

1 Grafana + Prometheus Deployment

1.1 Prometheus

1.1.1 Installation with Ansible

We have to run the next Ansible playbook to deploy Prometheus:

```
$ make prometheus
```

1.1.2 Configure Prometheus

Once we have Prometheus installed, we can configure it in the Prometheus configuration file; we can check a configuration example in the “**extras**” directory of the IK4-Data-Platform.

```
$ sudo vi /home/user/Prometheus/prometheus/prometheus.yml
```

1.1.3 Start/Stop/Restart Prometheus manually

To start, stop or restart Prometheus manually we can use the next service command:

```
sudo systemctl start prometheus
sudo systemctl stop prometheus
sudo systemctl restart prometheus
```

1.1.4 Prometheus UI

Prometheus is accessible by the next URL:

```
IP-address:9090
```

1.2 Grafana

1.2.1 Installation with Ansible

We have to run the next Ansible playbook to deploy Grafana:

```
$ make grafana
```

1.2.2 Grafana configuration

It is not necessary to change the Grafana configuration file, but if you need to configure something, this is the configuration file path:

```
$ /etc/grafana/grafana.ini
```

1.2.3 Grafana UI

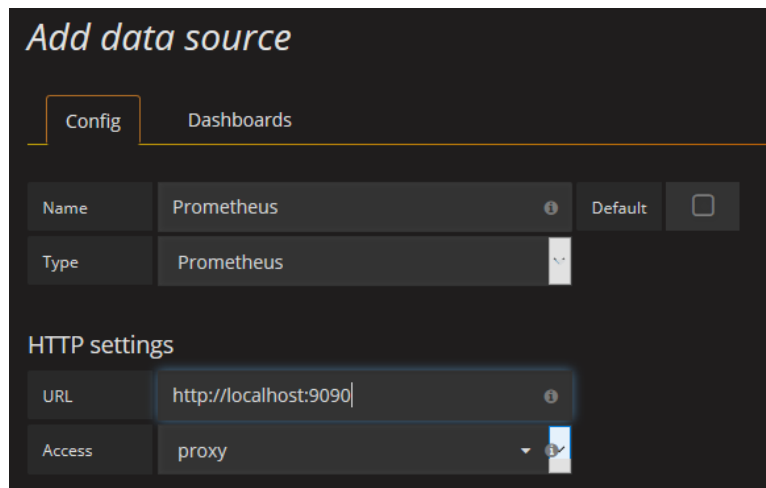
Grafana is accesible by the next URL:

```
IP-address:3000
```

1.2.4 Grafana add Datasource

We have to configure a DataSource for Grafana, where he is going to take the data. To do that we have to go to the “Datasources” tab of the Grafana UI, and we have to add a new DataSource:

*The URL is the URL where Prometheus is serving.



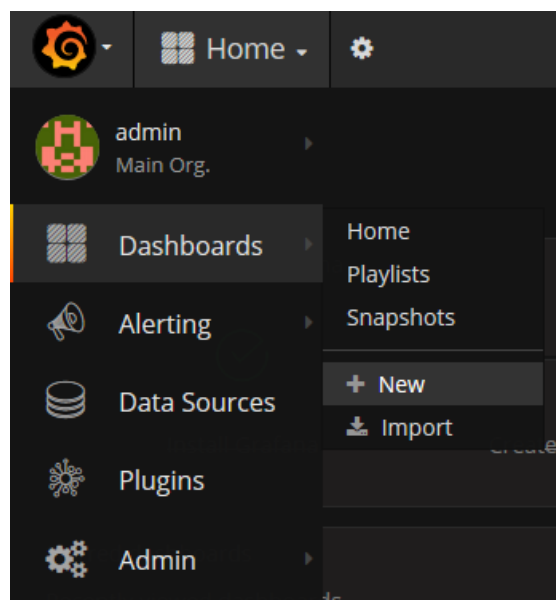
The screenshot shows the 'Add data source' interface in Grafana. It has two tabs: 'Config' (selected) and 'Dashboards'. Under the 'Config' tab, there are two rows: 'Name' with the value 'Prometheus' and a 'Default' checkbox, and 'Type' with a dropdown menu showing 'Prometheus'. Below these is the 'HTTP settings' section, which includes a 'URL' field with the value 'http://localhost:9090' and an 'Access' dropdown menu with the value 'proxy'.

1.2.5 Add Dashboards

We can create a Dashboard from scratch, so we have to select “new” as we can see in the illustration.

In case we want to import a existing Dashboard from the public Grafana Dashboard repository, we need to copy his ID. We can copy those IDs from the next URL: <https://grafana.com/dashboards/>, then we have to select “import” and we have to paste the ID of the Dashboard we want to import.

To import a Dashboard from our computer, we can select “import” and we can upload the JSON from our computer.

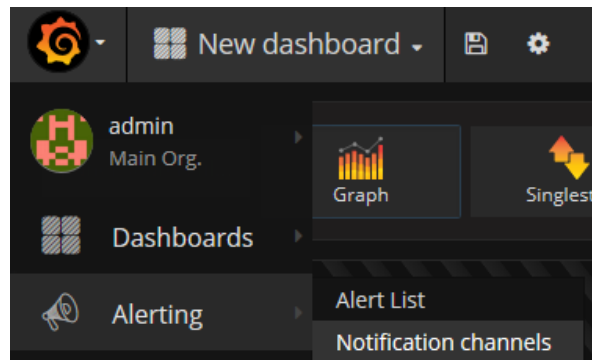


1.2.6 Slack Alerts

Prerequisites:

- Create a bot for Grafana in Slack.

On Grafana we have to select “notification channels” as we can see in the next illustration:



Once we are inside the notification channels, we have to select “new channel” and configure the channel like in the next illustration, changing the webhook URL.

*Recipient: Is the user or channel of Slack that is going to receive the notifications.

A screenshot of the 'New Notification Channel' configuration form in Grafana. The form has a dark theme. Fields include: 'Name' (Slack), 'Type' (Slack), 'Send on all alerts' (checked), and 'Include image' (checked). Below these is a section titled 'Slack settings' containing 'Url' (https://hooks.slack.com/services/T025EP2NP ...) and 'Recipient' (#Grafana).

1.3 Node_exporter

Node_Exporter is inside the Prometheus playbook, so if you want to deploy it, you will Access to Prometheus playbook, and configure the “inventory/hosts.ini” file with the hosts on which you want to deploy node_exporter.

1.3.1 Node_exporter service

```
$ sudo systemctl start node_exporter
$ sudo systemctl stop node_exporter
$ sudo systemctl restart node_exporter
```

1.4 JMX_exporter

JMX_Exporter is inside the Prometheus playbook, so if you want to deploy it, you will Access to Prometheus playbook, and configure the “inventory/hosts.ini” file with the hosts on which you want to deploy JMX_exporter.

1.4.1 JMX_exporter service

Before running the service, we need to compile the docker image:

```
$ sudo docker build -f /home/ikerlan/Dockerfile -t jmx_exporter .  
$ sudo systemctl start node_exporter  
$ sudo systemctl stop node_exporter  
$ sudo systemctl restart node_exporter
```