

---

## Cluster Monitoring stack for ARM / X86-64 platforms

The Prometheus Operator for Kubernetes provides easy monitoring definitions for Kubernetes services and deployment and management of Prometheus instances.

This has been tested on a hybrid ARM64 / X84-64 Kubernetes cluster deployed as this article.

This repository collects Kubernetes manifests, Grafana dashboards, and Prometheus rules combined with documentation and scripts to provide easy to operate end-to-end Kubernetes cluster monitoring with Prometheus using the Prometheus Operator. The container images support AMD64, ARM64, ARM and PPC64le architectures.

The content of this project is written in jsonnet and is an extension of the fantastic kube-prometheus project.

If you like this project and others I've been contributing and would like to support me, please check-out my Patreon page!

Components included in this package:

- The Prometheus Operator
- Highly available Prometheus
- Highly available Alertmanager
- Prometheus node-exporter
- kube-state-metrics
- CoreDNS
- Grafana
- SMTP relay to Gmail for Grafana notifications (optional)

There are additional modules (disabled by default) to monitor other components of the infrastructure. These can be enabled or disabled on `vars.jsonnet` file by setting the module `enabled` flag in `modules` to `true` or `false`.

The additional modules are:

- ARM-exporter to generate temperature metrics (works on some ARM boards like RaspberryPi)
- MetalLB metrics
- Traefik metrics
- Elasticsearch metrics
- APC UPS metrics
- Gmail SMTP relay module

There are also options to set the ingress domain suffix and enable persistence for Grafana and Prometheus.

---

The ingresses can use TLS with the default self-signed certificate from your Ingress controller by setting `TLSIngress` to `true` and use a custom certificate by creating the files `server.crt` and `server.key` and enabling the `UseProvidedCerts` parameter at `vars.jsonnet`.

Persistence for Prometheus and Grafana can be enabled in the `enablePersistence` section. Setting each to `true`, creates the volume PVCs. If no PV names are defined in `prometheusPV` and `grafanaPV`, the default StorageClass will be used to dynamically create the PVs. The sizes can be adjusted in `prometheusSizePV` and `grafanaSizePV`.

If using pre-created persistent volumes (samples in `samples`), check permissions on the directories hosting the files. The `UID:GID` for Prometheus is `1000:0` and for Grafana is `472:472`.

Changing these parameters require a rebuild of the manifests with `make` followed by `make deploy`. To avoid installing all pre-requisites like Golang, Jsonnet, Jsonnet-bundler, use the target `make docker` to build in a container.

## Quickstart (non K3s)

The repository already provides a set of compiled manifests to be applied into the cluster or the deployment can be customized thru the jsonnet files.

If you only need the default features and adjust your cluster URL for the ingress, there is no need to rebuild the manifests (and install all tools). Use the `change_suffix` target with argument `suffix=[suffixURL]` with the URL of your cluster ingress controller. If you have a local cluster, use the nip.io domain resolver passing `your_cluster_ip.nip.io` to the `suffix` argument. After this, just run `make deploy`.

```
1 # Update the ingress URLs
2 make change_suffix suffix=[suffixURL]
3
4 # Deploy
5 make deploy
```

To customize the manifests, edit `vars.jsonnet` and rebuild the manifests.

```
1 $ make vendor
2 $ make
3 $ make deploy
4
5 # Or manually:
6
7 $ make vendor
8 $ make
9 $ kubectl apply -f manifests/setup/
10 $ kubectl apply -f manifests/
```

---

If you get an error from applying the manifests, run the `make deploy` or `kubectl apply -f manifests/` again. Sometimes the resources required to apply the CRDs are not deployed yet.

If you enable the SMTP relay for Gmail in `vars.jsonnet`, the pod will be in an error state after deployed since it would not find the user and password on the “smtp-account” secret. To generate, run the `scripts/create_gmail_auth.sh` script.

## Quickstart on Minikube

You can also test and develop the monitoring stack on Minikube. First install minikube by following the instructions here for your platform. Then, follow the instructions similar to the non-K3s deployment:

```
1 # Start minikube (if not started)
2 minikube start
3
4 # Enable minikube ingress to allow access to the web interfaces
5 minikube addons enable ingress
6
7 # Get the minikube instance IP
8 minikube ip
9
10 # Run the change_suffix target
11 make change_suffix suffix=[minikubeIP.nip.io]
12
13 # or customize additional params on vars.jsonnet and rebuild
14 make vendor
15 make
16
17 # and deploy the manifests
18 make deploy
19
20 # Get the URLs for the exposed applications and open in your browser
21 kubectl get ingress -n monitoring
```

## Quickstart for K3s

To deploy the monitoring stack on your K3s cluster, there are four parameters that need to be configured in the `vars.jsonnet` file:

1. Set `k3s.enabled` to **true**.
2. Change your K3s master node IP(your VM or host IP) on `k3s.master_ip` parameter.
3. Edit `suffixDomain` to have your node IP with the `.nip.io` suffix or your cluster URL. This will be your ingress URL suffix.

- 
4. Set `traefikExporter enabled` parameter to `true` to collect Traefik metrics and deploy dashboard.

After changing these values to deploy the stack, run:

```
1 $ make vendor
2 $ make
3 $ make deploy
4
5 # Or manually:
6
7 $ make vendor
8 $ make
9 $ kubectl apply -f manifests/setup/
10 $ kubectl apply -f manifests/
```

If you get an error from applying the manifests, run the `make deploy` or `kubectl apply -f manifests/` again. Sometimes the resources required to apply the CRDs are not deployed yet.

If you enable the SMTP relay for Gmail in `vars.jsonnet`, the pod will be in an error state after deployed since it would not find the user and password on the “smtp-account” secret. To generate, run the `scripts/create_gmail_auth.sh` script.

## Ingress

Now you can open the applications:

To list the created ingresses, run `kubectl get ingress --all-namespaces`, if you added your cluster IP or URL suffix in `vars.jsonnet` before rebuilding the manifests, the applications will be exposed on:

- Grafana on `https://grafana.%5Byour_node_ip%5D.nip.io`,
- Prometheus on `https://prometheus.%5Byour_node_ip%5D.nip.io`
- Alertmanager on `https://alertmanager.%5Byour_node_ip%5D.nip.io`

## Updating the ingress suffixes

To avoid rebuilding all manifests, there is a make target to update the Ingress URL suffix to a different suffix. Run `make change_suffix suffix="[clusterURL]"` to change the ingress route IP for Grafana, Prometheus and Alertmanager and reapply the manifests.

---

## Customizing

The content of this project consists of a set of jsonnet files making up a library to be consumed.

### Pre-reqs

The project requires json-bundler and the jsonnet compiler. The Makefile does the heavy-lifting of installing them. You need Go (version 1.18 minimum) already installed:

```
1 git clone https://github.com/carlosedp/cluster-monitoring
2 cd cluster-monitoring
3 make vendor
4 # Change the jsonnet files...
5 make
```

After this, a new customized set of manifests is built into the `manifests` dir. To apply to your cluster, run:

```
1 make deploy
```

To uninstall, run:

```
1 make teardown
```

## Images

This project depends on the following images (all supports ARM, ARM64 and AMD64 thru manifests):

### Alertmanager Blackbox\_exporter Node\_exporter Snmp\_exporter Prometheus

- Source: <https://github.com/carlosedp/prometheus-ARM>
- Autobuild: <https://travis-ci.org/carlosedp/prometheus-ARM>
- Images:
  - <https://hub.docker.com/r/carlosedp/prometheus/>
  - <https://hub.docker.com/r/carlosedp/alertmanager/>
  - [https://hub.docker.com/r/carlosedp/blackbox\\_exporter/](https://hub.docker.com/r/carlosedp/blackbox_exporter/)
  - [https://hub.docker.com/r/carlosedp/node\\_exporter/](https://hub.docker.com/r/carlosedp/node_exporter/)
  - [https://hub.docker.com/r/carlosedp/snmp\\_exporter/](https://hub.docker.com/r/carlosedp/snmp_exporter/)

### ARM\_exporter

- Source: [https://github.com/carlosedp/docker-arm\\_exporter](https://github.com/carlosedp/docker-arm_exporter)

- 
- Autobuild: [https://travis-ci.org/carlosedp/docker-arm\\_exporter](https://travis-ci.org/carlosedp/docker-arm_exporter)
  - Images: [https://hub.docker.com/r/carlosedp/arm\\_exporter/](https://hub.docker.com/r/carlosedp/arm_exporter/)

### **Prometheus-operator**

- Source: <https://github.com/carlosedp/prometheus-operator>
- Autobuild: No autobuild yet. Use provided [build\\_images.sh](#) script.
- Images: <https://hub.docker.com/r/carlosedp/prometheus-operator>

### **Prometheus-adapter**

- Source: <https://github.com/DirectXMan12/k8s-prometheus-adapter>
- Autobuild: No autobuild yet. Use provided [build\\_images.sh](#) script.
- Images: <https://hub.docker.com/r/carlosedp/k8s-prometheus-adapter>

### **Grafana**

- Source: <https://github.com/carlosedp/grafana-ARM>
- Autobuild: <https://travis-ci.org/carlosedp/grafana-ARM>
- Images: <https://hub.docker.com/r/grafana/grafana/>

### **Kube-state-metrics**

- Source: <https://github.com/kubernetes/kube-state-metrics>
- Autobuild: No autobuild yet. Use provided [build\\_images.sh](#) script.
- Images: <https://hub.docker.com/r/carlosedp/kube-state-metrics>

### **Addon-resizer**

- Source: <https://github.com/kubernetes/autoscaler/tree/master/addon-resizer>
- Autobuild: No autobuild yet. Use provided [build\\_images.sh](#) script.
- Images: <https://hub.docker.com/r/carlosedp/addon-resizer>

*Obs.* This image is a clone of AMD64, ARM64 and ARM with a manifest. It's cloned and generated by the [build\\_images.sh](#) script

### **configmap\_reload**

- Source: <https://github.com/carlosedp/configmap-reload>
- Autobuild: <https://travis-ci.org/carlosedp/configmap-reload>
- Images: <https://hub.docker.com/r/carlosedp/configmap-reload>

### **prometheus-config-reloader**

- Source: <https://github.com/coreos/prometheus-operator/tree/master/contrib/prometheus-config-reloader>

- 
- Autobuild: No autobuild yet. Use provided `build_images.sh` script.
  - Images: <https://hub.docker.com/r/carlosedp/prometheus-config-reloader>

### **SMTP-server**

- Source: <https://github.com/carlosedp/docker-smtp>
- Autobuild: <https://travis-ci.org/carlosedp/docker-smtp>
- Images: <https://hub.docker.com/r/carlosedp/docker-smtp>

### **Kube-rbac-proxy**

- Source: <https://github.com/brancz/kube-rbac-proxy>
- Autobuild: No autobuild yet. Use provided `build_images.sh` script.
- Images: <https://hub.docker.com/r/carlosedp/kube-rbac-proxy>