# Plugin Anatomy


- plugin.json
- module.ts


Plugins come in different shapes and sizes. Before we dive deeper, let's look at some of the properties that are shared by all of them.

Every plugin you create will require at least two files: `plugin.json` and `src/module.ts`.

## plugin.json

When Grafana starts, it scans the plugin directory for any subdirectory that contains a `plugin.json` file. The `plugin.json` file contains information about your plugin, and tells Grafana about what capabilities and dependencies your plugin needs.

While certain plugin types can have specific configuration options, let's look at the mandatory ones:

- `type` tells Grafana what type of plugin to expect. Grafana supports three types of plugins: `panel`, `datasource`, and `app`.
- `name` is what users will see in the list of plugins. If you're creating a data source, this is typically the name of the database it connects to, such as Prometheus, PostgreSQL, or Stackdriver.
- `id` uniquely identifies your plugin, and should start with your Grafana username, to avoid clashing with other plugins. Sign up for a Grafana account to claim your username.

To see all the available configuration settings for the `plugin.json`, refer to the plugin.json Schema.

## module.ts

After discovering your plugin, Grafana loads the `module.ts` file, the entrypoint for your plugin. `module.ts` exposes the implementation of your plugin, which depends on the type of plugin you're building.

Specifically, `module.ts` needs to expose an object that extends GrafanaPlugin, and can be any of the following:

- PanelPlugin
- DataSourcePlugin
- AppPlugin

---

□□□ #2
□ □□□ □□□ 2026-10-31 16:23:19 UTC
□ □□□ □□□ 2026-10-31 16:23:22 UTC