Setting up a staging instance

In this article

You can set up a GitHub Enterprise Server instance in a separate, isolated environment, and use the instance to validate and test changes.

About staging instances

GitHub recommends that you set up a separate environment to test backups, updates, or changes to the configuration for your GitHub Enterprise Server instance. This environment, which you should isolate from your production systems, is called a staging environment.

For example, to protect against loss of data, you can regularly validate the backup of your production instance. You can regularly restore the backup of your production data to a separate GitHub Enterprise Server instance in a staging environment. On this staging instance, you could also test the upgrade to the latest feature release of GitHub Enterprise Server.

Tip: You may reuse your existing GitHub Enterprise license file as long as the staging instance is not used in a production capacity.

Considerations for a staging environment

To thoroughly test GitHub Enterprise Server and recreate an environment that's as similar to your production environment as possible, consider the external systems that interact with your instance. For example, you may want to test the following in your staging environment.

- Authentication, especially if you use an external authentication provider like SAML
- Integration with an external ticketing system
- Integration with a continuous integration server
- External scripts or software that use the GitHub Enterprise Server APIs
- External SMTP server for email notifications

Setting up a staging instance

You can set up a staging instance from scratch and configure the instance however you like. For more information, see "Setting up a GitHub Enterprise Server instance" and "Configuring GitHub Enterprise."

Alternatively, you can create a staging instance that reflects your production configuration by restoring a backup of your production instance to the staging instance.

1. Back up your production instance.
2. Set up a staging instance.
3. Configure GitHub Actions.
4. Configure GitHub Packages.
5. Restore your production backup.
6. Review the instance's configuration.
7. Apply the instance's configuration.

1. Back up your production instance

If you want to test changes on an instance that contains the same data and configuration as your production instance, back up the data and configuration from the production instance using GitHub Enterprise Server Backup Utilities. For more information, see "Configuring backups on your instance."

> **Warning**: If you use GitHub Actions or GitHub Packages in production, your backup will include your production configuration for external storage. To avoid potential loss of data by writing to your production storage from your staging instance, you must configure each feature in steps 3 and 4 before you restore your backup.

2. Set up a staging instance

Set up a new instance to act as your staging environment. You can use the same guides for provisioning and installing your staging instance as you did for your production instance. For more information, see "Setting up a GitHub Enterprise Server instance."

If you plan to restore a backup of your production instance, continue to the next step. Alternatively, you can configure the instance manually and skip the following steps.

3. Configure GitHub Actions

Optionally, if you use GitHub Actions on your production instance, configure the feature on the staging instance before restoring your production backup. If you don't use GitHub Actions, skip to "1. Configure GitHub Packages."

> **Warning**: If you don't configure GitHub Actions on the staging instance before restoring your production backup, your staging instance will use your production instance's external storage, which could result in loss of data. We strongly recommended that you use different external storage for your staging instance. For more information, see "Using a staging environment."

1. SSH into the staging instance. For more information, see "Accessing the administrative shell (SSH)."

   Shell

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

2. To configure the staging instance to use an external storage provider for GitHub Actions, enter one of the following commands.

   - Azure Blob Storage:

     ```shell
     ghe-config secrets.actions.storage.blob-provider "azure"
     ```

   - Amazon S3:

     ```shell
     ghe-config secrets.actions.storage.blob-provider "s3"
     ```

   - Google Cloud Storage:

     ```shell
     ghe-config secrets.actions.storage.blob-provider "gcs"
     ```

3. Configure the external storage connection by entering the following commands, replacing the placeholder values with actual values for your connection.

   - Azure Blob Storage:

     ```shell
     ghe-config secrets.actions.storage.azure.connection-string "CONNECTION
     STRING"
     ```

   - Amazon S3:

     ```shell
     ghe-config secrets.actions.storage.s3.bucket-name "S3 BUCKET NAME"
     ghe-config secrets.actions.storage.s3.service-url "S3 SERVICE URL"
     ghe-config secrets.actions.storage.s3.access-key-id "S3 ACCESS KEY ID"
     ghe-config secrets.actions.storage.s3.access-secret "S3 ACCESS SECRET"
     ```

     Optionally, to force path-style addressing for S3, also enter the following command.

     ```shell
     ghe-config secrets.actions.storage.s3.force-path-style true
     ```

   - Google Cloud Storage:

     ```shell
     ghe-config secrets.actions.storage.gcs.service-url "SERVICE URL"
     ghe-config secrets.actions.storage.gcs.bucket-name "BUCKET NAME"
     ghe-config secrets.actions.storage.gcs.access-key-id "HMAC ACCESS ID"
     ```

```
ghe-config secrets.actions.storage.gcs.access-secret "HMAC SECRET"
```

4. To prepare to enable GitHub Actions on the staging instance, enter the following command.

```shell
ghe-config app.actions.enabled true
```

4. Configure GitHub Packages

Optionally, if you use GitHub Packages on your production instance, configure the feature on the staging instance before restoring your production backup. If you don't use GitHub Packages, skip to "1. Restore your production backup."

Warning: If you don't configure GitHub Packages on the staging instance before restoring your production backup, your staging instance will use your production instance's external storage, which could result in loss of data. We strongly recommended that you use different external storage for your staging instance.

1. Review the backup you will restore to the staging instance.
   - If you took the backup with GitHub Enterprise Server Backup Utilities 3.5 or later, the backup includes the configuration for GitHub Packages. Continue to the next step.
   - If you took the backup with GitHub Enterprise Server Backup Utilities 3.4 or earlier, configure GitHub Packages on the staging instance. For more information, see "Getting started with GitHub Packages for your enterprise."

2. SSH into the staging instance. For more information, see "Accessing the administrative shell (SSH)."

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

3. Configure the external storage connection by entering the following commands, replacing the placeholder values with actual values for your connection.
   - Azure Blob Storage:

```shell
ghe-config secrets.packages.blob-storage-type "azure"
ghe-config secrets.packages.azure-container-name "AZURE CONTAINER NAME"
ghe-config secrets.packages.azure-connection-string "CONNECTION STRING"
```

   - Amazon S3:

```shell
ghe-config secrets.packages.blob-storage-type "s3"
```

```
ghe-config secrets.packages.service-url "S3 SERVICE URL"
ghe-config secrets.packages.s3-bucket "S3 BUCKET NAME"
ghe-config secrets.packages.aws-access-key "S3 ACCESS KEY ID"
ghe-config secrets.packages.aws-secret-key "S3 ACCESS SECRET"
```

4. To prepare to enable GitHub Packages on the staging instance, enter the following command.

```shell
ghe-config app.packages.enabled true
```

5. Restore your production backup

Use the `ghe-restore` command to restore the rest of the data from the backup. For more information, see "Configuring backups on your instance."

If the staging instance is already configured and you want to overwrite settings, certificate, and license data, add the `-c` option to the command. For more information about the option, see Using the backup and restore commands in the GitHub Enterprise Server Backup Utilities documentation.

6. Review the instance's configuration

To access the staging instance using the same hostname, update your local hosts file to resolve the staging instance's hostname by IP address by editing the `/etc/hosts` file in macOS or Linux, or the `C:\Windows\system32\drivers\etc` file in Windows.

Note: Your staging instance must be accessible from the same hostname as your production instance. Changing the hostname for your GitHub Enterprise Server instance is not supported. For more information, see "Configuring the hostname for your instance."

Then, review the staging instance's configuration in the Management Console. For more information, see "Administering your instance from the web UI."

Warning: If you configured GitHub Actions or GitHub Packages for the staging instance, to avoid overwriting production data, ensure that the external storage configuration in the Management Console does not match your production instance.

7. Apply the instance's configuration

To apply the configuration from the Management Console, click **Save settings**.

Further reading

- "About upgrades to new releases"