

Release Protocol

This page should be a comprehensive guide to releasing a geodashboard project.

Step-by-step guide to complete release for mixed v2 and v3 system

1. Development - Branches and Servers

a. Geodashboard v2

i. Core

1. Geodashboard V2 is used for multiple projects

- release will be decided by larger group - contacts are [Luigi Marini](#), [Indira Gutierrez Polo](#)
- v2 should be on the most recent version of development branch on project's development server
- For creating a geodashboard (core) release. One of the committers needs to create a release branch. The release number should follow semantic versioning <http://semver.org/>. In the release branch there need to be the following 3 changes:
 - Update version number in project/Build.scala
 - Update version number in package.json
 - In Changelog.md update version number and add release date
- After the release branch is merged, a tag needs to be created in the master branch. Instructions for creating a tag are here: <https://git-scm.com/book/en/v2/Git-Basics-Tagging>
- After creating the tag, create new branch, it's name could be custom 'master-to-develop-release-2.x.x' and merge that branch to develop.

2. Project specific configuration

- Create a project specific (gltg/ilnlrs/imlczo/seagrant) release branch. Follow semantic versioning to determine the number for the release: <http://semver.org/>. In the release branch there need to be 2 changes:
 - Update version number in package.json
 - In changelog.md update version number and add release date
- After the release branch is merged, a tag needs to be created in the master branch. Instructions for creating a tag are here: <https://git-scm.com/book/en/v2/Git-Basics-Tagging> - The tag name for version 1.9.0 should be 'v1.9.0', add the letter v to indicate version before the release number for the tag.
- After creating the tag, create new branch, it's name could be custom 'master-to-develop-release-2.x.x' and merge that branch to develop.

ii. Build

- If set up with Puppet, the development server should update automatically to the most recent build
 - If you need to update manually, ssh to the geodashboard development server

```
b. sudo /home/geodashboard/update-geodashboard.sh --force
```

b. Geodashboard v3

i. Software

- Each project should have a specific branch of v3 named by project name, which the master branch merged into the configuration
 - The master branch should be merged into the project branch which maintains it's specific configuration

ii. Build (local by user)

```
1. git clone ssh://git@opensource.ncsa.illinois.edu:7999/geod/geodashboard-v3.git
cd geodashboard-v3
checkout <project specific branch name>
yarn install
yarn run build
```

2. Copy the contents of the build directory of the system's proxy server

- check the nginx root directory
- ```
more /etc/nginx/sites-enabled/<site> | grep root
```

  - this returns the path to the nginx root directory
- cd to nginx root directory
- there should be a directory call 'gd3' (maybe named something else)
- backup the directory
- ```
f. cp -r gd3 gd3_backup
```

g. copy the new build files into the gd3 directory (delete the previous of just overwrite)

c. Clowder

- clowder on dev uses the clowder development branch
- should update automatically
- to update manually

```
1. sudo /home/clowder/update-clowder.sh --force
```

d. TEST THE SITE!!!

2. Development - Data

- a. Make sure all data is up to date and correct
- b. backup the database
 - i. On the database server:

```
sudo -u postgres pg_dump <geostream db name > geostream-backup.sql
```

- ii. If not sure of the geostream database name, on the clowder server:

```
more /home/clowder/clowder/custom/custom.conf | grep postgres.db
```

- c. Make sure the cache is updated. If not sure run (this will take a while depending upon how much data there is):

```
<host>/clowder/api/geostreams/cache
```

3. Production - Branches and Servers (after thoroughly testing dev)

a. Geodashboard

- i. Core - make sure the core master branch is ready
- ii. project config
 1. Merge develop into master
- iii. All software on production branches need to be updated manually on geodashboard server

```
sudo /home/geodashboard/update-geodashboard.sh --force
```

b. Clowder

- i. On clowder server

```
sudo /home/clowder/update-clowder.sh --force
```

4. Production - Data

- a. If there is new data on dev and you decide to move the dev database to prod database and move the dev cache to the prod cache:
 - i. turn on maintenance page
 - ii. ssh into postgres machine
 - iii. backup development and production databases

```
sudo -u postgres pg_dump geostream_dev > geostream_dev_$(date +"%Y-%m-%d").  
sql # took 42 minutes to create 6.8G db  
sudo -u postgres pg_dump geostream > geostream_$(date +"%Y-%m-%d").  
sql # took 52 minutes to create 9.5G db
```

- iv. Copy dev database to production temp database

1. stop clowder on dev
 2. copy

```
sudo -u postgres psql -c "create database geostream_prod_temp with template  
geostream_dev"
```

- v. delete the production database

1. Stop clowder on production machine and:

```
sudo -u postgres dropdb geostream
```

- vi. Rename production database

1. Stop clowder on dev machine
 2. sudo -u postgres psql -c "ALTER DATABASE geostream_prod_temp RENAME TO geostream;"

- vii. Restart both clowder instances
- viii. Copy cache from dev machine to prod machine
 - 1. on production machine delete cache
 - a. option 1, use clowder endpoint
 - b. option 2, on clowder machine delete cache files

```
sudo rm /home/clowder/cache/*
```

- 2. use ssh to copy cache from dev to production (rsync would be better) (using key would be better) . On production machine:

```
sudo scp <your_ubuntu_username_on_dev>@<dev_hostname>:/home/clowder/cache/* .
```

- ix. turn off maintenance page



Related articles

- [Release Protocol V3](#)
- [Deploy Geodashboard-V3](#)
- [Geostreams-api-v3 - Local](#)
- [Troubleshooting](#)
- [Release Protocol](#)