

Grafana frontend packages

This document contains information about Grafana frontend package versioning and releases.

Versioning

We use [Lerna](#) for packages versioning and releases.

All packages are versioned according to the current Grafana version:

- Grafana v6.3.0-alpha1 -> @grafana/* packages @ 6.3.0-alpha.1
- Grafana v6.2.5 -> @grafana/* packages @ 6.2.5
- Grafana - main branch version (based on package.json, i.e. 6.4.0-pre) -> @grafana/* packages @ 6.4.0-pre- (see details below about packages publishing channels)

Please note that @grafana/toolkit, @grafana/ui, @grafana/data, and @grafana/runtime packages are considered ALPHA even though they are not released as alpha versions.

Stable releases

Even though packages are released under a stable version, they are considered ALPHA until further notice!

Stable releases are published under the `latest` tag on npm. If there was alpha/beta version released previously, the `next` tag is updated to stable version.

Alpha and beta releases

Alpha and beta releases are published under the `next` tag on npm.

Automatic prereleases

Every commit to main that has changes within the `packages` directory is a subject of npm packages release. *ALL* packages must be released under version from lerna.json file with commit SHA added to it:

```
<lerna.json version>-<COMMIT_SHA>
```

Manual release

All of the steps below must be performed on a release branch, according to Grafana Release Guide.

Make sure you are logged in to npm in your terminal and that you are a part of Grafana org on npm.

1. Run `yarn packages:prepare` script from the root directory. This performs tests on the packages and prompts for the version of the packages. The version should be the same as the one being released.
 - Make sure you use semver convention. So, place a dot between prerelease id and prerelease number, i.e. 6.3.0-alpha.1
 - Make sure you confirm the version bump when prompted!
2. Commit changes (lerna.json and package.json files) - "Packages version update: <VERSION>"
3. Run `yarn packages:build` script that prepares distribution packages in `packages/grafana-*/dist`. These directories are going to be published to npm.
4. Depending whether or not it's a prerelease:

- When releasing a prerelease run `packages:publishNext` to publish new versions.
- When releasing a stable version run `packages:publishLatest` to publish new versions.
- When releasing a test version run `packages:publishTest` to publish test versions.

5. Push version commit to the release branch.

Building individual packages

To build individual packages, run:

```
grafana-toolkit package:build --scope=<ui|toolkit|runtime|data>
```

Setting up @grafana/* packages for local development

A known issue with @grafana/* packages is that a lot of times we discover problems on canary channel(see [versioning overview](#)) when the version was already pushed to npm.

We can easily avoid that by setting up a local packages registry and test the packages before actually publishing to npm.

In this guide you will set up [Verdaccio](#) registry locally to fake npm registry. This will enable testing @grafana/* packages without the need for pushing to main.

Setting up local npm registry

From your terminal:

1. Modify `/etc/hosts` file and add the following entry: `127.0.0.1 grafana-npm.local`
2. Navigate to `devenv/local-npm` directory.
3. Run `docker-compose up` . This will start your local npm registry, available at <http://grafana-npm.local:4873/>
4. Run `npm login --registry=http://grafana-npm.local:4873 --scope=@grafana` . This will allow you to publish any @grafana/* package into the local registry.
5. Run `npm config set @grafana:registry http://grafana-npm.local:4873` . This will config your npm to install @grafana scoped packages from your local registry.

Publishing packages to local npm registry

You need to follow [manual packages release procedure](#). The only difference is you need to run `yarn packages:publishDev` task in order to publish to your local registry.

From your terminal:

1. Run `yarn packages:prepare` .
2. Commit changes in package.json and lerna.json files
3. Build packages: `yarn packages:build`
4. Run `yarn packages:publishDev` .
5. Navigate to <http://grafana-npm.local:4873> and verify that version was published

Locally published packages will be published under `dev` channel, so in your plugin package.json file you can use that channel. For example:

```
// plugin's package.json
{
```

```
...  
"@grafana/data": "dev"  
}
```