About high availability configuration

In this article

Targeted failure scenarios
Network traffic failover strategies
Utilities for replication management
Further reading

In a high availability configuration, a fully redundant secondary GitHub Enterprise Server appliance is kept in sync with the primary appliance through replication of all major datastores.

When you configure high availability, there is an automated setup of one-way, asynchronous replication of all datastores (Git repositories, MySQL, Redis, and Elasticsearch) from the primary to the replica appliance. Most GitHub Enterprise Server configuration settings are also replicated, including the Management Console password. For more information, see "Administering your instance from the web UI."

GitHub Enterprise Server supports an active/passive configuration, where the replica appliance runs as a standby with database services running in replication mode but application services stopped.

After replication has been established, the Management Console is no longer accessible on replica appliances. If you navigate to the replica's IP address or hostname on port 8443, you'll see a "Server in replication mode" message, which indicates that the appliance is currently configured as a replica.

Note: There is a maximum of 8 high availability replicas (both passive and active/geo replicas) allowed for GitHub Enterprise Server.

Targeted failure scenarios

Use a high availability configuration for protection against:

- **Software crashes**, either due to operating system failure or unrecoverable applications.
- **Hardware failures**, including storage hardware, CPU, RAM, network interfaces, etc.
- **Virtualization host system failures**, including unplanned and scheduled maintenance events for AWS, Azure, or GCP.
- **Logically or physically severed network**, if the failover appliance is on a separate network not impacted by the failure.

A high availability configuration is not a good solution for:

- **Scaling-out**. While you can distribute traffic geographically using geo-replication, the performance of writes is limited to the speed and availability of the primary appliance. For more information, see "About geo-replication."
- **CI/CD load**. If you have a large number of CI clients that are geographically distant from your primary instance, you may benefit from configuring a repository cache. For

more information, see "About repository caching."

- **Backing up your primary appliance**. A high availability replica does not replace off-site backups in your disaster recovery plan. Some forms of data corruption or loss may be replicated immediately from the primary to the replica. To ensure safe rollback to a stable past state, you must perform regular backups with historical snapshots.
- **Zero downtime upgrades**. To prevent data loss and split-brain situations in controlled promotion scenarios, place the primary appliance in maintenance mode and wait for all writes to complete before promoting the replica.

Network traffic failover strategies

During failover, you must separately configure and manage redirecting network traffic from the primary to the replica.

DNS failover

With DNS failover, use short TTL values in the DNS records that point to the primary GitHub Enterprise Server appliance. We recommend a TTL between 60 seconds and five minutes.

During failover, you must place the primary into maintenance mode and redirect its DNS records to the replica appliance's IP address. The time needed to redirect traffic from primary to replica will depend on the TTL configuration and time required to update the DNS records.

If you are using geo-replication, you must configure Geo DNS to direct traffic to the nearest replica. For more information, see "About geo-replication."

Load balancer

A load balancer design uses a network device to direct Git and HTTP traffic to individual GitHub Enterprise Server appliances. You can use a load balancer to restrict direct traffic to the appliance for security purposes or to redirect traffic if needed without DNS record changes. We strongly recommend using a TCP-based load balancer that supports the PROXY protocol. DNS lookups for the GitHub Enterprise Server hostname should resolve to the load balancer. We recommend that you enable subdomain isolation. If subdomain isolation is enabled, an additional wildcard record (`*.HOSTNAME`) should also resolve to the load balancer. For more information, see "Enabling subdomain isolation."

During failover, you must place the primary appliance into maintenance mode. You can configure the load balancer to automatically detect when the replica has been promoted to primary, or it may require a manual configuration change. You must manually promote the replica to primary before it will respond to user traffic. For more information, see "Using GitHub Enterprise Server with a load balancer."

You can monitor the availability of GitHub Enterprise Server by checking the status code that is returned for the `https://HOSTNAME/status` URL. An appliance that can service user traffic will return status code `200` (OK). An appliance may return `503` (Service Unavailable) for this URL and other web or API requests for a few reasons:

- The appliance is a passive replica, such as the replica in a two-node high availability configuration.
- The appliance is in maintenance mode.
- The appliance is part of a geo-replication configuration, but is an inactive replica.

You can also use the Replication overview dashboard available at:

`https://HOSTNAME/setup/replication`

Utilities for replication management

People with administrative SSH access to an instance in a high-availability configuration can use command-line utilities to manage replication. For more information, see "Command-line utilities."

Further reading

- "Creating a high availability replica"
- "Network ports"

Legal

© 2023 GitHub, Inc. Terms Privacy Status Pricing Expert services Blog