+++ title = “Upgrade Grafana” description = “Guide for upgrading Grafana” keywords = [“grafana”, “configuration”, “documentation”, “upgrade”] weight = 700 +++

Upgrade Grafana

We recommend that you upgrade Grafana often to stay up to date with the latest fixes and enhancements. In order to make this a reality, Grafana upgrades are backward compatible and the upgrade process is simple and quick.

Upgrading is generally safe (between many minor and one major version) and dashboards and graphs will look the same. There may be minor breaking changes in some edge cases, which are outlined in the Release Notes and Changelog

Backup

We recommend that you backup a few things in case you have to rollback the upgrade.

- Installed plugins - Back them up before you upgrade them in case you want to rollback the Grafana version and want to get the exact same versions you were running before the upgrade.
- Configuration files do not need to be backed up. However, you might want to in case you add new configuration options after upgrade and then rollback.

Database backup

Before upgrading it can be a good idea to backup your Grafana database. This will ensure that you can always rollback to your previous version. During startup, Grafana will automatically migrate the database schema (if there are changes or new tables). Sometimes this can cause issues if you later want to downgrade.

sqlite If you use sqlite you only need to make a backup of your `grafana.db` file. This is usually located at `/var/lib/grafana/grafana.db` on Unix systems. If you are unsure what database you use and where it is stored check you grafana configuration file. If you installed grafana to custom location using a binary tar/zip it is usually in `<grafana_install_dir>/data`.

mysql

```
backup:
> mysqldump -u root -p[root_password] [grafana] > grafana_backup.sql

restore:
> mysql -u root -p grafana < grafana_backup.sql
```

postgres

```
backup:
> pg_dump grafana > grafana_backup

restore:
> psql grafana < grafana_backup
```

Ubuntu or Debian

You can upgrade Grafana by following the same procedure as when you installed it.

Upgrade Debian package If you installed Grafana by downloading a Debian package (`.deb`), then you can execute the same `dpkg -i` command but with the new package. It will upgrade your Grafana installation.

Go to the download page for the latest download links.

```
wget <debian package url>
sudo apt-get install -y adduser
sudo dpkg -i grafana_<version>_amd64.deb
```

Upgrade from APT repository If you installed Grafana from our APT repository, then Grafana will automatically update when you run apt-get upgrade to upgrade all system packages.

```
sudo apt-get update
sudo apt-get upgrade
```

Upgrade from binary .tar file If you downloaded the binary `.tar.gz` package, then you can just download and extract the new package and overwrite all your existing files. However, this might overwrite your config changes.

We recommend that you save your custom configuration changes in a file named `<grafana_install_dir>/conf/custom.ini`. This allows you to upgrade Grafana without risking losing your configuration changes.

Centos / RHEL

If you installed Grafana by downloading an RPM package you can just follow the same installation guide and execute the same `yum install` or `rpm -i` command but with the new package. It will upgrade your Grafana installation.

If you used our YUM repository:

```
sudo yum update grafana
```

Docker

This just an example, details depend on how you configured your grafana container.

```
docker pull grafana/grafana
docker stop my-grafana-container
docker rm my-grafana-container
docker run -d --name=my-grafana-container --restart=always -v /var/lib/grafana:/var/lib/graf
```

Windows

If you downloaded the Windows binary package you can just download a newer package and extract to the same location (and overwrite the existing files). This might overwrite your configuration changes. We recommend that you save your configuration changes in a file named `<grafana_install_dir>/conf/custom.ini` as this will make upgrades easier without risking losing your configuration changes.

Update plugins

After you have upgraded, we strongly recommend that you update all your plugins as a new version of Grafana can make older plugins stop working properly.

You can update all plugins using

```
grafana-cli plugins update-all
```

Upgrading to v5.0

The dashboard grid layout engine has changed. All dashboards will be automatically upgraded to new positioning system when you load them in v5. Dashboards saved in v5 will not work in older versions of Grafana. Some external panel plugins might need to be updated to work properly.

For more details on the new panel positioning system, refer to [panel size position]({{< relref "../dashboards/json-model.md#panel-size-position" >}})

Upgrading to v5.2

One of the database migrations included in this release will update all annotation timestamps from second to millisecond precision. If you have a large amount of annotations the database migration may take a long time to complete which may cause problems if you use systemd to run Grafana.

We've got one report where using systemd, PostgreSQL and a large amount of annotations (table size 1645mb) took 8-20 minutes for the database migration to complete. However, the grafana-server process was killed after 90 seconds by

systemd. Any database migration queries in progress when systemd kills the grafana-server process continues to execute in database until finished.

If you're using systemd and have a large amount of annotations consider temporary adjusting the systemd `TimeoutStartSec` setting to something high like `30m` before upgrading.

Upgrading to v6.0

If you have text panels with script tags they will no longer work due to a new setting that per default disallow unsanitized HTML. For more information about the new setting, refer to [disable sanitize html]({{< relref "../administration/configuration/#disable-sanitize-html" >}}).

Authentication and security

If you are using Grafana's builtin, LDAP (without Auth Proxy) or OAuth authentication all users will be required to login upon the next visit after the upgrade.

If you have `cookie_secure` set to `true` in the `session` section you probably want to change the `cookie_secure` to `true` in the `security` section as well. Ending up with a configuration like this:

```
[session]
cookie_secure = true

[security]
cookie_secure = true
```

The `login_remember_days`, `cookie_username` and `cookie_remember_name` settings in the `security` section are no longer being used so they're safe to remove.

If you have `login_remember_days` configured to 0 (zero) you should change your configuration to this to accomplish similar behavior, i.e. a logged in user will maximum be logged in for 1 day until being forced to login again:

```
[auth]
login_maximum_inactive_lifetime_days = 1
login_maximum_lifetime_days = 1
```

The default cookie name for storing the auth token is `grafana_session`. you can configure this with `login_cookie_name` in `[auth]` settings.

Upgrading to v6.2

Ensure encryption of data source secrets

Data sources store passwords and basic auth passwords in secureJsonData encrypted (AES-256 in CFB mode) by default. Existing data source will keep

working with unencrypted passwords. If you want to migrate to encrypted storage for your existing data sources you can do that by:

- For data sources created through UI, you need to go to data source config, re-enter the password or basic auth password and save the data source.
- For data sources created by provisioning, you need to update your config file and use secureJsonData.password or secureJsonData.basicAuthPassword field. See [provisioning docs]({{< relref "../administration/provisioning" >}}) for example of current configuration.

Embedding Grafana

If you're embedding Grafana in a `<frame>`, `<iframe>`, `<embed>` or `<object>` on a different website it will no longer work due to a new setting that per default instructs the browser to not allow Grafana to be embedded. For more information about embedding Grafana, refer to [configuration embedding]({{< relref "../administration/configuration/#allow-embedding" >}}) about this new setting.

Session storage is no longer used

In 6.2 we completely removed the backend session storage since we replaced the previous login session implementation with an auth token. If you are using Auth proxy with LDAP, a shared cached is used in Grafana, so you might want to configure [remote_cache] instead. If not, Grafana will fall back to using the database as a shared cache.

Upgrading Elasticsearch to v7.0+

The semantics of `max concurrent shard requests` changed in Elasticsearch v7.0, see release notes for reference.

If you upgrade Elasticsearch to v7.0+ you should make sure to update the data source configuration in Grafana so that version is `7.0+` and `max concurrent shard requests` properly configured. 256 was the default in pre v7.0 versions. In v7.0 and above 5 is the default.

Upgrading to v6.4

Annotations database migration

One of the database migrations included in this release will merge multiple rows used to represent an annotation range into a single row. If you have a large number of region annotations the database migration may take a long time to complete. See Upgrading to v5.2 for tips on how to manage this process.

Docker

Grafana's docker image is now based on Alpine instead of Ubuntu.

Plugins that need updating

- Splunk

Upgrading to v6.5

Pre Grafana 6.5.0, the CloudWatch datasource used the GetMetricStatistics API for all queries that did not have an ´id´ and did not have an ´expression´ defined in the query editor. The GetMetricStatistics API has a limit of 400 transactions per second (TPS). In this release, all queries use the GetMetricData API which has a limit of 50 TPS and 100 metrics per transaction. We expect this transition to be smooth for most of our users, but in case you do face throttling issues we suggest you increase the TPS quota. To do that, please visit the AWS Service Quotas console. For more details around CloudWatch API limits, see CloudWatch docs.

Each request to the GetMetricData API can include 100 queries. This means that each panel in Grafana will only issue one GetMetricData request, regardless of the number of query rows that are present in the panel. Consequently as it is no longer possible to set `HighRes` on a per query level anymore, this switch is now removed from the query editor. High resolution can still be achieved by choosing a smaller minimum period in the query editor.

The handling of multi-valued template variables in dimension values has been changed in Grafana 6.5. When a multi template variable is being used, Grafana will generate a search expression. In the GetMetricData API, expressions are limited to 1024 characters, so it might be the case that this limit is reached when a multi-valued template variable that has a lot of values is being used. If this is the case, we suggest you start using `*` wildcard as dimension value instead of a multi-valued template variable.

Upgrading to v6.6

The Generic OAuth setting `send_client_credentials_via_post`, used for supporting non-compliant providers, has been removed. From now on, Grafana will automatically detect if credentials should be sent as part of the URL or request body for a specific provider. The result will be remembered and used for additional OAuth requests for that provider.

Important changes regarding SameSite cookie attribute

Chrome 80 treats cookies as `SameSite=Lax` by default if no `SameSite` attribute is specified, see https://www.chromestatus.com/feature/5088147346030592.

Due to this change in Chrome, the `[security]` setting `cookie_samesite` configured to `none` now renders cookies with `SameSite=None` attribute compared to before where no `SameSite` attribute was added to cookies. To get the old

behavior, use value `disabled` instead of `none`, see [cookie_samesite in Configuration]({{< relref "../administration/configuration/#cookie-samesite" >}}) for more information.

> **Note:** There is currently a bug affecting Mac OSX and iOS that causes `SameSite=None` cookies to be treated as `SameSite=Strict` and therefore not sent with cross-site requests, see https://bugs.webkit.org/show_bug.cgi?id=198181 for details. Until this is fixed, `SameSite=None` might not work properly on Safari.

This version of Chrome also rejects insecure `SameSite=None` cookies. See https://www.chromestatus.com/feature/5633521622188032 for more information. Make sure that you change the `[security]` setting `cookie_secure` to `true` and use HTTPS when `cookie_samesite` is configured to `none`, otherwise authentication in Grafana won't work properly.

Upgrading to v7.0

PhantomJS removed

PhantomJS was deprecated in [Grafana v6.4]({{< relref "../whatsnew/whats-new-in-v6-4.md#phantomjs-deprecation" >}}) and starting from Grafana v7.0.0, all PhantomJS support has been removed. This means that Grafana no longer ships with a built-in image renderer, and we advise you to install the Grafana Image Renderer plugin.

Dashboard minimum refresh interval enforced

A global minimum dashboard refresh interval is now enforced and defaults to 5 seconds. For more information about this setting, refer to [minimum refresh interval]({{< relref "../administration/configuration/#min-refresh-interval" >}}).

Backend plugins

Grafana now requires backend plugins to be signed. If a backend plugin is not signed Grafana will not load/start it. This is an additional security measure to make sure backend plugin binaries and files haven't been tampered with. All Grafana Labs authored backend plugins, including Enterprise plugins, are now signed. It's possible to allow unsigned plugins using a configuration setting, but is something we strongly advise against doing. For more information about this setting, refer to [allow loading unsigned plugins]({{< relref "../administration/#allow_loading_unsigned_plugins" >}}).

Cookie path

Starting from Grafana v7.0.0, the cookie path does not include the trailing slash if Grafana is served from a subpath in order to align with RFC 6265. However, stale session cookies (set before the upgrade) can result in unsuccessful logins because they can not be deleted during the standard login phase due to the changed cookie path. Therefore users experiencing login problems are advised to manually delete old session cookies, or administrators can fix this for all users by changing the [`login_cookie_name`]({{< relref "../administration/#login-cookie-name" >}}), so the old cookie would get ignored.

Upgrading to v7.2

Ensure encryption of existing alert notification channel secrets

Before Grafana v7.2 alert notification channels did not store sensitive settings/secrets such as API tokens and password encrypted in the database. In Grafana v7.2, creating a new alert notification channel will store sensitive settings encrypted in the database.

The following alert notifiers have been updated to support storing their sensitive settings encrypted:

- Slack (URL and Token)
- Pagerduty (Integration Key)
- Webhook (Password)
- Prometheus Alertmanager (Basic Auth Password)
- Opsgenie (API Key)
- Sensu (Password)
- Telegram (BOT API Token)
- LINE (token)
- Pushover (API Token, User key)
- Threema Gateway (API Secret)

For existing alert notification channels, there is no automatic migration of storing sensitive settings encrypted, and they will continue to work as before. Migration must be done manually. Opening a configured alert notification channel in the UI and saving it will store sensitive settings encrypted and at the same time reset the historic unencrypted setting of that alert notification channel in the database.

> Please note that when migrating a notification channel and later downgrading Grafana to an earlier version, the notification channel will not be able to read stored sensitive settings and, as a result, not function as expected.

For provisioning of alert notification channels, refer to [Alert notification channels]({{< relref "../administration/provisioning.md#alert-notification-channels" >}}).

Upgrading to v7.3

AWS CloudWatch data source

The AWS CloudWatch data source's authentication scheme has changed in Grafana 7.3. Most importantly the authentication method *ARN* has been removed, and a new one has been added: *AWS SDK Default*. Existing data source configurations using the former will fallback to the latter. Assuming an IAM role will still work though, and the old *ARN* method would use the default AWS SDK authentication method under the hood anyway.

Since *ARN* has been removed as an authentication method, we have instead made it into an option for providing the ARN of an IAM role to assume. This works independently of the authentication method you choose.

The new authentication method, *AWS SDK Default*, uses the default AWS Go SDK credential chain, which at the time of writing looks for credentials in the following order:

1. Environment variables.
2. Shared credentials file.
3. If your application uses an ECS task definition or RunTask API operation, IAM role for tasks.
4. If your application is running on an Amazon EC2 instance, IAM role for Amazon EC2.

The other authentication methods, *Access and secret key* and *Credentials file*, have changed in regards to fallbacks. If these methods fail, they no longer fallback to other methods. e.g. environment variables. If you want fallbacks, you should use *AWS SDK Default* instead.

For more information and details, please refer to [Using AWS CloudWatch in Grafana]({{< relref "../datasources/aws-cloudwatch/aws-authentication.md" >}}).

User invites database migration

The database table *temp_user*, that tracks user invites, is subject to a database migration that changes the data type of the *created* and *updated* columns:

Database	Old data type	New data type
Sqlite	DATETIME	INTEGER
MySQL	DATETIME	INT
Postgres	TIMESTAMP	INTEGER

> Please note that if downgrading Grafana to an earlier version, you have to manually change the data type of the *created* and *updated* columns back to *old data type* , otherwise the user invite feature doesn't function as expected.

Snapshots database migration

The database table *dashboard_snapshot*, that stores dashboard snapshots, adds a new column *dashboard_encrypted* for storing an encrypted snapshot. NOTE: Only snapshots created on Grafana 7.3 or later will use this column to store snapshot data as encrypted. Snapshots created before this version will be unaffected and remain unencrypted.

Use of the root group in the Docker images

The Grafana Docker images use the `root` group instead of the `grafana` group. This change can cause builds to break for users who extend the Grafana Docker image. Learn more about this change in the [Docker migration instructions]({{< relref "docker/#migrate-to-v73-or-later">}})

Upgrading to v7.5

VictorOps Alert Notifier

The VictorOps alert notifier now accepts a `severity` tag, in a similar vein to the PagerDuty alert notifier. The possible values are outlined in the VictorOps docs.

For example, if you want an alert to be `INFO`-level in VictorOps, create a tag `severity=info` (case-insensitive) in your alert.

Upgrading to v8.0

Plugins

Grafana now requires all plugins to be signed. If a plugin is not signed Grafana will not load/start it. This is an additional security measure to make sure plugin files and binaries haven't been tampered with. All Grafana Labs authored plugins, including Enterprise plugins, are now signed. It's possible to allow unsigned plugins using a configuration setting, but is something we strongly advise against doing. For more information about this setting, refer to [allow loading unsigned plugins]({{< relref "../administration/#allow_loading_unsigned_plugins" >}}).

Grafana Live

Grafana now maintains persistent WebSocket connections for real-time messaging needs.

When WebSocket connection is established, Grafana checks the request Origin header due to security reasons (for example, to prevent hijacking of WebSocket connection). If you have a properly defined public URL (`root_url` server option) then the origin check should successfully pass for WebSocket requests originating from public URL pages. In case of an unsuccessful origin check, Grafana returns

a 403 error. It's also possible to add a list of additional origin patterns for the origin check.

To handle many concurrent WebSocket connections you may need to tune your OS settings or infrastructure. Grafana Live is enabled by default and supports 100 concurrent WebSocket connections max to avoid possible problems with the file descriptor OS limit. As soon as your setup meets the requirements to scale the number of persistent connections this limit can be increased. You also have an option to disable Grafana Live.

Refer to [Grafana Live configuration]({{< relref "../live/configure-grafana-live.md" >}}) documentation for more information.

Postgres, MySQL, Microsoft SQL Server data sources

Grafana v8.0 changes the underlying data structure to [data frames]({{< relref "../developers/plugins/data-frames.md" >}}) for the Postgres, MySQL, Microsoft SQL Server data sources. As a result, a *Time series* query result gets returned in a [wide format]({{< relref "../developers/plugins/data-frames.md#wide-format" >}}). To make the visualizations work as they did before, you might have to do some manual migrations.

For any existing panels/visualizations using a *Time series* query, where the time column is only needed for filtering the time range, for example, using the bar gauge or pie chart panel, we recommend that you use a *Table query* instead and exclude the time column as a field in the response. Refer to this issue comment for detailed instructions and workarounds.

Prefix added to series names When you have a query where there's a time value and a numeric value selected together with a string value that's not named *metric*, the graph panel renders series names as `value <hostname>` rather than just `<hostname>` which was the case before Grafana 8.

```
SELECT
  $__timeGroup("createdAt",'10m'),
  avg(value) as "value",
  hostname
FROM grafana_metric
WHERE $__timeFilter("createdAt")
GROUP BY time, hostname
ORDER BY time
```

There are two possible workarounds to resolve this problem:

1. In Grafana v8.0.3, use an alias of the string column selected as `metric`. for example, `hostname as metric`.
2. Use the [Standard field options/Display name]({{< relref "../panels/reference-standard-field-definitions.md#display-name" >}}) to format the alias. For

the preceding example query, you would use `${__field.labels.hostname}` option.

For more information, refer to the our relational databases documentation of Postgres({{< relref "../datasources/postgres.md#time-series-queries" >}}), MySQL({{< relref "../datasources/mysql.md#time-series-queries" >}}), [Microsoft SQL Server]({{< relref "../datasources/mssql.md#time-series-queries" >}}).

Upgrading to v8.1

Use of unencrypted passwords for data sources no longer supported

As of Grafana v8.1, we no longer support unencrypted storage of passwords and basic auth passwords.

> **Note:** Since Grafana v6.2, new or updated data sources store passwords and basic auth passwords encrypted. See [upgrade note]({{< relref "#ensure-encryption-of-data-source-secrets" >}}) for more information. However, unencrypted passwords and basic auth passwords were also allowed.

To migrate to encrypted storage, follow the instructions from the [v6.2 upgrade notes]({{< relref "#ensure-encryption-of-data-source-secrets" >}}). You can also use a `grafana-cli` command to migrate all of your data sources to use encrypted storage of secrets. See [migrate data and encrypt passwords]({{< relref "../administration/cli.md#migrate-data-and-encrypt-passwords" >}}) for further instructions.

Upgrading to 8.3

In 8.3, Grafana dashboards now reference data sources using an object with `uid` and `type` properties instead of the data source name property. A schema migration is applied when existing dashboards open. If you provision dashboards to multiple Grafana instances, then we recommend that you also provision data sources. You can specify the `uid` to be the same for data sources across your instances. If you need to find the `uid` for a data source created in the UI, check the URL of the data source settings page. The URL follows the pattern `/datasource/edit/${uid}`, meaning the last part is the `uid`.

Upgrading to 8.5

The concept of a `default` data source existed in Grafana since the beginning. However, the meaning and behavior were not clear. The default data source was not just the starting data source for new panels but it was also saved using a special value (null). This made it possible to change the default data source to another and have that change impact all dashboards that used the default data source.

This behavior was not very intuitive and creates issues for users who want to change the default without it impacting existing dashboards. That is why we are changing the behavior in 8.5. From now on, the `default` data source will not be a persisted property but just the starting data source for new panels and queries. Existing dashboards that still have panels with a `datasource` set to null will be migrated when the dashboard opens. The migration will set the data source property to the **current** default data source.