Replacing a cluster node

In this article

If a node fails in a GitHub Enterprise Server cluster, or if you want to add a new node with more resources, mark any nodes to replace as offline, then add the new node.

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 replacement of GitHub Enterprise Server cluster nodes

You can replace a functional node in a GitHub Enterprise Server cluster, or you can replace a node that has failed unexpectedly.

After you replace a node, your GitHub Enterprise Server instance does not automatically distribute jobs to the new node. You can force your instance to balance jobs across nodes. For more information, see "Rebalancing cluster workloads."

Warning: To avoid conflicts, do not reuse a hostname that was previously assigned to a node in the cluster.

Replacing a functional node

You can replace an existing, functional node in your cluster. For example, you may want to provide a virtual machine (VM) with additional CPU, memory, or storage resources.

To replace a functional node, install the GitHub Enterprise Server appliance on a new VM, configure an IP address, add the new node to the cluster configuration file, initialize the cluster and apply the configuration, then take the node you replaced offline.

1. Provision and install GitHub Enterprise Server with a unique hostname on the replacement node.
2. Using the administrative shell or DHCP, **only** configure the IP address of the replacement node. Don't configure any other settings.
3. To add the newly provisioned replacement node, on any node, modify the `cluster.conf` file to remove the failed node and add the replacement node. For example, this modified `cluster.conf` file replaces `ghe-data-node-3` with the newly

provisioned node, `ghe-replacement-data-node-3`:

```
[cluster "ghe-replacement-data-node-3"]
  hostname = ghe-replacement-data-node-3
  ipv4 = 192.168.0.7
  # ipv6 = fd12:3456:789a:1::7
  git-server = true
  pages-server = true
  mysql-server = true
  elasticsearch-server = true
  redis-server = true
  memcache-server = true
  metrics-server = true
  storage-server = true
```

4. From the administrative shell of the node with the modified `cluster.conf`, run `ghe-cluster-config-init`. This will initialize the newly added node in the cluster.
5. From the same node, run `ghe-cluster-config-apply`. This will validate the configuration file, copy it to each node in the cluster, and configure each node according to the modified `cluster.conf` file.
6. If you're taking a node offline that provides data services, such as `git-server`, `pages-server`, or `storage-server`, evacuate the node. For more information, see "Evacuating a cluster node running data services."
7. To mark the failed node offline, on any node, modify the cluster configuration file (`cluster.conf`) in the relevant node section to include the text `offline = true`.

   For example, this modified `cluster.conf` will mark the `ghe-data-node-3` node as offline:

   ```
   [cluster "ghe-data-node-3"]
   hostname = ghe-data-node-3
   offline = true
   ipv4 = 192.168.0.6
   # ipv6 = fd12:3456:789a:1::6
   ```

8. From the administrative shell of the node where you modified `cluster.conf`, run `ghe-cluster-config-apply`. This will validate the configuration file, copy it to each node in the cluster, and mark the node offline.
9. If you're replacing the primary MySQL or Redis node, in `cluster.conf`, modify the `mysql-master` or `redis-master` value with the replacement node name.

   For example, this modified `cluster.conf` file specifies a newly provisioned cluster node, `ghe-replacement-data-node-1` as the primary MySQL and Redis node:

   ```
   mysql-master = ghe-replacement-data-node-1
   redis-master = ghe-replacement-data-node-1
   ```

Replacing a node in an emergency

You can replace a failed node in your cluster. For example, a software or hardware issue may affect a node's availability.

To replace a node in an emergency, install the GitHub Enterprise Server appliance on a new VM, configure an IP address, take the failed node offline, apply the configuration,

add the new node to the cluster configuration file, initialize the cluster and apply the configuration, and optionally, evacuate the failed node.

1. Provision and install GitHub Enterprise Server with a unique hostname on the replacement node.

2. Using the administrative shell or DHCP, **only** configure the IP address of the replacement node. Don't configure any other settings.

3. To mark the failed node offline, on any node, modify the cluster configuration file (`cluster.conf`) in the relevant node section to include the text `offline = true`.

   For example, this modified `cluster.conf` will mark the `ghe-data-node-3` node as offline:

   ```
   [cluster "ghe-data-node-3"]
   hostname = ghe-data-node-3
   offline = true
   ipv4 = 192.168.0.6
   # ipv6 = fd12:3456:789a:1::6
   ```

4. From the administrative shell of the node where you modified `cluster.conf`, run `ghe-cluster-config-apply`. This will validate the configuration file, copy it to each node in the cluster, and mark the node offline.

5. To add the newly provisioned replacement node, on any node, modify the `cluster.conf` file to remove the failed node and add the replacement node. For example, this modified `cluster.conf` file replaces `ghe-data-node-3` with the newly provisioned node, `ghe-replacement-data-node-3`:

   ```
   [cluster "ghe-replacement-data-node-3"]
     hostname = ghe-replacement-data-node-3
     ipv4 = 192.168.0.7
     # ipv6 = fd12:3456:789a:1::7
     git-server = true
     pages-server = true
     mysql-server = true
     elasticsearch-server = true
     redis-server = true
     memcache-server = true
     metrics-server = true
     storage-server = true
   ```

6. If you're replacing the primary MySQL or Redis node, in `cluster.conf`, modify the `mysql-master` or `redis-master` value with the replacement node name.

   For example, this modified `cluster.conf` file specifies a newly provisioned cluster node, `ghe-replacement-data-node-1` as the primary MySQL and Redis node:

   ```
   mysql-master = ghe-replacement-data-node-1
   redis-master = ghe-replacement-data-node-1
   ```

7. From the administrative shell of the node with the modified `cluster.conf`, run `ghe-cluster-config-init`. This will initialize the newly added node in the cluster.

8. From the same node, run `ghe-cluster-config-apply`. This will validate the configuration file, copy it to each node in the cluster, and configure each node according to the modified `cluster.conf` file.

9. If you're taking a node offline that provides data services, such as `git-server`,

`pages-server`, or `storage-server`, evacuate the node. For more information, see "Evacuating a cluster node running data services."

Replacing the primary MySQL node

To provide database services, your cluster requires a primary MySQL node and at least one secondary MySQL node. For more information, see "About cluster nodes."

If you want to provide the VM for your primary MySQL node with more resources, or if the node fails, you can replace the node. To minimize downtime, add the new node to your cluster, replicate the MySQL data, and then promote the node. Some downtime is required during promotion.

1. Provision and install GitHub Enterprise Server with a unique hostname on the replacement node.
2. Using the administrative shell or DHCP, **only** configure the IP address of the replacement node. Don't configure any other settings.
3. To connect to your GitHub Enterprise Server instance, SSH into any of your cluster's nodes. From your workstation, run the following command. Replace HOSTNAME with the node's hostname. For more information, see "Accessing the administrative shell (SSH)."

   Shell

   ```
   ssh -p 122 admin@HOSTNAME
   ```

4. Open the cluster configuration file at `/data/user/common/cluster.conf` in a text editor. For example, you can use Vim. Create a backup of the `cluster.conf` file before you edit the file.

   Shell

   ```
   sudo vim /data/user/common/cluster.conf
   ```

5. The cluster configuration file lists each node under a `[cluster "HOSTNAME"]` heading. Add a new heading for the node and enter the key-value pairs for configuration, replacing the placeholders with actual values.
   - Ensure that you include the `mysql-server = true` key-value pair.
   - The following section is an example, and your node's configuration may differ.

   ```
   ...
   [cluster "HOSTNAME"]
     hostname = HOSTNAME
     ipv4 = IPV4-ADDRESS
     # ipv6 = IPV6-ADDRESS
     consul-datacenter = PRIMARY-DATACENTER
     datacenter = DATACENTER
     mysql-server = true
     redis-server = true
     ...
   ...
   ```

6. From the administrative shell of the node with the modified `cluster.conf`, run `ghe-cluster-config-init`. This will initialize the newly added node in the cluster.

7. From the administrative shell of the node where you modified `cluster.conf`, run `ghe-cluster-config-apply`. This will validate the configuration file, copy it to each node in the cluster, and mark the node offline.

8. Wait for MySQL replication to finish. To monitor MySQL replication from any node in the cluster, run `ghe-cluster-status -v`.

   Shortly after adding the node to the cluster, you may see an error for replication status while replication catches up. Replication can take hours depending on the instance's load and the last time the instance generated a database seed.

9. During your scheduled maintenance window, enable maintenance mode. For more information, see "Enabling and scheduling maintenance mode."

10. Ensure that MySQL replication is finished from any node in the cluster by running `ghe-cluster-status -v`.

    > **Warning**: If you do not wait for MySQL replication to finish, you risk data loss on your instance.

11. To set the current MySQL primary node to read-only mode, run the following command from of the instance's nodes.

    Shell

    ```
    echo "SET GLOBAL super_read_only = 1;" | sudo mysql
    ```

12. Wait until Global Transaction Identifiers (GTIDs) set on the primary and secondary MySQL nodes are identical. To check the GTIDs, run the following command from any of the instance's nodes.

    Shell

    ```
    ghe-cluster-each -r mysql -- 'echo "SELECT @@global.gtid_executed;" | sudo mysql'
    ```

13. After the GTIDs on the primary and secondary MySQL nodes match, update the cluster configuration by opening the cluster configuration file at `/data/user/common/cluster.conf` in a text editor.

    - Create a backup of the `cluster.conf` file before you edit the file.
    - In the top-level `[cluster]` section, remove the hostname for the node you replaced from the `mysql-master` key-value pair, then assign the new node instead. If the new node is also a primary Redis node, adjust the `redis-master` key-value pair.

    ```
    [cluster]
      mysql-master = NEW-NODE-HOSTNAME
      redis-master = NEW-NODE-HOSTNAME
      primary-datacenter = primary
    ...
    ```

14. From the administrative shell of the node where you modified `cluster.conf`, run `ghe-cluster-config-apply`. This will validate the configuration file, copy it to each node in the cluster, and mark the node offline.

15. Check the status of MySQL replication from any node in the cluster by running `ghe-cluster-status -v`.

16. If MySQL replication is finished, from any node in the cluster, disable maintenance mode. For more information, see "Enabling and scheduling maintenance mode."