Monitoring the health of your cluster

In this article

About GitHub Enterprise Server cluster health

Manually checking cluster status

Monitoring cluster status with Nagios

To ensure the performance and redundancy of a GitHub Enterprise Server cluster, you can monitor the cluster's health.

GitHub determines eligibility for clustering, and must enable the configuration for your instance's license. Clustering requires careful planning and additional administrative overhead. For more information, see "About clustering."

About GitHub Enterprise Server cluster health

A GitHub Enterprise Server cluster comprises multiple nodes, with redundant services distributed across two or more nodes. If an individual service or an entire node fails, users should not notice. Failures affect performance and redundancy, so it's important to monitor the health of your cluster. You can monitor the health of your cluster using a command-line utility or an external monitoring tool like Nagios.

You can also monitor the health of individual nodes using Node Eligibility Service. For more information, see "Monitoring the health of your cluster nodes with Node Eligibility Service."

Manually checking cluster status

GitHub Enterprise Server has a built-in command line utility for monitoring the health of the cluster. From the administrative shell, running the `ghe-cluster-status` command executes a series of health checks on each node including verification of connectivity and service status. The output shows all test results including the text `ok` or `error`. For example, to only display failing tests, run:

```
admin@ghe-data-node-0:~$ ghe-cluster-status | grep error
> mysql-replication ghe-data-node-0: error Stopped
> mysql cluster: error
```

Note: If there are no failing tests, this command produces no output. This indicates the cluster is healthy.

Monitoring cluster status with Nagios

You can configure Nagios to monitor GitHub Enterprise Server. In addition to monitoring basic connectivity to each of the cluster nodes, you can check the cluster status by

configuring Nagios to use the `ghe-cluster-status -n` command. This returns output in a format that Nagios understands.

Prerequisites

- Linux host running Nagios.
- Network access to the GitHub Enterprise Server cluster.

Configuring the Nagios host

1. Generate an SSH key with a blank passphrase. Nagios uses this to authenticate to the GitHub Enterprise Server cluster.

   ```
   nagiosuser@nagios:~$ ssh-keygen -t ed25519
   > Generating public/private ed25519 key pair.
   > Enter file in which to save the key (/home/nagiosuser/.ssh/id_ed25519):
   > Enter passphrase (empty for no passphrase): LEAVE BLANK BY PRESSING ENTER
   > Enter same passphrase again: PRESS ENTER AGAIN
   > Your identification has been saved in /home/nagiosuser/.ssh/id_ed25519.
   > Your public key has been saved in /home/nagiosuser/.ssh/id_ed25519.pub.
   ```

   Security Warning: An SSH key without a passphrase can pose a security risk if authorized for full access to a host. Limit this key's authorization to a single read-only command.

   Note: If you're using a distribution of Linux that doesn't support the Ed25519 algorithm, use the command:

   ```
   nagiosuser@nagios:~$ ssh-keygen -t rsa -b 4096
   ```

2. Copy the private key (`id_ed25519`) to the `nagios` home folder and set the appropriate ownership.

   ```
   nagiosuser@nagios:~$ sudo cp .ssh/id_ed25519 /var/lib/nagios/.ssh/
   nagiosuser@nagios:~$ sudo chown nagios:nagios
   /var/lib/nagios/.ssh/id_ed25519
   ```

3. To authorize the public key to run *only* the `ghe-cluster-status -n` command, use a `command=` prefix in the `/data/user/common/authorized_keys` file. From the administrative shell on any node, modify this file to add the public key generated in step 1. For example: `command="/usr/local/bin/ghe-cluster-status -n" ssh-ed25519 AAAA....`

4. Validate and copy the configuration to each node in the cluster by running `ghe-cluster-config-apply` on the node where you modified the `/data/user/common/authorized_keys` file.

   ```
   admin@ghe-data-node-0:~$ ghe-cluster-config-apply
   > Validating configuration
   > ...
   > Finished cluster configuration
   ```

5. To test that the Nagios plugin can successfully execute the command, run it interactively from Nagios host.

   ```
   nagiosuser@nagios:~$ /usr/lib/nagios/plugins/check_by_ssh -l admin -p 122 -H
   HOSTNAME -C "ghe-cluster-status -n" -t 30
   ```

```
> OK - No errors detected
```

6. Create a command definition in your Nagios configuration.

   Example definition

```
define command {
     command_name    check_ssh_ghe_cluster
     command_line    $USER1$/check_by_ssh -H $HOSTADDRESS$ -C "ghe-cluster-
status -n" -l admin -p 122 -t 30
}
```

7. Add this command to a service definition for a node in the GitHub Enterprise Server cluster.

   Example definition

```
define host{
     use                     generic-host
     host_name               ghe-data-node-0
     alias                   ghe-data-node-0
     address                 10.11.17.180
     }

define service{
       use                             generic-service
       host_name                       ghe-data-node-0
       service_description             GitHub Cluster Status
       check_command                   check_ssh_ghe_cluster
       }
```

After you add the definition to Nagios, the service check executes according to your configuration. You should be able to see the newly configured service in the Nagios web interface.