How To Monitor Hosts and Services with Icinga on Ubuntu 16.04

Published on May 5, 2017

Ubuntu Monitoring Ubuntu 16.04

Brian Boucheron

Introduction

Icinga is an open-source monitoring system used to monitor the health of networked hosts and services. In this tutorial we will use Icinga to set up two different kinds of monitoring configurations. The first is based on simple network checks of your host's external services, such as making a periodic HTTP request to your website. The other configuration uses a software agent running on the host to gather more detailed system information such as load and number of running processes.

Prerequisites

Before starting this tutorial, you should have completed the previous tutorial in this series, How To Install Icinga and Icinga Web on Ubuntu 16.04. This will leave you with the Icinga

core and Icinga Web interface running on a single host, which we'll refer to as the **icinga-master** node throughout.

You will also need some servers to monitor. We will use two Ubuntu 16.04 servers with Apache installed for our examples. You can use just the Apache portion of the LAMP tutorial mentioned above to set these up.

Step 1 – Setting up Simple Host Monitoring

One simple way to monitor a server with Icinga is to set up a regular check of its externally available services. So for a web host, we'd regularly ping the server's IP address and also try to access a web page. This will tell us if the host is up, and if the web server is functioning correctly.

Let's set up monitoring for a web server. Pick one of the Apache servers mentioned as a prerequisite and make sure the default Apache placeholder page is being served properly. We will call this server `web-1.example.com`. We won't need to log into it at all, all the health checks will be configured and executed on the master node.

> **Note:** Icinga always defaults to using the Fully Qualified Domain Name (FQDN) of any host it's dealing with. A FQDN is a hostname plus its domain name, so `web-1.example.com`, for example. If you don't have a proper domain set up for a host, the FQDN will be something like `web-1.localdomain`.
>
> These are fine to use, just be consistent, and if you don't have a "real" FQDN always use the server's IP address in any Icinga `address` field you configure.

Log into the master node. To add a new host, we need to edit Icinga's `hosts.conf` file. Open it in a text editor:

```
$ sudo nano /etc/icinga2/conf.d/hosts.conf
```

Copy

This will open a file with some explanatory comments and a single host *block* defined. The existing `object Host NodeName` configuration block defines the **icinga-master** host, which is the host we installed Icinga and Icinga Web on. Position your cursor at the bottom of the file, and add a new host:

/etc/icinga2/conf.d/hosts.conf

```
. . .
object Host " web-1.example.com " {
  import "generic-host"
  address = " web-1_server_ip "
  vars.http_vhosts["http"] = {
    http_uri = "/"
  }
  vars.notification["mail"] = {
    groups = [ "icingaadmins" ]
  }
}
```

This defines a host called `web-1.example.com`, imports some default host configs from a template called `generic-host`, points Icinga to the correct IP address, and then defines a few variables that tell Icinga to check for an HTTP response at the root (`/`) URL and notify the `icingaadmins` group via email when problems occur.

Save and close the file, then restart Icinga:

```
$ sudo systemctl restart icinga2
```

Copy

Switch back to the Icinga Web interface in your browser. The interface updates itself fairly rapidly, so you don't need to refresh the page. The new host information should populate in short order, and the health checks will change from **Pending** to **Ok** once Icinga gathers enough information.

This is a great way to monitor external services on a host, and there are other checks available for SSH servers, SMTP, and so on. But it'd also be nice to know more details about the internal health of the servers we're monitoring.

Next, we'll set up monitoring via an Icinga agent, so we can keep an eye on more detailed system information.

Step 2 – Setting up Agent-based Monitoring

Icinga provides a mechanism for securely communicating between a master and client node in order to run more extensive remote health checks. Instead of only knowing that our web server is successfully serving pages, we could also monitor CPU load, number of process, disk space and so on.

We're going to set up a second server to monitor. We'll call it `web-2.example.com`. We need to install the Icinga software on the remote machine, run some setup wizards to make the connection, then update some configuration files on the Icinga master node.

> **Note:** There are many ways to architect an Icinga installation, complete with multiple tiers of **master/satellite/client** nodes, high-availability failover, and multiple ways to share configuration details between nodes. We are setting up a simple two tier structure with one **master** node and multiple **client** nodes. Further, all configuration will be done on the **master** node, and our health check commands will be scheduled on the master node and pushed to the clients. The Icinga project calls this setup **Top Down Command Endpoint** mode.

Set up the Master Node

First, we need to set up the master node to make client connections. We do that by running the node setup *wizard* on our master node:

```
$ sudo icinga2 node wizard
```
Copy

This will start a script that asks us a few questions, and sets things up for us. In the following, we hit `ENTER` to accept most defaults. Non-default answers are highlighted. Some informational output has been removed for clarity:

```
Output
Please specify if this is a satellite setup ('n' installs a master setup) [Y/n]:  n
Starting the Master setup routine...
Please specify the common name (CN) [icinga-master.example.com]:  ENTER to accept the de
Checking for existing certificates for common name 'icinga-master.example.com'...
Certificates not yet generated. Running 'api setup' now.
Enabling feature api. Make sure to restart Icinga 2 for these changes to take effect.
Please specify the API bind host/port (optional):  ENTER
Bind Host []:  ENTER
Bind Port []:  ENTER
Done.

Now restart your Icinga 2 daemon to finish the installation!
```

Restart Icinga to finish updating the configuration:

```
$ sudo systemctl restart icinga2
```
Copy

Open up a firewall port to allow external connections to Icinga:

```
$ sudo ufw allow 5665
```

Copy

Now we'll switch to the client node, install Icinga and run the same wizard.

Set up the Client Node

Log into the server we're calling **web-2.example.com** . We need to install the Icinga repository again, and then install Icinga itself. This is the same procedure we used on the master node. First install the key:

```
$ curl -sSL https://packages.icinga.com/icinga.key | sudo apt-key add -
```

Copy

Open the `icinga.list` file:

```
$ sudo nano /etc/apt/sources.list.d/icinga.list
```

Copy

Paste in the repository details:

/etc/apt/sources.list.d/icinga.list

```
deb https://packages.icinga.com/ubuntu icinga-xenial main
```

Save and close the file, then update the package cache:

```
$ sudo apt-get update
```

Copy

Then, install `icinga2` . Note that *we do not need* the `icinga2-ido-mysql` package that we installed on the master node:

```
$ sudo apt-get install icinga2
```

Copy

Now we run through the node wizard on this server, but we do a *satellite* config instead of *master*:

```
$ sudo icinga2 node wizard
```

Copy

```
Output
Please specify if this is a satellite setup ('n' installs a master setup) [Y/n]:  y
Starting the Node setup routine...
Please specify the common name (CN) [ web-2.example.com ]:  ENTER
Please specify the master endpoint(s) this node should connect to:
Master Common Name (CN from your master setup):  icinga-master.example.com
Do you want to establish a connection to the master from this node? [Y/n]:  y
Please fill out the master connection information:
Master endpoint host (Your master's IP address or FQDN):  icinga-master_server_ip
Master endpoint port [5665]:  ENTER
Add more master endpoints? [y/N]:  ENTER
Please specify the master connection for CSR auto-signing (defaults to master endpoint h
Host [ icinga-master_server_ip ]:  ENTER
Port [5665]:  ENTER
```

The wizard will now fetch the public certificate from our master node and show us its details. Confirm the information, then continue:

```
Output
. . .
Is this information correct? [y/N]:  y
information/cli: Received trusted master certificate.

Please specify the request ticket generated on your Icinga 2 master.
 (Hint: # icinga2 pki ticket --cn ' web-2.example.com '):
```

At this point, switch back to your `master` server and run the command the wizard prompted you to. Don't forget `sudo` in front of it:

```
$ sudo icinga2 pki ticket --cn ' web-2.example.com '
```

Copy

The command will output a key. Copy it to your clipboard, then switch back to the client node, paste it in and hit `ENTER` to continue with the wizard.

```
Output
. . .
information/cli: Requesting certificate with ticket ' 5f53864a504a1c42ad69faa930bffa3a12

Please specify the API bind host/port (optional):
Bind Host []:  ENTER
Bind Port []:  ENTER
Accept config from master? [y/N]:  n
Accept commands from master? [y/N]:  y
Done.

Now restart your Icinga 2 daemon to finish the installation!
```

Now open up the Icinga port on your firewall:

```
$ sudo ufw allow 5665
```

Copy

And restart Icinga to fully update the configuration:

```
$ sudo systemctl restart icinga2
```

Copy

You can verify there's a connection between the two servers with `netstat`:

```
$ netstat | grep :5665
```

Copy

It might take a moment for the connection to be made. Eventually `netstat` will output a line showing an `ESTABLISHED` connection on the correct port.

```
Output
tcp        0      0  web-2_server_ip : 58188      icinga-master_server_ip :5665     ESTA
```

This shows that our servers have connected and we're ready to configure the client checks.

Configure the Agent Monitoring

Even though the master and client are now connected, there's still some setup to do on the master to enable monitoring. We need to set up a new host file. Switch back to the master node.

One important level of organization in an Icinga install is the concept of a *zone*. All client nodes must create their own zone, and report to a parent zone, in this case our master node. By default our master node's zone is named after its FQDN. We'll make a directory named after our master zone within Icinga's `zones.d` directory. This will hold the information for all the master zone's clients.

Create the zone directory:

```
$ sudo mkdir /etc/icinga2/zones.d/ icinga-master.example.com
```
Copy

We're going to create a services configuration file. This will define some service checks that we'll perform on any remote client node. Open the file now:

```
$ sudo nano /etc/icinga2/zones.d/ icinga-master.example.com /services.conf
```
Copy

Paste in the following, then save and close:

services.conf

```
apply Service "load" {
  import "generic-service"
  check_command = "load"
  command_endpoint = host.vars.client_endpoint
  assign where host.vars.client_endpoint
}

apply Service "procs" {
  import "generic-service"
  check_command = "procs"
  command_endpoint = host.vars.client_endpoint
  assign where host.vars.client_endpoint
}
```

This defines two service checks. The first will report on the CPU load, and the second will check the number of processes on the server. The last two lines of each service definition are important. `command_endpoint` tells Icinga that this service check needs to be sent to a

remote command endpoint. The `assign where` line automatically assigns the service check to any host that has a `client_endpoint` variable defined.

Let's create such a host now. Open a new file in the zone directory we previously created. Here we've named the file after the remote host:

```
$ sudo nano /etc/icinga2/zones.d/ icinga-master.example.com/ web-2.example.com
```
Copy

Paste in the following configuration, then save and close the file:

web-2.example.com.conf

```
object Zone " web-2.example.com " {
  endpoints = [ " web-2.example.com " ]
  parent = " icinga-master.example.com "
}

object Endpoint " web-2.example.com " {
  host = " web-2_server_ip "
}

object Host " web-2.example.com " {
  import "generic-host"
  address = " web-2_server_ip "
  vars.http_vhosts["http"] = {
    http_uri = "/"
  }
  vars.notification["mail"] = {
    groups = [ "icingaadmins" ]
  }
  vars.client_endpoint = name
}
```

This file defines a zone for our remote host, and ties it back to the parent zone. It also defines the host as an endpoint, and then defines the host itself, importing some default rules from the `generic-host` template. It also sets some `vars` to create an HTTP check and enable email notifications. Note that because this host has `vars.client_endpoint = name` defined, it will also be assigned the service checks we just defined in `services.conf`.

Restart Icinga to update the config:

```
$ sudo systemctl restart icinga2
```
Copy

Switch back to the Icinga Web interface, and the new host will show up with checks **Pending**. After a few moments, those checks should turn **Ok**. This means our client node is successfully running checks for the master node.

Conclusion

In this tutorial we set up two different types of monitoring with Icinga, external service checks and agent-based host checks. There's much more to learn about configuring and working with Icinga, so you'll likely want to dive deeper into Icinga's extensive documentation.

Note that if you get to a point where you require custom *check* commands, you'll need to sync these from the master to the client nodes using a *global configuration zone*. You can find out more about this specific feature here.

Finally, if you have a large number of servers to monitor, you might look into using configuration management software to automate your Icinga configuration updates. Our tutorial series Getting Started with Configuration Management give a comprehensive overview of the concepts and software involved.

About the authors