Configuring high availability replication for a cluster

In this article

About high availability replication for clusters
Prerequisites
Creating a high availability replica for a cluster
Monitoring replication between active and replica cluster nodes
Reconfiguring high availability replication after a failover
Disabling high availability replication for a cluster

You can configure a replica of your entire GitHub Enterprise Server cluster in a separate datacenter, allowing your cluster to fail over to redundant nodes.

About high availability replication for clusters

You can provide protection against disruption in a datacenter or cloud region by configuring a cluster deployment of GitHub Enterprise Server for high availability. In a high availability configuration, an identical set of replica nodes sync with the nodes in your active cluster. If hardware or software failures affect the datacenter with your active cluster, you can manually fail over to the replica nodes and continue processing user requests, minimizing the impact of the outage.

In a high availability configuration, nodes that host data services sync regularly with the replica cluster. Replica nodes run in standby and do not serve applications or process user requests.

We recommend configuring high availability as a part of a comprehensive disaster recovery plan for GitHub Enterprise Server clustering. We also recommend performing regular backups. For more information, see "Configuring backups on your instance."

Prerequisites

Hardware and software

For each existing node in your active cluster, you'll need to provision a second virtual machine with identical hardware resources. For example, if your cluster has 13 nodes and each node has 12 vCPUs, 96 GB of RAM, and 750 GB of attached storage, you must provision 13 new virtual machines that each have 12 vCPUs, 96 GB of RAM, and 750 GB of attached storage.

On each new virtual machine, install the same version of GitHub Enterprise Server that runs on the nodes in your active cluster. You don't need to upload a license or perform any additional configuration. For more information, see "Setting up a GitHub Enterprise Server instance."

Note: The nodes that you intend to use for high availability replication should be standalone GitHub Enterprise Server instances. Don't initialize the replica nodes as a second cluster.

Network

You must assign a static IP address to each new node that you provision, and you must configure a load balancer to accept connections and direct them to the nodes in your cluster's front-end tier.

For high availability, the latency between the network with the active nodes and the network with the replica nodes must be less than 70 milliseconds. We don't recommend configuring a firewall between the two networks. For more information about network connectivity between nodes in the replica cluster, see "Cluster network configuration."

Creating a high availability replica for a cluster

- Assigning active nodes to the primary datacenter
- Adding replica nodes to the cluster configuration file
- Example configuration

Assigning active nodes to the primary datacenter

Before you define a secondary datacenter for your replica nodes, ensure that you assign your active nodes to the primary datacenter.

1. SSH into any node in your cluster. For more information, see "Accessing the administrative shell (SSH)."
2. 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
   ```

3. Note the name of your cluster's primary datacenter. The `[cluster]` section at the top of the cluster configuration file defines the primary datacenter's name, using the `primary-datacenter` key-value pair.

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

   - Optionally, change the name of the primary datacenter to something more descriptive or accurate by editing the value of `primary-datacenter`.

4. The cluster configuration file lists each node under a `[cluster "HOSTNAME"]` heading. Under each node's heading, add a new key-value pair to assign the node to a datacenter. Use the same value as `primary-datacenter` from step 3 above. For example, if you want to use the default name (`default`), add the following key-value pair to the section for each node.

```
datacenter = primary
```

When you're done, the section for each node in the cluster configuration file should look like the following example. The order of the key-value pairs doesn't matter.

```
[cluster "HOSTNAME"]
  datacenter = default
  hostname = HOSTNAME
  ipv4 = IP-ADDRESS
  ...
...
```

Note: If you changed the name of the primary datacenter in step 3, find the `consul-datacenter` key-value pair in the section for each node and change the value to the renamed primary datacenter. For example, if you named the primary datacenter `primary`, use the following key-value pair for each node.

```
consul-datacenter = primary
```

5. Apply the new configuration. This command can take some time to finish, so we recommend running the command in a terminal multiplexer like `screen` or `tmux`.

   ```
   ghe-cluster-config-apply
   ```

6. After the configuration run finishes, GitHub Enterprise Server displays the following message.

   ```
   Finished cluster configuration
   ```

After GitHub Enterprise Server returns you to the prompt, you've finished assigning your nodes to the cluster's primary datacenter.

Adding replica nodes to the cluster configuration file

To configure high availability, you must define a corresponding replica node for every active node in your cluster. To create a new cluster configuration that defines both active and replica nodes, you'll complete the following tasks.

- Create a copy of the active cluster configuration file.
- Edit the copy to define replica nodes that correspond to the active nodes, adding the IP addresses of the new virtual machines that you provisioned.
- Merge the modified copy of the cluster configuration back into your active configuration.
- Apply the new configuration to start replication.

For an example configuration, see "Example configuration."

1. For each node in your cluster, provision a matching virtual machine with identical specifications, running the same version of GitHub Enterprise Server. Note the IPv4 address and hostname for each new cluster node. For more information, see "Prerequisites."

   Note: If you're reconfiguring high availability after a failover, you can use the old nodes from the primary datacenter instead.

2. SSH into any node in your cluster. For more information, see "Accessing the administrative shell (SSH)."

3. Back up your existing cluster configuration.

```
cp /data/user/common/cluster.conf ~/$(date +%Y-%m-%d)-cluster.conf.backup
```

4. Create a copy of your existing cluster configuration file in a temporary location, like `/home/admin/cluster-replica.conf`.

```
grep -Ev "(?:|ipv|uuid)" /data/user/common/cluster.conf > ~/cluster-
replica.conf
```

5. Remove the `[cluster]` section from the temporary cluster configuration file that you copied in the previous step.

```
git config -f ~/cluster-replica.conf --remove-section cluster
```

6. Decide on a name for the secondary datacenter where you provisioned your replica nodes, then update the temporary cluster configuration file with the new datacenter name. Replace `SECONDARY` with the name you choose.

```
sed -i 's/datacenter = default/datacenter = SECONDARY/g' ~/cluster-
replica.conf
```

7. Decide on a pattern for the replica nodes' hostnames.

> **Warning**: Hostnames for replica nodes must be unique and differ from the hostname for the corresponding active node.

8. Open the temporary cluster configuration file from step 3 in a text editor. For example, you can use Vim.

```
sudo vim ~/cluster-replica.conf
```

9. In each section within the temporary cluster configuration file, update the node's configuration. The cluster configuration file lists each node under a `[cluster "HOSTNAME"]` heading.

   - Change the quoted hostname in the section heading and the value for `hostname` within the section to the replica node's hostname, per the pattern you chose in step 7 above.
   - Add a new key named `ipv4`, and set the value to the replica node's static IPv4 address.
   - Add a new key-value pair, `replica = enabled`.

```
[cluster "NEW REPLICA NODE HOSTNAME"]
  ...
  hostname = NEW REPLICA NODE HOSTNAME
  ipv4 = NEW REPLICA NODE IPV4 ADDRESS
  replica = enabled
  ...
```

```
...
```

10. Append the contents of the temporary cluster configuration file that you created in step 4 to the active configuration file.

```
cat ~/cluster-replica.conf >> /data/user/common/cluster.conf
```

11. Designate the primary MySQL and Redis nodes in the secondary datacenter. Replace `REPLICA MYSQL PRIMARY HOSTNAME` and `REPLICA REDIS PRIMARY HOSTNAME` with the hostnames of the replica node that you provisioned to match your existing MySQL and Redis primaries.

```
git config -f /data/user/common/cluster.conf cluster.mysql-master-replica
REPLICA-MYSQL-PRIMARY-HOSTNAME
git config -f /data/user/common/cluster.conf cluster.redis-master-replica
REPLICA-REDIS-PRIMARY-HOSTNAME
```

Warning: Review your cluster configuration file before proceeding.

- In the top-level `[cluster]` section, ensure that the values for `mysql-master-replica` and `redis-master-replica` are the correct hostnames for the replica nodes in the secondary datacenter that will serve as the MySQL and Redis primaries after a failover.
- In each section for an active node named `[cluster "ACTIVE NODE HOSTNAME"]`, double-check the following key-value pairs.
  - `datacenter` should match the value of `primary-datacenter` in the top-level `[cluster]` section.
  - `consul-datacenter` should match the value of `datacenter`, which should be the same as the value for `primary-datacenter` in the top-level `[cluster]` section.
- Ensure that for each active node, the configuration has **one** corresponding section for **one** replica node with the same roles. In each section for a replica node, double-check each key-value pair.
  - `datacenter` should match all other replica nodes.
  - `consul-datacenter` should match all other replica nodes.
  - `hostname` should match the hostname in the section heading.
  - `ipv4` should match the node's unique, static IPv4 address.
  - `replica` should be configured as `enabled`.
- Take the opportunity to remove sections for offline nodes that are no longer in use.

To review an example configuration, see "Example configuration."

12. Initialize the new cluster configuration. This command can take some time to finish, so we recommend running the command in a terminal multiplexer like `screen` or `tmux`.

```
ghe-cluster-config-init
```

13. After the initialization finishes, GitHub Enterprise Server displays the following message.

```
Finished cluster initialization
```

14. Apply the new configuration. This command can take some time to finish, so we recommend running the command in a terminal multiplexer like `screen` or `tmux`.

```
ghe-cluster-config-apply
```

15. After the configuration run finishes, verify that cluster replication is correctly set up and working.

    ```
    ghe-cluster-repl-status
    ```

16. After the configuration run finishes, GitHub Enterprise Server displays the following message.

    ```
    Finished cluster configuration
    ```

17. Configure a load balancer that will accept connections from users after you fail over to the replica nodes. For more information, see "Cluster network configuration."

You've finished configuring high availability replication for the nodes in your cluster. Each active node begins replicating configuration and data to its corresponding replica node, and you can direct traffic to the load balancer for the secondary datacenter in the event of a failure. For more information about failing over, see "Initiating a failover to your replica cluster."

Example configuration

The top-level `[cluster]` configuration should look like the following example.

```
[cluster]
  mysql-master = HOSTNAME-OF-ACTIVE-MYSQL-MASTER
  redis-master = HOSTNAME-OF-ACTIVE-REDIS-MASTER
  primary-datacenter = PRIMARY-DATACENTER-NAME
  mysql-master-replica = HOSTNAME-OF-REPLICA-MYSQL-MASTER
  redis-master-replica = HOSTNAME-OF-REPLICA-REDIS-MASTER
  mysql-auto-failover = false
...
```

The configuration for an active node in your cluster's storage tier should look like the following example.

```
...
[cluster "UNIQUE ACTIVE NODE HOSTNAME"]
  datacenter = default
  hostname = UNIQUE-ACTIVE-NODE-HOSTNAME
  ipv4 = IPV4-ADDRESS
  consul-datacenter = default
  consul-server = true
  git-server = true
  pages-server = true
  mysql-server = true
  elasticsearch-server = true
  redis-server = true
  memcache-server = true
  metrics-server = true
  storage-server = true
  uuid = UUID SET AUTOMATICALLY
...
```

The configuration for the corresponding replica node in the storage tier should look like the following example.

- Important differences from the corresponding active node are **bold**.
- GitHub Enterprise Server assigns the value for `uuid` automatically, so you shouldn't define this value for replica nodes that you will initialize.
- The server roles, defined by `*-server` keys, match the corresponding active node.

```
...
[cluster "UNIQUE REPLICA NODE HOSTNAME"]
  replica = enabled
  ipv4 = IPV4 ADDRESS OF NEW VM WITH IDENTICAL RESOURCES
  datacenter = SECONDARY DATACENTER NAME
  hostname = UNIQUE REPLICA NODE HOSTNAME
  consul-datacenter = SECONDARY DATACENTER NAME
  consul-server = true
  git-server = true
  pages-server = true
  mysql-server = true
  elasticsearch-server = true
  redis-server = true
  memcache-server = true
  metrics-server = true
  storage-server = true
  uuid = DO NOT DEFINE
...
```

Monitoring replication between active and replica cluster nodes

Initial replication between the active and replica nodes in your cluster takes time. The amount of time depends on the amount of data to replicate and the activity levels for GitHub Enterprise Server.

You can monitor the progress on any node in the cluster, using command-line tools available via the GitHub Enterprise Server administrative shell. For more information about the administrative shell, see "Accessing the administrative shell (SSH)."

To monitor the replication of all services, use the following command.

```
ghe-cluster-repl-status
```

You can use `ghe-cluster-status` to review the overall health of your cluster. For more information, see "Command-line utilities."

Reconfiguring high availability replication after a failover

After you fail over from the cluster's active nodes to the cluster's replica nodes, you can reconfigure high availability in one of two ways. The method you choose will depend on the reason that you failed over, and the state of the original active nodes.

- Provision and configure a new set of replica nodes for each of the new active nodes in your secondary datacenter.
- Use the original active nodes as the new replica nodes.

The process for reconfiguring high availability is identical to the initial configuration of high availability. For more information, see "Creating a high availability replica for a cluster."

If you use the original active nodes, after reconfiguring high availability, you will need to unset maintenance mode on the nodes. For more information, see "Enabling and

scheduling maintenance mode."

Disabling high availability replication for a cluster

You can stop replication to the replica nodes for your cluster deployment of GitHub Enterprise Server.

1. SSH into any node in your cluster. For more information, see "Accessing the administrative shell (SSH)."
2. 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
   ```

3. In the top-level `[cluster]` section, delete the `redis-master-replica`, and `mysql-master-replica` key-value pairs.
4. Delete each section for a replica node. For replica nodes, `replica` is configured as `enabled`.
5. Apply the new configuration. This command can take some time to finish, so we recommend running the command in a terminal multiplexer like `screen` or `tmux`.

   ```
   ghe-cluster-config-apply
   ```

6. After the configuration run finishes, GitHub Enterprise Server displays the following message.

   ```
   Finished cluster configuration
   ```

After GitHub Enterprise Server returns you to the prompt, you've finished disabling high availability replication.