About geo-replication

In this article

Geo-replication on GitHub Enterprise Server uses multiple active replicas to fulfill requests from geographically distributed data centers.

Multiple active replicas can provide a shorter distance to the nearest replica. For example, an organization with offices in San Francisco, New York, and London could run the primary appliance in a datacenter near New York and two replicas in datacenters near San Francisco and London. Using geolocation-aware DNS, users can be directed to the closest server available and access repository data faster. Designating the appliance near New York as the primary helps reduce the latency between the hosts, compared to the appliance near San Francisco being the primary which has a higher latency to London.

The active replica proxies requests that it can't process itself to the primary instance. The replicas function as a point of presence terminating all SSL connections. Traffic between hosts is sent through an encrypted VPN connection, similar to a two-node high availability configuration without geo-replication.

Git requests and specific file server requests, such as LFS and file uploads, can be served directly from the replica without loading any data from the primary. Web requests are always routed to the primary, but if the replica is closer to the user the requests are faster due to the closer SSL termination.

Geo DNS, such as Amazon's Route 53 service, is required for geo-replication to work seamlessly. The hostname for the instance should resolve to the replica that is closest to the user's location.

Limitations

Writing requests to the replica requires sending the data to the primary and all replicas. This means that the performance of all writes is limited by the slowest replica, although new geo-replicas can seed the majority of their data from existing co-located geo-replicas, rather than from the primary. To reduce the latency and bandwidth caused by distributed teams and large CI farms without impacting write throughput, you can configure repository caching instead. For more information, see "About repository caching."

Geo-replication will not add capacity to a GitHub Enterprise Server instance or solve performance issues related to insufficient CPU or memory resources. If the primary

appliance is offline, active replicas will be unable to serve any read or write requests.

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

Monitoring a geo-replication configuration

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`

Further reading

- "Creating a high availability replica"