Red Hat Reference Architecture Series

OpenShift Enterprise 1.1

Monitoring the PaaS Infrastructure

Scott Collier, Sr. Principal Software Engineer
RHCA

Brett Lentz, Sr. Software Engineer
RHCE

Version 1.0

April 2013

redhat.

1801 Varsity Drive™
Raleigh NC 27606-2072 USA
Phone: +1 919 754 3700
Phone: 888 733 4281
Fax: +1 919 754 3701
PO Box 13588
Research Triangle Park NC 27709 USA

The GPG fingerprint of the security@redhat.com key is:
CA 20 86 86 2B D6 9D FC 65 F6 EC C4 21 91 80 CD DB 42 A6 0E

Comments and Feedback

In the spirit of open source, we invite anyone to provide feedback and comments on any reference architectures. Although we review our papers internally, sometimes issues or typographical errors are encountered. Feedback allows us to not only improve the quality of the papers we produce, but allows the reader to provide their thoughts on potential improvements and topic expansion to the papers.

Feedback on the papers can be provided by emailing refarch-feedback@redhat.com. Please refer to the title within the email.

Staying In Touch

Join us on some of the popular social media sites where we keep our audience informed on new reference architectures as well as offer related information on things we find interesting.

Like us on Facebook:

https://www.facebook.com/rhrefarch

Follow us on Twitter:

https://twitter.com/RedHatRefArch

Plus us on Google+:

https://plus.google.com/u/0/b/114152126783830728030/

Accessing Reference Architectures

There are 2 ways to access Red Hat reference architectures:

- The Red Hat customer portal at http://access.redhat.com using a customer login
- The Red Hat resource library at http://www.redhat.com/resourcelibrary/reference-architectures/

The benefits of using the customer portal is that in addition to the Red Hat reference architecture, the reference architecture team also provides scripts and configuration files used to build out the environment. When using the Red Hat resource library, only the reference architecture is available.

Table of Contents

1 Executive Summary

A Platform-as-a-Service, or PaaS, is a cloud application platform in which the application configuration, build, deployment, hosting, and administration is automated in an elastic cloud environment along with their supported stacks. OpenShift is a Platform as a Service (PaaS) solution from Red Hat. OpenShift provides a multi-language, auto-scaling, self-service, elastic cloud application platform built on a proven stack of Red Hat technologies. The OpenShift PaaS services are available as both a hosted service and on-premise PaaS product offering. OpenShift Enterprise is Red Hat's on-premise PaaS software solution that enables customers to deploy their own Private or Hybrid PaaS environment.

OpenShift Enterprise enables administrators to more quickly serve the needs of application developers by deploying a PaaS platform that streamlines the application service delivery process. OpenShift Enterprise enables administrators and developers to more quickly serve their needs by standardizing and accelerating developer work-flows. OpenShift Enterprise does this in a highly-efficient, fine-grained, multi-tenant cloud architecture that maximizes infrastructure utilization. This provides application developers with self-service access so they can easily deploy applications on-demand leveraging the languages, frameworks, and tools of their choosing. It ultimately increases developer efficiency and agility by allowing them to focus on what they do best, writing code, and in turn enables them to better meet the demands of the business.

OpenShift Enterprise is built on the strength of a proven stack of Red Hat technologies. It provides the freedom for developers and administrators to work the way they want to work. For developers, OpenShift Enterprise supports Java, Ruby, PHP, Python, and Perl programming languages with associated frameworks as well as additional platform services like MySQL and PostgreSQL databases, Jenkins Continuous Integration server, and more. OpenShift Enterprise "cartridges" provide these platform services. Cartridges are extensible, enabling customers to extend OpenShift to add their own custom services. Developers can access OpenShift Enterprise via a rich command line interface (CLI), web console, RESTful API, or Eclipse integrated development environment (IDE). For IT administrators, OpenShift Enterprise runs on the trusted Red Hat Enterprise Linux platform and leverages technologies such as SELinux and Cgroups to provide a secure and scalable multi-tenant environment, that allows efficient infrastructure utilization supporting application development and deployment. OpenShift also provides standards-based application run-time systems like JBoss Enterprise Application Platform, Tomcat, and Apache. OpenShift Enterprise can be deployed on-premise in a customer's data center or private cloud on their choice of virtualization platform or cloud provider. In addition, developers can work from a variety of workstations including Red Hat Enterprise Linux, Mac OS X and Microsoft Windows. OpenShift Enterprise is open source, based on the upstream OpenShift Origin project, that helps drive innovation and eliminate lock-in.

This reference architecture demonstrates how to monitor a OpenShift Enterprise

infrastructure by showing a detailed walk through using the Zabbix monitoring software and a few examples of what needs to be set up and monitored. In addition to the specific examples shown using Zabbix, general items are shown so that they can be applied and monitored from any monitoring software.

2 Components Overview

2.1 OpenShift Enterprise

2.1.1 Broker

The *broker* is the single point of contact for all application management activities. It is responsible for the following tasks:

- Manage user authentication
- DNS updates
- Application state
- Application orchestration
- Services and operations

In addition to being able to contact the *broker* directly via a RESTful API, the following methods may also be used

- Web console
- Command line interface (CLI) tools
- Eclipse JBoss Developer Studio

OpenShift Enterprise is composed of several components that make the above tasks possible. This reference architecture focuses on monitoring a distributing OpenShift Enterprise environment. In order to do that, a Zabbix server is installed and for OpenShift Enterprise, the following services are configured on separate systems, which are typically accessed by the *broker*:

- MongoDB
- ActiveMQ
- Authentication Services
- BIND

MongoDB is set up in a replicated configuration over three hosts. Although it is not required, in this configuration, the same hosts provide ActiveMQ services for messaging. In this configuration, if one *broker* or *broker support node* (bsn) host goes down, the OpenShift Enterprise infrastructure is still operational. Product documentation for OpenShift Enterprise

can be found on Red Hat's customer portal[1].

2.1.2 Node

An OpenShift *node* is a host that runs the applications. There can be many nodes that are deployed that a broker manages. A *node* supports gears that contain applications. Gears represent the system resources that an application is consuming. One of the features of OpenShift is that it provides a true multi-tenant environment. This is accomplished by using SELinux and Cgroup restrictions to separate each applications' resources and data and also providing quality of service controls.

A node and a broker can be on the same host but that is not recommended except in a proof of concept environment. This reference architecture has deployed the *node* services onto a set of dedicated *node* hosts.

2.2 Red Hat Enterprise Virtualization

The Red Hat Enterprise Virtualization portfolio is an end-to-end virtualization solution, with use cases for both servers and desktops, designed to overcome current IT challenges for consolidation, enable pervasive data center virtualization, and unlock unprecedented capital and operational efficiency. The Red Hat Enterprise Virtualization portfolio builds upon the Red Hat Enterprise Linux platform that is trusted by thousands of organizations on millions of systems around the world for their most mission-critical workloads. Combined with KVM, the latest generation of virtualization technology, Red Hat Enterprise Virtualization delivers a secure, robust virtualization platform with unmatched performance and scalability for Red Hat Enterprise Linux and Windows guests. Red Hat Enterprise Virtualization shares the same platform certification as Red Hat Enterprise Linux for compatibility and performance.

2.3 Zabbix

Zabbix is an enterprise class availability and performance monitoring solution. Zabbix is available in two forms, open source and commercial. This reference architecture uses the open source version of Zabbix that is available via Extra Packages for Enterprise Linux (EPEL). As such, readers of this reference architecture should not call into Red Hat for Support using Zabbix. For more information on Red Hat support policies, please see the customer portal[2]. While Zabbix offers many features, the features this reference architecture focus on are:

- Agent based monitoring
- Items
- Triggers
- Action driven events
- External script utilization

1 https://access.redhat.com/knowledge/docs/OpenShift_Enterprise/

2 https://access.redhat.com/support/offerings/production/soc.html

There are many different industry standard tools that can be used to pull the metrics needed for a successful monitoring system. This reference architecture uses Zabbix as the tool of choice to provide a real world example of how a OpenShift Enterprise architecture can be monitored. The same methods can be applied to any systems management software.

2.4 Red Hat Enterprise Linux

Red Hat Enterprise Linux 6, the latest release of Red Hat's trusted datacenter platform, delivers advances in application performance, scalability, and security. With Red Hat Enterprise Linux 6, physical, virtual, and cloud computing resources can be deployed within the data center. Red Hat Enterprise Linux 6.4 provides the following features and capabilities:

Reliability, Availability, and Security (RAS):

- More sockets, more cores, more threads, and more memory
- RAS hardware-based hot add of CPUs and memory is enabled
- Memory pages with errors can be declared as “poisoned” and can be avoided

File Systems:

- ext4 is the default file system and scales to 16TB
- XFS is available as an add-on and can scale to 100TB
- Fuse allows file systems to run in user space allowing testing and development on newer fuse-based file systems (such as cloud file systems)

High Availability:

- Extends the current clustering solution to the virtual environment allowing for high availability of virtual machines and applications running inside those virtual machines
- Enables NFSv4 resource agent monitoring
- Introduction of Cluster Configuration System (CCS). CCS is a command line tool that allows for complete CLI administration of Red Hat's High Availability Add-On

Resource Management:

- Cgroups organize system tasks so that they can be tracked and other system services can control the resources that Cgroup tasks may consume
- cpuset applies CPU resource limits to Cgroups, allowing processing performance to be allocated to tasks

3 Reference Architecture Configuration

This reference architecture is based on a distributed configuration that is documented on the Red Hat customer portal[3]. Please refer to the Deploying and Managing a Private PaaS with OpenShift Enterprise for all the details. A summary of the environment is described in **Illustration 1: OpenShift Enterprise Infrastructure**. Only the differences between the prior reference architecture and the Zabbix components are shown here.

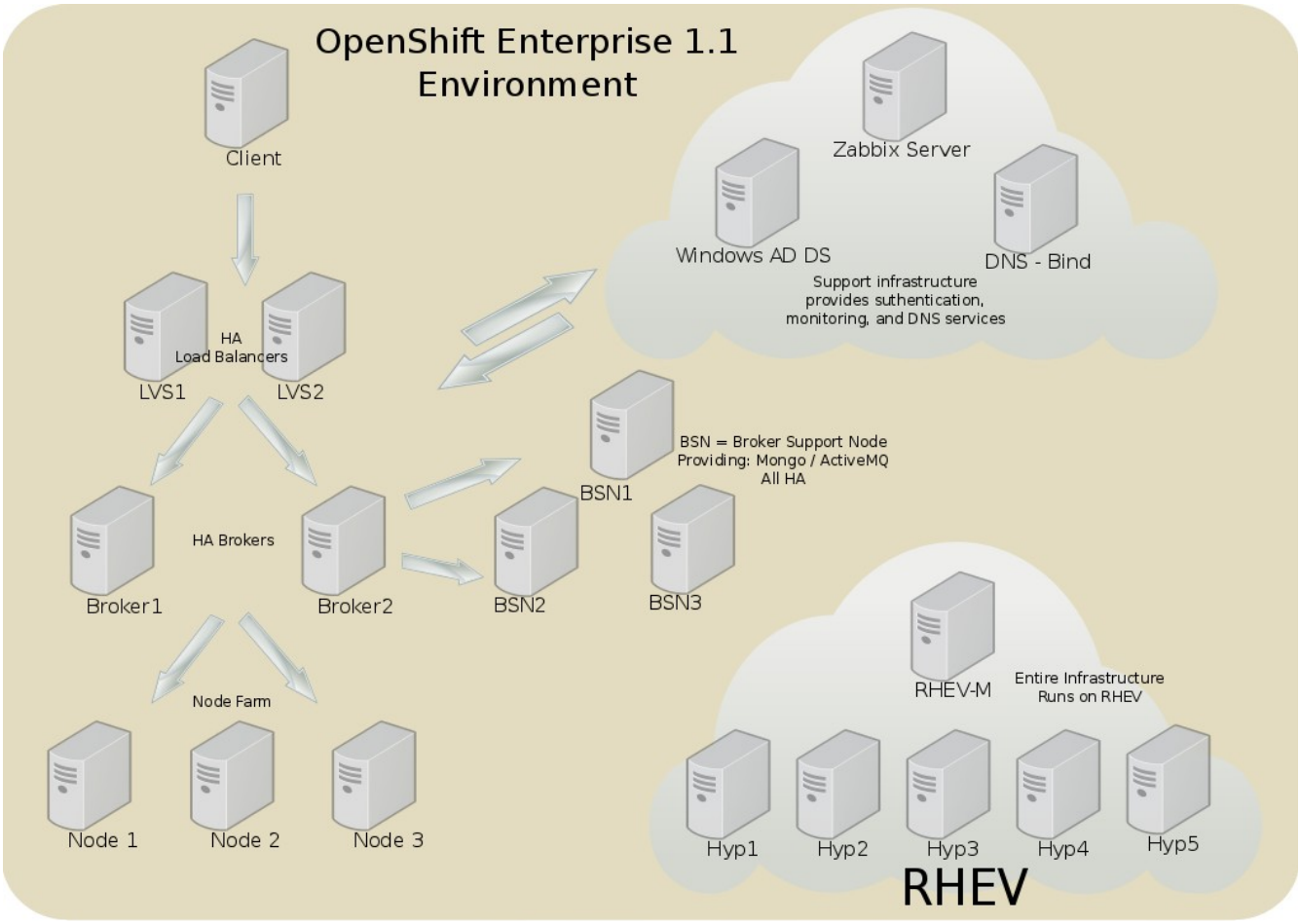


Illustration 1: OpenShift Enterprise Infrastructure

3 http://www.redhat.com/resourcelibrary/reference-architectures/deploying-and-managing-a-private-paas-with-openshift-enterprise

The servers that this reference architecture focus on monitoring are the *Brokers*, *Nodes* and *Broker Support Nodes*. The load balancers and the other servers can be monitored but were left out as there are not any OpenShift Enterprise specific items to report on. This does not mean, however, that they should be left out of the overall monitored solution.

Hostname	Software	Role	Zabbix Server
Zabbix Server	Red Hat Enterprise Linux 6	Client	2.0.5-1.el6

Table 3.1: Zabbix Server Software

Hostname	Software	Role	Zabbix Client
bsn{1,2,3}	Red Hat Enterprise Linux 6	Broker Support Node	2.0.5-1.el6
broker{1,2}	Red Hat Enterprise Linux 6	Broker	2.0.5-1.el6
node{1,2}	Red Hat Enterprise Linux 6	Node	2.0.5-1.el6

Table 3.2: Zabbix Client Software

3.1 Software Configuration

Configuration Files

All of the configuration files needed to reproduce this reference architecture are included on the Red Hat customer portal. A log-in is required to access the configuration files. A list of all the files are available in **Appendix B: Configuration Files**.

4 Monitoring Overview

There are several hosts involved in a distributed OpenShift Enterprise infrastructure. These hosts include *Brokers*, *Broker Support Nodes*, *Nodes*, Load balancers, etc. Monitoring metrics for these hosts can be broken down into two categories. The first category is standard system specific metrics. These metrics include items like CPU utilization, memory utilization, file system size, uptime, etc. The second category is targeted towards metrics that are interesting from an OpenShift Enterprise perspective. This reference architecture focuses on the latter of the two categories.

This chapter shows several items that can be monitored. These items are categorized by the type of host. For example, the broker is going to have specific metrics that a node might not have. The items proposed in this paper are not a comprehensive list. There are items in every environment that may or may not need to be monitored that depends on use case, environment, and operational goals.

Keep in mind that when collecting data to monitor, that data is nothing more than a number. The data becomes interesting when it is acted upon by going below or exceeding a certain threshold. Every administrator is responsible for deciding which thresholds are important for his or her environment. By design, Zabbix separates data collection “items” from operational or business logic “triggers” that combine to determine alert behaviors “actions”.

There are different actions that can take place depending on what the administrator wants to do. For example, when an administrator is setting up the check to see if SELinux is enabled on a node, a decision must be made at that time as to what happens if SELinux reports as disabled. There are a couple of options, the first is to send some sort of notification to the administrator so that an action can be taken, the second is to have the monitoring software try a self-healing approach.

4.1 Broker Monitoring

app status

This metric is useful to make sure that applications are functioning properly. One potential check that can be used is “rhc app status”. If that command succeeds, users should be able to work with their applications.

available gears for "x" district

This metric reports back how many more gears can be created per district. This is helpful for determining when more nodes need to be added to the district to increase capacity. Depending on the environment, node provisioning may be an automated or manual process.

output of check_create_app

This metric ensures that applications can be created. If this check succeeds then users should be able to create applications.

selinux module status

This metric ensures that SElinux is enabled and set to enforcing.

inactive nodes in a district

This metric checks to make sure each node in the district is active. If the check succeeds then that node should be available to run gears. If not, then the administrator needs to check and see why the node was marked as inactive.

4.2 Node Monitoring

number of running apps

This metric reports how many applications are running on the node. This is useful for capacity reporting and identifying over-utilized or under-utilized nodes.

node active capacity (%)

This metric reports how many gears are currently active out of the total possible number of gears that could possibly run on that node.

number of idled apps

This metric reports back how many applications are idle on the node. This is helpful when trying to decide how much the node can and should be over-committed.

number of mcollectived processes

The brokers use MCollective and ActiveMQ to communicate with the nodes. If MCollective is not accessible, no new apps can be created and existing apps cannot be changed.

number of new apps

This metric shows the number of new applications on the node since last count. Setting a threshold at “X” number lets the administrator know when the node is close to exceeding capacity.

number of not started apps

This metric reports the number of applications on the node that are not idled and not started. Depending on implementation, this could indicate localized problems with user's applications, or with a particular cartridge, or widespread issues with the node itself.

number of total apps

This metric reports the number of applications on the node that are either not started, started but idled, and running.

number of unknown apps

This metric is a catch-all for apps that don't fall into a pre-defined state. Anything that shows up here tends to be indicative of a bug, as all app states should be defined.

selinux module status

This metric reports back whether or not SELinux is installed on the node and whether or not it is enforcing.

4.3 Broker Support Node Monitoring

activemq_heap_committed

This is the amount of memory guaranteed to be available to activemq. Depending on the linux kernel's memory overcommit settings, this can be a more accurate indicator of ActiveMQ performance than OS-level memory utilization.

activemq_heap_used

This is a generalized ActiveMQ memory utilization metric. Monitoring it enables tracking performance characteristics of ActiveMQ.

activemq_nonheap_committed

This is a generalized ActiveMQ memory utilization metric. Monitoring it enables tracking performance characteristics of ActiveMQ.

activemq queue count

Number of messages in ActiveMQ's message queues. This could indicate a performance problem or misconfiguration when there are high numbers of messages lingering in the queues.

activemq thread count

This measures the current number ActiveMQ threads. If this number is elevated, it could indicate a high amount of broker usage. Excessive broker activity may be a sign of performance issues, configuration issues, or malicious users.

5 How to Monitor with Zabbix

The stages of monitoring with Zabbix include the following:

1. Monitor some metric with a Zabbix agent and return some value to the Zabbix server.
2. Use triggers to define operational requirements as they relate to changes in the value.
3. If the value exceeds some given threshold, cause an event and take a defined action. This could be a self healing mechanism like restarting a service or sending a email to an on-call administration.

To deploy this monitoring infrastructure follow these steps:

1. Deploy Zabbix Server
2. Configure Zabbix server
   a) Notifications
   b) Host Groups
3. Configure Zabbix templates
   a) Broker
   b) Node
   c) Broker Support Nodes
4. Deploy Zabbix Agent on *Broker Support Nodes*, *Brokers,* and *Nodes*.
5. Add hosts to Zabbix web based user interface and assign templates to hosts
6. Configure events

Each of these steps are discussed in detail. Steps 2-6 can be repeated to add more hosts / items as required. The example used in this reference architecture is configuring a *node* host with a “Number of Running Applications” metric.

5.1 Deploy Zabbix Server

The Zabbix server runs on a dedicated host. In this reference architecture the Zabbix host is a virtual machine in the reference architecture lab. The Zabbix host is running Red Hat Enterprise Linux 6.4 with PostgreSQL and Zabbix server packages. For more information about installing and configuring Zabbix, please see the Zabbix documentation[4].

Install the Zabbix and PostgreSQL packages:

1. Install EPEL.

```
# yum install http://mirror.itc.virginia.edu/fedora-epel/6/i386/epel-release-6-8.noarch.rpm
```

2. Install the Zabbix packages. In this reference architecture PostgreSQL is used as the

4 http://www.zabbix.com/documentation.php

database, Zabbix supports either PostgreSQL or MySQL.

```
# yum install zabbix20-web-pgsql.noarch zabbix20-web.noarch zabbix20-server-
pgsql.x86_64 zabbix20-web.noarch zabbix20.x86_64 zabbix20-agent.x86_64
postgresql.x86_64  postgresql-docs.x86_64 postgresql-server.x86_64
```

Configure the Postgresql database:

1. Initialize the Postgresql database.

```
# service postgresql initdb
```

2. Allow the database to listen on all interfaces.

```
# cp /var/lib/pgsql/data/postgresql.conf{,.orig}

# sed -i "s/#listen_addresses = 'localhost'/listen_addresses = '\*'/"
/var/lib/pgsql/data/postgresql.conf
```

3. Configure authentication for PostgreSQL. Ensure that these PostgreSQL security settings are configured to meet the security requirements of the organization in which PostgreSQL is deployed.

```
# cp /var/lib/pgsql/data/pg_hba.conf{,.orig}

# tail -n 4 /var/lib/pgsql/data/pg_hba.conf
host    all         all         ::1/128               trust
local   all         all                               trust
local   zabbix         zabbix                               trust
host    zabbix         zabbix         10.16.138.23/21         trust
```

4. Start PostgreSQL and ensure that the database starts on boot.

```
# service  postgresql start
Starting postgresql service:                               [  OK  ]

# chkconfig postgresql on
```

5. Change to the postgres user. As the postgres user, create the database and assign the appropriate permissions.

```
# su – postgres

$ psql
psql (8.4.13)
Type "help" for help.

# create database zabbix;

# create user zabbix createdb password 'PASSWORD';

# \q
```

6. As the postgres user, import the Zabbix data.

```
$ cd /usr/share/zabbix-postgresql

$ cat schema.sql | psql -d zabbix -U zabbix
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index
"maintenances_pkey" for table "maintenances"
CREATE TABLE
CREATE INDEX
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index "hosts_pkey"
for table "hosts"
CREATE TABLE
CREATE INDEX
CREATE INDEX
CREATE INDEX
CREATE INDEX
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index "groups_pkey"
for table "groups"
CREATE TABLE
CREATE INDEX
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index
"screens_pkey" for table "screens"

                          ... <OUTPUT OMMITTED>...

$ cat images.sql | psql -d zabbix -U zabbix
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1


                          ... <OUTPUT OMMITTED>...

$ cat data.sql | psql -d zabbix -U zabbix
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
                          ... <OUTPUT OMMITTED>...
```

7. Change back to the root user, configure PHP for Zabbix.

```
# cp /etc/php.ini{,.orig}

# sed -i 's/;date.timezone =/date.timezone="America\/New_York"/'
/etc/php.ini
```

```
# sed -i 's/post_max_size = 8M/post_max_size = 16M/' /etc/php.ini

# sed -i 's/max_execution_time = 30/max_execution_time = 300/' /etc/php.ini

# sed -i 's/max_input_time = 60/max_input_time = 300/' /etc/php.ini
```

8. Configure the firewall to allow the appropriate traffic. Open ports for the web server and Zabbix client traffic.

```
# lokkit –-service=http

# lokkit --port=10051:tcp
```

9. Configure SELinux.

```
# setsebool -P httpd_can_network_connect 1
```

10. Configure the timeout for the Zabbix server.

```
# sed -i 's/# Timeout=3/Timeout=30/' /etc/zabbix_server.conf
```

11. Enable the services and ensure they start on boot.

```
# service httpd start
Starting httpd:                                            [  OK  ]

# service zabbix-server restart
Shutting down Zabbix server:                               [FAILED]

# chkconfig httpd on

# chkconfig zabbix-server on
```

5.2 Configure Zabbix Server

After the Zabbix packages have been installed and the database has been set up, the next step is to run through the Zabbix installation wizard. This reference architecture is structured so that the configuration is first done via the Zabbix web based interface and then the Zabbix agents are configured on each host. Both activities are required for this monitoring solution to work.

5.2.1 Complete Installation Wizard

1. Connect to the Zabbix server.

   http://zabbix.osop.cloud.lab.eng.bos.redhat.com/zabbix

Illustration 2: Start Configuration

2. Check the pre-requisites.

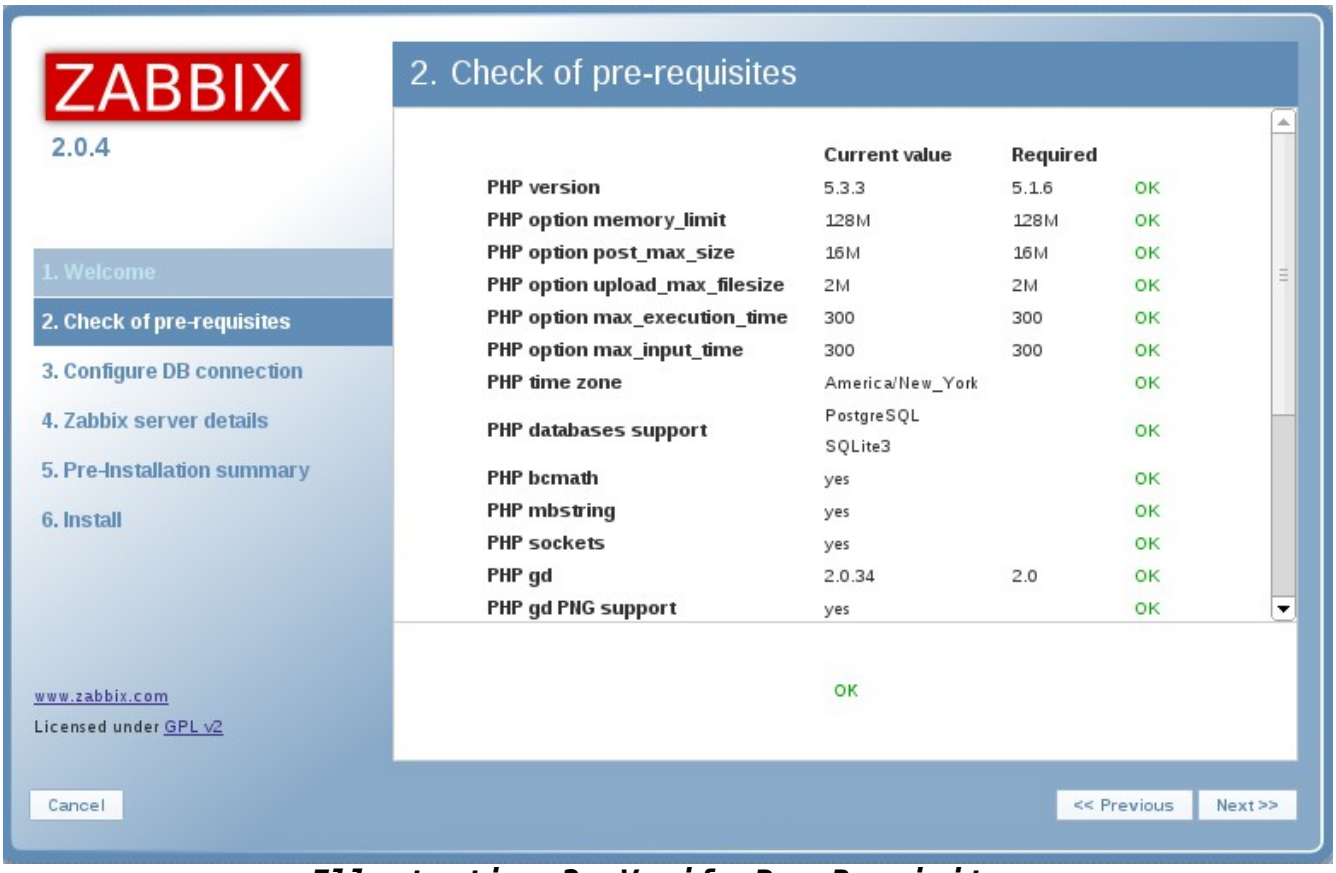


Illustration 3: Verify Pre-Requisites

3. Configure the database connection. Provide the username and password that was set up in section **5.1 Deploy Zabbix Server**.

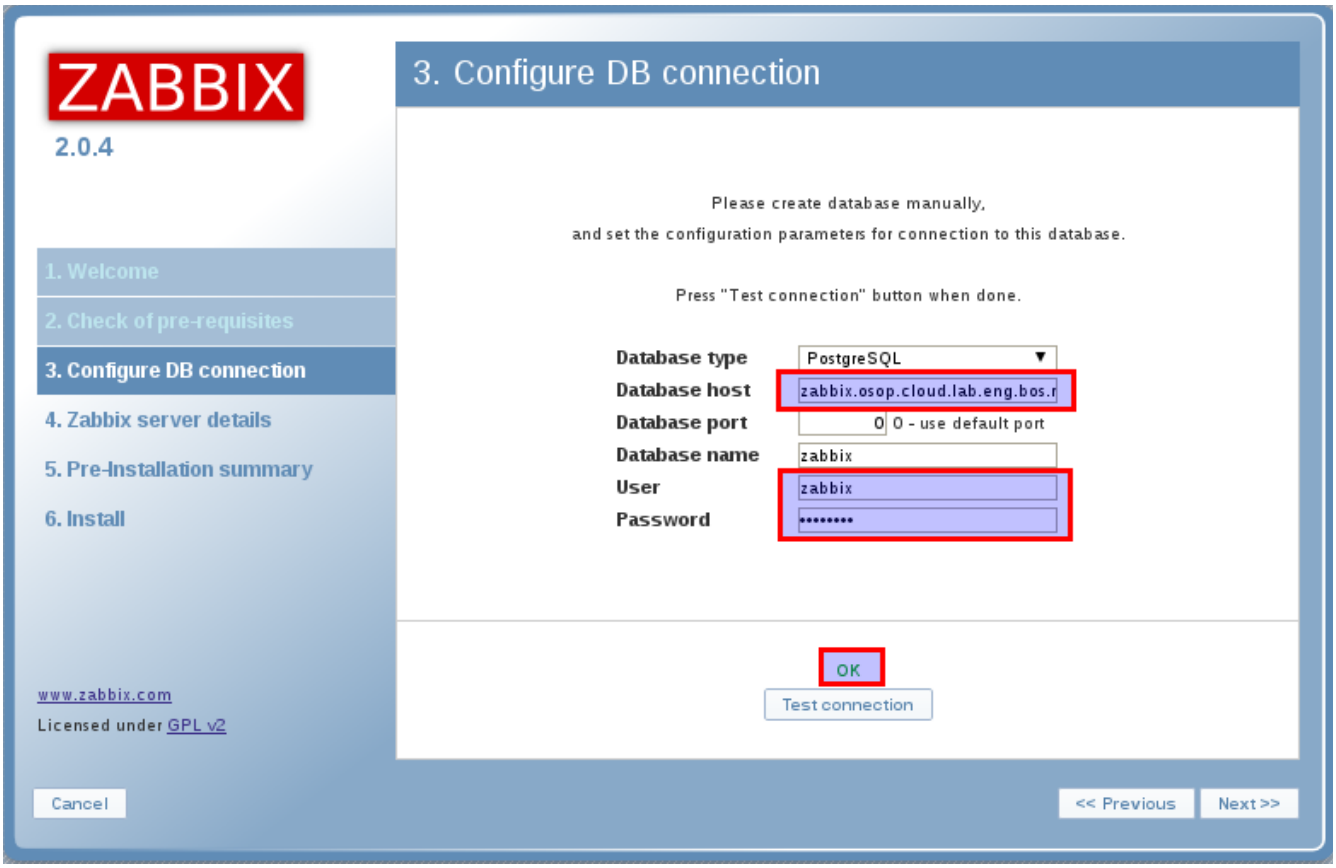


Illustration 4: Connect to Database

4. Provide the Zabbix server details.

Illustration 5: Enter Server Details

5. Check the summary.

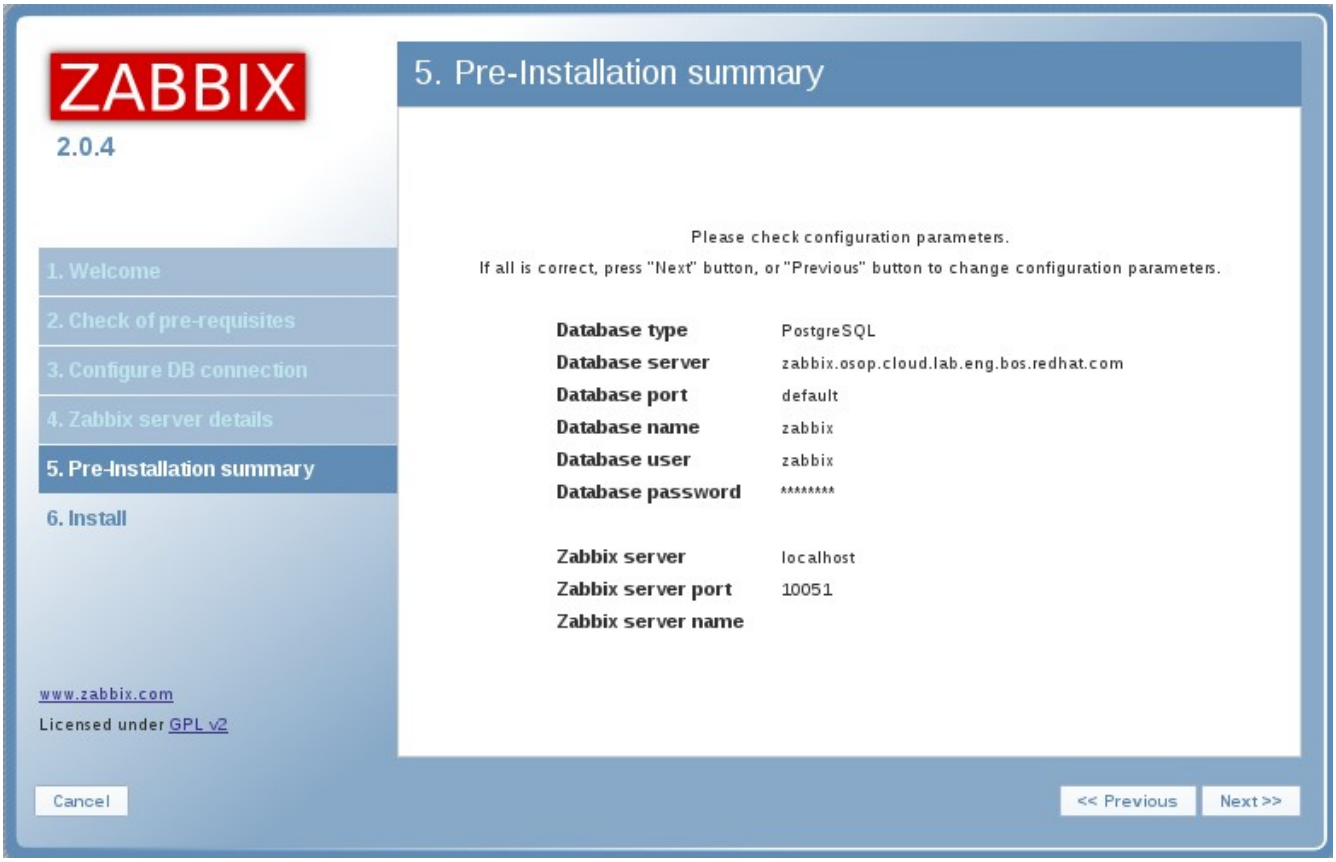


Illustration 6: Summary

6. Finalize the installation.

Illustration 7: Complete Configuration

5.2.2 Configure Email Notifications

Zabbix classifies message delivery mechanisms as “Media Types”. This reference architecture uses the email media type to send notifications via email to a system administrator if an event is triggered. Zabbix supports other media types such as SMS, Jabber, Ez Texting, and Customer Alert Scripts. However, these media types are outside the scope of this reference architecture.

1. Log into the Zabbix web based interface. The default username and password is *Admin* | *zabbix*.

2. Hover over **Administration** and click on **Users**, finally click on the **Admin** user.

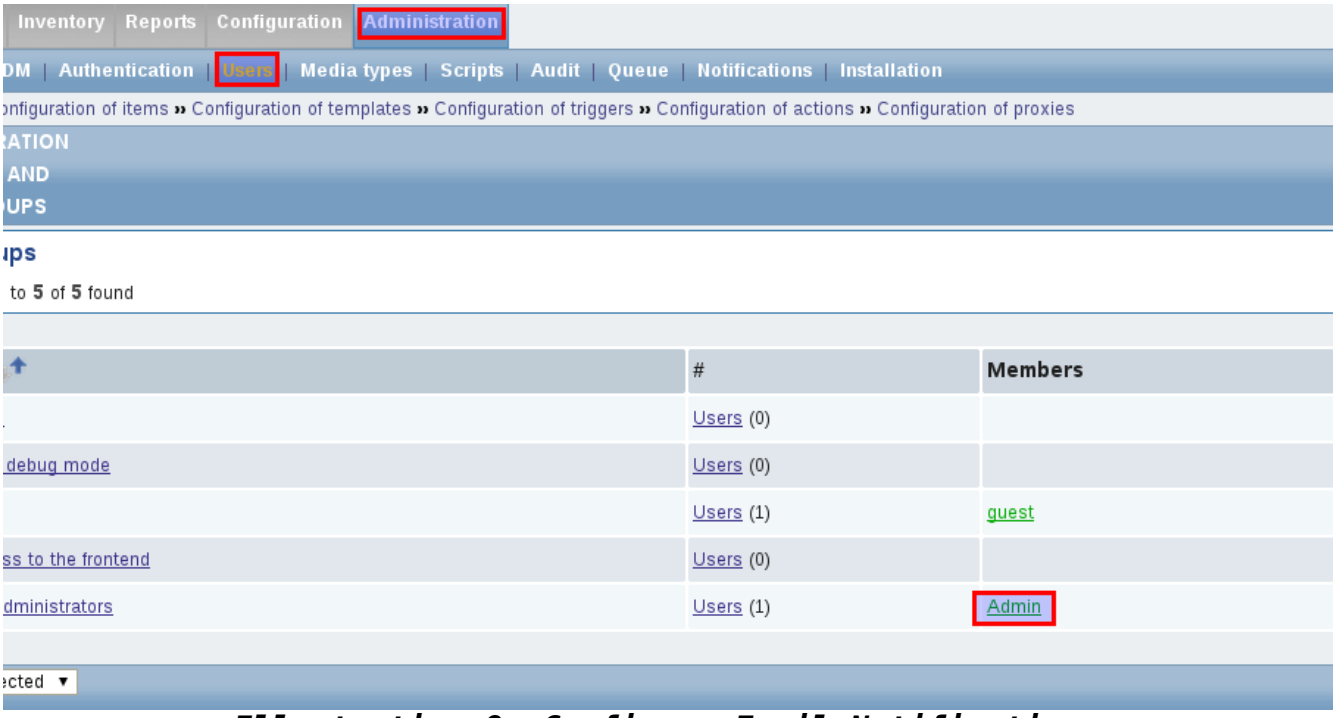


Illustration 8: Configure Email Notifications

3. Click on the **Media** tab and click **Add**.

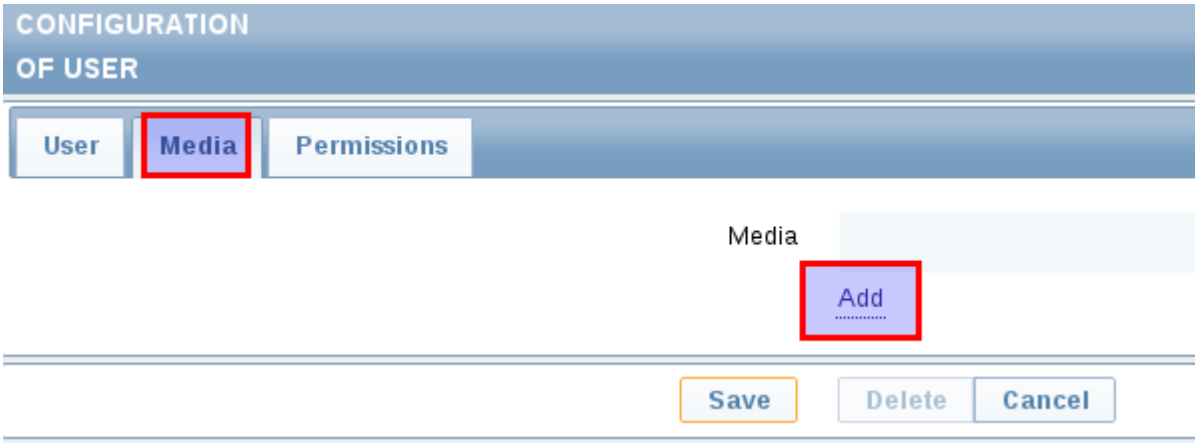


Illustration 9: Add Email as a Notification

4. Provide **Send to** email address and click **Add**.

Illustration 10: Finalize the Email Media Type Configuration

5. Click **Save**.

5.2.3 Set Up Notifications on Zabbix Server

1. Configure Zabbix so that it can send email alerts to administrators. Hover over **Administration**, click on **Media Types,** and finally **Email**.

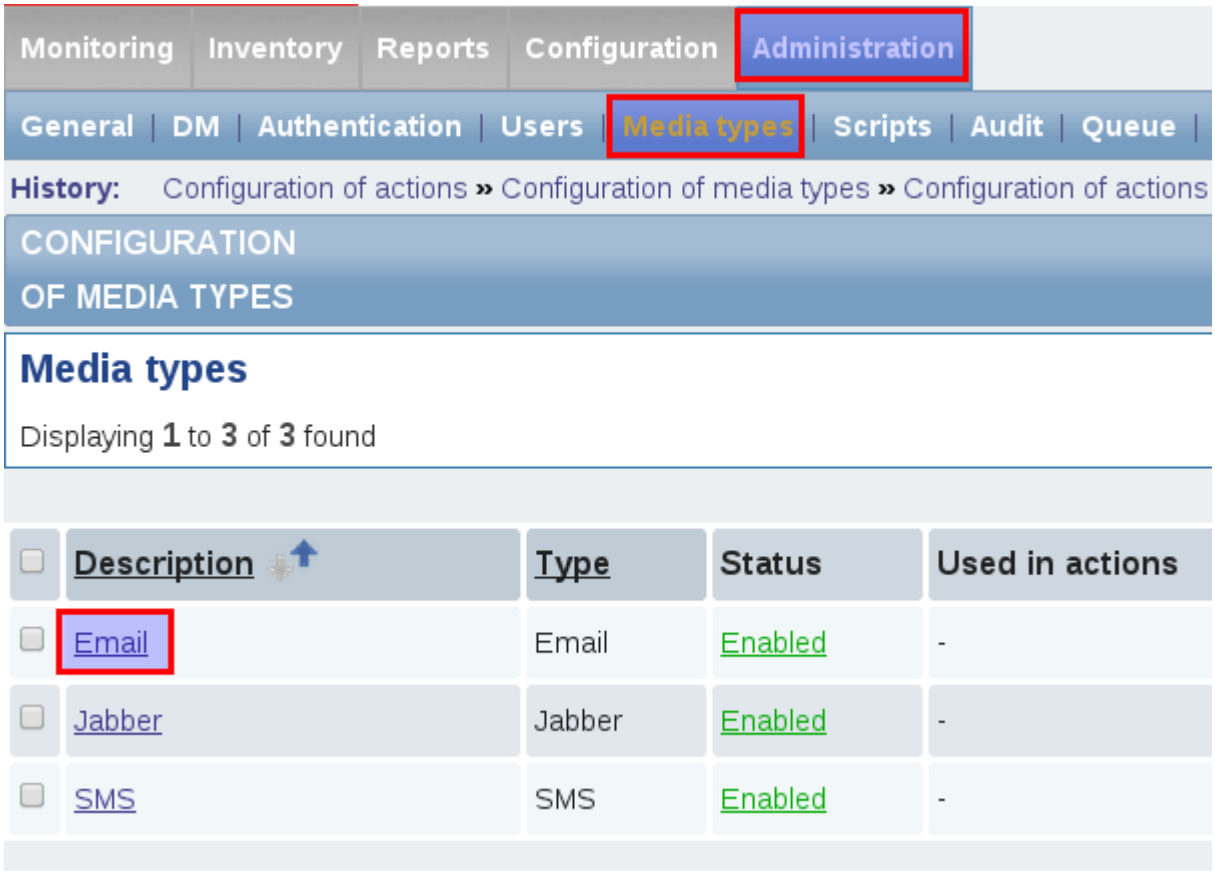


Illustration 11: Media Types

The **Used in actions** field is populated once an action is associated with this **Media type**.

2. Provide a **SMTP server**, **SMTP helo** and **SMTP email**. Click **Save**.

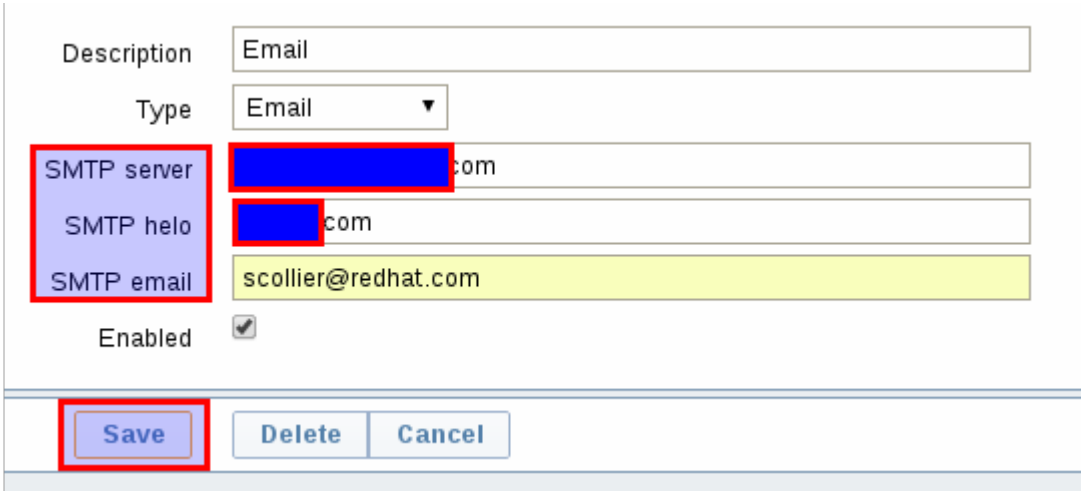


Illustration 12: SMTP Server Details

Configure Zabbix Host Groups

Host groups are useful for separating hosts out. For example, this reference architecture shows how to create host groups for *nodes*, *brokers,* and finally *broker support nodes*. When hosts are deployed and added to Zabbix, they go into one of these host groups.

Create a Host Group

1. Hover over **Configuration** and click **Host Groups,** and then click **Create host group**.

Illustration 13: Configure Host Groups

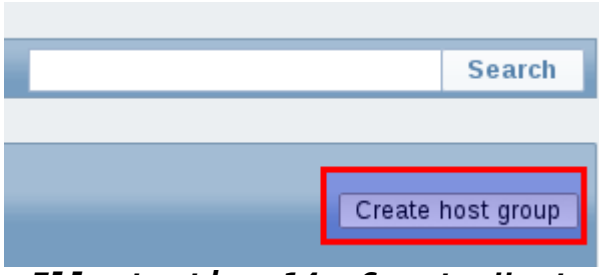


Illustration 14: Create Host Group

2. Create the following host groups by providing a **Group name** and clicking **Save**.

   OpenShift_Nodes_Host_Group

   OpenShift_Broker_Host_Group

   OpenShift_BSN_Host_Group

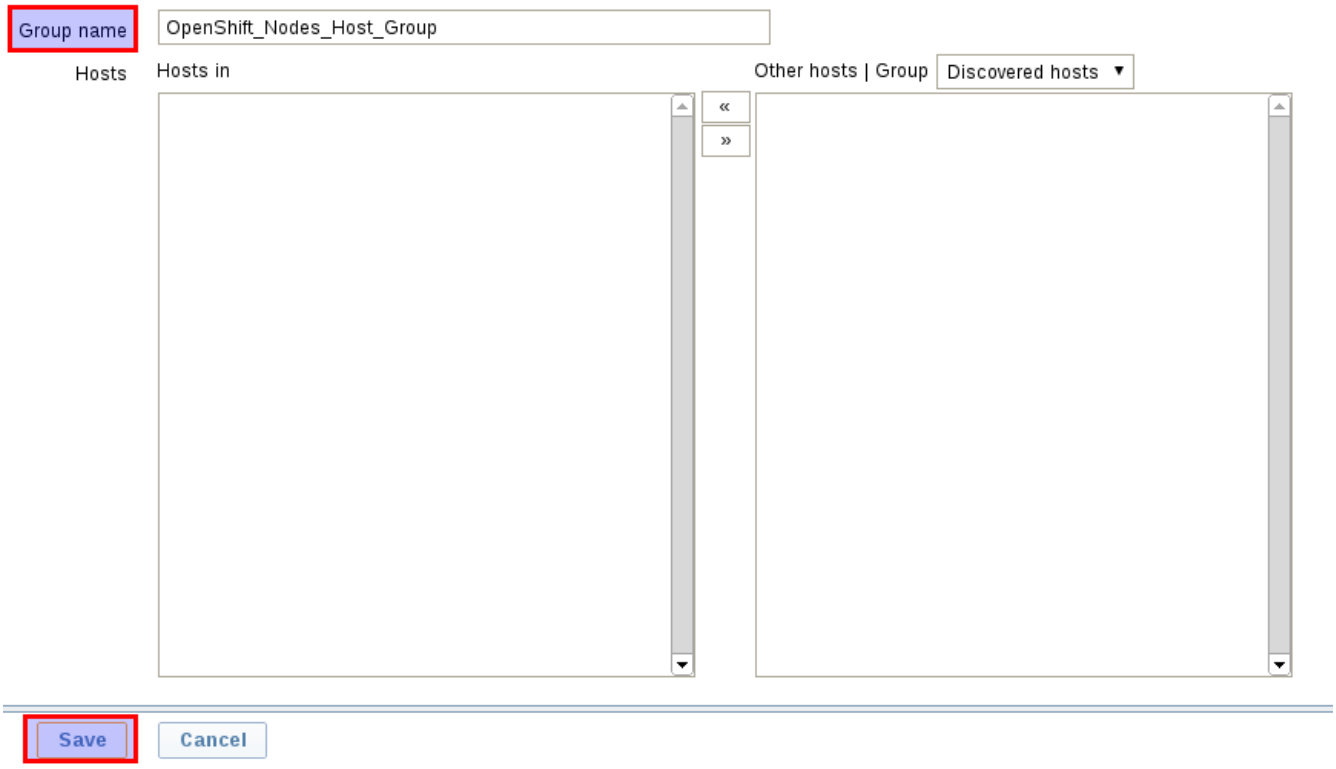


Illustration 15: Provide Group Name

The host groups should now show up if you hover over **Configuration** and click on **Host Groups**.

5.3 Add Hosts to Zabbix

This section demonstrates how to add a host to Zabbix via the web user interface. At this point, the Zabbix agent has not been installed on the host. This is covered in **5.7 Zabbix Agents**.

1. Hover over **Configuration**, click **Hosts**, and then click **Create host**.

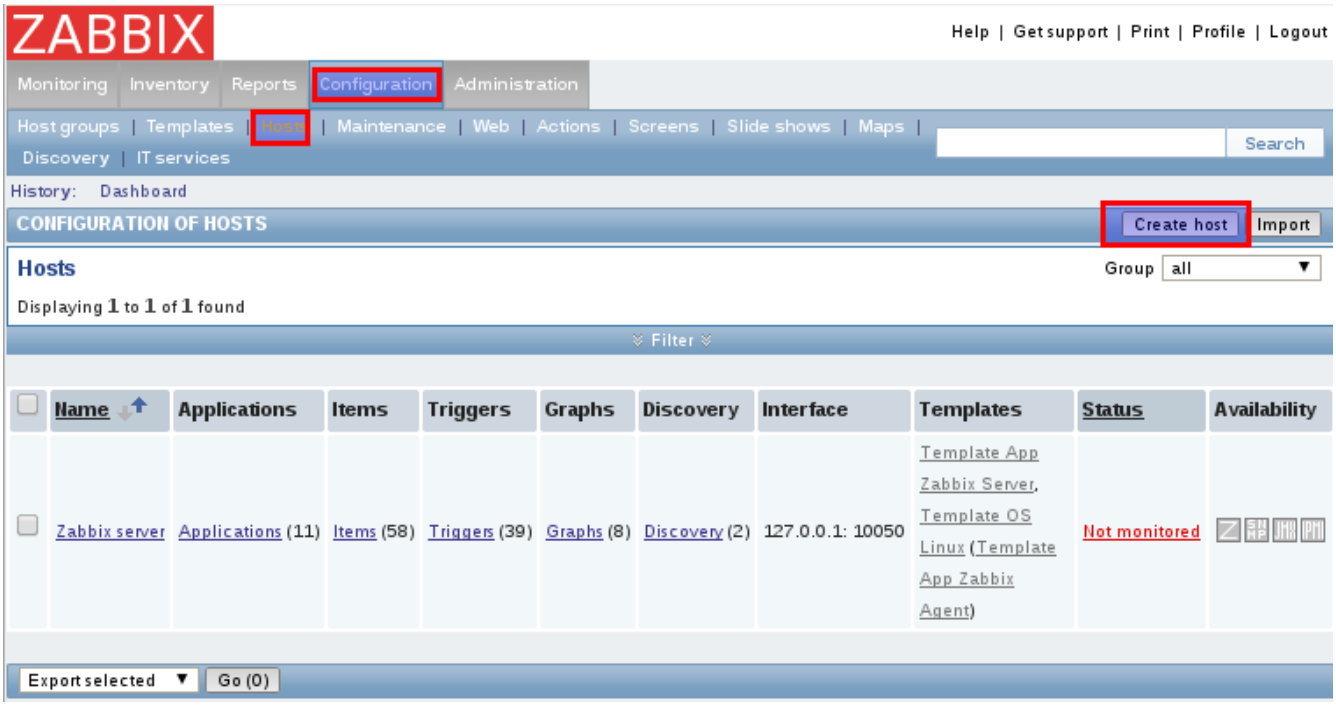


Illustration 16: Create Host

2. Configure parameters for the host as shown in **Illustration 17: Details for Host**. Provide the **Host name**, **Visible name**, choose the **OpenShift_Nodes_Host_Group,** and set up the **Agent Interfaces**. Finally, click **Save**.

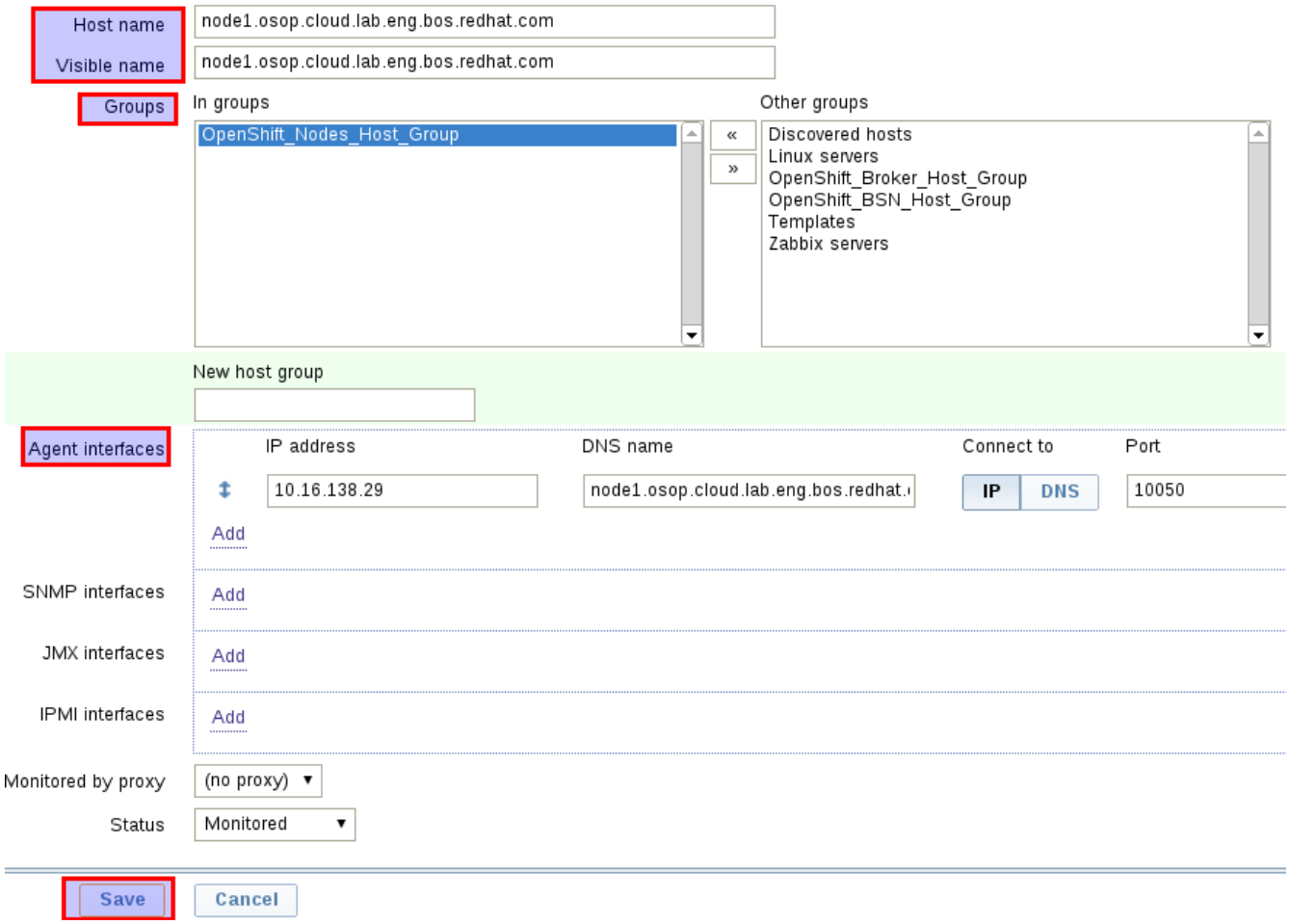


Illustration 17: Details for Host

3. Repeat these steps to add the following hosts in **Table 5.3.1: Repeat on the Following Host Groups**.

Hostname	Group
node{2,3}	OpenShift_Node_Host_Group
broker{1,2}	OpenShift_Broker_Host_Group
bsn{1,2,3}	OpenShift_BSN_Host_Group

Table 5.3.1: Repeat on the Following Host Groups

5.4 Configure Zabbix Templates

Per the Zabbix documentation[5], templates are a set of entities that can be conveniently applied to multiple hosts. Entities may be items, triggers, graphs, applications, screens, and low-level discovery rules.

This section describes how to create a template for a node. The process needs to be repeated for each additional template.

Create a node template

1. Hover over **Configuration** and click **Templates** and finally click **Create template**.

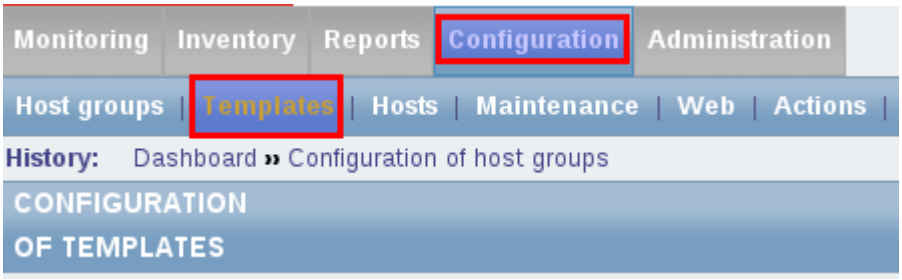


Illustration 18: Create Template

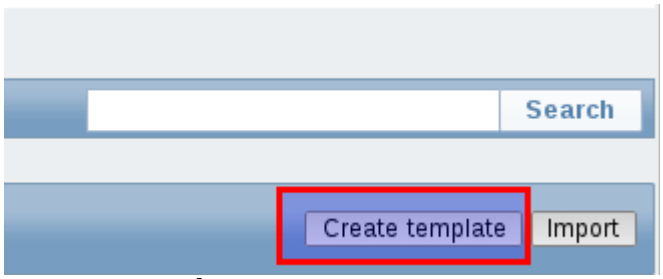


Illustration 19: Create Template

5 https://www.zabbix.com/documentation/2.0/manual/config/templates

2. Provide a **Template name** and add the *OpenShift_Nodes_Host_Group* to the **Groups** section and click **Save**.

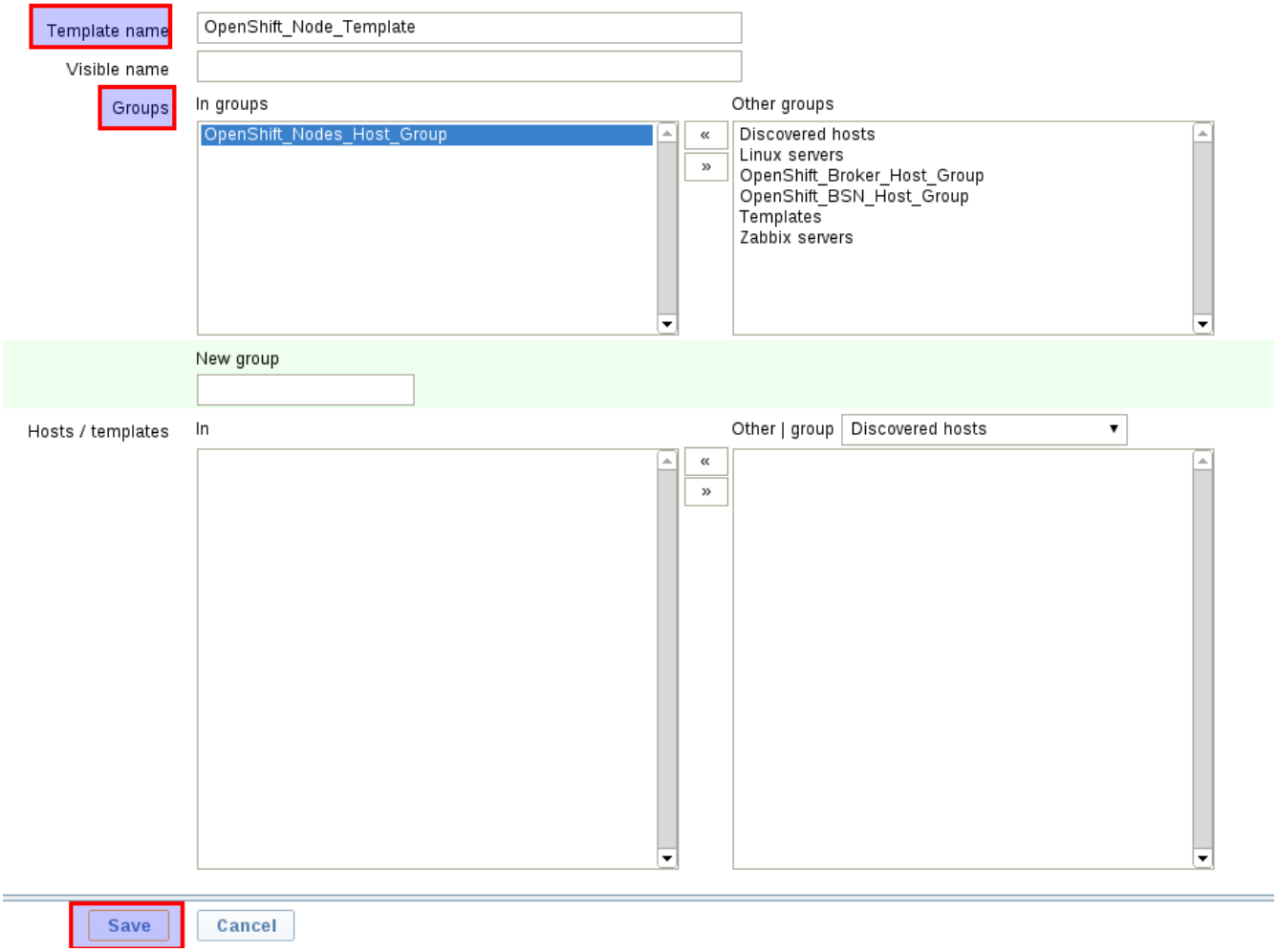


Illustration 20: Template Details

3. Repeat these steps to add the following templates shown in **Table 5.4.1: Create the Following Templates**.

Template Name	Group
OpenShift_BSN_Template	OpenShift_BSN_Host_Group
Openshift_Broker_Template	OpenShift_Broker_Host_Group

Table 5.4.1: Create the Following Templates

5.4.1 Create Items

Items are used to gather data from a host[6]. An item is an individual metric and one way to attach many items to a host is to use a template like shown here.

5.4.1.1 Node Items

1. Hover over **Configuration**, click on **Templates,** and then click the **Items** link for the *OpenShift_Node_Template* template.

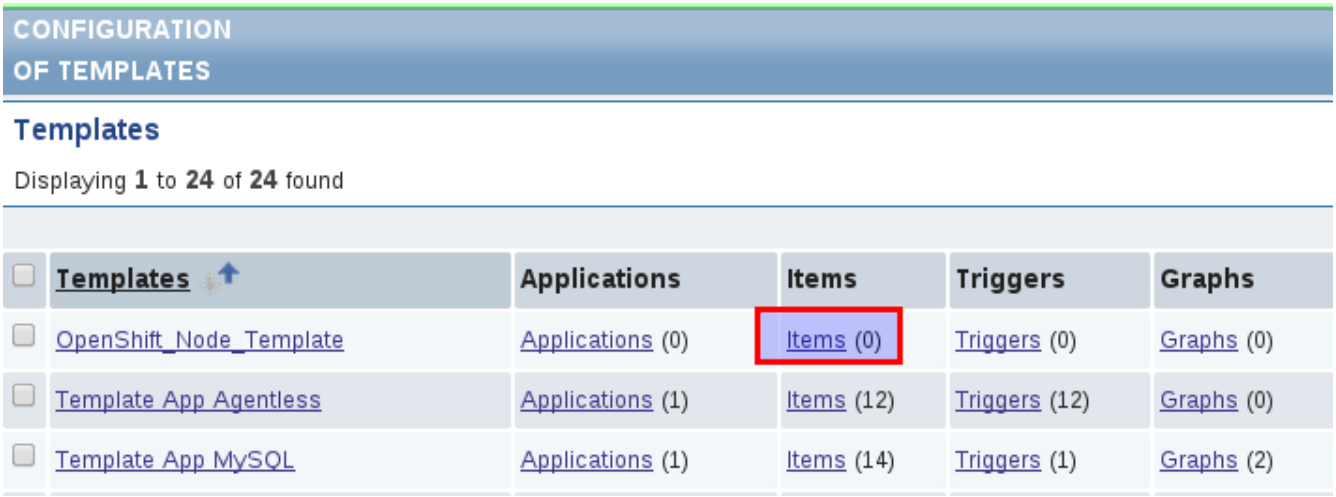


Illustration 21: Select Items

6 https://www.zabbix.com/documentation/2.0/manual/config/items

2. Click **Create Item** to add values for the Number of OpenShift Apps.

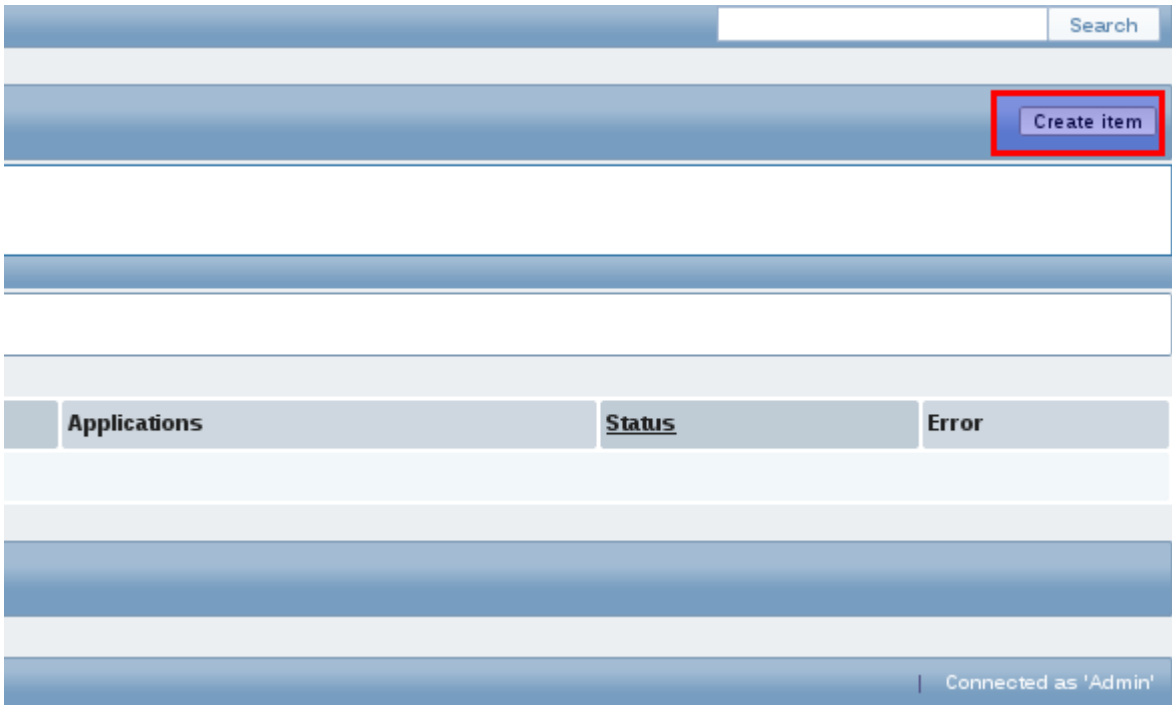


Illustration 22: Create Items

3. Provide the **Name**, **Key**, **Type of information (Character)**, create a **New Application,** and click **Save**.

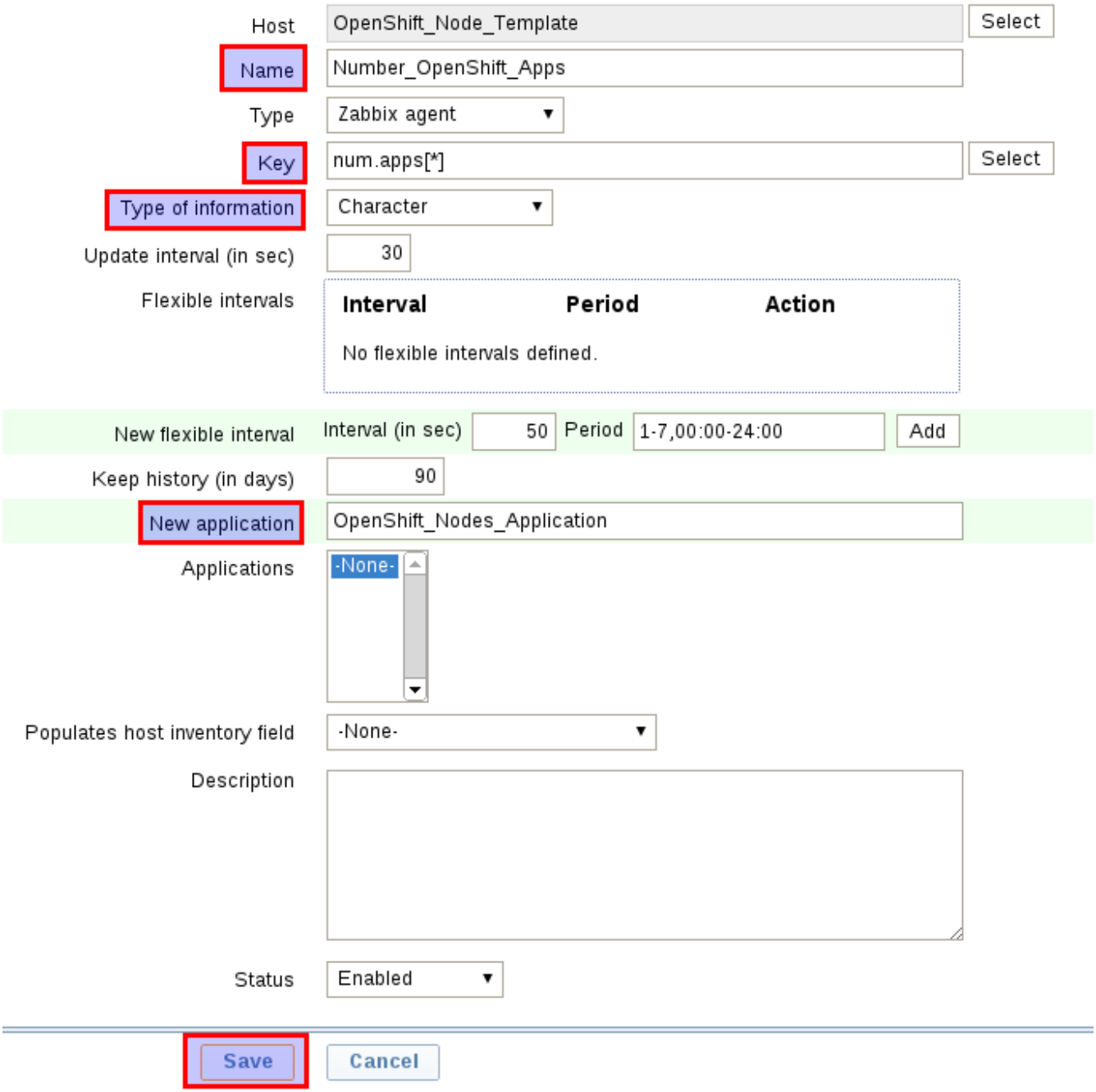


Illustration 23: Item Details

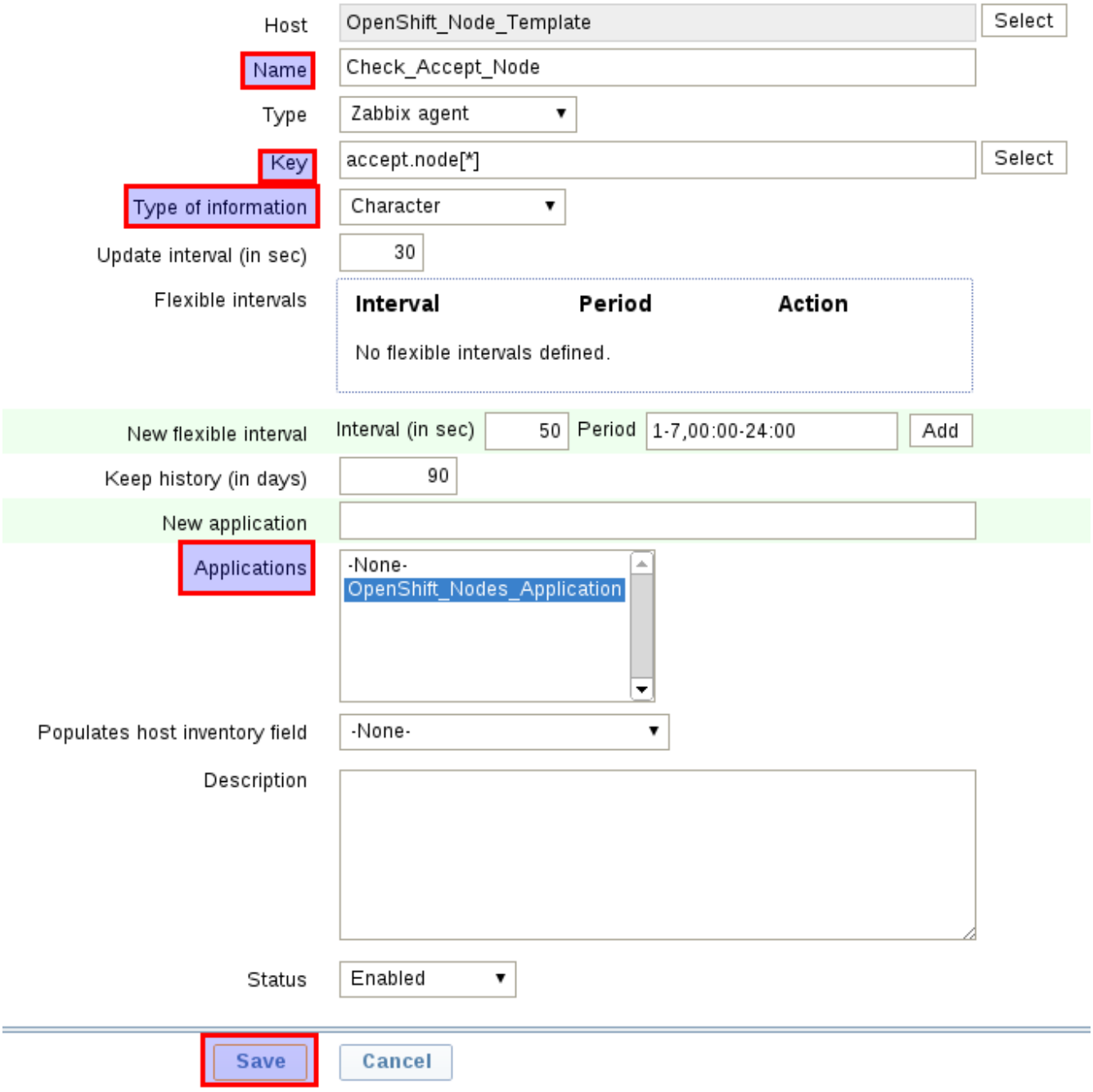


Illustration 24: Accept Node Item Details

At this point there should be one *OpenShift_Nodes_Application* with two items: *Check_Accept_Node* and *Number_OpenShift_Apps* that are all associated with the *OpenShift_Node_Template*.

5.4.1.2 Broker Items

Add items for the broker here.

1. Hover over **Configuration**, click on **Templates** and then **Items** for the *OpenShift_Broker_Template*.
2. Click **Create Item**.
3. Provide a **Host**, **Name**, **Key**, change the **Type of information,** create a new application called **OpenShift_Broker_Application,** and click **Save**.

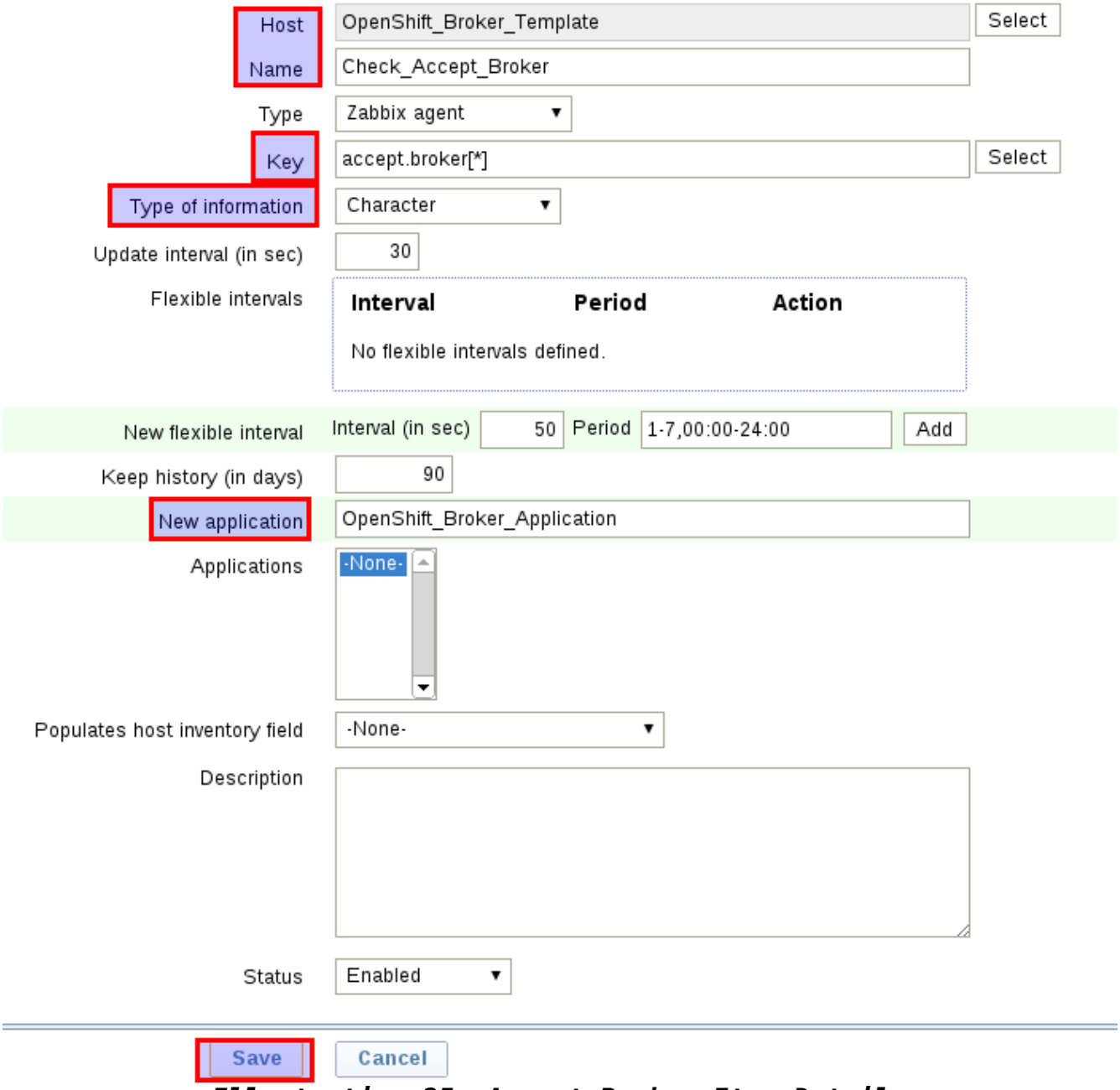


Illustration 25: Accept Broker Item Details

5.4.1.3 Broker Support Node Items

Add items for the broker support nodes.

1. Hover over **Configuration**, click on **Templates,** and then **Items** for the *OpenShift_BSN_Template*.
2. Click **Create Item**.
3. Provide a **Host**, **Name**, **Key**, change the **Type of information,** create a new application

called **OpenShift_BSN_Application** and click **Save**.

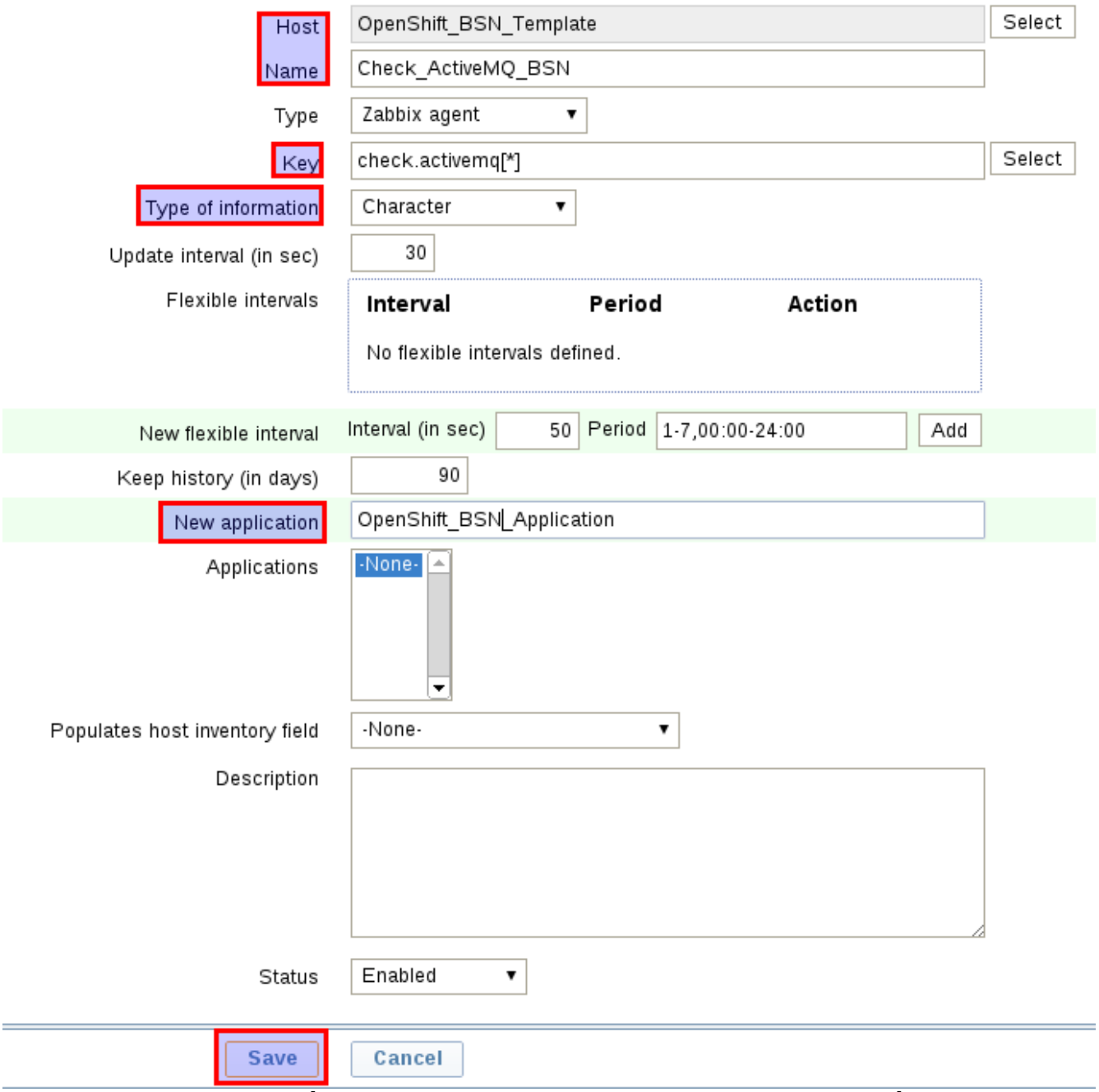


Illustration 26: Broker Support Node Item Details

5.4.1.4 Triggers

Per the Zabbix documentation, triggers are logical expressions that evaluate data gathered by items and represent the current system state. Triggers may have the following status, **OK** or **PROBLEM**. Trigger expressions can be build with varying degrees of complexity.

5.4.1.5 Node Triggers

Create Triggers for the *OpenShift_Node_Template*.

1. Hover over **Configuration** and click on **Templates.** Finally click on the **Triggers** link for the *OpenShift_Node_Template*.

Illustration 27: Create Triggers

2. Click **Create trigger**.

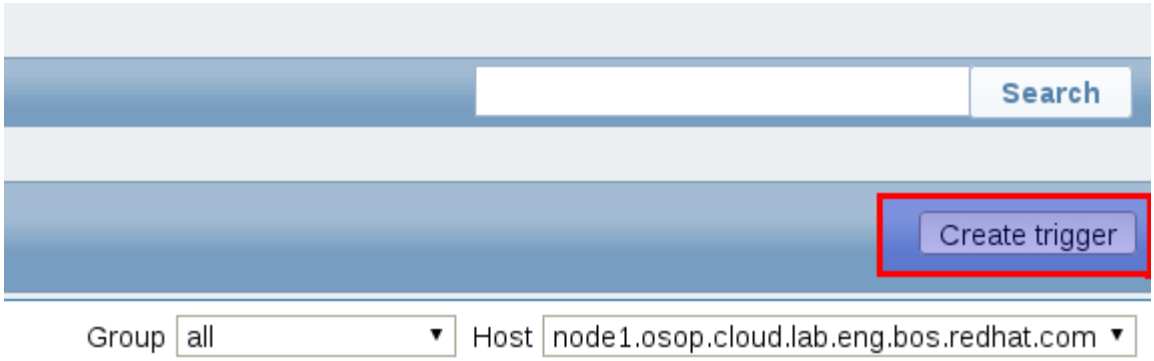


Illustration 28: Click Create Trigger

3. Provide a **Name**, **Expression,** and click **Warning**. The expression checks the *num.apps* item and trigger if the number of apps is greater than 2. This is only for demonstration purposes. The value can be changed to whatever is appropriate for the resources that are available to OpenShift Enterprise nodes. After the information is provided, click **Save**. The expression is in the format of "{Node Name: Item to Monitor.Expression}".

```
{OpenShift_Node_Template:num.apps[*].prev(0)}>2
```

Illustration 29: Node Expression

At this point there should be one application with two items and 1 trigger.

5.4.1.6 Broker Triggers

Create Triggers for the *OpenShift_Node_Template*.

1. Hover over **Configuration** and click on **Templates**, finally Click on the **Triggers** link for the OpenShift_Broker_Template. Click **Create trigger**.
2. Provide a **Name**, **Expression** and click **Warning**. The expression checks the status of the broker and whether or not **`oo-accept-broker`** passed. This is only for demonstration purposes. That value can be changed to whatever is appropriate for the resources that are available to OpenShift Enterprise nodes. After the information is provided, click **Save**. The expression is in the format of "{Node Name: Item to Monitor.Expression}".

   ```
   {OpenShift_Broker_Template:accept.broker[*].prev(0)}>0
   ```

3. Click on the **Add** button by the **Expression** field for the expression wizard.

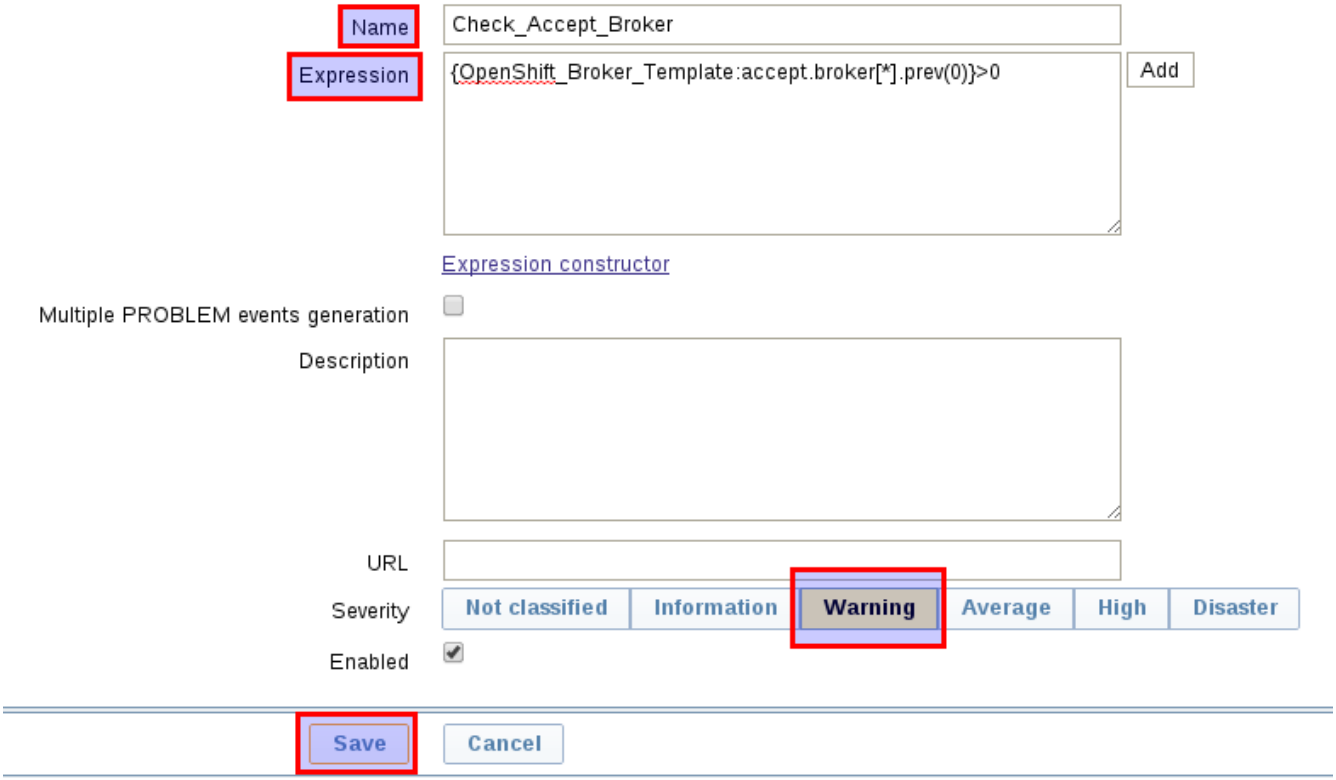


Illustration 30: Broker Expression

5.4.1.7 Broker Support Node Triggers

Create Triggers for the *OpenShift_Node_Template*.

1. Hover over **Configuration** and click on **Templates.** Finally click on the **Triggers** link for the *OpenShift_BSN_Template*. Click **Create trigger**.
2. Provide a **Name**, **Expression,** and click **Warning**. The expression checks the status of the broker support node and whether or not ActiveMQ is running. This is only for demonstration purposes. That value can be changed to whatever is appropriate for the resources that are available to OpenShift Enterprise nodes. After the information is provided, click **Save**. The expression is in the format of "{Node Name: Item to Monitor.Expression}".

```
{OpenShift_BSN_Template:check.activemq[*].prev(0)}>0
```

3. Click on the **Add** button by the **Expression** field for the expression wizard.

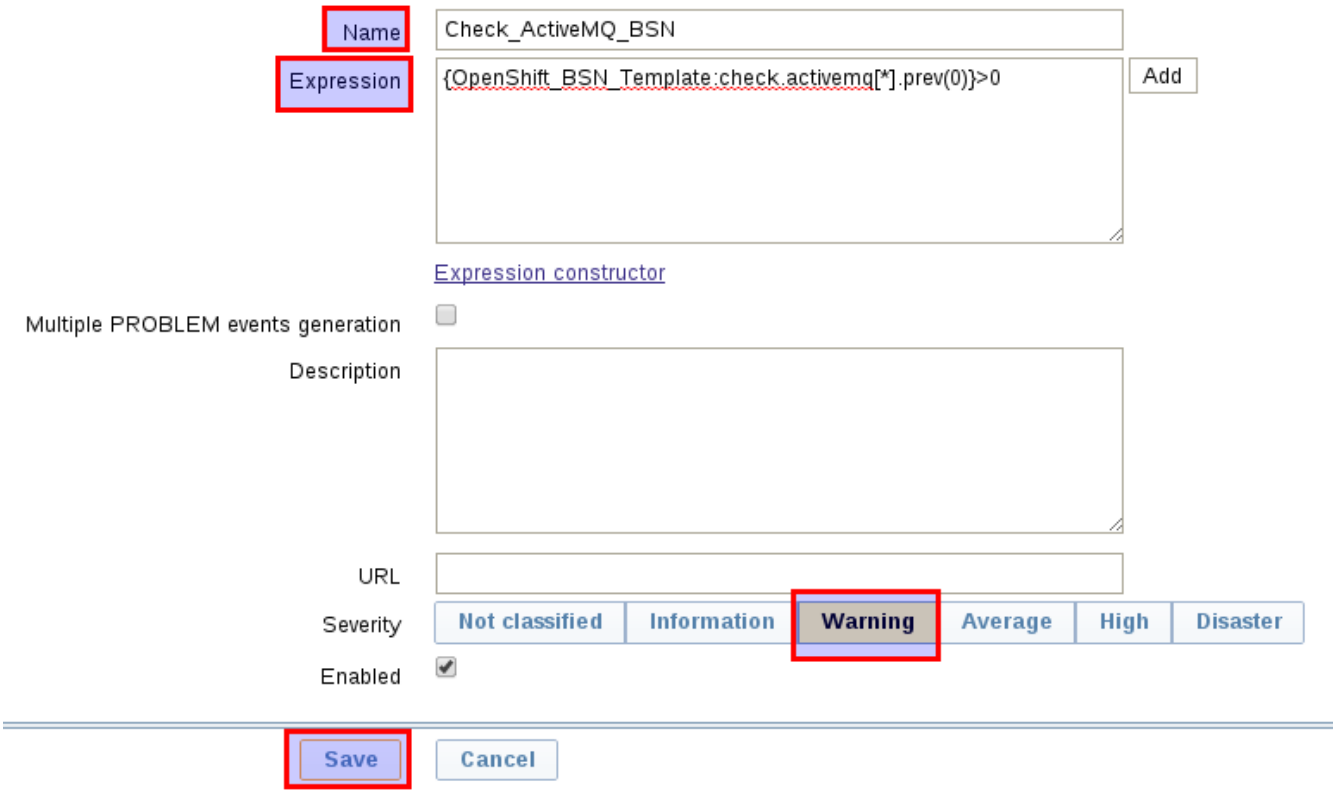


Illustration 31: BSN Expression

Configuration summary

At this point there should be

- 3 templates
- 3 Applications
- 4 Items
- 3 Triggers

5.5 Create Actions

Actions take place as the result of an event[7]. Actions can be defined for triggers, discovery, and auto-registration. The action demonstrated here is for the *To_Many_Applications* item. When too many applications are running on the node host it sends a notification to the Zabbix administrator via email.

1. Hover over **Configuration**, click **Actions,** and then click **Create action**.

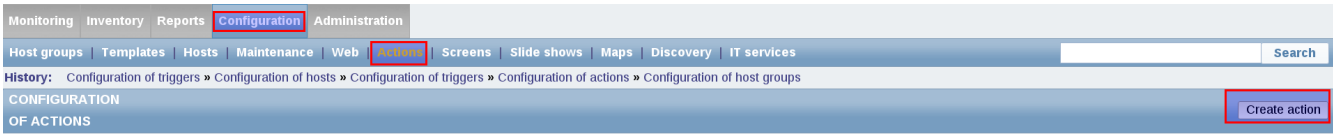


Illustration 32: Create Action

2. Click the **Action** tab and provide a **Name**

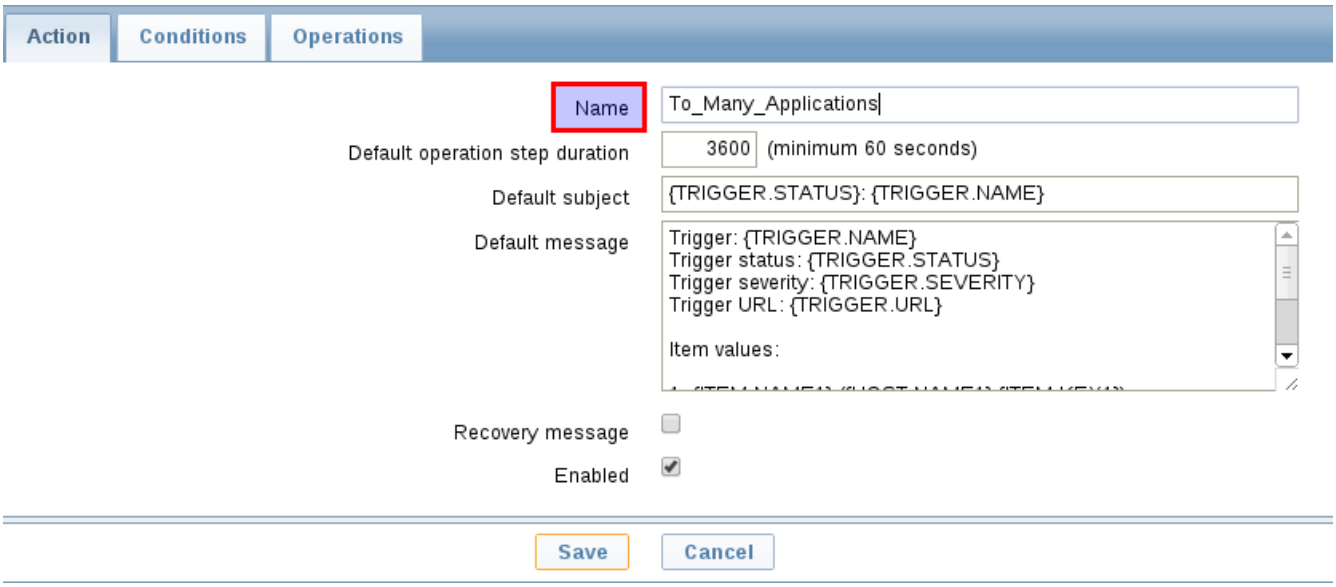


Illustration 33: Action Tab

7 https://www.zabbix.com/documentation/2.0/manual/config/notifications/action

3. Click the **Conditions** tab and remove the existing conditions A and B. Add a **New Condition** that matches the name of the trigger.

Illustration 34: Conditions

4. Click on the **Operations** tab and then click **New** to define a new operation.

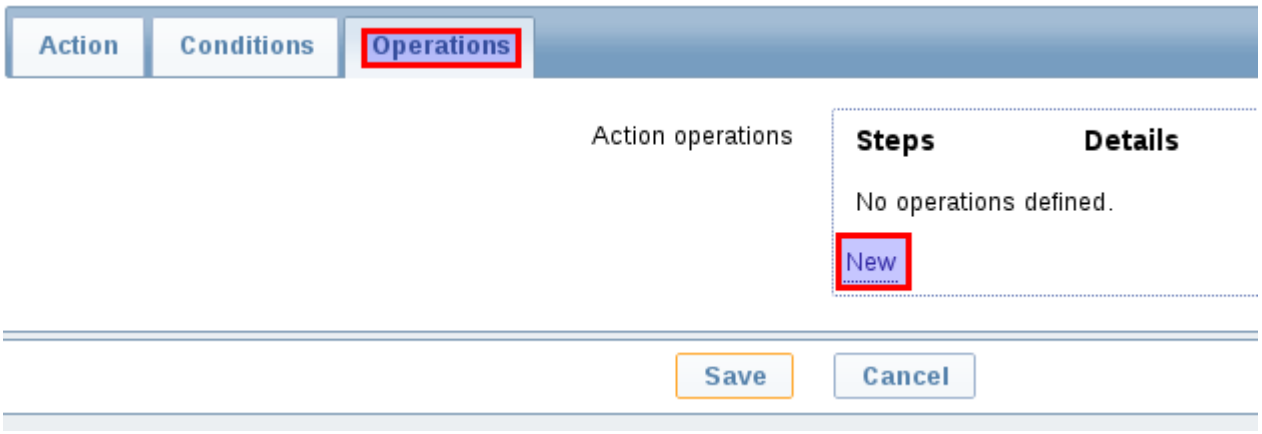
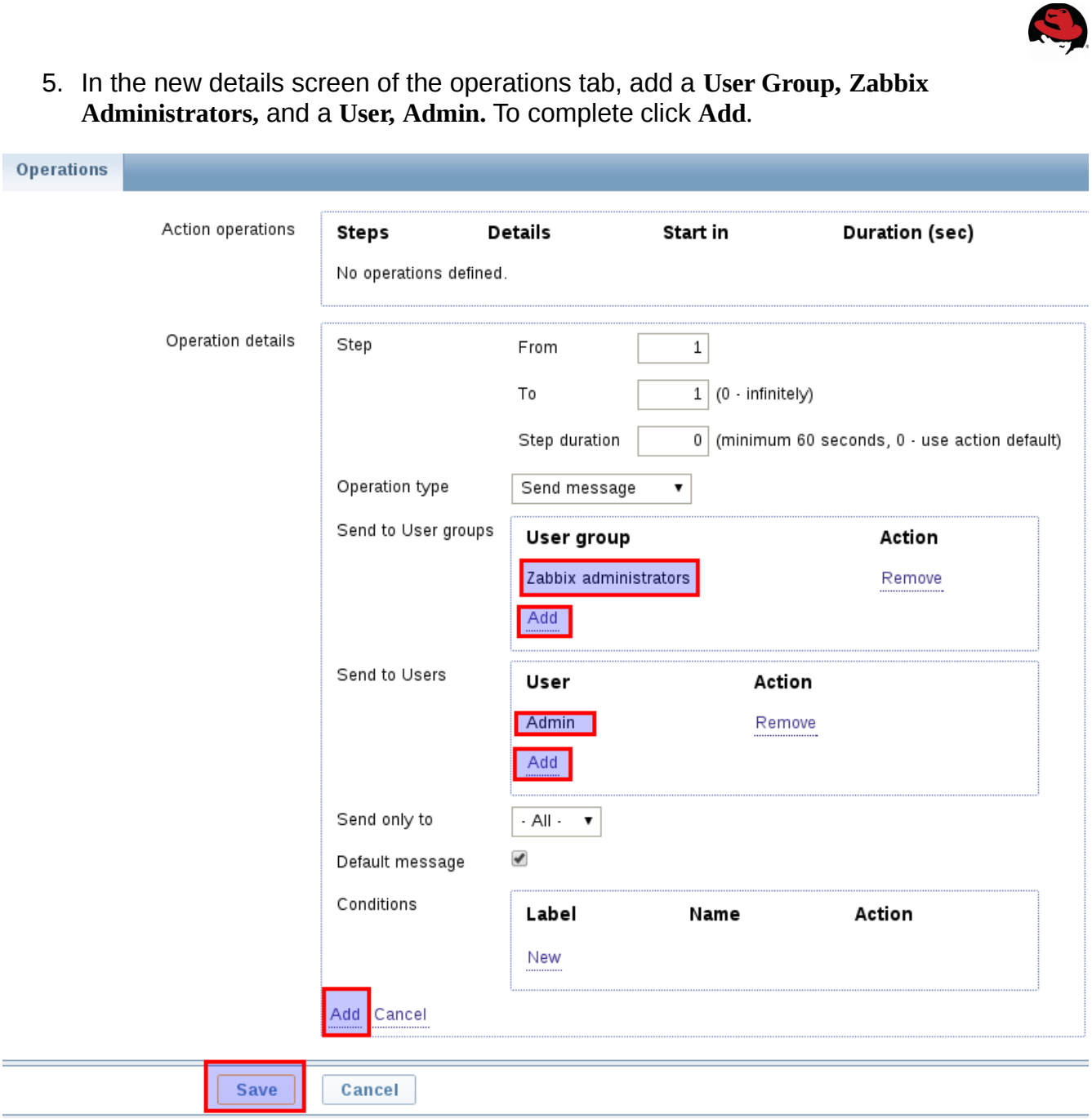


Illustration 35: Operations Tab



5. In the new details screen of the operations tab, add a **User Group, Zabbix Administrators,** and a **User, Admin.** To complete click **Add**.

Illustration 36: Action Details

6. Click **Save** on the **Action Operations** screen.

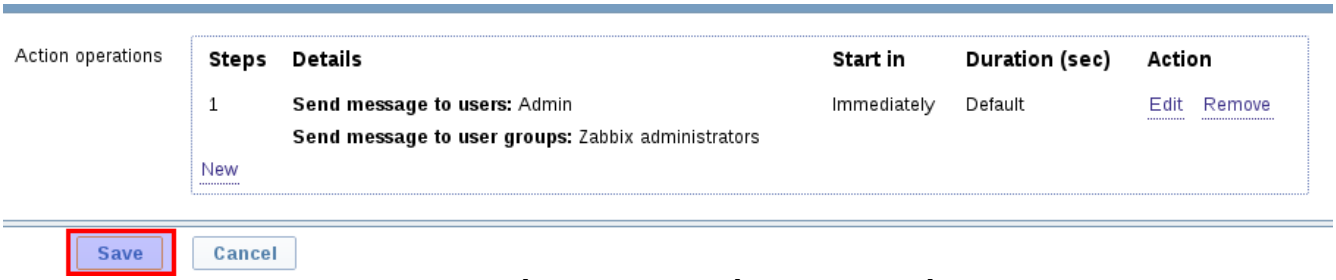


Illustration 37: Action Operations

7. Repeat these steps to create actions for the BSN and Brokers.

5.6 Templates

Associate Template to Node Host.

1. Hover over **Configuration,** click **Hosts,** and click the **node1** link.

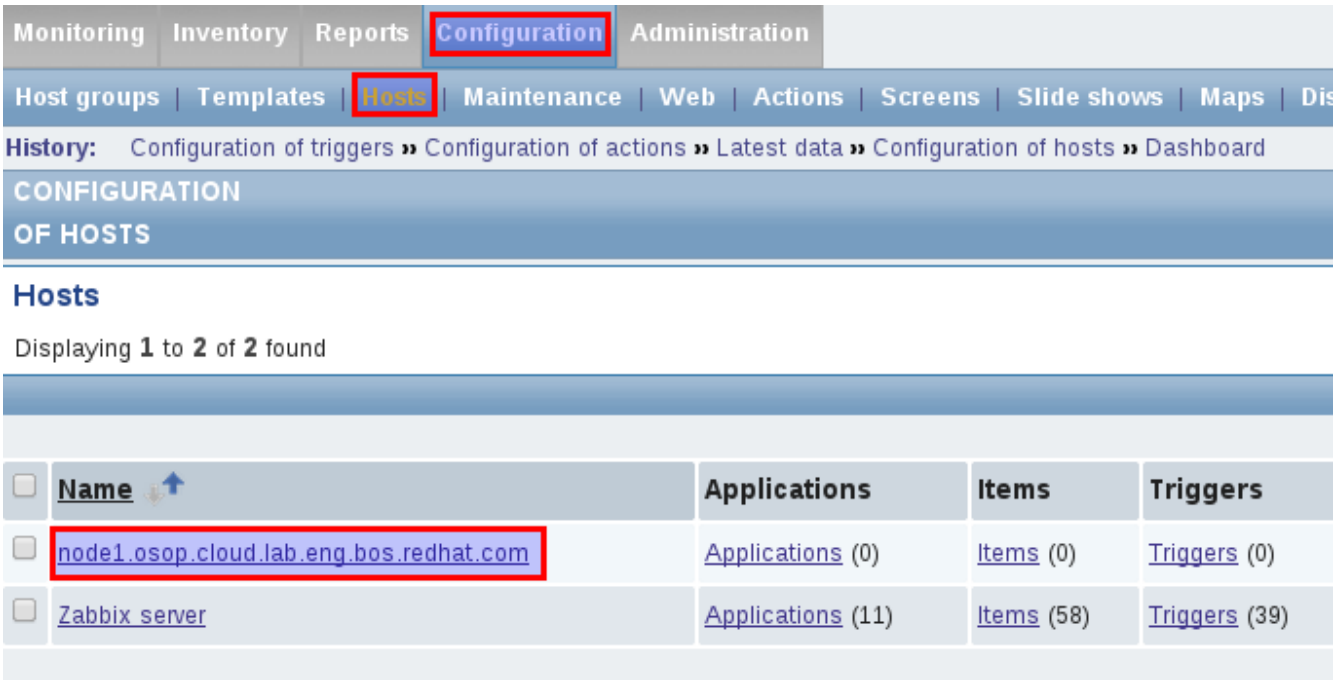


Illustration 38: Associate Templates

2. Select the **Templates** tab and click **Add**.

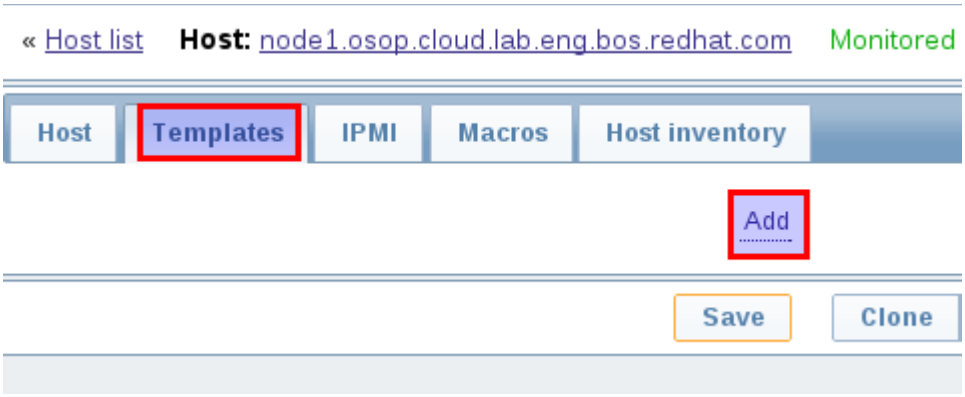


Illustration 39: Add Template

3. Change to the *OpenShift_Nodes_Host_Group* and select the *OpenShift_Node_Template*.

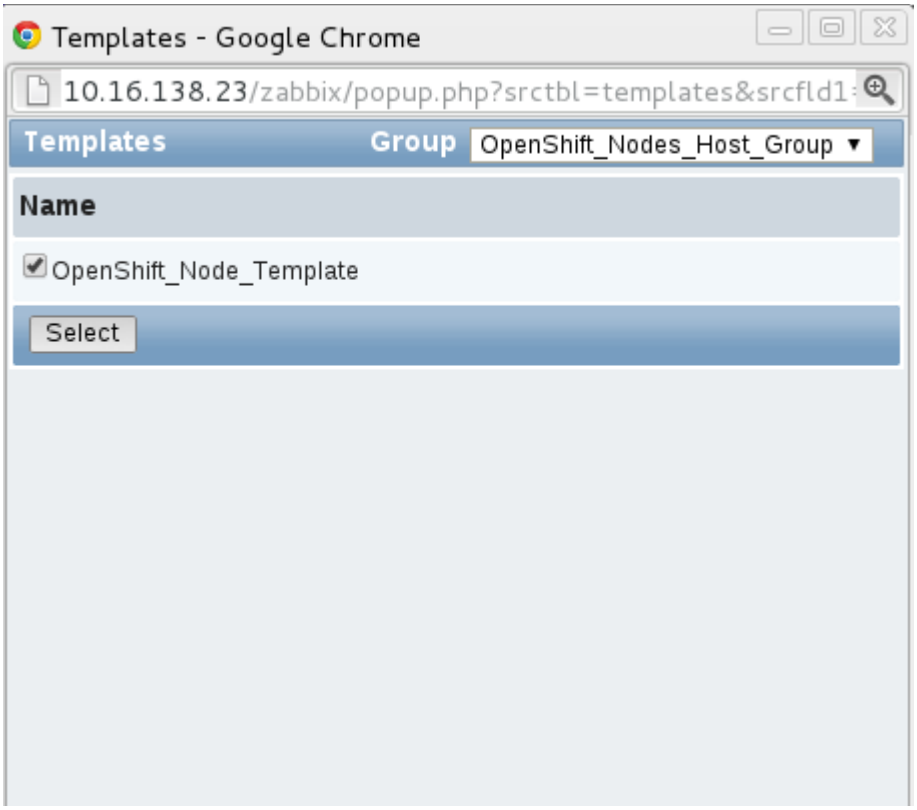


Illustration 40: Select Template

4. Click **Save**.
5. Repeat steps 1-4 to associate the remaining hosts to templates.

5.7 Zabbix Agents

At this point the OpenShift Enterprise hosts should have been added to the Zabbix database via the web interface. Now the agent needs to be deployed and configured on the hosts so that it can communicate and report values to the Zabbix server.

5.7.1 Node Agent

1. Open the port for the Zabbix agent so that the Zabbix agent can communicate with the Zabbix server.

```
# lokkit --port=10050:tcp
```

2. Configure the */etc/sudoers* file to allow the Zabbix user to execute commands on the system with the appropriate permissions. The directory */var/lib/zabbix/bin* and the scripts it contains are created a few steps later.

```
# sed -i -e 's/^Defaults.*requiretty/# &/' /etc/sudoers

# echo "zabbix ALL=NOPASSWD: /var/lib/zabbix/bin/check_number_openshift_apps
*" >> /etc/sudoers

# echo "zabbix ALL=NOPASSWD: /var/lib/zabbix/bin/check_node_accept_status *"
>> /etc/sudoers
```

3. Install the Extra Packages for Enterprise Linux (EPEL) repository so the Zabbix packages can be installed.

```
# rpm -ivh http://mirror.itc.virginia.edu/fedora-epel/6/i386/epel-release-6-
8.noarch.rpm
```

4. Install the Zabbix agent and facter.

```
# yum -y install facter zabbix20-agent.x86_64
```

5. Enable the Zabbix agent on boot.

```
# chkconfig zabbix-agent on
```

6. Configure the Zabbix agent.

```
# cp /etc/zabbix_agentd.conf{,.orig}
# sed -i 's/Server=127.0.0.1/Server=10.16.138.23/' /etc/zabbix_agentd.conf

# sed -i 's/Hostname=Zabbix
server/Hostname=node1.osop.cloud.lab.eng.bos.redhat.com/'
/etc/zabbix_agentd.conf

# sed -i 's/#
Include=\/etc\/zabbix_agentd.conf.d\//Include=\/etc\/zabbix\/zabbix_agentd.d\
//' /etc/zabbix_agentd.conf
```

```
# sed -i 's/# Timeout=3/Timeout=30/' /etc/zabbix_agentd.conf
```

These options point the Zabbix agent to the Zabbix server, configure the hostname correctly, and provide the location to the include directory that has the optional parameters in it as-well-as increases the timeout for agent operations.

7. Print changes to confirm correct configuration.

```
# grep -v -e ^# -e ^$ /etc/zabbix_agentd.conf
PidFile=/var/run/zabbix/zabbix_agentd.pid
LogFile=/var/log/zabbix/zabbix_agentd.log
LogFileSize=0
DebugLevel=4
Server=10.16.138.23
ServerActive=127.0.0.1
Hostname=node1.osop.cloud.lab.eng.bos.redhat.com
Timeout=30
Include=/etc/zabbix/zabbix_agentd.d/
```

8. Create the directory for the configuration file. The file that goes in this directory is called upon by the Zabbix agent because of the *INCLUDE* directive in the */etc/zabbix_agentd.conf* file. The file is called *node.conf*.

```
# mkdir /etc/zabbix/zabbix_agentd.d
```

9. Add the number of *APPLICATION CHECKS* and the *ACCEPT NODE* checks to the *node.conf* file.

```
# cat << EOF >> /etc/zabbix/zabbix_agentd.d/node.conf
UserParameter=accept.node[*],sudo
/var/lib/zabbix/bin/check_node_accept_status
UserParameter=num.apps[*],sudo
/var/lib/zabbix/bin/check_number_openshift_apps
> EOF
```

These parameters match what was provided for the items in the Zabbix web user interface during item creation. At this point we are providing the items and a location for the scripts that are run to gather the data.

10. Create the directory for the scripts.

```
# mkdir -p /var/lib/zabbix/bin
```

11. Create the */var/lib/zabbix/bin/check_number_openshift_apps* script by adding the following contents. This script uses the **`oo-idler-stats`** tool to pull some information from the node and then the script parses that information and finally returns a value that Zabbix can consume via the *ECHO* statement.

    The script returns a value of the number of applications on the node.

```
#!/bin/bash
```

```
NUMAPPS=$(oo-idler-stats | awk '{ print $1 }')

if [[ "$NUMAPPS" == "OK:" ]]; then
  echo "0"
else
  echo $NUMAPPS
fi
```

12. Create the */var/lib/zabbix/bin/check_node_accept_status* script by adding the following contents. This script uses the **oo-accept-node** utility to send a 0 if the system passes the check and a 1 if the system fails the check. This can be graphed over time to determine when the node failed the **oo-accept-node** tool.

```
#!/bin/bash

RESULT=$(/usr/sbin/oo-accept-node &> /dev/null)

echo $?
```

13. Mark the scripts as executable.

```
# chmod +x /var/lib/zabbix/bin/check_node_accept_status
# chmod +x /var/lib/zabbix/bin/check_number_openshift_apps
```

14. Run the scripts manually to ensure the proper values are being returned. These values are passed to Zabbix.

```
# ./check_node_accept_status
0

# ./check_number_openshift_apps
0
```

15. Restart the Zabbix server process.

```
# service zabbix-server restart
Shutting down Zabbix server:                               [  OK  ]
Starting Zabbix server:                                       [  OK  ]
```

16. Restart the agent on *node1*.

```
# service zabbix-agent restart
Shutting down Zabbix agent:                                   [FAILED]
Starting Zabbix agent:                                        [  OK  ]
```

Confirm items are being updated in interface

1. Click **Monitoring** | **Latest Data** expand node1 and view the **Last Value**. This process can take up to 1-2 minutes. If you change to the latest data screen and the node is not listed there, check the **Show items without data** and then click **Filter**. The values will be displayed even though the monitoring hasn't taken place yet.

Name	Last check	Last value
OpenShift_Nodes (2 Items)		
Check_Accept_Node	20 Mar 2013 00:14:50	0
Number_OpenShift_Apps	20 Mar 2013 00:15:18	0

The **Last value** should be 0 for the **Check_Accept_Node** item until the node fails the **oo-accept-node** test and the **Number_OpenShift_Apps** remains 0 until applications are created and added to the node. This confirms the updates are working.

5.7.2 Broker Agent

1. Open the port for the Zabbix agent so that the Zabbix agent can communicate with the Zabbix server.

```
# lokkit --port=10050:tcp
```

2. Configure the */etc/sudoers* file to allow the Zabbix user to execute commands on the system with the appropriate permissions. The directory */var/lib/zabbix/bin* and the scripts it contains are created a few steps later.

```
# sed -i -e 's/^Defaults.*requiretty/# &/' /etc/sudoers

# echo "zabbix ALL=NOPASSWD:
/var/lib/zabbix/bin/check_broker_accept_status *" >> /etc/sudoers
```

3. Install the Extra Packages for Enterprise Linux (EPEL) repository so the Zabbix packages can be installed.

```
# rpm -ivh http://mirror.itc.virginia.edu/fedora-epel/6/i386/epel-
release-6-8.noarch.rpm
```

4. Install the Zabbix agent and facter.

```
# yum -y install facter zabbix20-agent.x86_64
```

5. Enable the Zabbix agent on boot.

```
# chkconfig zabbix-agent on
```

6. Configure the Zabbix agent.

```
# cp /etc/zabbix_agentd.conf{,.orig}
# sed -i 's/Server=127.0.0.1/Server=10.16.138.23/'
/etc/zabbix_agentd.conf

# sed -i 's/Hostname=Zabbix
server/Hostname=broker1.osop.cloud.lab.eng.bos.redhat.com/'
/etc/zabbix_agentd.conf

# sed -i 's/#
Include=\/etc\/zabbix_agentd.conf.d\//Include=\/etc\/zabbix\/zabbix_agent
d.d\//' /etc/zabbix_agentd.conf

# sed -i 's/# Timeout=3/Timeout=30/' /etc/zabbix_agentd.conf
```

7. Print changes to confirm correct configuration.

```
# grep -v -e ^# -e ^$ /etc/zabbix_agentd.conf
PidFile=/var/run/zabbix/zabbix_agentd.pid
LogFile=/var/log/zabbix/zabbix_agentd.log
LogFileSize=0
DebugLevel=4
Server=10.16.138.23
ServerActive=127.0.0.1
Hostname=broker1.osop.cloud.lab.eng.bos.redhat.com
Timeout=30
Include=/etc/zabbix/zabbix_agentd.d/
```

8. Create the directory for the configuration file. The file that goes in this directory is called upon by the Zabbix agent because of the *INCLUDE* directive in the */etc/zabbix_agentd.conf* file. The file is called *node.conf*.

```
mkdir /etc/zabbix/zabbix_agentd.d
```

9. Add the number of *APPLICATION CHECKS* and the *ACCEPT NODE* checks to the *node.conf* file

```
# cat << EOF >> /etc/zabbix/zabbix_agentd.d/broker.conf
UserParameter=accept.broker[*],sudo
/var/lib/zabbix/bin/check_broker_accept_status
> EOF
```

10. Create the directory for the scripts.

```
# mkdir -p /var/lib/zabbix/bin
```

11. Create the */var/lib/zabbix/bin/check_broker_accept_status* script by adding the following contents. This script uses the **`oo-accept-broker`** tool to pull some information from the node and then the script parses that information and finally returns a value that Zabbix can consume via the *ECHO* statement.

The script returns a value of the number of applications on the node.

```
#!/bin/bash

RESULT=$(/usr/sbin/oo-accept-broker &> /dev/null)

echo $?
```

12. Run the scripts manually to ensure the proper values are being returned. These return values are passed to Zabbix.

```
# ./check_broker_accept_status
0
```

13. Start the Zabbix agent on node1.

```
# service zabbix-agent restart
Shutting down Zabbix agent:                                [FAILED]
Starting Zabbix agent:                                     [  OK  ]
```

14. Restart the Zabbix server process.

```
# service zabbix-server restart
Shutting down Zabbix server:                               [  OK  ]
Starting Zabbix server:                                    [  OK  ]
```

5.7.3 Broker Support Node Agent

1. Open the port for the Zabbix agent so that the Zabbix agent can communicate with the Zabbix server.

```
# lokkit --port=10050:tcp
```

2. Configure the */etc/sudoers* file to allow the Zabbix user to execute commands on the system with the appropriate permissions. The directory */var/lib/zabbix/bin* and the scripts it contains are created a few steps later.

```
# sed -i -e 's/^Defaults.*requiretty/# &/' /etc/sudoers

# echo "zabbix ALL=NOPASSWD: /var/lib/zabbix/bin/check_activemq *" >>
/etc/sudoers
```

3. Install the Extra Packages for Enterprise Linux (EPEL) repository so the Zabbix packages can be installed.

```
# rpm -ivh http://mirror.itc.virginia.edu/fedora-epel/6/i386/epel-
release-6-8.noarch.rpm
```

4. Install the Zabbix agent and facter.

```
# yum install facter zabbix20-agent.x86_64
```

5. Enable the Zabbix agent on boot.

```
# chkconfig zabbix-agent on
```

6. Configure the Zabbix agent.

```
# cp /etc/zabbix_agentd.conf{,.orig}
# sed -i 's/Server=127.0.0.1/Server=10.16.138.23/'
/etc/zabbix_agentd.conf

# sed -i 's/Hostname=Zabbix
server/Hostname=node1.osop.cloud.lab.eng.bos.redhat.com/'
/etc/zabbix_agentd.conf

# sed -i 's/#
Include=\/etc\/zabbix_agentd.conf.d\//Include=\/etc\/zabbix\/zabbix_agent
d.d\//' /etc/zabbix_agentd.conf

# sed -i 's/# Timeout=3/Timeout=30/' /etc/zabbix_agentd.conf
```

7. Print changes to confirm correct configuration.

```
# grep -v -e ^# -e ^$ /etc/zabbix_agentd.conf
PidFile=/var/run/zabbix/zabbix_agentd.pid
LogFile=/var/log/zabbix/zabbix_agentd.log
LogFileSize=0
DebugLevel=4
Server=10.16.138.23
ServerActive=127.0.0.1
Hostname=node1.osop.cloud.lab.eng.bos.redhat.com
Timeout=30
Include=/etc/zabbix/zabbix_agentd.d/
```

8. Create the directory for the configuration file. The file that goes in this directory is called upon by the Zabbix agent because of the *INCLUDE* directive in the */etc/zabbix_agentd.conf* file. The file is called *bsn.conf.*

```
# mkdir /etc/zabbix/zabbix_agentd.d
```

9. Add the number of *APPLICATION CHECKS* and the *ACCEPT NODE* checks to the *bsn.conf* file.

```
# cat << EOF >> /etc/zabbix/zabbix_agentd.d/bsn.conf
UserParameter=check.activemq[*],sudo /var/lib/zabbix/bin/check_activemq
> EOF
```

10. Create the directory for the scripts.

```
# mkdir -p /var/lib/zabbix/bin
```

11. Create the */var/lib/zabbix/bin/check_activemq* script by adding the following contents. This script checks for the existence of the ActiveMQ lock file and returns a value that Zabbix can consume via the *ECHO* statement.

The script returns a value of the number of applications on the node.

```
#!/bin/bash

if [[ -f "/var/lock/subsys/ActiveMQ" ]]; then
      echo 0
else
      echo 1
fi
```

12. Run the scripts manually to ensure the proper values are being returned. These values are passed to Zabbix.

```
# /var/lib/zabbix/bin/check_activemq
0
```

13. Restart the Zabbix server process.

```
# service zabbix-server restart
Shutting down Zabbix server:                               [  OK  ]
Starting Zabbix server:                                       [  OK  ]
```

14. Restart the agent on *bsn1*.

```
# service zabbix-agent restart
Shutting down Zabbix agent:                                [FAILED]
Starting Zabbix agent:                                        [  OK  ]
```

Repeat these steps for BSN{2,3} changing the hostname when necessary.

5.8 Confirm Email Notifications are Sent

This section uses the node configuration as an example. Zabbix should send an email alert to the Administrators group as well as the Zabbix user.

1. Create a few applications with the rhc tool and then go to **Monitoring | Latest data** in the Zabbix interface. Once the value for running applications exceeds 2, an email should be sent. Run the following command as a OpenShift Enterprise user on a client machine.

```
$ for i in `seq 1 4`; do rhc app create test$i -t php-5.3; done
```

Name	Last check	Last value
OpenShift_Nodes (2 Items)		
Check_Accept_Node	20 Mar 2013 21:56:50	0
Number_OpenShift_Apps	20 Mar 2013 21:57:19	4

Illustration 41: Confirm Operations

2. The notification email content appears as:

```
Trigger: To_Many_Applications
Trigger status: PROBLEM
Trigger severity: Warning
Trigger URL:

Item values:

1. Number_OpenShift_Apps
(node1.osop.cloud.lab.eng.bos.redhat.com:num.apps[*]): 3
2. *UNKNOWN* (*UNKNOWN*:*UNKNOWN*): *UNKNOWN*
3. *UNKNOWN* (*UNKNOWN*:*UNKNOWN*): *UNKNOWN*
```

6 Conclusion

Providing an OpenShift Enterprise infrastructure is key to helping developers stay productive and focus on mission-critical metrics. Part of that story is deploying the infrastructure in a distributed environment and monitoring that environment for availability.

Monitoring should be part of any systems administrator's tool-set to help keep an eye on the environment. This reference architecture was written with two audiences in mind. First, for customers who need a specific example of monitoring with an enterprise class tool that is freely available. Second, customers who have existing tool-sets who need examples of what to monitor.

With the examples provided, it is expected that a systems administrator can successfully monitor his or her OpenShift Enterprise infrastructure for both performance and availability.

Appendix A: Contributors

Contributor	Title	Contribution
Matt Hicks	Director of Software Engineering	Review
Joe Fernandes	Senior Product Marketing Manager	Review
Brett Thurber	Principal Software Engineer	Review

Table 6.1: Contributors

Appendix B: Configuration Files

All configuration files can be downloaded from the Red Hat customer portal[8]. In addition to the configuration files, Zabbix XML files for the host and templates are available for import. Make sure to substitute your information for what is in the XML files. The following files are included:

8 https://access.redhat.com/site/node/344703/40/1

Node	Files	Description
broker{1,2,3}	/etc/zabbix_agentd.conf	Provides agent directives
	/etc/sudoers	Allows proper access to run scripts
	/ var/lib/zabbix/bin/check_broker_accept_status	Script to check metrics on broker
	/etc/zabbix/zabbix_agentd.d/broker.conf	Zabbix external configuration file
	/etc/sysconfig/iptables	Firewall configuration file
bsn{1,2,3}	/etc/zabbix_agentd.conf	Provides Agent directives
	/etc/sudoers	Allows proper access to run scripts
	/var/lib/zabbix/bin/check_activemq	Scripts to check metrics on bsn hosts
	/etc/zabbix/zabbix_agentd.d/bsn.conf	Zabbix external configuration file
	/etc/sysconfig/iptables	Firewall configuration
node{1,2,3}	/etc/zabbix_agentd.conf	Provides agent directives
	/etc/sudoers	Allows proper access to run scripts
	/ var/lib/zabbix/bin/check_node_accept_status	Scripts to check metrics on bsn hosts
	/ var/lib/zabbix/bin/check_number_openshift_apps	Zabbix external configuration file
	/etc/sysconfig/iptables	Firewall configuration
zabbix	/etc/zabbix_server.conf	Zabbix server configuration file
	/etc/php.ini	PHP configuration file
	/var/lib/pgsql/postgresql.conf	PostgreSQL configuration File
	/var/lib/pgsql/pg_hba.conf	PostgreSQL configuration file

	/etc/sysconfig/iptables	Firewall configuration file
	postgres_commands	PostgreSQL script to configure database
Zabbix XML	zabbix_export_templates.xml	These XML files can be used as examples and imported into Zabbix for reference
	zabbix_export_hosts.xml	

Table 6.2: Configuration Files

Appendix C: Zabbix Log Files

The log files for the Zabbix server are located in:

/var/log/zabbix/zabbix_server.log

Enable debug logging on the Zabbix server by changing the DebugLevel=3 to DebugLevel=4 in the */etc/zabbix_agentd.conf* configuration file.

The log files for the Zabbix agent are located in:

/var/log/zabbix/zabbix_agentd.log

Enable debug logging on the Zabbix server by changing the DebugLevel=3 to DebugLevel=4 in the */etc/zabbix_agentd.conf* configuration file.

Appendix D: Revision History

Revision 1.0	April 2013	Scott Collier
Initial Release		

Table 6.3: Revision History