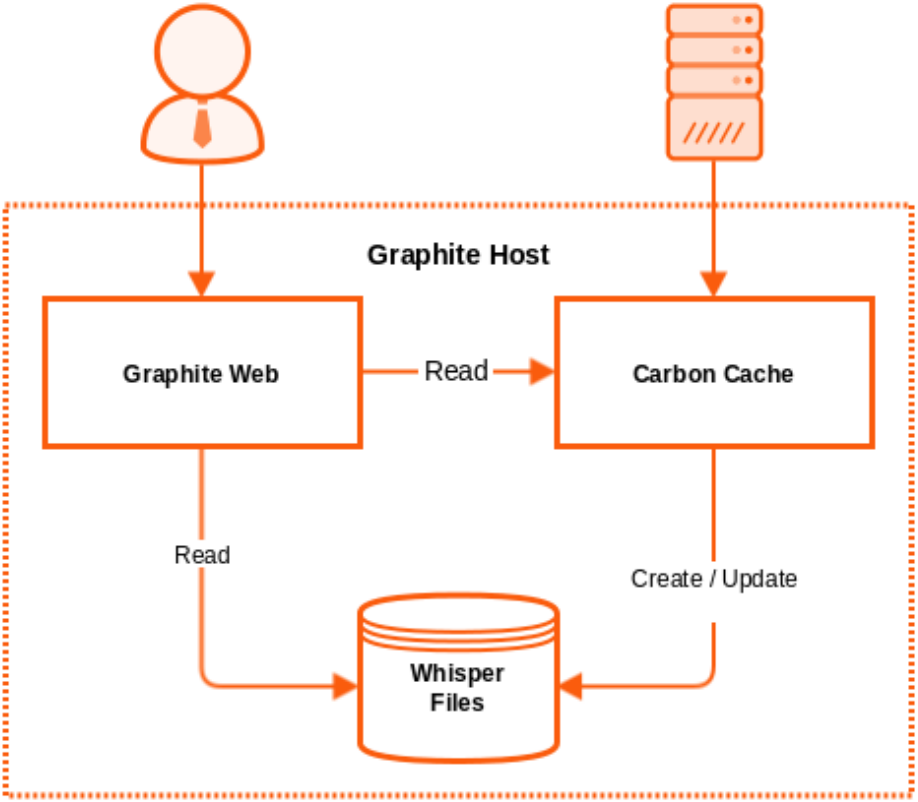
6 Graphite

What is Graphite?

- Store numeric time-series data
- Render graphs of this data on demand
- No collection of data
- Components:
    - Carbon Cache
    - Whisper Files
    - Graphite Web
- Language: Python

Graphite is not just a single product. It consists of multiple components which work together to build a complete performance monitoring solution.

Graphite Single Node Setup



Bareos Graphite Poller

- Reads performance data from Bareos and feeds it into a Graphite backend
- Available from GitHub (https://github.com/aussendorf/bareos-contrib/tree/master/misc/performance/graphite)
- Written in Python by Maik Aussendorf from dass IT (https://www.dass-it.de)
- Requires `python-bareos` from contrib repository
- Import:
    - Events
    - Jobstats
    - Devicestats

Bareos Graphite Poller – Example

```
# vim /etc/bareos/graphite-poller.conf
[director]
server=bareos.localdomain
name=bareos-dir
password=SECRET

[graphite]
server=graphite.localdomain
port=2003
```

Send events ("-e"), jobstats ("-j") and devicestats ("-v") from the past to Graphite Carbon Cache:

```
# ./bareos-graphite-poller.py -c ../graphite-poller.conf \
-e -j -v
JCKKDKWOKDodlOWKALIDCL
```

Bareos Graphite Poller – Structure

```
/var/lib/carbon/whisper/bareos/
└── bareos-dir
    ├── jobs
    │   └── BackupClient1
    │       ├── jobCount.wsp
    │       ├── totalBytes.wsp
    │       └── totalFiles.wsp
    ├── jobsRunning.wsp
    ├── pools
    │   ├── Differential
    │   │   ├── totalBytes.wsp
    │   │   └── totalVolumes.wsp
    │   ├── Full
    │   │   ├── totalBytes.wsp
    │   │   ├── totalVolumes.wsp
    │   │   └── volumeStatus
    │   │       └── Append.wsp
    │   ├── Incremental
    │   │   ├── totalBytes.wsp
    │   │   ├── totalVolumes.wsp
    │   │   └── volumeStatus
    │   │       └── Append.wsp
    │   └── Scratch
    │       ├── totalBytes.wsp
    │       ├── totalVolumes.wsp
    │       └── volumeStatus
    │           └── Append.wsp
    ├── totalBytes.wsp
    ├── totalFiles.wsp
    └── totalJobs.wsp

11 directories, 18 files
```

check_graphite – Plugin

- Available from GitHub (https://github.com/obfuscurity/nagios-scripts)
- Written in Ruby by Jason Dixon
- Plugin Check Command comes with Icinga 2.5 (https://dev.icinga.org/issues/12256)

check_graphite – Installation

```
# yum install rubygem-rest-client
# git clone https://github.com/obfuscurity/nagios-scripts.g
# cp nagios-scripts/check_graphite /usr/lib64/nagios/plugin
# vim /usr/lib64/nagios/plugins/check_graphite
#!/usr/bin/env ruby
+ ##                        $VERBOSE
+ ### -W0  NO Warnings      nil
+ ### -W1  Quiet            false
+ ### -W2  Verbose          true
+ BEGIN { $VERBOSE = nil }
...
# /usr/lib64/nagios/plugins/check_graphite --help
```

check_graphite – Example

```
# ./check_graphite -u graphite-web -m bareos.bareos-dir.job
.BackupClient1.jobCount -w 10 -c 5 -d 1440
WARNING metric count: 6.375 threshold: 10
```

check_graphite – Service Apply Rule Example

```
apply Service "BackupjobCount" {
  import "generic-service"

  check_command = "graphite"

  vars.graphite_url = "graphite-web"
  vars.graphite_metric = "bareos.bareos-dir.jobs.$host.name
  .jobCount"
  vars.graphite_duration = "1440"
  vars.graphite_warning = "10"
  vars.graphite_critical = "5"
  vars.graphite_link_graph = true

  assign where host.vars.os == "Linux"
}

# systemctl reload icinga2.service
# icinga2 object list --type Service --name BackupjobCount
```