+++ title = ” High availability” description = “High Availability” keywords = [“grafana”, “alerting”, “tutorials”, “ha”, “high availability”] weight = 450 +++

High availability

The Grafana alerting system has two main components: a `Scheduler` and an internal `Alertmanager`. The `Scheduler` is responsible for the evaluation of your [alert rules]({{< relref “./fundamentals/evaluate-grafana-alerts.md” >}}) while the internal Alertmanager takes care of the **routing** and **grouping**.

When it comes to running Grafana alerting in high availability the operational mode of the scheduler is unaffected such that all alerts continue be evaluated in each Grafana instance. Rather the operational change happens in the Alertmanager which **deduplicates** alert notifications across Grafana instances.

{{< figure src=“/static/img/docs/alerting/unified/high-availability-ua.png” class=“docs-image–no-shadow” max-width= “750px” caption=“High availability” >}}

The coordination between Grafana instances happens via a Gossip protocol. Alerts are not gossiped between instances. It is expected that each scheduler delivers the same alerts to each Alertmanager.

The two types of messages that are gossiped between instances are:

- Notification logs: Who (which instance) notified what (which alert)
- Silences: If an alert should fire or not

These two states are persisted in the database periodically and when Grafana is gracefully shutdown.

Enable high availability

To enable high availability support you need to add at least 1 Grafana instance to the `[ha_peer]` configuration option within the `[unified_alerting]` section:

1. In your custom configuration file ($WORKING_DIR/conf/custom.ini), go to the `[unified_alerting]` section.
2. Set `[ha_peers]` to the number of hosts for each grafana instance in the cluster (using a format of host:port) e.g. `ha_peers=10.0.0.5:9094,10.0.0.6:9094,10.0.0.7:9094`
3. Gossiping of notifications and silences uses both TCP and UDP port 9094. Each Grafana instance will need to be able to accept incoming connections on these ports.
4. Set `[ha_listen_address]` to the instance IP address using a format of host:port (or the Pod’s IP in the case of using Kubernetes) by default it is set to listen to all interfaces (`0.0.0.0`).

Kubernetes

If you are using Kubernetes, you can expose the pod IP through an environment variable via the container definition such as:

```
env:
- name: POD_IP
  valueFrom:
    fieldRef:
      fieldPath: status.podIP
```