Upgrading GitHub Enterprise Server

In this article

Upgrade GitHub Enterprise Server to get the latest features and security updates.

> **Who can use this feature**
> Site administrators can upgrade a GitHub Enterprise Server instance.

About upgrades to GitHub Enterprise Server

GitHub Enterprise Server is constantly improving, with new functionality and bug fixes introduced through feature and patch releases. You are responsible for upgrades to your instance. For more information, see "About upgrades to new releases."

To upgrade an instance, you must plan and communicate the upgrade, choose the appropriate package, back up your data, and then perform the upgrade.

Prerequisites

To successfully upgrade your GitHub Enterprise Server instance, the instance's data disk must be at least 15% free. GitHub recommends ensuring there is more free storage on the disk. In some rare cases, for customers with large data volumes, this threshold may differ.

Preparing to upgrade

To prepare for an upgrade, plan the upgrade path, optionally upgrade GitHub Actions runners, and schedule a maintenance window.

1. Determine an upgrade strategy and choose a version to upgrade to. For more information, see "Upgrade requirements" and refer to the Upgrade assistant to find the upgrade path from your current release version.
2. Create a fresh backup of your instance's primary node with the GitHub Enterprise Server Backup Utilities. For more information, see the README in the GitHub Enterprise Server Backup Utilities project documentation.

> **Note:** Your GitHub Enterprise Server Backup Utilities version needs to be the same version as, or at most two versions ahead of, your GitHub Enterprise Server instance. For more information, see "Configuring backups on your instance."

3. If your GitHub Enterprise Server instance uses ephemeral self-hosted runners for GitHub Actions and you've disabled automatic updates, upgrade your runners to the version of the runner application that your upgraded instance will run.
4. If you are upgrading using an upgrade package, schedule a maintenance window for GitHub Enterprise Server end users. If you are using a hotpatch, maintenance mode is not required.

   > **Note:** The maintenance window depends on the type of upgrade you perform. Upgrades using a hotpatch usually don't require a maintenance window. Sometimes a reboot is required, which you can perform at a later time. Following the versioning scheme of MAJOR.FEATURE.PATCH, patch releases using an upgrade package typically require less than five minutes of downtime. Feature releases that include data migrations take longer depending on storage performance and the amount of data that's migrated. For more information, see "Enabling and scheduling maintenance mode."

Taking a snapshot

A snapshot stores the state of a virtual machine (VM) at a point in time. GitHub highly recommends taking a snapshot before upgrading your VM so that if an upgrade fails, you can revert your VM back to the snapshot. GitHub only recommends taking a VM snapshot when the instance's VM is powered down, or when the instance is in maintenance mode and all background jobs have finished.

If you're upgrading to a new feature release, you must take a VM snapshot. If you're upgrading to a patch release, you can attach the existing data disk.

There are two types of snapshots:

- **VM snapshots** save your entire VM state, including user data and configuration data. This snapshot method requires a large amount of disk space and is time consuming.
- **Data disk snapshots** only save your user data.

  > **Notes:**
  >
  > - Some platforms don't allow you to take a snapshot of just your data disk. For these platforms, you'll need to take a snapshot of the entire VM.
  > - If your hypervisor does not support full VM snapshots, you should take a snapshot of the root disk and data disk in quick succession.

Platform	Snapshot method	Documentation
Amazon AWS	Disk	Create Amazon EBS snapshots in the AWS documentation
Azure	VM	Back up an Azure VM from the VM settings in Microsoft Learn
Hyper-V	VM	Enable or disable checkpoints in Hyper-V in Microsoft Learn

Google Compute Engine	Disk	Create and manage disk snapshots in the Google Cloud documentation
VMware	VM	Taking Snapshots of a Virtual Machine in VMware Docs

Choosing an upgrade package

You can upgrade a GitHub Enterprise Server instance to a new patch release or to a new feature release. To upgrade to a patch release, you can use a hotpatch or an upgrade package. To upgrade to a feature release, you must use an upgrade package.

A GitHub Enterprise Server instance comprises one or more nodes. The upgrade process you must follow depends on how many nodes your instance has. For more information, see "About GitHub Enterprise Server."

- Upgrading with a hotpatch
  - Upgrading a standalone instance using a hotpatch
  - Upgrading an instance with multiple nodes using a hotpatch
- Upgrading with an upgrade package
  - Upgrading a standalone instance using an upgrade package
  - Upgrading an instance with multiple nodes using an upgrade package

Upgrading with a hotpatch

You can upgrade GitHub Enterprise Server to the latest patch release using a hotpatch.

You can use hotpatching to upgrade to a newer patch release, but not a feature release. For example, you can upgrade from 2.10.1 to 2.10.5 because they are in the same feature series, but not from 2.10.9 to 2.11.0 because they are in a different feature series.

Hotpatches do not generally require a reboot. If a hotpatch does require a reboot, the GitHub Enterprise Server release notes will indicate the requirement.

Hotpatches require a configuration run, which can cause a brief period of errors or unresponsiveness for some or all services on your GitHub Enterprise Server instance. You are not required to enable maintenance mode during installation of a hotpatch, but doing so will guarantee that users see a maintenance page instead of errors or timeouts. For more information, see "Enabling and scheduling maintenance mode."

Using the Management Console, you can install a hotpatch immediately or schedule it for later installation. You can use the administrative shell to install a hotpatch with the `ghe-upgrade` utility. For more information, see "Upgrade requirements."

Notes:

- If your GitHub Enterprise Server instance is running a release candidate build, you can't upgrade with a hotpatch.
- Installing a hotpatch using the Management Console is not available for clusters. To install a hotpatch for a cluster, see "Upgrading a cluster."

- Upgrading a standalone instance using a hotpatch
- Upgrading an instance with multiple nodes using a hotpatch

Upgrading a standalone instance using a hotpatch

If you're upgrading an instance with one node using a hotpatch, and your target is a patch release, you can upgrade using Management Console. To upgrade to a feature release, you must use the administrative shell.

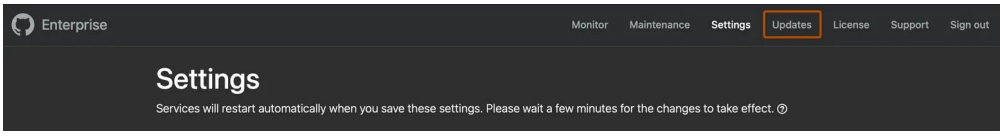
- Installing a hotpatch using the Management Console
- Installing a hotpatch using the administrative shell

Installing a hotpatch using the Management Console

You can use the Management Console to upgrade with a hotpatch by enabling automatic updates. You will then be presented with the latest available version of GitHub Enterprise Server that you can upgrade to.

If the upgrade target you're presented with is a feature release instead of a patch release, you cannot use the Management Console to install a hotpatch. You must install the hotpatch using the administrative shell instead. For more information, see "Installing a hotpatch using the administrative shell."

1. Enable automatic updates. For more information, see "Enabling automatic update checks."
2. From an administrative account on GitHub Enterprise Server, in the upper-right corner of any page, click 🚀.
3. If you're not already on the "Site admin" page, in the upper-left corner, click **Site admin**.
4. In the "🚀 Site admin" sidebar, click **Management Console**.
5. In the top navigation bar, click **Updates**.



6. When a new hotpatch has been downloaded, select the **Install package** dropdown menu.
   - To install immediately, click **Now**.
   - To install later, select a later date.
7. Click **Install**.

Installing a hotpatch using the administrative shell

> **Note:** If you've enabled automatic update checks, you don't need to download the upgrade package and can use the file that was automatically downloaded. For more information, see "Enabling automatic update checks."

1. SSH into your GitHub Enterprise Server instance. If your instance comprises multiple nodes, for example if high availability or geo-replication are configured, SSH into the primary node. If you use a cluster, you can SSH into any node. For more information about SSH access, see "Accessing the administrative shell (SSH)."

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

2. Browse to the GitHub Enterprise Server Releases page. Next to the release you are upgrading to, click **Download**, then click the **Upgrading** tab. Copy the URL for the upgrade hotpackage (*.hpkg* file).
3. Download the upgrade package to your GitHub Enterprise Server instance using `curl`:

   ```
   admin@HOSTNAME:~$ curl -L -O UPGRADE-PKG-URL
   ```

4. Run the `ghe-upgrade` command using the package file name:

   ```
   admin@HOSTNAME:~$ ghe-upgrade GITHUB-UPGRADE.hpkg
   *** verifying upgrade package signature...
   ```

5. If at least one service or system component requires a reboot, the hotpatch upgrade script notifies you. For example, updates to the kernel, MySQL, or Elasticsearch may require a reboot.

Upgrading an instance with multiple nodes using a hotpatch

If you are installing a hotpatch, you do not need to enter maintenance mode or stop replication.

- Upgrading the primary node using a hotpatch
- Upgrading additional nodes using a hotpatch

Upgrading the primary node using a hotpatch

For instructions to upgrade the primary node, see "Installing a hotpatch using the administrative shell."

Upgrading additional nodes using a hotpatch

To upgrade an instance that comprises multiple nodes, such as a high-availability or geo-replication configuration, you must repeat the following procedure on each replica node, one at a time.

1. To upgrade the node, follow the instructions in "Installing a hotpatch using the administrative shell."
2. Connect to the replica node over SSH as the `admin` user on port 122:

   ```
   ssh -p 122 admin@REPLICA_HOST
   ```

3. Verify the upgrade by running:

   ```
   ghe-version
   ```

4. Repeat the steps above for each additional node.

Upgrading with an upgrade package

While you can use a hotpatch to upgrade to the latest patch release within a feature series, you must use an upgrade package to upgrade to a newer feature release. For example to upgrade from 2.11.10 to 2.12.4 you must use an upgrade package since these are in different feature series. For more information, see "Upgrade requirements."

- Upgrading a standalone instance using an upgrade package
- Upgrading an instance with multiple nodes using an upgrade package

Upgrading a standalone instance using an upgrade package

Note: If you've enabled automatic update checks, you don't need to download the upgrade package and can use the file that was automatically downloaded. For more information, see "Enabling automatic update checks."

1. SSH into your GitHub Enterprise Server instance. If your instance comprises multiple nodes, for example if high availability or geo-replication are configured, SSH into the primary node. If you use a cluster, you can SSH into any node. For more information about SSH access, see "Accessing the administrative shell (SSH)."

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

2. Browse to the GitHub Enterprise Server Releases page. Next to the release you are upgrading to, click **Download**, then click the **Upgrading** tab. Select the appropriate platform and copy the URL for the upgrade package (*.pkg* file).

3. Download the upgrade package to your GitHub Enterprise Server instance using `curl`:

   ```
   admin@HOSTNAME:~$ curl -L -O UPGRADE-PKG-URL
   ```

4. Enable maintenance mode and wait for all active processes to complete on the GitHub Enterprise Server instance. For more information, see "Enabling and scheduling maintenance mode."

   Note: When upgrading the primary node in a high availability configuration, the instance should already be in maintenance mode if you are following the instructions in "Upgrading the primary node."

5. Run the `ghe-upgrade` command using the package file name:

   ```
   admin@HOSTNAME:~$ ghe-upgrade GITHUB-UPGRADE.pkg
   *** verifying upgrade package signature...
   ```

6. Confirm that you'd like to continue with the upgrade and restart after the package signature verifies. The new root filesystem writes to the secondary partition and the instance automatically restarts in maintenance mode:

   ```
   *** applying update...
   This package will upgrade your installation to version VERSION-NUMBER
   Current root partition: /dev/xvda1 [VERSION-NUMBER]
   Target root partition:  /dev/xvda2
   ```

```
Proceed with installation? [y/N]
```

7. Optionally, during an upgrade to a feature release, you can monitor the status of database migrations using the `ghe-migrations` utility. For more information, see "Command-line utilities."

8. After the instance restarts, the upgrade will continue in the background. You cannot unset maintenance mode until the process completes. To monitor progress, read the output in `/data/user/common/ghe-config.log`. For example, you can tail the log by running the following command:

   ```
   tail -f /data/user/common/ghe-config.log
   ```

9. Optionally, after the upgrade, validate the upgrade by configuring an IP exception list to allow access to a specified list of IP addresses. For more information, see "Enabling and scheduling maintenance mode."

10. For single node upgrades, disable maintenance mode so users can use your GitHub Enterprise Server instance.

    > **Note**: After you upgrade an instance in a high availability configuration, you should remain in maintenance mode until you have upgraded all of the replica nodes and replication is current. For more information, see "Upgrading additional nodes with an upgrade package."

Upgrading an instance with multiple nodes using an upgrade package

To upgrade an instance that comprises multiple nodes using an upgrade package, you must upgrade the primary node, then upgrade any additional nodes.

- Upgrading the primary node with an upgrade package
- Upgrading additional nodes with an upgrade package

Upgrading the primary node with an upgrade package

> **Warning:** When replication is stopped, if the primary fails, any work that is done before the replica is upgraded and the replication begins again will be lost.

1. On the primary node, enable maintenance mode and wait for all active processes to complete. For more information, see "Enabling and scheduling maintenance mode."

2. Connect to the replica node over SSH as the `admin` user on port 122:

   ```
   ssh -p 122 admin@REPLICA_HOST
   ```

3. To stop replication on all nodes, run `ghe-repl-stop` on each node.

4. To upgrade the primary node, follow the instructions in "Upgrading a standalone instance using an upgrade package."

Upgrading additional nodes with an upgrade package

To upgrade an instance that comprises multiple nodes, such as a high-availability or geo-replication configuration, you must repeat the following procedure on each replica node, one at a time.

1. Upgrade the node by following the instructions in "Upgrading a standalone instance using an upgrade package."
2. Connect to the replica node over SSH as the `admin` user on port 122:

   ```
   ssh -p 122 admin@REPLICA_HOST
   ```

3. Verify the upgrade by running:

   ```
   ghe-version
   ```

4. On the replica node, to start replication, run `ghe-repl-start`.
5. On the replica node, to make sure replication services are running correctly, run `ghe-repl-status`. This command will return `OK` for all services when a successful replication is in progress and the replica has upgraded. If the command returns `Replication is not running`, the replication may still be starting. Wait about one minute before running `ghe-repl-status` again.

   Notes:

   - While the resync is in progress `ghe-repl-status` may indicate that replication is behind. For example, you may see the following message.

     ```
     CRITICAL: git replication is behind the primary by more than 1007 repositories
     and/or gists
     ```

   - If GitHub Actions is enabled on your GitHub Enterprise Server instance, you may see a message like the following. This message is expected when replication is paused due to maintenance mode being set on the primary appliance. Once maintenance mode is unset, this message should be resolved.

     ```
     CRITICAL: mssql replication is down, didn't find Token_Configuration!
     ```

   If `ghe-repl-status` did not return `OK`, and the explanation isn't listed in the note above, contact GitHub Enterprise Support. For more information, see "Contacting GitHub Support."
6. Repeat the steps above for each additional node.
7. After you have upgraded the last replica node and the resync is complete, disable maintenance mode so users can use your GitHub Enterprise Server instance.

Restoring from a failed upgrade

If an upgrade fails or is interrupted, you should revert your instance back to its previous state. The process for completing this depends on the type of upgrade.

Rolling back a patch release

To roll back a patch release, use the `ghe-upgrade` command with the `--allow-patch-rollback` switch. Before rolling back, replication must be temporarily stopped by running `ghe-repl-stop` on all replica nodes. When rolling back an upgrade, you must use an

upgrade package file with the *.pkg* extension. Hotpatch package files with the *.hpkg* extension are not supported.

```
ghe-upgrade --allow-patch-rollback EARLIER-RELEASE-UPGRADE-PACKAGE.pkg
```

A reboot is required after running the command. Rolling back does not affect the data partition, as migrations are not run on patch releases.

After the rollback is complete, restart replication by running `ghe-repl-start` on all nodes.

For more information, see "Command-line utilities."

Rolling back a feature release

To roll back from a feature release, restore from a VM snapshot to ensure that root and data partitions are in a consistent state. For more information, see "Taking a snapshot."

Further reading

- "About upgrades to new releases"

Legal

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