Project 5 - Creating a Monitoring Solution

Goal:

The goal of this project is to create a system to monitor the availability of other systems. When you complete this project, you'll have a way to know if one of your systems or services is unavailable. In the real-world this is crucial, especially if the systems you are responsible for provide important functions for the company or organization you work for.

During this project I refer to the documentation for the application. It's located here: https://icinga.com/learn/. You won't need it if you follow the steps in this project, however you can look at it for yourself to see how I came up with this installation process.

Instructions:

Create a Virtual Machine

First, start a command line session on your local machine. Next, move into the working folder you created for this course.

```
cd linuxclass
```

Initialize the vagrant project using the usual process of creating a directory, changing into that directory, and running "vagrant init". We'll name this vagrant project "icinga".

```
mkdir icinga
cd icinga
vagrant init jasonc/centos8
```

Configure the Virtual Machine

Edit the Vagrantfile and set the hostname of the virtual machine to "icinga". Also, assign the IP address of 10.23.45.30 to the machine.

```
config.vm.hostname = "icinga"
config.vm.network "private_network", ip: "10.23.45.30"
```

Start the Virtual Machine

Now you're ready to start the VM and connect to it.

```
vagrant up
vagrant ssh
```

Install Apache

Icinga provides a web interface where you can check the status of hosts and services that you are monitoring. This web application, like many, are LAMP based. We'll start out by installing the first component of the LAMP stack, the Apache HTTP Server.

```
sudo dnf install -y httpd
```

Install PHP

By reading the documentation for Icinga, we learn that it uses a couple of additional PHP modules. We'll install them alongside of PHP.

```
sudo dnf install -y php php-gd php-intl php-ldap php-opcache
```

Configure PHP

Icinga makes use of PHP date functions, so we need to tell PHP what timezone to operate in. To date that, we'll need to update `/etc/php.ini`. It's a good idea to create a backup of a file before you edit it. Let's do that first.

```
sudo cp /etc/php.ini /etc/php.ini.bak
```

Now you're ready to make the required change. Remember that the nano editor is an easy-to-use command line text editor option.

```
sudo nano /etc/php.ini
```

In the "[Date]" section of the `/etc/php.ini` file, change the following line from:

```
;date.timezone =
```

To:

```
date.timezone = "UTC"
```

You don't have to use UTC. Feel free to use your timezone. For a list of supported time zones, visit http://php.net/manual/en/timezones.php. Some examples include "America/New_York", "America/Chicago", "America/Denver", and "America/Los_Angeles". For example, you could use this line of configuration:

```
date.timezone = "America/New_York"
```

Start and Enable the Web Server

Now that the web server and PHP are installed, we can go ahead and start it. We want it to be enabled on boot, so we'll enable it as well.

```
sudo systemctl start httpd
sudo systemctl enable httpd
```

Install MariaDB

Go ahead and install MariaDB. While you're at it, start and enable it.

```
sudo dnf install -y mariadb-server
sudo systemctl start mariadb
sudo systemctl enable mariadb
```

Secure MariaDB

Let's secure the default MariaDB installation. The only information you need to supply is the password for the root database user. For our purposes of practicing our skills we can use a simple password such as "root123". In the real world I would suggest using a stronger password. For the remaining questions you can accept the defaults by pressing ENTER.

```
sudo mysql_secure_installation
```

Example:

```
...
Enter current password for root (enter for none): (press ENTER)
Set root password? [Y/n] (press ENTER)
New password: root123
Re-enter new password: root123
Remove anonymous users? [Y/n] (press ENTER)
Disallow root login remotely? [Y/n] (press ENTER)
Remove test database and access to it? [Y/n] (press ENTER)
Reload privilege tables now? [Y/n] (press ENTER)
```

Create Databases for the Applications

Use the mysqladmin command to create a database named "icinga". This database will be used to store historical monitoring data. Enter the root database password when prompted.

```
mysqladmin -u root -p create icinga
```

Next, create a database named "icingaweb". This database will be used by the web front end to Icinga.

```
mysqladmin -u root -p create icingaweb
```

Create DB Users for the Databases

First, connect to the MariaDB server using the mysql client. Next, use the GRANT command to create the user, set the password, and allow full permissions to the "icinga" database. Name the user "icinga" and set the password to "icinga123".

```
mysql -u root -p
> GRANT ALL on icinga.* to icinga@localhost identified by 'icinga123';
```

Create another user named "icingaweb" and grant it privileges to the icingaweb database. Use "icingaweb123" as the password. Remember to flush the privileges so that permissions for both users become active.

```
> GRANT ALL on icingaweb.* to icingaweb@localhost identified by 'icingaweb123';
> FLUSH PRIVILEGES;
> exit
```

Configure the Icinga Repository

Icinga provides a repository so you can install their software. To add their repository, run the following command.

```
sudo dnf install -y http://mirror.linuxtrainingacademy.com/icinga/icinga-rpm-release.noarch.rpm
```

Internet download location:
https://packages.icinga.com/epel/8/release/noarch/icinga-rpm-release/icinga-rpm-release-8-4.el8.icinga.noarch.rpm

Install Icinga

Now that dnf knows where to find the packages for Icinga, we can install it. Let's install the icinga service, the icinga web front end, the icinga2 package that allows for MariaDB connectivity and the icingacli programs, so we can access icinga via the command line.

```
sudo dnf install -y icinga2 icingaweb2 icingacli icinga2-ido-mysql
```

Configure the Database

Earlier we created the Icinga database, but now we need to configure it. What we need to do is make MariaDB read and execute the Icinga supplied configuration. It's really a series of SQL database commands. To do that, run the following command.

```
mysql -u root -p icinga < /usr/share/icinga2-ido-mysql/schema/mysql.sql
```

The less than character (<) causes the contents of the file to be read in by the program. It's like you started the `mysql` client and typed in the contents of the `/usr/share/icinga2-ido-mysql/schema/mysql.sql` file.

This is one form of I/O (Input/Output) redirection. It's very common for applications to give you SQL files that you need to use in order to create the structure of the database for that application. Let's make sure the command actually created a structure for this database by using the `mysqlshow` command. When you pass a database to the `mysqlshow` command, it returns a list of tables in that database.

```
mysqlshow -u root -p icinga
```

Now we need to tell Icinga how to connect to the database. Edit the `/etc/icinga2/features-available/ido-mysql.conf` file.

```
sudo nano /etc/icinga2/features-available/ido-mysql.conf
```

Remove the double forward slashes (//) from the file in order to set the user, password, host and database. Make sure the password is set to "icinga123". When you are done, the configuration file will look like this:

```
/**
 * The IdoMysqlConnection type implements MySQL support
 * for DB IDO.
 */

object IdoMysqlConnection "ido-mysql" {
  user = "icinga"
  password = "icinga123"
  host = "localhost"
  database = "icinga"
}
```

By the way, Icinga uses "C-Style" comments. A single line comment starts a double forward slash: `//`. To comment an entire section, start the comment with `/*` and close it with `*/`. This allows you to comment out multiple lines without having to change every single commented line.

Enable the IDO-MySQL Feature

Icinga is modular, so you can enable different modules, which they call features. We want to enable the "ido-mysql" feature so that icigna uses the database we just configured to store historical data.

```
sudo icinga2 feature enable ido-mysql
```

You can see what features are enabled by running the following command.

```
sudo icinga2 feature list
```

Install Monitoring Plugins

Now that we have the service installed, we want to be able to monitor hosts and applications. Remember that Icinga can use Nagios monitoring plugins, so we'll install those. However, the packages for those plugins are located in the EPEL repository. First, add the EPEL repository:

```
sudo dnf install -y epel-release
```

One of the nagios monitors, check_disk_smb, requires a Perl package that is not in the default CentOS repository or in the EPEL repository. However, it is in the PowerTools repository. It's disabled by default, so enable it now:

```
sudo dnf config-manager --set-enabled powertools
```

(NOTE: The PowerTools repository is an official RedHat/CentOS repository. It contains a number of packages that may be required as dependencies when installing other applications.)

Now dnf will be able to find the nagios plugins. Install them with this command:

```
sudo dnf install -y nagios-plugins-all
```

Prepare the Server for Clients

This one time process needs to be completed if you will be using an Icinga client installed locally on the machines you plan to monitor.

To prepare this server for that purpose, run the node wizard. Because the goal is to perform a master setup, be sure to answer "**n**" to the first question of "Please specify if this is a satellite setup ('n' installs a master setup) [Y/n]:". Also, answer "**n**" to the last question of "Do you want to disable the inclusion of the conf.d directory [Y/n]:". Accept the defaults for the remaining questions by simply pressing ENTER for thos questions.

```
sudo icinga2 node wizard
```

Example:

```
Please specify if this is a agent/satellite setup ('n' installs a master
setup) [Y/n]: n
...
Please specify the common name (CN) [icinga]: (press ENTER)
Master zone name [master]: (press ENTER)
Do you want to specify additional global zones? [y/N]: (press ENTER)
Bind Host []: (press ENTER)
Bind Port []: (press ENTER)
Do you want to disable the inclusion of the conf.d directory [Y/n]: n
```

Start Icinga

Now we are ready to start the icinga2 service and enable it to start on system boot.

```
sudo systemctl start icinga2.service
sudo systemctl enable icinga2.service
```

Configure the Web front end

Because the web front end comes with some Apache configuration, we need to restart Apache so that the changes are recognized.

```
sudo systemctl restart httpd
```

Next, we need to determine what the Icinga API password is. It is randomly generated by the node wizard.

```
sudo cat /etc/icinga2/conf.d/api-users.conf
```

Note the value on the "password" line. In this example the password is "a4b5bfd036fecf3c". Note: The quotation marks are NOT part of the password.

```
     password = "a4b5bfd036fecf3c"
```

Next, create a setup token. This token is used to prove to the web front end that you are the administrator of Icinga.

```
sudo icingacli setup token create
```

Open a web browser on your local system and navigate to: http://10.23.45.30/icingaweb2/setup. Here, you'll enter the token you just created into the web application. Click the button labeled "Next".

Now you can simply follow the guided installation process. Below is a list of screen names followed by any required information. Many times you will accept the defaults. If you don't see suggested values, accept the defaults.

Modules

Accept the defaults by clicking "Next."

Icinga Web 2

Accept the defaults by clicking "Next."

NOTE: Ignore the "PHP module Imagick is missing" message. You would only need that PHP module if you intended to export graphs in PDF format.

Authentication

Accept the defaults by clicking "Next."

Database Resource

Resource Name: icingaweb_db
Database Type: MySQL
Host: localhost
Port: (leave blank - the default)
Database Name: icingaweb
Username: icingaweb
Password: icingaweb123
Character Set: (leave blank - the default)
Use SSL: (leave unchecked - the default)

Click "Validate Configuration"
Click "Next"

NOTE: If you get prompted for an additional database user with extra privileges, use the MariaDB "root" user and the password you created for that root user. If you followed these instructions, that password will be "root123".

NOTE: If you get an error, see the "Database Troubleshooting Tips" lesson for information on how to correct the issue.

Authentication Backend

Accept the defaults by clicking "Next."

Administration

Username: admin
Password: admin
Repeat password: admin

Click "Next."

Application Configuration

Accept the defaults by clicking "Next."

You've configured Icinga Web 2 successfully

Click "Next."

Welcome to the configuration of the monitoring module for Icinga Web 2

Click "Next."

Monitoring Backend

Accept the defaults by clicking "Next."

Monitoring IDO Resource

Resource Name : icinga_ido
Database Type: MySQL
Host: localhost
Port: (leave blank - the default)
Database Name: icinga
Username: icinga
Password: icinga123
Character Set: (leave blank - the default)
Use SSL: (leave unchecked - the default)

Click "Validate Configuration"
Click "Next"

NOTE: If you get an error, see the "Database Troubleshooting Tips" lesson for information on how to correct the issue.

Command Transport

Transport Name: icinga2
Transport Type: Icinga 2 API
Host: localhost
Port: 5665
SSH port to connect to on the remote Icinga instance
API Username: root
API Password: (Use the value noted from above. Hint: return to the command line and look at the /etc/icinga2/conf.d/api-users.conf file)

Click "Validate Configuration"
Click "Next"

Monitoring Security

Accept the defaults by clicking "Next."

You've configured the monitoring module successfully

Click "Finish"

Log into the Web Front End

After the installation is complete, you can access Icinga via the web at http://10.23.45.30/icingaweb2. Log in with the username of "admin" and the password of "admin". Feel free to explore the web interface and get acquainted with the various views.

Create the Master Zone Configuration Directory

When you ran the Icinga node wizard, a default zone named "master" was created. In Icinga, a zone is a trust hierarchy. For example, members of the master Icinga zone are allowed to send their Icinga check results to the master server. When we start to monitor other servers, which are called Icinga clients or Icinga satellites, they will be part of the master zone.

Configuration for a zone resides in the `/etc/icinga2/zones.d/ZONE_NAME` directory. Let's create the configuration directory for the master zone.

```
sudo mkdir /etc/icinga2/zones.d/master
```

All the configuration for members of the master zone will reside in the `/etc/icinga2/zones.d/master` directory.

Move the Default Monitoring Configuration Into the Master Zone Directory

Icinga includes some default configuration that resides in the `/etc/icinga2/conf.d` directory. The `/etc/icinga2/conf.d/hosts.conf` file contains example configuration to monitor the icinga host itself. Because the Icinga host itself is in the master zone, lets move its configuration into the appropriate directory.

Also, it's a good practice to name the configuration file for a host the same name as the host with ".conf" appended. So, not only will you move the `hosts.conf` file into the master zone directory, you will rename it to `icinga.conf`:

```
sudo mv /etc/icinga2/conf.d/hosts.conf /etc/icinga2/zones.d/master/icinga.conf
```

Because we've reorganized the files in our Icinga configuration, it's a good idea to restart the Icinga service to make sure that we didn't introduce any errors or make any mistakes. Let's restart Icinga:

```
sudo systemctl restart icinga2.service
```

You can return to the web interface at http://10.23.45.30/icingaweb2 and see the same results as before.

Create an DirectoryIndex for the Default DocumentRoot

You might have noticed a warning for the HTTP service. This is because the default Apache welcome page configuration sends a 403 Forbidden message. Apache will only send the 403 Forbidden message if the DocumentRoot does not contain a DirectoryIndex page. We can either disable this check or address the 403 Forbidden message by creating an index file. Let's do the latter and create an `index.html` file in the web server's default DocumentRoot directory.

```
sudo nano /var/www/html/index.html
```

Add the following contents to the file and save your changes.

```
<html>
<body>
<a href="/icingaweb2">Icinga</a>
</body>
</html>
```

This simple web page creates a link to the Icinga Web front end. Visit http://10.23.45.30 in your web browser to see that the HTML page is working.

Next, click on the link to visit the Icinga Web front end (http://10.23.45.30/icingaweb2). Verify that the warning message has disappeared. If it hasn't, wait a few minutes to give Icinga time to recheck the HTTP service.

Update the Default Icinga Host Monitoring Configuration

The HTTP check that Icinga is performing currently ensures that a static HTML file can be served via the web server. However, the Icinga Web front end is written in PHP. If there is a problem serving PHP pages, Icinga will not detect it with the current monitor. So, lets monitor the Icinga web front end at http://10.23.45.30/icingaweb2. To do that, lets update the default monitoring configuration by editing the `/etc/icinga2/zones.d/master/icinga.conf` file.

```
sudo nano /etc/icinga2/zones.d/master/icinga.conf
```

Icinga uses "C-Style" comments. To comment out a single line start it with a double forward slash: `//`. You can also comment an entire section by starting the comment with `/*` and closing it with `*/`. This allows you to comment out multiple lines without having to change every single commented line.

Uncomment the Icinga Web 2 section by changing this:

```
  //vars.http_vhosts["Icinga Web 2"] = {
  //  http_uri = "/icingaweb2"
  //}
```

To this:

```
  vars.http_vhosts["Icinga Web 2"] = {
    http_uri = "/icingaweb2"
  }
```

Save your changes and restart Icinga.

```
sudo systemctl restart icinga2.service
```

Look at the check for http://10.23.45.30/icingaweb2. It should be reported as "OK".

Add a Host to Monitoring

Now let's add some configuration to monitor the "kanboard" host. Create a configuration file for the kanboard host by editing it.

```
sudo nano /etc/icinga2/zones.d/master/kanboard.conf
```

Add the following lines to the file and save it.

```
object Host "kanboard" {
  import "generic-host"
  address = "10.23.45.25"
  vars.os = "Linux"
}
```

The import line causes the configuration for the "generic-host" template to be applied to this host. That configuration lives in the `/etc/icinga2/conf.d/templates.conf` file. This allows you to quickly change a setting in one place and have it applied to multiple hosts. Primarily this configuration tells Icinga how often to perform checks against the host.

The address line tells Icinga what the IP address of the host is. If you have DNS configured in your environment you can use a DNS hostname here, but just know that if you have a problem with DNS then it will cause the checks to fail. Strongly consider using IP addresses as we are doing here.

Icinga allows for custom variables, or custom attributes. When you see "vars.SOME_NAME", it is a custom variable. Here, we set vars.os to be "Linux." This custom variable is used to group hosts in the web front end. So, all the hosts that have "Linux" set for vars.os will be grouped together in the "Linux Servers" group. All the hosts that have "Windows" set for vars.os will be grouped together in the Windows group. You can create your own groups by modifying `/etc/icinga2/conf.d/groups.conf`.

Restart Icinga so that it will read the updated configuration and start to monitor the new host.

```
sudo systemctl restart icinga2.service
```

Open the web frontend (http://10.23.45.30/icingaweb2/monitoring/list/hosts) and check the status of the new host. It will start out in a pending state, meaning Icinga hasn't checked the host yet. After it is checked, Icinga will report if the host is either up or down. If the kanbord VM is running, it will report UP. If it's not running, it will report DOWN.

Open up a new command session and change the state of the kanboard machine. I'm going to assume it was not running and therefore I'm going to start it.

```
cd linuxclass
cd kanboard
vagrant up
```

Return to the Icinga web interface and watch the host status change from "DOWN" to "UP". This may take a few minutes while Icinga performs the checks.

Prepare the Server for the Client

For each Icinga client you install, you'll need to create a ticket for it on the master Icinga server. This is part of the certificate process that allows for secure communications between the master Icinga server and the client machine.

Make sure you're connected to the Icinga server by switching back to your original command line session. Create a ticket for the "kanboard" host.

```
sudo icinga2 pki ticket --cn 'kanboard'
```

You'll need this ticket number when you configure the client.

Install the Icinga Client

Connect to the kanboard VM. Remember to switch to the command line session associated with the kanboard host.

```
vagrant ssh
```

Start off by enabling the Icinga repository.

```
sudo dnf install -y http://mirror.linuxtrainingacademy.com/icinga/icinga-rpm-release.noarch.rpm
```

Next, enabled the EPEL repository.

```
sudo dnf install -y epel-release
```

Now enable the PowerTools repository.

```
sudo dnf config-manager --set-enabled powertools
```

Finally, install Icinga and the Nagios monitors.

```
sudo dnf install -y icinga2 nagios-plugins-all
```

Run the node wizard to tell the client about the master Icinga server. Use the information in bold below to supply the node wizard with the appropriate information.

```
sudo icinga2 node wizard
```

[This space intentionally left blank. Instructions continue on the following page.]

Example:

```
Please specify if this is an agent/satellite setup ('n' installs a master
setup) [Y/n]: (press ENTER)
Please specify the common name (CN) [kanboard]: (press ENTER)
Please specify the master endpoint(s) this node should connect to:
Master/Satellite Common Name (CN from your master/satellite node): icinga
Do you want to establish a connection to the parent node from this node?
[Y/n]: (press ENTER)
Please specify the master/satellite connection information:
Master/Satellite endpoint host (IP address or FQDN): 10.23.45.30
Master/Satellite endpoint port [5665]: (press ENTER)
Add more master/satellite endpoints? [y/N]: (press ENTER)
…
Is this information correct? [y/N]: y
Please specify the request ticket generated on your Icinga 2 master
(optional).
 (Hint: # icinga2 pki ticket --cn 'kanboard'): (Use ticket from above.)
Please specify the API bind host/port (optional):
Bind Host []: (press ENTER)
Bind Port []: (press ENTER)
Accept config from parent node? [y/N]: y
Accept commands from parent node? [y/N]: y
Local zone name [kanboard]: (press ENTER)
Parent zone name [master]: (press ENTER)
Do you want to specify additional global zones? [y/N]: (press ENTER)
Do you want to disable the inclusion of the conf.d directory [Y/n]:
(press ENTER)
```

Start icinga on the kanboard host so it will load this configuration. Remember to enable the service as well.

```
sudo systemctl start icinga2.service
sudo systemctl enable icinga2.service
```

Add Additional Monitors for the Client on the Server

Return to your command line session on the Icinga server. Insert the following configuration on the master Icinga server in `/etc/icinga2/zones.d/master/kanboard.conf`. Be sure to insert it to the top of the file and **leave the existing contents in place**.

```
sudo nano /etc/icinga2/zones.d/master/kanboard.conf
```

```
object Endpoint "kanboard" {
  host = "10.23.45.25"
}

object Zone "kanboard" {
  endpoints = [ "kanboard" ]
  parent = "master"
}
```

When you are done editing the file, it will look like this:

```
object Endpoint "kanboard" {
  host = "10.23.45.25"
}

object Zone "kanboard" {
  endpoints = [ "kanboard" ]
  parent = "master"
}

object Host "kanboard" {
  import "generic-host"
  address = "10.23.45.25"
  vars.os = "Linux"
}
```

This configuration tells the master Icinga server how to connect to the client and gives it permission to request remote check commands for that client. What you have done is created a Zone named "kanboard" that contains one host named "kanboard". You told Icinga that the parent zone of "kanboard" is "master". This allows the master server to communicate with the kanboard host and for the kanboard host to communicate with the master server.

Let's check the load average on the kanboard host. Add the following configuration to the **bottom** of `/etc/icinga2/zones.d/master/kanboard.conf` on the master Icinga server.

```
object Service "load" {
  import "generic-service"
  check_command = "load"
  host_name = "kanboard"
  command_endpoint = "kanboard"
}
```

This configuration creates a service called "load" for the kanboard host. Just like our host imported configuration from the `/etc/icinga2/conf.d/templates.conf` file, this service imports the

"generic-service" configuration from there as well. Primarily this configuration tells Icinga how often to perform checks on the service.

The "check_command" line tells Icinga what check to perform. The load check looks at the systems load and will create warning and critical statuses based on the system's load. A system's load is a calculation which shows, in general, how loaded or busy a server is.

The host_name line associates this check with our "kanboard" host.

Finally, the command_endpoint line tells Icinga to perform the check on the "kanboard" host. If you do not specify a command_endpoint, then Icinga will perform the check from itself. Since we want to know what is happening inside the client, we use the command_endpoint directive.

Since we've updated the configuration, we need to restart icinga.

```
sudo systemctl restart icinga2.service
```

Go back to the web front end and confirm that the load service now appears for the kanboard host.

Let's add even more checks. Add the following configuration to the bottom of `/etc/icinga2/zones.d/master/kanboard.conf` on the master Icinga server.

```
object Service "swap" {
  import "generic-service"
  check_command = "swap"
  host_name = "kanboard"
  command_endpoint = "kanboard"
}

object Service "disk" {
  import "generic-service"
  check_command = "disk"
  host_name = "kanboard"
  command_endpoint = "kanboard"
}
```

These checks will cause the swap space and disk usage to be monitored for the kanboard host.

Since we've updated the configuration, we need to restart icinga.

```
sudo systemctl restart icinga2.service
```

Go back to the web front end and confirm that the swap and disk service checks now appear for the kanboard host.

Process Checks

Let's monitor the processes that are running inside the client. To do that, we can use the "procs" check command. For each process we want to monitor we can create a new service. Add the following configuration to the bottom of `/etc/icinga2/zones.d/master/kanboard.conf` on the master Icinga server.

```
object Service "proc-sshd" {
  import "generic-service"
  check_command = "procs"
  vars.procs_command = "sshd"
  vars.procs_critical = "1:"
  host_name = "kanboard"
  command_endpoint = "kanboard"
}

object Service "proc-httpd" {
  import "generic-service"
  check_command = "procs"
  vars.procs_command = "httpd"
  vars.procs_critical = "1:50"
  host_name = "kanboard"
  command_endpoint = "kanboard"
}

object Service "proc-mysqld" {
  import "generic-service"
  check_command = "procs"
  vars.procs_command = "mysqld"
  vars.procs_critical = "1:1"
  host_name = "kanboard"
  command_endpoint = "kanboard"
}

object Service "proc-rsyslog" {
  import "generic-service"
  check_command = "procs"
  vars.procs_command = "rsyslogd"
  vars.procs_critical = "1:1"
  host_name = "kanboard"
  command_endpoint = "kanboard"
}
```

Many of the plugin check commands allow you to use custom attributes (variables) to control their behavior. Here we've used the proc_command custom attribute (vars.proc_command) to tell the monitor which process to look for. Also, we used procs_crititcal to control what constitutes a critical status. We provide a range to procs_critical. The format for the range is "min:max" or "min:" or ":max". So, if we want at least one process, we use a range of "1:". If we want exactly 1 process we use a range of "1:1". If we want at least 1 process, but no more than 50 we would use a range of "1:50". Any numbers outside the configured range causes a critical status to be raised.

You can find documentation on the plugins and the available custom attributes in the Icinga documentation:
http://docs.icinga.org/icinga2/latest/doc/module/icinga2/chapter/plugin-check-commands

Monitoring MariaDB

We've already configured a process monitor for the "mysqld" process. We can take this to the next level and actually start monitoring the service itself because we want to know if the database is functional. Sometimes a process may be running on a system, but the service that it provides isn't functioning correctly. Let's cover our bases here and monitor the MariaDB service in addition to its process.

Note: Some would argue that you only need to monitor the service. If the MySQL process isn't running, then of course the service will fail too. Also, it doesn't matter if the process is running if the service doesn't work. In any case, if it causes too much noise, you can adjust your checks to your liking.

We want to create a MariaDB user, so we can actually connect to the database to see if it's working. To do that, switch to the command line session associated with the kanboard host and start the mysql client.

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

Now we create an "icinga" user with a password of "icinga123". Remember to flush the privileges.

```
CREATE USER 'icinga' IDENTIFIED BY 'icinga123';
FLUSH PRIVILEGES;
exit
```

Add the following configuration to the bottom of `/etc/icinga2/zones.d/master/kanboard.conf` on the master Icinga server. Make sure you're connected to the Icinga server by switching back to your original command line session.

```
object Service "mysql" {
  import "generic-service"
  check_command = "mysql"
  vars.mysql_username = "icinga"
  vars.mysql_password = "icinga123"
  host_name = "kanboard"
  command_endpoint = "kanboard"
}
```

Here we are using the "mysql" check. We use the check's custom attributes to tell it what credentials to use when connecting to the database. Of course, we associate this check with the kanboard server and run the check on the kanboard host itself.

Since we've updated the configuration, we need to restart icinga.

```
sudo systemctl restart icinga2.service
```

Now the mysql service will appear in the web front end.

Let's add one final check for the kanboard host. Since it runs a web application, let's make sure the web service is monitored. This will be an external check that will run from the Icinga server. To add this check, we can use a custom attribute on the host. Edit `/etc/icinga2/zones.d/master/kanboard.conf` on the master Icinga server and **change the kanboard host stanza** to the following. (**NOTE: DO NOT APPEND A WHOLE NEW STANZA. Change the existing host stanza.**)

```
object Host "kanboard" {
  import "generic-host"
  address = "10.23.45.25"
  vars.os = "Linux"
  vars.http_vhosts["http"] = {
    http_uri = "/"
  }
}
```

Since we've updated the configuration, we need to restart icinga.

```
sudo systemctl restart icinga2.service
```

Verify the monitor is in place by checking the web front end.

Notifications

Up until this point we've been using the web front end to look at the status of hosts and services. Let's tell Icinga to send an email when something is wrong, so we don't have to constantly monitor the web interface. To do that, **update** the kanboard host stanza in `/etc/icinga2/zones.d/master/kanboard.conf` to look like the following.

```
object Host "kanboard" {
  import "generic-host"
  address = "10.23.45.25"
  vars.os = "Linux"
  vars.http_vhosts["http"] = {
    http_uri = "/"
  }
  vars.notification["mail"] = {
    groups = [ "icingaadmins" ]
  }
}
```

This tells Icinga to email anyone in the icingaadmins group. Let's change the email address in `/etc/icinga2/conf.d/users.conf` from icinga@localhost to vagrant@localhost. By the way, if you wanted to create more users and groups, this is the place to do that.

```
sudo nano /etc/icinga2/conf.d/users.conf
```

Since we've updated the configuration, we need to restart icinga.

```
sudo systemctl restart icinga2.service
```

Stop the kanboard host to create an alarm. Switch to the command line session associated with the kanboard host, disconnect from the host and stop it.

```
exit
vagrant halt
```

Now watch in the web interface as the system goes from UP to DOWN. After the system has been down for 5 minutes, it will send an email. To check for mail, use the mail command. Make sure you're connected to the Icinga server by switching back to your original command line session.

```
mail
```

To read a message, type the message number and hit enter. For example: "1<ENTER>". To quit the mail reader, type "q" and hit enter.

OPTIONAL: Configure the System to Send Outbound Emails

If you would like to send email notifications outside the server, follow the steps in the earlier lesson on email. You will configure the Icinga VM to send emails to an SMTP relay host.

Final Configuration

This is the final and complete monitoring configuration for the kanboard host. (NOTE: This configuration is continued on the next two pages.)

`/etc/icinga2/zones.d/master/kanboard.conf`

```
object Endpoint "kanboard" {
  host = "10.23.45.25"
}

object Zone "kanboard" {
  endpoints = [ "kanboard" ]
  parent = "master"
}

object Host "kanboard" {
  import "generic-host"
  address = "10.23.45.25"
  vars.os = "Linux"
  vars.http_vhosts["http"] = {
    http_uri = "/"
  }
  vars.notification["mail"] = {
    groups = [ "icingaadmins" ]
  }
}

object Service "load" {
  import "generic-service"
  check_command = "load"
  host_name = "kanboard"
  command_endpoint = "kanboard"
}
```

```
object Service "swap" {
  import "generic-service"
  check_command = "swap"
  host_name = "kanboard"
  command_endpoint = "kanboard"
}

object Service "disk" {
  import "generic-service"
  check_command = "disk"
  host_name = "kanboard"
  command_endpoint = "kanboard"
}

object Service "proc-sshd" {
  import "generic-service"
  check_command = "procs"
  vars.procs_command = "sshd"
  vars.procs_critical = "1:"
  host_name = "kanboard"
  command_endpoint = "kanboard"
}

object Service "proc-httpd" {
  import "generic-service"
  check_command = "procs"
  vars.procs_command = "httpd"
  vars.procs_critical = "1:50"
  host_name = "kanboard"
  command_endpoint = "kanboard"
}

object Service "proc-mysqld" {
  import "generic-service"
  check_command = "procs"
  vars.procs_command = "mysqld"
  vars.procs_critical = "1:1"
  host_name = "kanboard"
  command_endpoint = "kanboard"
}
```

```
object Service "proc-rsyslog" {
  import "generic-service"
  check_command = "procs"
  vars.procs_command = "rsyslogd"
  vars.procs_critical = "1:1"
  host_name = "kanboard"
  command_endpoint = "kanboard"
}

object Service "mysql" {
  import "generic-service"
  check_command = "mysql"
  vars.mysql_username = "icinga"
  vars.mysql_password = "icinga123"
  host_name = "kanboard"
  command_endpoint = "kanboard"
}
```